

Zabbix 3.4 手册

书栈(BookStack.CN)

目 录

致谢

Zabbix 手册

概述

1. 介绍

 1 手册结构

 2 Zabbix是什么

 3 Zabbix功能点

 4 Zabbix概述

 5 Zabbix 3.4.0的新特征

2. 定义

3. Zabbix 进程

 1 服务器

 2 Agent

 3 Proxy

 4 Java gateway

 5 Sender

 6 Get

4. 安装

 1 部署Zabbix

 2 安装要求

 3 从部署包安装

 4 从源代码安装

 5 从容器安装

 6 升级步骤

 7 已知问题

 8 模版变更

 9 3.4.0 升级日志

5. 快速开始

 1 登陆和配置用户

 2 新建主机

 3 新建监控项

 4 新建触发器

 5 获取问题通知

 6 新建模版

6. Zabbix设备

7. 配置

1 主机和主机组

[1 配置一台主机](#)

[2 资产管理](#)

[3 批量更新](#)

2 监控项

[1 创建一个监控项](#)

[1 监控项的Key](#)

[2 自定义间隔](#)

[2 监控项类型](#)

[1 Zabbix 代理](#)

只用于Windows的监控项Key

[2 SNMP代理](#)

[1 动态索引](#)

[2 特殊OID](#)

[3 SNMP trap](#)

[4 IPMI检查](#)

[5 简单检查](#)

[1 VMware监控项Key](#)

[6 日志文件监控](#)

[7 计算监控项](#)

[8 内部检查](#)

[9 SSH检查](#)

[10 Telnet检查](#)

[11 外部检查](#)

[12 汇总检查](#)

[13 捕捉器监控项](#)

[14 JMX监控](#)

[15 ODBC监控](#)

[3 历史与趋势](#)

[4 用户自定义参数](#)

[1 扩展Zabbix代理](#)

[5 可加载模块](#)

[6 Windows性能计数器](#)

[7 批量更新](#)

[8 值映射](#)

[9 应用](#)

[10 队列](#)

[11 值缓存](#)

3 触发器

- 1 配置一个触发器
 - 2 触发器表达式
 - 3 触发器依赖
 - 4 触发器严重性
 - 5 自定义触发器的严重性
 - 6 单位符号
 - 7 批量更新
 - 8 预测触发功能
 - 9 事件标签
- ### 4 事件
- 1 触发器事件生成
 - 2 手动关闭问题事件
 - 3 其他事件来源
- ### 5 事件关联
- ### 6 可视化
- 1 图形
 - 1 简单图形
 - 2 自定义图表
 - 3 特设图形
 - 2 拓扑图
 - 1 配置拓扑图
 - 2 链接指示器
 - 3 聚合图形
 - 1 聚合图形元素
 - 4 幻灯片演示
- ### 7 模板
- 1 配置模板
 - 2 链接/取消链接
 - 3 嵌套
- ### 8 事件通知
- 1 Media类型
 - 1 E-mail
 - 2 SMS
 - 3 Jabber
 - 4 Ez Texting
 - 2 Actions

1 条件

2 操作

1 发送消息

2 远程命令

3 附加操作

4 在信息中使用宏

3 恢复操作

4 Escalations

3 接收不受支持的项目的通知

9 宏

1 宏函数

2 用户宏

3 自动发现 (LLD) 宏

10 用户和用户组

1 配置用户

2 权限

3 用户组

8. 服务监控

9. Web 监控

1 Web 监控项

2 场景示例

10. 虚拟机监控

虚拟机 discovery 关键字段

11. 维护

12. 正则表达式

13. 事件确认

14. 配置导出V导入

1 主机组

2 模板

3 主机

4 网络拓扑图

5 聚合图形

15. 发现

1 网络发现

配置网络发现规则

2 自动注册

3 自动发现 (LLD)

关于自动发现(LLD)的注意事项

16. 分布式监控

1 代理

17. 加密

1 使用证书

2 使用共享密钥

3 故障排除

1 连接类型或权限问题

2 证书问题

3 PSK问题

18. Web界面

1 前端部分

1 监控中

1 仪表板

2 问题

3 概述

4 Web监测

5 最新数据

6 触发器

7 图形

8 聚合图形

9 拓扑图

10 自动发现

11 IT服务

2 库存

1 概述

2 主机

3 报告

1 Zabbix的状态

2 可用性报告|

3 触发器前100

4 审计

5 动作日志

6 通知

4 配置

1 主机组

2 模板

3 主机

1 应用

2 Items项

3 触发器

4 图

5 发现规则

6 Web场景

4 维护

5 动作

6 事件关联

7 发现

8 IT服务

5 管理

1 常规设置

2 Proxies

3 身份认证

4 用户组

5 用户

6 媒介类型

7 脚本

8 队列

2 用户资料

1 全局通知

2 浏览器中的声音

3 全局搜索

4 前端维护模式

5 页面参数

6 定义

7 制作自己的主题

19. API

方法参考

Action 动作

> 动作对象

action.create

action.delete

action.get

action.update

API 信息

apiinfo.version

Correlation 关联

> Correlation object 关联对象

```
correlation.create  
correlation.delete  
correlation.get  
correlation.update
```

Graph item 图形项

> Graph item object 图形项目对象

```
graphitem.get  
Graph prototype 原型图  
> Graph prototype object 图形原型对象  
graphprototype.create  
graphprototype.delete  
graphprototype.get  
graphprototype.update
```

Graph 图形

> Graph object 图形对象

```
graph.create  
graph.delete  
graph.get  
graph.update
```

Host interface 主机接口

> Host interface object 主机接口对象

```
hostinterface.create  
hostinterface.delete  
hostinterface.get  
hostinterface.massadd  
hostinterface.massremove  
hostinterface.replacehostinterfaces  
hostinterface.update
```

Host prototype 主机原型

> Host prototype object

```
hostprototype.create  
hostprototype.delete  
hostprototype.get  
hostprototype.update
```

Icon map 图标拓扑图

> Icon map object 图标拓扑图对象

```
iconmap.create
```

iconmap.delete

iconmap.get

iconmap.update

Item prototype Item原型

> Item prototype object

itemprototype.create

itemprototype.delete

itemprototype.get

itemprototype.update

LLD rule LLD规则

> LLD rule object LLD规则对象

discoveryrule.copy

discoveryrule.create

discoveryrule.delete

discoveryrule.get

discoveryrule.update

Maintenance 维护

> Maintenance object

maintenance.create

maintenance.delete

maintenance.get

maintenance.update

Map 地图

> Map object

map.create

map.delete

map.get

map.update

Media type媒体类型

> Media type object媒体类型对象

mediatype.create

mediatype.delete

mediatype.get

mediatype.update

Media媒介

> Media object媒体对象

usermedia.get

Problem 异常

> Problem object

problem.get

Proxy 代理

> Proxy object

proxy.create

proxy.delete

proxy.get

proxy.update

Service 服务

> Service object 服务对象

service.adddependencies

service.addtimes

service.create

service.delete

service.deletedependencies

service.deletetimes

service.get

service.getsla

service.update

Template screen item 模板聚合图形项

> Template screen item object 模板聚合图形项对象

templatescreenitem.get

Template screen 模板聚合图形

> Template screen object 模板聚合图形对象

templatescreen.copy

templatescreen.create

templatescreen.delete

templatescreen.get

templatescreen.update

Trigger prototype (触发器原型)

> Trigger prototype object (触发器原型对象)

triggerprototype.create

triggerprototype.delete

triggerprototype.get

triggerprototype.update

Web 场景

httpertest.create

httpertest.delete

http://test.get

http://test.update

Web场景对象

主机

> Host object 主机对象

host.create

host.delete

host.get

host.massadd

host.massremove

host.massupdate

host.update

主机组

> Host group object 主机组对象

hostgroup.create

hostgroup.delete

hostgroup.get

hostgroup.massadd

hostgroup.massremove

hostgroup.massupdate

hostgroup.update

事件

> Event object

event.acknowledge

event.get

值映射

> 值映射对象

valuemap.create

valuemap.delete

valuemap.get

valuemap.update

历史数据

> History object 历史对象

history.get

发现主机

> 主机发现对象

dhost.get

发现服务

> 服务发现对象

dservice.get

发现检查

> Discovery check object

dcheck.get

发现规则

> Discovery rule object 发现规则对象

drule.create

drule.delete

drule.get

drule.update

告警

> Alert object

alert.get

图像

> Image object 图像对象

image.create

image.delete

image.get

image.update

应用集

> 应用集对象

application.create

application.delete

application.get

application.massadd

application.update

模板

> Template object 模板对象

template.create

template.delete

template.get

template.massadd

template.massremove

template.massupdate

template.update

用户

> User object

user.addmedia
user.create
user.delete
user.deletemedia
user.get
user.login
user.logout
user.update
user.updatemedia
user.updateprofile

用户宏

> 用户宏对象
usermacro.create
usermacro.createglobal
usermacro.delete
usermacro.deleteglobal
usermacro.get
usermacro.update
usermacro.updateglobal

用户组

> 用户组对象
usergroup.create
usergroup.delete
usergroup.get
usergroup.massadd
usergroup.massupdate
usergroup.update

监控项

> Item object
item.create
item.delete
item.get
item.update

聚合图形

> Screen对象
screen.create
screen.delete
screen.get

screen.update

聚合图形项目

> 聚合图形项目对象

screenitem.create

screenitem.delete

screenitem.get

screenitem.update

screenitem.updatebyposition

脚本

> Script object(脚本对象)

script.create

script.delete

script.execute

script.get

script.getscriptsbyhosts

script.update

触发器

> Trigger object触发器对象

trigger.adddependencies

trigger.create

trigger.delete

trigger.deletedependencies

trigger.get

trigger.update

趋势

> Trend object趋势对象

trend.get

配置

configuration.export

configuration.import

附录1.参考

附录2.从3.2到3.4的变化

附录

1 常见问题\疑难解答

2 安装

1 数据库创建脚本

2 Windows 下的Zabbix agent

3 Elasticsearch setup

3 后端配置

- 1 Zabbix server
- 2 Zabbix proxy
- 3 Zabbix agent (UNIX)
- 4 Zabbix agent (Windows)
- 5 Zabbix Java 网关
- 6 "Include"参数的特别说明

4 监控项

- 1 不同平台支持的监控项
- 2 参数vm.memory.size
- 3 被动和主动代理检查
- 4 返回值的编码
- 5 大文件支持
- 6 不可达V不可用 主机设置
- 7 传感器
- 8 proc.mem 监控项中memtype参数类型的注意事项
- 9 在proc.mem和proc.num项目中选择进程的注意事项
- 10 net.tcp.service 和 net.udp.service 检查的实现细节

5 触发器

- 1 支持的触发器函数
- 6 宏
 - 1 Macros supported by location
- 7 设定时间段
- 8 执行指令
- 9 监控方案
- 10 性能调优
- 11 版本兼容性
- 12 数据库错误处理
- 13 Zabbix sender dynamic link library for Windows

致谢

当前文档《Zabbix 3.4 手册》由进击的皇虫使用书栈(BookStack.CN)进行构建，生成于 2019-06-01。

书栈(BookStack.CN)仅提供文档编写、整理、归类等功能，以及对文档内容的生成和导出工具。

文档内容由网友们编写和整理，书栈(BookStack.CN)难以确认文档内容知识点是否错漏。如果您在阅读文档获取知识的时候，发现文档内容有不恰当的地方，请向我们反馈，让我们共同携手，将知识准确、高效且有效地传递给每一个人。

同时，如果您在日常工作、生活和学习中遇到有价值有营养的知识文档，欢迎分享到书栈(BookStack.CN)，为知识的传承献上您的一份力量！

如果当前文档生成时间太久，请到书栈(BookStack.CN)获取最新的文档，以跟上知识更新换代的步伐。

内容来源：[Zabbix](https://www.zabbix.com/documentation/3.4/zh/manual) <https://www.zabbix.com/documentation/3.4/zh/manual>

文档地址：<http://www.bookstack.cn/books/zabbix3.4>

书栈官网：<http://www.bookstack.cn>

书栈开源：<https://github.com/TruthHun>

分享，让知识传承更久远！感谢知识的创造者，感谢知识的分享者，也感谢每一位阅读到此处的读者，因为我们都会成为知识的传承者。

Zabbix 手册

欢迎使用Zabbix 3.4软件使用手册，本手册可以帮助用户利用Zabbix实现对从简单到复杂的监控任务的高效管理。

= Overview ===

概述

Zabbix支持POSIX 正则表达式

There are two ways of using regular expressions in Zabbix:在Zabbix中有两种方法使用正则表达式

- manually entering a regular expression
- 手动输入正则表达式
- using a global regular expression created in Zabbix
- 使用在Zabbix中创建的全局正则表达式

Regular expressions

正则表达式

You may manually enter a regular expression in supported places. Note that the expression may not start with @ because that symbol is used in Zabbix for referencing global regular expressions.你可以在受支持位置中手动输入正则表达式。请注意，表达式不能以@开头，因为该符号在Zabbix中用于引用全局正则表达式。

Global regular expressions

全局正则表达式

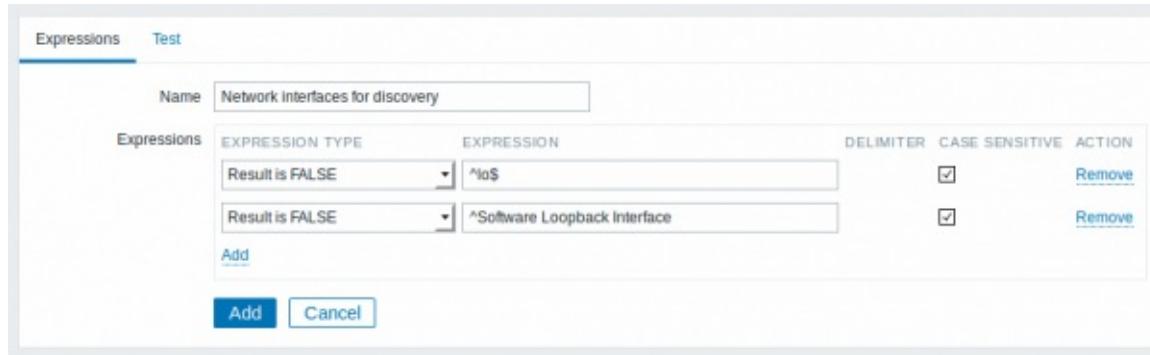
There is an advanced editor for creating and testing complex regular expressions in Zabbix frontend.有一个高级编辑器用于在Zabbix前端中创建和测试复杂的正则表达式。

Once a regular expression has been created this way, it can be used in several places in the frontend by referring to its name, prefixed with @, for example, @mycustomregexp.一旦以这种方式创建了正则表达式，它可以在前端的多个地方通过引用其名称（前缀为@）来使用，例如@mycustomregexp

To create a global regular expression:创建全局正则表达式

- Go to: *Administration* → *General*
- 进入：管理 (*Administration*) → 一般 (*General*)
- Select *Regular expressions* from the dropdown
- 从下拉列表中选择正则表达式 (*Regular expressions*)
- Click on *New regular expression*
- 点击新的正则表达式 (*New regular expression*)

The **Expressions** tab allows to set the regular expression name and add subexpressions.
表达式选项卡允许设置正则表达式名称和添加子表达式。



参数	说明
名称 (<i>Name</i>)	设置正则表达式名称。允许任何Unicode字符。
表达式 (<i>Expressions</i>)	单击表达式区域中的添加 (Add) 以添加新的子表达式。
表达式类型 (<i>Expression type</i>)	选择表达式类型:字符串已包含 (Character string included) - 匹配子字符串包含任何字符串 (Any character string included) - 匹配逗号分隔列表中的任何子字符串字符串未包含 (Character string not included) - 匹配除了子字符串之外的任何字符串结果为真 (Result is TRUE) - 匹配正则表达式结果为假 (Result is FALSE) - 不匹配正则表达式
表达式 (<i>Expression</i>)	输入子字符串/正则表达式。

Since Zabbix 2.4.0, a forward slash (/) in the expression is treated literally, rather than a delimiter. This way it is possible to save expressions containing a slash, whereas previously it would produce an error.

A custom regular expression name in Zabbix may contain commas, spaces, etc. In those cases where that may lead to misinterpretation when referencing (for example, a comma in the parameter of an item key) the whole reference may be put in quotes like this: "@My custom regexp for purpose1, purpose2". Regular expression names must not be quoted in other locations (for example, in LLD rule properties).

A custom regular expression may consist of multiple subexpressions, and it can be tested in the **Test** tab by providing a test string.

Result	Expression type	Expression	Result
Result is FALSE		<code>^Software Loopback Interface</code>	TRUE
Result is FALSE		<code>^(In)?[Ll]oop[Bb]ack[0-9._]*\$</code>	TRUE
Result is FALSE		<code>^NULL[0-9.]*\$</code>	TRUE
Result is FALSE		<code>^[Uu]o[0-9.]*\$</code>	FALSE
Result is FALSE		<code>^Ss]ystem\$</code>	TRUE
Result is FALSE		<code>^Nu[0-9.]*\$</code>	TRUE
Combined result			FALSE

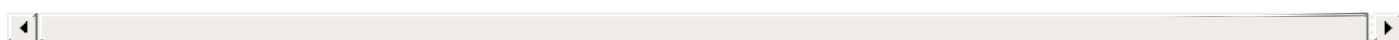
[Update](#) [Clone](#) [Delete](#) [Cancel](#)

Results show the status of each subexpression and total custom expression status.

Regular expression support by location

Location	Regexp support	Global regexp support	Comments
Macro functions			
	<code>regsub()</code>	Yes	No pattern parameter
<code>iregsub()</code>			
Trigger functions			
	<code>count()</code>	Yes	Yes pattern parameter if operator parameter is <i>regexp</i> or <i>iregexp</i>
<code>logeventid()</code>	pattern parameter		
<code>iregexp()</code>			
<code>regexp()</code>			
Low-level discovery			
	Yes	Yes	<i>Filter field</i>
Web monitoring			
	Yes	No	<i>Variables with a regex:</i> <i>prefixRequired</i> <i>string field</i>

Zabbix agent items				
	eventlog[]	Yes	Yes	regexp, severity, source, eventid parameters
log[]	regexp parameter			
log.count[]				
logrt[]	Yes/No	regexp parameter supports both, file_regexp parameter supports non- global expressions only		
logrt.count[]				
proc.cpu.util[]	No	cmdline parameter		
proc.mem[]				
proc.num[]				
sensor[]		device and sensor parameters on Linux 2.4		
system.hw.macaddr[]		interface parameter		
system.sw.packages[]		package parameter		
vfs.dir.size[]		regex_incl and regex_excl parameters		
vfs.file.regexp[]		regexp parameter		
vfs.file.regmatch[]				
web.page.regexp[]				
SNMP traps				
	snmptrap[]	Yes	Yes	regexp parameter
Icon mapping				
	Yes	Yes		<i>Expression</i> field



1. 介绍

请使用侧边导航栏来访问此章节的内容。

1 手册结构

结构

Zabbix 3.4手册的内容分为几个章节和子章节，以便于访问特定的感兴趣的主題。

当您导航到相应的章节时，请确保展开该章节的文件夹以完整显示子章节和单个页面中包含的内容。

页面之间交叉链接的相关内容将会尽可能的提供出来，确保用户不会错过相关信息。

章节

[简介](#) 提供关于当前Zabbix软件的常用信息。阅读本章节将为您选择使用Zabbix提供一些好的理由。

[Zabbix 概念](#) 解释了Zabbix中使用的术语，并且提供了Zabbix组件的详细信息。

[安装 和 快速入门](#)章节可以帮助您开始使用Zabbix。[Zabbix 应用](#) 是一个替代的方案，通过本章节，可以快速的使用Zabbix应用，并了解Zabbix应用是什么。

[配置](#) 是本手册中内容最多最重要的章节之一。它包含一些重要的建议，关于如何设置Zabbix来监控您的环境、如何从主机设置到获取必要的数据、如何查看数据、如何配置告警通知和在出现问题时执行远程命令等。

[IT 服务](#) IT服务章节详细的说明了如何利用Zabbix提升监控环境的高稳定性。

[Web 监控](#) 可以帮助您学会怎么样去监控Web网站的可用性。

[虚拟机监控](#)提供了如何配置VMware虚拟机环境监控的方法。[维护, 正则表达式, 事件确认 and XML 导出/导入](#) 这些章节进一步说明了如何全面的使用Zabbix软件的功能。

[发现](#) 功能包含网络设备自动发现的指令，主动监控的指令，文件系统自动发现的指令，网络接口自动发现的指令等。

[分布式监控](#) 可以使用Zabbix系统支撑更庞大更复杂的环境。

[加密](#) 功能可以实现Zabbix组件之间的通讯加密。

[Web 界面](#) 包含Zabbix Web界面使用的特定信息。

[API](#) 章节详细的说明了Zabbix API的使用。

详细的技术细节表包含在[附录](#)中。附录也包含常见问题的详细解答。

2 Zabbix是什么

概述

Zabbix 是由Alexei Vladishev创建，目前由Zabbix SIA在持续开发和支持。

Zabbix 是一个企业级的分布式开源监控方案。

Zabbix是一款能够监控各种网络参数以及服务器健康性和完整性的软件。Zabbix使用灵活的通知机制，允许用户为几乎任何事件配置基于邮件的告警。这样可以快速反馈服务器的问题。基于已存储的数据，Zabbix提供了出色的报告和数据可视化功能。这些功能使得Zabbix成为容量规划的理想方案。

Zabbix支持主动轮询和被动捕获。Zabbix所有的报告、统计信息和配置参数都可以通过基于Web的前端页面进行访问。基于Web的前端页面可以确保您从任何方面评估您的网络状态和服务器的健康性。适当的配置后，Zabbix可以在IT基础架构监控方面扮演重要的角色。对于只有少量服务器的小型组织和拥有大量服务器的大型公司也同样如此。

Zabbix是免费的。Zabbix是根据GPL通用公共许可证第2版编写和发行的。这意味着它的源代码都是免费发行的，可供公众任意使用。

[商业支持](#) 由Zabbix公司提供。

[了解更多Zabbix特性](#).

Zabbix的用户

世界各地许多不同规模的组织将Zabbix作为主要的监控平台。

3 Zabbix功能点

概述

Zabbix是一个高度集成的网络监控解决方案，一个简单的安装包中提供多样性的功能。

数据收集

- 可用性和性能检查
- 支持SNMP（包括主动轮训和被动获取），IPMI，JMX，VMware监控
- 自定义检查
- 按照自定义的间隔收集需要的数据
- 通过server/proxy+agents来执行

灵活的阈值定义

- 您可以非常灵活的定义问题阈值，称之为触发器，触发器从后端数据库获取参考值

高度可配置化的告警

- 可根据递增机制，接收方和媒介类型自定义发送告警通知
- 使用宏变量可以使告警通知更加高效有用
- 自动相应动作可包含远程命令

实时图表绘制

- 使用内置图表绘制功能可以将监控项的内容实时绘制成图表

Web监控功能

- Zabbix可以追踪模拟鼠标在Web网站上的点击操作，来检查Web的功能和响应时间

丰富的可视化选项

- 支持创建自定义的图表，一个试图集中展现多个监控项
- 网络拓扑图
- 以仪表盘的样式自定义大屏展现和幻灯片轮询播放
- 报表
- 监控内容的高级（业务）视图

历史数据存储

- 数据库数据
- 可配置历史数据
- 内置数据管理机制 (housekeeping)

配置简单

- 将被监控对象添加为主机
- 在数据库中获取主机进行监视
- 应用模板来监控设备

使用模板

- 在模板中分组检查
- 模板可以关联其他模板

网络发现

- 自动发现网络设备
- 监控代理自动注册
- 发现文件系统，网络接口和SNMP OID值

快捷的Web界面

- PHP Web前端
- 可从任何地方访问
- 你可以定制自己的操作方式
- 审核日志

Zabbix API

- Zabbix API为Zabbix 提供了对外的可编程接口，用于批量操作，第三方软件集成和其他目的

权限管理系统

- 安全用户认证
- 特定用户可以限制访问特定的视图

功能强大，易于扩展的agent

- 部署在被监控对象上
- 支持Linux和Windows

二进制代码

- 为了性能和更少内存的占用，用C语言编写
- 便于移植

为复杂环境准备

- 使用Zabbix proxy代理服务器，使得远程监控更简单

4 Zabbix概述

结构

Zabbix由几个主要的软件组件构成，这些组件的功能如下。

Server

Zabbix server 是agent程序报告系统可用性、系统完整性和统计数据的核心组件，是所有配置信息、统计信息和操作数据的核心存储器。

数据库存储

所有配置信息和Zabbix收集到的数据都被存储在数据库中。

Web界面

为了从任何地方和任何平台都可以轻松的访问Zabbix，我们提供基于Web的Zabbix界面。该界面是Zabbix Server的一部分，通常(但不一定)跟Zabbix Server运行在同一台物理机器上。

如果使用SQLite，Zabbix Web界面必须要跟Zabbix Server运行在同一台物理机器上。

Proxy代理服务器

Zabbix proxy 可以替Zabbix Server收集性能和可用性数据。Proxy代理服务器是Zabbix软件可选择部署的一部分；当然，Proxy代理服务器可以帮助单台Zabbix Server分担负载压力。

Agent监控代理

Zabbix agents 监控代理 部署在监控目标上，能够主动监控本地资源和应用程序，并将收集到的数据报告给Zabbix Server。

数据流

此外，了解Zabbix内部的数据流同样很重要。监控方面，为了创建一个监控项(item)用于采集数据，必须先创建一个主机(host)。告警方面，在监控项里创建触发器(trigger)，通过触发器(trigger)来触发告警动作(action)。因此，如果你想收到Server X_CPU负载过高的告警，你必须：1. 为_Server X创建一个host并关联一个用于对CPU进行监控的监控项(Item)。2. 创建一个Trigger，设置成当CPU负载过高时会触发3. Trigger被触发，发送告警邮件虽然看起来有很多步骤，但是使用模板的话操作起来其实很简单，Zabbix这样的设计使得配置机制非常灵活易用。

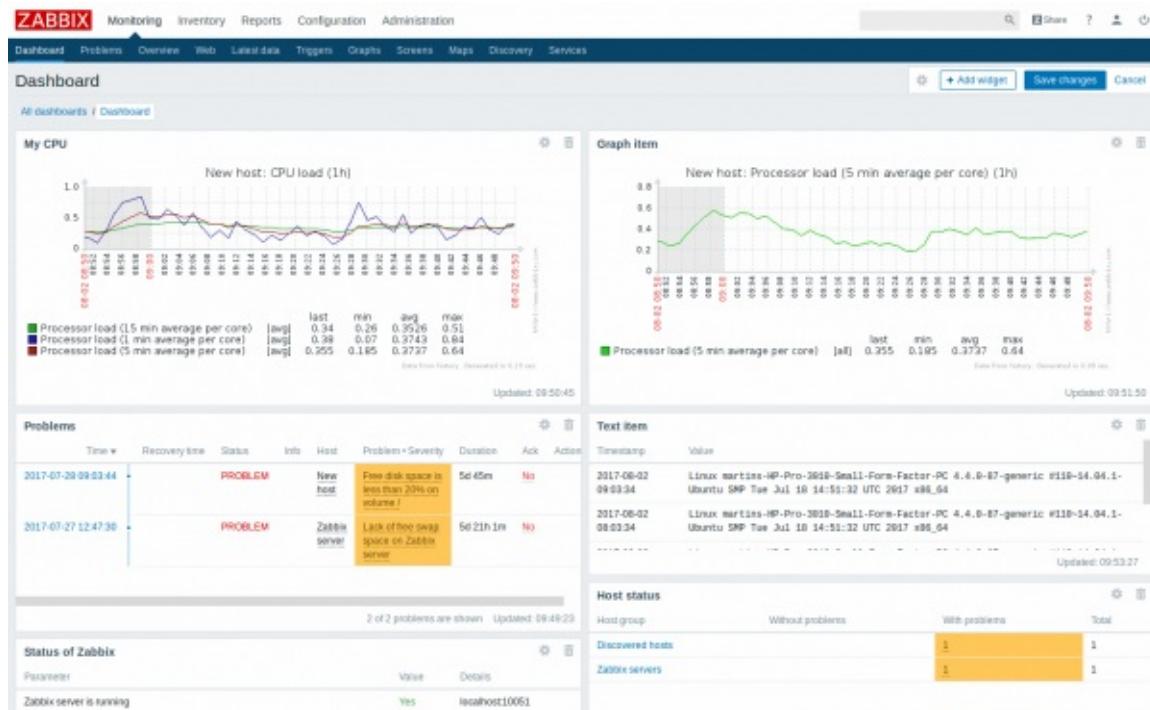
5 Zabbix 3.4.0的新特征

5.1 新仪表板

新版本的Zabbix仪表板将汇总、概览重要信息这一功能提升到了一个新的水平，新版本现在可以支持建立多个仪表板，而在Zabbix之前的版本中只有一个。

每个仪表板由可以自定义的小部件组成，用户可以自己选择其中的内容。这部分升级是通过集成了Zabbix Screen中最优秀的一些功能和Dashboard的功能而实现的。因此，新版本的仪表板小部件包含以前的仪表板常用小部件，也新加入了Zabbix Screen管受好评的功能（如图形，简单图形，地图，触发器等）

许多以前可用于构建Zabbix Screen的元素现在可以作为仪表板的小部件放置在仪表板上，小部件也可以自定义名称。



还有全新的小部件：

- **问题** - 这个小部件替换了上一个版本中 最近20个故障 部件的功能，采用类似于监控 → 问题 部分的方式展示问题。
- **Map 导航树** - 这个小部件允许构建现有Map的等级结构，[点击查看更多](#) .

仪表板中的过滤功能已经被删除，反之，过滤功能可以应用于各个小部件，如主机状态，系统状态，etc.

更多详细内容，参见：

- [仪表板](#)
- [仪表板部件](#)

5.1.1 Map导航树

这个新的部件允许构建现有map的层次结构，同时也能显示每个map包含的问题的统计信息。

如果将Map小部件链接到导航树，它变得更加强大。在这种情况下，单击导航树中的map名称将在Map小部件中完全显示map内容。



层次结构中的第一级map的统计信息显示了所有子map和其自身问题的总和。

5.2 网络设备监控模板

为了提供开箱即用的监控功能给网络设备，如交换机和路由器，已经开发了基于SNMPv2的新模板。这些模板用于许多网络设备，更多信息，参见：

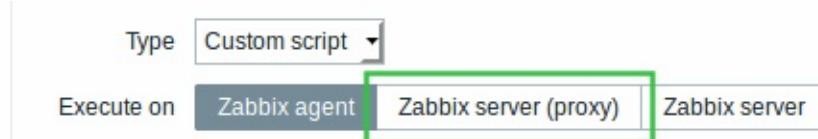
- [标准化网络模板](#)

5.3 Proxies代理支持远程命令

如果目标主机是通过Zabbix proxy代理实现监控的话，过去无法由Zabbix Agent代理来执行远程命令和全局脚本。类似的，命令也不能被proxy代理自身执行，命令始终只能由Zabbix Server来执行。

在Zabbix 3.4中，远程命令和全局脚本在由proxy代理监控的主机上可以正确执行，该命令由目标主机上的agent代理执行。

还可以通过proxy代理本身执行远程命令或全局脚本，这种方式作为执行操作/全局脚本配置中的新选项提供 - 由server或proxy代理执行命令，具体取决于主机是由server还是proxy代理监控。



请注意，默认情况下未启用proxy代理执行远程命令。在安全环境中（加密，SSH等），可以通过将“EnableRemoteCommands”参数设置为1来启用proxy代理上的执行远程命令。即使禁用远程命令，也可以执行IPMI，SSH和Telnet脚本。

5.4 从属监控项

有一个监控项一次收集多个指标的场景，或者同时收集相关指标，例如：

- 单个内核的CPU使用率
- 流入/流出/总的网络流量

为了支持这项功能，Zabbix现在支持从属监控项目。从属监控项有一个主监控项，它在一个查询策略中收集所有的指标。从属监控项使用主监控项的数据来收集她们的数据，主监控项的新数据值自动填充从属监控项的值。

也可参见：[从属监控项](#)

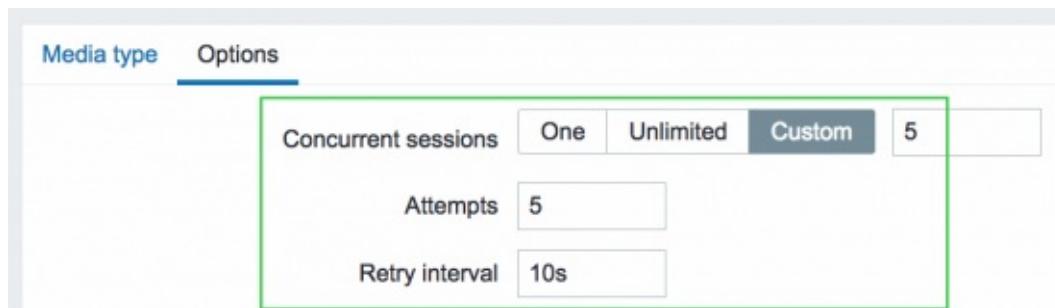
5.5 告警并行处理

在以前的版本中使用单个告警器进程来发送问题通知，告警是一个一个的发出，在大规模的环境中有大量事件紧挨连续发生的情况下，告警可能会发生延迟。类似地，在实时性较高和实时性较低的媒体类型(如短信和电子邮件)混合存在的环境中，可能会存在延时，邮件的发送需要等待短信发送完成。

在新版本中，并行处理告警功能已经实现，有一个新的告警管理器进程，如果需要，可以向多个“worker”进程分发告警。媒体类型被并行处理，每个媒体类型可以配置最大并发会话数，但服务器上的告警器进程总数只能由新的 `StartAlerters` 参数限制，每个触发器生成的告警都会顺序的进行处理。

还有其他相关的更新：

- 有三个可用的新告警处理选项在媒体类型配置中：并发会话，重试 和 重试间隔：



- 数据库看门狗进程的功能已经合并到告警管理器中，并且看门狗进程本身也被删除。

5.6 已通知的问题确认

现在可以在确认触发器生成的问题时收到Zabbix所有可用方式的通知，为了实现这一点，而创建了一种新的操作类型，称为确认操作，其有自己的专用配置选项卡。

The screenshot shows the 'Acknowledgement operations' tab selected. It includes fields for 'Default subject' (Acknowledged: {TRIGGER.NAME}) and 'Default message' (containing placeholders for user full name, acknowledgement date/time, and message). Below these are operational details: 'Notify all who left acknowledgement and comments' and 'Send message to users: Admin (Zabbix Administrator) via SMS'.

在已确认的通知信息中可以包含用户和用户作为确认输入的文本，确认通知可以发送给指定的用户/用户组和/或者所有确认问题并留下评论的用户。

远程命令也可以在问题确认时执行。

也可以参见：[确认操作](#)

5.7 监控项数据预处理

在将数据存储在数据库中之前，预先处理监控项数据，Zabbix已经有了几个选项，例如计算增量值，使用自定义乘数，转换值类型或修剪长文本值。这些选项作为监控项配置中的单独属性存在，或者是硬编码。

在新版本中，所有监控项数据预处理都汇集在一起，并在监控项配置的新功能预处理选项卡中下放入用户手中。

The screenshot shows the 'Preprocessing' tab for an item. It lists three steps: 'Change per second', 'Custom multiplier' (set to 8), and 'Regular expression'. The 'Regular expression' step includes fields for 'pattern' and 'output'.

新预处理选项

几个新增的数据预处理功能选项：

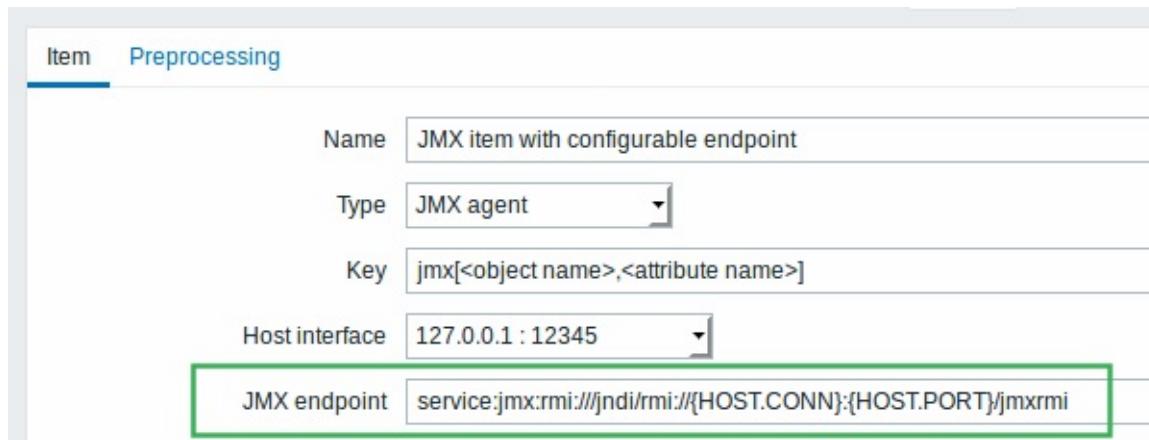
- 正则表达式 - 通过正则表达式/输出模板找到匹配内容；
- XML XPath - 使用XPath从XML数据中提取数值或片段；
- JSON Path - 使用JSONPath从JSON数据中提取数值或片段（仅支持一组有限的JSONPath功能）。

另外 *Delta* (简单更改) 和 *Delta* (每秒速度) 选项已重命名为简单更改和每秒更改。

有关监控项数据预处理选项和更多详细信息的完整列表，请参阅[监控项配置](#)。

5.8 可配置的JMX端点

以前，一个JMX端点是在Zabbix中进行硬编码，这是有局限性的，因为有一些应用程序使用不同的端点。为了解决这个限制，在JMX监控项配置中，新添加了可配置作为单独字段的JMX端点支持，当您打开JMX监控项时，最初该字段将填充默认端点，但现在可以自由修改。



在JMX端点字段中支持{HOST.*}宏变量，用户宏变量和低级别发现的宏变量。

5.9 JMX低级别发现

新的 `jmx.discovery [<discovery mode>, <object name>]` 监控项支持JMX对象的低级别发现，此监控项允许指定是发现MBean还是MBean属性，以及要查找的模式。更多细节，参见[JMX发现部分](#)

5.10 用于正则表达式的PCRE库

Zabbix中的正则表达式支持已从POSIX扩展正则表达式切换到[Perl Compatible Regular Expressions \(PCRE\)](#)，以增强正则表达式和前端的一致性。

从以前的版本升级时，请务必阅读相应的[升级注意事项](#)！

5.11 Web监控中的URL编码支持

以前web监控中的变量值不变的情况下，变量值的任何URL编码只能手动完成。

URL自动编码

现在，Web监控方案步骤中输入的GET和POST变量值将自动进行URL编码，无论使用什么类型的数据（文本，宏，场景级别变量或其任何组合）都应用此编码方式。在执行步骤之前执行编码（数据保存在数据库中未编码）。

变量灵活的URL编码

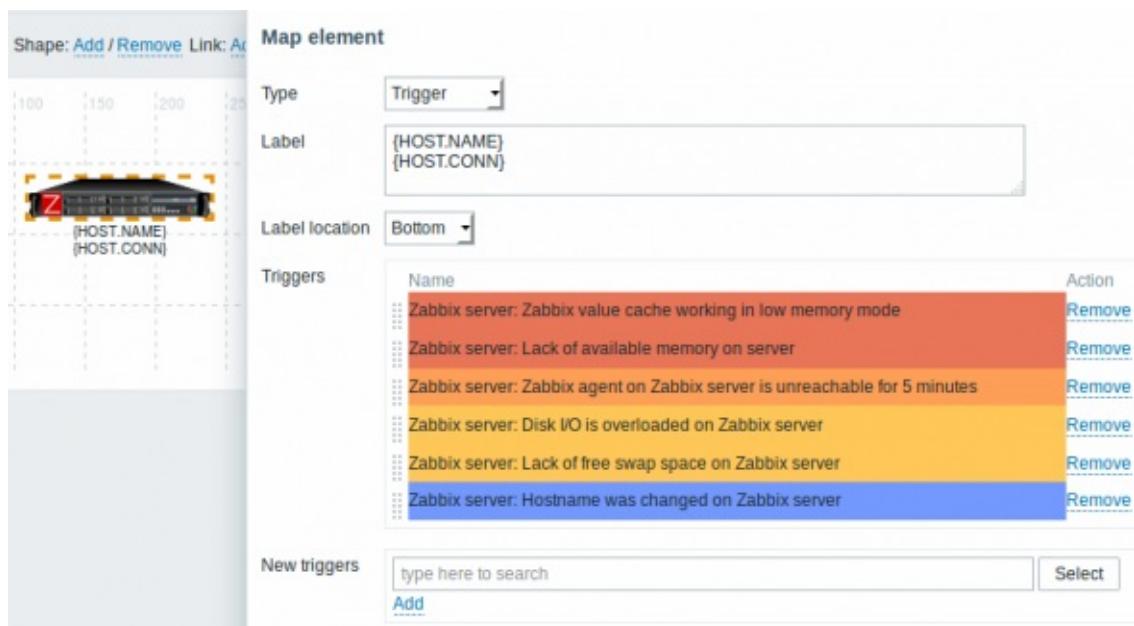
可以灵活地对URL或变量（宏）的值进行编码/解码，这取决于Web监控方案步骤中选定的后变量设置。例如：

版本	变量语法	结果
3.4版之前	{user}	变量值按原样传递。
3.4版		

也可参见：[网络map配置](#)

支持多触发器

当创建一个触发类型的map元素时，现在可以为此元素选择多个触发器，而不是像之前一样只能选择单个触发器。



此外，展开单个问题

在map属性中的设置已经重新设计成三项选择，这里的新选项称为问题数量并扩展最关键的一个。此选项与多重触发器支持相关，如果选择此选项，最重要的问题（最高严重度触发）将通过其名称显示在map元素上，而问题总数将显示在另一行中。



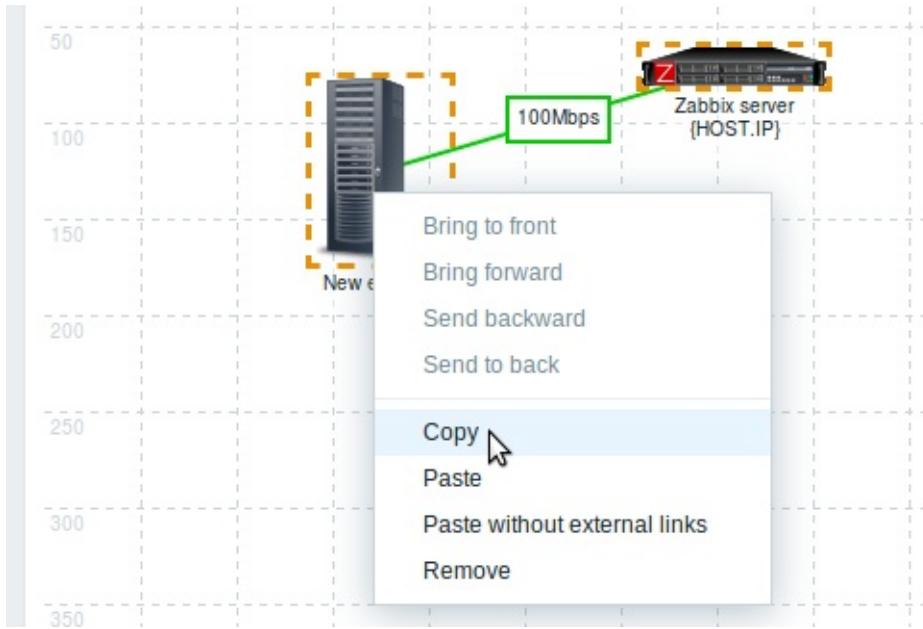
也可参见：[网络map配置](#)

移动元素

map元素的拖拉支持功能已经引入，通过点击所选元素之一，按住鼠标按钮并将光标移动到所需位置，可以将几个选定的元素移动到map中的另一个位置。

复制和粘贴元素

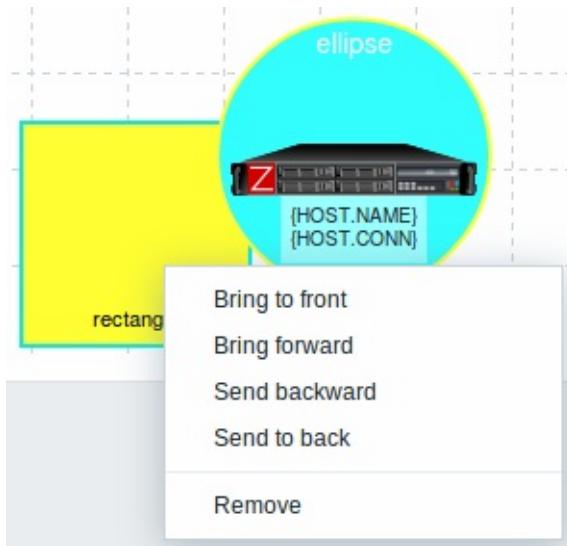
Map元素当被选中时，现在可以在同一个map中复制和粘贴。



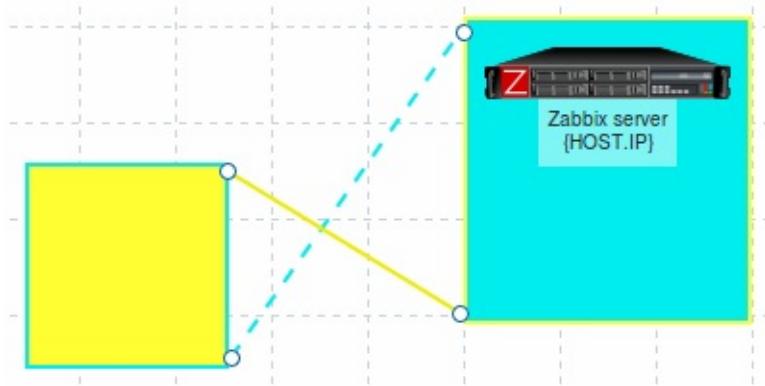
多个选定的元素现在也可以一起拖放到另一个地方。

添加形状和线条

矩形和椭圆形现在可以在网络map中 [添加](#)。这些形状主要是视觉展现，例如，矩形形状可以用作背景来对某些主机分组。形状不是map元素，不能配置链接，文字可以加入形状中。



自由绘制的线条是现在可以添加到网络map中的另一个元素。



5.12 在时间段内支持宏和时间后缀

在监控项更新间隔和Zabbix中指定时间段的更多位置，现在支持用户宏和时间后缀（如30秒，5分钟，2小时，1d，1w）。请注意，在某些时间段内，仅添加了用户宏支持，而仅在后缀支持中。有关支持的位置的完整列表，请参阅：

- [用户宏支持](#)
- [时间后缀](#)

用户宏对快速配置更改非常有用。例如，可以为监控项更新间隔定义用户宏，然后，如果需要更改监控项轮询频率，则只需更改用户宏的值，并在所有使用宏的监控项中更改监控项的更新间隔。

此外，监控项原型更新间隔和历史/趋势存储周期字段现在支持低级发现宏。

在相关的更新中，时间段的一些上限/下限已经改变。重要的是，监控项的历史存储期现在可以短至1小时。

5.13 事件标签中的主机宏支持

主机宏 - `{HOST.HOST<1-9>}`, `{HOST.NAME<1-9>}`, `{HOST.CONN<1-9>}`, `{HOST.DNS<1-9>}`,
`{HOST.IP<1-9>}`, `{HOST.PORT<1-9>}`, `{HOST.ID<1-9>}`现在支持事件标签名称和值，使得更容易在模板上指定与主机相关的标签，或将全局事件关联到其主机。

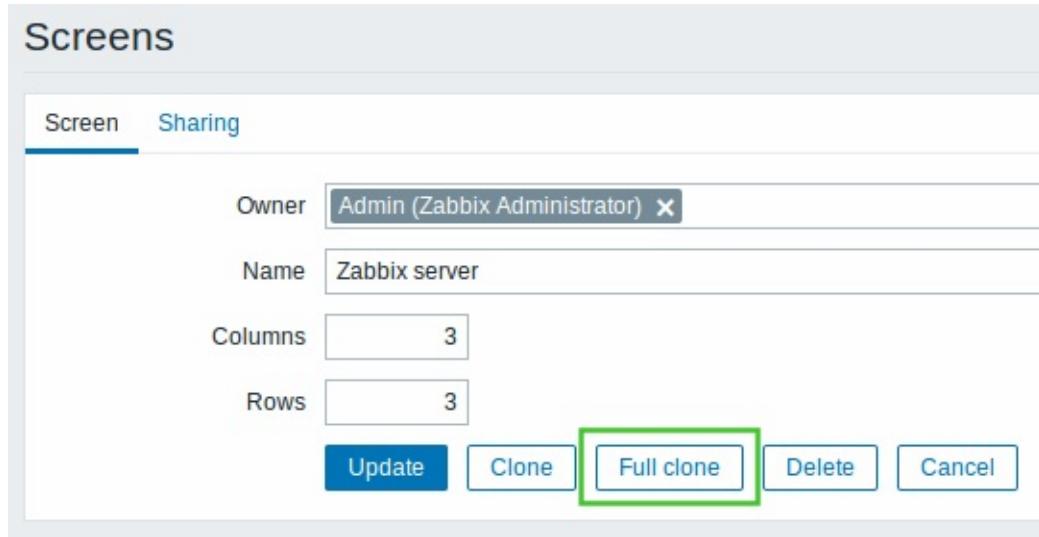
5.14 前端的改进

5.14.1 去掉对IE9和IE10的支持

不再提供对微软IE9和IE10浏览器的支持。

5.14.2 完整克隆screens和maps

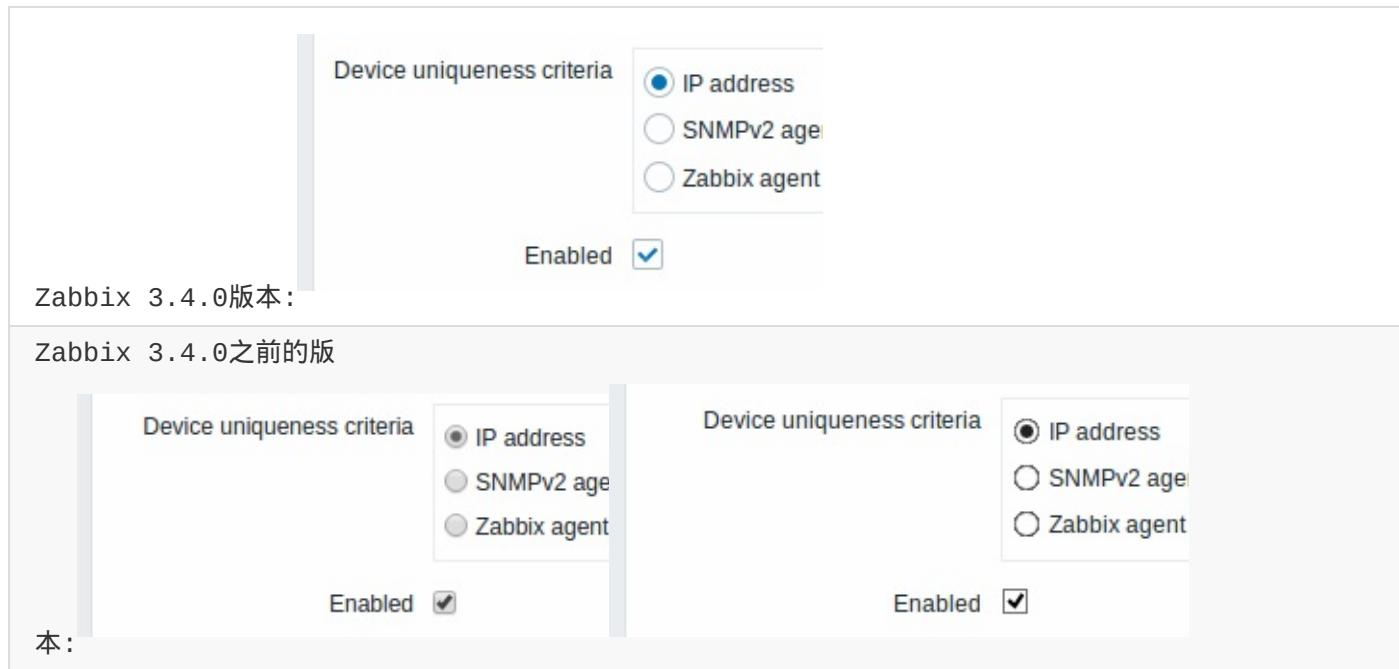
单个screen和模板screens以及网络map现在可以完全克隆，这意味着不仅可以布局，而且可以对所有screen/map元素进行克隆。



要完全克隆screen，请单击完全克隆按钮，这将临时保存screenid，然后给screen一个不同的名称，然后单击添加按钮，将创建一个新的，包含所有符合screen布局元素的screen。

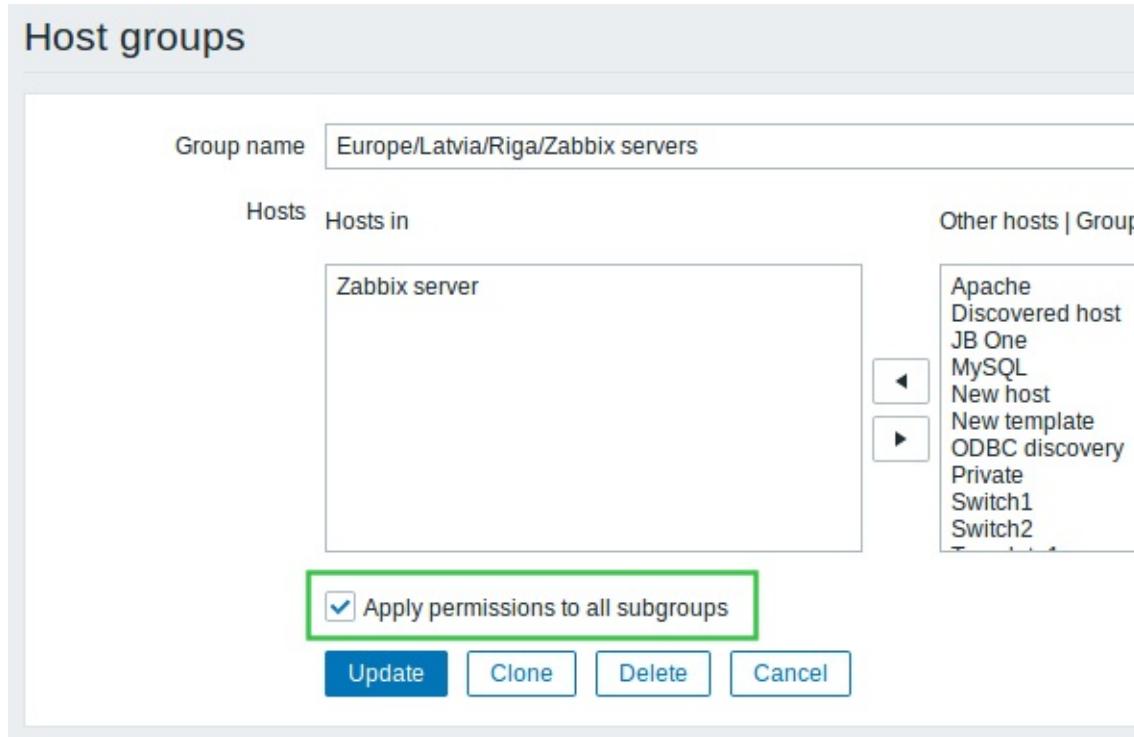
5.14.3 单选按钮和复选框风格的统一

单选按钮和复选框的风格已在不同浏览器实现统一。



5.14.4 将相同的权限应用于嵌套的主机组

在主机组配置中有一个新选项，允许将父主机组的权限级别设置为所有嵌套的主机组，如果选中此复选框并更新主机组，对于可能已将不同权限分配给嵌套主机组的用户组，将在嵌套组上强制执行父组的权限级别。这是一个一次性选项，它不保存在数据库中，仅适用于Zabbix超级管理员用户，并且只在编辑现有主机组时才可用。



5.14.5 字段大小的增加

- 监控项，监控项过滤器，低级发现规则和网络发现规则配置中的SNMP OID字段长度从255字节增加到512字节
- 监控项配置中的灵活间隔周期字段长度由255字节增加到1024字节
- 用户媒体中何时激活字段长度由100字节增加到1024字节。

5.14.6 其他

- 现在非超级管理员用户也可以隐藏SQL报错。更多详细信息，请参阅Zabbix前端中的ZBX_SHOW_SQL_ERRORS的[定义](#)。

5.15 守护进程改善

- SNMP OID长度限制由255字节增加到512字节。
- Zabbix可以读取的SNMP陷阱文件大小限制从 2^{31} (2 GiB) 增加到 2^{63} (8 EiB)。
- Java网关增加了对AtomicBoolean, AtomicInteger和AtomicLong类型的支持。
- 优化了Server-Proxy代理数据交换协议，以减少从Proxy代理服务器发送历史（主机可用性，监控项历史记录，发现和自动注册）数据到Server的连接数。当然，Server也将接受来自3.2（及更旧版本）Proxy代理的历史数据，保持部分向下的兼容性。
- 内部自动发现和Agent代理自动注册事件的数据存储期已从365天降至1天。
- 错误消息最大长度已从128个符号增加到2048个符号或触发器和告警错误。这应该减少错误消息被裁剪的可能性。错误消息可以在配置 - > 主机 - > 触发器和用于警告的报告 - > 操作日志中查看，滚动鼠标时查看错误图标。

- 信号量和共享内存管理已经重新设计，以消除Zabbix守护程序之间以及Zabbix和其他应用程序之间的IPC相关冲突的可能性。
- 配置同步期间的缓存锁定时间已经减少。

5.15.1 IPMI轮询

以前一个BMC控制器可以被不同的进程查询，为了提高轮询速度，每个IPMI轮询器都保存一个连接缓存。有了大量的IPMI轮询器的情况下，可以轻松的超载BMC控制器。

自Zabbix 3.4以来，已经添加了一个新的IPMI管理器进程来规划对IPMI轮询器的IPMI检查。现在主机总是被同一个IPMI轮询器轮询，这样减少了与BMC控制器打开的连接数，通过这些更新，可以安全地增加IPMI轮询器的数量，而不必担心BMC控制器重载。只要有一个IPMI轮询器启动时，IPMI管理器进程也会自动启动。

5.15.2 配置参数

Zabbix Server配置新增了一个 `StartAlerters` 参数，此版本引入并行处理的告警机制，“`StartAlerters`”参数决定了Zabbix Server可以启动多少个报警器进程。

Zabbix Server和proxy配置新增一个 `SocketDir` 参数，此参数指向存储内部Zabbix套接字文件的目录（默认为`/tmp`）。Server和Proxy代理使用不同的套接字文件名，因此在同一系统上运行的Server和Proxy代理使用相同的“`SocketDir`”是安全的。但是，在同一系统中运行多个Server或Proxy代理将需要不同的“`SocketDir`”配置。

`EnableRemoteCommands` 和 `LogRemoteCommands` 参数已被添加到Proxy代理配置文件中，因此现在也通过Proxy代理支持远程命令，两者默认都是禁用的。

虽然 `MaxLinesPerSecond` Agent代理参数的上限保持不变（1000），当涉及到Agent代理可以读取的总行数时，Agent代理每秒可以向Zabbix Server发送的新日志行限制现在可以乘以10（而不是4）。

5.15.3 关于agent代理的公制线程崩溃的更多信息

以前，如果Zabbix agent代理公制线程崩溃，则只会记录一行错误消息。在这种情况下，agent代理日志文件中提供了附加信息，包括程序计数器，寄存器，堆栈帧（仅在32位版本中）和回溯。在其他额外的改进中，执行公制线程的记录返回值已从数字更新为字符串，以提高可读性。

5.16 监控项的变更和改进

已新增一个新的 `vfs.dir.size` Agent代理监控项来监控目录的大小，这个监控项支持UNIX和Windows平台。

对于 `proc.num` Agent代理监控项，支持两个附加状态：

- disk* - process in uninterruptible disk sleep (usually I/O)
- disk* - 不间断磁盘，睡眠中的进程（通常是I/O）
- trace* - process is stopped by job control signal
- trace* - 由作业控制信号的进程停止

添加一个新的`zabbix[host, discovery, interfaces]`内部监控项来返回在Zabbix前端配置的所有主机接口，这

个监控项可以使用在[低级返现](#)中。

5.17 低级发现

- 添加了对触发原型表达式的函数参数中的LLD宏的支持。
- 添加在触发原型名称，描述和标签中使用ITEM.VALUE, ITEM.LASTVALUE的功能参数LLD宏支持。
- 添加LLD宏在函数参数中的原型名称简单宏如{host: key [] . func ()}的支持。

5.18 脚本和命令返回码检查

现在Zabbix检查用户参数，远程命令和system.run[]监控项的退出代码，而没有对“nowait”标志以及Zabbix服务器执行的脚本（告警，外部脚本和全局脚本）的检查。在脚本或命令执行过程发生错误的情况下，Zabbix在前端提供[错误描述](#)，并创建相应的日志条目。

Step	Time	User	Details	Status	Info
Problem					
1	2017-01-03 16:00:00	Admin (Zabbix Administrator)	failer	Failed	i
1	2017-01-03 16:00:00		Remote command	Media type error Segmentation fault	x

2. 定义

概述

在本节中，你可以了解一些Zabbix常用术语的含义。

定义

主机__*(host)*

- 一台你想监控的网络设备，用IP或域名表示

主机组__*(host group)*

- 主机的逻辑组；它包含主机和模板。一个主机组里的主机和模板之间并没有任何直接的关联。通常在给不同用户组的主机分配权限时候使用主机组。

监控项__*(item)*

- 你想要接收的主机的特定数据，一个度量数据。

触发器__*(trigger)*

- 一个被用于定义问题阈值和“评估”监控项接收到的数据的逻辑表达式

当接收到的数据高于阈值时，触发器从“OK”变成“Problem”状态。当接收到的数据低于阈值时，触发器保留/返回一个“OK”的状态。

事件__*(event)*

- 单次发生的需要注意的事情，例如触发器状态改变或发现有监控代理自动注册

异常__*(problem)*

- 一个处在“异常”状态的触发器

动作__*(action)*

- 一个对事件做出反应的预定义的操作。

一个动作由操作(例如发出通知)和条件(当时操作正在发生)组成

升级__*(escalation)*

- 一个在动作内执行操作的自定义场景；发送通知/执行远程命令的序列

媒介__*(media)*

- 发送告警通知的手段；告警通知的途径

通知__*(notification)*

2. 定义

- 利用已选择的媒体途径把跟事件相关的信息发送给用户

远程命令 *(remote command)*

- 一个预定义好的，满足一些条件的情况下，可以在被监控主机上自动执行的命令

模版 *(template)*

- 一组可以被应用到一个或多个主机上的实体（监控项，触发器，图形，聚合图形，应用，*LLD*, *Web*场景）的集合

模版的任务就是加快对主机监控任务的实施；也可以使监控任务的批量修改更简单。模版是直接关联到每台单独的主机上。

应用 *(application)*

- 一组监控项组成的逻辑分组

web 场景 *(web scenario)*

- 利用一个或多个HTTP请求来检查网站的可用性

前端 *(frontend)*

- Zabbix提供的web界面

Zabbix API

- Zabbix API允许你使用JSON RPC协议来创建、更新和获取Zabbix对象（如主机、监控项、图形和其他）信息或者执行任何其他的自定义的任务

Zabbix server

- Zabbix软件实现监控的核心程序，主要功能是与Zabbix proxies和Agents进行交互、触发器计算、发送告警通知；并将数据集中保存等

Zabbix agent

- 一个部署在监控对象上的，能够主动监控本地资源和应用的程序

Zabbix proxy

- 一个帮助Zabbix Server收集数据，分担Zabbix Server的负载的程序

3. Zabbix 进程

请使用侧边导航栏来访问Zabbix进程章节的内容。

1 服务器

概 览

Zabbix server是整个Zabbix软件的核心程序。

Server通过轮询和捕获数据，计算是否满足触发器条件，向用户发送通知。它是Zabbix监控代理和Proxy代理报告系统可用性和完整性数据的核心组件。Server自身可以通过简单服务远程检查网络服务(如Web服务器和邮件服务)。

Sever是一个包含了被存储了所有配置，统计方面的和可操作数据的中央仓库，它是监控系统问题升级以至于激活警告管理器的Zabbix中的实体。

基本的Zabbix服务器起作用分三个不同的组件；他们是：Zabbix服务器，Web前端和数据库存储。

Zabbix的所有配置信息都存储在服务器和Web前端进行交互的数据库中。Zabbix的所有配置信息都存储在服务器和Web前端进行交互的数据库中。例如，当你通过Web前端（或者API）新增一个条目时，它会被添加到数据库的item表里。然后，Zabbix服务器以每分钟一次的频率查询item表中的活动列表，接着将它存储在Zabbix服务器中的缓存里。这就是为什么Zabbix前端所做的任何更改最多需要花费两分钟才能显示在最新的数据段的原因。

服务进程

Zabbix服务器进程是以守护进程（Deamon）运行的。服务器的启动可以通过执行以下命令来完成：

```
1. shell> cd sbin
```

上述命令在大多数的GNU/Linux系统下都可以正常完成。如果是其他系统，你可能要尝试一下命令来运行：

```
1. shell> ./zabbix_server
```

你可以使用Zabbix server下的命令行参数：

1. -c --config <file>	配置文件的绝对路径（默认路径： <code>/etc/zabbix/zabbix_server.conf</code> ）
2. -R --runtime-control <选项>	执行管理能力
3. -h --help	帮助提示
4. -V --version	显示版本号

运行控制台不支持OpenBSD和NetBSD系统。

命令行参数示例：

```
1. shell> zabbix_server -c /usr/local/etc/zabbix_server.conf
2. shell> zabbix_server --help
3. shell> zabbix_server -V
```

runtime control

选项：

Option	Description	Target
config_cache_reload	重新加载配置缓存。忽略当前已加载的缓存。	
housekeeper_execute	启动管家程序。忽略当前正在进行中的管家程序。	
log_level_increase[=<目标>]	增加日志的级别，如果没有指定目标则影响所有的进程。	pid - 进程标识 (1 ~ 65535) 进程类型 - 指定进程的所有类型 (例如, poller) 进程类型, N - 进程类型和编号 (例如, poller, 3)
log_level_decrease[=<目标>]	降低日志的基本，如果没有指定目标则影响所有的进程。	

单一zabbix进程的日志基本改变后，进程的PIIDs的值也会改变，允许的范围为1~65535. 对大用户<进程类型，N>目标选项可更改单个进程的日志级别

例如，使用runtime control重新加载server的配置缓存：

```
1. shell> zabbix_server -c /usr/local/etc/zabbix_server.conf -R config_cache_reload
```

例如，使用runtime control触发管家服务执行：

```
1. shell> zabbix_server -c /usr/local/etc/zabbix_server.conf -R housekeeper_execute
```

例如，使用runtime control改变日志的级别：

```
1. 增加所有进程的日志级别:
2.
3. shell> zabbix_server -c /usr/local/etc/zabbix_server.conf -R log_level_increase
4.
5. 增加第二个poller进程的日志级别:
6.
7. shell> zabbix_server -c /usr/local/etc/zabbix_server.conf -R log_level_increase=poller,2
8.
9. 增加PID为1234的进程日志级别:
10.
11. shell> zabbix_server -c /usr/local/etc/zabbix_server.conf -R log_level_increase=1234
12.
13. 降低http poller进程的日志级别:
14.
15. shell> zabbix_server -c /usr/local/etc/zabbix_server.conf -R log_level_decrease="http poller"
```

进程用户

Zabbix server定义了使用非root用户运行，启动后运行在非root用户的环境下，所以使用非root用户运行server是没有任何问题的。

如果您想尝试root用户运行，它会切换到一个硬编码的用户，您可以参考 [present](#)，您需要修改配置文件中参数'AllowRoot'的值

如果Zabbix server和agent运行在同一台服务器上, 建议您使用不同的用户运行server和agent. 否则, 如果两者都运行相同的用户, 代理可以访问服务器的配置文件, 任何Zabbix管理员级别的用户都可以很容易地检索server的信息, 例如, 数据库密码.

配置文件

请看 [配置文件](#) 有关Zabbix_server的详细配置选项.

启动脚本

当系统启动/关机时启动脚本用来自动启动/停止Zabbix进程, 脚本放在目录misc/init.d下.

支持平台

由于安全要求和服务器关键任务的操作, UNIX系统是唯一能够提供必要性能, 容错和恢复能力的操作系统. Zabbix运转也是市场领先版本.

Zabbix server支持以下平台:

- Linux
- Solaris
- AIX
- HP-UX
- Mac OS X
- FreeBSD
- OpenBSD
- NetBSD
- SCO Open Server
- Tru64/OSF1

Zabbix可以运行在其他类Unix操作系统上.

2 Agent

概况

Zabbix agent部署在监控的目标上，主动监测本地的资源和应用(硬件驱动，内存，处理器统计等)。

Zabbix agent收集本地的操作信息并将数据报告给Zabbix server用于进一步处理。一旦出现异常（比如硬盘空间已满或者有崩溃的服务进程），Zabbix server会主动警告管理员指定机器上的异常。Zabbix agents 的极端高效缘于它可以利用本地系统调用来完成统计数据的收集。

被动 (passive) 和主动 (active) 检查

Zabbix agents可以执行被动和主动两种检查方式。

在[passive check](#) 模式中agent应答数据请求，Zabbix server（或者proxy）询问agent数据，如CPU 的负载情况，然后Zabbix agent回送结果。

[Active checks](#) 处理过程将相对复杂。Agent必须首先从Zabbix sever索取监控项列表以进行独立处理，然后周期性地发送新的值给server。

执行被动或主动检查是通过选择相应的监测项目类型来配置的。[item type](#). Zabbix agent处理监控项类型有'Zabbix agent'和'Zabbix agent (active)'。

支持的平台

Zabbix agent支持以下平台：

- Linux
- IBM AIX
- FreeBSD
- NetBSD
- OpenBSD
- HP-UX
- Mac OS X
- Solaris: 9, 10, 11
- Windows: 支持2000后所有桌面和服务器版。

安装

参见Zabbix agent安装指令 [installation instructions](#)

总体而言32位的Zabbix agent可以在64位系统上运行。但有时也会因为个例而无法正常运行。

运行 agent进程

Zabbix agent运行在被监控主机上。

Zabbix agent在UNIX上以守护进程运行。Agent（独立的守护进程）可以通过执行代码运行：

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

Windows上的Zabbix agent是以Windows服务的形式运行的。您可以在主机上运行Zabbix agent的单个实例或多个实例。单个实例可以使用默认配置文件或命令行中指定的配置文件。在多个实例的情况下，每个agent实例必须有自己独立的配置文件（其中一个实例可以使用默认配置文件）。

以下命令参数可以在Zabbix agent中使用：

参数	描述
UNIX 和 Windows agent	
-c – config <config-file>	配置文件的绝对路径。您可以使用此选项来制定配置文件，而不是使用默认文件。在UNIX中，默认文件是 /usr/local/etc/zabbixagentd.conf 要么通过 compile-time 变量 – sysconfdir 或者 –prefix 来设置在Windows中，默认文件是 c:\zabbix_agentd.conf
-p –print	显示已知监控项并推出。Note：为了同时返回用户参数 user parameter 您必须制定配置文件（如果不是在指定位置的话）。
-t –test <item key>	测试指定监控项并退出。Note：为了同时返回用户参数 user parameter 您必须制定配置文件（如果不是在指定位置的话）。
-h –help	显示帮助信息
-V – version	显示版本号
仅UNIX agent	
-R – runtime-control <option>	执行管理功能。参见 运行时控制 。 runtime control 。
仅Windows agent	
-m – multiple-agents	使用多agent实例（使用 -i, -d, -s, -x）。T为了区分实例的服务名称，每项服务名都会包含来自配置文件里的主机名值。
仅Windows agent (功能)	
-i – install	以服务的形式安装Zabbix Windows agent
-d – uninstall	卸载Zabbix Windows agent服务

-s -start	开始Zabbix Windows agent服务
-x -stop	停止Zabbix Windows agent 服务

有关在Windows上安装和运行Zabbix agent的细节，可以参看 [更多细节 more details](#)。使用命令行参数的特殊例子

- 显示所有内置监控项和它们的值
- 使用指定的配置文件中定义的“mysql.ping”键来测试用户参数
- 在Windows下使用默认路径下的配置文件c:\zabbix_agentd.conf安装一项 Zabbix agent服务
- 使用位于与agent可执行文件同一文件夹中的配置文件zabbix_agentd.conf为Windows安装“Zabbix Agent [Hostname]”服务，并通过从配置文件中的唯一Hostname值扩来为命名。

```
shell> zabbix_agentd -print
```

```
1. shell> zabbix_agentd -t "mysql.ping" -c /etc/zabbix/zabbix_agentd.conf
2. shell> zabbix_agentd.exe -i
3. shell> zabbix_agentd.exe -i -m -c zabbix_agentd.conf
```

运行时控制

使用运行时控制选项，您可以更改agent进程的日志级别。

选项	描述	目标
log_level_increase[=<target>]	增加日志级别。如果没有特别指出，将会赋给所有进程。	
log_level_decrease[=<target>]	降低日志级别。如果没有特别指出，将会赋给所有进程。	

用于更改单独Zabbix agent的日志级别的PID可用范围为1到65535。对于具有较大PID的系统，可以使用<process type, N>目标选项来更改单独进程的日志级别。

Examples :

- 给所有进程增加日志级别
- 给第二监听进程增加日志级别
- 给PID号为1234的进程增加日志级别
- 给所有主动检查进程降低日志级别

```
1. shell> zabbix_agentd -R log_level_increase
2. shell> zabbix_agentd -R log_level_increase=listener,2
3. shell> zabbix_agentd -R log_level_increase=1234
4. shell> zabbix_agentd -R log_level_decrease="active checks"
```

运行时控制当前不支持OpenBSD, NetBSD和Windows。

进程用户

Zabbix agent是unix平台上设计在非root账户下的。它会以其他任何非root用户启动的进程一样的方式运行。所以，您可以使用任意非root用户运行agent，且不会产生任何问题。

如果您在'root'账户下运行，它将切换到硬编码的“zabbix”用户，该用户必须存在于您的系统上。如果您只想以'root'方式运行proxy，您必须在proxy配额文件里修改'AllowRoot'参数。

配置文件

更多配置Zabbix agent的细节，请参阅配置文件选项。[zabbix_agentd](#) 或者[Windows agent](#)。

退出码

在2.2版本前，Zabbix agent 成功退出时退出码为0，异常时退出码为255。从2.2版本开始，Zabbix agent成功退出时退出码为0，异常则退出码为1。

3 Proxy

Overview

Zabbix Proxy是一个可以从一个或多个受监控设备收集监控数据，并将信息发送到Zabbix sever的进程，基本上是代表sever工作的。所有收集的数据都在本地进行缓存，然后传送到proxy所属的Zabbix sever。

部署Proxy是可选的，但是可能非常有益于分散单个Zabbix sever的负载。如果只有proxy收集数据，sever上的进程就会减少CPU消耗和磁盘I / O负载。

Zabbix proxy是完成远程区域、分支机构、没有本地管理员的网络的集中监控的理想解决方案。

Zabbix proxy需要使用独立的数据库。

Zabbix proxy数据库可以使用SQLite, MySQL, PostgreSQL. 使用Oracle或IBM DB2数据库时会有一定风险和限制（例如在低等级发现规则中的 返回值）`return values of low-level discovery rules.`

请参阅：[在分布环境中使用Proxy](#)

Proxy 进程

Zabbix proxy以守护进程的方式运行。Proxy可以通过执行代码运行。

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

您可以使用Zabbix agent下的命令行参数：

1. -c --config <file>	配置文件的绝对路径
2. -R --runtime-control <option>	执行管理功能
3. -h --help	帮助
4. -V --version	显示版本号

运行时控制当前不支持OpenBSD和NetBSD。

命令行参数示例：

```
1. shell> zabbix_proxy -c /usr/local/etc/zabbix_proxy.conf
2. shell> zabbix_proxy --help
3. shell> zabbix_proxy -V
```

运行时控制

运行时控制选项：

选项	描述	目标
	重新加载配置缓存。若当前已被加载，则忽略。激活Zabbix	

	proxy将会连接到Zabbix server并查询配置数据。	
housekeeper_execute	开始管理程序。如果管理程序目前正在运行中，则被忽略。	
log_level_increase[=<target>]	提升日志级别。如果没有特别指出，将会赋给所有进程。	pid - 进程号 (1 到 65535) process type -All 特指类型的进程(例如 poller) process type,N - 进程类型和号码 (例如, poller,3)
log_level_decrease[=<target>]	降低日志级别，如果没有特别指出，将会赋给所有进程。	

用于更改单独Zabbix进程的日志级别的PID允许范围为1到65535。对于具有较大PID的系统，可以使用<process type, N>目标选项来更改单独进程的日志级别。

使用运行时控制来重新加载配置缓存的例子：

```
1. shell> zabbix_proxy -c /usr/local/etc/zabbix_proxy.conf -R config_cache_reload
```

使用运行时控制来触发管家的执行的例子

```
1. shell> zabbix_proxy -c /usr/local/etc/zabbix_proxy.conf -R housekeeper_execute
```

使用运行时控制更改日志级别的示例：

```
1. Increase log level of all processes:
2. shell> zabbix_proxy -c /usr/local/etc/zabbix_proxy.conf -R log_level_increase
3.
4. Increase log level of second poller process:
5. shell> zabbix_proxy -c /usr/local/etc/zabbix_proxy.conf -R log_level_increase=poller,2
6.
7. Increase log level of process with PID 1234:
8. shell> zabbix_proxy -c /usr/local/etc/zabbix_proxy.conf -R log_level_increase=1234
9.
10. Decrease log level of all http poller processes:
11. shell> zabbix_proxy -c /usr/local/etc/zabbix_proxy.conf -R log_level_decrease="http poller"
```

进程用户

Zabbix proxy是设计在非root账户下的。它会以其他任何非root用户启动的进程一样的方式运行。所以，您可以使用任意非root用户运行agent，且不会产生任何问题。

如果您在'root'账户下运行，它将切换到硬编码的“zabbix”用户，该用户必须存在于您的系统上。如果您只想以'root'方式运行proxy，您必须在proxy配额文件里修改'AllowRoot'参数。

配置文件

更多Zabbix proxy配置选项细节，可参见 [配置文件](#)。

支持的平台

Zabbix proxy和Zabbix sever一样，在相同的[支持平台](#) 下运行.

4 Java gateway

概述

.从Zabbix 2.0版本开始，以Zabbix守护进程的形式原生支持监控JMX程序出现了，它被称为Zabbix Java gateway。Zabbix gateway 是用Java语言写成。要查得一台主机特定的JMX计数器值，Zabbix server向Zabbix Java gateway发送请求，后者使用[JMX管理API](#) 去请求远程的有关应用。应用不需要额外安装软件，只需要启动时在命令行指定 `-Dcom.sun.management.jmxremote` 即可。

Java gateway接受来自Zabbix Server或Agent的传入连接，且只能用作“被动proxy”。与Zabbix proxy相反，它也可以从Zabbix agent调用（Zabbix agent不能被链接）。每个Java gateway的访问都直接在Zabbix sever或 proxy配置文件中配置，因此每个Zabbix sever或Zabbix agent只能配置一个Java gateway。如果一台主机具 JMX agent 类型的监控项和其他类型的监控项，则只将JMX agent监控项传递到Java gateway进行检索。

当在Java gateway上的一个监控项值更新了，Zabbix server或agent将连接Java gateway请求查询该值，Java gateway会依次retrieves并传回到server或proxy。同样的，Java gateway不会缓存任何值。

Zabbix sever或proxy具有连接到Java gateway特定类型的进程，由 START_POLLERS 选项控制。在内部，Java gateway启动多个由该选项控制的线程。在sever端，如果连接超过 Timeout 秒，则将终止，但Java gateway可能仍忙于从JMX计数器检索值。为了解决这个问题，由于Zabbix 2.0.15，Zabbix 2.2.10和Zabbix 2.4.5在Java gateway中有TIMEOUT选项，允许为JMX网络操作设置超时。

Zabbix server或agent将尽可能地将请求集中到一个JMX目标（受监控项间隔影响），并将它们发送到单一连接中的Java gateway，以获得更好的性能。建议 StartJavaPollers 小于或等于 START_POLLERS，否则可能导致当连接Java gateway时而Java gateway没有多余的线程进行处理

以下内容将详细讲描述如何获得和运行Zabbix Java gateway，如何配置Zabbix server(或proxy)利用Zabbix Java gateway完成JMX监控，以及如何配置Zabbix GUI里的监控项，以匹配特殊JMX计数器。

4.1 获取Java gateway

有两种方式得到Java gateway，一种是通过Zabbix网站下载Java gateway包，另一种是通过源码编译Java gateway。

4.1.1 通过Zabbix网站下载

Zabbix Java gateway包 (RHEL, Debian, Ubuntu)可在 <http://www.zabbix.com/download.php> 下载。.

4.1.2 通过源码编译

为了编译Java gateway，您需要在运行`./configure`时加上`-enable-java`选项。建议在安装时指定`-prefix`选项而非使用默认的`/usr/local`，因为在安装Java gateway时将创建整个目录树，而并非单一的可执行文件

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

使用make完成Java gateway编译并打包成一个JAR文件。需要注意的是：这一步将会需要javac和jar，因此您需要

保证它们在路径中

```
1. $ make
```

现在您将在src/zabbix_java/bin下得到zabbix-java-gateway-\$VERSION.jar文件。如果您对在指定的目录下使用Java gateway满意，那么您可以完成配置和运行Java gateway，否则，请确保有足够的权限运行make install。

```
1. $ make install
```

4.2 Java gateway 文件预览

不论您怎样获得了Java gateway，您应该在\$PREFIX/sbin/zabbix_java下有shell脚本、JAR文件和配置文件。这些文件都在以下文件中。

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

Java gateway JAR 文件本身。

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

Java gateway依赖 [Logback](#), [SLF4J](#), 和[Android JSON](#) 库。

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

Logback的配置文件

```
1. shutdown.sh
2. startup.sh
```

用于启动和停止Java gateway的快捷脚本

```
1. settings.sh
```

用于启动和关闭的配置文件如上

4.3 配置和运行Java gateway

默认情况下，Java gateway监听10052端口。如果您计划使用不同的端口来运行Java gateway，需要通过setting.sh脚本指定下的端口。请访问[Java gateway 配置文件](#) 获得如何指定该选项以及其他选项。

10052端口并没有在 [IANA注册](#) ..

一旦完成了配置，您可以通过startup脚本来启动Java gateway

```
1. $ ./startup.sh
```

如果您不需要Java gateway，您可以运行shutdown脚本关闭它

```
1. $ ./shutdown.sh
```

与Sever和Proxy不同，Java gateway是轻量级的，不需要数据库的支持

4.4 配置Sever使用Java gateway

现在Java gateway已在运行，接下来您要告诉Zabbix server在哪里找到Zabbix Java gateway。因此你需要在[server配置文件](#)中指定JavaGateway及JavaGateway端口（JavaGatewayPort）。如果JMX应用采用Zabbix agent进行监控的话，您需要在[proxy 配置文件](#) 中配置对应的连接参数。

```
1. JavaGateway=192.168.3.14
2. JavaGatewayPort=10052
```

默认情况下，server并不会产生任何与JMX监控进程。但如果您想使用完成JMX监控，您需要指定Java轮询器的预分支实例数（pre-forked instances），您也可过同类的方式指定常见的轮询器和捕获器。

```
1. StartJavaPollers=5
```

一旦您完成了相关配置，不要忘记重启Sever或Proxy

4.5 Debugging Java gateway

当Java gateway出现问题，或者在前端看到的监控项报错信息不充分时，您也可以通过查看Java gateway日志文件获得更多信息

默认情况下，Java gateway将记录日志到/tmp/zabbix_java.log文件中，日志级别为“info”。若您觉得“info”级别得到的信息并不充分，则需要修改级别为“debug”。你可以通过修改lib/logback.xml将<root>标签更改为“debug”以让日志级别的增加。

```
1. <root level="debug">
2.   <appender-ref ref="FILE" />
3. </root>
```

需要注意的是，与Zabbix server或proxy不同，修改完logback.xml并不需要重启Zabbix Java gateway。修改后的配置将会自动被加载。当您完成了debug，您可以将日志级别改为“info”。

如果您想将日志记录到其他文件或者其他存储媒介，如数据库，按照您的需求调整logback.xml文件即可。想要了解更多请访问 [Logback手册](#)。有时为了方便进行debug，采用控制台应用的方式会比守护进程更为有用。为此，需要注释掉setting.sh脚本中 PID_FILE 变量。一旦没有找到 PID_FILE 参数，startup.sh脚本将直接以控制台应用方式运行。Logback也将使用lib/logback-console.xml文件。不仅仅只是记录到控制台，记录级别也将变更

为“debug”。最后，请注意，由于Java gateway使用SLF4J进行日志记录，您可以通过将适当的JAR文件放在lib文件夹里，来用您所选的框架来替换Logback。更多细节，请访问 [[<http://www.slf4j.org/manual.html> | SLF4J手册]]。

5 Sender

综述

Zabbix sender 是一种命令行应用，它可以将性能数据发送到Zabbix server进行处理。该应用通常用在长时间运行的用户脚本，用于定期发送可用性和性能数据。

运行Zabbix sender

运行 Zabbix UNIX sender的例子：

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

其中：

- z - Zabbix server 主机 (IP 地址也可以使用)
- s - 受监控主机的技术名称(与Zabbix前端注册的相同)
- k - 监控项的值
- o - 要发送的值

包含空格的选项，必须被双引号括起来。

Zabbix sender可以通过一个输入文件发送多个值。更多信息，参阅[Zabbix sender manpage](#)。Zabbix sender支持UTF-8编码的字符串（类UNIX系统和Windows都可以），且不会在文件首有字节顺序标记（BOM）。

Zabbix sender在Windows系统同样也可以运行：

```
1. zabbix_sender.exe [选项]
```

从Zabbix 1.8.4开始，zabbix_sender实时发送场景已经得到改进，现在它可以收集传递多个值，并将它们连续一次性地发送到sever。两个间隔不超过0.2秒的值可以放在同一个堆栈中，但是最大合并时间仍然是1秒。

当非法参数输入到指令配置文件时，Zabbix sender将会终止（不遵从 parameter=value 概念）

6 Get

概述

Zabbix get 是一种命令行应用，它可以用与Zabbix agent进行通信，并从agent那里获取所需的信息。该应用通常被用于Zabbix agent故障排除。

运行 Zabbix get

UNIX下运行Zabbix get，从agent那里获取处理器的负载值的例子：

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

另一个运行Zabbix get 从一个网站上捕获一个字符串的例子：

```
1. shell> cd bin
2. shell> ./zabbix_get -s 192.168.1.1 -p 10050 -k "web.page.regexp[www.zabbix.com,,,\"USA: ([a-zA-Z0-9.-]+)\",,\\"1\"]"
```

注意：监控项值这里包含了空格，所以引号用来为shell标记。 引号并不是监控项值的一部分，他们会自动被shell过滤，不会传给Zabbix agent

Zabbix get 支持以下命令行参数：

1. -s --host <host name or IP>	指定主机名或主机的IP地址.
2. -p --port <port number>	指定主机上运行代理的端口号. 默认端口10050.
3. -I --source-address <IP address>	指定源IP地址.
4. -k --key <item key>	指定需要获取值的监控项.
5. -h --help	帮助提示.
6. -V --version	显示版本号.

更多信息，参阅 Zabbix get manpage Zabbix get手册。[Zabbix get manpage](#)。

Zabbix get在Windows系统同样也可以运行

```
1. zabbix_get.exe [选项]
```

4. 安装

请使用侧边导航栏来访问安装部分的内容。

1 部署Zabbix

概述

部署Zabbix有四种途径：

- 从[分发包](#)进行安装
- 下载最新的源代码归档，并[自行编译](#)
- 从[容器](#)安装
- 下载[虚拟应用](#)

可以在[Zabbix下载页面](#)下载最新的源代码和虚拟应用，该链接提供最新的版本。如需要下载旧版本，可以查看稳定版本下方的链接。

2 安装要求

硬件

内存和磁盘

Zabbix同时需要物理内存和磁盘空间。刚开始使用Zabbix，建议128MB物理内存和256MB可用磁盘空间。然而，具体需要的内存大小和磁盘空间要根据主机数量和监控参数而定。如果你计划对监控的参数进行长期保存，你应该考虑至少在数据库中预留几个GB的空间，以用来保留历史数据。每个Zabbix的守护进程需要与数据库服务器建立多个连接。分配给连接的内存数量，取决于数据库引擎的配置。

你使用的内存越多，你的数据库（也包括Zabbix）工作得越快！

CPU

根据监控参数及选择的数据库引擎，Zabbix，特别是Zabbix数据库，可能需要大量的CPU资源，

其他硬件

如果需要启用短信（SMS）通知功能，需要串行通讯口（serial communication port）和串行GSM调制解调器（serial GSM modem）。USB转串行转接器也同样可以工作。

硬件配置示例

下表是几个硬件配置的示例：

名称	平台	CPU/内存	数据库	监控主机数量
小型	CentOS	虚拟应用	MySQL InnoDB	100
中型	CentOS	2 CPU cores/2GB	MySQL InnoDB	500
大型	RedHat Enterprise Linux	4 CPU cores/8GB	RAID10 MySQL InnoDB or PostgreSQL	>1000
巨大型	RedHat Enterprise Linux	8 CPU cores/16GB	快速RAID10 MySQL InnoDB or PostgreSQL	>10000

具体的配置极其依赖于Active Item数量和轮询频率。如需要进行大规模部署，强烈建议将数据库进行独立部署。

支持平台

由于监控服务器的安全要求及关键任务的特性，UNIX是唯一可以持续提供必要性能、容错性和扩展性的操作系统。Zabbix可以运行在市场上的主流版本。

经测试，Zabbix可运行在下列平台：

- Linux
- IBM AIX
- FreeBSD

- NetBSD
- OpenBSD
- HP-UX
- Mac OS X
- Solaris
- Windows: Windows 2000以后的所有桌面和服务器版本 (只可运行Zabbix agent)

Zabbix可能也可以运行在其他Unix类操作系统。

如果使用加密编译, 那么Zabbix会禁用核心转储(Core dumps); 同时, 如果系统不允许禁用核心转储, 那么Zabbix无法启动。

软件

Zabbix基于先进的Apache Web服务器、领先的数据库引擎和PHP脚本语言进行构建。

数据库管理系统

软件	版本	备注
MySQL	5.0.3或以上	使用MySQL作为Zabbix后端数据库。需要InnoDB引擎。
Oracle	10g或以上	使用Oracle作为Zabbix后端数据库。
PostgreSQL	8.1或以上	使用PostgreSQL作为Zabbix后端数据库。建议使用PostgreSQL 8.3以上的版本。以 提供更好的VACUUM性能 。
SQLite	3.3.5或以上	使用SQLite作为Zabbix后端数据库。
IBM DB2	9.7或以上	使用IBM DB2作为Zabbix后端数据库。

对于IBM DB2的支持仅供测试!

虽然Zabbix proxy可以正常使用SQLite3, 但是不推荐Zabbix server使用SQLite3。自Zabbix 2.4.0起,, Zabbix server和前端同时进行数据库访问, 甚至可能导致数据库中断!

前端

Zabbix前端需要使用下列软件:

软件	版本	备注
Apache	1.3.12或以上	
PHP	5.4.0或以上	
PHP扩展包:		

<i>gd</i>	2.0或以上	PHP GD扩展包必须支持PNG图片 (<i>-with-png-dir</i>), JPEG (<i>-with-jpeg-dir</i>) images and FreeType 2 (<i>-with-freetype-dir</i>).
<i>bcmath</i>		php-bcmath (<i>-enable-bcmath</i>)
<i>ctype</i>		php-ctype (<i>-enable-ctype</i>)
<i>libXML</i>	2.6.15或以上	php-xml or php5-dom, 由分发者提供单独的部署包。
<i>xmlreader</i>		php-xmlreader, 由分发者提供单独的部署包。
<i>xmlwriter</i>		php-xmlwriter, 由分发者提供单独的部署包。
<i>session</i>		php-session, 由分发者提供单独的部署包。
<i>sockets</i>		php-net-socket (<i>-enable-sockets</i>). 用户脚本支持所需要的组件。
<i>mbstring</i>		php-mbstring (<i>-enable-mbstring</i>)
<i>gettext</i>		php-gettext (<i>-with-gettext</i>). 用于翻译的运行。
<i>ldap</i>		php-ldap. 只有当在前端使用LDAP认证时才需要。
<i>ibm_db2</i>		使用IBM DB2作为Zabbix后端数据库所需要的组件。
<i>mysqli</i>		使用MySQL作为Zabbix后端数据库所需要的组件。
<i>oci8</i>		使用Oracle作为Zabbix后端数据库所需要的组件。
<i>pgsql</i>		使用PostgreSQL作为Zabbix后端数据库所需要的组件。
<i>sqlite3</i>		使用SQLite作为Zabbix后端数据库所需要的组件。

Zabbix 可能也可以运行在旧版本的Apache, MySQL, Oracle, 和PostgreSQL上。

如果需要使用默认DejaVu以外的字体, 可能会需要PHP的[imagerotate](#)功能。如果缺少这个功能, 在监控 (Monitoring) → 概要 (Overview) 的标题栏及其他位置, 字体可能无法正常地显示。该功能只用在使用 bundled GD编译PHP时才可用。在Debian和某些分发版本中, 这个问题不存在。

客户端浏览器

必须启用Cookies和Java Script功能。支持最新版本的Google Chrome, Mozilla Firefox, Microsoft Internet Explorer和Opera。其他浏览器(如Apple Safari, Konqueror)可能也支持Zabbix。

服务器

要求	描述
<i>OpenIPMI</i>	支持IPMI功能所需要的组件。
<i>libssh2</i>	支持SSH功能所需要的组件。需要1.0或以上版本。
<i>fping</i>	支持 ICMP ping功能 所需要的组件。
<i>libcurl</i>	支持WEB监控, VMware监控及SMTP认证所需要的组件。对于SMTP认证, 需要7.20.0或以上版本。
<i>libiksemel</i>	支持Jabber功能所需要的组件。
<i>libxml2</i>	支持VMware监控所需要的组件。
<i>net-snmp</i>	支持SNMP监控所需要的组件。

Java网关 (Java gateway)

如果你从源代码库或者归档中获得Zabbix，那么必要的依赖关系已经包含在了源代码树中了。

如果你从分发包中获得Zabbix，那么必要的依赖关系已经由封装系统提供了。

在上述两种情况下，我们可以准备部署软件了，而不需要下载额外的依赖包。

然而，如果你希望使用这些依赖关系所涉及到的安装包的其他版本（比如你正在为其他Linux分发版准备安装包），可参考下表，这些库的版本可以正常运行Java gateway。这些库的其他版本可能也可以正常运行Zabbix。

下表列出了目前在源代码中与Java gateway捆绑的JAR文件版本：

库	许可	网站
<i>logback-core-0.9.27.jar</i>	EPL 1.0, LGPL 2.1	http://logback.qos.ch/
<i>logback-classic-0.9.27.jar</i>	EPL 1.0, LGPL 2.1	http://logback.qos.ch/
<i>slf4j-api-1.6.1.jar</i>	MIT License	http://www.slf4j.org/
<i>android-json-4.3_r3.1.jar</i>	Apache License 2.0	https://android.googlesource.com/platform/libcore/+/master/

Java gateway编译和运行在Java 1.6及以上版本。如需要对Java gateway预编译版本进行编译，建议使用Java 1.6进行编译，从而保证至少可以在一个版本的Java上正常运行。

数据库容量

Zabbix配置数据需要使用固定的磁盘空间，而且这个空间不会过多增长。

Zabbix数据库容量主要依赖于下列这些参数，这些参数也决定了存储历史数据所需要的空间：

- 每秒处理值的数量 (Number of processed values per second)

这个参数是指每秒钟Zabbix server收到的新值数量的平均数。比如，如果我们有3000个监控项 (item)，监控周期是60s，经计算所得，每秒处理值的数量为 $3000/60 = 50$ 。

这意味着每秒钟有50个新值写入Zabbix数据库。

- 历史 (History) 数据的回收清理设置 (Housekeeper)

Zabbix会在一个固定周期内保存收到的值。正常情况下保留数周或者数月。每一个新收到的值会占用一定数量的磁盘空间以存放数据和索引。

所以，如果我们每秒钟收到50个值，且希望保留30天的历史数据，值的总数将大约在 $(30 * 243600) * 50 = 129,600,000$ ，即大约130M个值。

根据所使用的数据库引擎，以及收到值的类型【浮点 (floats)，整型 (integers)，字符串 (strings)，日志

文件 (log files) 等】，单个值的磁盘使用量从40字节到数百个字节不等。一般而言，数值型 (Numeric) 的监控项占用大约90字节。按之前的例子，这意味着130M个值需要占用 $130M \cdot 90 \text{ bytes} = *10.9GB$ 的磁盘空间。

文本 (text) / 日志 (log) 类型的监控项值的大小无法准确地预测，但你可以按每个值大约500字节来计算。

- 趋势 (Trends) 数据的回收清理设置 (Housekeeper)

Zabbix为trends表中的每个监控项的值，保留一组数据：一个小时的最大值 / 最小值 / 平均值 / 数量。这些数据用于趋势图表和历史图表的展现。用户无法自定义这一小时的保留周期。

根据数据库的类型，Zabbix数据库需要为每组值总共占用约90字节的空间。如果你需要保留趋势数据5年，那么3000个监控项值，每年需要 $3000 \times 24 \times 3600 \times 90 = 2.2GB$ 的空间，即5年需要 $*11GB$ 的空间。

- 事件 (Events) 数据的回收清理设置 (Housekeeper)

每个Zabbix事件需要大约170字节的磁盘空间。很难估计Zabbix每天生成的事件数量。最糟糕的情况下，我们可能需要假设Zabbix每秒会生成一个事件。

这意味着，如果我们需要保留3年的事件，需要 $3 \times 365 \times 24 \times 3600 \times 170 = *15GB$ 的磁盘空间。

下表列出了用于计算Zabbix系统所需磁盘空间的计算公式：

范围	所需磁盘空间的计算公式 (单位: 字节)
Zabbix配置文件	固定大小。一般10MB或更少。
历史 (History)	$\text{days}(\text{items}/\text{refresh rate}) \cdot 24 \cdot 3600 \text{bytes}$ items : 监控项数量 days : 保留历史数据的天数 refresh rate : 监控项平均轮询时间 bytes : 保留单个值所需要占用的字节数，依赖于数据库引擎，一般大约90字节。
趋势 (Trends)	$\text{days}(\text{items}/3600) \cdot 24 \cdot 3600 \text{bytes}$ items : 监控项数量 days : 保留趋势数据的天数 bytes : 保留单个趋势数据所需要占用的字节数，依赖于数据库引擎，一般大约90字节。
事件 (Events)	$\text{day}(\text{events}/24 \cdot 3600) \text{bytes}$ events : 每秒事件数。最糟糕的情况下，每秒一 (1) 个事件。 days : 保留事件数据的天数 bytes : 保留单个事件所需要占用的字节数，依赖于数据库引擎，一般大约90字节。

根据现实环境中使用的MySQL后端数据库的统计，数值型 (Numeric) 监控项的值平均占用约90个字节，事件 (Events) 平均占用约170个字节。

因此，所需要的磁盘总空间按下列方法计算：配置 (Configuration) + 历史 (History) + 趋势 (Trends) + 事件 (Events) 安装完Zabbix，磁盘空间不会立即被分配。数据库大小根据回收清理 (housekeeper) 设置，在某些时间点增长或停止增长。

时钟同步

对于Zabbix稳定运行而言，服务获得精确的系统时间是非常重要的。[ntp](#) 是一个最流行的用于同步主机和其他服务器之间的时间的后台程序。对于所有运行Zabbix组件的系统，强烈建议这些系统的时间保持同步。

如果时间未同步，Zabbix将在建立数据连接之后，根据得到的客户端 / 服务器的时间戳，将获得值的时间戳转化为Zabbix server的时间，并且会根据客户端-服务器的时间差对获得值的时间戳进行调整。为了保持简单，并且避免可能的并发问题出现，网络延迟会被忽略。因此，通过主动连接(active agent, active proxy, sender)获得的时间戳数据包含网络延迟，通过被动连接(passive proxy)获得的数据已经减去了网络延迟。所有其他检查服务

器花费的时间和它们的时间戳不调整。



3 从部署包安装

从分发包安装

大多数主流的操作系统分发版本都提供了Zabbix部署包。你可以使用这些部署包安装Zabbix。

操作系统分发版本的源码库中可能会缺少Zabbix的最新版本。

从Zabbix官方的源码库安装

Zabbix SIA为Red Hat Enterprise Linux, Debian和Ubuntu LTS系统提供官方RPM和DEB部署包。

可通过repo.zabbix.com下载部署包文件。该服务器同时提供yum和apt源码库。这里提供从部署包安装Zabbix的详细教程。

Red Hat Enterprise Linux / CentOS

支持版本: RHEL 7, Oracle Linux 7, CentOS 7

一些组件的部署包(如agent, proxy等),同时也支持RHEL 5和RHEL 6。

安装源码库配置部署包

安装源码库配置部署包。这个部署包包含了yum配置文件。

```
1. # rpm -ivh http://repo.zabbix.com/zabbix/3.4/rhel/7/x86_64/zabbix-release-3.4-1.el7.noarch.rpm
```

安装Zabbix部署包

安装Zabbix部署包。以下是使用Mysql数据库安装Zabbix server、WEB前端的示例。

Zabbix官方源码库也提供fping, iksemel, libssh2部署包这些包位于*non-supported*目录。

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

只安装Zabbix Agent的示例。

```
1. # yum install zabbix-agent
```

安装初始化数据库

在MySQL上安装Zabbix数据库和用户,请参看下列指导步骤。[MySQL数据库创建脚本](#)。

然后导入初始架构(Schema)和数据。

```
1. # cd /usr/share/doc/zabbix-server-mysql-3.4.0
2. # zcat create.sql.gz | mysql -uroot zabbix
```

启动Zabbix Server进程

在zabbix_server.conf中编辑数据库配置

```
1. # vi /etc/zabbix/zabbix_server.conf
2. DBHost=localhost
3. DBName=zabbix
4. DBUser=zabbix
5. DBPassword=zabbix
```

启动Zabbix Server进程

```
1. # systemctl start zabbix-server
```

编辑Zabbix前端的PHP配置

Zabbix前端的Apache配置文件位于 /etc/httpd/conf.d/zabbix.conf 。一些PHP设置已经完成了配置。

```
1. php_value max_execution_time 300
2. php_value memory_limit 128M
3. php_value post_max_size 16M
4. php_value upload_max_filesize 2M
5. php_value max_input_time 300
6. php_value always_populate_raw_post_data -1
7. # php_value date.timezone Europe/Riga
```

依据所在时区，你可以取消 “date.timezone” 设置的注释，并正确配置它。在配置文件更改后，需要重启Apache Web服务器。

```
1. # systemctl start httpd
```

Zabbix前端可以在浏览器中通过 <http://zabbix-frontend-hostname/zabbix> 进行访问。默认的用户名 / 密码为 Admin/zabbix。

Debian / Ubuntu

支持版本：Debian 7 (Wheezy) and 8 (Jessie), Ubuntu 14.04 LTS (Trusty Tahr), 16.04 LTS (Xenial Xerus)

安装源码库配置部署包

安装源码库配置部署包。这个部署包包含了apt配置文件。

在 Debian 7 上安装 Zabbix 3.4:

```
1. # wget http://repo.zabbix.com/zabbix/3.4/debian/pool/main/z/zabbix-release/zabbix-release_3.4-1+wheezy_all.deb
2. # dpkg -i zabbix-release_3.4-1+wheezy_all.deb
3. # apt-get update
```

在 Debian 8 上安装 Zabbix 3.4:

```
1. # wget http://repo.zabbix.com/zabbix/3.4/debian/pool/main/z/zabbix-release/zabbix-release_3.4-1+jessie_all.deb
2. # dpkg -i zabbix-release_3.4-1+jessie_all.deb
3. # apt-get update
```

在 Ubuntu 14.04 LTS 上安装 Zabbix 3.4:

```
1. # wget http://repo.zabbix.com/zabbix/3.4/ubuntu/pool/main/z/zabbix-release/zabbix-release_3.4-1+trusty_all.deb
2. # dpkg -i zabbix-release_3.4-1+trusty_all.deb
3. # apt-get update
```

在 Ubuntu 16.04 LTS 上安装 Zabbix 3.4:

```
1. # wget http://repo.zabbix.com/zabbix/3.4/ubuntu/pool/main/z/zabbix-release/zabbix-release_3.4-1+xenial_all.deb
2. # dpkg -i zabbix-release_3.4-1+xenial_all.deb
3. # apt-get update
```

安装Zabbix部署包

使用mysql数据库安装Zabbix server、WEB前端的示例。

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

只安装Zabbix Agent的示例。

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

安装初始化数据库

在MySQL上安装Zabbix数据库和用户，请参考下列指导步骤。 [MySQL数据库创建脚本](#)。

然后导入初始架构（Schema）和数据

```
1. # cd /usr/share/doc/zabbix-server-mysql
2. # zcat create.sql.gz | mysql -uroot zabbix
```

启动Zabbix Server进程

在zabbix_server.conf中编辑数据库配置

```
1. # vi /etc/zabbix/zabbix_server.conf
2. DBHost=localhost
3. DBName=zabbix
4. DBUser=zabbix
5. DBPassword=zabbix
```

启动Zabbix Server进程

```
1. # service zabbix-server start
```

编辑Zabbix前端的PHP配置

Zabbix前端的Apache配置文件位于 `/etc/apache2/conf.d/zabbix` 或者 `/etc/apache2/conf-enabled/zabbix.conf`。一些PHP设置已经完成了配置。

```
1. php_value max_execution_time 300
2. php_value memory_limit 128M
3. php_value post_max_size 16M
4. php_value upload_max_filesize 2M
5. php_value max_input_time 300
6. php_value always_populate_raw_post_data -1
7. # php_value date.timezone Europe/Riga
```

依据所在时区，你可以取消 “`date.timezone`” 设置的注释，并正确配置它。在配置文件更改后，需要重启Apache Web服务器。

```
1. # service apache2 restart
```

Zabbix前端可以在浏览器中通过 <http://zabbix-frontend-hostname/zabbix> 进行访问。默认的用户名 / 密码为 Admin/zabbix。

4 从源代码安装

你可以通过源代码编译，获得最新版本的Zabbix。

本页面提供从源代码安装Zabbix的详细教程。

1 安装Zabbix守护进程

1 下载源代码归档

前往 [Zabbix download page](#) 并下载源代码归档。当下载完毕后，执行下列命令解压缩源代码：

```
1. $ tar -zxvf zabbix-3.4.0.tar.gz
```

在命令行中输入正确的Zabbix版本号。版本号必须和下载的归档包文件名相同。

2 创建用户账户

对于所有Zabbix的守护进程，需要一个无特权的用户。如果Zabbix守护进程以一个无特权的用户账户启动，那么它会使用该用户运行。

然而，如果一个守护进程以‘root’用户启动，它会切换为‘zabbix’用户账户，且这个用户必须存在。在Linux系统中，可以使用下面命令建立一个用户（该用户属于自己的用户组，“zabbix”）：

```
1. groupadd zabbix
2. useradd -g zabbix zabbix
```

对于Zabbix前端的安装，不需要使用单独的用户账户。

如果Zabbix `server` 和 `agent` 运行在同一台计算机上，建议使用不同的账户运行Server和Agent。否则，如果两个进程使用了同一个用户，Agent就可以访问Server的配置文件，并可轻易地读取Zabbix中任何管理员级别的用户，比如数据库密码。

使用 `root`，`bin` 或其他特殊权限的账户运行Zabbix是一个安全风险。

3 创建Zabbix数据库

对于Zabbix `server` 和 `proxy` 守护进程以及Zabbix前端，都需要连接到一个数据库。Zabbix `agent` 不需要数据库的支持。

`SQL 脚本` 用于创建数据库架构（schema）并插入数据集（dataset）。Zabbix `proxy` 数据库只需要数据库架构（schema），而Zabbix `server` 数据库在建立数据库架构（schema）后，还需要数据集（dataset）。

建立Zabbix数据库后，可以开始对Zabbix进行编译。

4 配置源代码

当配置Zabbix `server` 或者 `proxy` 的源代码时，需要指定所使用的数据库类型。每个Zabbix `server` 或者 `proxy` 进程在同一时间内只能使用一种数据库类型。

如果需要查看所有支持的配置选项，可在解压缩后的Zabbix源代码目录中，运行下列命令：

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

如果需要为Zabbix server和agent配置源代码，可以按下列格式运行命令：

```
1. ./configure --enable-server --enable-agent --with-mysql --enable-ipv6 --with-net-snmp --with-libcurl --with-libxml2
```

自Zabbix 3.0.0版本起，SMTP认证需要`--with-libcurl` 配置选项，同时要求curl 7.20.0或者更改版本。\\自Zabbix 2.2.0版本起，虚拟机监控需`--with-libcurl` 和`--with-libxml2` 配置选项。

为Zabbix server配置源代码(以使用PostgreSQL为例)，你可以运行下列命令：

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

为Zabbix proxy配置源代码(以使用SQLite为例)，你可以运行下列命令：

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

为Zabbix Agent配置源代码，你可以运行下列命令：

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

你可以使用`--enable-static` 开关静态连接类库。如果你打算在不同的服务器之间分发已编译的二进制文件，你必须使用这个标签以使得这些二进制文件在不依赖于所需要的类库的情况下也可常运行。请注意`--enable-static` 在 Solaris 系统下无效。

不建议在搭建Zabbix server时使用`--enable-static` 选项。

为了静态搭建Zabbix server，你必须拥有每个所需的外部类库的静态版本。配置脚本中不提供这些类库的严格检查。

命令行工具`zabbix_get` 和`zabbix_sender` 只有在`--enable-agent` 选项启用时才会被编译。

如果MySQL客户端类库不再默认的位置，需要在MySQL的配置文件中添加可选路径`--with-mysql=/<文件路径>/mysql_config`。

这可以有效解决，一个系统上安装了多个版本的MySQL或者MariaDB的情况。

使用`--with-ibm-db2` 开关以指定CLI API的位置。使用`--with-oracle` 开关以指定OCI API的位置。

如需要使用加密，可以参照[使用加密方式编译Zabbix](#)。

5 安装组件

如果从SVN安装，需要先运行下列命令：

```
$ make dbschema
```

```
1. make install
```

这一步需要使用一个拥有足够权限的用户来运行（一般而言如 'root'，或者使用 `sudo`）。

运行 `make install` 将使用在 /usr/local/sbin 下的守护进程二进制文件 (zabbix_server, zabbix_agentd, zabbix_proxy) 和在 /usr/local/bin 下的客户端二进制文件进行默认安装。

如需要指定 /usr/local 以外的位置，可在之前的配置源代码的步骤中使用 `--prefix` 开关，比如 `--prefix=/home/zabbix`。在这个案例中，守护进程二进制文件会被安装在 `<prefix>/sbin` 下，工具会安装在 `<prefix>/bin` 下。帮助文件会安装在 `<prefix>/share` 下。

6 查看和编辑配置文件

- 在下列路径编辑 Zabbix Agent 的配置文件 `/usr/local/etc/zabbix_agentd.conf`

你需要配置为每台安装了 `zabbix_agentd` 的主机配置这个文件。

你必须在这个文件中指定 Zabbix server 的 IP 地址。从其他主机发起的请求会被拒绝。

- 在下列路径编辑 Zabbix server 的配置文件 `/usr/local/etc/zabbix_server.conf`

你必须指定数据库名称，用户名和密码（如果使用的话）。

如果使用 SQLite，必须指定数据库文件的全路径。数据库用户名和密码不是必填项。

如果进行小规模部署（最多 10 台被监控主机），其余的参数你可以使用默认设置。如果你需要最大化 Zabbix server（或者 Zabbix proxy）的性能，你需要更改其他默认参数。可参考 [性能调优](#) 章节以了解详细信息。

- 如果你已经安装了 Zabbix proxy，可以在下列路径编辑 proxy 的配置文件 `/usr/local/etc/zabbix_proxy.conf`

你必须指定 server 的 IP 地址和 proxy 的主机名称（必须被 server 识别），同时也需要指定数据库名称，用户名和密码（如果使用的话）。

如果使用 SQLite，必须指定数据库文件的全路径。数据库用户名和密码不是必填项。

7 启动守护进程

在 Zabbix server 端启动 `zabbix_server`。

```
1. shell> zabbix_server
```

确认你的系统允许分配 36MB（可能略多一些）的共享内存。否则 Zabbix server 可能无法启动。你会在 Zabbix server 的日志文件中看到“Cannot allocate shared memory for <type of cache>.” 的提示。这可能在 FreeBSD, Solaris 8 上发生。查看本页底部的[“查看其他”](#) 章节，以寻找如何配置共享的方法。

在所有的被监控机器上启动 `zabbix_agentd`。

```
1. shell> zabbix_agentd
```

确认你的系统允许分配 2MB 的共享内存，否则 agent 可能无法启动。你会在 agent 的日志文件中看到“Cannot allocate shared memory for collector.” 的提示。这可能发生在 Solaris 8 上。

如果你已经安装了zabbix proxy，启动zabbix_proxy。

```
1. shell> zabbix_proxy
```

2 安装Zabbix web界面

复制PHP文件

Zabbix前端使用PHP写的，所以必须运行在PHP支持的Web服务器上。只需要简单的从frontends/php路径下复制PHP文件到Web服务器的HTML文件目录，就可以完成安装。

Apache Web服务器的HTML文件目录的一般包括：

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

建议使用子目录替代HTML根目录。可以使用下列命令，以创建一个子目录并复制Zabbix的前端文件到这个目录下（注意替换为实际的目录）：

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

如果准备从SVN安装英语以外的语言，你必须生成翻译文件。可以运行下列命令：

```
1. locale/make_mo.sh
```

需要gettext安装包的 `msgfmt` 组件。

另外，使用英语以外的语言，需要在Web服务器上安装该语言对应的locale。参见“用户文件”页面中的“[查看其他](#)”板块，以寻找如何安装它（如果需要的话）。

安装前端

第一步

在你的浏览器中，打开Zabbix URL: <http://<服务器IP或主机名>/zabbix>

你可以看到前端安装向导的第一个页面。



第二步

确认满足所有的软件安装前置条件。



前置条件	最低要求	描述
PHP 版本	5.4.0	
PHP <code>memory_limit</code> 选项	128MB	位于 <code>php.ini:memorylimit = 128M</code>
<code>_PHP_post_max_size</code> 选项	16MB	位于 <code>php.ini:postmax_size = 16M</code>
<code>_PHP_upload_max_filesize</code> 选项	2MB	位于 <code>php.ini:uploadmax_filesize = 2M</code>
<code>_PHP_max_execution_time</code> 选项	300 秒 (值允许为0和-1)	位于 <code>php.ini:maxexecution_time = 300</code>
<code>_PHP_max_input_time</code> 选项	300 秒 (值允许为0和-1)	位于 <code>php.ini:maxinput_time = 300</code>
<code>_PHP session.auto_start</code> 选项	必须禁用	In <code>php.ini:session.autostart = 0</code>
数据库支持	以下任选其一： IBM DB2, MySQL, Oracle, PostgreSQL, SQLite	必须安装下列模块中的一种： <code>ibm_db2</code> , <code>mysql</code> , <code>oci8</code> , <code>pgsql</code> , <code>sqlite3</code>
<code>_bcmath</code>		<code>php-bcmath</code>
<code>mbstring</code>		<code>php-mbstring</code>
PHP <code>mbstring.func_overload</code> 选项	必须禁用	位于 <code>php.ini:mbstring.funcoverload = 0</code>
<code>_PHP always_populate_raw_post_data</code> 选项	必须禁用	只适用于 PHP 5.6.0 或更高的版本。位于 <code>php.ini:alwayspopulate_raw_post_data = -1</code>
<code>_sockets</code>		<code>php-net-socket</code> . 用于支持用户脚本。
<code>gd</code>	2.0或更高	<code>php-gd</code> . PHP GD 扩展必须支持PNG影像。 (<code>-with-png-dir</code>), JPEG (<code>-with-jpeg-dir</code>) 影像和FreeType 2 (<code>-with-freetype-dir</code>).
<code>libxml</code>	2.6.15	<code>php-xml</code> 或者 <code>php5-dom</code>
<code>xmlwriter</code>		<code>php-xmlwriter</code>
<code>xmlreader</code>		<code>php-xmlreader</code>
<code>ctype</code>		<code>php-ctype</code>
<code>session</code>		<code>php-session</code>
<code>gettext</code>		<code>php-gettext</code> 自Zabbix 2.2.1期, PHP <code>gettext</code> 扩展不是安装Zabbix的强制性要求。如果 <code>gettext</code> 没有安装, 前端也可以照常运行, 但翻译将不可用。

可选的前置条件也会在列表中显示。不满足的可选前置条件将会用橙色显示, 同时以标示为*Warning*状态。即使存在不满足的可选前置条件, 安装仍可以继续进行。

如果需要更改Apache的用户或者用户组, 需要验证对会话目录的权限。否则Zabbix安装可能无法继续。

第三步

输入连接数据库的详细信息。Zabbix数据库必须是已经创建好的。



第四步

输入Zabbix Server的详细信息。



第五步

检查设置信息。



第六步

下载配置文件，将它放置在Web服务器HTML文档子目录（即你复制Zabbix PHP文件的目录）的conf/路径下。



如果Web服务器用户对conf/目录有写入权限，配置文件将会自动保存，并且直接继续下一步操作。

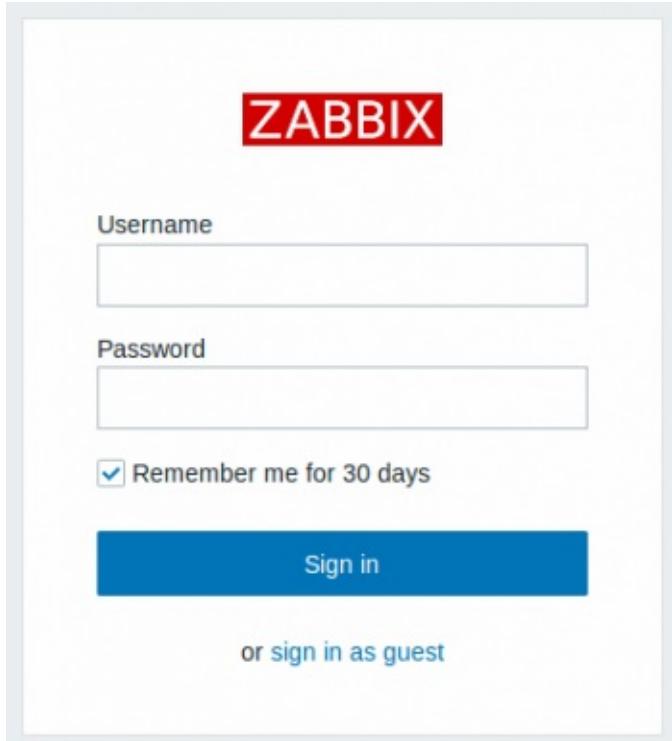
第七步

完成安装。



第八步

Zabbix前端已经就绪！默认的用户名是**Admin**，密码是**zabbix**。



前往 [开始使用Zabbix](#).

查看其他

- [如何为Zabbix守护进程配置共享内存](#)

5 从容器安装

Docker

Zabbix为每个Zabbix组件都提供了Docker镜像，以加速便携（portable）和自给自足（self-sufficient）的容器部署和升级过程。

Zabbix组件支持MySQL和PostgreSQL数据库，支持Apache2和Nginx作为Web服务器。这些镜像被分成多个不同的镜像。

Docker base images

Zabbix组件提供了Ubuntu和Alpine Linux的基础镜像：

镜像	版本
alpine	latest
ubuntu	trusty

如果基础镜像升级了，所有的镜像被配置为重建成最新版本的镜像。

Docker源文件

通过github.com上的Zabbix 官方源代码库，任何人都可以跟踪Docker文件的变更记录。基于官方Docker文件，你可以制作分支项目或者你自己的镜像

结构

所有Zabbix组件都下列Docker源码库中提供：

- Zabbix agent - [zabbix/zabbix-agent](https://github.com/zabbix/zabbix-agent)
- Zabbix server
 - Zabbix server (支持MySQL数据库) - [zabbix/zabbix-server-mysql](https://github.com/zabbix/zabbix-server-mysql)
 - Zabbix server (支持PostgreSQL数据库) - [zabbix/zabbix-server-pgsql](https://github.com/zabbix/zabbix-server-pgsql)
- Zabbix web接口
 - 基于Apache2 web服务器及支持MySQL数据库的Zabbix web接口 - [zabbix/zabbix-web-apache-mysql](https://github.com/zabbix/zabbix-web-apache-mysql)
 - 基于Nginx web服务器及支持MySQL数据库的Zabbix web接口 - [zabbix/zabbix-web-nginx-mysql](https://github.com/zabbix/zabbix-web-nginx-mysql)
 - 基于Nginx web服务器及支持PostgreSQL数据库的Zabbix web接口 - [zabbix/zabbix-web-nginx-pgsql](https://github.com/zabbix/zabbix-web-nginx-pgsql)

- Zabbix proxy
 - Zabbix proxy (支持SQLite3数据库) - [zabbix/zabbix-proxy-sqlite3](#)
 - Zabbix proxy (支持MySQL数据库) - [zabbix/zabbix-proxy-mysql](#)
- Zabbix Java Gateway - [zabbix/zabbix-java-gateway](#)

另外，对于SNMP trap的支持，这里只基于Ubuntu Trusty提供额外的源代码库([zabbix/zabbix-snmptraps](#))。这可以关联Zabbix server和Zabbix proxy使用。

版本

Zabbix组件的每个源码库都包含了下列标签：

- `latest` - 基于Alpine Linux的最新稳定版本的Zabbix组件镜像
- `alpine-latest` - 基于Alpine Linux的最新稳定版本的Zabbix组件镜像
- `ubuntu-latest` - 基于Ubuntu Linux的最新稳定版本的Zabbix组件镜像
- `alpine-3.2-latest` - 基于Alpine Linux的Zabbix 3.2最新次要版本的组件镜像
- `ubuntu-3.2-latest` - 基于Ubuntu Linux的Zabbix 3.2最新次要版本的组件镜像
- `alpine-3.2.` - 基于Alpine Linux的Zabbix 3.2不同次要版本的组件镜像，代表具体的次要版本
- `ubuntu-3.2.` - 基于Ubuntu Linux的Zabbix 3.2不同次要版本的组件镜像，代表具体的次要版本

使用方法

环境变量

所有Zabbix组件的镜像提供了环境变量用以控制配置。这些环境变量在每个组件的源码库中列出。这些环境变量作为Zabbix配置文件的选项，但有不同的命名方法。比如，Zabbix server配置文件中的 `ZBX_LOGSLOWQUERIES` 和Zabbix proxy配置文件中的 `LogSlowQueries` 是同一个配置选项。

一些配置选项不允许更改。比如：`PIDFile` 和 `LogType`。

一些组件存在特定的环境变量，而这些环境变量可能在Zabbix官方的配置文件中不存在：

变量	组件	描述
<code>DBSERVER_HOST</code>	ServerProxyWeb 接口	这个变量指的是MySQL或者PostgreSQL服务器的IP或者DNS名称。默认情况下，这个值为 <code>mysql-server</code> 或者 <code>postgres-server</code> ，依据使用MySQL或者PostgreSQL而定。
<code>DB_SERVER_PORT</code>	ServerProxyWeb 接口	这个变量指的是MySQL或者PostgreSQL服务器的端口。默认情况下，这个值为 '3306' 或者 '5432'，依据使用MySQL或者PostgreSQL而定。
<code>MYSQL_USER</code>	ServerProxyWeb 接口	MySQL数据库用户。默认情况下，这个值为 'zabbix'。
<code>MYSQL_PASSWORD</code>	ServerProxyWeb	MySQL数据库密码。默认情况下，这个值为 'zabbix'。

<code>MYSQL_PASSWORD</code>	接口	MySQL数据库密码。默认情况下，这个值为'zabbix'。
<code>MYSQL_DATABASE</code>	ServerProxyWeb 接口	Zabbix数据库名称。默认情况下，Zabbix server上的这个值为'zabbix'，Zabbix proxy上的这个值为'zabbix_proxy'。
<code>POSTGRES_USER</code>	ServerWeb 接口	PostgreSQL数据库用户。默认情况下，这个值为'zabbix'。
<code>POSTGRES_PASSWORD</code>	ServerWeb 接口	PostgreSQL数据库密码。默认情况下，这个值为'zabbix'。
<code>POSTGRES_DB</code>	ServerWeb 接口	Zabbix数据库名称。默认情况下，Zabbix server上的这个值为'zabbix'，Zabbix proxy上的这个值为'zabbix_proxy'。
<code>TZ</code>	Web 接口	PHP格式中的时区。所有支持的时区列表，可参见 php.net 。默认情况下，这个值为'Europe/Riga'。
<code>ZBX_SERVER_NAME</code>	Web 接口	Web界面右上角显示的Zabbix安装名称。默认情况下，这个值为'Zabbix Docker'。
<code>ZBX_JAVAGATEWAY_ENABLE</code>	ServerProxy	是否启用Zabbix Java gateway以收集Java相关检查数据。默认情况下，这个值为"false"。
<code>ZBX_ENABLE_SNMP_TRAPS</code>	ServerProxy	是否启用SNMP trap功能。这要求 <code>zabbix-snmptraps</code> 实例，并且共享卷/var/lib/zabbix/snmptraps_ 给 Zabbix server或Zabbix proxy。

卷 (Volumes)

镜像允许使用一些挂载点。根据Zabbix组件的类型，这些挂载点各不相同：

卷	描述
Zabbix agent	
<code>/etc/zabbix/zabbix_agentd.d</code>	这个卷允许包含*.conf文件，并使用 <code>UserParameter</code> 功能以扩展 Zabbix agent。
<code>/var/lib/zabbix/modules</code>	这个卷允许加载额外的模块，通过 <code>LoadModule</code> 功能以扩展 Zabbix agent。
<code>/var/lib/zabbix/enc</code>	这个卷用于存放TLS相关文件。这些文件名指定指定使用 <code>ZBXTLSCAFILE</code> ， <code>ZBX_TLSCRLFILE</code> ， <code>ZBX_TLSKEY_FILE</code> 和 <code>ZBX_TLSPSKFILE</code> 环境变量。
Zabbix server	
<code>/usr/lib/zabbix/alertscripts</code>	这个卷用于自定义告警脚本。即 <code>zabbix_server.conf</code> 中 <code>AlertScriptsPath</code> 参数。
<code>/usr/lib/zabbix/externalscripts</code>	这个卷用于外部检查。即 <code>zabbix_server.conf</code> 中 <code>ExternalScripts</code> 参数。
<code>/var/lib/zabbix/modules</code>	这个卷允许加载额外的模块，通过 <code>LoadModule</code> 功能以扩展 Zabbix server。
<code>/var/lib/zabbix/enc</code>	这个卷用于存放TLS相关文件。这些文件名指定指定使用 <code>ZBXTLSCAFILE</code> ， <code>ZBX_TLSCRLFILE</code> ， <code>ZBX_TLSKEY_FILE</code> 和 <code>ZBX_TLSPSKFILE</code> 环境变量。
<code>/var/lib/zabbix/ssl/certs</code>	这个卷用于存放用于客户端认证的SSL客户端证书文件。即 <code>zabbix_server.conf</code> 中 <code>SSLCertLocation</code> 参数。
<code>/var/lib/zabbix/ssl/keys</code>	这个卷用于存放用于客户端认证的SSL私钥文件。即 <code>zabbix_server.conf</code> 中的 <code>SSLKeyLocation</code> 参数。

/var/lib/zabbix/ssl/sslca	件。即 <code>zabbix_server.conf</code> 中的 <code>SSLCALocation</code> 参数。
/var/lib/zabbix/snmptraps	这个卷用于存放 <code>snmptraps.log</code> 文件。在创建 Zabbix server 实例时，它可以通过 <code>zabbix-snmptraps</code> 容器被共享，并通过 Docker 中的 <code>volumesfrom</code> 选项被继承。可以通过共享卷，并切换 <code>ZBX_ENABLE_SNMP_TRAPS</code> 环境变量的开关为 ' <code>true</code> ' 来开启 SNMP trap 处理功能。
/var/lib/zabbix/mibs	这个卷允许添加新的 MIB 文件。它不支持子目录。所有的 MIB 文件必须放在 <code>/var/lib/zabbix/mibs</code> 下。
Zabbix proxy	
/usr/lib/zabbix/externalscripts	这个卷用于外部检查。即 <code>zabbix_proxy.conf</code> 中 <code>ExternalScripts</code> 参数。
/var/lib/zabbix/modules	这个卷允许加载额外的模块，通过 <code>LoadModule</code> 功能以扩展 Zabbix server。
/var/lib/zabbix/enc	这个卷用于存放 TLS 相关文件。这些文件名指定使用 <code>ZBX_TLSCAFIE</code> , <code>ZBX_TLSCRLFILE</code> , <code>ZBX_TLSKEYFILE</code> 和 <code>ZBX_TLSPSKFILE</code> 环境变量。
/var/lib/zabbix/ssl/certs	这个卷用于存放用于客户端认证的 SSL 客户端证书文件。即 <code>zabbix_proxy.conf</code> 中 <code>SSLCertLocation</code> 参数。
/var/lib/zabbix/ssl/keys	这个卷用于存放用于客户端认证的 SSL 私钥文件。即 <code>zabbix_proxy.conf</code> 中 <code>SSLKeyLocation</code> 参数。
/var/lib/zabbix/ssl/sslca	这个卷用于存放用于 SSL 服务器证书认证的证书颁发机构 (CA) 文件。即 <code>zabbix_proxy.conf</code> 中 <code>SSLCALocation</code> 参数。
/var/lib/zabbix/snmptraps	这个卷用于存放 <code>snmptraps.log</code> 文件。在创建 Zabbix server 实例时，它可以通过 <code>zabbix-snmptraps</code> 容器被共享，并通过 Docker 中的 <code>volumesfrom</code> 选项被继承。可以通过共享卷，并切换 <code>ZBX_ENABLE_SNMP_TRAPS</code> 环境变量的开关为 ' <code>true</code> ' 来开启 SNMP trap 处理功能。
/var/lib/zabbix/mibs	这个卷允许添加新的 MIB 文件。它不支持子目录。所有的 MIB 文件必须放在 <code>/var/lib/zabbix/mibs</code> 下。
基于 Apache2 web 服务器的 Zabbix web 接口	
/etc/ssl/apache2	这个卷允许为 Zabbix web 接口启用 HTTPS 功能。这个卷必须包含用于 Apache2 SSL 连接的两个文件： <code>ssl.crt</code> 和 <code>ssl.key</code> 。
基于 Nginx web 服务器的 Zabbix web 接口	
/etc/ssl/nginx	这个卷允许为 Zabbix web 接口启用 HTTPS 功能。这个卷必须包含用于 Nginx SSL 连接的两个文件： <code>ssl.crt</code> 和 <code>ssl.key</code> 。
Zabbix snmptraps	
/var/lib/zabbix/snmptraps	这个卷包含以接收到的 SNMP trap 命名的 <code>snmptraps.log</code> 日志文件。
/var/lib/zabbix/mibs_	这个卷允许添加新的 MIB 文件。它不支持子目录。所有的 MIB 文件必须放在 <code>/var/lib/zabbix/mibs</code> 下。

其他信息可以在 Docker Hub 上的 Zabbix 官方源码库查看。

使用方法示例

示例 1

这个示例展现了如何运行支持 MySQL 数据库的 Zabbix server，基于 Nginx web 服务器运行 Zabbix web 接口，以

这个示例展现了如何运行支持MySQL数据库的Zabbix server，基于Nginx web服务器运行Zabbix web接口，以及Zabbix Java gateway。

1. 启动一个空的MySQL服务器实例

```
1. # docker run --name mysql-server -t \
2.   -e MYSQL_DATABASE="zabbix" \
3.   -e MYSQL_USER="zabbix" \
4.   -e MYSQL_PASSWORD="zabbix_pwd" \
5.   -e MYSQL_ROOT_PASSWORD="root_pwd" \
6.   -d mysql:5.7
```

1. 启动Zabbix Java gateway实例

```
1. # docker run --name zabbix-java-gateway -t \
2.   -d zabbix/zabbix-java-gateway:latest
```

1. 启动Zabbix server实例，并关联这个实例到已创建的MySQL服务器实例

```
1. # docker run --name zabbix-server-mysql -t \
2.   -e DB_SERVER_HOST="mysql-server" \
3.   -e MYSQL_DATABASE="zabbix" \
4.   -e MYSQL_USER="zabbix" \
5.   -e MYSQL_PASSWORD="zabbix_pwd" \
6.   -e MYSQL_ROOT_PASSWORD="root_pwd" \
7.   -e ZBX_JAVAGATEWAY="zabbix-java-gateway" \
8.   --link mysql-server:mysql \
9.   --link zabbix-java-gateway:zabbix-java-gateway \
10.  -p 10051:10051 \
11.  -d zabbix/zabbix-server-mysql:latest
```

Zabbix server实例暴露10051/TCP端口(Zabbix trapper)给主机。

1. 启动Zabbix web 接口，并将它与MySQL服务器实例和Zabbix server实例关联

```
1. # docker run --name zabbix-web-nginx-mysql -t \
2.   -e DB_SERVER_HOST="mysql-server" \
3.   -e MYSQL_DATABASE="zabbix" \
4.   -e MYSQL_USER="zabbix" \
5.   -e MYSQL_PASSWORD="zabbix_pwd" \
6.   -e MYSQL_ROOT_PASSWORD="root_pwd" \
7.   --link mysql-server:mysql \
8.   --link zabbix-server-mysql:zabbix-server \
9.   -p 80:80 \
10.  -d zabbix/zabbix-web-nginx-mysql:latest
```

Zabbix web 接口暴露80/TCP端口(HTTP)给主机。

示例 2

口，以及SNMP trap功能。

1. 启动空的PostgreSQL服务器实例

```
1. # docker run --name postgres-server -t \
2.   -e POSTGRES_USER="zabbix" \
3.   -e POSTGRES_PASSWORD="zabbix" \
4.   -e POSTGRES_DB="zabbix_pwd" \
5.   -d postgres:latest
```

1. 启动Zabbix snmptraps实例

```
1. # docker run --name zabbix-snmptraps -t \
2.   -v /zbx_instance/snmptraps:/var/lib/zabbix/snmptraps:rw \
3.   -v /var/lib/zabbix/mibs:/usr/share/snmp/mibs:ro \
4.   -p 162:162/udp \
5.   -d zabbix/zabbix-snmptraps:latest
```

Zabbix snmptrap实例暴露162/UDP端口(SNMP traps)给主机。

1. 启动Zabbix server实例并关联这个实例到已创建的PostgreSQL服务器实例

```
1. # docker run --name zabbix-server-pgsql -t \
2.   -e DB_SERVER_HOST="postgres-server" \
3.   -e POSTGRES_USER="zabbix" \
4.   -e POSTGRES_PASSWORD="zabbix" \
5.   -e POSTGRES_DB="zabbix_pwd" \
6.   -e ZBX_ENABLE_SNMP_TRAPS="true" \
7.   --link postgres-server:postgres \
8.   -p 10051:10051 \
9.   --volumes-from zabbix-snmptraps \
10.  -d zabbix/zabbix-server-pgsql:latest
```

Zabbix server实例暴露10051/TCP端口(Zabbix trapper)给主机。

1. 启动Zabbix web 接口，并将它与PostgreSQL服务器实例和Zabbix server实例关联

```
1. # docker run --name zabbix-web-nginx-pgsql -t \
2.   -e DB_SERVER_HOST="postgres-server" \
3.   -e POSTGRES_USER="zabbix" \
4.   -e POSTGRES_PASSWORD="zabbix" \
5.   -e POSTGRES_DB="zabbix_pwd" \
6.   --link postgres-server:postgres \
7.   --link zabbix-server-pgsql:zabbix-server \
8.   -p 443:443 \
9.   -v /etc/ssl/nginx:/etc/ssl/nginx:ro \
10.  -d zabbix/zabbix-web-nginx-pgsql:latest
```

Zabbix web 接口暴露443/TCP端口(HTTPS)给主机。/etc/ssl/nginx目录必须包含指定名称的证书。

Docker Compose

Zabbix为Docker提供了compose文件用于定义和运行多容器 (multi-container) 的Zabbix组件。这些compose文件可以在github.com上的Zabbix docker官方代码库中找到：<https://github.com/zabbix/zabbix-docker>。这些compose文件以示例方式添加，它们有非常多的分支，比如proxy的文件支持MySQL和SQLite3。

这里包含了一些不同版本的compose文件：

文件名称	描述
<code>docker-compose_v2_alpine_mysql_latest.yaml</code>	这个compose文件运行了基于Alpine Linux上的Zabbix 3.2最新版本的组件，支持MySQL数据库。
<code>docker-compose_v2_alpine_mysql_local.yaml</code>	这个compose文件本地构建了基于和运行了Alpine Linux上的Zabbix 3.2最新版本的组件，支持MySQL数据库。
<code>docker-compose_v2_alpine_pgsql_latest.yaml</code>	这个compose文件运行了基于Alpine Linux上的Zabbix 3.2最新版本的组件，支持PostgreSQL数据库。
<code>docker-compose_v2_alpine_pgsql_local.yaml</code>	这个compose文件本地构建了基于和运行了Alpine Linux上的Zabbix 3.2最新版本的组件，支持PostgreSQL数据库。
<code>docker-compose_v2_ubuntu_mysql_latest.yaml</code>	这个compose文件运行了基于Ubuntu 14.04上的Zabbix 3.2最新版本的组件，支持MySQL数据库。
<code>docker-compose_v2_ubuntu_mysql_local.yaml</code>	这个compose文件本地构建了基于和运行了Ubuntu 14.04上的Zabbix 3.2最新版本的组件，支持MySQL数据库。
<code>docker-compose_v2_ubuntu_pgsql_latest.yaml</code>	这个compose文件运行了基于Ubuntu 14.04上的Zabbix 3.2最新版本的组件，支持PostgreSQL数据库。
<code>docker-compose_v2_ubuntu_pgsql_local.yaml</code>	这个compose文件本地构建了基于和运行了Ubuntu 14.04上的Zabbix 3.2最新版本的组件，支持PostgreSQL数据库。

Docker compose文件只支持Docker Compose Version 2.

存储

Compose文件已经配置为支持主机的本地存储。当你使用compose文件运行Zabbix组件时，Docker Compose将在compose文件所在的文件夹中创建一个 `zbx_env` 目录。这个目录将包含上文卷 (Volumes) 一节中提到的相同结构，以用于数据库存储。

`/etc/localtime` 和 `/etc/timezone` 卷下的文件为只读模式。

环境变量文件

在github.com上存放compose文件的同一个目录里，你可以在compose文件中找到每个组件的默认环境变量。这些环境变量文件的命名类似于 `.env_<组件类型>` .

示例

示例 1

```
1. # docker-compose -f ./docker-compose_v2_alpine_mysql_latest.yaml up -d
```

这个命令将会为每个Zabbix组件下载最新的Zabbix 3.2镜像，并以detach模式运行。

不要忘记从github.com上的Zabbix官方源码库中，下载 `.env_<组件类型>` 文件及相关的compose文件。

示例 2

```
1. # docker-compose -f ./docker-compose_v2_ubuntu_mysql_local.yaml up -d
```

这个命令将会下载Ubuntu 14.04的基础镜像，然后在本地构建Zabbix 3.2，并以detach模式运行。

6 升级步骤

概要

本节提供了成功升级到Zabbix 3.4的必要步骤。

Zabbix 3.2.x, 3.0.x, 2.4.x, 2.2.x 及 2.0.x 可以直接升级到 Zabbix 3.4。如果需要从更早的版本升级，参考 Zabbix 2.0 和更早期版本的文档。

虽然不强制要求（但建议）升级Zabbix agents, Zabbix server和proxy必须使用[相同的大版本](#)。因此，在server-proxy架构的安装过程中，Zabbix server和所有的proxy必须停机升级。

为了在升级过程中将停机时间和数据丢失降低到最小，建议先停机升级Zabbix server，然后再逐个停机升级和启动Zabbix proxy。当所有的Proxy升级完毕后，再启动Zabbix Server。在Zabbix server的停机期间，运行状态的Proxy将会持续收集和存储数据，并在Zabbix server启动运行时，将这些数据发送给Zabbix server。在Zabbix server停机期间的任何问题（problems）的通知（notifications），只会在升级完后的server启动后再生成。注意使用SQLite数据库的Proxy，Proxy升级前的历史数据将会丢失。这是因为SQLite数据库的升级不支持，而且SQLite的数据库文件需要被手动移除。当Proxy第一次启动而SQLite数据库文件不存在时，Proxy将会自动创建这个文件。

数据库升级到3.4版本可能需要较长的时间，取决于数据库大小。

从 3.2.x 到 3.4 升级之前：

- 阅读 [3.4.0升级日志](#)
- 检查 [3.4.0安装要求](#)

如果从早期版本升级，也要阅读下列版本的升级日志：[2.0 -> 2.2](#), [2.2 -> 2.4](#), [2.4 -> 3.0](#) 和 [3.0 -> 3.2](#)。

Server升级过程

1 停止Zabbix server

停止Zabbix server以确保没有新的数据写入数据库。

2 备份现有的Zabbix数据库

这是非常重要的步骤。确认你已经备份了你的数据库，以防止升级过程失败（如磁盘空间不足，断电及其他意外问题）。

3 备份配置文件，PHP文件和Zabbix二进制文件

保留一份Zabbix二进制文件，配置文件及PHP文件目录的备份。

4 安装新的server二进制文件

使用这个[指导手册](#)以通过源代码编译Zabbix server。

5 检查server配置参数

[zabbix_server.conf](#)中的一些参数发生了变化，同时增加了新的参数。你可能需要检查这些参数。

6 启动新的Zabbix二进制

启动新的二进制，通过检查日志以确认二进制是否成功启动。

Zabbix server将会自动升级数据库。开始后，Zabbix server报告当前（强制和非强制的）和需要的数据库版本。如果当前强制版本低于需要的版本，Zabbix server自动执行所需要的数据库补丁。数据库升级的开始和进度信息（百分比）将会写入Zabbix server的日志文件中。当升级完成后，将会在日志文件中写入一条“database upgrade fully completed”的信息。如果任何升级补丁失败，Zabbix server将不会启动，即使数据库当前强制版本比所需版本高，Zabbix server也不会启动。只有当数据库当前的强制版本与需要的强制版本相符时，Zabbix server才会启动。

```
1. 8673:20161117:104750.259 current database version (mandatory/optional): 03040000/03040000
2. 8673:20161117:104750.259 required mandatory version: 03040000
```

在你启动server之前：

- 确认数据库用户有足够的权限（创建表，删除表，建立索引，删除索引）
- 确认拥有足够的空余磁盘空间

7 安装新的Zabbix web接口

所需的PHP最低版本为5.4.0。按需升级并按[安装指导](#)操作。

Proxy升级过程

1 停止Zabbix proxy

停止Zabbix proxy。

2 备份配置文件和Zabbix proxy二进制文件

保留一份Zabbix proxy二进制文件和配置文件的备份。

3 安装新的proxy二进制文件

使用这个[指导手册](#)以通过源代码编译Zabbix proxy。

4 检查proxy配置参数

[zabbix_proxy.conf](#)中的一些配置参数可能发生了变化，同时增加了新的参数。你可能需要检查这些参数。

5 启动新的Zabbix proxy

启动新的Zabbix proxy。通过检查日志文件以确认proxy是否成功启动。

Zabbix proxy将会自动升级数据库。数据库升级操作同[Zabbix server](#)的数据库升级类似。

Agent升级过程

不强制要求升级Zabbix agent。你只需要升级那些需要使用新功能的agents

1 停止Zabbix agent

停止Zabbix agent。

2 备份配置文件和Zabbix agent二进制文件

保留一份Zabbix agent二进制文件和配置文件的备份。

3 安装新的agent二进制文件

使用这个[指导手册](#)以通过源代码变异Zabbix agent。

你也可以通过下面的链接选择下载预编译的Zabbix agent: [Zabbix download page](#)。

4 检查agent配置参数

`zabbix_agentd.conf`中的一些配置参数可能发生了变化，同时增加了新的参数。你可能需要检查这些参数。

5 启动新的Zabbix agent

启动新的Zabbix agent。检查日志文件以确认agent是否成功启动。

7 已知问题

IPMI检查

如果使用Debian / Ubuntu中标准的OpenIPMI类库，IPMI检查可能无法正常工作。启用OpenSSL开关并重新编译OpenIPMI类库，可以解决这个问题。可参照[ZBX-6139](#)。

SSH检查

一些Linux分发版本，如Debian, Ubuntu，如果使用安装包安装了libssh2类库，则系统不支持使用密码加密私钥。查看[ZBX-4850](#)了解详细内容。

ODBC检查

如果Zabbix server或者proxy使用了MySQL作为数据库，由于[upstream bug](#)，MySQL ODBC类库可能会无法正常工作。查看[ZBX-7665](#)获得详细信息及变通方法。

HTTPS检查

在Web场景 (Web scenarios) 中，如果目标服务器配置了不允许TLS v1.0及以下版本的协议，使用https协议和Zabbix agent检查 `net.tcp.service[https...]` 和 `net.tcp.service.perf[https...]` 可能会失败。查看[ZBX-9879](#)了解详细信息和变通方法。

Web监控

Zabbix server在CentOS 6, CentOS 7，以及可能在其他相关的Linux分发版本上存在内存泄漏问题。这是由于在Web scenarios中启用了“SSL verify peer”，从而引发[upstream bug](#)。查看[ZBX-10486](#)了解详细信息和变通方法。

图表 (Graphs)

切换到夏令时 (Daylight Saving Time, DST) 会导致显示X轴标签错误 (如日期重复，日期缺失等)。

日志文件监控

当文件系统使用空间为100%时，如果日志文件仍然在被追加，`log[]` 和 `logrt[]` 监控项会反复从头重新读取日志文件。(查看[ZBX-10884](#)了解更多信息)

MySQL中的慢查询

如果查找监控项的不存在的值，Zabbix server生成慢选择查询 (slow select queries)。这是由于MySQL 5.6/5.7版本中一个已知的[问题](#) 造成的。一种变通方法是在MySQL中禁用index_condition_pushdown优化。详细的讨论可参考[ZBX-10652](#)。

8 模版变更

本页列出了所有Zabbix内置模版的变更。根据这些变化，建议对已安装的模版进行修改。可以通过导入最新版本的模版，或者通过手动配置这些变更。

Zabbix 3.2.0 模版变更

*Template OS Windows*模版中添加了一条新的低级别发现 (low-level discovery) 规则

`service.discovery`。这个规则包含了 `service.info[#{SERVICE.NAME},state]` 监控项原型用以监控服务状态。

为了扩展该*Template OS Windows*模版，可以从

https://www.zabbix.org/wiki/Zabbix_Templates/Official_Templates导入。

9.3.4.0 升级日志

Zabbix 3.4.0 尚未发布。

以下是从Zabbix 3.2.x升级到Zabbix 3.4.0的升级日志。

对于Zabbix 3.2.0以前的版本，也可以升级到Zabbix 3.4.0，访问[升级过程](#) 版块查看所有关于从Zabbix旧版本升级的相关信息。

额外依赖

IPMI监控目前需要安装libevent库(1.4或以后的版本)。

区分大小写的MySQL数据库

区分大小写的MySQL数据库需要特定的数据库网络支持。建议在新安装过程中，创建一个区分大小写的MySQL数据库。如果之前你创建了一个使用utf8字符集的MySQL数据库，为了使存储的数据支持大小写敏感，你需要将字符集转换成utf8_bin。

配置参数变更

在Zabbix server和proxy的配置中，新增了一个新的配置参数：`SocketDir`。这个参数指定了Zabbix内部socket文件存放的路径（默认为`/tmp`）。如果一个服务器上同时运行了一个server和一个proxy，由于server和proxy使用不同的socket文件名，因此对server和proxy使用相同的`SocketDir`参数值是安全的。但是，如果在一个服务器上运行了多个server或者proxy，需要使用不同的`SocketDir`配置。

监控项值的预处理选项

监控项值的预处理选项已经进行了统一整合，并在监控项管理中展开成单独的一个版块。因此，之前在监控项 / 监控项原型下的多个单独的预处理参数被废弃了：

- 数据类型 (*Data type*)，使用自定义系数 (*Use custom multiplier*) and 存储值 (*Store value*) 栏位
- API: `data_type` , `multiplier` 和 `delta`
- XML导出: `data_type` , `multiplier` 和 `delta` 标签

在升级期间，所有监控项 / 监控项远行中的上述属性会自动转换成新的预处理参数。同样的，当导入旧版本的XML文件时，这些选项也会被自动转换成对应的预处理选项。

选择嵌入主机组的语法

如果从Zabbix 3.2.0和3.2.1版本升级，请留意包含嵌入子组 (nested subgroups) 的语法进行了更改。

在Zabbix 3.2.0和3.2.1版本中，父主机组包含了嵌入式主机组，父组以`hostgroup/`进行指定。从Zabbix 3.2.2及Zabbix 3.4.0起，废弃了'/'的语法。简单的使用具体的父主机组名称，即可以包含嵌入的主机组。这意味着：如果指定了一个主机组，比如在动作条件 (action conditions) 中，将会静默地包含所有嵌入的主机组。

协议变更

前端站点和server之间的通信协议进行了改变。脚本执行时，前端到server的请求中添加一个新的参数：“sid”。这个参数获得会话的ID(认证令牌)，并用它来检查用户执行脚本的权限。另外，脚本对于主机权限（读 / 写，只读和拒绝）的额外检查也被添加到了请求中。如果缺少访问主机的必要权限，server会返回拒绝执行脚本的提示。

API变更

废弃了 `isreadable()` 和 `iswritable()` 方法，同时也移除了 `proxy.interfaces` 参数。

执行命令/脚本的变更

在Zabbix 3.4种，由于引入了命令 / 脚本的退出代码的检查。如果退出代码不为0，[告警脚本](#)可以被多次执行。虽然拥有“nowait”标签的监控项的行为没有变化，但是，由于对于退出代码的额外检查，之前通过Zabbix server执行的使用用户变量配置的监控项，外部检查监控项，以及[system.run](#)监控项，这些退出代码不为0，因此可能会成为“Not supported”状态。

5. 快速开始

请使用侧边导航栏来访问快速开始部分的内容。

1 登陆和配置用户

简介

本章你会学习如何登陆Zabbix，以及在Zabbix内建立一个系统用户。

登陆



这是Zabbix的“欢迎”界面。输入用户名 **Admin** 以及密码 **zabbix** 以作为Zabbix超级用户登陆。

登陆后，你将会在页面右下角看到“以管理员连接 (Connected as Admin)”。同时会获得访问配置 (Configuration) 和 管理 (Administration) 菜单的权限。

暴力破解攻击的保护机制

为了防止暴力破解和词典攻击，如果发生连续尝试五次登陆失败，Zabbix界面将在暂停30秒。

在下次成功登陆后，将会在界面上尝试登录失败的IP地址。

增加用户

可以在管理 (Administration) → 用户 (Users) 下 查看用户信息。



Zabbix在安装后只定义了两个用户。

- 'Admin' 用户是Zabbix的一个超级管理员，拥有所有权限。
- 'Guest' 用户是一个特殊的默认用户。如果你没有登陆，你访问Zabbix的时候使用的其实是“guest”权限。默认情况下，“guest”用户对Zabbix中的对象没有任何权限。

点击 创建用户 (Create user) 以增加用户。

在添加用户的表单中，确认将新增的用户添加到了一个已有的[用户组](#)，比如 'Zabbix administrators'。



默认情况下，没有为新增的用户定义媒介 (media，即通知发送方式)。如需要创建，可以到‘媒介 (Media)’标签下，然后点击增加 (Add)。



在这个对话框中，为用户输入Email地址。

你可以为媒介指定一个时间活动周期，(访问[时间周期说明](#)页面，查看该字段格式的描述)。默认情况下，媒介一直是

活动的。你也可以通过自定义触发器严重等级来激活媒介， 默认所有的等级都保持开启。

点击新增 (Add)， 然后在用户属性表单中点击新增 (Add)。新的用户将出现在用户清单中。



增加权限

一个新用户默认没有权限访问主机。在组 (Groups) 下，点击用户所在的组 (这里为'Zabbix administrators')，为用户提升权限。在组的属性表单下，点击权限 (Permissions) 标签。



为了使这个用户对Linux servers组拥有只读权限，点击用户组选择栏位旁边的选择 (Select) 按钮。



在这个对话框中，勾选'Linux servers'旁边的复选框，然后点击选择 (Select)。Linux servers组会在选择栏位中显示。点击'读 (Read)'按钮以设置权限，然后点击添加 (Add) 将所列出的权限分配给这个组。在用户组属性表单中，点击更新 (Update)。

在Zabbix中，对于主机的访问权限是分配给[用户组 \(user groups\)](#)，而不是单独的用户 (users)。

完成！你现在可以尝试使用这个新用户的访问Zabbix了！

2 新建主机

概要

通过本节，你将会学习到如何建立一个新的主机。

Zabbix中的主机（Host）是一个你想要监控的网络实体（物理的，或者虚拟的）。Zabbix中，对于主机的定义非常灵活。它可以时一台物理服务器，一个网络交换机，一个虚拟机或者一些应用。

增加主机

Zabbix中，可以通过配置（*Configuration*）→ 主机（*Hosts*）菜单，查看已配置的主机信息。默认已有一个名为'Zabbix server'的预先定义好的主机。但我们需要学习如何添加另一个。

点击创建主机（*Create host*）以添加新的主机，这将向我们显示一张主机配置表格。



至少需要填写下列字段：

主机名称（*Host name*）

- 输入一个主机名称，可以使用字母数字、空格、点“.”、中划线“-”、下划线“_”。

组

- 从右边的选择框中，选择一个或者多个组，然后点击 « 移动它们到'所在组（In groups）'选择框。

所有访问权限都分配到主机组，而不是单独的主机。这也是主机需要属于至少一个组的原因。

IP地址

- 输入主机的IP地址。注意如果这是Zabbix server的IP地址，它必须是Zabbix agent配置文件中‘Server’参数的值。

暂时保持其他选项的默认值。

当完成后，点击添加（*Add*）。你可以在主机列表中看到你新添加的主机。

如果可用性（*Availability*）列中的ZBX图标是红色的，通信可能存在一些问题。将你的鼠标移动到上面查看错误信息。如果这个图标是灰色的，说明目前状态还没更新。确认Zabbix server正在运行，同时过一会儿刷新这个页面。

3 新建监控项

概要

本节你会学习如何新建一个监控项 (Item)。

监控项是Zabbix中获得数据的基础。没有监控项，就没有数据—因为一个主机中只有监控项定义了单一的指标或者需要获得的数据。

添加监控项

主机包含了所有的监控项。如果需要配置一个监控项的示例，我们需要前往配置 (*Configuration*) → 主机 (*Hosts*) 并找到我们已创建的'新主机 (New host)'。

在'新主机 (New host)'行中，监控项 (*Items*) 的链接旁的数量会显示为'0'。点击这个链接，然后点击创建监控项 (*Create item*)，将会显示一个监控项定义表格。

Item Preprocessing

Name	CPU Load		
Type	Zabbix agent		
Key	system.cpu.load Select		
Host interface	192.168.3.31 : 32050		
Type of information	Numeric (float)		
Units			
Update interval	30s		
Custom intervals	Type	Interval	Period
	Flexible	Scheduling	50s 1-7,00:00-24:00
	Remove		
	Add		
History storage period	7d		
Trend storage period	365d		
Show value	As is show value mappings		
New application			
Applications	<ul style="list-style-type: none"> -None- CPU Filesystems General Memory Network interfaces OS Performance Processes Security 		
Populates host inventory field	-None-		
Description			
Enabled	<input checked="" type="checkbox"/>		
Add Cancel			

对于监控项的示例，需要输入以下必要的信息：

名称 (**Name**)

- 输入 *CPU Load* 作为值。在列表中和其他地方，都会显示这个值作为监控项名称。

值 (Key)

- 手动输入 `system.cpu.load` 作为值。这是监控项的一个技术上的名称，用于识别获取信息的类型。这个特定值需要是Zabbix Agent预定义值中的一种。

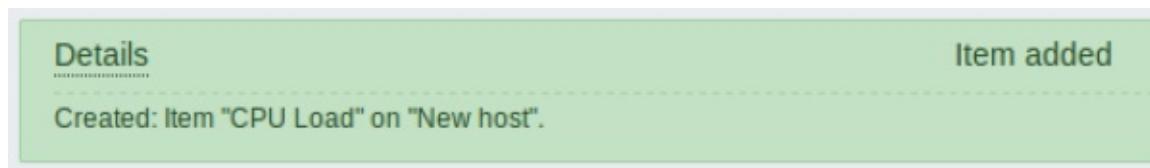
信息类型 (Type of information)

- 在此处选择 `Numeric (float)`。这个属性定义了想获得数据的格式。

你也需要减少[监控项历史](#)保留的天数，7或者14天。对于数据库而言，最佳实践是避免数据库保留过多的历史数据。

我们暂时保持[其他选项](#)的默认值。

当完成后，点击添加 (Add)。新的监控项将出现在监控项列表中。点击列表中的详细 (Details) 以查看具体细节。



查看数据

当一个监控项定义完成后，你可能好奇它具体获得了什么值。前往监控 (Monitoring) → 最新数据 (Latest data)，点击 - other - 前面的 +，然后查看你之前定义的监控项和获得的值。

HOST	NAME	LAST CHECK	LAST VALUE	CHANGE
New host	- other - (1 item)			
	CPU Load	2015-08-08 16:00:49	2.24	-0.26

同时，第一次获得的监控项值最多需要60秒才能到达。默认情况下，这是服务器读取变化后的配置文件，获取并执行新的监控项的频率。

如果你在‘变化 (Change)’列中没有看到值，可能到目前为止只获得了一次值。等待30秒以获得新的监控项值。

如果你在当前界面中没有看到监控项的信息，请确认：

- 你输入的监控项‘值 (Key)’和‘信息类型 (Type of information)’同截图中的一致
- agent和server都在运行状态
- 主机状态为‘监控 (Monitored)’并且它的可用性图标是绿色的
- 监控项处于启用状态

图表

当监控项运行了一段时间后，可以查看可视化图表。[简单图表](#) 适用于任何被监控的数值型 (numeric) 监控项，且不需要额外的配置。这些图表会在运行时生成。

前往监控 (Monitoring) → 最新数据 (Latest data)，然后点击监控项后的‘图表 (Graph)’链接以查看图

表。



4 新建触发器

概述

本节你会学习如何配置一个触发器 (trigger)。

监控项只是用于收集数据。如果需要自动评估收到的数据，我们则需要定义触发器。触发器包含了一个表达式，这个表达式定义了数据的可接受的阈值级别。

如果收到的数据超过了这个定义好的级别，触发器将被“触发”，或者进入“异常 (Problem)”状态—从而引起我们的注意，让我们知道有问题发生。如果数据再次恢复到合理的范围，触发器将会到“正常 (Ok)”状态。

添加触发器

为监控项配置触发器，前往配置 (Configuration) → 主机 (Hosts)，找到'新增主机 (New host)'，点击旁边的触发器 (Triggers)，然后点击创建触发器 (Create trigger)。这将会向我们展现一个触发器定义表单。



对于触发器，有下列必填项：

名称 (Name)

- 输入 `CPU load too high on 'New host' for 3 minutes` 作为值。这个值会作为触发器的名称被现实在列表和其他地方。

表达式 (Expression)

- 输入：`{New host:system.cpu.load.avg(180)}>2`

值时触发器的表达式。确认这个表达式输入正确，包括所有的符号。此处，监控项值(`system.cpu.load`)用于指出具体的监控项。这个特定的表达式大致是说如果3分钟内，CPU负载的平均值超过2，那么就触发了问题的阈值。你可以查看更多的[触发器表达式语法](#)信息。

完成后，点击添加 (Add)。新的触发器将会显示在触发器列表中。

显示触发器状态

当一个触发器定义完毕后，你可能想查看它的状态。

前往监控 (Monitoring) → 触发器 (Triggers) 以查看。3分钟后（我们需要等待3分钟以评估这个触发器的3分钟平均值），触发器会在这里显示。应该会有一个绿色的'OK'在'状态 (Status)'列中闪烁。



闪烁意味着这个触发器状态最近30分钟内发生过变化。

如果此处出现一个闪烁的红色'PROBLEM'，显然，这说明了CPU负载已经超过了你在触发器里定义的阈值级别。

5 获取问题通知

概述

在本节中，你会学习如何在Zabbix中以通知（notifications）的方式配置报警（alerting）。

当监控项收集了数据后，触发器会根据异常状态触发报警。根据一些报警机制，它也会通知我们一些重要的事件，而不需要我们直接在Zabbix前端进行查看。

这就是通知（Notifications）的功能。E-mail是最常用的异常通知发送方式。我们将会学习如何配置e-mail通知。

E-mail设置

Zabbix中最初内置了一些预定义的通知发送方式。E-mail通知是其中的一种。

前往管理（Administration）→ 媒体类型（Media types），点击预定义媒体类型列表中的Email，以配置E-mail。



这将向我们展现e-mail设置定义表单。



根据你的环境，设置SMTP服务器，SMTP helo，SMTP e-mail的值。

'SMTP email'将作为Zabbix通知的'发件人（From）'地址。

一切就绪后，点击 更新（Update）。

现在你已经配置了'Email'作为一种可用的媒体类型。一个媒体类型必须通过发送地址来关联用户（如同我们在[配置一个新用户]中做的），否则它将无法生效。== 新建动作 == 发送通知是Zabbix中 [manual:config:notifications:action|动作（actions）执行的操作之一。因此，为了建立一个通知，前往配置（Configuration）→ 动作（Actions），然后点击创建动作（Create action）。



在这个表单中，输入这个动作的名称。

{TRIGGER.STATUS} 和 {TRIGGER.NAME} 是宏（macros）或者变量，可以在Default subject 和 Default message 区域查看。会以实际的触发器状态和触发器名称的值替代。

在大多数简单的例子中，如果我们不添加更多的指定条件，这个动作会在触发器从 'Ok' 变为 'Problem' 时发生。

我们还需要定义这个动作具体做了什么 — 即在 操作（Operations） 标签页中执行的操作。点击新建（New），将会打开一个操作表单。



这里，在发送给用户 (*Send to Users*) 块中点击添加 (*Add*)，然后选择我们之前定义的用户 ('user')。选择 'Email' 作为 *Send only to* 的值。完成后，在操作明细区域中，点击添加 (*Add*)。

这是一个简单的动作配置步骤，即点击动作表单中的添加 (*Add*)。

获得通知

现在，发送通知配置完成，我们看看它如何将通知发送给实际接收人。为了实现这个目的，我们需要你主机的负载，这样我们的触发器才会被触发，我们会收到异常通知。

打开主机的控制台，并运行：

```
1. cat /dev/urandom | md5sum
```

你需要运行一个或者多个[这样的进程](#)。

现在，前往监控 (*Monitoring*) → 最新数据 (*Latest data*)，查看 'CPU Load' 的值是否已经增长。记住，为了使我们的触发器触发 (*fire*)，'CPU Load' 的值需要在在3分钟运行的过程中超过2。一旦满足这个条件：

- 在监控 (*Monitoring*) → 触发器 (*Triggers*) 中，你会看到这个触发器的状态呈现一个绿色闪烁的 'Problem'
- 你的e-mail中，会收到一个异常通知

如果通知功能没有正常工作：

- 再次验证e-mail设置和动作设置已经被正确配置
- 确认你创建的用户对生成事件的主机至少拥有读 (read) 权限。正如[添加用户](#)步骤中提到的，'Zabbix administrators' 用户组中的用户必须对 'Linux servers' 主机组 (该主机所属组) 至少拥有读 (read) 权限。
- 另外，你可以在报告 (*Reports*) → 动作日志 (*Action log*) 中检查动作日志。

6 新建模版

概述

在本节中，你将会学习如何配置一个模版。

我们在之前的章节中学会了如何配置监控项、触发器，以及如果从主机上获得问题的通知。

虽然这些步骤提供了很大的灵活性，但仍然需要很多步骤才能完成。如果我们需要配置上千台主机，一些自动化操作会带来更多便利性。

模版 (templates) 功能可以实现这一点。模版允许对有用的监控项、触发器和其他对象进行分组，只需要一步就可以对监控主机应用模版，以达到反复重用的目的。

当一个模版链接到一个主机后，主机会继承这个模版中的所有对象。简单而言，一组预先定义好的检查会被快速应用到主机上。

添加模版

开始使用模版，你必须先创建一个。在配置 (*Configuration*) → 模版 (*Templates*) 中，点击创建模版 (*Create template*)。这将会像我们展现一个模版配置表格。



需要输入以下必填字段：

模版名称 (*Template name*)

- 输入一个模版名称。可以使用数字、字母、空格及下划线。

组 (*Groups*)

- 从右边的选择框中选择一个或者多个组，点击 « 移动它们到'所在组 (In groups)'选择框。模版必须属于至少一个组。

完成后，点击添加 (*Add*)。你新建的模版可以在模版列表中查看。



你可以在这看到模版信息。但这个模版中没有任何信息—没有监控项、触发器或者其他对象。

在模版中添加监控项

为了在模版中添加监控项，前往 'New host' 的监控项列表。在配置 (*Configuration*) → 主机 (*Hosts*)，点击 'New host' 旁边的监控项 (*Items*)。

然后：

- 选中列表中 'CPU Load' 监控项的选择框

- 点击列表下方的复制 (Copy)
- 选择想要复制这个监控项的目标模版



- 点击复制 (Copy)

你现在可以前往配置 (*Configuration*) → 模版 (*Templates*)，'新模版 (New template)' 中会有一个新的监控项。

我们目前至创建了一个监控项，但你可以用同样的方法在模版中添加其他的监控项，触发器以及其他对象，直到完成满足特定需求（如监控OS，监控单个应用）的完整的对象组合。

链接模版到主机

准备一个模版后，将它链接到一个主机。前往配置 (*Configuration*) → 主机 (*Hosts*)，点击'新主机 (New host)' 打开表单，前往模版 (**Templates**) 标签页。

点击链接新模版 (*Link new templates*) 旁边的选择 (*Select*)，在弹出的窗口中，点击我们创建模版的名称 ('New template')，它会出现在链接新模版 (*Link new templates*) 区域，点击添加 (*Add*)。这个模版会出现在已链接模版 (*Linked templates*) 列表中。



点击更新 (*Update*) 保存配置。现在，新模版及其所有的对象被添加到了主机。

你可能会想到，我们可以使用同样的方法将模版应用到其他主机。任何在模版级别的监控项、触发器及其他对象的变更，也会传递给所有链接该模版的主机。

链接预定义模版到主机

你可能注意到，Zabbix为各种操作系统、设备以及应用准备一些预定义的模版。为了快速部署监控，你可能会将它们中的一些与主机关联。但请注意，一些模版需要根据你的实际环境进行合适的调整。比如：一些检查项是不需要的，一些轮询周期过于频繁。

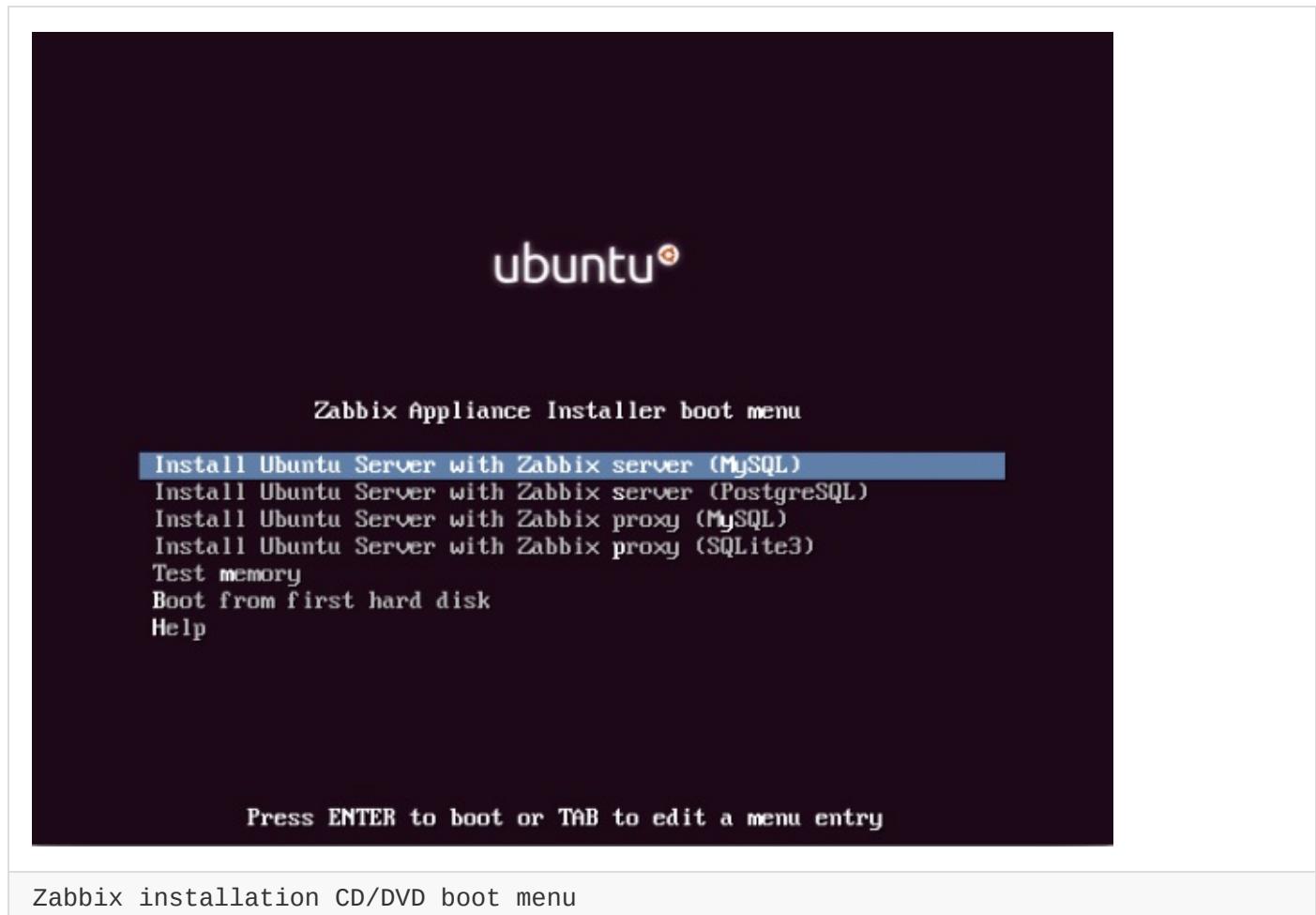
可参考该链接，查看更多关于[模版](#)的信息。

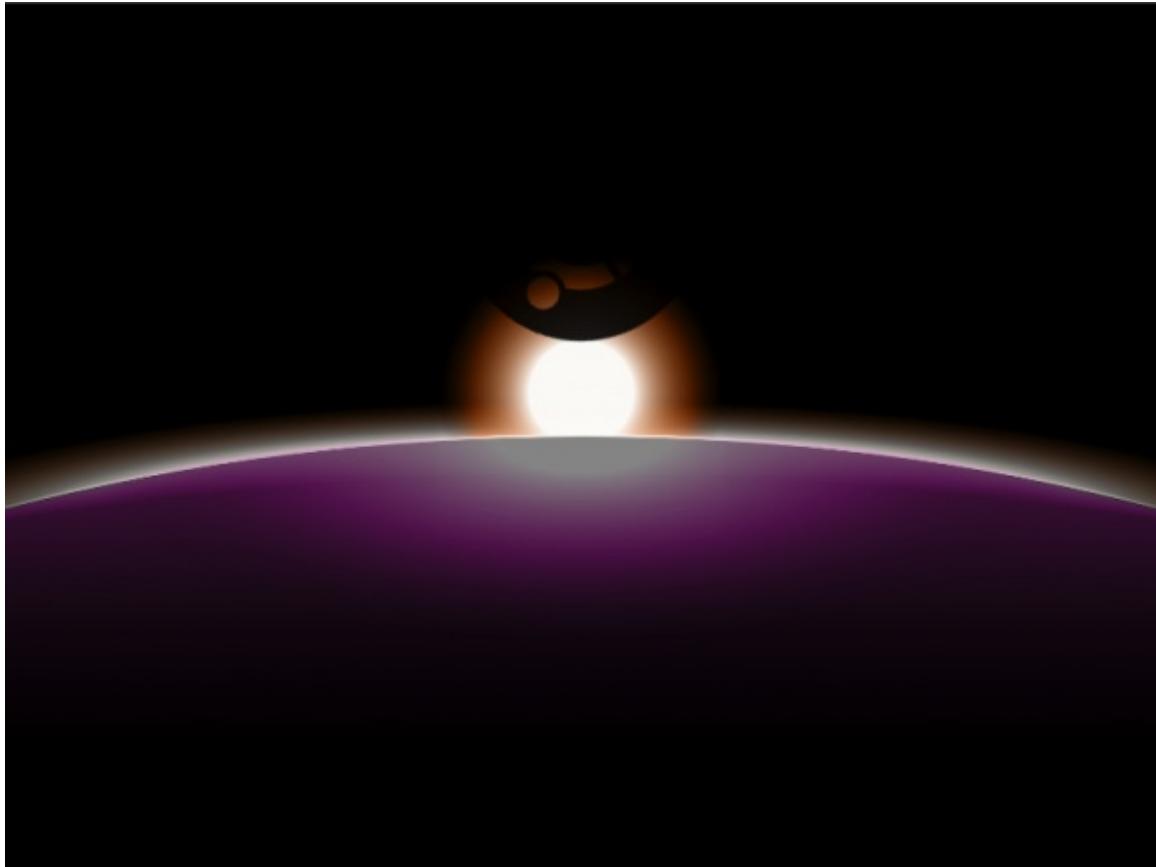
6. Zabbix设备

概述

用户可下载 [<http://www.zabbix.com/download.php#appliance>] Zabbix设备或Zabbix设备安装光盘映像，以作为手动设置、或重新使用现有服务器的备选项。Zabbix server (MySQL), Zabbix server (PostgreSQL), Zabbix proxy (MySQL) 以及Zabbix proxy (SQLite 3) 可使用Zabbix设备光盘，进行即时部署。

Zabbix设备虚拟机已备好MySQL数据库以支持Zabbix server，且是通过使用Zabbix设备安装光盘构建而成。





Booting Zabbix appliance

Zabbix装置及安装CD版本皆基于以下Ubuntu版本：

Zabbix 设备版本	Ubuntu 版本
3.0.0	14.04.3

Zabbix设备在以下格式中可用：：

- vmdk ((VMware/虚拟箱))
- OVF (开放虚拟化格式)
- KVM
- HDD/flash图像, USB闪存盘
- 实时 CD/DVD
- Xen客机
- Microsoft VHD (Azure)
- Microsoft VHD (Hyper-V)

若要运行，启动此装置，在已通过动态主机配置协议：`http:<host_ip>/zabbix`的IP上，点击浏览器它具有Zabbix server配置，运行MySQL数据库，同时具有可用前端。该装置曾使用名为*Preseed*文件的标准Ubuntu/Debian特征进行构建。

1.1 更改Ubuntu配置

应用于基础Ubuntu的配置略有更改。== 存储库 ==官方Zabbix存储库 [repository](http://repo.zabbix.com/zabbix/3.0/ubuntu) 已被添加至 /etc/apt/sources.list: ## Zabbix 存储库 deb <http://repo.zabbix.com/zabbix/3.0/ubuntu> trusty main deb-src <http://repo.zabbix.com/zabbix/3.0/ubuntu> trusty main == 防火墙 ==该装置使用iptables防火墙。以下为预定义规则：打开SSH端口(22 TCP)； 打开Zabbix agent (10050 TCP) 以及Zabbix trapper (10051 TCP) 端口； 打开HTTP (80 TCP) 以及HTTPS (443 TCP) 端口； 打开SNMP Trap端口 (162 UDP)； 打开连到NTP端口 (53 UDP) 的输出连接； ICMP pakets限制为每秒5个数据包； * 终止所有其他的连入连接。

1.1.1 附加Packages

已添加多方面基础设施，一般来说，使用Zabbix工作及监测时更加简易： : *iptables-persistent mc htop snmptrapfmt * snmp-mibs-downloader*这些安装包一些被Zabbix使用，一些帮助用户配置或管理装备设置

1.1.2 使用静态ip地址

默认情况下，设备使用DHCP来获取IP地址。静态地址详细说明： 以root用户身份登陆； 在你最擅用的编辑器中打开/etc/network/interfaces 文件； * _iface eth0 inet dhcp → iface eth0 inet static 在 iface eth0 inet static之后添加以下行列： 地址<设备的IP地址> 子网掩码<网络掩码> 网关<网关地址> 执行命令 *sudo ifdown eth0 && sudo ifup eth0.*<note>有关其他可行选项的相关信息，参见官方Ubuntu文档 documentation.</note>要配置DNS，，在 /etc/resolv.conf中添加域名服务器条目，在其自身行列中将每一个域名服务器设定为： *nameserver 192.168.1.2._

1.1.3 更改时区

默认情况下，该设备对系统时钟使用UTC。要想改变时区，，则将/usr/share/zoneinfo 中的相应文件复制到 /etc/localtime，例如： *cp /usr/share/zoneinfo/Europe/Riga /etc/localtime*

1.1.4 局部更改

该设备包含一些区域设置更改： * 包含语言： _en_US.UTF-8, ru_RU.UTF-8, ja_JP.UTF-8, cs_CZ.UTF-8, ko_KR.UTF-8, it_IT.UTF-8, pt_BR.UTF-8, sk_SK.UTF-8, uk_UA.UTF-8, fr_FR.UTF-8, pl.UTF-8; * Default locale is en_US.UTF-8.这些更改都须支持多语种Zabbix网络接口。_

1.1.5 其他更改

- 网络工作被配置为使用DHCP来获取IP地址； Utility *fping* 设置为具有4710权限，且由*zabbix - uid* 组件所有，只允许由Zabbix组件使用； Ntpd配置为与公共池服务器ntp.ubuntu.com同步； Ext4文件系统 使用逻辑卷管理器； “UseDNS no”添加到SSH服务配置文件etc/ssh/sshd_config中，以避免长时间SSH 连接等待； * 守护进程snmpd无法使用 /etc/default/snmpd 配置文件。

1.2 Zabbix配置

Zabbix安装设备有下列密码及其他配置更改：

1.2.1 凭证（登陆：密码）

系统： 设备：zabbix数据库：root:<随机> zabbix:<随机><note>D数据库密码在安装过程中随机生成.Root 密码储存在/root/.my.cnf文件中，无需在“root”账户下输入密码.</note>Zabbix 前端：管理员：zabbix要更改数据库用户密码，需在以下位置更改：MySQL； /etc/zabbix/zabbix_server.conf； * /etc/zabbix/web/zabbix.conf.php.

1.2.2 文件位置

- 配置文件存放在/etc/zabbix中。Zabbix server.proxy 以及agent日志文件存放 在/var/log/zabbix。Zabbix前端存放在 /usr/share/zabbix。zabbix 用户目录主页为 */var/lib/zabbix。

1.2.3 Zabbix配置更改

- Zabbix前端服务器名称设置为“zabbix 设备”； 前端时区设置为欧洲/里加（在 */etc/apache2/conf-available/zabbix.conf中修改）；

1.2.4 配置防护

如果您正在运行该设备的实时CD / DVD版本，或者出于其他原因，不能拥有持久存储。您可以备份整个数据库，包括所有配置和收集的数据。要创建备份，则运行： sudo mysqldump zabbix | bzip2 -9 > dbdump.bz2现在您可以将文件dbdump.bz2 传输到另一台机器要想从备份中进行恢复，将其传输到设备上并执行 bzcat dbdump.bz2 | sudo mysql zabbix<note important>确保恢复过程中zabbix server终止运行。</note>

1.3 前端接入

默认各处前端接入。前端可访问 <http://<host>/zabbix>. 可在 /etc/apache2/conf-available/zabbix.conf中定制，修改此文件后，须重启网页服务器。为此，使用SSH登陆并且执行： service apache2 restart_

1.4 防火墙

默认情况下，只有更改条目中所列出的端口是打开的。要打开额外端口，只需修改 “/etc/iptables/rules.v4”

or “/etc/iptables/rules.v6” 文件，并重新下载防火墙规则： service iptables-persistent reload

1.5 监测能力

以下项目为Zabbix安装提供支持： SNMP IPMI Web 监测 VMware 监测 Jabber 通知 EZ Texting 通知 ODBC SSH2 IPv6 SNMP Traps * Zabbix Java Gateway

1.6 SNMP traps

Zabbix设备运用 `_snmptrapfmt` 处理 SNMP traps. 被设置为接收各处 traps. A无需验证。如果您想启用身份验证，需要更改 `/etc/snmp/snmptrapd.conf`文件并指定所需的授权设置。所有 traps都存储在 `/var/log/zabbix/snmptrapfmt.log`文件中。在文件达到2GB大小之前，它由logrotate驱动。

1.7 升级

Zabbix设备安装包可以升级。要想升级，则运行： sudo apt-get -only-upgrade install zabbix*

1.8 命名，初始化和其他脚本

已提供适当的初始脚本。要想控制Zabbix server，运行以下任一程序： service zabbix-server status将 `server` 替换为 `agent` 成为Zabbix agent守护进程，，或替换为`proxy` 以成为Zabbix proxy守护进程。

1.8.1 增加可用磁盘空间

<note warning>在尝试任何步骤之前，请创建所有数据备份。.</note>A设备中可用磁盘空间可能不够，这种情况下，可以扩展磁盘，为此，可首先在虚拟化环境中拓展块设备，然后执行以下步骤：启动`fdisk` 以更改分区大小。作为root身份以执行：`fdisk /dev/sda`This will start `fdisk` on disk `sda`. 接下来，通过发出以下指令来切换至分区：`u`<note important>不要通过输入 `c`来禁用DOS兼容模式，否则将破坏分区.</note>之后删除现有分区，并创建一个合乎需求的新分区。多数情况下，你会接受可用最大值，这会将文件系统扩展到你为虚拟磁盘提供的任一可用大小。为此，在`fdisk`提示中输入以下序列：`d n p 1 (accept default 63) (accept default max)`如果你想为额外分区（swap等）留些空间，你可以在最后一个区域输入另一个值。完成后，发出以下指令以保存更改：`w` 创建分区（新磁盘或拓展磁盘）后，创建物理卷：： `pvcreate /dev/sdb1`<note warning>例子中所使用的分区名称为 `/dev/sdb1` 在您的磁盘中，名称和分区号可能不同。您可以使用 `fdisk -l /dev/sdb` 检查分区号.</note>检查新创建的物理卷：`pvdisplay /dev/sdb1`检查可用物理卷。必须有2卷 `zabbix-vg`以及新创建的：`pvs`用新创建的物理卷扩展现存卷组：`vgextend zabbix-vg /dev/sdb1`检查“`zabbix-vg`”卷组 `vgdisplay`现在用自由PE空间扩展您的逻辑卷 `lvextend -l +100%FREE` `/dev/mapper/zabbix-vg-root`调整您的`root`卷（可在系统中完成）：`resize2fs /dev/mapper/zabbix-vg-root`重新启动虚拟机（因为我们修改的分区目前正在使用）。如此，现在文件系统应扩展到分区大小。检查“`/dev/mapper/zabbix-vg-root`”卷：`df -h`

1.9具体格式说明

1.9.1 Xen

为Xen Server转换图像要想通过citrix Xenserver使用Xen映像，则需要您转换磁盘映像。因此需要： 创建一个虚拟磁盘，至少要与图像一般大小 为该磁盘找到UUID xe vdi-list params=all 如果有多个磁盘，它们可以按照在创建虚拟磁盘时分配的那样，通过姓名参数，即姓名标签，进行过滤 导入图像 xe vdi-import filename="image.raw" uuid="*<UUID>*" 使用说明引自Brain Radford的博客.

1.9.2 VMware

在vmdk格式下的图像可直接在VMware 播放器、服务器以及工作站产品中使用。要想在ESX、ESXI以及vSphere中使用，则需要转换为使用VMware转换器

1.9.3 HDD/flash 图像 (raw)

```
dd if=./zabbix_appliance_3.2.0_x86_64.raw of=/dev/sdc bs=4k conv=fdatasync 用闪存、硬盘  
装置取代dev和sdc
```

1.10 已知问题

7. 配置

请使用左侧导航栏来访问“配置”这一章节的内容

1 主机和主机组

什么是“主机”？

一般来讲，Zabbix主机是指你希望监控的那些设备，例如服务器、工作站、交换机等等。

创建主机是使用Zabbix过程中其中一个首要任务。例如，如果你想在一台服务器“X”上监控一些参数，你必须首先创建一个主机称之为“服务器X”，然后就可以寻找并增加监控项到这台“服务器X”上。

主机组是由主机组成的。

以下是 [创建和配置一台主机](#) 的过程 .

1 配置一台主机

概述

按照以下步骤在Zabbix前端创建一台主机：

- 定位到：配置 → 主机
- 在右侧点击 创建主机（或者在主机名上编辑一台已有的主机）
- 在表单中输入主机的相关参数

你可以在已经存在的主机上使用 *Clone* 和 *Full clone* 按钮的形式创建一个新的主机，点击 *Clone* 将保留所有的主机参数和模板链接（保留所有的模板入口），*Full clone* 将额外保留直属实体（应用集、监控项、触发器、视图、底层自动发现规则和Web定制的场景）。

注意：当主机被克隆时，它将保留原来在模板上的所有模板实体。在现有主机级别上（例如更改的监控间隔、修改正则表达式或添加原型到底层发现规则）所做的任何实体修改将不会克隆到新主机，而是与模板一致。

配置

这个 **Host** 标签页包含了通用的主机属性：

属性	描述
<i>Host name</i>	输入一个唯一的主机名。允许有字母、空格、圆点、破折号和下划线。注意：由于 Zabbix agent 运行在你所配置的那台主机上，所以此 agent 配置文件 的参数 Hostname 必须和这里输入的主机名是一致的。在配置 主动代理检查 的过程中参数中的主机名也是需要的。
<i>Visible name</i>	显示名称。如果你设置了这个名称，它将会在列表、拓扑图等地方显示。此属性支持 UTF-8。
<i>Groups</i>	选择主机所属主机组。一个主机必须至少属于一个主机组。
<i>New host group</i>	可以创建一个新的组并和主机关联。如果为空表示忽略。
<i>Interfaces</i>	支持这几种主机接口类型：Agent, SNMP, JMX 和 IPMI. 要增加一个新接口，在 Interfaces 区域点击 Add ，输入 IP/DNS, Connect to 和 Port 信息。注意：用在任何监控项的接口都不能被删除，并且 Remove 链接是灰色的。在 SNMP 接口使用 Use bulk requests 选项来为每个接口启用和禁用SNMP请求的 批量处理。
<i>IP address</i>	主机的IP地址（可选）。
<i>DNS name</i>	主机的DNS名称（可选）。
<i>Connect to</i>	点击对应的按钮告诉Zabbix服务器采用哪种模式从代理端获取数据： <u>IP</u> - 连接到主机的IP地址（推荐） <u>DNS</u> - 连接到主机的DNS名称

<i>Port</i>	TCP/UDP 端口. 默认端口: Zabbix agent 10050, SNMP agent 161 , JMX 12345 , IPMI 623.
<i>Default</i>	选择单选按钮设置默认接口 .
<i>Description</i>	填写主机描述。
<i>Monitored by proxy</i>	主机可以被Zabbix服务器或者Zabbix代理服务器监控: (no proxy) - Zabbix服务器监控 主机 Proxy name - Zabbix代理服务器“代理服务器名称”监控主机

Templates 选项卡允许你将templates 链接到主机。所有实体（监控项，触发器，图表和应用集）将从模板继承。

要链接一个新模板，请开始在*Link new templates* 区域键入，直到匹配键入的模板列表出现。向下滚动选择你希望链接的模板。当所有的模板链接完成后，单击*Add*.

要取消链接模板，请使用*Linked templates*区域的两个选项之一：

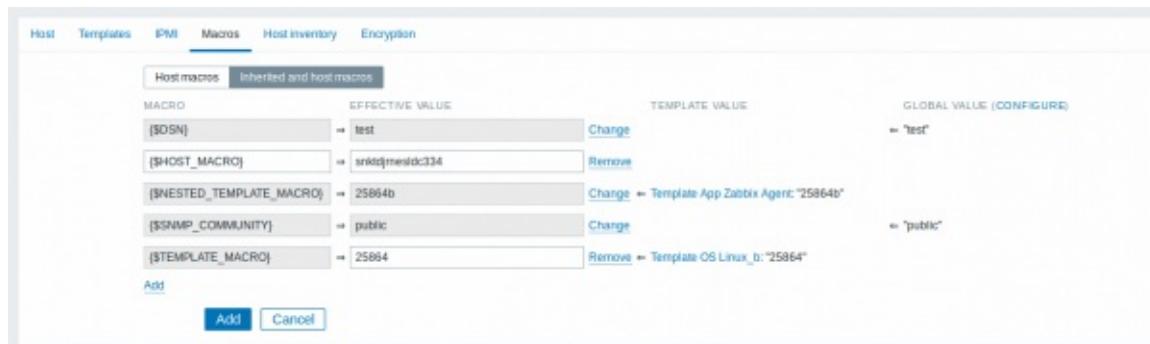
- *Unlink* - 取消链接模板，但保留它的监控项、触发器和图表
- *Unlink and clear* - 取消链接模板并删除所有它的监控项、触发器和图表

列出的模板名可以点击跳转到模板配置表单。

IPMI 选项卡包含 IPMI 管理属性。

参数	描述
<i>Authentication algorithm</i>	选择认证算法。
<i>Privilege level</i>	选择权限级别。
<i>Username</i>	认证用户名。
<i>Password</i>	认证用户密码。

Macros 选项卡允许你定义主机级别的 用户宏。如果你选择了 *Inherited* 和 *host macros* 选项，你也可以在这里查看模板级的宏以及全局宏。那里是为主机定义全部用户宏的地方，用户宏显示解析的值以及来源。



为方便考虑，提供了相应模板和全局宏配置的链接。还可以在主机级别编辑一个模板/全局宏，有效地创建主机上宏的副本。

Host inventory 选项卡允许你为主机手工输入 库存 信息。你还可以选择启用 自动 库存量，或者禁用此主机的库存量。

Encryption 选项卡允许你请求与主机 加密的 链接。

参数	描述
<i>Connections to host</i>	Zabbix服务器或Zabbix代理服务器如何连接到主机上的Zabbix Agent: 无加密(默认) ; 使用PSK(预共享密钥)或者证书。
<i>Connections from host</i>	从主机选择允许的连接类型(例如Zabbix agent和Zabbix Sender)。可以同时选择多种连接类型(对于测试及切换至其他连接类型时有帮助)。默认是"No encryption"。
<i>Issuer</i>	允许颁发证书。证书首先会通过CA(认证机构)认证。如果是有效的,则由CA签名,然后可以使用Issuer字段来进一步限制允许的CA。如果你的Zabbix安装使用多个CA证书,则该字段可以被重复使用。如果这个字段为空,那么任何CA都是可以被接受的。
<i>Subject</i>	允许的证书主题。证书首先通过CA验证。如果它是有效的,由CA签名,则Subject字段可以用于仅允许一个Subject字符串值。如果此字段为空,则接受由配置的CA签名的任何有效证书。
<i>PSK identity</i>	预共享密钥身份字符串。
<i>PSK</i>	预共享密钥(hex-string)。如果Zabbix使用GnuTLS或者OpenSSL库,最大长度:512位十六进制数,如果Zabbix使用mbed(PolarSSL)库,则是64位十六进制(32字节PSK)。 示例: 1f87b595725ac58dd977beef14b97461a7c1045b9a1c963065002c5473194952

配置主机组

要在Zabbix页面配置主机组,请执行以下步骤:

- 进入: Configuration → Host groups
- 在页面右上方单击 Create Group
- 在表单中输入参数

Group name: Europe/Latvia/Riga/Zabbix servers

Hosts: Zabbix server

Other hosts | Group: All

Apache
Discovered host
JB One
MySQL
New host
New template
ODBC discovery
Private
Switch1
Switch2

Apply permissions to all subgroups

Update Clone Delete Cancel

参数	描述
	输入唯一的主机组名称。要创建一个嵌套的主机组,请使用'/'正斜杠分隔符,例

Group name	如 <code>Europe/Latvia/Riga/Zabbix servers</code> . 即使不存在这3个父机组(<code>Europe/Latvia/Riga</code>) , 你也可以创建该组。在这种情况下, 创建父机组取决于使用者; 它们不会自动创建。不允许有正反斜杠, 不支持反斜杠转义'/'。Zabbix 3.2.0支持主机组的嵌套。
Hosts	选择主机、组成员。主机组可能有0个、1个或多个主机。

嵌套主机组的权限

- 当将子主机组创建到现有的父机组时, 对该子进程的[用户组](#) 权限将从父级继承, (例如, 如果 `Riga` 已经存在, 创建 `Riga/Zabbix servers`)
- 将父机组创建到现有的子机组时, 不会设置父级的权限 (例如, 如果 `Riga/Zabbix servers` 已经存在, 创建 `Riga`)

2 资产管理

概述

你可以将联网设备的资产信息保存在Zabbix里。

Zabbix管理页面有一个特殊的*Inventory* 菜单。 但你一开始不会看到任何数据，这里你也不能输入任何资产相关的信息。资产信息是在配置主机时人工录入建立的资产数据，或者通过使用某些自动填充选项完成的录入。

构建资产库

手动模式

当配置一台主机时，在 *Host inventory* 选项卡中，你可以输入设备类型、序列号、位置、负责人等详细信息 - 这些数据将被写入资产库存。

如果主机库存信息中包含URL，并以“http”或“https”开头，则会在*Inventory*中呈现可点击的链接。

自动模式

主机资产也可以自动填充。为了使其正常工作，配置主机时，*Host inventory*选项卡中的清单模式必须设置为*Automatic*。

然后，你可以通过[配置主机监控项](#) 以其值显示任何主机资产字段，指示监控项配置中具有相应属性（称为项目将填充主机资产的字段）的目标字段。

以下是对资产自动发现有用的监控项：

- `system.hw.chassis[full|type|vendor|model|serial]` - 默认是 [full]，需要root权限
- `system.hw.cpu[all|cpunum,full|maxfreq|vendor|model|curfreq]` - 默认是 [all, full]
- `system.hw.devices[pci|usb]` - 默认是 [pci]
- `system.hw.macaddr[interface,short|full]` - 默认是 [all, full]， interface支持正则表达式
- `system.sw.arch`
- `system.sw.os[name|short|full]` - 默认是 [name]
- `system.sw.packages[package,manager,short|full]` - 默认是 [all, all, full]， package支持正则表达式

资产模式选择

可以在主机配置过程中选择资产模式。

根据[Administration → General → Other](#) 中的默认主机资产模式设置，选择新主机的默认清单模式。

对于通过网络发现或自动注册操作添加的主机，可以定义 *Set host inventory mode* 操作，选择手动或自动模式。 此操作将覆盖*Default host inventory mode*设置。

资产清单概述

所有现存资产清单数据的详细信息在 *Inventory* 菜单中可用。

在 *Inventory* → *Overview* 你可以通过资产清单的大量字段获取主机的数量。

在 *Inventory* → *Hosts* 你可以看到所有具有资产清单信息的主机。 单击主机名将以表单显示库存明细。

Host inventory

Overview **Details**

Host name: Zabbix server_1
Visible name: Zabbix server

Agent interfaces	IP address	DNS name	Connect to	Port	Default
	192.168.3.220		IP DNS	10050	<input checked="" type="radio"/>

SNMP interfaces	IP address	DNS name	Connect to	Port	Default
	127.0.0.1		IP DNS	161	<input checked="" type="radio"/>

OS: Linux linux-qvvt 3.11.10-21-default #1 SMP Mon Jul 21 15:28:46 U

Description: Added on 2015-07-28.

Monitoring: Web Latest data Triggers Problems Graphs Screens

Configuration: Host Applications 13 Items 81 Triggers 47 Graphs 12 Discovery 3 Web 1

Cancel

Overview 标签展示：

参数	描述
<i>Host name</i>	主机名称。单击名称将打开一个菜单，其中包含了为主机定义的脚本。主机名显示为橙色图标，表示主机正在维护中。
<i>Visible name</i>	主机的显示名称（如已定义）
<i>Host (Agent, SNMP, JMX, IPMI) interfaces</i>	此区域提供了为主机配置的接口的详细信息。
<i>OS</i>	主机的操作系统资产清单字段（如已定义）。
<i>Hardware</i>	主机硬件清单字段（如已定义）。
<i>Software</i>	主机软件清单字段（如已定义）。
<i>Description</i>	主机描述。
<i>Monitoring</i>	与监控部分的链接，其中包含该主机的这些数据：Web, Latest data, Triggers, Problems, Graphs, Screens.
<i>Configuration</i>	链接到此主机的这些配置部分：Host, Applications, Items, Triggers, Graphs, Discovery, Web. 配置的实体的数量在每个链接之后的括号中列出。

Details选项卡显示填充的所有资产清单字段（不为空）。

资产清单宏

有可用于通知的主机资产清单宏{INVENTORY.*}，例如：

“服务器在{INVENTORY.LOCATION1}有问题，负责人是{INVENTORY.CONTACT1}，电话号码{INVENTORY.POC.PRIMARY.PHONE.A1}。”

关于更多详细信息，请参阅[Macros supported by location](#) 页面。

3 批量更新

概述

有时你可能想要一次更改多个主机的某些属性，那么你可以使用批量更新功能来代替打开每个主机进行编辑。

使用批量更新

要批量更新某些主机，请执行以下操作：

- 在[主机](#)中，在要更新的主机之前选中复选框
- 点击下方的 *Mass update* 按钮
- 跳转到属性对应所需的选项卡(*Host*, *Templates*, *IPMI* 或者 *Inventory*)
- 标记要更新的任何属性的复选框，并为其输入新值

The screenshot shows a software interface for managing hosts. At the top, there's a navigation bar with tabs: Host (which is selected), Templates, IPMI, Inventory, and Encryption. Below the tabs, there are several configuration fields:

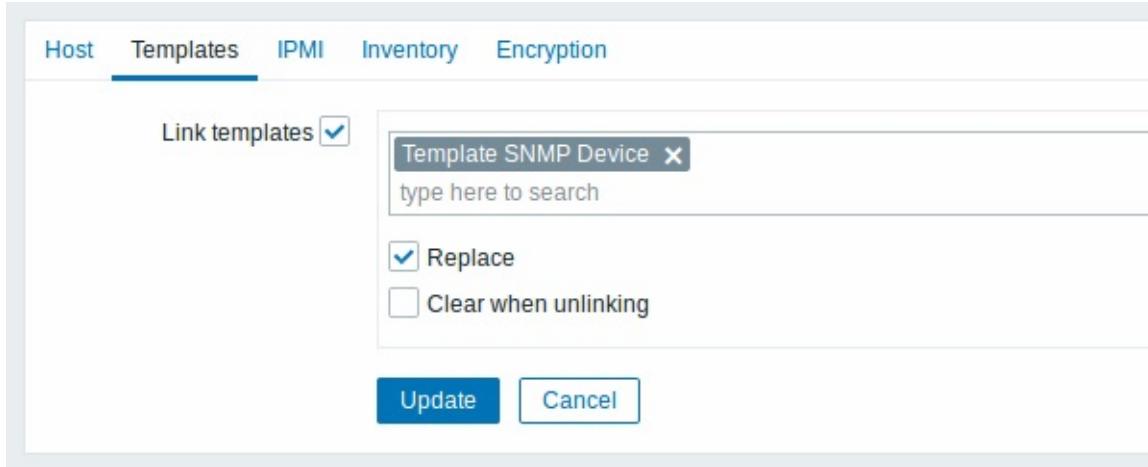
- Replace host groups:** A checkbox is checked, and a dropdown menu labeled "Discovered hosts" is open, showing a search bar with placeholder text "type here to search".
- Add new or existing host groups:** A checkbox is checked, and a dropdown menu with a search bar is shown.
- Description:** A checkbox is unchecked, and the value "Original" is displayed.
- Monitored by proxy:** A checkbox is checked, and a dropdown menu shows "(no proxy)".
- Status:** A checkbox is unchecked, and the value "Original" is displayed.

 At the bottom of the form are two buttons: a blue "Update" button and a white "Cancel" button.

Replace host groups 将从任何现有主机组中删除主机，并将其替换为该字段中指定的主机。

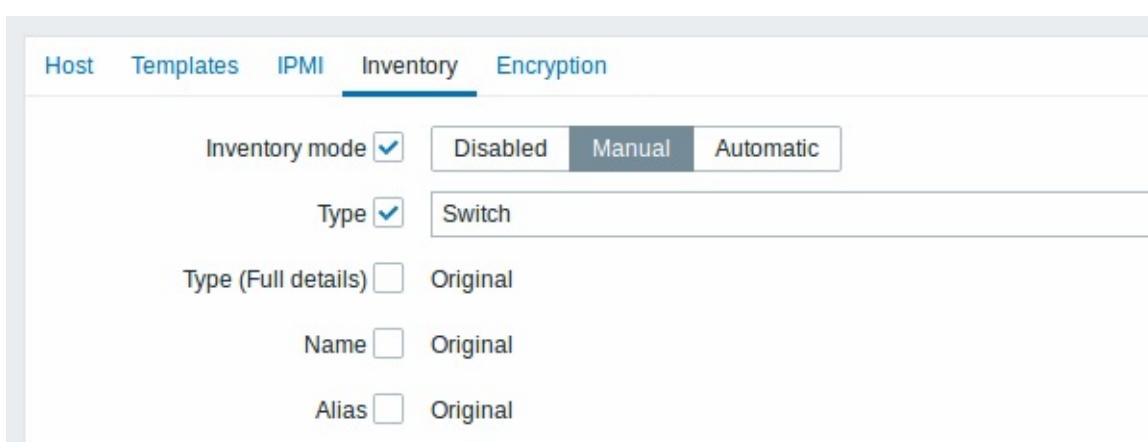
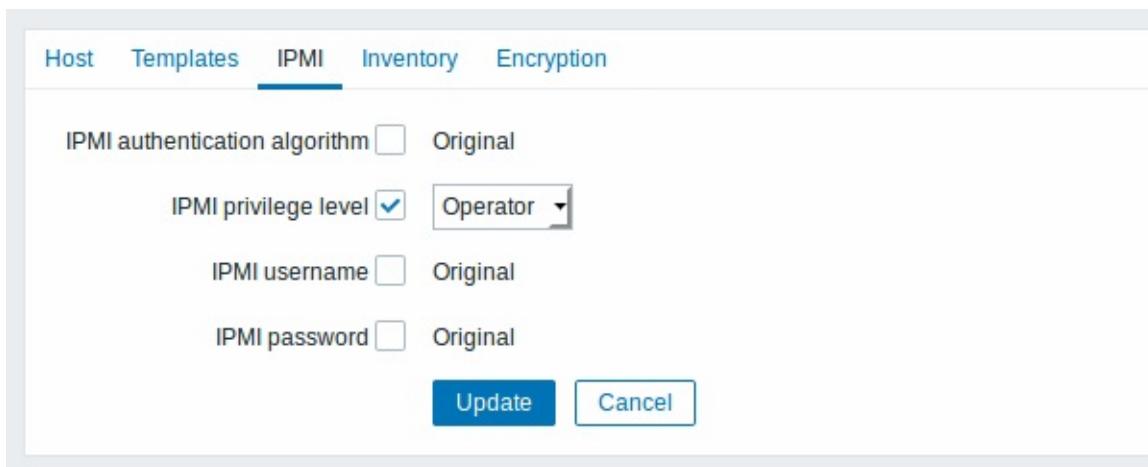
Add new or existing host groups 允许从现有主机组指定其它主机组，或为主机输入全新的主机组。

这两个字段都是自动完成的 - 开始输入它们提供了匹配的主机组的下拉列表。如果主机组是新的，它也会出现在下拉列表中，并在字符串后用(*new*)表示。只需向下滚动即可选择。



要在**Templates**选项卡中更新模板链接，请选择*Link templates*，并在自动填充字段中开始输入模板名称，直到出现一个提供匹配模板的下拉列表。只需向下滚动即可选择要链接的模板。

*Replace*选项将允许链接新模板，同时取消链接到之前与主机链接的任何模板。*Clear when unlinking*选项将不仅可以取消链接任何以前链接的模板，还可以删除所有继承自它们的元素（监控项、触发器等）。



完成所有必需的更改后，单击*Update*，所有选定主机的属性将相应更新。

2 监控项

概述

监控项是从主机收集的数据信息。

配置主机后，你需要添加一些监控项以开始获取实际数据。

一个监控项是一个独立的指标。快速添加多个监控项的一种方法是将一个预定义的模板附加到主机。然而，为了优化系统性能，您可能需要对模板进行微调，使其只有真正需要的监控项被频繁的监控到。

在单个监控项中，你可以指定从主机收集哪些数据。

为此，你可以使用[监控项key](#)。从而，具有名称为system.cpu.load的监控项将收集处理器负载的数据，而名为net.if.in的监控项将收集传入的流量信息。

要用key指定更多的参数，请在key后添加方括号。例如，system.cpu.load[avg5]将返回最近5分钟的处理器负载平均值，而net.if.in[eth0]将显示接口eth0中的流量。

对于所有支持的监控项类型和监控项的Key，请参阅[监控项类型](#)的各个部分。

继续[创建和配置监控项](#)。

1 创建一个监控项

概述

要在Zabbix管理页面创建一个监控项，请执行以下操作：

- 进入到： *Configuration* → *Hosts*
- 在主机所在的行单击 *Items*
- 点击屏幕右上角的*Create item*
- 输入表单中监控项的参数

你也可以通过打开一个监控项，按*Clone* 按钮，然后以不同的名称保存。

配置

Item 选项卡包含了常规监控项属性：

Item Preprocessing

Name	Incoming network traffic on \$1				
Type	Zabbix agent	<input type="button" value="Select"/>			
Key	net.if.in[enp0s3]				
Host interface	192.168.3.220 : 10050				
Type of information	Numeric (unsigned)				
Units	bps				
Update interval	1m				
Custom intervals	Type	Interval	Period		
	Flexible	Scheduling	50s	1-7,00:00-24:00	<input type="button" value="Remove"/>
	Flexible	Scheduling	{\$FLEX_INTERVAL}	{\$FLEX_PERIOD}	<input type="button" value="Remove"/>
	Flexible	Scheduling	wd1-5h9-18		<input type="button" value="Remove"/>
	Flexible	Scheduling	{\$SCHEDULING}		<input type="button" value="Remove"/>
	Add				
History storage period	1w				
Trend storage period	365d				
Show value	As is			show value mappings	
New application					
Applications	<ul style="list-style-type: none"> -None- CPU Filesystems General Memory Network interfaces OS Performance Processes Security ... 				
Populates host inventory field	-None-				
Description					
Enabled	<input checked="" type="checkbox"/>				
	<input type="button" value="Add"/>	<input type="button" value="Cancel"/>			

参数

描述

Name	这里命名监控项名称。可以使用以下宏: \$1, \$2...\$9 - 指的是监控项的第1、2...9个参数例如: \$1上的可用磁盘空间如果监控项的key是 "vfs.fs.size[/, free]", 说明将自动更改为 "Free disk space on /"
Type	监控项类型。参考单个 监控项类型 章节.
Key	监控项key. 可支持的 监控项的key 能够在各个监控项类型中找到。这个key在单个主机中必须是唯一的。如果key的类型是'Zabbix agent'、'Zabbix agent (active)'、'Simple check' 或者 'Zabbix aggregate'，则此key必须被 Zabbix agent 或者 Zabbix server支持。也可以查看: 正确的 key的格式 .
Host interface	选择主机接口。编辑主机级别的监控项时，此字段可用。
Type of information	执行转换后存储在数据库中的数据类型(如果有)。 Numeric (unsigned) - 64位无符号整数 Numeric (float) - 浮点数可以存储负值。允许范围: -99999999999.9999 到 99999999999.9999. 从Zabbix 2.2开始，也支持科学计数值。例如。 $1e+7, 1e-4$ 。 Character - 短文本数据 Log - 具有可选日志相关属性的长文本数据(timestamp, source, severity, logeventid) Text - 长文本数据下表 表格 列出了文本数据的限制。
Units	如果设置了单位符号，Zabbix将在收到数据后再加工处理，并使用设置单位后缀进行显示。默认情况下，如果原始值超过1000，则除以1000，并相应显示。例如，如果设置 bps 并接收到值为881764，则将显示为881.76 Kbps。特殊处理用于B(字节)，Bps(每秒字节数)单位，除以1024. 因此，如果单位设置为B或Bps，Zabbix将显示: 1 为 1B/1Bps1024 为 1KB/1KBps1536 为 1.5KB/1.5KBps如果使用以下与时间相关的单位，则使用特殊处理: unixtime - 转换成"yyyy.mm.dd hh:mm:ss"。要正确转换，接收的值必须是数字(无符号)类型的信息。 uptime - 转换为 "hh:mm:ss" 或者 "N days, hh:mm:ss"例如，如果你收到的值为881764(秒)，则显示为"10天, 04:56:04"s - 转换成"yyy mmm ddd hhh mmm sss ms"; 参数被视为秒数。例如，如果您收到的值为881764(秒)，则显示为"10d 4h 56m"只显示3个主要单位，如"1m 15d 5h"或"2h 4m 46s"。如果没有显示天数，则仅显示两个级别 - "1m 5h" (不显示分钟，秒或毫秒)。 如果该值小于0.001，将被转换成"<1 ms"。请参阅 单位黑名单 .
Update interval (in sec)	每N秒钟检索一次这个项目的新值。注意：如果设置为"0"，则不会轮询该项。但是，如果自定义间隔(灵活/调度)也存在非零值，则会在自定义间隔持续时间期间轮询该项。
Custom intervals	你可以创建用于检查监控项的自定义规则： Flexible - 创建更新间隔的异常(间隔不同的频率) Scheduling - 创建自定义轮询时间表。详细信息请查看 自定义间隔 . 从Zabbix 3.0.0开始支持时间表。注意：不适用于Zabbix Agent的活动监控项。
History storage period(in days)	在数据库中保留详细历史记录的天数，housekeeper将删除较旧的数据。从Zabbix 2.2开始，在Administration → General → Housekeeper 中可以覆盖该值。如果存在全局设置，将显示一条警告消息： History storage period <input type="text" value="1w"/> Overridden by global housekeeping settings (1d) 建议保留最小可能天数的记录值，以减少数据库中的历史记录的大小。你可以保留较长的趋势数据，而不是保存长期的历史数据。参见历史和趋势。请参考 历史与趋势 .
Trend storage period(in days)	在数据库中保留N天的详细历史记录(小时最小，最大，平均值，计数)。housekeeper将删除较旧的数据。从Zabbix 2.2开始，在Administration → General → Housekeeper 中可以覆盖该值。如果存在全局设置，将显示一条警告消息： Trend storage period <input type="text" value="365d"/> Overridden by global housekeeping settings (7d) 注意：保持趋势不适用于非数字数据 - 字符，日志和文本。参考 历史与趋势 .
Show value	将值映射应用于此监控项。值映射不会改变收到的值，仅用于显示数据。它只适用于整数项。例如，“Windows service states”.
Log time format	仅适用于日志类型的监控项。支持的占位符: y: Year (1970-2038) M: Month (01-12) d: Day (01-31) h: Hour (00-23) m: Minute (00-59) s: Second (00-59) 如果留空，则不会解析时间戳。例如，从Zabbix Agent日志文件中考虑以下几行：“23480: 20100328: 154718.045 Zabbix代理启动。Zabbix 1.8.2 (修订 11211)。”它以PID的六个字符位置开始，后跟日期，时间和行的其余部分。该行的日志时

	间格式为“pppppp: yyyyMMdd: hhmmss”。请注意，“p”和“：“字符只是占位符，只能是“yMdhs”。
New application	输入监控项的新应用程序的名称。
Applications	将监控项链接到一个或多个现有应用程序。
Populates host inventory field	你可以选择项目的值将填充的主机资产字段，如果你为主机启用了自动发现模式 资产管理 ，这将会起作用。
Description	输入监控项描述。
Enabled	选中该复选框以启用该项目。

当编辑主机级别上的现有[模板](#)级别的监控项时，多个字段是只读的。你可以使用表单标题中的链接并转到模板级别并在其中进行编辑，但请记住，模板级别上的更改将更改模板链接到的所有主机的项目。

文本数据限制

文本数据限制取决于数据库后台设置：

数据库	信息类型		
Character	Log	Text	
Mysql	255 characters	65536 bytes	65536 bytes
Postgresql	255 characters	65536 characters	65536 characters
Oracle	255 characters	65536 characters	65536 characters
DB2	255 bytes	2048 bytes	2048 bytes

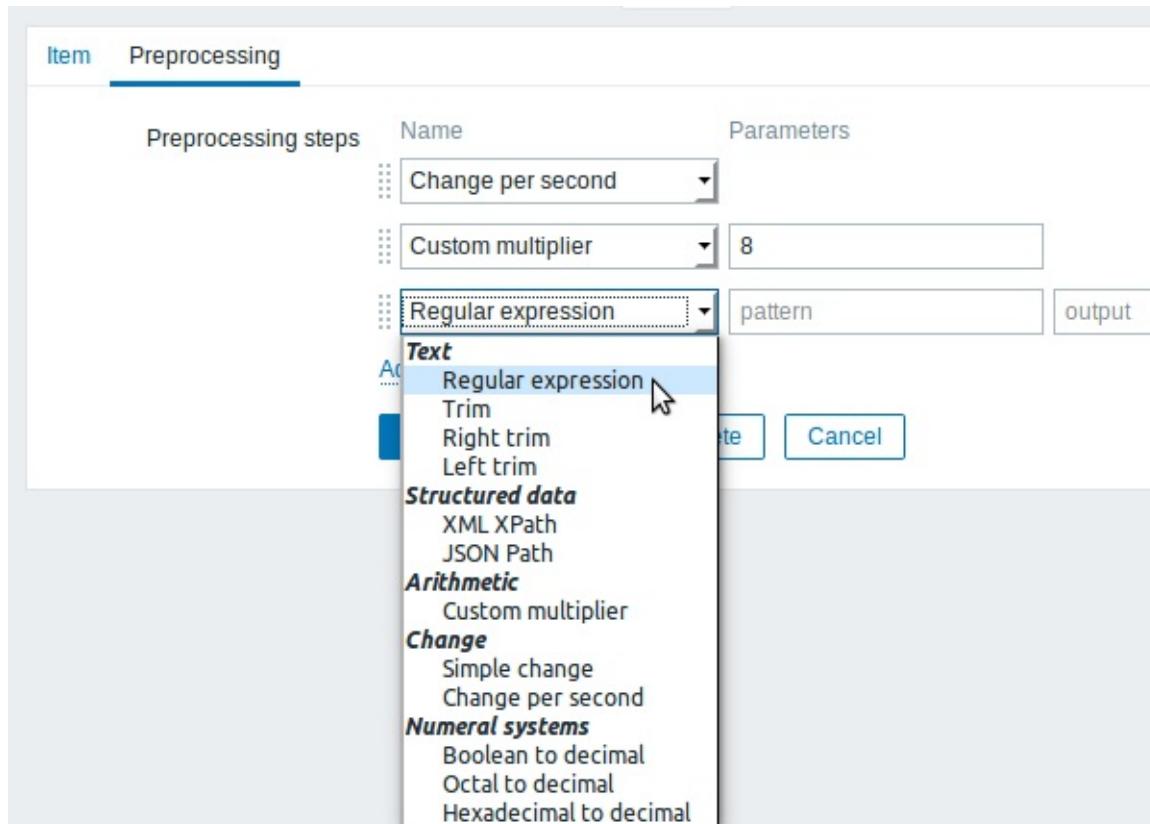
单位黑名单

默认情况下，为监控项指定单位将导致添加乘数前缀 - 例如，单元B的值2048将显示为2KB。对于预定义的硬编码单元列表是不允许的：

- ms
- RPM
- rpm
- %

请注意，小写和大写**rpm** (*rpm*和*RPM*) 字符串都被列入黑名单。

Preprocessing 选项卡允许为接收的值定义转换规则。在将值保存到数据库之前，可以进行一次或多次转换。转换按照定义的顺序执行。所有预处理都由Zabbix服务器完成。



转换	描述
<i>Custom multiplier</i>	将值乘以指定的整数或浮点值。使用此选项将以KB, MBps等接收的值转换为B, Bps, 否则Zabbix无法正确设置前缀(K, M, G等)。从Zabbix 2.2开始, 也支持使用科学符号。例如。 1e + 70。
<i>Right trim</i>	从值的末尾删除指定的字符。
<i>Left trim</i>	从值的起始处删除指定的字符。
<i>Trim</i>	从值的起始和结尾删除指定的字符。
<i>Regular expression</i>	将值与<pattern>正则表达式匹配, 并用<output>替换值。 正则表达式支持用\N序列提取最多10个捕获的组。 \ 参数: pattern - 正则表达式 output - 输出格式化模板。一个\N(其中N = 1 ... 9)转义序列被替换为第N个匹配组。
<i>Boolean to decimal</i>	将值从布尔格式转换为十进制。文本表示被转换为0或1.因此, "TRUE"存储为1, "FALSE"存储为0.所有值都以不区分大小写的方式进行匹配。当前被认为的布尔值值如下: TRUE - true, t, yes, y, on, up, running, enabled, availableFALSE - false, f, no, n, off, down, unused, disabled, unavailable此外, 任何非零数值都被认为是TRUE, 0被认为是FALSE。
<i>Octal to decimal</i>	将八进制格式的值转换为十进制。
<i>Hexadecimal to decimal</i>	将值从十六进制格式转换为十进制。
<i>Delta</i>	计算当前值和上一个值之间的差值。评估为 value-prev_value , 其中 value - current value; prevvalue - 以前收到的值每个项目只允许一个delta操作。
<i>Delta per second</i>	计算每秒速度的值变化(当前值和上一个值的差值)。计算为(value-prevvalue) / (time-prev_time), 其中 value - 当前值; prev_value - 当前收到的值; time - 当前时间戳, prev_time - 以前值的时间戳。这个设置是非常有用的, 以获得每秒不断增长的速度值。如果当前值小于上一个值, Zabbix将丢弃该差异(不存储)并等待另一个值。这有助于正常工作, 例如, 32位SNMP计数器的包装(溢出)。注意: 由于此计算可能产生浮点数, 建议将'Type of information'设置为 <i>_Numeric (float)</i> , 即使传入的原始值是整数。这对于小数部分尤其重要。如果浮点值很大并且可能超过'float'字段长度, 在这种情况下, 整个值可能会丢失, 实际上建议使用 <i>Numeric (无符号)</i> , 因此只会修剪

小数部分。每个监控项只允许一个delta操作。

如果将信息类型设置为数字（无符号），将使用自定义乘数或存储值作为Delta（每秒速度），并且生成的计算值实际为浮点数，则计算值仍被接受为正确的值，通过修剪小数部分并将该值存储为整数。

不支持的监控项

如果由于某种原因无法检索该值，则该监控项可能不被支持。这些监控项仍然以固定的间隔重新检查，可在[管理章节](#)中进行配置。



1 监控项的Key

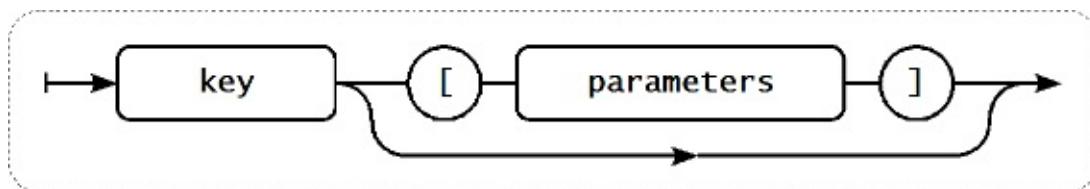
1.1 灵活和非灵活的参数

一个灵活的参数是一个接受参数的参数。 例如，在`vfs.fs.size []`中，星号符号 '*' 表示一个灵活的参数。 '*' 是作为参数传递给其它属性的任何字符串。范例如下：

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

1.2 key的格式

监控项key的格式，包括关键参数，必须遵循语法规则。 以下插图描述了支持的语法。 每个点的允许元素和字符可以通过跟随箭头来确定 - 如果有一些块可以通过线路到达，则允许，如果不是，则不允许。



要构建一个有效的监控项的Key，首先指定Key的名称，然后选择是否具有参数，如果都两个都满足则被执行。

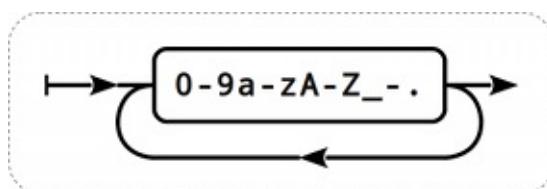
Key名称

Key名本身具有有限的允许字符范围，允许的字符是：

1. `0-9a-zA-Z_-.`

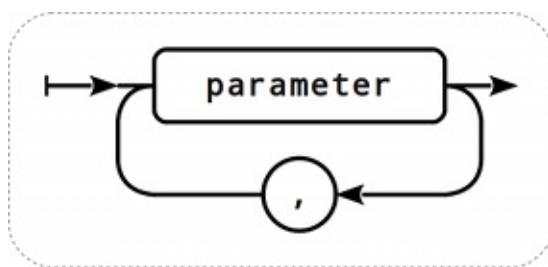
即：

- 所有的数字；
- 所有的小写字母；
- 所有大写字母；
- 下划线；
- 减号；
- 点 .

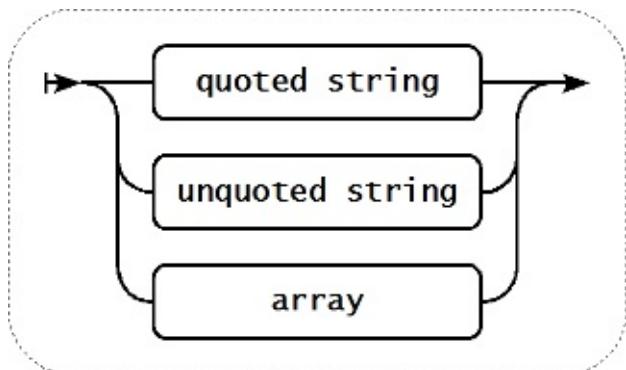


Key参数

监控项的key可以有多个逗号分隔的参数。



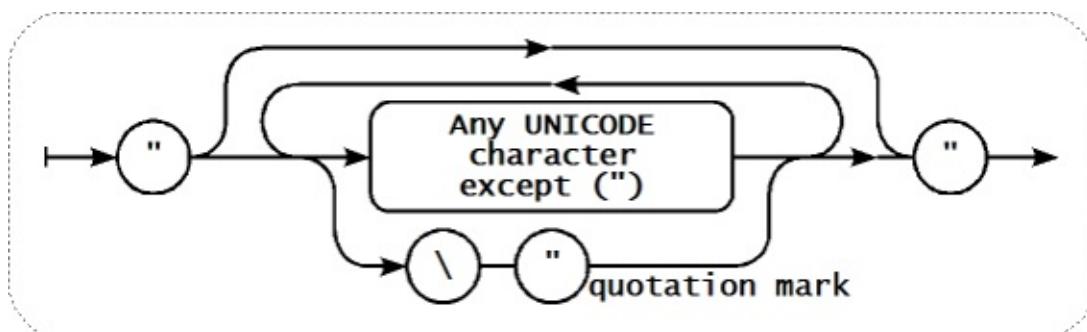
每个key参数可以是带引号、无引号的字符串或数组。



参数也可以为空，此时使用默认值。在这种情况下，如果指定了其它参数，则必须添加对应数量的逗号。例如，key icmping [,, 200, , 500]将指定每ping一次的时间间隔为200毫秒，超时时间为500毫秒，所有其它参数为默认值。

参数 - 带引号

如果key参数带引号，则允许任何Unicode字符，如果包含双引号则需要被反斜杠转义。

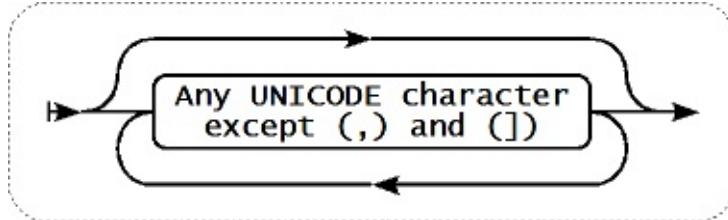


要引用监控项Key参数，请仅使用双引号，不支持单引号。

参数 - 不带引号

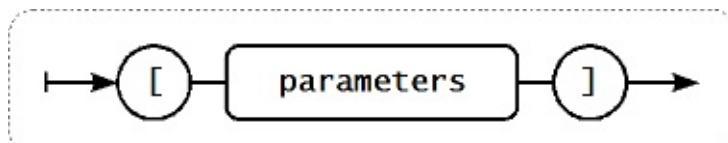
如果key参数是一个不带引号的字符串，除逗号和右方括号 (]) 之外，允许任何Unicode字符。

1 监控项的Key



参数 - 数组

如果key参数是一个数组，它需要包含在方括号中，其中各个参数需要符合多个参数的规则和语法。



2 自定义间隔

概述

可以创建关于选中监控项的自定义时间规则。这两种方式是灵活的时间间隔：允许重新定义默认的更新间隔和调度，从而可以在特定时间或次序执行监控项的检查。

灵活的间隔

灵活的间隔允许重定义特定时间段的默认更新间隔。 灵活的间隔被定义为间隔和周期，其中：

- 间隔 - 指定时间段的更新间隔
- 周期 - 灵活间隔有效的时间段（周期格式请参阅详细说明[时间周期](#)）

可以定义多达七个灵活的时间间隔。如果多个灵活间隔设置有冲突，则在冲突周期中使用最小的间隔值。请注意，如果灵活间隔的最小值为“0”，则不会进行轮询。在灵活间隔之外，使用默认更新间隔。

请注意，如果灵活间隔等于周期的长度，则该监控项将被精确检查一次。如果灵活间隔大于周期，则可能会检查该监控项一次，或者完全不检查该监控项（因此不建议这样配置）。如果灵活间隔小于周期，监控项将至少被检查一次。

如果灵活间隔设置为“0”，则在灵活间隔期间不轮询监控项，并在周期结束后根据默认更新间隔恢复轮询。示例：

间隔	周期	描述
10	1-5, 09:00-18:00	监控项将在工作时间内每10秒检查一次。
0	1-7, 00:00-7:00	监控项不会在夜间检查。
0	7-7, 00:00-24:00	监控项在星期日不会被检查。
60	1-7, 12:00-12:01	监控项将在每天12:00点检查。请注意，这种被用作计划检查的一般性方法从Zabbix 3.0开始建议使用调度间隔来实现。

调度间隔

调度间隔用于在特定时间检查监控项。虽然灵活间隔被设计为重新定义默认监控项的更新间隔，但是调度间隔用于指定独立执行的检查计划。

调度间隔定义为：`md<filter>wd<filter>h<filter>m<filter>s<filter>` 其中：

- `md` - month days
- `wd` - week days
- `h` - hours
- `m` - minutes

- **s** – seconds

`<filter>` 用于指定其前缀的值(days, hours, minutes, seconds) 并被定义为：`[<from>[-<to>]][/<step>]`
`[,<filter>]` 其中：

- 和定义匹配值的范围(包括)。如果忽略，则过滤器匹配 - 范围。如果也被省略，则过滤器匹配所有可能的值。
- 通过该范围定义数字值的跳过。默认情况下，的值为1，这意味着所有定义范围的值都匹配。

虽然过滤器定义是可选的，但必须至少使用一个过滤器。过滤器必须有一个范围或定义的`<step>`值。

如果没有定义低级过滤器，则一个空的filter既与“0”匹配，又匹配所有可能的值。例如，如果省略小时过滤器，仅当分钟和秒的过滤器也被省略则只有“0”小时将匹配，否则空的小时过滤器将匹配所有小时值。

它们各自的过滤器前缀的有效`<from>`和`<to>`值分别为：

前缀	描述	<code><from></code>	<code><to></code>
md	Month days	1-31	1-31
wd	Week days	1-7	1-7
h	Hours	0-23	0-23
m	Minutes	0-59	0-59
s	Seconds	0-59	0-59

`<from>`值必须小于或等于`<to>`值。`<step>`值必须大于或等于1且小于或等于`<to>` - `<from>`。

单个数字月份、小时、分钟和秒值可以前缀为0. 例如`md01-31`和`h/02`是有效间隔，但`md01-031`和`wd01-07`不是。

在Zabbix管理Web端，多个调度间隔以单独的行输入。在Zabbix API中，它们以分号“;”连接成单个字符串作为分隔符。

如果时间匹配了几个间隔，则只执行一次。例如，`wd1h9; h9`将在星期一上午9点执行一次。

示例：

间隔	描述
<code>m0-59</code>	每分钟执行一次
<code>h9-17/2</code>	从9:00开始每2小时执行一次 (9:00, 11:00 ...)
<code>m0,30 or m/30</code>	在每小时的hh:00 和 hh:30执行
<code>m0,5,10,15,20,25,30,35,40,45,50,55 or m/5</code>	每5分钟执行
<code>wd1-5h9</code>	每周一至周五9:00
<code>wd1-5h9-18</code>	每个星期一到星期五在9: 00, 10: 00, ..., 18:00
<code>h9,10,11 or h9-11</code>	每天上午9:00, 10:00和11:00
<code>md1h9m30</code>	每个月的第一天在9:30
<code>md1wd1h9m30</code>	如果是星期一，每个月的第一天在9:30执行

h9m/30	在9:00, 9:30执行
h9m0-59/30	在9:00, 9:30执行
h9, 10m/30	在9:00, 9:30, 10:00, 10:30执行
h9-10m30	在9:30, 10:30执行
h9m10-40/30	在9:10, 9:40执行
h9, 10m10-40/30	在9:10, 9:40, 10:10, 10:40执行
h9-10m10-40/30	在9:10, 9:40, 10:10, 10:40执行
h9m10-40	在9:10, 9:11, 9:12, ... 9:40执行
h9m10-40/1	在9:10, 9:11, 9:12, ... 9:40执行
h9-12, 15	在9:00, 10:00, 11:00, 12:00, 15:00执行
h9-12, 15m0	在9:00, 10:00, 11:00, 12:00, 15:00执行
h9-12, 15m0s30	在上午9时30分, 上午10时30分, 11时30分, 12时30分, 15时30分执行
h9-12s30	在9:00:30, 9:01:30, 9:02:30 ... 12:58:30, 12:59:30执行
h9m/30;h10	在9:00, 9:30, 10:00执行

2 监控项类型

概述

监控项类型包含从系统获取数据的多种方式。每个监控项类型都有一组自己支持的监控项key和所需的参数。

以下监控项类型由Zabbix提供：

- [Zabbix代理检查](#)
- [SNMP代理检查](#)
- [SNMP traps](#)
- [IPMI检查](#)
- [简单检查](#)
- [VMware监控](#)
- [日志文件监控](#)
- [计算监控项](#)
- [Zabbix内部检查](#)
- [SSH检查](#)
- [Telnet检查](#)
- [外部检查](#)
- [汇总检查](#)
- [捕捉器监控项](#)
- [JMX监控](#)
- [ODBC监控](#)

所有监控项类型的详细描述都包含在本章的各个小节中。即使监控项类型提供了大量的数据收集的方式，你还可以通过用户参数或可加载模块进一步扩展数据收集方式。

一些监控检查由Zabbix服务器执行（称作无代理监控），而其它监控检查则需要Zabbix agent或者Zabbix Java网关（使用JMX监视）执行。

如果特定的项目类型需要特定的接口（如IPMI检查需要主机上的IPMI接口），该接口必须存在于主机定义中。

可以在主机定义中设置多个接口：Zabbix agent，SNMP agent，JMX和IPMI。如果一个监控项使用多个接口，它将搜索可用的主机接口（按照以下顺序：agent→SNMP→JMX→IPMI）直到找到连接的第一个匹配的接口。

返回文本的所有监控项（字符，日志，文本信息类型）都可以返回空格（如适用）和值设置为空的字符串。

1 Zabbix 代理

概述

这些检查与Zabbix代理进行通信实现数据的采集。

一共有[被动和主动](#) 两种agent模式。在配置监控项时，你可以选择所需的类型：

- *Zabbix agent* - 被动模式，Zabbix Server向Agent索要数据
- *Zabbix agent (active)* - 主动模式，Agent主动上报数据给Zabbix Server

支持的监控项key

下表提供了可用的Zabbix代理监控项的详细信息。

请参考：

- [不同平台支持的监控项](#)
- [只用于Windows的监控项Key](#)

必填和可选参数

没有尖括号的参数是强制性的。标有尖括号<>的参数是可选的。

Key	描述	返回值	参数
agent.hostname	Agent主机名.	String	
agent.ping	Agent可用性 检查	Nothing - 不可用1 - 可 用	
agent.version	Zabbix Agent的版本	字符串	
kernel.maxfiles	系统支持的打 开文件的最大 数量	整数	
kernel.maxproc	系统支持的最 大进程数	整数	

log[file,<regexp>,<encoding>,<maxlines>,<mode>,<output>,<maxdelay>]	日志文件监控。	Log	file - 日志文件完整路径和名 描述所需模式的正则表达式 enc 码 标识符maxlines - Agent Zabbix服务器或代理的每秒最 数覆盖 zabbix_agentd.conf 的“MaxLinesPerSecond”值 的值: all (默认值), skip - 跳过处理老数据 (仅影响新创 建的监控项)。 output - 可选项, 输 出转义序列替换为匹配的文本, = 1 ... 9) 转义序列被替换为第 (如果N超过捕获组的数量, 则 串)。 maxdelay - 最大延迟 型: float。 值: 0- (默认) 件行; > 0.0-忽略旧行, 以便 在“maxdelay”秒内获取最近分 前请阅读 maxdelay 注释 !
log.count[file,<regexp>,<encoding>,<maxproclines>,<mode>,<maxdelay>]	日志文件监控 中匹配行的数 量。	整数	file - 日志文件完整的路径和 - 正则表达式 encoding - 编 符 maxproclines - Agent将 行数。默认值为 4'MaxLinesPerSecond'在 zab 配置文件 。 mode - 可选的值: a skip - 跳过处理老数据 (仅影 控项)。 maxdelay - 最大延 float. 值: 0 - (默认) 从 志; > 0.0 - 忽略旧行, 以便 在“maxdelay”秒内获取最近分 用前请阅读 maxdelay 参数 的注
logrt[fileregexp,<regexp>,<encoding>,<maxlines>,<mode>,<output>,<maxdelay>]	支持监控轮询 的日志文件。	Log	file_regex - 文件名以及工 义的文件名的绝对路径。 reges 配内容的正则表达式。 encodin 识符maxlines - Agent发送到 器或者Proxy服务器的每秒最大 参数将重写配置文件 zabbix_a 的参数 'MaxLinesPerSecond' 可选的值: all (默认), skip 旧数据 (仅影响新创建的监控项 - 一个可选的输出格式模板。 替换为匹配文本, 而\nN (其中N 转义序列被替换为第N个匹配组 获组的数量, 则为空字符串)。 最大延迟 (秒)。 类型: floa (默认) 不忽略日志文件行; 行, 以便在“maxdelay”秒内获 行。 使用前请阅读 maxdelay 参
logrt.count[file_regex,<regexp>,<encoding>,<maxproclines>,<mode>,<maxdelay>]			

<maxdelay>]	支持对循环日志文件监控中匹配的行数。	整型	file_reexp - 文件名以及意义的文件名的绝对路径。 regexp 配内容的正则表达式。 encoding - 文本文件的字符集。 maxproclines - Agent 大新生成行数。默认值为 4。 MaxLinesPerSecond 定义每秒最大行数。 zabbix_agent 配置文件。 *mode 值: _all (默认), skip - 跳过 (仅影响新创建的监控项)。 maxdelay - 最大延迟 (秒)。类型: float (默认) 不忽略日志文件行; > 0 行, 以便在“maxdelay”秒内获得行。使用前请阅读 maxdelay 参数。
net.dns[<ip>, name, <type>, <timeout>, <count>, <protocol>]	检查DNS服务是否开启。	0 - DNS宕了 (服务器没有响应或DNS解析失败) 1 - DNS正在运行	ip - DNS服务器的IP地址 (默认为空, 在Windows上被忽略) name - 要查询的DNS名称 type - 要查询的类型 (默认为 SOA) timeout (在Windows上忽略) - 请求的超时秒数 (默认为1秒) count (在Windows上忽略) - 请求的尝试次数 (默认为2) protocol - 用于执行DNS查询: udp (默认) 或者 tcp
net.dns.record[<ip>, name, <type>, <timeout>, <count>, <protocol>]	执行一个DNS查询	字符串与所需类型的信息	ip - DNS服务器的IP地址 (默认为空, 在Windows上被忽略) name - 要查询的DNS名称 type - 要查询的类型 (默认为 SOA) timeout (在Windows上忽略) - 请求的超时秒数 (默认为1秒) count (在Windows上忽略) - 请求的尝试次数 (默认为2) protocol - 用于执行DNS查询: udp (默认) 或者 tcp
net.if.collisions[if]	Number of out-of-window collisions.	整型	if - 网卡名称
net.if.discovery	网络接口列表 用于低级发现。	JSON 对象	
net.if.in[if, <mode>]	网卡流入量统计。	整型	if - 网卡名 (Unix); 网卡 MAC 地址 (Linux); 网卡 IPv4 地址 (Windows) mode - 值: bytes - 字节数 (默认) packets - 数据包数量 errors - 错误数量 drop - 丢弃数量

net.if.out[if,<mode>]	网卡流出量统计。	整型	if - 网卡名称 (Unix); 网卡IPv4地址 (Windows) mode - 值: bytes - 字节数(默认) packets - 数据包数量 errors - 错误数量 drop - 被丢弃的数据包数量
net.if.total[if,<mode>]	网卡的进出流量统计信息的总和。	整型	if - 网卡名称(Unix); 网卡IPv4地址(Windows) mode - 值: bytes - 字节数(默认) packets - 数据包数量 errors - 错误数量 drop - 被丢弃的数据包数量
net.tcp.listen[port]	检查此TCP端口是否处于监听状态。	0 - 未监听 1 - 处于监听状态	port - TCP端口号
net.tcp.port[<ip>,port]	检查是否可以将TCP连接到指定的端口。	0 - 不能连接 1 - 可以连接	ip - IP地址 (默认是 127.0.0.1) - 端口号
net.tcp.service[service,<ip>,<port>]	检查服务是否正在运行并接受TCP连接。	0 - 服务down了 1 - 服务正在运行	service - 如下任一服务: <i>smtp, ftp, http, pop, nntp, imap, tcp, https, telnet</i> 信息) ip - IP地址 (默认是 127.0.0.1) port - 端口号 服务端口号)
net.tcp.service.perf[service,<ip>,<port>]	检测TCP服务质量	0 - 服务停止。 seconds - 连接到服务花费的时间 (秒)	service - 如下任一服务: <i>smtp, ftp, http, pop, nntp, tcp, https, telnet</i> (参考 述) ip - IP 地址(默认为 127.0.0.1) port - 端口号 服务端口号)
net.udp.listen[port]			

	检测UDP端口是否处于监听状态。	0 - 未监听。 1 - 处在监听状态。	port - UDP端口
<code>net.udp.service[service,<ip>, <port>]</code>			
	检查服务是否正在运行并能响应UDP请求。	0 - 服务停了。1 - 服务正在运行	service - <i>ntp</i> (参考 详细信息) IP地址 (默认是127.0.0.1) 端口号 (默认使用标准服务端口号)
<code>net.udp.service.perf[service, <ip>,<port>]</code>			
	检测UDP服务的性能	0 - 服务停了 seconds - 等待服务响应的秒数	service - <i>ntp</i> (参考 详细信息) 地址 (默认为 127.0.0.1) (默认使用标准服务端口号)
<code>proc.cpu.util[<name>,<user>, <type>,<cmdline>,<mode>, <zone>]</code>	进程CPU利用率百分比。	浮点型	name - 进程名 (默认为 <i>all processes</i>) user - 用户名 users type - CPU利用率类型 (默认), <i>user</i> , <i>system</i> cmdline - 按命令行过滤 (支持正则表达式) mode - 收集模式: <i>avg1</i> (默认), <i>avg5</i> , <i>avg15</i> zone - 目标区域: <i>cpu</i> (默认), <i>all</i> . 此参数仅在Solaris上支持。从Zabbix 3.0.3开始, 如在Solaris上编译且没有区域支持区域的较新Solaris上运行, 参数为缺省值或当前值, 则代理程序将返回NOTSUPPORTED (该代理程序不支持仅当前区)。但是, 在这种情况下, zone 参数值 <i>all</i> 。
<code>proc.mem[<name>,<user>,<mode>, <cmdline>,<memtype>]</code>	用户进程使用的内存。	整型	name - 进程名 (默认是全部进程) user - 用户名 (默认是 全部用户) mode - 收集模式: <i>_avg</i> , <i>max</i> , <i>min</i> , <i>sum</i> (缺省值) cmdline - 按命令行过滤 (支持正则表达式) memtype - 进程使用的内存类型
<code>proc.num[<name>,<user>, <state>,<cmdline>]</code>	进程数量。	整型	name - 进程名称 (默认是 <i>_processes</i>) user - 用户名 users state - 可选的值: <i>_all</i> (default), <i>run</i> , <i>sleep</i> , <i>zombie</i> cmdline - 按命令行过滤 (支持正则表达式)
<code>sensor[device,sensor,<mode>]</code>			

	硬件传感器读数。	浮点型	device - 设备名称 sensor mode - 可能的值: <i>avg</i> , <i>max</i> ,省略此参数，则会对设备和传感器)。
在Linux 2.6以后的版本上读取 /sys/class/hwmon 请参阅Linux 上 sensor 项目的更详细说明。			
在OpenBSD上读取 <i>hw.sensors</i> MIB文件示例: ⇒ <code>sensor[cpu0,temp0]</code> → CPU的温度 ⇒ <code>sensor["cpu[0-2]",temp,avg]</code> → 前三个CPU温度的平均值从Zabbix 1.8.4开始支持OpenBSD。			
system.boottime	系统启动时间	整数 (Unix时间戳)	
system.cpu.discovery	检测到的 CPU/CPU内核 列表。用于低 级发现。	JSON对象	
system.cpu.intr	设备中断数	整数	
system.cpu.load[<cpu>, <mode>]	CPU负载).	浮点数	cpu - 可能的值: <i>all</i> (default), <i>percpu</i> (总负载除以在线CPU) 可能的值: <i>avg1</i> (一分钟平均值), <i>avg5</i> , <i>avg15</i>
system.cpu.num[<type>]	CPU的数量	整数	type - 可能的值: <i>online</i> (在)
system.cpu.switches	上下文交换的 数量。	整数	
system.cpu.util[<cpu>, <type>, <mode>]	CPU利用率。	浮点型	cpu - <CPU数量> 或者 <i>all</i> 值) type - 可能的值: <i>idle</i> , (默认值), <i>system</i> (Window 值), <i>iowait</i> , <i>interrupt</i> , <i>steal</i> , <i>guest</i> (在Linux kernel 2.6.24 以及以上支持), <i>guest</i> (在 Linux kernels 2.6.33 以 持) mode - 可能的值: <i>avg1</i> (默 认值), <i>avg5</i> , <i>avg15</i>
system.hostname[<type>]			

	系统主机名。	字符串型	type (仅Windows不得在其它 - 可能的值: <i>netbios</i> (默认))
system.hw.chassis[<info>]	机架信息。	字符串	info - 完整的 (默认)、型号 或供应商之一
system.hw.cpu[<cpu>, <info>]	CPU信息	字符串或者整 型	cpu - <CPU数量> 或者 全部 - 可能的值: <i>full</i> (默认), <i>c maxfreq</i> , <i>model</i> 或者 <i>vendo r</i>
system.hw.devices[<type>]	列出PCI或者 USB设备	文本型	type - <i>pci</i> (默认) 或者 <i>us b</i>
system.hw.macaddr[<interface>, <format>]	列出MAC地址	字符串型	interface - <i>all</i> (默认) 否 则表达式 format - <i>full</i> (默 认) 或 <i>short</i>
system.localtime[<type>]	系统时间	整数 - 类型 为 <i>utc</i> 字符串 - 类型 为 <i>local</i>	type - 可能的值: <i>utc</i> - (默 认以来的时间 (1970年1月1日00 UTC), 以秒为单位。 \ <i>local</i> <i>mm-dd, hh:mm:ss.nnn</i> , + 的时间)
system.run[command, <mode>]	在主机上运行 指定的命令。	命令执行的文 本结果1 - 模 式为 <i>nowait</i> (不 管命 令结 果如 何)	command - 要执行的命令 mod 值: <i>wait</i> - 等待执行结束 (默 认), <i>nowait</i> - 不等待
system.stat[resource, <type>]			ent - 该分区有权接收的处理 (<i>float</i>) kthr , < type > - 分 状态的信息: <i>r</i> - 平均可运行内 (<i>float</i>) <i>b</i> - 虚拟内存管理器

	系统信息。	整型或者浮点型	的平均内核线程数 (float) me<type> - 有关虚拟和真实内存信息: avm - 活动虚拟页面 (是自由列表的大小 (整数) page , 关于页面错误和分页活动的信息文件页面输入 (float) fi - 输出 (float) fo - 从调页空分页的页面 po - 页面分页到调 (float) fr - 页面被释放 ((浮点) sr - 通过页面替换算 (float) faults , <type> - 率: in - 设备中断 (float) cs - 内核线程上 (float) cpu , <type> - 处理分比的细分: us - 用户时间 (系统时间 (float) id - 空闲 (float) wa - 系统具有未完成 I/O 请求 (float) 的空闲请求 (float) pc - 消耗的物理处理 (float) ec - 被授权的容量消耗 (float) lbusy - 表示在用户时发生的逻辑处理器利用率的百分比 (float) app - 表示共享池中处理器 (float) disk , <type> - 信息: bps - 表示以每秒字节为单位 (或写入) 驱动器的数据量 (int) 表示发送到物理磁盘/磁带的每秒字节数 (float) 此监控项从Zabbix 1.8.10 版本起支持
system.sw.arch	软件架构信息。	字符串型	
system.sw.os[<info>]	操作系统信息	字符串	info - 可能的值: <i>full</i> (默认) 或者 <i>name</i>
system.sw.packages[<package>, <manager>, <format>]	列出已安装的软件包。	文本	package - <i>_all</i> (默认) 或者 manager - <i>_all</i> (默认) 或者 format - <i>full</i> (默认) 或者 <i>list</i>
system.swap.in[<device>, <type>]	交换 (从设备到内存) 统计。	整型	device - 用于交换的设备 (是 <i>_all</i>) type - 可能的值: <i>count</i> (swapins的数量), <i>sectors</i> (换入的扇区), <i>pages</i> (换入的页). 有关信息请参考 支持的平台

system.swap.out[<device>, <type>]	交换 (从内存到设备) 统计。	整型	device - 用于交换的设备 (是all) type - 可能的值: <i>count</i> (swapouts的数量), <i>sector</i> (域), <i>pages</i> (换出的页). 有信息请参考 支持的平台
system.swap.size[<device>, <type>]	交换空间大小 (以字节为单位) 或百分比 (total)。	Integer - 字节 Float - 百分比	device - 用于交换的设备 (是all) type - 可能的值: <i>free</i> (空间, 默认值), <i>pfree</i> (空闲分比), <i>pused</i> (使用交换空间), <i>total</i> (总交换空间), <i>used</i> (间)
system.uname	系统相关信息	字符串	
system.uptime	系统正常运行时间 (以秒为单位)	整数	
system.users.num	已登录用户数	整数	
vfs.dev.read[<device>, <type>, <mode>]	磁盘读取统计信息。	整数 - 类型 为 <i>_sectors</i> , <i>operations</i> , <i>bytes_Float</i> - 类型 为 <i>_sps</i> , <i>ops</i> , <i>bps</i>	device - 磁盘设备 (默认为 - 可能的值: <i>sectors</i> , <i>ops</i> , <i>bytes</i> , <i>sps</i> , <i>ops</i> , <i>bps</i> 必须因为各种操作系统的默认值不同 <i>ops</i> , <i>bps</i> 表示: <i>sectors</i> , <i>operations</i> , <i>bytes per second</i> respectively. mode - 可能 (1分钟平均值, 默认), <i>avg5</i> , 参数仅支持的 类型 为: <i>sps</i> ,
vfs.dev.write[<device>, <type>, <mode>]			

	磁盘写入统计信息。	整数 - 类型 为sectors, operations, bytes浮点型 - 类型 为sps, ops, bps	device - 磁盘设备 (默认为: 可能的值: sectors, operat bytes, sps, ops, bps因为的默认值有所不同, 所以这个参定。sps, ops, bps 代表: operations, bytes per s respectively. mode - 可能(1分钟平均值, 默认), avg5,参数仅支持这些 类型 : sps,
<code>vfs.dir.size[dir,<regexincl>, <regex_excl>,<mode>, <max_depth>]</code>	目录大小 (以字节为单位)。	整数	dir - 目录的绝对路径 rege 则表达式描述包含的文件名模式包括所有文件; 空字符串是默认值) regex_excl - 正则表达式除的文件名模式 (如果为空不排空字符串是默认值) mode - 可能值:_apparent (默认) - 获得大小, 而不是磁盘利用率(作为 dir), disk - 获取磁盘使用 du -s -B1 dir). 和du命令不同, vfs.dir.size 监控项在计算将隐藏的文件记录帐户 (作为在 dir 内). max_depth - 目录的最大深度。 -1 (默认) - 不会遍历到子目录。
<code>vfs.file.cksum[file]</code>	文件 checksum校验, 由UNIX cksum算法计算实现。	整型	file - 文件全路径
<code>vfs.file.contents[file, <encoding>]</code>	检索文件的内容。	文本	file - 文件全路径 encodin 标识符
<code>vfs.file.exists[file]</code>	检测文件是否存在。	0 - 不存在 1 - 常规文件或到常规存在文件的link (符号或硬)	file - 文件的全路径
<code>vfs.file.md5sum[file]</code>	文件的MD5 checksum。	字符串(文件的MD5哈希)	file - 文件的全路径
<code>vfs.file.regexp[file,regexp, <encoding>,<start line>,<end</code>			

<code>line>, <output>]</code>	查找文件中的字符串。	包含匹配字符串的行，或由可选输出参数指定的行。	<code>file</code> - 文件完整路径 <code>regexp</code> 表达式 <code>encoding</code> - 编码页 <code>line</code> - 满足查询到的第一行的文件的第1行)。 <code>end line</code> - 一行的数量(默认为文件的最后行)。 <code>output</code> - 一个可选的转义序列替换为匹配的 \N (其中 N = 1 ... 9) 转义序列一个匹配组(如果N超过捕获组的字符串)。
<code>vfs.file.regmatch[file, regexp, <encoding>, <start line>, <end line>]</code>	查询文件中的字符串。	0 - 不匹配 1 - 匹配	<code>file</code> - 文件全路径 <code>regexp</code> 表达式 <code>encoding</code> - 编码页 <code>line</code> - 满足查询到的第一行的文件的第1行)。 <code>end line</code> - 一行的数量(默认为文件的最后
<code>vfs.file.size[file]</code>	文件大小(按字节)。	整数	<code>file</code> - 文件全路径
<code>vfs.file.time[file, <mode>]</code>	文件时间信息。	整数(Uinix时间戳)	<code>file</code> - 文件全路径 <code>mode</code> - 可能的值: <code>modify</code> (默认) - 更新时间 - 最后一次访问时间, <code>change</code> 修改时间
<code>vfs.fs.discovery</code>	挂载的文件系统列表。用于低级发现。	JSON对象	
<code>vfs.fs.inode[fs, <mode>]</code>	inode的数量或百分比。	整型 - 针对数量浮点值 - 针对百分比	<code>fs</code> - 文件系统 <code>mode</code> - 可能的(默认), <code>free</code> , <code>used</code> , <code>pfree</code> 分比), <code>pused</code> (已用, 百分比)
<code>vfs.fs.size[fs, <mode>]</code>	磁盘空间, 以字节为单位, 用百分比表示。	整数 - 针对字节浮点 - 针对百分比	<code>fs</code> - 文件系统 <code>mode</code> - 可能的(默认), <code>free</code> , <code>used</code> , <code>pfree</code> 分比), <code>pused</code> (已用, 百分比)
<code>vm.memory.size[<mode>]</code>	内存大小, 以字节为单位, 以百分比表示。	整数 - 用于字节浮点 - 用于百分比	<code>mode</code> - 可能的值: <code>total</code> (<code>active</code> , <code>anon</code> , <code>buffers</code> , <code>exec</code> , <code>file</code> , <code>free</code> , <code>inact</code> , <code>pinned</code> , <code>shared</code> , <code>wired</code> , <code>pused</code> (已用, 百分比), <code>available</code> (可用, 百分比)

web.page.get[host,<path>,<port>]	获取网页内容。	网页源码	host - 主机名 path - HTML (默认是/) port - 端口号 (默认是)
web.page.perf[host,<path>,<port>]	加载完整网页的时间 (以秒为单位)。	浮点	host - 主机名 path - HTML (默认是/) port - 端口号 (默认是)
web.page.regex[host,<path>,<port>,<regexp>,<length>,<output>]	在网页上查找字符串。	匹配的字符串，或由可选的“输出”参数指定	host - 主机名 path - HTML (默认是/) port - 端口号 (默认是) - GNU正则表达式 length - 匹配数 output - 一个可选的输出 \0转义序列替换为匹配的文本， = 1 ... 9) 转义序列被替换为第 (如果N超过捕获组的数量，则：串)。

一个特定于Linux的注意事项。Zabbix Agent必须具有权限读取文件系统/*proc*。来自www.grsecurity.org的内核修补程序限制非特权用户的访问权限。

可用的编码

encoding 参数用于指定处理相应监控项检查的编码，以便获取的数据不会被破坏。有关支持的编码（代码页标识符）的列表，请参阅相应的文档，例如[libiconv](#) (GNU Project) 或Microsoft Windows SDK文档“代码页标识符”的文档。

If empty **encoding** is passed, then UTF-8 (default locale for newer Unix/Linux distributions, see your system's settings) or ANSI with system-specific extension (Windows) is used by default. 如果传递空 **encoding**，则默认使用UTF-8 (用于较新的Unix/Linux发行版的默认语言环境，请参阅系统设置) 或使用具有系统特定扩展名 (Windows) 的ANSI。

关于监控项的一些疑难问题

- 如果与passive agent一起使用，服务器配置中的超时值可能需要高于代理配置文件中的超时值。否则，该监控项可能无法获取任何值，因为服务器请求代理程序首先超时。



只用于Windows的监控项Key

监控项Key

该表描述了你可以使用Zabbix Windows Agent的监控项Key的详细信息。

Key	描述	返回值	参数
eventlog[name,<regexp>,<severity>,<source>,<eventid>,<maxlines>,<mode>]	事件日志监控。	日志	name - 事件日志的名称 regexp - 所需模式 severity - 正则表达式描述的严重程度: "Information", "Warning", "Error", "Critical", "Verbose" (从Zabbix在Windows Vista或更高版本上运行) source - 事件源的正则表达式 (从Zabbix 2.2.0开始支持) eventid - 事件标识符的正则表达式 maxlines - 每秒行数。此参数覆盖zabbix_agentd.win中"MaxLinesPerSecond"的值 mode - 行数模式 (默认), skip - 跳过处理旧数据 (仅影响新)
net.if.list	网卡列表 (包括接口类型, 状态, IPV4地址, 描述)。	文本型	
perfcounter[counter,<interval>]	Windows 性能计数器的值。	整数, 浮点, 字符串或者文本(取决于请求)	counter - 计数器的路径 interval - 监控间隔。The interval 必须在1到900 (包含为1)。
proc_info[process,<attribute>,<type>]			

	关于具体进程的各种信息。	浮点型	process - 进程名 attribute - 所需的表现类型 (当具有相同名称的多个进程存在)
service.discovery	Windows服务列表。用于 低级别发现 。	JSON对象	
service.info[service, <param>]	有关服务的信息。	整型 - 使用的参数为 <code>state</code> , <code>startup_String</code> - 使用的参数是 <code>_displayname</code> , <code>path</code> , <code>user_Text</code> - 使用的参数是 <code>_description</code> 特定的状态:0 - 运行,1 - 暂停,2 - 开始等待,3 - 暂停等待,4 - 继续等待,5 - 停止等待,6 - 停止,7 - 未知,255 - 没有这样的服务专用于启动:0 - 自动的,1 - 自动延迟,2 - 手动,3 - 禁用,4 - 未知	service - 一个真实的服务名称或显示名 理单元所示 param - <code>state</code> (默认), <code>dp</code> <code>path</code> , <code>user</code> , <code>startup</code> 或者 <code>descrip</code>
services[<type>, <state>, <exclude>]		0 - 如果为空文本	type - <code>_all</code> (默认), <code>automatic</code> , <code>m</code> disabledstate - <code>all</code> (默认), <code>stop</code> <code>start_pending</code> , <code>stop_pending</code> , <code>run</code>

	服务列表	<i>continue_pending, pause_pending</i> pausedexclude - 从结果中排除的服务，双引号列出，用逗号分隔，不含空格。	
wmi.get[<namespace>, <query>]			
	执行WMI查询并返回第一个选定的对象。	整型, 浮点, 字符串或者文本(取决于请求)	namespace - WMI名字空间 query - WMI查询语句
vm.vmemory.size[<type>]	虚拟空间大小(以字节计)或百分比(总计)。	整型 - 用于字节 浮点 - 用于百分比	* type - 可能的值: <i>available</i> (虚拟内存可用), <i>available</i> (可用的虚拟内存百分比), 内存, 百分比), <i>total</i> (总虚拟内存, 黑色的虚拟内存)

监控Windows服务

本教程提供了有关设置Windows服务监控的step-by-step说明。以下假设Zabbix服务器和代理已配置并可操作。

Step 1

获取服务名称。

你可以通过转到MMC服务管理单元并提供服务的属性来获取该名称。在“常规”选项卡中，将看到一个名为“服务名称”的字段。下面的值是设置监控项时使用的名称。

例如，如果要监控“workstation”服务，那么你的服务可能是：`lanmanworkstation`。

Step 2

[配置一个监控项](#) 用于监控服务。

监控项`service.info[service,<param>]` 检索有关特定服务的信息。根据你需要的信息，指定 `param` 选项接受以下值: `displayname`, `state`, `path`, `user`, `startup` 或者 `description`. 默认值是 `state` 如果 `param` 没有指定(`service.info[service]`)。

返回值的类型取决于选择的参数：整数用于 `state` 和`startup`; 字符串用于 `displayname`, `path` 和`user`; 文本用于 `description`.

示例：

- Key: `service.info[lanmanworkstation]`
- Type of information: Numeric (unsigned)
- Show value: 选择 Windows服务状态值映射

两个值映射可用Windows service state和Windows service startup type 将数值映射到前端中的文本表

示。

Windows服务的发现

[低级别发现](#) 提供了一种在计算机上为不同实体自动创建项目、触发器和图形的方法。 Zabbix可以自动开始监控机器上的Windows服务，无需知道服务的确切名称，也可以手动创建每个服务的项目。过滤器可用于仅为感兴趣的服务生成实际监控项、触发器和图形。



2 SNMP代理

概述

你可能希望在启用SNMP的设备（如打印机、交换机、路由器或UPS）上使用SNMP监控，并尝试安装完整的操作系统和Zabbix代理是不可能的。

为了能够监控SNMP代理在这些设备上提供的数据，Zabbix服务器初始化配置时必须具有SNMP支持。

仅通过UDP协议执行SNMP检查。

从Zabbix 2.2.3开始，Zabbix服务器和代理守护进程在单个请求中查询多个值的SNMP设备。这会影响各种SNMP监控项（常规SNMP项目，具有动态索引的SNMP项目和SNMP低级别发现），它使SNMP处理更加高效。请参阅下面的[内部批量处理机制](#)，了解内部工作原理。从Zabbix 2.4开始，它还为每个接口提供了一个“使用批量请求”的设置，允许为无法正确处理它们的设备禁用批量请求。

从Zabbix 2.2.7和Zabbix 2.4.2开始，Zabbix服务器和代理守护程序的日志在收到不正确的SNMP响应时会打印类似以下内容：

从Zabbix 2.2开始Zabbix服务器和代理守护程序在执行SNMP检查时使用对应的超时配置参数。另外，在单个不成功的SNMP请求（超时/错误凭据）之后，守护程序不执行重试。之前，SNMP库默认超时和重试值（分别为1秒和5次重试）。

从Zabbix 2.2.8和Zabbix 2.4.2开始，Zabbix服务器和代理守护程序将始终至少重试一次：通过SNMP库的重试机制或通过[内部批量处理机制](#)。

如果监控SNMPv3设备，请确保msgAuthoritativeEngineID（也称为snmpEngineID或“引擎ID”）从不被两台设备共享。根据[RFC 2571](#)（3.1.1.1节），每个设备必须是唯一的。

配置SNMP监控

要通过SNMP开始监控设备，必须执行以下步骤：

步骤 1

使用SNMP接口为设备[创建一个主机](#)。

输入IP地址。你可以使用自动添加一套监控项提供的SNMP模板之一（SNMP设备模板等）。但是，模板可能与主机不兼容。单击Add以保存主机。

SNMP检查不使用代理端口，请忽略它。

步骤 2

找出要监控项目的SNMP字符串（或OID）。

要获取SNMP字符串列表，请使用snmpwalk命令（你应该作为Zabbix安装的一部分安装的net-snmp）或等效工具：

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

这里的“2c”代表SNMP版本，你也可以将其替换为“1”，以在设备上指定SNMP版本为v1。

它会返回给你一个SNMP字符串及其最后一个值的列表。如果不是，那么SNMP 'community' 可能与标准的'public'不同，在这种情况下，请找出它是什么。

然后，你可以浏览列表，直到找到要监控的字符串，例如：如果要监视端口3上的交换机的字节，你将使用此行中的 `IF-MIB :: ifInOctets.3` 字符串：

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

你现在可以使用`snmpget`命令找出'IF-MIB :: ifInOctets.3'的数字OID：

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

请注意，字符串中的最后一个数字是你要监控的端口号。请参考：[动态索引](#)。

它将给你如下所示：

```
1. .1.3.6.1.2.1.2.2.1.10.3 = Counter32: 3472126941
```

重复一遍，OID中的最后一个号码是端口号。

3COM似乎是使用数百个端口号，例如 端口1=端口101，端口3=端口103，但思科使用常规数字，例如。 端口3=3。

一些最常用的SNMP OID，Zabbix将[自动转换为数字表示](#)。

在上面的例子中，值类型是“Counter32”它在内部对应于ASN_COUNTER类型。完整的支持类型包括 ASN_COUNTER, ASN_COUNTER64, ASN_UNSIGNED, ASN_INTEGER, ASN_INTEGER64, ASN_FLOAT, ASN_DOUBLE, ASN_TIMETICKS, ASN_GAUGE, ASN_IPADDRESS, ASN_OCTET_STR 和 ASN_OBJECT_ID (从2.2.8, 2.4.3之后)。这些类型大致对应于`snmpget`输出的“Counter32”, “Counter64”, “UInteger32”, “INTEGER”, “Float”, “Double”, “Timeticks”, “Gauge32”, “IpAddress”, “OCTET STRING”, “OBJECT IDENTIFIER”，但也有可能显示为 “STRING”, “Hex-STRING”, “OID” 或者其它，这取决于显示提示的表达方式。

步骤 3

创建一个监控项。

所以现在回到Zabbix并点击前面创建的SNMP主机的监控项。 根据你在创建主机时是否使用模板，你将拥有与主机相关的SNMP监控项列表或为空。我们假设你要使用`snmpwalk`和`snmpget`刚采集的信息创建监控项，因此单击 Create item。在新的监控项表单中，输入监控项“名称”。确保‘Host interface’字段中有你的交换机/路由器，并将‘Type’字段更改为“SNMPv* agent”。 输入community（通常是public），并将你之前检索到的文本或数字OID输入到‘SNMP OID’字段中，例如：.1.3.6.1.2.1.2.2.1.10.3

输入SNMP ‘Port’为161，‘Key’为有意义的内容，例如。 SNMP-InOctets-Bps。 如果你希望它们与默认值不同，请选择一个自定义乘数（如果需要），并输入‘Update interval’ 和‘History storage period’。 将‘Type of information’设置为数字（浮点数），‘Store value’设置为增量（每秒速度）（重要！否则你将从SNMP设备获取累积值，而不是最新的变化）。

Items

All hosts / Zabbix server Enabled ZBX SNMP JMX IPMI Applications 13 Items 81 Triggers 47

Item Preprocessing

Name	SNMP: InOctets (Bps)
Type	SNMPv3 agent
Key	SNMP-InOctets-Bps
Host interface	127.0.0.1 : 161
SNMP OID	.1.3.6.1.2.1.2.2.1.10.3
Context name	
Security name	
Security level	authPriv
Authentication protocol	MD5 SHA
Authentication passphrase	
Privacy protocol	DES AES
Privacy passphrase	
Port	161
Type of information	Numeric (float)

现在保存监控项，进入*Monitoring → Latest data* 来获取你的SNMP数据！

请注意SNMPv3监控的具体选项：

参数	描述
Context name	输入上下文名称以标识SNMP子网上的监控项。从Zabbix 2.2开始支持SNMPv3监控的上下文名称。用户宏在此字段中解析。
Security name	输入安全名称用户宏在此字段中解析。
Security level	选择安全级别： noAuthNoPriv - 不使用身份验证或隐私协议 AuthNoPriv - 认证协议被使用，但不使用隐私协议 AuthPriv - 使用身份验证和隐私协议
Authentication protocol	选择认证协议 - MD5 或者 SHA .
Authentication passphrase	输入验证密码。用户宏在此字段中解析。
Privacy protocol	选择隐私协议 - DES 或者 AES .
Privacy passphrase	输入隐私密码。用户宏在此字段中解析。

从Zabbix 2.2开始，SHA和AES协议支持SNMPv3认证，除此之外还支持MD5和DES。

示例 1

一般范例：

参数	描述
Community	public
OID	1.2.3.45.6.7.8.0 (or .1.2.3.45.6.7.8.0)
Key	<用作触发器引用的唯一字符串>例如，“my_param”.

请注意，OID可以以数字或字符串形式给出。但是，在某些情况下，字符串OID必须转换为数字表示。实用程序snmpget可用于此目的：

```
1. shell> snmpget -On localhost public enterprises.ucdavis.memory.memTotalSwap.0
```

在配置Zabbix源时指定了-with-net-snmp标志，可以监视SNMP参数。

示例 2

监控正常运行时间：

参数	描述
Community	public
Oid	MIB::sysUpTime.0
Key	router.uptime
Value type	Float
Units	uptime
Multiplier	0.01

批处理的内部工作

从2.2.3开始Zabbix服务器和代理查询SNMP设备在单个请求中的多个值。这会影响多种类型的SNMP监控项：

- 常规SNMP监控项；
- 具有动态索引的SNMP监控项；
- SNMP低级发现规则。

具有相同参数的单个接口上的所有SNMP监控项都将同时进行查询。前两种类型的监控项由轮询器分批采集最多128个监控项，而低级发现规则如前所述单独处理。

在较低级别上，执行查询值的操作有两种：获取多个指定对象和游历OID树。

对于“getting”，GetRequest-PDU最多使用128个变量绑定。对于“walking”，GetNextRequest-PDU用于SNMPv1和GetBulkRequest，“max-repetitions”字段最多128个用于SNMPv2和SNMPv3。

因此，每个SNMP监控项类型的优势如下：

- 常规SNMP项目受益于“getting”的改进；
- 具有动态索引的SNMP监控项受益于“getting”和“walking”改进：“getting”用于索引验证，“walking”用于构建缓存；
- SNMP低级发现规则受益于“walking”的改进。

然而，有一个技术问题，并非所有设备都能够根据请求返回128个值。有些总是给出正确的回应，其它情况则会以“tooBig(1)”错误做出回应，或者一旦潜在的回应超过了一定的限度，则一律不回应。

为了找到最佳数量的对象来查询给定的设备，Zabbix使用以下策略。它在请求中查询“值1”时谨慎开始。如果成功，它会在请求中查询“值2”。如果再次成功，则查询请求中的“值3”，并通过将查询对象的数量乘以1.5来继续，导致以下请求大小的顺序：1, 2, 3, 4, 6, 9, 13, 19, 28, 42, 63, 94, 128。

然而，一旦设备拒绝给出适当的响应（例如，对于42个变量），Zabbix会做两件事情。

首先，对于当前项批次，它将单个请求中的对象数减半，并查询21变量。如果设备存在，那么查询应该在绝大多数情况下work，因为28变量已知可以工作，21显着小于此。然而，如果仍然失败，那么Zabbix会一个一个地回退到查询值。如果在这一点上仍然失败，那么设备必定不会响应（请求大小不是问题）。

Zabbix对后续监控项批次做的第二件事是，从最后一个成功的变量数开始（在我们的示例中为28）继续将请求大小递增1，直到命中。例如，假设最大响应大小为32变量，后续请求的大小将为29, 30, 31, 32, 33。最后一个请求将失败，Zabbix将永远不再发出大小为33的请求。从那时起，Zabbix将为此设备查询最多32变量。

如果大规模查询失败，这个变量数量可能意味着两种可能。设备用于限制响应大小的确切标准是不知道的，但是我们尝试使用变量的数量近似。所以第一种可能性是在一般情况下，这个数量的变量是围绕设备的实际响应大小限制的：有时响应小于限制，有时它大于限制。第二种可能性是，任一方向的UDP数据包丢失了。由于这些原因，如果Zabbix获取失败的查询，它可以减少变量的最大数量，以便更深入地知道设备的合适范围，但从2.2.8开始只有两次。

在上面的例子中，如果32变量查询失败，Zabbix将会将计数减少到31。如果发生这种情况，Zabbix也会将计数减少到30。然而，Zabbix不会将计数减少到30以下，因为它会假设进一步的故障是由于UDP数据包丢失而不是设备的限制。

但是，如果由于其它原因，设备无法正确处理批量请求，并且上述探索式功能不起作用，因此从Zabbix 2.4开始对于允许禁用该设备批量请求的每个接口都有“使用批量请求”设置。

1 动态索引

概述

虽然你可能会在SNMP OID中找到所需的索引号（例如网络接口），但有时你不能完全依赖不变的索引号。

索引号可能是动态的 - 它们可能会随时间而改变，因此你的监控项可能会停止工作。

为了避免这种情况，可以定义一个考虑到索引号改变的可能性的OID。

例如，如果需要检索索引值以匹配Cisco设备上的**GigabitEthernet0/1**接口的**ifInOctets**，请使用以下OID：

```
1. ifInOctets["index", "ifDescr", "GigabitEthernet0/1"]
```

语法

使用OID的特殊语法：

```
<OID of data>["index", "<base OID of index>", "<string to search for>"]
```

参数	描述
OID of data	主OID用于监控项上的数据检索。
index	处理方法。目前支持一种方法： index - 搜索索引，并将其附加到数据OID
base OID of index	该OID将被搜索以获取与该字符串对应的索引值。
string to search for	用于在进行查找时与值精确匹配的字符串。区分大小写。

示例

获取**apache**进程的内存使用率。

如果使用这种OID语法：

```
1. HOST-RESOURCES-MIB::hrSWRunPerfMem["index", "HOST-RESOURCES-MIB::hrSWRunPath", "/usr/sbin/apache2"]
```

索引号将在这里查找：

```
1. ...
2. HOST-RESOURCES-MIB::hrSWRunPath.5376 = STRING: "/sbin/getty"
3. HOST-RESOURCES-MIB::hrSWRunPath.5377 = STRING: "/sbin/getty"
4. HOST-RESOURCES-MIB::hrSWRunPath.5388 = STRING: "/usr/sbin/apache2"
5. HOST-RESOURCES-MIB::hrSWRunPath.5389 = STRING: "/sbin/sshd"
6. ...
```

现在我们有索引5388。索引将附加到此数据OID，以便接收我们感兴趣的值：

```
1. HOST-RESOURCES-MIB::hrSWRunPerfMem.5388 = INTEGER: 31468 KBytes
```

索引查找缓存

当请求动态索引项时，Zabbix检索并缓存base OID下的整个SNMP表用于索引（即使早发现了匹配）。这是为了在另一个监控项稍后引用相同的base OID - Zabbix将在缓存中查找索引，而不是再次查询被监视的主机。请注意，每个轮询器进程使用单独的缓存。

在所有随后的值检索操作中，仅验证找到的索引。如果没有改变将请求结果值；如果已更改，则会重建高速缓存 - 遇到已更改索引的每个轮询器再次建立SNMP索引表。

2 特殊OID

一些最常用的SNMP OID自动转换为Zabbix的数字表示。例如，如果Index被翻译为1.3.6.1.2.1.2.2.1.1，则将ifIndex.0转换为1.3.6.1.2.1.2.2.1.1.0。

该表包含特殊OID的列表。

特殊OID	标识符	描述
ifIndex	1.3.6.1.2.1.2.2.1.1	每个接口的唯一值。
ifDescr	1.3.6.1.2.1.2.2.1.2	包含有关接口信息的文本字符串。该字符串应包括制造商的名称、产品名称和硬件接口的版本。
ifType	1.3.6.1.2.1.2.2.1.3	接口的类型，根据物理/链路协议，在协议栈的网络层“下面”进行快速区分。
ifMtu	1.3.6.1.2.1.2.2.1.4	可以在接口上发送/接收的最大数据报的大小，以八位字节指定。
ifSpeed	1.3.6.1.2.1.2.2.1.5	接口当前带宽的估计，以位/秒为单位。
ifPhysAddress	1.3.6.1.2.1.2.2.1.6	协议层的接口地址在协议栈的“网络层”之下。
ifAdminStatus	1.3.6.1.2.1.2.2.1.7	接口的当前管理状态。
ifOperStatus	1.3.6.1.2.1.2.2.1.8	接口的当前操作状态。
ifInOctets	1.3.6.1.2.1.2.2.1.10	接口上接收的八位字节总数，包括成帧字符。
ifInUcastPkts	1.3.6.1.2.1.2.2.1.11	传送到较高层协议的子网单播报文数量。
ifInNUcastPkts	1.3.6.1.2.1.2.2.1.12	传送到较高层协议的非单播（即子网广播或子网多播）数据包的数量。
ifInDiscards	1.3.6.1.2.1.2.2.1.13	即使没有检测到错误，也被选择丢弃的出栈数据包的数量，以防止它们被传输。丢弃这样的数据包的一个可能的原因是释放缓冲区空间。
ifInErrors	1.3.6.1.2.1.2.2.1.14	包含错误的入栈数据包数量，阻止它们传递到较高层协议。
ifInUnknownProtos	1.3.6.1.2.1.2.2.1.15	通过接口接收到的数据包数量由于未知或不受支持的协议而被丢弃。
ifOutOctets	1.3.6.1.2.1.2.2.1.16	从接口传出的八位字节总数，包括帧字符。
ifOutUcastPkts	1.3.6.1.2.1.2.2.1.17	要求发送更高级别协议的数据包的总数，并且没有寻址到此子层的多播或广播地址，包括丢弃或未发送的数据包。
ifOutNUcastPkts	1.3.6.1.2.1.2.2.1.18	要求发送更高级别协议的数据包的总数，并且被发送到该子层的多播或广播地址，包括丢弃或未发送的数据包。
ifOutDiscards	1.3.6.1.2.1.2.2.1.19	即使没有检测到错误，也被选择丢弃的出栈数据包的数量，以防止它们被传输。丢弃这样的数据包的一个可能的原因是释放缓冲区空间。
ifOutErrors	1.3.6.1.2.1.2.2.1.20	由于错误而无法传输的出栈数据包的数量。
ifOutQLen	1.3.6.1.2.1.2.2.1.21	输出包队列的长度（以包为单位）。

3 SNMP trap

概述

接收SNMP trap与查询支持SNMP的设备相对。

在这种情况下，信息是从支持SNMP的设备发送的，由Zabbix收集或“trapped”。

通常情况下发送trap是发生变化并且代理连接到端口162上的服务器（而不是用于查询的代理端的端口161）。 使用trap可以检测在查询间隔期间发生的一些可能被查询数据丢失的短期问题。

在Zabbix中接收SNMP trap旨在使用snmptrapd和内置机制之一来传递trap到Zabbix - 一个perl脚本或SNMPTT。

接收trap的工作流程：

- **snmptrapd** 收到trap
- snmptrapd将trap传递给SNMPTT或调用Perl接收器
- SNMPTT或Perl trap接收器解析，格式化并将trap写入文件
- Zabbix SNMP trap读取并解析trap文件
- 对于每个trap，Zabbix发现主机接口与接收的trap地址匹配的所有“SNMP trap”监控项。请注意，在匹配期间只使用主机接口中选定的“IP”或“DNS”。
- 对于每个找到的监控项，将trap与“snmptrap[regexp]”中的regexp进行比较。 trap设置为all匹配项的值。如果没有找到匹配的监控项，并且有一个“snmptrap.fallback”监控项，则将trap设置为该值。
- 如果trap未设置为任何监控项的值，Zabbix默认记录不匹配的trap。（这由管理 - >常规 - >其它中的“记录不匹配的SNMP trap (Log unmatched SNMP traps)”配置。）

3.1 配置SNMP Trap

在Web前端配置以下字段用于此监控项类型：

- 你的主机必须具有SNMP接口

在配置→主机中，在主机接口字段中设置具有正确IP或DNS地址的SNMP接口。将每个收到的trap的地址与所有SNMP接口的IP和DNS地址进行比较，以查找相应的主机。

- 配置监控项

在**Key**字段中使用一个SNMP trap Key：

Key	描述	返回值	注释

<code>snmptrap[regexp]</code>		
捕获与 正则表达式 中指定的正则表达式匹配的所有SNMP trap。如果正则表达式未指定，则捕获任何trap。	SNMP trap	该监控项只能用于SNMP接口此监控项从Zabbix 2.0.0开始支持注意：从Zabbix 2.0.5开始，该监控项的参数支持用户宏和全局正则表达式。
<code>snmptrap.fallback</code>		
捕获未被该接口的任何 <code>snmptrap[]</code> 监控项捕获的所有SNMP trap。	SNMP trap	该监控项只能用于SNMP接口。该监控项从Zabbix 2.0.0以后支持

目前不支持多行正则表达式匹配。

将要解析的时间戳的信息类型设置为'Log'。请注意，其它格式（如“数字”）也是可以接受的，但可能需要自定义trap处理程序。

要使SNMP trap监控工作，必须首先正确设置。

3.2 Setting up SNMP trap monitoring

配置 Zabbix 服务器/代理服务器

要读取trap，必须将Zabbix服务器或代理服务器配置为启动SNMP trap进程，并指向由SNMPTT或perl trap接收器写入的trap文件。为此，请编辑配置文件([zabbix_server.conf](#) 或者 [zabbix_proxy.conf](#))：

- StartSNMPTrapper=1
- SNMPTrapperFile=[TRAP FILE]

如果使用systemd参数[PrivateTmp](#)，则该文件不太可能在/tmp下使用。

配置SNMPTT

首先，snmptrapd应该配置为使用SNMPTT。

为了获得最佳性能，应将SNMPTT配置为使用[snmptt.handler-embedded](#)的守护进程，并将trap传递给它。有关SNMPTT的配置，请查看其主页上的说明：<http://snmptt.sourceforge.net/docs/snmptt.shtml>

当SNMPTT配置为接收trap时，配置SNMPTT记录trap：

- 将trap记录到Zabbix将读取的trap文件中：log_enable = 1log_file = [TRAP FILE]
- 设置日期时间格式：date_time_format = %H:%M:%S %Y/%m/%d = [DATE TIME FORMAT]

现在格式化Zabbix的trap来识别它们（编辑snmptt.conf）：

- 每个FORMAT语句应以“ZBXTRAP [address]”开头，其中[address]将与Zabbix上SNMP接口的IP地址和DNS地址进行比较。例如：EVENT coldStart .1.3.6.1.6.3.1.1.5.1 “Status Events”
NormalFORMAT ZBXTRAP \$aA Device reinitialized (coldStart)
- 请参阅下面的SNMP trap格式说明，了解更多信息。

不要使用未知的trap - Zabbix将无法识别它们。未知trap可以通过在snmptt.conf中定义一个常规事件来处理：
EVENT general .* “General event” Normal

配置 Perl trap 接收器

要求：Perl，Net-SNMP使用-enable-embedded-perl编译（默认情况下从Net-SNMP 5.4支持）Perl trap接收器（查找misc/snmptrap/zabbix_trap_receiver.pl）可以直接从snmptrapd将trap传递给Zabbix服务器。配置过程：

- 将perl脚本添加到snmptrapd配置文件（snmptrapd.conf）中，例如：perl do "[FULL PATH TO PERL RECEIVER SCRIPT]"；
- 配置接收器，例如：\$SNMPTrapperFile = '[TRAP FILE]';\$DateTimeFormat = '[DATE TIME FORMAT]'；

如果没有引用脚本名称，snmptrapd将拒绝启动消息，类似：

```
1. Regexp modifiers "/l" and "/a" are mutually exclusive at (eval 2) line 1, at end of line
2. Regexp modifier "/l" may not appear twice at (eval 2) line 1, at end of line
```

SNMP trap 格式

所有定制的perl trap接收器和SNMPTT trap配置必须按以下方式格式化trap：[timestamp] [the trap, part 1] ZBXTRAP [address] [the trap, part 2]，说明

- [timestamp] - 用于日志监控项的时间戳
- ZBXTRAP - 头表示新的trap从此行开始
- [address] - 用于查找此trap的主机的IP地址

注意，“ZBXTRAP”和“[address]”将在处理过程中从消息中删除。如果trap格式化为其它方式，Zabbix也许能意外的解析trap。 trap示例：11:30:15 2011/07/27 .1.3.6.1.6.3.1.1.5.3 Normal “Status Events” localhost - ZBXTRAP 192.168.1.1 Link down on interface 2. Admin state: 1. Operational state: 2This will result in the following trap for SNMP interface with IP=192.168.1.1:11:30:15 2011/07/27 .1.3.6.1.6.3.1.1.5.3 Normal “Status Events” localhost - Link down on interface 2. Admin state: 1.

3.3 系统要求

大文件支持

Zabbix为SNMP trap文件提供了“大文件支持”。Zabbix可以读取的最大文件大小为 2^{63} (8 EiB)。请注意，文件系统可能会对文件大小施加下限。

日志回旋记录

Zabbix不提供任何日志轮换系统（它应由用户处理）。 日志轮换应该首先重命名旧文件，然后才能将其删除，以免丢失trap：

- Zabbix在最后一个已知位置打开trap文件，并转到步骤3
- Zabbix通过比较inode号和定义trap文件的inode号，检查当前打开的文件是否已经旋转。如果没有打开的文件，Zabbix将重置最后一个位置并转到步骤1。

- Zabbix从当前打开的文件中读取数据并设置新的位置。
- 新数据被解析。如果这是旋转的文件，文件将关闭并返回到步骤2。
- 如果没有新的数据，Zabbix sleep 1秒钟，然后回到步骤2。

文件系统

由于Trap文件的执行，Zabbix需要文件系统支持inode来区分文件（该信息由stat()调用获取）。

3.4 设置示例

本示例使用snmptrapd + SNMPTT将陷阱传递给Zabbix服务器。设置：

- **zabbix_server.conf** - 配置Zabbix启动SNMP trap并设置trap文件：
StartSNMPTrapper=1SNMPTrapperFile=/tmp/my_zabbix_traps.tmp
- **snmptrapd.conf** - 添加SNMPTT作为trap处理程序：traphandle default snmptt
- **snmptt.ini** - 配置输出文件和时间格式：log_file = /tmp/my_zabbix_traps.tmpdate_time_format = %H:%M:%S %Y/%m/%d
- **snmptt.conf** - 定义默认trap格式：EVENT general .* "General event" NormalFORMAT ZBXTRAP \$aA \$ar
- 创建一个SNMP监控项测试：Host's SNMP interface IP: 127.0.0.1Key:
snmptrap["General"]Log time format: hh:mm:ss yyyy/MM/dd

结果如下：

- 用于发送trap的命令：snmptrap -v 1 -c public 127.0.0.1 '.1.3.6.1.6.3.1.1.5.3' '0.0.0.0' 6 33 '55' .1.3.6.1.6.3.1.1.5.3 s "teststring000"
- 接收到的trap:15:48:18 2011/07/26 .1.3.6.1.6.3.1.1.5.3.0.33 Normal "General event" localhost - ZBXTRAP 127.0.0.1 127.0.0.1
- 测试监控项的值:15:48:18 2011/07/26 .1.3.6.1.6.3.1.1.5.3.0.33 Normal "General event" localhost - 127.0.0.1

这个简单的例子使用SNMPTT作为traphandle。为了在生产系统上获得更好的性能，请使用嵌入式Perl将trap从snmptrapd传递到SNMPTT或直接传递到Zabbix。

3.5 请参阅

- 基于CentOS的SNMP trap教程zabbix.org



4 IPMI检查

概述

你可以在Zabbix中监控智能平台管理接口（IPMI）设备的运行状况和可用性。要执行IPMI检查，Zabbix服务器必须首先[配置IPMI支持](#)。

IPMI是计算机系统的远程“关闭”或“带外”管理的标准接口。它可以独立于操作系统直接从所谓的“带外”管理卡监视硬件状态，还可以完全启动机器。

Zabbix IPMI监控仅适用于支持IPMI的设备(HP iLO, DELL DRAC, IBM RSA, Sun SSP, 等等).

也可以参考IPMI检查的[已知问题](#)。

配置

主机配置

主机必须配置为处理IPMI检查。必须添加IPMI接口，必须定义相应的IP和端口号，并且必须定义IPMI认证参数。

更多细节请查看[主机定义](#)。

服务器配置

默认情况下，Zabbix服务器未配置为启动任何IPMI轮询，因此任何添加的IPMI监控项将无法正常工作。要更改此选项，请以root身份打开Zabbix服务器配置文件(`zabbix_server.conf`)，并查找以下行：

```
1. # StartIPMIPollers=0
```

取消注释，并设置poller计数为3，如下：

```
1. StartIPMIPollers=3
```

保存文件，然后重新启动`zabbix_server`。

监控项配置

配置主机级别的[监控项](#)时：

- 对于主机接口，选择IPMI IP和端口
- 选择'IPMI agent'作为类型
- 指定IPMI传感器（例如在Dell Poweredge上的 'FAN MOD 1A RPM' ）
- 在主机中输入唯一的监控项key（例如，`ipmi.fan.rpm`）
- 在这种情况下，选择相应的信息类型（“Numeric (float)”，对于离散传感器：“Numeric (unsigned)”），单位（最可能的“rpm”）和任何其它必需监控项属性

超时和会话终止

IPMI消息超时和重试计数在OpenIPMI库中定义。由于目前OpenIPMI的设计，无论在接口还是监控项级别都不能在Zabbix中使这些值进行配置。

LAN的IPMI会话不活动超时时间为60 +/- 3秒。目前无法使用OpenIPMI定期发送激活会话命令。如果没有从Zabbix到特定BMC的IPMI项检查超过在BMC中配置的会话超时，则超时超时后的下一次IPMI检查将由于单个消息超时、重试或接收错误而超时。之后，打开一个新的会话，并启动BMC的完全重新扫描。如果要避免BMC的不必要的rescans，建议将IPMI监控项轮询间隔设置为低于BMC中配置的IPMI会话不活动超时。

关于IPMI离散传感器的注意事项

要在主机上找到传感器启动Zabbix服务器，启用**DebugLevel=4**。等待几分钟，并在Zabbix服务器日志文件中查找传感器发现记录：

```

1. $ grep 'Added sensor' zabbix_server.log
2. 8358:20130318:111122.170 Added sensor: host:'192.168.1.12:623' id_type:0 id_sz:7 id:'CATERR' reading_type:0x3
   ('discrete_state') type:0x7 ('processor') full_name:'(r0.32.3.0).CATERR'
3. 8358:20130318:111122.170 Added sensor: host:'192.168.1.12:623' id_type:0 id_sz:15 id:'CPU Therm Trip'
   reading_type:0x3 ('discrete_state') type:0x1 ('temperature') full_name:'(7.1).CPU Therm Trip'
4. 8358:20130318:111122.171 Added sensor: host:'192.168.1.12:623' id_type:0 id_sz:17 id:'System Event Log'
   reading_type:0x6f ('sensor specific') type:0x10 ('event_logging_disabled') full_name:'(7.1).System Event Log'
5. 8358:20130318:111122.171 Added sensor: host:'192.168.1.12:623' id_type:0 id_sz:17 id:'PhysicalSecurity'
   reading_type:0x6f ('sensor specific') type:0x5 ('physical_security') full_name:'(23.1).PhysicalSecurity'
6. 8358:20130318:111122.171 Added sensor: host:'192.168.1.12:623' id_type:0 id_sz:14 id:'IPMI Watchdog'
   reading_type:0x6f ('sensor specific') type:0x23 ('watchdog_2') full_name:'(7.7).IPMI Watchdog'
7. 8358:20130318:111122.171 Added sensor: host:'192.168.1.12:623' id_type:0 id_sz:16 id:'Power Unit Stat'
   reading_type:0x6f ('sensor specific') type:0x9 ('power_unit') full_name:'(21.1).Power Unit Stat'
8. 8358:20130318:111122.171 Added sensor: host:'192.168.1.12:623' id_type:0 id_sz:16 id:'P1 Therm Ctrl %'
   reading_type:0x1 ('threshold') type:0x1 ('temperature') full_name:'(3.1).P1 Therm Ctrl %'
9. 8358:20130318:111122.172 Added sensor: host:'192.168.1.12:623' id_type:0 id_sz:16 id:'P1 Therm Margin'
   reading_type:0x1 ('threshold') type:0x1 ('temperature') full_name:'(3.2).P1 Therm Margin'
10. 8358:20130318:111122.172 Added sensor: host:'192.168.1.12:623' id_type:0 id_sz:13 id:'System Fan 2'
    reading_type:0x1 ('threshold') type:0x4 ('fan') full_name:'(29.1).System Fan 2'
11. 8358:20130318:111122.172 Added sensor: host:'192.168.1.12:623' id_type:0 id_sz:13 id:'System Fan 3'
    reading_type:0x1 ('threshold') type:0x4 ('fan') full_name:'(29.1).System Fan 3'
12. 8358:20130318:111122.172 Added sensor: host:'192.168.1.12:623' id_type:0 id_sz:14 id:'P1 Mem Margin'
    reading_type:0x1 ('threshold') type:0x1 ('temperature') full_name:'(7.6).P1 Mem Margin'
13. 8358:20130318:111122.172 Added sensor: host:'192.168.1.12:623' id_type:0 id_sz:17 id:'Front Panel Temp'
    reading_type:0x1 ('threshold') type:0x1 ('temperature') full_name:'(7.6).Front Panel Temp'
14. 8358:20130318:111122.173 Added sensor: host:'192.168.1.12:623' id_type:0 id_sz:15 id:'Baseboard Temp'
    reading_type:0x1 ('threshold') type:0x1 ('temperature') full_name:'(7.6).Baseboard Temp'
15. 8358:20130318:111122.173 Added sensor: host:'192.168.1.12:623' id_type:0 id_sz:9 id:'BB +5.0V'
    reading_type:0x1 ('threshold') type:0x2 ('voltage') full_name:'(7.1).BB +5.0V'
16. 8358:20130318:111122.173 Added sensor: host:'192.168.1.12:623' id_type:0 id_sz:14 id:'BB +3.3V STBY'
    reading_type:0x1 ('threshold') type:0x2 ('voltage') full_name:'(7.1).BB +3.3V STBY'
17. 8358:20130318:111122.173 Added sensor: host:'192.168.1.12:623' id_type:0 id_sz:9 id:'BB +3.3V'
    reading_type:0x1 ('threshold') type:0x2 ('voltage') full_name:'(7.1).BB +3.3V'
18. 8358:20130318:111122.173 Added sensor: host:'192.168.1.12:623' id_type:0 id_sz:17 id:'BB +1.5V P1 DDR3'
    reading_type:0x1 ('threshold') type:0x2 ('voltage') full_name:'(7.1).BB +1.5V P1 DDR3'
19. 8358:20130318:111122.173 Added sensor: host:'192.168.1.12:623' id_type:0 id_sz:17 id:'BB +1.1V P1 Vccp'
    reading_type:0x1 ('threshold') type:0x2 ('voltage') full_name:'(7.1).BB +1.1V P1 Vccp'
20. 8358:20130318:111122.174 Added sensor: host:'192.168.1.12:623' id_type:0 id_sz:14 id:'BB +1.05V PCH'
    reading_type:0x1 ('threshold') type:0x2 ('voltage') full_name:'(7.1).BB +1.05V PCH'
```

要解码IPMI传感器类型和状态，请在<http://www.intel.com/content/www/us/en/servers/ipmi/ipmi-specifications.html>(在撰写本文时，最新的文件

是<http://www.intel.com/content/dam/www/public/us/documents/product-briefs/second->

[gen-interface-spec-v2.pdf](#))获取IPMI 2.0规范的副本

开始的第一个参数是“reading_type”。从规范中使用“表42-1，事件/读取类型代码范围”来解码“reading_type”代码。我们示例中的大多数传感器都有“reading_type: 0x1”，这意味着是“threshold”传感器。“表42-3，传感器类型代码”表示：“类型：0x1”表示温度传感器；“类型：0x2” - 电压传感器；“类型：0x4” - 风扇等阈值传感器有时称为“模拟”传感器，因为它们测量连续参数，如温度，电压，每分钟转数。

另一个例子 - 一个带有“read_type: 0x3”的传感器。“表42-1，事件/读取类型代码范围”表示读取类型代码02h-0Ch表示“通用离散”传感器。离散传感器具有多达15个可能的状态（换句话说-最多15个有意义的位）。例如，对于具有“type: 0x7”的传感器“CATERR”，“表42-3，传感器类型代码”表示此类型“处理器”，各个位的含义是：00h（最低有效位）- IERR；01h - 散热等。

在我们的示例中有几个传感器具有“reading_type: 0x6f”。对于这些传感器，“表42-1，事件/读取类型代码范围”建议使用“表42-3，传感器类型代码”来解码位的含义。例如，传感器“Power Unit Stat”的类型为“0x9”，表示“Power Unit”。Offset 00h表示“PowerOff / Power Down”。换句话说，如果最低有效位为1，则服务器断电。为了测试这个位，可以使用band与掩码1的功能。触发表达式可能就像{www.zabbix.com:Power Unit Stat.band (#1,1)} = 1警告服务器关机。

关于OpenIPMI-2.0.16, 2.0.17, 2.0.18和2.0.19中离散传感器名称的注释

OpenIPMI-2.0.16, 2.0.17和2.0.18中的离散传感器的名称通常在附近附加一个额外的“0”（或其它数字或字母）。例如，当ipmitool和OpenIPMI-2.0.19将传感器名称显示为“PhysicalSecurity”或“CATERR”时，在OpenIPMI-2.0.16, 2.0.17和2.0.18中，名称分别为“PhysicalSecurity0”或“CATERR0”。

当使用OpenIPMI-2.0.16, 2.0.17和2.0.18配置IPMI项目时，请在IPMI代理监控项的IPMI传感器字段中使用以“0”结尾的名称。当你的Zabbix服务器升级到使用OpenIPMI-2.0.19（或更高版本）的新Linux发行版时，具有这些IPMI离散传感器的监控项将变为“不支持”。你必须更改其IPMI传感器名称（最后删除“0”），并等待一段时间才能再次转为“Enabled”。

关于阈值和离散传感器同时可用的注意事项

一些IPMI代理提供了相同名称的阈值传感器和离散传感器。在2.2.8和2.4.3之前的Zabbix版本中，选择了第一个提供的传感器。从2.2.8和2.4.3版本以后，偏向于阈值传感器。

连接终止注意事项

如果不执行IPMI检查（由于任何原因：所有主机IPMI监控项禁用/不支持、主机已禁用/已删除、主机维护等），IPMI连接将从Zabbix服务器或代理服务器终止3到4小时，具体时间取决于Zabbix服务器/代理服务器何时启动。

5 简单检查

概述

简单的检查通常用于远程无代理监控服务。

请注意，Zabbix代理不需要简单的检查。Zabbix服务器/代理服务器负责处理简单的检查（使外部连接等）。

使用简单检查的例子：

```
1. net.tcp.service[ftp,,155]
2. net.tcp.service[http]
3. net.tcp.service.perf[http,,8080]
4. net.udp.service.perf[ntp]
```

简单检查监控项配置中的用户名和密码字段通常用于VMware监控项目，否则忽略。

被支持的简单检查

支持的简单检查列表：

请参考：

- [VMware监控项Key](#)

Key	描述	返回值	参数	注释
icmpping[<target>,<packets>,<interval>,<size>,<timeout>]	主机可通过ICMP ping访问。	0 - ICMP ping失败 1 - ICMP ping成功	target - 主机IP或者DNS名 packets - 数据包数量 interval - 连续数据包之间的时间（以毫秒为单位） size - 数据包大小（以字节为单位） timeout - 超时（以毫秒为单位）	示例：⇒ icmpping[,4]如果返回四个的至少一个数据包该项将返回1。可参考： 黑白表 。
icmppingloss[<target>,<packets>,<interval>,<size>,<timeout>]	丢失数		target - 主机IP或者DNS名 packets - 数据包数量 interval -	

	据包的百分比。	Float.	连续数据包之间的时间(以毫秒为单位) size - 数据包大小(以字节为单位) timeout - 超时(以毫秒为单位)	可参考: 默认值表 。
<code>icmppingsec[<target>, <packets>, <interval>, <size>, <timeout>, <mode>]</code>				
	ICMP ping响应时间(以秒为单位)。	Float.	target - 主机IP或者DNS名 packets - 数据包数量 interval - 连续数据包之间的时间(以毫秒为单位) size - 数据包大小(以字节为单位) timeout - 超时(以毫秒为单位) mode - 可能的值: <i>min</i> , <i>max</i> , <i>avg</i> (default)	如果主机不可用(达到超时该监控项将返回0。如果返于0.0001秒, 该值将设置0.0001秒。可参考: 默认值表)
<code>net.tcp.service[service, <ip>, <port>]</code>	检查服务是否正在运行并接受TCP连接。	0 - 服务停止 1 - 服务正在运行	service - 可能的值: <i>ssh</i> , <i>ldap</i> , <i>smtp</i> , <i>ftp</i> , <i>http</i> , <i>pop</i> , <i>nntp</i> , <i>imap</i> , <i>tcp</i> , <i>https</i> , <i>telnet</i> (参考 细节) ip - IP地址或者DNS名(默认使用主机IP/DNS) port - 端口号(默认使用标准服务端口号)。	示例: ⇒ <code>net.tcp.service[ftp]</code> → 可用于测试TCP端口45_服务器的可用性。请注意, 使用服务指示端口是必需的。\\查可能会产生系统守护程序中额外的其它消息(通常SMTP和SSH会话)。\\检查议(如993端口上的IMAP到995上的POP)当前不受支持。解决方法, 请使用 <code>net.tcp.service[tcp, <ip>, port]</code> 进行检查。 <i>http</i> 和 <i>telnet</i> 服务从Zabbix开始支持。
<code>net.tcp.service.perf[service, <ip>, <port>]</code>	检查TCP服务的性能。	0 - 服务不可用 seconds - 连接到服务时花费的秒数	service - 可能的值: <i>ssh</i> , <i>ldap</i> , <i>smtp</i> , <i>ftp</i> , <i>http</i> , <i>pop</i> , <i>nntp</i> , <i>imap</i> , <i>tcp</i> , <i>https</i> , <i>telnet</i> (参考 详细信息) ip - IP地址或者DNS名(默认使用主机IP/DNS)	示例: ⇒ <code>net.tcp.service.perf[http]</code> → 可以用来测试SSH服务器响应速度。请注意, 使用端口是必需的。目前只查加密协议(如端口993上IMAP或端口995上的POP)解决方法, 请使用 <code>net.tcp.service.perf[<ip>, port]</code> 进行如下检

		主机 IP/DNS) port - 端口号 (默认使用标准服务端口号)。	查。https 和 telnet , Zabbix 2.0开始支持。在 Zabbix 2.0之前调用 tcpperf。
net.udp.service[service,<ip>,<port>]	检查服务是否正在运行并响应UDP请求。	0 - 服务停了1 - 服务正在运行	service - 可能的值: _ntp (参考 详细信息) ip - IP地址或DNS名称 (默认使用主机IP/DNS) port - 端口号 (默认使用标准服务端口号)。 示例: ⇒ net.udp.service[ntp] → 可用于测试UDP端口45_务的可用性。这个监控项从Zabbix 3.0开始支持, 但于以前版本的 net.tcp.service[] 监控可以使用ntp服务。
net.udp.service.perf[service,<ip>,<port>]	检查 UDP 服务的性能。	0 - 服务停止 seconds - 等待服务响应的秒数	service - 可能的值: <i>ntp</i> (参考 详细信息) ip - IP地址或DNS名称 (默认使用主机IP/DNS) port - 端口号 (默认使用标准服务端口号)。 示例: ⇒ net.udp.service.perf → 可用于测试NTP服务的响应时间。从Zabbix 3.0以后支持项, 但是在以前的版本中服务可用于 net.tcp.service[] 监控

超时处理

Zabbix将不会处理比Zabbix服务器/代理服务器配置文件中定义的超时秒数更长的简单检查。

ICMP ping

Zabbix使用外部实用程序**fping**来处理ICMP ping。

该实用程序不是Zabbix发行版的一部分, 必须另外安装。如果该实用程序丢失, 那么其权限或其位置与Zabbix服务器/代理服务器配置文件 (“FpingLocation”参数) 中设置的位置不匹配, 则不会处理ICMP ping (**icmpping**, **icmppingloss**, **icmppingsec**)。

fping必须由用户执行Zabbix守护程序运行as和setuid root。以root用户身份运行这些命令, 以便设置正确的权限:

- ```
1. shell> chmod 4710 /usr/sbin/fping
2. shell> chown root:zabbix /usr/sbin/fping
```

还要检查一下, 如果用户zabbix属于组zabbix, 则运行:

- ```
1. shell> groups zabbix
```

如果不是通过如下命令添加:

```
1. shell> usermod -a -G zabbix zabbix
```

ICMP检查参数的默认值、限制值和描述值：

参数	单位	描述	Fping 的标志	默认 设置	允许的 限制by Zabbix
fping	Zabbix	最小	最大		
packets	number	到目标的请求数据包的数量	-C	3	1 10000
interval	milliseconds	在连续数据包之间等待的时间	-p	1000	20 无限
size	bytes	数据包大小(以字节为单位)\x86上的56个字节,\x86_64上的68个字节	-b	56 or 68	24 65507
timeout	milliseconds	最后一个包发送后等待的超时(受“-C”标志影响)	-t	500	50 无限

警告：根据平台和版本，fping默认值可能有所不同 - 如果有疑问，请检查fping文档。

Zabbix将IP地址通过3个icmpping*key中的任一个写入临时文件，然后将其传递给fping。如果监控项具有不同的Key参数，则只有具有相同Key参数的参数被写入到单个文件。写入单个文件的所有IP地址将通过fping并行检查，因此Zabbix icmp pinger进程将花费固定的时间来忽略文件中的IP地址数量。



1 VMware监控项Key

监控项Key

该表提供了可用于监视VMware环境的简单检查的详细信息。

Key	描述	返回值	参数
vmware.cluster.discovery[<url>]	发现VMware集群。	JSON对象	url - VMware服务的URL
vmware.cluster.status[<url>, <name>]	VMware集群状态	整数:0 - 灰色;1 - 绿色;2 - 黄色;3 - 红色	url - VMware服务URLname - VMware集群名称
vmware.eventlog[<url>]	VMware事件日志	Log	url - VMware服务URL
vmware.fullname[<url>]	VMware服务全名称	字符串	url - VMware服务URL
vmware.hv.cluster.name[<url>, <uuid>]	VMware管理层集群名	字符串	url - VMware服务URLuuid - VMware虚拟管理层主机名
vmware.hv.cpu.usage[<url>, <uuid>]	VMware虚拟机管理程序处理器使用情况(Hz)。	整型	url - VMware服务URLuuid - VMware虚拟机管理器主机名
vmware.hv.datacenter.name[<url>, <uuid>]	VMware		

	虚拟机管理器数据 中心名	字符串	url - VMware服务URL uuid - VMware虚拟机管理器主机名
vmware.hv.datastore.discovery[<url>, <uuid>]	发现 VMware 虚拟机管 理程序数 据存储。	JSON 对象	url - VMware服务URL uuid - VMware虚拟机管理程序主机名
vmware.hv.datastore.read[<url>, <uuid>, <datastore>, <mode>]	从数据存 储区读取 操作的平 均时间 (毫 秒)。	整型 2	url - VMware服务URL uuid - VMware虚拟机管理程序主机名 datastore - 数据储存库名称 mode - 延迟(默认)
vmware.hv.datastore.size[<url>, <uuid>, <datastore>, <mode>]	VMware 数据存储 空间以字 节为单 位, 或占 总数的百 分比。	整数 - 字 节数 浮点 - 百 分比	url - VMware服务URL uuid - VMware虚拟机管理程序主机 名 datastore - 数据存储库名 mode - 可能的值:total (默 认), free, pfree (剩余百分 比), uncommitted
vmware.hv.datastore.write[<url>, <uuid>, <datastore>, <mode>]	对数据存 储区进 行写操 作的平 均时间 (毫 秒)。	整型 2	url - VMware服务URL uuid - VMware虚拟机管理器主机 名 datastore - 数据储存库名 mode - latency (默认)
vmware.hv.discovery[<url>]	发现 VMware 虚拟机管 理程序。	JSON 对象	url - VMware服务URL
vmware.hv.fullname[<url>, <uuid>]	VMware 虚拟机管 理器名。	字符串	url - VMware服务URL uuid - VMware虚拟机管理器主机名
vmware.hv.hw.cpu.freq[<url>, <uuid>]	VMware 虚拟机管 理程序处 理器频率	整数	url - VMware服务URL uuid - VMware虚拟机管理器主机名

		(Hz)。		
vmware.hv.hw.cpu.model[<url>,<uuid>]	VMware虚拟机管理程序处理器模式。	字符串	url - VMware服务URL uuid - VMware虚拟机管理器主机名	
vmware.hv.hw.cpu.num[<url>,<uuid>]	VMware虚拟机管理程序上的处理器内核数。	整数	url - VMware服务URL uuid - VMware虚拟机管理器主机名	
vmware.hv.hw.cpu.threads[<url>,<uuid>]	VMware虚拟机管理程序上的处理器线程数。	整型	url - VMware服务URL uuid - VMware虚拟机管理器主机名	
vmware.hv.hw.memory[<url>,<uuid>]	VMware虚拟机管理程序总内存大小(字节)。	整数	url - VMware服务URL uuid - VMware虚拟机管理器主机名	
vmware.hv.hw.model[<url>,<uuid>]	VMware虚拟机管理器模型。	字符串	url - VMware服务URL uuid - VMware虚拟机管理器主机名	
vmware.hv.hw.uuid[<url>,<uuid>]	VMware虚拟机管理器BIOS UUID.	字符串	url - VMware服务URL uuid - VMware虚拟机管理器主机名	
vmware.hv.hw.vendor[<url>,<uuid>]	VMware虚拟机管理器提供商名。	字符串	url - VMware服务URL uuid - VMware虚拟机管理器主机名	
vmware.hv.memory.size.ballooned[<url>,<uuid>]	VMware虚拟机气球内存大	整数	url - VMware服务URL uuid - VMware虚拟机管理器主机名	

	小(字节)。		VMware虚拟机管理器主机名
vmware.hv.memory.used[<url>,<uuid>]	VMware虚拟机管理程序使用内存大小(字节)。		
vmware.hv.network.in[<url>,<uuid>,<mode>]	VMware虚拟机管理程序网络输入统计(每秒字节数)。	整数 2	url - VMware服务URL uuid - VMware虚拟机管理器主机名 mode - bps(默认)
vmware.hv.network.out[<url>,<uuid>,<mode>]	VMware虚拟机管理程序网络输出统计(每秒字节数)。	整数 2	url - VMware服务URL uuid - 拟机管理器主机名 mode - bps(默认)
vmware.hv.perfcounter[<url>,<uuid>,<path>,<instance>]	VMware虚拟机管理程序性能计数器值。	整数 2	url - VMware服务URL uuid - 拟机管理器主机名 path - 性能仪器路径 instance - 性能计数器实例。对聚合值使用空实例(默认)
vmware.hv.sensor.health.state[<url>,<uuid>]	VMware虚拟机管理程序运行状况汇总传感器。	整数:0 - 灰色;1 - 绿色;2 - 黄色;3 - 红色	url - VMware服务URL uuid - 拟机管理器主机名
vmware.hv.status[<url>,<uuid>]	VMware	整数:0 - 灰色;1	

	理器状态。	色;2 - 黄 色;3 - 红 色	拟机管理器主机名
vmware.hv.uptime[<url>,<uuid>]	VMware虚拟机管理程序正常运行时间(秒)。	整数	url - VMware服务URL uuid - VMware虚拟机管理器主机名
vmware.hv.version[<url>,<uuid>]	VMware虚拟机管理器版本。	字符串	url - VMware服务URL uuid - VMware虚拟机管理器主机名
vmware.hv.vm.num[<url>,<uuid>]	VMware虚拟机管理程序上的虚拟机数量。	整数	url - VMware服务URL uuid - VMware虚拟机管理器主机名
vmware.version[<url>]	VMware服务版本。	字符串	url - VMware服务URL
vmware.vm.cluster.name[<url>,<uuid>]	VMware虚拟机名	字符串	url - VMware服务URL uuid - VMware虚拟机主机名
vmware.vm.cpu.num[<url>,<uuid>]	VMware虚拟机上的处理器数。	整数	url - VMware服务URL uuid - VMware虚拟机主机名
vmware.vm.cpu.ready[<url>,<uuid>]	虚拟机准备好但无法计划在物理CPU上运行的时间百分比。 CPU准备时间取决于主机上的虚拟机数量及其CPU负载	整数 2	url - VMware服务URL uuid - VMware虚拟机主机名

	CPU负载 (%)。		
vmware.vm.cpu.usage[<url>,<uuid>]	VMware虚拟机处理器使用率 (Hz)。	整数	url - VMware服务URL uuid - VMware虚拟机主机名
vmware.vm.datacenter.name[<url>,<uuid>]	VMware虚拟机数据中心名称。	字符串	url - VMware服务URL uuid - VMware虚拟机主机名
vmware.vm.discovery[<url>]	VMware虚拟机自动发现。	JSON对象	url - VMware服务URL
vmware.vm.hv.name[<url>,<uuid>]	VMware虚拟机管理程序名称。	字符串	url - VMware服务URL uuid - VMware虚拟机主机名
vmware.vm.memory.size[<url>,<uuid>]	VMware虚拟机总内存大小 (字节)。	整数	url - VMware服务URL uuid - VMware虚拟机主机名
vmware.vm.memory.size.ballooned[<url>,<uuid>]	VMware虚拟机气球内存大小 (字节)。	整数	url - VMware服务URL uuid - VMware虚拟机主机名
vmware.vm.memory.size.compressed[<url>,<uuid>]	VMware虚拟机压缩内存大小 (字节)。	整数	url - VMware服务URL uuid - VMware虚拟机主机名
vmware.vm.memory.size.private[<url>,<uuid>]	VMware虚拟机专用内存大	整数	url - VMware服务URL uuid - VMware虚拟机主机名

	节)。		
vmware.vm.memory.size.shared[<url>, <uuid>]	VMware 虚拟机共享内存大小(字节)。	整数	url - VMware服务URL uuid - VMware虚拟机主机名
vmware.vm.memory.size.swapped[<url>, <uuid>]	VMware 虚拟机交换内存大小(字节)。	整数	url - VMware服务URL uuid - VMware虚拟机主机名
vmware.vm.memory.size.usage.guest[<url>, <uuid>]	VMware 虚拟机 guest 虚拟机内存使用量(字节)。	整数	url - VMware服务URL uuid - VMware虚拟机主机名
vmware.vm.memory.size.usage.host[<url>, <uuid>]	VMware 虚拟机主机内存使用量(字节)。	整数	url - VMware服务URL uuid - VMware虚拟机主机名
vmware.vm.net.if.discovery[<url>, <uuid>]	发现 VMware 虚拟机网络接口。	JSON 对象	url - VMware服务URL uuid - VMware虚拟机主机名
vmware.vm.net.if.in[<url>, <uuid>, <instance>, <mode>]	VMware 虚拟机网络接口输入统计(每秒字节/数据包)。	整数 2	url - VMware服务URL uuid - VMware虚拟机主机名 instance 网卡实例 mode - bps (默认)/ - 每秒字节/数据包
vmware.vm.net.if.out[<url>, <uuid>, <instance>, <mode>]	VMware 虚拟机网		url - VMware服务URL uuid -

	虚拟机网卡输出统计(每秒字节/数据包)。	整数 2	url - VMware服务URL uuid - VMware虚拟机主机名 instance 网卡实例 mode - bps (默认)/ - 每秒字节/数据包
vmware.vm.perfcounter[<url>,<uuid>,<path>,<instance>]	VMware虚拟机性能计数器值。	整数 2	url - VMware服务URL uuid - VMware虚拟机主机名 path - 性能计数器路径 1 instance - 性能计数器实例。对聚合值使用空实例(默认)
vmware.vm.powerstate[<url>,<uuid>]	VMware虚拟机电源状态。	整数:0 - 关闭电源;1 - 打开电源;2 - 暂停	url - VMware服务URL uuid - VMware虚拟机主机名
vmware.vm.storage.committed[<url>,<uuid>]	VMware虚拟机提交存储空间(字节)。	整数	url - VMware服务URL uuid - VMware虚拟机主机名
vmware.vm.storage.uncommitted[<url>,<uuid>]	VMware虚拟机未提交的存储空间(字节)。	整数	url - VMware服务URL uuid - VMware虚拟机主机名
vmware.vm.storage.unshared[<url>,<uuid>]	VMware虚拟机非共享存储空间(字节)。	整数	url - VMware服务URL uuid - VMware虚拟机主机名
vmware.vm.uptime[<url>,<uuid>]	VMware虚拟机正常运行时间(秒)。	整数	url - VMware服务URL uuid - VMware虚拟机主机名

<uuid>]	发现 VMware 虚拟机磁 盘设备。	JSON 对象	url - VMware服务URL uuid - VMware虚拟机主机名
vmware.vm.vfs.dev.read[<url>,<uuid>, <instance>,<mode>]	VMware 虚拟机磁 盘设备读 取统计信 息(每秒的 字节数/操 作数)。	整数 2	url - VMware服务URL uuid - VMware虚拟机主机名 instance 磁盘设备实例。 mode - bps (默 认)/ops - 字节/操作每秒
vmware.vm.vfs.dev.write[<url>,<uuid>, <instance>,<mode>]	VMware 虚拟机磁 盘设备写 入统计信 息(每秒字 节数/操 作数)。	整数 2	url - VMware服务URL uuid - VMware虚拟机主机名 instance 磁盘设备实例 mode - bps (默 认)/ops - 字节/操作每秒
vmware.vm.vfs.fs.discovery[<url>,<uuid>]	发现 VMware 虚拟机文 件系统。	JSON 对象	url - VMware服务URL uuid - VMware虚拟机主机名
vmware.vm.vfs.fs.size[<url>,<uuid>, <fsname>,<mode>]	VMware 虚拟机文 件系统统 计(字 节/百分 比)。	整数	url - VMware服务URL uuid - VMware虚拟机主机名 fsname - 文件系统名 mode - total/free/used/pfree/pu

脚注

1 VMware性能计数器路径具有 `group/counter[rollup]` 格式，其中

- `group` - 性能计数器组，例如 `cpu`
- `counter` - 性能计数器名称，例如 `usagemhz`
- `rollup` - 性能计数器卷起类型，例如 `average`

所以上面的例子会给出如下的计数器路径：`cpu/usagemhz[average]`

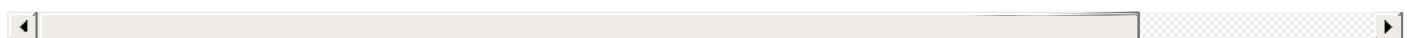
性能计数器组描述、计数器名称和汇总类型可以在[VMware文档](#)中找到。

2 这些监控项的值是从VMware性能计数器获得的，VMwarePerfFrequency参数用于刷新Zabbix VMware缓存中的数据：

- vmware.hv.datastore.read
- vmware.hv.datastore.write
- vmware.hv.network.in
- vmware.hv.network.out
- vmware.hv.perfcounter
- vmware.vm.cpu.ready
- vmware.vm.net.if.in
- vmware.vm.net.if.out
- vmware.vm.perfcounter
- vmware.vm.vfs.dev.read
- vmware.vm.vfs.dev.write

更多信息

有关如何配置Zabbix以监控VMware环境的详细信息，请参阅[虚拟机监控](#)。



6 日志文件监控

概述

Zabbix可用于集中监控和分析具有/不具有日志转动能力的日志文件。

当日志文件包含某些字符串或字符串模式时，通知信息可用于警告用户。

要监控日志文件，必须具有：

- Zabbix代理在主机上运行
- 日志监控项设置

受监控的日志文件的大小限制取决于[大文件支持](#)。

配置

验证代理参数

确保在[代理配置文件中](#)：

- 'Hostname'参数与前端的主机名匹配
- "ServerActive"参数中的服务器被指定用于处理活动检查

监控项配置

配置一个日志 [监控项](#)：



专为日志监控项的输入：

Type	这里选择 Zabbix agent (active) 。
Key	设置： <code>log[/path/to/file/filename,<regexp>,<encoding>,<maxlines>,<mode>,<output>,<maxdelay>]</code> 或 者 <code>logrt[/path/to/file/regexp_describing_filename_pattern,<regexp>,<encoding>,<maxlines>,<mode>,<output>,<maxdelay>]</code> Zabbix代理将通过内容正则表达式（如果存在）过滤日志文件的条目。如果只需要匹配行的数量： <code>log.count[/path/to/file/file_name,<regexp>,<encoding>,<maxproclines>,<mode>,<maxdelay>]</code> 或 者 <code>logrt.count[/path/to/file/regexp_describing_filename_pattern,<regexp>,<encoding>,<maxproclines>,<mode>,<maxdelay>]</code> 。确保文件具有“zabbix”用户的读取权限，否则监控项状态将被设置为“不支持”。更多细节请查看 log 、 log.count 、 logrt 和 logrt.count 条目在支持的 Zabbix代理监控项 Key 章节。
_Type of information	在这里，针对log和logrt选择Log，针对log.count和logrt.count选择Numeric (unsigned)。如果可选地使用 <code>output</code> 参数，则可以选择除“日志”之外的适当类型的信息。请注意，选择非日志类型的信息将导致丢失本地时间戳。
Update interval (in sec)	该参数定义了Zabbix代理将检查日志文件中的任何更改的频率。将其设置为1秒将确保你能尽快获得新记录。

Log time format

在此字段中，你可以选择指定用于解析日志行时间戳的模式。如果留空，则不会解析时间戳。支持的占位符：**y**: 年 (0001-9999) **M**: 月 (01-12) **d**: 日 (01-31) **h**: 小时 (00-23) **m**: 分钟 (00-59) **s**: 秒 (00-59) 例如，从Zabbix代理日志文件中查看以下行：“23480:20100328:154718.045 Zabbix agent started. Zabbix 1.8.2 (revision 11211).”它以PID的六个字符位置开始，后跟日期、时间和行的其余部分。此行的日志时间格式为“pppppp:yyyyMMdd:hhmmss”. 请注意，“p”和“：“字符只是占位符，只能是“yMdhms”。

重要信息

- 服务器和代理在两个计数器中保留监控日志大小和上次修改时间（对于logrt）的跟踪。此外：
 - 代理还在内部使用inode编号（在UNIX/GNU/Linux上）、文件索引（在Microsoft Windows上）和前512个日志文件字节的MD5的求和，以便在日志文件被截断和旋转时改进决策。
 - 在UNIX/GNU/Linux系统上，假定存储日志文件的文件系统会报告索引节点号，它可用于跟踪文件。
 - 在Microsoft Windows上Zabbix代理确定日志文件所在的文件系统类型，并使用：
 - 在NTFS文件系统上64位文件索引。
 - 在ReFS文件系统（仅从Microsoft Windows Server 2012开始支持）128位文件ID。
 - 在文件索引改变的文件系统（例如FAT32, exFAT）上，当日志文件旋转导致具有相同最近修改时间的多个日志文件时，使用fall-back（回退）算法是在不确定的条件下采取的明智方法。
 - inode号，文件索引和MD5总和由Zabbix代理在内部收集。它们不传输到Zabbix服务器，并且在Zabbix代理停止时丢失。
 - 不要使用“touch”实用程序修改日志文件的最后修改时间，不要在以后恢复原始名称的情况下复制日志文件（这将更改文件inode号）。在这两种情况下，文件将被视为不同的，将从头开始进行分析，这可能会导致重复的告警。
 - 如果logrt[]监控项有几个匹配的日志文件，并且Zabbix代理程序跟随其中最新的日志文件，同时最新的日志文件被删除，则在“<目录>”中会出现一条警告消息“没有文件匹配””。Zabbix代理将忽略修改时间小于最近日期的日志文件。
- 代理人从上一次停止的时候开始读取日志文件。
- 在代理刚刚启动或已收到以前被禁用或不支持的监控项的情况下，已经分析的字节数（大小计数器）和最后修改时间（时间计数器）存储在Zabbix数据库中并发送到代理，以确保代理从此开始读取日志文件。
- 每当日志文件变得小于代理已知的日志大小计数器时，计数器将重置为零，并且代理从开始位置读取日志文件。
- 如果目录中存在多个匹配文件，且最后修改时间相同，则代理会尝试以相同的修改时间对所有日志文件进行正确分析，并避免跳过数据或分析相同的数据两次（尽管有时不能保证）。代理不承担任何特定的日志文件轮换方案。当提供具有相同修改时间的多个日志文件时，代理将以字典顺序降序处理它们。因此，对于某些旋转方案，日志文件将按原始顺序进行分析。对于其它旋转方案，原始日志文件顺序将不会被执行，这可能导致以更改顺序报告匹配的日志文件记录（如果日志文件的上次修改时间不同，则不会发生问题）。
- Zabbix代理处理日志文件的新记录每更新间隔秒。
- Zabbix代理不会每秒发送超过日志文件的最大值。该限制可防止网络和CPU资源的重载，并覆盖[代理配置文件](#)

中由**MaxLinesPerSecond**参数提供的默认值。

- 要找到所需的字符串，Zabbix将处理比**MaxLinesPerSecond**中设置的新行多4倍。因此，如果**log[]**或**logrt[]**监控项的更新间隔为1秒，则默认情况下，代理将分析不超过80个日志文件记录，并在一次检查中向Zabbix服务器发送不超过20个匹配记录。通过在代理配置文件中增加**MaxLinesPerSecond**或在监控项Key中设置**maxlines**参数，可以在一次检查中将限制最多增加4000个分析的日志文件记录和1000个匹配记录发送到Zabbix服务器。如果更新间隔设置为2秒，则一次检查的限制将被设置为更新间隔1秒的2倍。
- 此外，日志和日志计数值始终限于代理发送缓冲区大小的50%，即使其中没有非日志值。因此，为了在一个连接（而不是几个连接）中发送最大值，代理**BufferSize**参数必须至少为**maxlines** × 2。
- 在没有日志监控项的情况下，所有代理缓冲区大小都用于非日志值。当日志值进入时，根据需要替换较旧的非日志值，最多指定为50%。
- 对大于256kB的日志文件记录，只有第一个256kB与正则表达式匹配，其余的记录将被忽略。但是，如果Zabbix代理在处理长记录时停止，代理内部状态将丢失，并且可以在代理重新启动后再次分析不同的长记录。
- “\”路径分隔符的特殊注意事项：如果**file_format**是“`file.log`”，则不应该有“`file`”目录，因为不可能明确地定义是否转义了“.”，以及是否为第一个文件名符号。
- 仅在文件名中支持**logrt**的正则表达式，不支持目录正则表达式匹配。
- 在UNIX平台上，如果要找的日志文件的目录不存在，则**logrt[]**监控项将变为NOTSUPPORTED。
- 在Microsoft Windows上，如果目录不存在，则监控项将不会变为NOTSUPPORTED（例如，如果目录在监控项Key中拼写错误）。
- 没有用于**logrt[]**监控项的日志文件不会使其NOTSUPPORTED。读取**logrt[]**监控项的日志文件的错误将作为告警记录到Zabbix代理日志文件中，但不要使监控项NOTSUPPORTED。
- Zabbix代理日志文件可以帮助你找出为什么**log[]**或**logrt[]**监控项成为NOTSUPPORTED。Zabbix可以监视其代理日志文件，除了在**DebugLevel=4**时。

提取正则表达式的匹配部分

有时我们可能只想从目标文件中提取感兴趣的值，而不是在找到正则表达式匹配时返回整行。

自Zabbix 2.2.0以后，日志监控项有能力从匹配的行提取所需的值。这通过**log**和**logrt**监控项中的附加输出参数来实现。

使用‘output’参数允许指示我们可能感兴趣的匹配的子组。

例如

```
1. log[/path/to/the/file,"large result buffer allocation.*Entries: ([0-9]+),,\,\1]
```

应该可以返回在以下内容中找到的条目数：

```
1. Fr Feb 07 2014 11:07:36.6690 */ Thread Id 1400 (GLEWF) large result
2. buffer allocation - /Length: 437136/Entries: 5948/Client Ver: >=10/RPC
```

```
3. ID: 41726453/User: AUser/Form: CFG:ServiceLevelAgreement
```

The reason why Zabbix will return only the number is because 'output' here is defined by \1 referring to the first and only subgroup of interest: ([0-9]+)Zabbix只返回数字的原因是因为这里的'output'是由\1定义的，指的是第一个也是唯一的想要的子组: ([0-9]+)

而且，通过提取和返回数字的能力，该值可用于定义触发器。

使用maxdelay参数

日志监控项中的“maxdelay”参数允许忽略日志文件中的一些较旧的行，以便在“maxdelay”秒内获取最近分析的行。

指定'maxdelay'>0可能导致忽略重要的日志文件记录和错过的报警。只有在必要时才使用。

默认情况下，日志监控项将跟踪出现在日志文件中的所有新行。但是，有些应用程序在某些情况下开始在其日志文件中写入大量的消息。例如，如果数据库或DNS服务器不可用，则此类应用程序会将数千个几乎相同错误消息的日志文件像洪水般爆发出来，直到恢复正常操作。默认情况下，所有这些消息将被完全分析，并将匹配的行发送到配置为“log”和“logrt”监控项的服务器。

内置防过载保护包括一个可配置的“maxlines”参数（保护服务器免受太多进入匹配的日志行）和 $4 * \text{maxlines}$ 限制（保护主机CPU和I/O免受代理在一次检查中的过载）。然而，内置保护有两个问题。首先，向服务器报告大量潜在的不太有用的消息，并消耗数据库中的空间。第二，由于每秒分析的线路数量有限，代理可能落后于最新的日志记录数小时。你可能希望尽快了解日志文件中的当前情况，而不是检查数小时的历史记录。

这两个问题的解决方案是使用'maxdelay'参数。如果指定'maxdelay'>0，则在每次检查期间，测量处理的字节数、剩余字节数和处理时间。从这些数字中，代理计算估计延迟 - 分析日志文件中所有剩余记录所需的秒数。

如果延迟不超过“maxdelay”，则代理程序会照常分析日志文件。

如果延迟大于“maxdelay”，代理将通过“跳过”到新的估计位置来忽略日志文件块，以便可以在“maxdelay”秒内分析剩余的行。

请注意，代理甚至不会将忽略的行读入缓冲区，而是计算要在文件中跳转的大致位置。

跳过日志文件行的事实记录在代理日志文件中，如下所示：

```
1. 14287:20160602:174344.206 item:"logrt[/home/zabbix32/test[0-9].log",ERROR,,1000,,,120.0]"
2. logfile:"/home/zabbix32/test1.log" skipping 679858 bytes
3. (from byte 75653115 to byte 76332973) to meet maxdelay
```

“to byte”数字是近似值，因为在“跳转”之后，代理将文件中的位置调整到可能在文件中更早的日志行的开头。

根据增长速度与分析日志文件的速度的不同，你可能会看到没有“jumps”、少有或经常“jumps”、大或小的“jumps”，甚至每次检查中的“jumps”都很小。系统负载和网络延迟的波动也会影响延迟的计算，因此“jumps”可以跟上“maxdelay”参数。

不建议设置'maxdelay' < 'update interval'（这可能会导致频繁的“jumps”）。

代理和服务器之间的通信失败时的操作

来自`log[]`和`logrt[]`监控项的每个匹配行以及每个`log.count[]`和`logrt.count[]`监控项检查的结果都需要代理发送缓冲区中指定的50%区域中的空闲时隙。缓冲区元素定期发送到服务器（或代理服务器），缓冲区可以再次释放。

虽然代理发送缓冲区中的指定日志区域中有空闲时隙，并且代理和服务器（或代理服务器）之间的通信失败，但是日志监控结果在发送缓冲区中累积。这有助于缓解短暂的通信故障。

在更长的通信故障期间，所有日志插槽都被占用，并采取以下措施：

- `log[]`和`logrt[]`监控项检查已停止。当通信恢复并且缓冲器中的空闲插槽可用时，从先前的位置恢复检查。若没有匹配的行丢失，稍后再报告。
- 如果`maxdelay=0`（默认），则`log.count[]`和`logrt.count[]`监控被停止。这种行为类似于上述的`log[]`和`logrt[]`监控项。请注意，这可能会影响`log.count[]`和`logrt.count[]`结果：例如，一次检查计算出日志文件中有100个匹配行，但是由于缓冲区中没有空闲插槽，因此停止检查。当通信恢复时，代理计数相同的100条匹配行，还有70条新的匹配行。代理会发送`count=170`，就好像它们在一次检查中发现的一样。
- `log.count[]`和`logrt.count[]`检查与`maxdelay>0`：如果在检查期间没有“跳转”，则行为类似于上述。如果在日志文件行上发生“跳转”，则保留“跳转”之后的位置，同时计算结果被丢弃。因此，即使在通信失败的情况下，代理也试图跟上越来越多的日志文件。

7 计算监控项

概述

你可以基于其它监控项创建计算监控项。

因此，计算监控项是创建虚拟数据源的一种方式。这些值将根据算术表达式定期计算。所有计算都由Zabbix服务器完成，与Zabbix代理或代理服务器执行的计算无关。

生成的数据将存储在Zabbix数据库中，与其它任何监控项一样 - 这意味着存储历史和趋势值，以便快速生成图表。计算的监控项可用于触发器表达式，由宏或其它实体引用，与任何其它监控项类型相同。

要使用计算监控项，请选择监控项类型 **Calculated**。

可配置字段

key 是唯一的监控项标识符（每个主机）。你可以使用支持的符号创建任何Key名称。

计算定义应在**Formula** 字段中输入。公式和密钥之间几乎没有连接。Key参数不能以任何方式用于公式中。

一个简单公式的正确语法是：

```
1. func(<key>|<hostname:key>,<parameter1>,<parameter2>,...)
```

当：

参数	定义
func	触发器表达式支持的 functions : last、min、max、avg、count等
key	其数据要使用的另一个监控项的密钥。它可以被定义为key或hostname: key。注意：将整个Key放在双引号（“...”）中，强烈建议避免由于Key内的空格或逗号而导致错误的解析。如果Key中也有引用的参数，则必须使用反斜杠（\）来转义这些双引号。 参考下面的 示例5
parameter(s)	功能参数（如果需要）。

从计算监控项公式引用的所有监控项都必须存在并且正在收集数据（[功能和不支持的监控项](#)除外）。另外，如果更改引用监控项的Key，则必须使用该Key手动更新任一公式。

如果用于引用函数参数或常量，公式中的[用户宏](#)将被扩展。如果引用函数、主机名、监控项Key、监控项key参数或运算符，则用户宏将不会被扩展。

更复杂的公式可以使用函数，运算符和括号的组合。你可以使用触发器表达式支持的所有功能和[operators](#)。请注意，语法略有不同，但逻辑和运算符优先级完全相同。

与触发器表达式不同，Zabbix根据监控项更新间隔处理计算监控项，而不是在接收到新值时处理。

如果计算结果为浮点值，且如果计算的信息类型为Numeric（无符号），则该值将被修剪为整数。

在几种情况下，计算监控项可能不受支持：

- 参考监控项

- 没有找到
- 被禁止了
- 属于一个被禁止的主机
- 不支持(查阅例外情况 [功能和不受支持的监控项，具有不支持的监控项和未知值的表达式 和 操作](#))
- 没有数据来计算一个函数
- 被零除
- 使用不正确的语法

在Zabbix 1.8.1中引入了对计算监控项的支持。从Zabbix3.2开始，计算监控项在某些情况下可能涉及不符合要求的项目，如这些所述 [功能和不受支持的监控项，具有不支持的监控项和未知值的表达式 和 操作](#)。

用法示例

示例 1

计算'/'上可用磁盘空间的百分比。

使用**last**功能：

```
1. 100*last("vfs.fs.size[/,free]"/last("vfs.fs.size[/,total]"))
```

Zabbix将获取最新的空闲和总磁盘空间值，并根据给定的公式计算百分比。

示例 2

计算Zabbix处理的数值的10分钟平均值。

使用**avg**功能：

```
1. avg("Zabbix Server:zabbix[wcache,values]",600)
```

请注意，大量使用长时间计算的监控项可能会影响Zabbix服务器的性能。

示例 3

计算eth0上的总带宽。

两个功能总和：

```
1. last("net.if.in[eth0,bytes]") + last("net.if.out[eth0,bytes]")
```

示例 4

计算入站流量的百分比。

更复杂的表达式：

```
1. 100*last("net.if.in[eth0,bytes]"/(last("net.if.in[eth0,bytes]"+last("net.if.out[eth0,bytes]")))
```

示例 5

在计算监控项中正确使用聚合。

记下双引号在引号键内如何转义：

```
1. last("grpsum[\"video\", \"net.if.out[eth0,bytes]\", \"last\"]") /  
last("grpsum[\"video\", \"nginx_stat.sh[active]\", \"last\"]")
```

8 内部检查

概述

内部检查可以监控Zabbix的内部进程。换句话说，你可以监控Zabbix服务器或Zabbix代理服务器的运行情况。

内部检查是：

- 在Zabbix服务器 - 主机是否被服务器监控
- 在Zabbix代理服务器 - 主机是否被代理服务器监控

内部检查由服务器或代理服务器执行，无论主机维护状态如何（从Zabbix 2.4.0起）。

要使用此监控项，请选择**Zabbix internal**监控项类型。

内部检查由Zabbix轮询器处理。

被支持的检查

- 没有尖括号的参数是常量，例如 `zabbix[host,<type>,available]` 中的'host'和'available'。在监控项Key中使用它们。
- 仅当主机被服务器监控时，才能收集“代理服务器不支持”的监控项和监控项参数的值。反之亦然，“服务器不支持”的值仅当主机被代理服务器监控时采集。

Key	▲	描述	返回值	
<code>zabbix[boottime]</code>		Zabbix服务器或Zabbix代理服务器进程的启动时间（按秒）。	整数	
<code>zabbix[history]</code>		存储在HISTORY表中的数量值。	整数	如果使用MySQL InnoDB、Or acle！（代理服务器不支持）
<code>zabbix[historylog]</code>		存储在 HISTORY_LOG 表中的数量值。	整数	如果使用MySQL InnoDB、Or acle！\从Zabbix 1.8.3开始支 持）
<code>zabbix[history_str]</code>		存储在 HISTORY_STR 表中的数量值。	整数	如果使用MySQL InnoDB、Or acle！（代理服务器不支持）

zabbix[history_text]	存储在 HISTORY_TEXT 表中的数量值。	整数	如果使用MySQL InnoDB、Or acle！从Zabbix 1.8.3开始支 持）
zabbix[history_uint]	存储在 HISTORY_UINT 表中的值数。	整数	如果使用MySQL InnoDB、Or acle！从Zabbix 1.8.3开始支 持）
zabbix[host,,items]	主机上启用的监 控项的数量（受 支持和不受支 持）。	整数	从Zabbix 3.0.0.开始支持I
zabbix[host,,items_unsupported]	主机上启用的不 受支持的监控项 数量。	整数	从Zabbix 3.0.0.开始支持I
zabbix[host,,maintenance]	当前主机的维护 状态。	0 - 主机处 于正常状态, 1 - 主机处于维 护状态但采集 数据, 2 - 主 机处于维护状 态不采集数 据.	此监控项始终由Zabbix服 务器或代理服务器上）。代理 项。第二个参数必须为空，并 从Zabbix 2.4.0.开始支持
zabbix[host,<type>,available]	主机上特殊类型 的检查。该监控 项的值对应于主 机列表中的可用 性图标。	0 - 不可用, 1 - 可用, 2 - 未知.	有效的类型是：agent, snm 有关主机不可达/不可用的配置 开始支持
zabbix[hosts]	监控主机数量	整数	从Zabbix 2.2.0开始支持
zabbix[items]	已启用监控项的 数量（受支持和 不受支持的）。	整数	
zabbix[items_unsupported]	不支持的监控项 数量。	整数	
zabbix[java,,<param>]		如果<param> 为ping，则	

	有关Zabbix Java网关的信息。	返回“1”。可以使用 nodata()触发功能来检查Java网关的可用性。如果<param>是版本，则返回Java网关的版本。示例：“2.0.0”	<param>的有效值是：_ping空，并保留供将来使用。从Z
zabbix[process,<type>,<mode>,<state>]	时间是一个特定的Zabbix进程或一组进程（由<type>和<mode>标识）以百分比形式在<state>中使用。仅在最后一分钟计算。如果<mode>是没有运行的Zabbix进程号（例如，运行<mode>的5个轮询器被指定为6），则此监控项将变为不受支持的状态。最小和最大值是指单个进程的使用百分比。因此，如果在一组3个轮询器中，每个进程的使用百分比为2, 18和66，则min将返回2，max将返回66。进程报告它们在共享内存中正在做什么，而自我监视进程每秒总结一次数据。状态改变（忙/空闲）在更改时被注册 - 因此一个进程变成繁忙的寄存器，并且在它变得空闲之前不改变或更新状态。这确保即使完全挂起的进程也被正确地注册为100%忙。目前，“busy”表示“not sleeping”，但在将来可能会引入额外的状态 - 等待锁，执行数据库查询等。	时间百分比。 浮点	目前支持以下进程类型： ale 服务器不支持) configurat 据的内存中缓存的进程 data 者 (不支持Zabbix服务器)di 则发出警告消息 (代理服务器发现进程 escalator - act 持) heartbeat sender - Zabbix服务器) history sy 者 housekeeper - 删除旧历 web轮询检查器 icmp pinger manager - IPMI轮询管理i 器 java poller - Java检通用轮询器 proxy poller - 理服务器不支持) self-mon 信息的进程 snmp trapper - manager - 手动关闭问题的持) timer - 与时间相关的管理服务器不支持) trapper - trapperunreachable pol 器 vmware collector - 负 VMware数据收集器注意：你这些进程类型。有效模式是： av 值 (默认) count - 返回给 <state>**max - 最大值mi number> - 过程号 (在1和果4个 trappers 正在运行，则是： busy - 进程处于忙状态， - 进程处于空闲状态，什么都 zabbix[process,poller,内，轮询进程的平均花费时间 pinger",max,busy] → 在 pinger进程花费最多时间 → syncer",2,busy] → 在最些操作花费的时间 → zabbix → 当前运行的 trapper 进程的持

	在Linux和大多数其它系统上，解析度是1/100秒。		
zabbix[proxy,<name>,<param>]	有关Zabbix代理服务器的信息。	整数	<name> - 代理服务器名支持(<param>):lastaccess - 息的时间戳示例：→ zabbix[proxy,"Germany"] 发器功能 可用于检查代理的可该监控项始终由Zabbix服务器或代理服务器上)。
zabbix[proxy_history]	代理服务器历史表中等待发送到服务器的值的数量。	整数	从Zabbix 2.2.0开始支持(→)
zabbix[queue,<from>,<to>]	队列中被监视的监控项数量至少延迟了从<from>秒，但小于<to>秒。	整数	<from> - 默认：6秒<to> symbols (s,m,h,d,w) 被和 to 从Zabbix 1.8.3.
zabbix[rcache,<cache>,<mode>]	Zabbix配置缓存的可用性统计信息。	整数(大小)；浮动(百分比)。	缓存：bufferMode:total 用缓冲区大小 pfree - 可用存区大小
zabbix[requiredperformance]	Zabbix服务器或Zabbix代理服务器所需性能，以每秒新增的值计算。	浮点	与Reports → Zabbix状态中值”大致相关。从Zabbix 1.8.3开始支持。
zabbix[trends]	存储在TRENDS表中的数量值。	整数	如果使用MySQL InnoDB、Oracle！(代理服务器不支持)
zabbix[trends_uint]	存储在TRENDS_UINT表中的数量值。	整数	如果使用MySQL InnoDB、Oracle！\从Zabbix 1.8.3 开始支持)
zabbix[triggers]	Zabbix数据库中启用的触发器数量，启用主机上将启用所有监控项。	整数	(代理服务器不支持)

zabbix[uptime]	Zabbix服务器或代理服务器正常运行时间(按秒计)。	整数	
zabbix[vcache,buffer,<mode>]	Zabbix值缓存的可用性统计信息。	整数(大小);浮点(百分比)。	模式:total - 缓冲区的总大 pfree - 可用缓冲区百分比 - 已用的缓冲区百分比从Zab 务器不支持)
zabbix[vcache,cache,<parameter>]	Zabbix值缓存的有效性统计。	整数使用模式 参数:0 - 正常模式,1 - 低内存模式	参数:requests - 总请求数 中取出的历史值)misses - 取的历史值)mode - 值缓存 2.2.0开始支持, 模式参数从 理服务器不支持)你可以使用 值, 以获得每秒的统计数据。
zabbix[vmware,buffer,<mode>]	Zabbix vmware缓存的 可用性统计信 息。	整数(大 小); 浮动 (百分比)。	模式:total - 缓冲区的总大 pfree - 可用缓冲区百分比 - 已用的缓冲区百分比从Zab
zabbix[wcache,<cache>,<mode>]	Zabbix写缓存的统计和可用性。	缓存	模式 指定<cache>是必需的。
values	all(默认)	由Zabbix服 务器或 Zabbix代理 服务器处理的 值的总数(不 支持的监控项 除外)。	整数
float	处理的浮点值的 数量。	整数	计数器。
uint	处理的无符号整 数值的数量。	整数	计数器。
str	处理的字符/字 符串值的数量。	整数	计数器。
log	处理日志值的数 量。	整数	计数器。
text	已处理文本值的 数量。	整数	计数器。
not supported	监控项处理导致 项目不受支持或 保持该状态的次	整数	计数器。Not supported 模

	数。		
history	pfree(默认)	免费历史缓冲区的百分比。	浮点
free	可用历史缓冲区大小	整数	
total	历史缓冲区总大小	整数	
used	已用的历史缓冲区大小	整数	
index	pfree(默认)	可用的历史索引缓冲区的百分比。	浮点
free	可用历史索引缓冲区的大小。	整数	
total	历史记录索引缓冲区的总大小。	整数	
used	已用的历史索引缓冲区的大小。	整数	
trend	pfree(默认)	可用趋势缓存的百分比。	浮点
free	可用趋势缓存大小	整数	(代理服务器不支持)
total	趋势缓存总大小	整数	(代理服务器不支持)
used	已用的趋势缓存大小	整数	(代理服务器不支持)



9 SSH检查

概述

运行SSH检查是作为无代理监控的。SSH检查不需要Zabbix代理。

执行SSH检查Zabbix服务器必须[初始化配置](#)为SSH2支持。

最低支持的libssh2库版本为1.0.0

配置

密码验证

SSH检查提供两种身份验证方法，一种是用户/密码对，另一种是基于密钥文件。

如果你不打算使用密钥，则除了将libssh2连接到Zabbix，你不需要额外的配置（如果从源代码构建）。

密钥文件认证

要对SSH监控项使用基于密钥的身份验证，需要对服务器配置进行某些更改。

以root身份打开Zabbix服务器配置文件(`zabbix_server.conf`)，并查找以下行：

```
1. # SSHKeyLocation=
```

取消注释，并设置完整路径到公钥和私钥所在的文件夹：

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

保存文件，然后重新启动`zabbix_server`。

`/home/zabbix`在这里是zabbix用户帐户的主目录，而`.ssh`是一个目录，默认情况下，公钥和私钥将由主目录中的`ssh-keygen`命令生成。

通常来自不同操作系统发行版的`zabbix-server`的安装包在不太明显的地方（与系统帐户一样）创建了一个带有主目录的zabbix用户帐户。例如，对于CentOS，它是`/var/lib/zabbix`，对于Debian，它是`/var/run/zabbix`。

在开始生成密钥之前，可以考虑将主目录重新分配到更熟悉的地方（更加直观）。与上面提到的`SSHKeyLocation` Zabbix server配置参数相对应。

如果根据[安装章节](#)手动添加了zabbix帐户，则可以跳过这些步骤，因为在这种情况下，最可能的主目录已经位于`/home/zabbix`。

要更改zabbix用户帐户的设置，必须停止所有正在使用它的进程：

```
1. # service zabbix-agent stop
2. # service zabbix-server stop
```

要更改主目录位置以尝试移动它（如果存在），应执行一个命令：

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

主目录绝对可能不存在于旧的地方（例如在CentOS中），所以应该在新的地方创建。一个可靠的尝试是：

```
1. # test -d /home/zabbix || mkdir /home/zabbix
```

为了确保所有操作都是安全的，可以执行其它命令来设置主目录的权限：

```
1. # chown zabbix:zabbix /home/zabbix
2. # chmod 700 /home/zabbix
```

以前被停止的进程现在可以重新启动：

```
1. # service zabbix-agent start
2. # service zabbix-server start
```

现在可以通过命令执行生成公钥和私钥的步骤：

```
1. # sudo -u zabbix ssh-keygen -t rsa
2. Generating public/private rsa key pair.
3. Enter file in which to save the key (/home/zabbix/.ssh/id_rsa):
4. Created directory '/home/zabbix/.ssh'.
5. Enter passphrase (empty for no passphrase):
6. Enter same passphrase again:
7. Your identification has been saved in /home/zabbix/.ssh/id_rsa.
8. Your public key has been saved in /home/zabbix/.ssh/id_rsa.pub.
9. The key fingerprint is:
10. 90:af:e4:c7:e3:f0:2e:5a:8d:ab:48:a2:0c:92:30:b9 [email protected]
11. The key's randomart image is:
12. +--[ RSA 2048]----+
13. |                               |
14. |                               |
15. |       o                   |
16. | .     o                   |
17. |+    . S                  |
18. | .+   o =                 |
19. |E .   * =                |
20. |=o . . * .               |
21. |... oo.o+                |
22. +-----+
```

Note：默认情况下，在`/home/zabbix/.ssh`目录中生成公钥和私钥（分别为`id_rsa.pub`和`id_rsa`），该目录对应于Zabbix服务器的`SSHKeyLocation`配置参数。

`ssh-keygen`工具和SSH服务器可以支持“rsa”以外的其它Key，但Zabbix使用的libssh2可能不支持。

Shell配置表

对于将通过SSH检查监视的每个主机，此步骤只应执行一次。

通过使用以下命令，在远程主机10.10.10.10上安装公钥文件，以便可以使用root帐户执行SSH检查：

```

1. # sudo -u zabbix ssh-copy-id [email protected]
2. The authenticity of host '10.10.10.10 (10.10.10.10)' can't be established.
3. RSA key fingerprint is 38:ba:f2:a4:b5:d9:8f:52:00:09:f7:1f:75:cc:0b:46.
4. Are you sure you want to continue connecting (yes/no)? yes
5. Warning: Permanently added '10.10.10.10' (RSA) to the list of known hosts.
6. [email protected]'s password:
7. Now try logging into the machine, with "ssh '[email protected]'", and check in:
8. .ssh/authorized_keys
9. to make sure we haven't added extra keys that you weren't expecting.

```

现在可以使用zabbix用户的默认私钥（/home/zabbix/.ssh/id_rsa）检查SSH登录了：

```
1. # sudo -u zabbix ssh [email protected]
```

如果登录成功，则shell中的配置部分完成，并且可以关闭远程SSH会话。

监控项配置

要执行的实际命令必须放在监控项配置中的执行脚本域中。可以通过将多个命令放置在新行上来执行多个命令。在这种情况下，返回的值也将被格式化为多列。



需要SSH监控项特定信息的字段有：

参数	描述	说明
Type	在这里选择SSH agent。	
Key	格式为ssh.run的唯一（每个主机）监控项Keyssh.run[<unique short description>,<ip>,<port>,<encoding>]	<unique short description> 是必需的，对于每个主机的所有SSH监控项都应该是唯一的。默认端口为22，而不是分配给该项目的接口中指定的端口
Authentication method	“密码”或者“公钥Key”的其中一个	
User name	用户名在远程主机上进行身份验证。必需的	
Public key file	如果身份验证方法为“公钥”，则为公钥的文件名。必需的。	示例：id_rsa.pub - 由命令ssh-keygen生成的默认公钥文件名。
Private key file	如果身份验证方法为“私钥”，则为私钥的文件名。必需的。	示例：id_rsa - 默认私钥文件名
Password orKey passphrase	验证密码，或者密码如果用于私钥密码	如果没有使用密码短语，则将密钥密码短字段留空。另请参阅有关密码短语使用的已知问题
Executed script	使用SSH远程会话执行shell命令	示例：date +%s &> service mysql-server status & ps auxww grep httpd wc -l

libssh2库可能会将可执行脚本截断到~32kB。

10 Telnet检查

概述

执行Telnet检查作为无代理监视。 Telnet监控不需要Zabbix代理。

可配置字段

要执行的实际命令必须放在监控项配置中的执行脚本字段中。可以通过将多个命令放置在新行上来执行多个命令。在这种情况下，返回值也将形成为多列。

支持的shell提示符可以是：

- \$
- #
- >
- %

以这些字符之一结尾的telnet提示行将从返回值中删除，但仅在命令列表中的第一个命令中，即仅在telnet会话的开始处。

Key	描述	注释
<code>telnet.run[<unique short description>,<ip>,<port>,<encoding>]</code>	使用telnet连接在远程设备上运行命令	

如果telnet检查返回非ASCII字符的值，并以非UTF8编码，则应正确指定key的<encoding>参数。有关详细信息，请参阅[返回值的编码](#)。

11 外部检查

概述

外部检查是由Zabbix服务器通过[运行shell脚本](#)或二进制执行的检查。

外部检查不需要在被监控的主机上运行任何代理。

监控项Key的语法是：

```
1. script[<parameter1>,<parameter2>,...]
```

当：

参数	定义
script	shell脚本或二进制文件的名称。
parameter(s)	可选的命令行参数。

如果你不想将任何参数传递给脚本，可以使用：

```
1. script[] 或者
2. script
```

Zabbix服务器将查找定义为外部脚本的位置的目录（Zabbix服务器配置文件中的参数“ExternalScripts”），然后执行该命令。该命令将以Zabbix用户执行，因此，任何访问权限或环境变量都应在包装器脚本中处理，并且该命令的权限应允许该用户执行它。只有指定目录中的命令才可执行。

不要过度使用外部检查！由于每个脚本都需要Zabbix服务器启动fork进程，运行太多的脚本会降低Zabbix的性能。

用法示例

使用第一个参数“-h”执行脚本check_oracle.sh。第二个参数将由IP地址或DNS名称替代，这取决于主机属性中的选择。

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

假设主机配置为使用IP地址，Zabbix将执行：

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

外部检查结果

检查的返回值与标准错误一起通过标准输出（从Zabbix 2.0起返回完整输出，并且从Zabbix 3.4开始，执行结果的退出代码也被[checked](#)）。

在标准错误输出的情况下，文本（字符、日志或文本信息类型）的监控项将被支持。

如果没有找到请求的脚本，Zabbix服务器没有执行权限或执行退出代码不匹配0（零），将不支持监控项，并且将设置相应的错误消息。在超时的情况下，监控项将被标记为不受支持，将显示相应的错误消息，脚本的分支进程将被杀死。

12 汇总检查

概述

在汇总检查中，Zabbix通过直接从数据库中查询监控信息，然后进行信息聚合。

聚合检查不需要在被监控的主机上运行任何代理。

语法

聚合监控项Key的语法是：

```
1. groupfunc["host group","item key",itemfunc,timeperiod]
```

支持的组功能 (groupfunc) 是：

组功能	描述
<i>grpavg</i>	平均值
<i>grpmax</i>	最大值
<i>grpmin</i>	最小值
<i>grpsum</i>	值求和

可以通过逗号分隔的数组来包含多个主机组。指定父主机组将包括父组和所有包含监控项的嵌套主机组。

从聚合监控项Key引用的所有监控项必须存在并且正在收集数据。只有主机被启用并且监控项也被启用才能进行聚合计数。

如果引用的监控项Key被更改，则必须手动更新聚合监控项的Key。

支持的监控项的功能 (itemfunc) 是：

监控项功能	描述
<i>avg</i>	平均值
<i>count</i>	数值
<i>last</i>	最后一次的值
<i>max</i>	最大值
<i>min</i>	最小值
<i>sum</i>	值的和

*timeperiod*参数指定最近收集的值的时间段。为了方便，可以在此参数中使用支持的单位符号，例如'5m'（分钟）而不是'300'（秒）或'1d'（天）而不是'86400'（秒）。

在该时间段内不支持数值 (count, 前缀为 #)。

如果第三个参数（监控项功能）为*last*，服务器将忽略时间段，因此可以省略：

```
1. groupfunc["host group","item key",last]
```

如果聚合产生了浮点值，同时如果聚合的监控项信息类型为Numeric（无符号），则将其修剪为整数。

如果出现以下情况有可能不支持汇总监控项：

- 没有找到引用的监控项（如果监控项Key不正确，则不存在任何监控项或所有包含的组都不正确）
- 没有数据用来计算一个函数

用法示例

用于聚合检查的Key示例：

示例 1

主机组'MySQL Servers'的总磁盘空间。

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

示例 2

主机组'MySQL Servers'的平均处理器负载。

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

示例 3

主机组'MySQL Servers'每秒查询值的5分钟聚合。

```
1. grpavg["MySQL Servers",mysql.qps,avg,5m]
```

示例 4

多个主机组中所有主机上的平均CPU负载。

```
1. grpavg[[ "Servers A", "Servers B", "Servers C"],system.cpu.load,last]
```

13 捕捉器监控项

概述

捕捉器监控项接收传入的数据，而不是查询它。

这对于你可能想要“推送”到Zabbix的任何数据都是适用的。

要使用捕捉器监控项，你必须：

- 在Zabbix里建立一个捕捉器监控项
- 将数据送给Zabbix

配置

监控项配置

配置捕捉器监控项：

- 进入： *Configuration* → *Hosts*
- 点击 *Items* 在主机的行
- 点击 *Create item*
- 输入表单中项目的参数



需要捕捉器监控项的特定信息的字段是：

Type	这里选择 Zabbix trapper 。
Key	输入用于在发送数据时识别该监控项的密钥。
Type of information	选择与将要发送的数据格式对应的信息类型。
Allowed hosts	以逗号分隔的IP地址列表或主机名，可选择以CIDR表示法。\\如果指定，传入连接将仅从这里列出的主机接收。\\如果启用IPv6支持，则'127.0.0.1'，':: 127.0 .0.1'，':: ffff: 127.0.0.1'被平等对待，':::/0'将允许任何IPv4或IPv6地址。\\'0.0.0.0/0'可用于允许任何IPv4地址。\\注意，“IPv4兼容的IPv6地址”(0000::/96前缀)被支持，但 RFC4291 不推荐使用)。\\示例：Server = 127.0.0.1,192.168.1.0/24, ::1,2001:db8::/32,zabbix.domain 空格和 用户宏 从Zabbix 2.2.0起允许使用。

在保存监控项之前，你可能需要等待最长60秒，直到服务器从配置缓存更新中提取更改，然后才能发送值。

数据发送

在最简单的情况下，我们可以使用**zabbix_sender**实用程序发送一些“测试值”：

```
1. zabbix_sender -z <server IP address> -p 10051 -s "New host" -k trap -o "test value"
```

要发送值，要使用这些Key：

- z - 指定Zabbix服务器的IP地址
- p - 指定Zabbix服务器端口号（默认为10051）
- s - 指定主机（请确保在此处使用'技术含义'主机名称，而不是'显示意义'的名称）
- k - 指定我们刚刚定义的监控项的Key
- o - 指定要发送的实际值

Zabbix trapper进程不会扩展监控项密钥中使用的宏。

显示

这是监控的结果输出：□ == 时间戳 == 如果使用“zabbix_sender”从具有时间戳的文件发送值，则会调整这些时间戳以匹配服务器时间。例如，如果监控项的时间戳记为“10:30:50”，则“zabbix_sender”机器上的当前时间为“10:40:03”，Zabbix服务器计算机上的当前时间为“10:40 : 05”，则监控项的值将被存储在数据库中，时间戳为“10:30:52”。同样地，如果一个值首先发送到Zabbix代理，后来将其发送到Zabbix服务器，则时间戳将首先被调整为匹配Zabbix代理时间，然后将其调整为匹配Zabbix服务器时间。

14 JMX监控

Overview

JMX监控可用于监视Java应用程序的JMX计数器。

自Zabbix 2.0以来，JMX监视器以Zabbix守护进程方式运行，名为“Zabbix Java gateway”。

要检索主机上特定JMX计数器的值，Zabbix服务器查询Zabbix Java gateway，该网关又使用[JMX管理API](#)来远程查询感兴趣的应用程序。

有关Zabbix Java gateway的设置和更多详细信息，请参见各自的手册部分。

为Java应用程序启用远程JMX监视

Java应用程序不需要安装任何其它软件，但需要使用以下指定的命令行选项启动以支持远程JMX监视。

在最小情况下，如果你只想通过在本地主机上监控一个简单的Java应用程序，而不需要执行安全性，那么请使用以下选项启动它们：

```
1. java \
2. -Dcom.sun.management.jmxremote \
3. -Dcom.sun.management.jmxremote.port=12345 \
4. -Dcom.sun.management.jmxremote.authenticate=false \
5. -Dcom.sun.management.jmxremote.ssl=false \
6. -jar /usr/share/doc/openjdk-6-jre-headless/demo/jfc/Notepad/Notepad.jar
```

这使Java可以侦听来自本地主机的端口12345上的传入JMX连接，并告知不要求身份验证或SSL。

如果要允许其它接口上的连接，请将-Djava.rmi.server.hostname参数设置为该接口的IP。

如果你希望更加严格的安全性，可以使用更多Java选项。例如，下一个示例使用更多的启动选项，并将其打开到更广泛的网络，而不仅仅是本地主机。

```
1. java \
2. -Djava.rmi.server.hostname=192.168.3.14 \
3. -Dcom.sun.management.jmxremote \
4. -Dcom.sun.management.jmxremote.port=12345 \
5. -Dcom.sun.management.jmxremote.authenticate=true \
6. -Dcom.sun.management.jmxremote.password.file=/etc/java-6-openjdk/management/jmxremote.password \
7. -Dcom.sun.management.jmxremote.access.file=/etc/java-6-openjdk/management/jmxremote.access \
8. -Dcom.sun.management.jmxremote.ssl=true \
9. -Djavax.net.ssl.keyStore=$YOUR_KEY_STORE \
10. -Djavax.net.ssl.keyStorePassword=$YOUR_KEY_STORE_PASSWORD \
11. -Djavax.net.ssl.trustStore=$YOUR_TRUST_STORE \
12. -Djavax.net.ssl.trustStorePassword=$YOUR_TRUST_STORE_PASSWORD \
13. -Dcom.sun.management.jmxremote.ssl.need.client.auth=true \
14. -jar /usr/share/doc/openjdk-6-jre-headless/demo/jfc/Notepad/Notepad.jar
```

这些设置中的大多数（如果不是全部）可以在`/etc/java-6-openjdk/management/management.properties`（或者系统上的任何文件位置）中指定。

请注意，如果你希望使用SSL，则必须通过向Java网关添加“`-Djavax.net.ssl.*`”选项来修改`startup.sh`脚本，以便知道在哪里可以找到密钥和信任存储。

查看[使用JMX监控和管理](#)的详细说明。

在Zabbix Web管理端配置JMX接口和监控项

Java网关在运行时，服务器知道在哪里找到它，而Java应用程序开始进行远程JMX监视，现在可以在Zabbix GUI中配置接口和监控项了。

配置JMX接口

首先在感兴趣的主机上创建一个JMX类型的接口：



添加JMX代理项

对于你感兴趣的每个JMX计数器，都可以添加一个**JMX代理**类型的监控项。如果你已经在Java应用程序上配置了身份验证，那么你还可以指定用户名和密码。

下面的屏幕截图中的关键配置 `jmx["java.lang:type=Memory", "HeapMemoryUsage.used"]`。关键包括2个参数：

- `object name` - 它代表MBean的对象名称
- `attribute name` - 一个MBean属性名称，可选的复合数据字段名称以点分隔

有关JMX监控项Key的更多详细信息，请参阅下文。



如果要监视一个“true”或“false”的布尔计数器，那么你将信息的类型指定为“`Numeric (unsigned)`”，数据类型指定为“`Boolean`”。服务器将分别将布尔值存储为1或0。

JMX监控项Key更详细的信息

简单属性

An MBean object name is nothing but a string which you define in your Java application. An attribute name, on the other hand, can be more complex. In case an attribute returns primitive data type (an integer, a string etc.) there is nothing to worry about, the key will look like this: MBean对象名称只不过是在Java应用程序中定义的字符串。另一方面，属性名称可能更复杂。如果一个属性返回原始数据类型（一个整数、一个字符串等），那就没有什么可担心的了，这个Key类似如下：

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

在此示例中，对象名称为“com.example: Type = Hello”，属性名称为“weight”，可能返回的值类型应为“Numeric (float)”。

属性返回复合数据

当属性返回复合数据时将会更加复杂。例如：属性名称是“apple”，它返回一个表示其参数的哈希，如“weight”，“color”等。Key可能如下所示：

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

这是通过使用点符号来分割属性名称和哈希键的方式。同样的，如果一个属性返回嵌套的复合数据，这些部分由一个点分隔开：

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

关于点的问题

到现在为止还挺好。但是，如果属性名称或散列键包含点符号怎么办？这是一个例子：

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

这是一个问题。如何告诉Zabbix属性名称是“all.fruits”，而不只是“全部”？如何区分属于名称和散列键点名称的一部分点？

在2.0.4之前Zabbix Java网关无法处理这种情况，用户留下了UNSUPPORTED项。从2.0.4开始解决了此问题，所有你需要做的就是用一个反斜线来转义名字的一部分点：

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

同样的方法，如果你的散列键包含一个你可以回避的点：

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

其它问题

反斜杠字符也应该被转义：

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

如果对象名称或属性名称包含空格或逗号、双引号：

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

这就是全部了。JMX监控快乐！

15 ODBC监控

概述

ODBC监控对应于Zabbix Web管理端中的数据库监控器监控项类型。

ODBC是用于访问数据库管理系统（DBMS）的C语言中间件API。ODBC由Microsoft开发，后来移植到其它平台。

Zabbix可以查询ODBC支持的任何数据库。为了实现监控，Zabbix不直接连接到数据库，而是使用ODBC中设置的ODBC接口和驱动。该功能允许为多个目的更加有效地监控不同的数据库 - 例如，监控特定的数据库队列、使用统计信息等。Zabbix支持unixODBC，它是最常用的开源ODBC API实现之一。

安装unixODBC

安装unixODBC的建议方法是使用Linux操作系统的默认软件包存储库。在流行的Linux发行版中，unixODBC默认包含在软件包存储库中。如果不可用，可以在unixODBC主页获取：<http://www.unixodbc.org/download.html>。

使用yum包管理器在基于RedHat/Fedora的系统上安装unixODBC：

```
1. shell> yum -y install unixODBC unixODBC-devel
```

使用zypper软件包管理器在基于SUSE的系统上安装unixODBC：

```
1. # zypper in unixODBC-devel
```

unixODBC-devel包需要使用unixODBC support来编译Zabbix。

安装unixODBC驱动

应该为将要被监控的数据库安装unixODBC数据库驱动。 unixODBC有一个受支持的数据库和驱动程序的列表：<http://www.unixodbc.org/drivers.html>。 在一些Linux发行版中，数据库驱动程序包含在包存储库中。 使用yum包管理器在基于RedHat/Fedora的系统上安装MySQL数据库驱动：

```
1. shell> yum install mysql-connector-odbc
```

使用zypper软件包管理器在基于SUSE的系统上安装MySQL数据库驱动程序：

```
1. zypper in MyODBC-unixODBC
```

配置unixODBC

通过编辑`odbcinst.ini`和`odbc.ini`文件来完成ODBC配置。要确认配置文件位置，请键入：

```
1. shell> odbcinst -j
```

odbcinst.ini用于列出已安装的ODBC数据库驱动程序：

```
1. [mysql]
2. Description = ODBC for MySQL
3. Driver      = /usr/lib/libmyodbc5.so
```

参数详细信息：

属性	描述
<i>mysql</i>	数据库驱动程序名称。
<i>Description</i>	数据库驱动描述。
<i>Driver</i>	数据库驱动程序库位置。

odbc.ini用于定义数据源：

```
1. [test]
2. Description = MySQL test database
3. Driver      = mysql
4. Server      = 127.0.0.1
5. User        = root
6. Password    =
7. Port        = 3306
8. Database    = zabbix
```

参数详细信息：

属性	描述
<i>test</i>	数据源名称 (DSN)。
<i>Description</i>	数据源描述。
<i>Driver</i>	数据库驱动名称。 - 被指定在文件 <code>odbcinst.ini</code>
<i>Server</i>	数据库服务器的 IP/DNS。
<i>User</i>	数据库连接的用户。
<i>Password</i>	数据库连接用户的密码。
<i>Port</i>	数据库连接端口。
<i>Database</i>	数据库名称。

要验证ODBC连接是否正常工作，应测试与数据库的连接。 可以使用isql实用程序（包含在unixODBC包中）：

```
1. shell> isql test
2. +-----+
3. | Connected! |
4. | |
5. | sql-statement |
6. | help [tablename] |
7. | quit |
8. |
```

```

9. +-----+
10. SQL>

```

使用ODBC support编译Zabbix

要启用ODBC支持，Zabbix应该使用以下标志进行编译：

1. `--with-unixodbc[=ARG]` 使用odbc驱动程序与unixODBC包

[从源代码](#)查看Zabbix安装的更多信息。

在Zabbix Web前端配置监控项

配置数据库的 [监控项](#)：



专用于数据库监控项的必要输入：

Type	选择数据库监控器。
Key	输入 <code>db.odbc.select [uniquedescription, data_source_name]</code> 。这里唯一的描述将用于识别触发器中的监控项等。必须按照 <code>odbc.ini</code> 中的指定设置数据源名称 (DSN)。
_User name	输入数据库用户名 (如果用户在 <code>odbc.ini</code> 中指定，则可选)
Password	输入数据库用户密码 (如果在 <code>odbc.ini</code> 中指定密码，则为可选项)
SQL query	输入SQL查询
Type of information	了解查询将返回什么类型的信息很重要，以便在此处正确选择。 使用不正确的类型的信息监控项将不受支持。

重要信息

- 该查询的执行时间不能超过服务器上的[Timeout](#)参数。从Zabbix 2.0.8开始，[Timeout](#)参数值也用作ODBC登录超时（请注意，根据ODBC驱动程序，登录超时设置可能会被忽略）。
- 该查询只能返回一个值。
- 如果查询返回多个列，则只读取第一列。
- 如果查询返回多行，则只读取第一行。
- SQL命令必须以 `select` 开始。
- SQL命令不能包含任何换行符。
- 另请参考ODBC的[已知问题](#)。

错误信息

从Zabbix 2.0.8开始，ODBC错误消息被构造为字段以提供更详细的信息。示例：

```
1. Cannot execute ODBC query:[SQL_ERROR]:[42601][7][ERROR: syntax error at or near ";" ; Error while executing the query]|
2. ----- | -----
----- | -----
3. | | `-- Native error code           `-- error message.
`-- Record separator
4. | | `-- SQLState
5. `-- Zabbix message  `-- ODBC return code
```

请注意，错误消息长度限制为2048字节，因此消息可以被截断。如果有多个ODBC诊断记录，Zabbix会尝试把它们连起来，只要长度限制允许。

3 历史与趋势

概述

历史和趋势是在Zabbix中存储数据的两种方式。

历史保持每个收集的值，而趋势是每个小时的平均信息。

保持历史

你可以设置保持多天的历史记录：

- 在监控项属性中 [form](#)
- 在批量更新监控项时
- 在[配置housekeeper任务时](#)

任何较旧的数据将由housekeeper移除。

一般来讲，强烈建议保持的历史数据尽可能少，而不是使数据库过量保存历史信息。

你可以保留更多的趋势数据，而不是保存很长的历史数据。例如，你可以保存14天的历史和5年的趋势数据。

你可以通过参考历史与趋势数据的 [数据库大小页面](#)，了解需要多少空间。

虽然保持较短的历史记录，你仍然可以查看图中的旧数据，因为图形将使用趋势值显示旧数据。

如果历史记录设置为“0”，则该监控项将仅更新库存，不会对触发功能进行评估。

作为保存历史的替代方法，考虑使用可加载模块的[历史信息导出](#)功能。

保存趋势数据

趋势是一种内置的历史数据压缩机制，可存储数字类型的每小时的最小值、最大值、平均值和总数值。

你可以设置保留几天的趋势：

- 在监控项属性设置时 [form](#)
- 在批量更新监控项时
- 在设置Housekeeper任务时

趋势通常可以比历史保持更长时间。任何较旧的数据将由housekeeper移除。

如果趋势设置为“0”，Zabbix服务器根本不会计算或存储趋势。

趋势计算和存储使用原始值相同的数据类型。结果是无符号数据类型值的平均值计算是四舍五入的，并且值间隔越小，结果将越不精确。例如，如果监控项的值为0和1，平均值将为0，而不是0.5。

此外，重新启动服务器可能会导致当前小时的无符号数据类型平均值计算的精度损失。

4 用户自定义参数

概述

有时你可能想要运行一个代理检查，它不是用Zabbix预定义的，这时你会用到用户参数来帮忙。

你可以编写一个命令来检索所需的数据，并将其包含在用户参数[代理配置文件](#)中（'UserParameter' 配置参数）。

用户参数具有以下语法：

```
1. UserParameter=<key>, <command>
```

如你看到的，用户参数还包含一个Key。一个用户参数包含一个Key。输入你选择的Key，这将很容易引用（它在主机中必须是唯一的）。重新启动代理。

然后，当[配置一个监控项](#)时，输入要从执行的用户参数中引用该命令的Key。

用户参数是由Zabbix代理执行的命令。最多可以返回512KB的数据。`/bin/sh` 在UNIX操作系统下用于命令行解释器。用户参数满足代理检查超时，如果达到超时，则分支用户参数进程终止。

请参考：

- [使用用户参数分步教程](#)
- [命令执行](#)

用户参数示例

简单的命令：

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

代理将始终使用'ping'键为一个监控项返回'1'。

一个更复杂的例子：

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

如果MySQL服务器是活动状态，代理将返回'1'，否则为0。

灵活的用户参数

灵活的用户参数使用Key接受参数。这样一个灵活的用户参数可以作为创建几个监控项的基础。

灵活的用户参数具有以下语法：

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

参数	描述
Key	唯一的监控项Key。[]定义该Key接收括号内的参数。在配置监控项时给出参数。
<i>*Command</i>	执行命令以评估Key的值。仅适用于灵活的用户参数：你可以使用命令中的位置引用\$ 1 ... \$ 9来引用监控项Key中的相应参数。Zabbix解析监控项Key的[]中包含的参数，并相应地替换\$ 1, ..., \$ 9。\$ 0将由原始命令（在扩展\$ 0, ..., \$ 9之前）替换为运行。不管它们是用双引号（"）还是单引号（'）括起来，都会解析位置引用。要使用位置引用不变，请指定双美元符号 - 例如，awk'{print \$\$2}'。在这种情况下，执行命令时，“\$\$2”实际上会变成“\$2”。

使用\$符号的位置引用仅由灵活的用户参数搜索并由Zabbix代理替代。对于简单的用户参数，跳过此类引用处理，因此任何\$号引用都不是必需的。

默认情况下，用户参数中不允许使用某些符号。请查阅 [不安全的用户参数](#)的完整列表文档。

示例 1

有些事情很简单：

```
1. UserParameter=ping[*],echo $1
```

我们可以定义无限数量的监控项，用于监视所有具有格式ping [something]的设置。

- ping[0] - 总是返回 '0'
- ping[aaa] - 总是返回 'aaa'

示例 2

让我们增添更多的意义！

```
1. UserParameter=mysql.ping[*],mysqladmin -u$1 -p$2 ping | grep -c alive
```

此参数可用于监视MySQL数据库的可用性。我们可以传递用户名和密码：

```
1. mysql.ping[zabbix,our_password]
```

示例 3

有多少行匹配文件中的正则表达式？

```
1. UserParameter=wc[*],grep -c "$2" $1
```

此参数可用于计算文件中的行数。

```
1. wc[/etc/passwd,root]
2. wc[/etc/services,zabbix]
```

命令结果

命令的返回值与标准错误一起按照标准输出。

在标准错误输出的情况下，文本（字符、日志或文本信息类型）监控项将被支持。

返回文本（字符、日志、文本信息类型）的用户参数可以返回空格。 在无效结果的情况下，或执行退出代码不匹配0（零）的监控项将不被支持。

1 扩展Zabbix代理

本教程提供了有关如何使用[用户自定义参数](#)扩展Zabbix代理功能的分步说明。

步骤 1

编写一个脚本或命令行来检索所需的参数。

例如，我们可以编写以下命令来获取MySQL服务器执行的查询总数：

```
1. mysqladmin -uroot status | cut -f4 -d";" | cut -f1 -d"S"
```

执行时，该命令返回SQL查询的总数。

步骤 2

将命令添加到zabbix_agentd.conf中：

```
1. UserParameter=mysql.questions,mysqladmin -uroot status | cut -f4 -d";" | cut -f1 -d"S"
```

mysql.questions 是一个唯一的标识符。它可以是任何有效的Key标识符，例如，*queries*。

通过使用带有“-t”标志的Zabbix代理测试此参数（但是如果在根目录下运行请注意，当作为守护程序启动时，代理可能具有不同的权限）：

```
1. zabbix_agentd -t mysql.questions
```

步骤 3

重新启动Zabbix代理。

代理将重新加载配置文件。

使用[zabbix_get](#)实用程序测试此参数。

步骤 4

使用Key=mysql.questions添加新监控项到被监控的主机。监控项的类型必须是Zabbix Agent或Zabbix Agent（活动）。

请注意，必须在Zabbix服务器上正确设置返回值的类型。否则Zabbix将不接收它们。

5 可加载模块

5.1 概述

可加载模块提供了一种关于Zabbix性能扩展的选项。

可以通过以下方式扩展Zabbix功能：

- `user parameters` (代理指标)
- `external checks` (无代理监控)
- `system.run[]` Zabbix agent item.

它们能运行的很好，但有一个主要的缺点，即`fork()`。Zabbix必须在每次处理用户指标时分配一个新进程，这对于性能不利。这一般来讲不是严重的问题，但是在监控嵌入式系统，具有大量监控参数或具有复杂逻辑或启动时间很长的复杂脚本的情况下，这也许是一个严重的问题。

支持可加载模块提供了扩展Zabbix代理、服务器、代理服务器的方法，同时不牺牲性能。

可加载模块基本上是Zabbix守护程序使用的共享库，并在启动时加载。该库应包含某些功能，以便Zabbix进程可能会检测到该文件确实是可加载和使用的模块。

可加载模块具有许多优点。卓越的性能和可实现任何逻辑的能力是非常重要的，但更最重要的优势是使用和共享Zabbix模块的开发能力。它有助于可靠的维护，并有助于更轻松、更独立于Zabbix核心代码库提供新功能。

5.2 模块API

为了将共享库按照Zabbix模块进行处理，它应该执行和导出多个功能。Zabbix模块API目前有六个功能，其中只有一个强制性的，另外五个是可选的。

5.2.1 强制接口

唯一的强制功能是 `zbx_module_api_version()`：

```
1. int zbx_module_api_version(void);
```

此函数应该返回此模块的API版本，为了模块加载此版本必须匹配Zabbix支持的模块API版本。Zabbix支持的模块API版本为`ZBXMODULEAPIVERSION`。所以这个函数应该返回一个常数。用于此目的的旧常数`ZBX_MODULEAPIVERSION_ONE`现在被定义为等于`ZBX_MODULEAPI_VERSION`以保持兼容性，但不推荐使用。

5.2.2 可选接口

可选的接口是 `zbx_module_init()`, `zbx_module_item_list()`, `zbx_module_item_timeout()`, `zbx_module_history_write_cbs()` 和 `zbx_module_uninit()`:

```
1. int zbx_module_init(void);
```

该功能应该对模块进行必要的初始化（如果有的话）。如果成功，应该返回`ZBX_MODULE_OK`。否则，应该返回

ZBX_MODULE_FAIL。在后一种情况下，Zabbix将不会启动。

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

此功能应返回模块支持的监控项列表。每个监控项都是在ZBX_METRIC结构中定义的，有关详细信息，请参阅下面的部分。列表由ZBX_METRIC结构终止，“key”字段为NULL。

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

如果模块导出zbx_module_item_list()，那么Zabbix使用该函数来指定Zabbix配置文件中由模块实现的监控项检查应该遵循的超时设置。这里，“timeout”参数是以秒为单位。

```
1. ZBX_HISTORY_WRITE_CBS zbx_module_history_write_cbs(void);
```

该函数应该返回回调函数Zabbix服务器或代理服务器将用于导出不同数据类型的历史记录。回调函数作为ZBX_HISTORY_WRITE_CBS结构的字段提供，如果模块对某种类型的历史不感兴趣，则字段可以为NULL。

```
1. int zbx_module_uninit(void);
```

此功能应执行必要的反初始化（如果有的话），如释放分配的资源、关闭文件描述符等。

模块加载时、Zabbix启动时，所有功能除了zbx_module_uninit()都将被调用一次。在卸载模块时、Zabbix关闭时会调用一次。

5.2.3 定义监控项

每个监控项都在ZBX_METRIC结构中定义：

```
1. typedef struct
2. {
3.     char      *key;
4.     unsigned   flags;
5.     int       (*function)();
6.     char      *test_param;
7. }
8. ZBX_METRIC;
```

这里，key是监控项Key（例如，“dummy.random”），标志是CF_HAVEPARAMS或0（取决于监控项是否接受参数），function是实现该项目的C函数（例如，“zbx_module_dummy_random”），test_param是使用“-p”标志启动Zabbix代理时使用的参数列表（例如，“1,1000”，可以为NULL）。示例定义如下所示：

```
1. static ZBX_METRIC keys[] =
2. {
3.     { "dummy.random", CF_HAVEPARAMS, zbx_module_dummy_random, "1,1000" },
4.     { NULL }
5. }
```

实现一个监控项的每个函数应该接受两个指针参数，第一个是AGENT_REQUEST类型，第二个是AGENT_RESULT类型的参数：

```

1. int zbx_module_dummy_random(AGENT_REQUEST *request, AGENT_RESULT *result)
2. {
3.     ...
4.
5.     SET_UI64_RESULT(result, from + rand() % (to - from + 1));
6.
7.     return SYSINFO_RET_OK;
8. }
```

如果监控项的值成功获取，这些函数应返回SYSINFO_RET_OK。否则，它们应该返回SYSINFO_RET_FAIL。有关如何从AGENT_REQUEST获取信息以及如何在AGENT_RESULT中设置信息的详细信息，请参阅下面的示例“dummy”模块。

5.2.4 提供历史记录导出回调

模块可以指定按类型导出历史数据的函数：Numeric (float)，Numeric (unsigned)，Character，Text和Log：

```

1. typedef struct
2. {
3.     void (*history_float_cb)(const ZBX_HISTORY_FLOAT *history, int history_num);
4.     void (*history_integer_cb)(const ZBX_HISTORY_INTEGER *history, int history_num);
5.     void (*history_string_cb)(const ZBX_HISTORY_STRING *history, int history_num);
6.     void (*history_text_cb)(const ZBX_HISTORY_TEXT *history, int history_num);
7.     void (*history_log_cb)(const ZBX_HISTORY_LOG *history, int history_num);
8. }
9. ZBX_HISTORY_WRITE_CBS;
```

每个人都应该以“history_num”元素的“history”数组为参数。根据要导出的历史数据类型，“history”分别是以下结构的数组：

```

1. typedef struct
2. {
3.     zbx_uint64_t itemid;
4.     int clock;
5.     int ns;
6.     double value;
7. }
8. ZBX_HISTORY_FLOAT;
9.
10. typedef struct
11. {
12.     zbx_uint64_t itemid;
13.     int clock;
14.     int ns;
15.     zbx_uint64_t value;
16. }
17. ZBX_HISTORY_INTEGER;
```

```

18.
19. typedef struct
20. {
21.     zbx_uint64_t      itemid;
22.     int              clock;
23.     int              ns;
24.     const char       *value;
25. }
26. ZBX_HISTORY_STRING;
27.
28. typedef struct
29. {
30.     zbx_uint64_t      itemid;
31.     int              clock;
32.     int              ns;
33.     const char       *value;
34. }
35. ZBX_HISTORY_TEXT;
36.
37. typedef struct
38. {
39.     zbx_uint64_t      itemid;
40.     int              clock;
41.     int              ns;
42.     const char       *value;
43.     const char       *source;
44.     int              timestamp;
45.     int              logeventid;
46.     int              severity;
47. }
48. ZBX_HISTORY_LOG;

```

数据写入Zabbix数据库并保存在值缓存中之后，Zabbix服务器或代理服务器历史记录进程将在历史记录同步过程结束时使用回调。

只有原始值可用于通过代理模块导出。（自定义乘法器将不能应用，三角形将不被计算等）

5.2.5 构建模块

模块是要在Zabbix源代码树中构建的，因为模块API依赖于在Zabbix头文件中定义的一些数据结构。

可加载模块最重要的头是**include/module.h**，它定义了这些数据结构。另一个有用的头是**include/sysinc.h**，它执行包含必要的系统头，这本身就有助于**include/module.h**正常工作。

为了**include/module.h**和**include/sysinc.h**被包括在内，应该首先在**.bbbix**源代码树的根目录中运行**./configure**命令（不带参数）。这将创建**include/config.h**文件，其中包括/**sysinc.h**依赖。（如果你获得Zabbix源代码作为Subversion存储库检出，则**./configure**脚本不存在，应首先运行**./bootstrap.sh**命令生成它。）

在上述情况都考虑到之后，一切都为建立模块准备好了。该模块应包括**sysinc.h**和**module.h**，构建脚本应确保这两个文件位于包含路径中。有关详细信息，请参见下面的“dummy”模块示例。

另一个有用的头是`include/log.h`, 它定义了`zabbix_log()`函数, 可用于记录和调试目的。

5.3 配置参数

Zabbix代理、服务器和代理服务器支持两种模块参数：

- `LoadModulePath` – 可加载模块位置的全路径
- `LoadModule` – 模块在启动时加载。模块必须位于`LoadModulePath`指定的目录中。允许包含多个`LoadModule`参数。

例如, 要扩展Zabbix代理, 我们可以添加以下参数:

```
1. LoadModulePath=/usr/local/lib/zabbix/agent/
2. LoadModule=mariadb.so
3. LoadModule=apache.so
4. LoadModule=kernel.so
5. LoadModule=dummy.so
```

代理启动后, 将从`/usr/local/lib/zabbix/agent`目录加载`mariadb.so`, `apache.so`, `kernel.so`和`dummy.so`模块。如果模块丢失、权限不匹配或共享库不是Zabbix模块, 则会失败。

5.4 Web前端的配置

Zabbix代理、服务器和代理服务器支持可加载模块。因此, Zabbix Web前端中的监控项类型取决于模块的加载位置。如果模块加载到代理中, 则监控项类型应为“Zabbix代理”或“Zabbix代理(活动)”。如果模块加载到服务器或代理服务器, 则监控项类型应为“简单检查”。

通过Zabbix模块的历史导出不需要任何前端配置。如果模块由服务器或代理服务器成功加载, 并提供返回至少一个非NULL回调函数的`zbx_module_history_write_cbs()`函数, 那么历史导出将自动启用。

5.5 Dummy模块

Zabbix包含用C语言编写的示例模块。该模块位于`src/modules/dummy`下:

```
1. [email protected]:~trunk/src/modules/dummy$ ls -l
2. -rw-rw-r-- 1 alex alex 9019 Apr 24 17:54 dummy.c
3. -rw-rw-r-- 1 alex alex    67 Apr 24 17:54 Makefile
4. -rw-rw-r-- 1 alex alex  245 Apr 24 17:54 README
```

该模块有详细的文档, 它可以用作你自己的模块的模板。

在`./configure`已经在Zabbix源代码树的根目录中运行, 如上所述, 只需运行`make`即可构建`dummy.so`。

```
1. /*
2. ** Zabbix
3. ** Copyright (C) 2001-2016 Zabbix SIA
4. **
5. ** This program is free software; you can redistribute it and/or modify
6. ** it under the terms of the GNU General Public License as published by
```

```

7. ** the Free Software Foundation; either version 2 of the License, or
8. ** (at your option) any later version.
9. **
10. ** This program is distributed in the hope that it will be useful,
11. ** but WITHOUT ANY WARRANTY; without even the implied warranty of
12. ** MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
13. ** GNU General Public License for more details.
14. **
15. ** You should have received a copy of the GNU General Public License
16. ** along with this program; if not, write to the Free Software
17. ** Foundation, Inc., 51 Franklin Street, Fifth Floor, Boston, MA 02110-1301, USA.
18. **/
19.
20. #include "sysinc.h"
21. #include "module.h"
22.
23. /* the variable keeps timeout setting for item processing */
24. static int item_timeout = 0;
25.
26. /* module SHOULD define internal functions as static and use a naming pattern different from Zabbix internal
   */
27. /* symbols (zbx_*) and loadable module API functions (zbx_module_*) to avoid conflicts
   */
28. static int dummy_ping(AGENT_REQUEST *request, AGENT_RESULT *result);
29. static int dummy_echo(AGENT_REQUEST *request, AGENT_RESULT *result);
30. static int dummy_random(AGENT_REQUEST *request, AGENT_RESULT *result);
31.
32. static ZBX_METRIC keys[] =
33. /*      KEY          FLAG          FUNCTION      TEST PARAMETERS */
34. {
35.     {"dummy.ping",      0,        dummy_ping,      NULL},
36.     {"dummy.echo",      CF_HAVEPARAMS, dummy_echo,      "a message"},
37.     {"dummy.random",    CF_HAVEPARAMS, dummy_random,    "1,1000"},
38.     {NULL}
39. };
40.
41. ****
42. *
43. * Function: zbx_module_api_version
44. *
45. * Purpose: returns version number of the module interface
46. *
47. * Return value: ZBX_MODULE_API_VERSION - version of module.h module is
48. *                 compiled with, in order to load module successfully Zabbix
49. *                 MUST be compiled with the same version of this header file
50. *
51. ****
52. int zbx_module_api_version(void)
53. {
54.     return ZBX_MODULE_API_VERSION;
55. }
56.
57. ****
58. *

```

```

59.  * Function: zbx_module_item_timeout
60.  *
61.  * Purpose: set timeout value for processing of items
62.  *
63.  * Parameters: timeout - timeout in seconds, 0 - no timeout set
64.  *
65. ****
66. void     zbx_module_item_timeout(int timeout)
67. {
68.     item_timeout = timeout;
69. }
70.
71. ****
72. *
73. * Function: zbx_module_item_list
74. *
75. * Purpose: returns list of item keys supported by the module
76. *
77. * Return value: list of item keys
78. *
79. ****
80. ZBX_METRIC      *zbx_module_item_list(void)
81. {
82.     return keys;
83. }
84.
85. static int      dummy_ping(AGENT_REQUEST *request, AGENT_RESULT *result)
86. {
87.     SET_UI64_RESULT(result, 1);
88.
89.     return SYSINFO_RET_OK;
90. }
91.
92. static int      dummy_echo(AGENT_REQUEST *request, AGENT_RESULT *result)
93. {
94.     char      *param;
95.
96.     if (1 != request->nparam)
97.     {
98.         /* set optional error message */
99.         SET_MSG_RESULT(result, strdup("Invalid number of parameters."));
100.        return SYSINFO_RET_FAIL;
101.    }
102.
103.    param = get_rparam(request, 0);
104.
105.    SET_STR_RESULT(result, strdup(param));
106.
107.    return SYSINFO_RET_OK;
108. }
109.
110. ****
111. *

```

```

112. * Function: dummy_random
113. *
114. * Purpose: a main entry point for processing of an item
115. *
116. * Parameters: request - structure that contains item key and parameters
117. *           request->key - item key without parameters
118. *           request->nparam - number of parameters
119. *           request->timeout - processing should not take longer than
120. *                         this number of seconds
121. *           request->params[N-1] - pointers to item key parameters
122. *
123. *           result - structure that will contain result
124. *
125. * Return value: SYSINFO_RET_FAIL - function failed, item will be marked
126. *                 as not supported by zabbix
127. *                 SYSINFO_RET_OK - success
128. *
129. * Comment: get_rparam(request, N-1) can be used to get a pointer to the Nth
130. *           parameter starting from 0 (first parameter). Make sure it exists
131. *           by checking value of request->nparam.
132. *
133. ****
134. static int    dummy_random(AGENT_REQUEST *request, AGENT_RESULT *result)
135. {
136.     char      *param1, *param2;
137.     int       from, to;
138.
139.     if (2 != request->nparam)
140.     {
141.         /* set optional error message */
142.         SET_MSG_RESULT(result, strdup("Invalid number of parameters."));
143.         return SYSINFO_RET_FAIL;
144.     }
145.
146.     param1 = get_rparam(request, 0);
147.     param2 = get_rparam(request, 1);
148.
149.     /* there is no strict validation of parameters for simplicity sake */
150.     from = atoi(param1);
151.     to = atoi(param2);
152.
153.     if (from > to)
154.     {
155.         SET_MSG_RESULT(result, strdup("Invalid range specified."));
156.         return SYSINFO_RET_FAIL;
157.     }
158.
159.     SET_UI64_RESULT(result, from + rand() % (to - from + 1));
160.
161.     return SYSINFO_RET_OK;
162. }
163.
164. ****

```

```

165. *
166. * Function: zbx_module_init
167. *
168. * Purpose: the function is called on agent startup
169. *           It should be used to call any initialization routines
170. *
171. * Return value: ZBX_MODULE_OK - success
172. *                 ZBX_MODULE_FAIL - module initialization failed
173. *
174. * Comment: the module won't be loaded in case of ZBX_MODULE_FAIL
175. *
176. *****/
177. int     zbx_module_init(void)
178. {
179.     /* initialization for dummy.random */
180.     srand(time(NULL));
181.
182.     return ZBX_MODULE_OK;
183. }
184.
185. *****/
186. *
187. * Function: zbx_module_uninit
188. *
189. * Purpose: the function is called on agent shutdown
190. *           It should be used to cleanup used resources if there are any
191. *
192. * Return value: ZBX_MODULE_OK - success
193. *                 ZBX_MODULE_FAIL - function failed
194. *
195. *****/
196. int     zbx_module_uninit(void)
197. {
198.     return ZBX_MODULE_OK;
199. }
200.
201. *****/
202. *
203. * Functions: dummy_history_float_cb
204. *             dummy_history_integer_cb
205. *             dummy_history_string_cb
206. *             dummy_history_text_cb
207. *             dummy_history_log_cb
208. *
209. * Purpose: callback functions for storing historical data of types float,
210. *           integer, string, text and log respectively in external storage
211. *
212. * Parameters: history      - array of historical data
213. *               history_num - number of elements in history array
214. *
215. *****/
216. static void    dummy_history_float_cb(const ZBX_HISTORY_FLOAT *history, int history_num)
217. {

```

```

218.     int      i;
219.
220.     for (i = 0; i < history_num; i++)
221.     {
222.         /* do something with history[i].itemid, history[i].clock, history[i].ns, history[i].value, ... */
223.     }
224. }
225.
226. static void    dummy_history_integer_cb(const ZBX_HISTORY_INTEGER *history, int history_num)
227. {
228.     int      i;
229.
230.     for (i = 0; i < history_num; i++)
231.     {
232.         /* do something with history[i].itemid, history[i].clock, history[i].ns, history[i].value, ... */
233.     }
234. }
235.
236. static void    dummy_history_string_cb(const ZBX_HISTORY_STRING *history, int history_num)
237. {
238.     int      i;
239.
240.     for (i = 0; i < history_num; i++)
241.     {
242.         /* do something with history[i].itemid, history[i].clock, history[i].ns, history[i].value, ... */
243.     }
244. }
245.
246. static void    dummy_history_text_cb(const ZBX_HISTORY_TEXT *history, int history_num)
247. {
248.     int      i;
249.
250.     for (i = 0; i < history_num; i++)
251.     {
252.         /* do something with history[i].itemid, history[i].clock, history[i].ns, history[i].value, ... */
253.     }
254. }
255.
256. static void    dummy_history_log_cb(const ZBX_HISTORY_LOG *history, int history_num)
257. {
258.     int      i;
259.
260.     for (i = 0; i < history_num; i++)
261.     {
262.         /* do something with history[i].itemid, history[i].clock, history[i].ns, history[i].value, ... */
263.     }
264. }
265.
266. ****
267. *
268. * Function: zbx_module_history_write_cbs
269. *
270. * Purpose: returns a set of module functions Zabbix will call to export

```

```

271. *      different types of historical data *
272. *
273. * Return value: structure with callback function pointers (can be NULL if *
274. *           module is not interested in data of certain types) *
275. *
276. *****/
277. ZBX_HISTORY_WRITE_CBS    zbx_module_history_write_cbs(void)
278. {
279.     static ZBX_HISTORY_WRITE_CBS    dummy_callbacks =
280.     {
281.         dummy_history_float_cb,
282.         dummy_history_integer_cb,
283.         dummy_history_string_cb,
284.         dummy_history_text_cb,
285.         dummy_history_log_cb,
286.     };
287.
288.     return dummy_callbacks;
289. }

```

该模块导出三个新监控项：

- `dummy.ping` - 总是返回 '1'
- `dummy.echo[param1]` - 返回第一个参数，例如 `dummy.echo[ABC]` 将返回ABC
- `dummy.random[param1, param2]` - 返回param1-param2范围内的随机数，例如，`dummy.random[1,1000000]`

5.6 限制

仅针对Unix平台实现可加载模块的支持。这意味着它不适用于Windows代理。

在某些情况下，模块可能需要从`zabbix_agentd.conf`读取相关的配置参数。如果你需要使用模块来使用某些配置参数，那么你应该可以实现对特定于模块的配置文件的解析。

6 Windows性能计数器

概述

你可以使用`_perf_counter[]_Key`有效的监控Windows性能计数器。

示例：

```
1. perf_counter["\Processor(0)\Interrupts/sec"]
```

或者

```
1. perf_counter["\Processor(0)\Interrupts/sec", 10]
```

有关使用此Key的更多信息，请参阅 [特定于Windows的监控项Key](#) .

为了获得可用于监控的性能计数器的完整列表，你可以运行：

```
1. typeperf -qx
```

数字表达

由于性能计数器的命名可能因不同的Windows服务器而异，具体取决于本地设置，因此在创建用于监视具有不同区域设置的多台Windows计算机的模板时会引发一定的问题。

同时，每个性能计数器也可以通过其数字形式来引用，无论语言设置如何，它都是唯一的，因此你可以使用数字表示而不是字符串。

找出数字等价物，运行 `regedit`，然后找到 `HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows NT\CurrentVersion\Perflib\009` .

注册表项包含这样的信息：

```
1. 1
2. 1847
3. 2
4. System
5. 4
6. Memory
7. 6
8. % Processor Time
9. 10
10. File Read Operations/sec
11. 12
12. File Write Operations/sec
13. 14
14. File Control Operations/sec
15. 16
```

```
16. File Read Bytes/sec  
17. 18  
18. File Write Bytes/sec  
19. ....
```

这里，你可以找到性能计数器的每个字符串的相应数字，例如 '\System\% Processor Time'：

```
1. System -> 2  
2. % Processor Time -> 6
```

然后，你可以使用这些数字来表示路径：

```
1. \2\6
```

性能计数器参数

你可以部署一些PerfCounter参数来监控Windows性能计数器。

例如，你可以将它们添加到Zabbix代理配置文件中：

```
1. PerfCounter=UserPerfCounter1, "\Memory\Page Reads/sec", 30  
2. 或者  
3. PerfCounter=UserPerfCounter2, "\4\24", 30
```

使用这些参数，你可以简单地使用UserPerfCounter1 或者 UserPerfCounter2 作为创建相应监控项的Key。

记住在更改配置文件后重新启动Zabbix代理。

故障处理

有时，Zabbix代理无法在基于Windows 2000的系统中检索性能计数值，因为pdh.dll文件已过时。它显示为Zabbix代理和服务器日志文件中的失败消息。在这种情况下，pdh.dll应该更新到更新的5.0.2195.2668版本。

7 批量更新

概述

有时你可能想要一次更改多个监控项的某些属性。你可以使用批量更新功能，而不是打开每个单独的监控项进行编辑。

使用批量更新

要批量更新某些监控项，请执行以下操作：

- 在列表中标记要更新的监控项的复选框
- 在列表下方，点击 *Mass update*
- 标记要更新的属性的复选框
- 输入新的属性值，然后单击*Update*

Security level Original

Authentication protocol Original

Authentication passphrase Original

Privacy protocol Original

Privacy passphrase Original

Port Original

Type of information Original

Data type Original

Units Original

Authentication method Original

User name Original

Public key file Original

Private key file Original

Password Original

Custom multiplier (0 - Disabled) Original

Update interval (in sec)

Flexible intervals Original

History storage period (in days)

Trend storage period (in days) Original

Status Original

Log time format Original

Store value Original

Show value Original

Allowed hosts Original

Replace applications Original

Add new or existing applications Original

Description Original

Replace applications 将从任何现有应用程序中删除该监控项，并将其替换为该字段中指定的监控项。

Add new or existing applications 允许从现有的应用程序中指定其它应用程序，或者为监控项输入全新的应用程序。

这两个字段都是自动完成的 - 开始输入提供了匹配应用程序的下拉列表。如果应用程序是新的，它也会出现在下拉列表中，并在该字符串后面由(*new*)表示。只需向下滚动即可选择。

8 值映射

Overview

For a more “human” representation of received values, you can use value maps that contain the mapping between numeric values and string representations.

Value mappings can be used in both the Zabbix frontend and notifications sent by email/SMS/jabber etc.

For example, an item which has value '0' or '1' can use value mapping to represent the values in a human-readable form:

- '0' → 'Not Available'
- '1' → 'Available'

Or, a backup related value map could be:

- 'F' → 'Full'
- 'D' → 'Differential'
- 'I' → 'Incremental'

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

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

Value mappings, starting with Zabbix 3.0, can be exported/imported, either separately, or with the respective template or host.

Configuration

To define a value map:

- Go to: *Administration* → *General*
- Select *Value mapping* from the dropdown
- Click on *Create value map* (or on the name of an existing map)



Parameters of a value map:

Parameter	Description
-----------	-------------

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

To add a new individual mapping, click on *Add*.

How this works

For example, one of the predefined agent items 'Ping to the server (TCP)' uses an existing value map called 'Service state' to display its values.



In the item [configuration form](#) you can see a reference to this value map in the *Show value* field:



So in *Monitoring → Latest data* the mapping is put to use to display 'Up' (with the raw value in parentheses).



In the *Latest data* section displayed values are shortened to 20 symbols. If value mapping is used, this shortening is not applied to the mapped value, but only to the raw value separately (displayed in parenthesis).

A value being displayed in a human-readable form is also easier to understand when receiving notifications.

Without a predefined value map you would only get this:



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

9 应用

概述

应用程序用于对逻辑组中的监控项进行分组。

例如，MySQL服务器应用程序可以包含与MySQL服务器相关的所有监控项：MySQL的可用性、磁盘空间、处理器负载、每秒事务数、慢查询数等。

应用程序也用于分组Web场景。

如果你正在使用应用程序，那么在*Monitoring → Latest data*数据中，你将看到在各自应用程序下分组的监控项和Web场景。

配置

要使用应用程序，你必须先创建它们，然后将监控项或Web场景链接到它们。

要创建应用程序，请执行以下操作：

- 进入 *Configuration → Hosts* 或者 *Templates*
- 点击 *Applications* 旁边的所需的主机或模板
- 点击 *Create application*
- 输入应用名，然后点击 *Add* 保存



你也可以直接在监控项属性表单中创建一个新的应用程序。

监控项与属性表单中的应用程序相关联。选择该监控项将属于的一个或多个应用程序。

Web场景在Web场景定义表单中链接到应用程序。选择场景所属的应用程序。

10 队列

概述

队列显示正在等待刷新的监控项。队列只是一个逻辑表达的数据。 Zabbix中没有IPC队列或任何其它队列机制。

由代理监控的监控项也包括在队列中 - 它们将被计入排队等待代理历史数据的更新周期。

只有具有预定刷新次数的监控项才会显示在队列中。这意味着以下监控项类型从队列中排除：

- 日志、logrt和事件日志激活的Zabbix代理监控项
- SNMP trap 监控项
- trapper 监控项
- web monitoring 监控项

队列显示的统计信息是Zabbix服务器性能是否健康的指标。

使用JSON协议直接从Zabbix服务器检索队列。 该信息仅在Zabbix服务器运行时可用。

读取队列

要读取队列，请转到*Administration → Queue*。在右侧的下拉列表中选择*Overview*。



这里的图片通常是“绿色”，所以我们可以假设服务器运行的很好。

队列显示一个监控项等待5秒钟，还有5个监控项等待30秒。知道这些意味着什么是很棒的。

要做到这一点，请在右上角的下拉列表中选择*Details*。 现在，你可以看到这些延迟监控项的列表了。



有了这些细节，你有可能找出为什么这些监控项被延迟了。

With one or two delayed items there perhaps is no cause for alarm. They might get updated in a second. However, if you see a bunch of items getting delayed for too long, there might be a more serious problem. 有一个或两个延迟的监控项，也许没有任何可担心的原因。它们可能会在一秒钟内得到更新。 但是，如果你看到一些监控项延迟太久，可能会出现更严重的问题。



是不是监控代理宕了？

队列项

可以使用特殊的内部监控项zabbix [queue, <from>, <to>]来监视Zabbix中队列的运行的状况。它将返回延迟设定的时间量的监控项数量。有关更多信息，请参阅[内部监控项](#)。

11 值缓存

概述

为了计算触发表达式，以及让计算/聚合监控项和一些宏更快，从Zabbix 2.2开始Zabbix服务器支持值的缓存选项。

该内存中缓存可用于访问的历史数据，而不是直接对数据库进行SQL调用。如果缓存中不存在历史值，则从数据库请求缺少的值，并相应地更新缓存。

要启用值缓存功能，Zabbix服务器[配置文件](#)支持可选的**ValueCacheSize**参数。

有两个内部的监控项来监控值缓存：zabbix [vcache, buffer, <mode>]和zabbix [vcache, cache, <parameter>]。查看更多[有内部监控项](#)的细节。

3 触发器

概述

触发器是“评估”由项目采集的数据并表示当前系统状况的逻辑表达式。

当监控项用于采集系统的数据时，始终遵循这些数据是非常不切合实际的，因为这些数据始终在等待一个令人担忧或者值得关注的状态。然而这个“评估”数据的工作可以留给触发器表达式。

触发器表达式允许定义一个什么状况的数据是“可接受”的阈值。因此，如果接收的数据超过了可接受的状态，则触发器会被触发 - 或将状态更改为PROBLEM.

一个触发器可以拥有下面几种状态：

值	描述
OK	这是一个正常的触发器状态，在旧版本的Zabbix中称为FALSE。
PROBLEM	通常意味着触发了某些事情。例如，处理器的负载较高。在旧版本的Zabbix中称为TRUE。

每当Zabbix server接收到作为表达式一部分的新值时，都会重新计算触发器状态（表达式）。

如果在表达式中使用基于时间的函数(`nodata()`, `date()`, `dayofmonth()`, `dayofweek()`, `time()`, `now()`)，触发器就会由Zabbix `timer`进程每30秒重新计算一次。如果在表达式中同时使用基于时间和非基于时间的函数，当接收到一个新值和每隔30秒都会重新计算触发器的状态。

你可以[构建不同复杂程度的触发器表达式](#)

1 配置一个触发器

概述

配置一个触发器，请执行以下操作：

- 点击Zabbix上方菜单栏的*Configuration → Hosts*
- 在Host那一行点击*Triggers*
- 在右上角点击*Create Trigger*（或者在触发器名称上编辑一个现有的触发器）
- 在打开的页面输入触发器的参数

配置

Trigger标签页包含了所有必需的触发器属性。



参数	描述
<i>Name</i>	触发器名称。这个名称可能包含支持的macros: <code>{HOST.HOST}</code> , <code>{HOST.NAME}</code> , <code>{HOST.CONN}</code> , <code>{HOST.DNS}</code> , <code>{HOST.IP}</code> , <code>{ITEM.VALUE}</code> , <code>{ITEM.LASTVALUE}</code> 和 <code>{\$MACRO}</code> 。 <code>\$1, \$2...\$9</code> 宏可以用来指代第一、第二至第九个表达式的常量。备注：\$1-\$9如果引用了相对简单的常量或易懂的表达式，宏将会正确解析。例如，如果表达式为>New host:system.cpu.load[percpu,avg1].last()>5，则名为“Processor load above \$1 on {HOST.NAME}”的触发器名称将自动更改为“Processor load above 5 on New host”
<i>Severity</i>	通过点击对应的按钮来设置所需的触发器severity
<i>Problem expression</i>	用于定义问题条件的逻辑expression。
<i>OK event generation</i>	OK event generation选项: Expression - OK events are generated based on the same expression as problem events; Recovery expression - OK events are generated if the recovery expression evaluates to TRUE and the problem expression evaluates to FALSE; None - in this case the trigger will never return to an OK state on its own.Supported since Zabbix 3.2.0.
<i>Recovery expression</i>	Logical expression used to define the conditions when the problem is resolved.This field is optional and only available if 'Recovery expression' is selected for <i>OK event generation</i> .Supported since Zabbix 3.2.0.
<i>Tag for matching</i>	Enter event tag name to use for event correlation.This field is displayed if 'All problems if tag values match' is selected for the <i>OK event closes</i> property and is mandatory in this case.Supported since Zabbix 3.2.0.
<i>Tags</i>	Set custom tags to mark trigger events. Event tags can be used for event correlation, in action conditions and will also be seen in <i>Monitoring → Problems</i> .Tags are a pair of tag name and value. You can use only the name or pair it with a value.User macros, user macro context, low-level discovery macros and macro functions: <code> {{\$ITEM.VALUE}.regsub(pattern, output)}, {{\$ITEM.VALUE}.iregsub(pattern, output)}</code> are supported in event tags. Low-level discovery macros can be used

	inside macro context. If the total length of expanded value exceeds 255, it will be cut to 255 characters. Supported since Zabbix 3.2.0.
Allow manual close	Check to allow manual closing of problem events generated by this trigger. Manual closing is possible when acknowledging problem events. This field is available if event acknowledgement is activated in Administration → General . Supported since Zabbix 3.2.0.
URL	如果此处不为空，则在此处输入的URL可在点击 Monitoring → Triggers 的trigger name的时候链接于此。可以在触发器URL字段 - {TRIGGER.ID}，多个{HOST.*}宏（Zabbix3.0.0以后的版本）和用户宏（Zabbix3.0.0以后的版本）中使用宏。
Description	文本字段用来提供更多关于触发器的信息。可能包含修复特定问题的说明、相关负责人的联系方式等。从Zabbix2.2开始，描述可能包含与触发器名称相同的一组宏。
Enabled	如果需要，取消选中此框将禁用触发器。

Dependencies标签页包含触发器的所有[dependencies](#)。

点击[Add](#)来添加一个新的依赖关系。

你也可以打开一个现有的触发器，点击[Clone](#)按钮，以一个不同的名称保存为新的触发器。

2 触发器表达式

概述

触发器中使用的表达式是非常灵活的。你可以使用他们去创建关于监控统计的复杂逻辑测试。

一个简单有效的表达式看起来像：

```
1. {<server>:<key>.<function>(<parameter>)}<operator><constant>
```

1 函数

触发器函数允许引用收集的值，当前时间和其他因素。

可以使用的[被支持函数](#)的完整列表。

2 函数参数

大多数数字型的函数接受秒数来作为参数。

你可以使用前缀#来指定参数具有不同的含义：

函数内容	含义	
<code>sum(600)</code>	600秒内所有值的总和	
<code>sum(#5)</code>	最后5个值的总和	

函数last当以#作为前缀使用时，值具有不同的含义，它会让她会选择第N个的上一个值，所以给定值3、7、2、6、5（按照时间顺序，第一个值3为最新值），`last(#2)` 将返回值为7，`last(#5)` 将返回值为5。

`avg, count, last, min and max` 函数支持额外的第二个参数 `time_shift`（时间偏移量）。这个参数允许从过去一段时间内引用数据。例如，`avg(1h,1d)`将会返回一天前1小时的平均值。

触发器需要使用history历史数据来计算。如果历史数据不可用（特别是关于`time_shift`时间偏移量），则无法使用趋势信息，因此必须至少保持触发器函数所预期这段时间的历史信息。

你可以在触发器表达式中使用支持的[单位符号](#)，例如“5m”（分钟）可以被“300”秒代替，“1d”（天）可以被“86400”秒代替。“1k”代表“1024”bytes。

3 运算符

触发器支持的运算符（在执行中优先级递减）

优先级	运算符	定义	未知值的注释
1	-	Unary minus	-Unknown → Unknown
2	not	逻辑非 (NOT)	not Unknown → Unknown

3	*	乘	$0 * \text{Unknown} \rightarrow \text{Unknown}(\text{yes}, \text{Unknown}, \text{not } 0 - \text{to not loseUnknown in arithmetic operations})$ $1.2 * \text{Unknown} \rightarrow \text{Unknown}$
	/	相除	$\text{Unknown} / 0 \rightarrow \text{errorUnknown}$ $1.2 / \text{Unknown} \rightarrow \text{Unknown}$
4	+	相加	$1.2 + \text{Unknown} \rightarrow \text{Unknown}$
	-	相减	$1.2 - \text{Unknown} \rightarrow \text{Unknown}$
5	<	小于. 该运算符定义: $A < B \Leftrightarrow (A \leq B - 0.000001)$	$1.2 < \text{Unknown} \rightarrow \text{Unknown}$
	\leq	小于等于.	$\text{Unknown} \leq \text{Unknown} \rightarrow \text{Unknown}$
	>	大于. 该运算符定义: $A > B \Leftrightarrow (A \geq B + 0.000001)$	
	\geq	大于等于.	
6	=	相等. 该运算符定义: $A = B \Leftrightarrow (A > B - 0.000001) \text{ and } (A < B + 0.000001)$	
	\neq	不等于. 该运算符定义: $A \neq B \Leftrightarrow (A \leq B - 0.000001) \text{ or } (A \geq B + 0.000001)$	
7	and	逻辑与 (AND)	$0 \text{ and } \text{Unknown} \rightarrow 0$ $1 \text{ and } \text{Unknown} \rightarrow \text{Unknown}$ $\text{Unknown} \text{ and } \text{Unknown} \rightarrow \text{Unknown}$
8	or	逻辑或 (OR)	$1 \text{ or } \text{Unknown} \rightarrow 1$ $0 \text{ or } \text{Unknown} \rightarrow \text{Unknown}$ $\text{Unknown} \text{ or } \text{Unknown} \rightarrow \text{Unknown}$

not, **and** and **or** 运算符区分大小写, 而且必须为小写。它们也必须被空格或括号包围。

所有运算符中, 除了unary - and **not**, 都有从左到右的关联性。Unary - and **not** All operators, except unary - and **not**, have left-to-right associativity. Unary - and **not** are non-associative (meaning $-(-1)$ and **not** (**not** 1) should be used instead of -1 and **not not** 1).

计算结果:

- $<$, \leq , $>$, \geq , $=$, \neq 如果指定的关系为真 (true), 则运算符将会在触发器表达式中产生“1”; 如果指定的关系为假 (false), 则返回“0”。如果一个运算对象为“Unknown”, 那么结果为Unknown;
- **and** 对于已知的运算对象, 如果两个运算对象的比较不等于“0”, 则运算符将会在触发器表达式中产生“1”, 否则, 它产生“0”; 对于未知的运算对象, 如果两个运算对象的比较等于“0”, 则会产生“0”, 否则, 则会产生“Unknown”;
- **or** 对于已知的运算对象, 如果其中任意一个运算对象的比较不等于“0”, 则运算符会在触发器表达式中产生“1”, 否则, 它产生“0”; 对于未知的运算对象进行“or”运算, 则只有当一个运算对象的比较不等于“0”, 才会产生“1”, 否则, 它会产生“Unknown”;
- 如果运算单位的值不等于“0”, 那么逻辑否定运算符**not**对于已知运算对象的结果为“0”;
- The result of the logical negation operator **not** for a known operand is '0' if the value of its operand compares unequal to '0'; '1' if the value of its operand compares equal to '0'. For unknown operand **not** yields 'Unknown'.

4 触发器示例

示例 1

触发器名称: Processor load is too high on www.zabbix.com。触发器表达式如下:

```
1. {www.zabbix.com:system.cpu.load[all,avg1].last()}>5
```

“www.zabbix.com:system.cpu.load[all,avg1]” 给出了被监控对象参数的简短名称。它指定了服务器是“www.zabbix.com”，监控项的键值是“system.cpu.load[all,avg1]”。通过使用函数“last()”获取最近一次获取的值。最后，“>5”表示来自主机www.zabbix.com的最后一次获取的负载值大于5时触发器就会进入PROBLEM状态。

示例 2

触发器名称: www.zabbix.com is overloaded。触发器表达式如下:

```
1. {www.zabbix.com:system.cpu.load[all,avg1].last()}>5 or {www.zabbix.com:system.cpu.load[all,avg1].min(10m)}>2
```

当负载大于5或者最近10分钟内负载大于2，表达式为“TURE”，就会使触发器进入PROBLEM状态。

示例 3

触发器名称: /etc/passwd has been changed。触发器表达式如下:

使用了函数“diff”:

```
1. {www.zabbix.com:vfs.file_cksum[/etc/passwd].diff()}=1
```

当文件/etc/passwd检查的checksum值与最近的值不同时，表达式为“TURE”，就会使触发器进入PROBLEM状态。同样的表达式还可以用于监控重要的文件，比如文件/etc/passwd、/etc/inetd.conf、/kernel等等。

示例 4

触发器名称: Someone is downloading a large file from the Internet。触发器表达式如下:

使用函数“min”:

```
1. {www.zabbix.com:net.if.in[eth0,bytes].min(5m)}>100K
```

当网络适配器“eth0”在5分钟内接收的字节大于100KB，表达式为“TURE”，就会使触发器进入PROBLEM状态。

示例 5

触发器名称: Both nodes of clustered SMTP server are down。触发器表达式如下:

注意: 在同一个表达式中使用了两个不同的主机

```
1. {smtp1.zabbix.com:net.tcp.service[smtp].last()}=0 and {smtp2.zabbix.com:net.tcp.service[smtp].last()}=0
```

当SMTP服务器“smtp1.zabbix.com”和“smtp2.zabbix.com”都停止，表达式为“TURE”，就会使触发器进入PROBLEM状态。

示例 6

触发器名称：Zabbix agent needs to be upgraded。触发器表达式如下：

使用函数“str()”：

```
1. {zabbix.zabbix.com:agent.version.str("beta8")}=1
```

如果Zabbix agent有beta8版本（大概为1.0beta8），表达式为“TURE”，就会使触发器进入PROBLEM状态。

示例 7

触发器名称：Server is unreachable。触发器表达式如下：

```
1. {zabbix.zabbix.com:icmpping.count(30m,0)}>5
```

当主机“zabbix.zabbix.com”在30分钟内超过5次不可达，表达式为“TURE”，就会使触发器进入PROBLEM状态。

示例 8

触发器名称：No heartbeats within last 3 minutes。触发器表达式如下：

使用函数“nodata()”：

```
1. {zabbix.zabbix.com:tick.nodata(3m)}=1
```

'tick'必须为'Zabbix trapper'类型。为了使这个触发器工作，监控项'tick'必须要定义，这个主机应使用zabbix_sender定期发送此参数的数据，如果在180秒内还未收到zabbix_sender发送的数据，那么触发器的状态就会变成PROBLEM。

示例9

CPU activity at night time触发器的名称为：CPU activity at night time

使用了函数time()：

```
1. {zabbix:system.cpu.load[all,avg1].min(5m)}>2 and {zabbix:system.cpu.load[all,avg1].time()}>000000 and {zabbix:system.cpu.load[all,avg1].time()}<060000
```

只有在凌晨0点到6点整，最后5分钟内cpu load大于2，触发器的状态才会变更为PROBLEM。

示例 10

触发器名称：Check if client local time is in sync with Zabbix server time使用了函数fuzzytime()：

```
1. {MySQL_DB:system.localtime.fuzzytime(10)}=0
```

当MySQL_DB的本地时间与Zabbix server之间的时间相差10秒以上，就会使触发器的状态变更为PROBLEM。

示例 11

触发器名称为：Comparing average load today with average load of the same time yesterday (using a second `time_shift` parameter).

```
1. {server:system.cpu.load.avg(1h)}/{server:system.cpu.load.avg(1h,1d)}>2
```

This expression will fire if the average load of the last hour tops the average load of the same hour yesterday more than two times.如果最后一小时的平均cpu load超过前一天的同一小时两倍，就会使触发器的状态变更为PROBLEM。

示例 12

使用了另一个监控项来获得触发器的阈值：

```
1. {Template PfSense:hrStorageFree[#SNMPVALUE].last()<{Template PfSense:hrStorageSize[#SNMPVALUE].last()*0.1}
```

如果hrStorageFree低于10%，就会使触发器的状态变更为PROBLEM。

示例 13

使用 `evaluation result` 来获取触发器的数量超过阈值。

```
1. ({server1:system.cpu.load[all,avg1].last()>5} + ({server2:system.cpu.load[all,avg1].last()>5} + ({server3:system.cpu.load[all,avg1].last()>5})>=2
```

如果表达式中的两个触发器表达式的结果大于5，就会使触发器的状态变更为PROBLEM。

5 滞后 (Hysteresis)

Sometimes we need an interval between an OK and Problem states, rather than a simple threshold. For example, we would like to define a trigger which becomes Problem when server room temperature goes above 20C and we want it to stay in that state until the temperature drops below 15C.有时候我们需要一个触发器状态OK和PROBLEM之间的区间，而不是简单的阈值。例如，我们想定义一个触发器，当机房的室温超过20摄氏度时，我们希望它保持这个状态，直至温度低于15摄氏度，触发器的状态才会变更为OK。

In order to do this, we first define the trigger expression for the problem event. Then select 'Recovery expression' for *OK event generation* and enter another expression for the OK event.为了做到这一点，我们首先定义一个PROBLEM事件的触发器表达式，然后为OK事件选择'Recovery expression'，并为OK事件输入不同的表达式。

示例 1

触发器名称：Temperature in server room is too high.

Problem expression:

```
1. {server:temp.last()}>20
```

Recovery expression:

```
1. {server:temp.last()}<=15
```

示例 2

触发器名称: Free disk space is too low.

Problem expression: 在最近5分钟内文件系统/的空闲空间小于10GB。

```
1. {server:vfs.fs.size[/, free].max(5m)}<10G
```

Recovery expression: 在最近10分钟内文件系统/的空闲空间大于40GB。

```
1. {server:vfs.fs.size[/, free].min(10m)}>40G
```

6 具有不受支持的监控项和未知值的表达式

Versions before Zabbix 3.2 are very strict about unsupported items in a trigger expression. Any unsupported item in the expression immediately renders trigger value to `Unknown`. Zabbix 3.2之前的版本对关于触发器表达式中的不受支持的监控项非常严格。表达式中任何不受支持的监控项都将触发器值立即显示为 `Unknown`。

Since Zabbix 3.2 there is a more flexible approach to unsupported items by admitting unknown values into expression evaluation:从Zabbix 3.2开始对于不受支持的监控项，通过将它们引入到表达式评估中，产生了一种更加灵活的表达方式：

- For some functions their values are not affected by whether an item is supported or unsupported. Such functions are now evaluated even if they refer to unsupported items. See the list in [functions and unsupported items](#).
- 对于一些函数，它们的值不受监控项是否支持与不支持的影响，即使她们引用了不支持的监控项，也会对这些函数进行评估。关于这些函数，请参阅[functions and unsupported items](#)。
- Logical expressions with OR and AND can be evaluated to known values in two cases regardless of unknown operands:
- 具有OR和AND的逻辑表达式可以在两种情况下评估为已知值，不用管未知的运算对象：

- “1 `or` Unsuported_item1.some_function() `or` Unsuported_item2.some_function() `or` ...” can be evaluated to '1' (True),
- “1 `or` Unsuported_item1.some_function() `or` Unsuported_item2.some_function() `or` ...”可以被评估为'1' (True),
- “0 `and` Unsuported_item1.some_function() `and`

`Unsupported_item2.some_function() and ...` can be evaluated to '0' (False). Zabbix tries to evaluate logical expressions taking unsupported items as `Unknown` values. In the two cases mentioned above a known value will be produced; in other cases trigger value will be `Unknown`.

- “`0 and Unsupported_item1.some_function() and Unsupported_item2.some_function()`”可以被评估为‘0’ (False). \Zabbix尝试评估将将不受支持的监控项作为 `Unknown` 值的逻辑表达式。在上述两种情况下，将产生一个Known值；在其他情况下，触发器将产生 `Unknown` 值。
- If a function evaluation for supported item results in error, the function value is `Unknown` and it takes part in further expression evaluation.
- 如果对受支持的监控项的一个函数评估结果为错误，那么这个函数的值为 `Unknown`，并且它将参与进一步的表达式评估。

Note that unknown values may “disappear” only in logical expressions as described above. In arithmetic expressions unknown values always lead to result `Unknown` (except division by 0). 备注：未知值只会在上述逻辑表达式中“消失”。在算数表达式中未知值总会导致结果为 `Unknown`（除以0除外）。

If a trigger expression with several unsupported items evaluates to `Unknown` the error message in the frontend refers to the last unsupported item evaluated. 如果具有多个不受支持的监控项的触发器表达式评估为 `Unknown`，那么在前端中的错误消息值的是最后一个不受支持的项目。

3 触发器依赖

概述

有时候，一台主机的可用性取决于另一台主机。例如，如果一台路由设备宕机，则路由设备后端的服务器全部会变得无法访问。如果这两者都设置了触发器，你可能会收到关于这两者宕机的报警，而事实上只有路由设备是真正存在故障的。

这就是主机之间某些依赖关系可能有用的地方，依赖关系设置的通知可能会被抑制，而只发送根本问题的通知。

虽然Zabbix不支持主机之间的直接依赖关系，但是它们可以定义另外一种更加灵活的方式—触发器依赖关系。一个触发器可能有多个依赖于它的触发器。

所以在上面提到的例子中，我们打开服务器的触发器配置的窗口，并设置依赖于路由设备的相应触发器。有了这样的依赖关系，只要它所依赖的触发器处于“PROBLEM”状态，服务器触发器就不会改变状态，因此不会执行依赖的动作，同时也不会发送通知。

如果路由设备和服务器同时宕机，如果有依赖关系，那么Zabbix将不会执行服务器的触发动作。

如果当前触发器所依赖的触发器的状态从“PROBLEM”变更为“UNKNOWN”，那么则不会执行这个依赖触发器的动作。

值得注意的是，如果触发器所依赖的触发器被禁用、监控项，甚至主机被禁用，则依赖触发器的事件/动作将不会被抑制。

同样的

- 触发器依赖可以从任何主机触发器添加到另一个主机触发器，只要它不会导致循环的依赖即可。
- 触发器依赖可以从模板添加到另一个模板，如果模板A的触发器依赖于模板B的触发器，一个主机要链接到模板A的话，那么它同时要链接到模板B（因为模板A的触发器依赖于模板B的触发器），但是主机却可以单独链接到模板B，因为模板B是被依赖的。
- 触发器依赖可以从模板的触发器添加到主机触发器。在这种情况下，如果主机链接这样的模板，那么主机也会创建链接到的模板所依赖的主机触发器。例如，如果模板的某些触发器依赖路由器（或主机）的触发器，那么但凡链接了这个模板的，都会依赖于路由器（或主机）的触发器。
- 可能不会添加从主机触发器到模板触发器的触发器依赖。
- 触发器依赖可以从一个触发器原型添加到另一个触发器原型（在同一个Low-level discovery规则中）或一个真实的触发器。触发器原型可能不依赖于从不同LLD规则的触发器原型或从触发器原型中创建的一个触发器。主机触发器原型不会依赖于来自模板的触发器。

配置

要定义触发器依赖，打开触发器[configuration form](#)的Dependencies选项卡，在“Dependencies”选项卡点击Add进行添加，并选择所依赖的一个或多个触发器。



点击*Update*更新，就可以看到所依赖的触发器列表。



几个依赖关系的示例

例如， 主机的前面有Router1， Router2的前面有Router1。

```
1. Zabbix - Router1 - Router2 - Host
```

如果Router1宕掉，那么很明细那Host和Router2就不可达了，但我们并不想收到关于Host、Router1和Router2宕掉这三个通知。

于是我们定义了如下两个依赖关系：

```
1. 'Host is down' trigger depends on 'Router2 is down' trigger  
2. 'Router2 is down' trigger depends on 'Router1 is down' trigger
```

在改变“Host is down”触发器的状态之前，Zabbix将会检查相应触发器的依赖关系，如果找到，并且一个触发器处于“Problem”状态，则触发器状态将不会发生改变，因此不会执行actions动作，也不会发出相应的通知。

Zabbix递归执行此检查，如果Router1或Router2是不可达的状态，那么Host触发器则不会更新。

4 触发器严重性

触发器严重性定义了触发器的重要程度，Zabbix支持以下触发器的严重程度：

SEVERITY	DEFINITION	COLOUR
Not classified	Unknown severity.	Grey
Information	For information purposes.	Light blue
Warning	Be warned.	Yellow
Average	Average problem.	Orange
High	Something important has happened.	Light red
Disaster	Disaster. Financial losses, etc.	Red

不同严重性的用途为：

- 触发器的可视化表现，不同的颜色代表不同的严重程度。
- 全局报警音频。不同的音频代表不同的严重程度。
- 用户媒介，不同的用户媒介（通知渠道）代表不同的严重程度。例如，SMS表示高严重性，email表示其他严重性。
- 根据触发严重程度的情况来限定不同的actions动作。

可以[customise trigger severity names and colours](#).

5 自定义触发器的严重性

可以在Administration → General → Trigger severities选项中配置于严重性相关GUI元素的触发器严重性的名称和颜色。配置的颜色可以在所有的GUI主题中共享。==== 翻译自定义触发器严重性的名称 ====<note important>如果使用Zabbix的前端Web页面进行自定义触发器严重性的名称，默认情况下，翻译的自定义触发器严重性的名称将覆盖Zabbix原有的翻译名称。</note>默认的触发器严重性名称可适用于所有语言环境的都有翻译。如果改变了触发器严重性的名称，则会在所有的语言环境中都使用此名称，因此在其他的语言环境中需要进行额外的手动翻译。自定义严重性名称的翻译步骤：设置所需的自定义严重性名称，例如“Important”编辑<frontend_dir>/locale/<required_locale>/LC_MESSAGES/frontend.po 添加如下两行：<code>msgid "Important" msgstr "<translation string>"</code> and save file. 创建.mo文件为<frontend_dir>/locale/README的描述这里的msgid应该对应新的自定义严重性名称，并且msgstr应该是特定语言的翻译。例如，如果是zh_CN环境，那么应该修改zh_CN下的frontengd.po文件。应该在修改每个严重性名称后执行此过程。

6 单位符号

概述

在Zabbix，不得不使用一些较大的数字，例如“86400”秒代表一天，这些数字既然不便又容易出错。所以可以使用一些适当的单位符号（或后缀）来简化Zabbix触发器和监控项的键值。

上例中，你可以输入“1d”来代替“86400”即可。后缀为乘法函数。

触发器表达式

触发器expression常量和函数参数支持时间和内存大小的后缀

你可以使用如下时间单位后缀：

- **s** - 秒（通常情况下，不带任何时间单位后缀就表示s）
- **m** - 分钟
- **h** - 小时
- **d** - 天
- **w** - 周

时间单位后缀也受zabbix[queue,<from>,<to>]的internal item的参数和aggregate checks最后一个参数的支持。

对于内存大小你可以使用如下单位后缀：

- **K** - kilobyte
- **M** - megabyte
- **G** - gigabyte
- **T** - terabyte

可以使用的其他单位后缀

单位符号也用在前端数据的可读表述。

在Zabbix Server和前端都支持这些符号：

- **K** - kilo
- **M** - mega
- **G** - giga
- **T** - tera

当前端展示的item值为B、Bps时，那么使用base 2 (1K=1024)。反之使用base 10 (1K=1000)另外前端也支持如下符号展示：

- **P** - peta
- **E** - exa
- **Z** - zetta
- **Y** - yotta

使用案例

通过使用一些适当的后缀，你可以编写易懂和易维护的触发器表达式，例如下面这些表达式：

```
1. {host:zabbix[proxy,zabbix_proxy,lastaccess]}>120
2. {host:system.uptime[] .last()}<86400
3. {host:system.cpu.load.avg(600)}<10
4. {host:vm.memory.size[available] .last()}<20971520
```

可以被修改为：

```
1. {host:zabbix[proxy,zabbix_proxy,lastaccess]}>2m
2. {host:system.uptime.last()}<1d
3. {host:system.cpu.load.avg(10m)}<10
4. {host:vm.memory.size[available] .last()}<20M
```

7 批量更新

概述

使用批量更新，可以一次更改一些触发器的某些属性，从而节省了打开每个触发器进行单独编辑的需要。

使用批量更新

进行批量更新某些触发器，请执行以下操作：

- 在触发器列表中选中要更新触发器的复选框；
- 点击下面批量更新*Mass update*的按钮；
- 标记要更新的属性的复选框；
- 标记要更新属性的新值，点击下面的更新*Update*按钮。

The screenshot shows the 'Mass update' dialog box. At the top, there is a 'Severity' dropdown with options: Not classified, Information, Warning, Average (which is selected and highlighted in orange), High, and Disaster. Below this are two sections: 'Replace dependencies' and 'Replace tags'. The 'Replace dependencies' section contains a 'Name' field with the value 'Zabbix server 1: Disk I/O is overloaded on {HOST.NAME}' and an 'Add' button. The 'Replace tags' section contains 'tag' and 'value' input fields, a 'Remove' button, and an 'Add' button. At the bottom, there is a 'Allow manual close' checkbox with options 'No' (selected) and 'Yes', and 'Update' and 'Cancel' buttons.

Replace dependencies and *Replace tags*将使用批量更新中指定的触发器依赖关系/标签（如果有）来替换已存在的触发器依赖关系/标签。

8 预测触发功能

概述

有时候有即将到来的问题的迹象。可以发现这些迹象，以便提前采取行动，以防止或至少最小化问题的影响。

Zabbix具有基于历史数据预测受监视系统的未来行为的工具。这些工具通过预测触发功能实现。

1 功能

需要知道的两件事是如何定义问题状态以及需要多少时间来采取行动。有两种方法可以设置一个关于潜在的不必要的情况的触发信号。第一：触发器必须在系统发生“时间作用”之后才会发生故障状态。第二：当系统在不到“时间行为”的时候达到问题状态时，触发器必须触发。 使用相应的触发器功能是**forecast**和 **timeleft**。请注意，两个功能的基本统计分析基本相同。 您可以设置触发器，以您喜欢的方式，以类似的结果。

2 参数

这两个功能使用几乎相同的参数集。列表请参见 [支持的功能](#)。

2.1 时间间隔

首先你应该指明Zabbix在预测的时候应该分析的历史时间段。你可以通过“秒”或“#num”参数和可选的 `time_shift` 以熟悉的方式进行操作，就像使用 `avg`，`count`，`delta`，`max`，`min` 和 `sum` 功能。

2.2 预测范围

(**forecast only**)\参数 `time` 指定了将来Zabbix应该在多大程度上推断其在历史数据中找到的依赖关系。 无论是否使用“`time_shift`”，“时间”始终从当前时刻算起

2.3 阈值

`(timeleft only)\参数“阈值”指定分析项目必须达到的值，如果从上方或从下方没有差异。一旦我们确定了f(t)（见下文），我们应该求解方程f(t) = threshold，如果没有这样的根，返回更接近现在和从右到左的根或99999999999.9999。`

`<note tip>当项目值接近门槛然后交叉时，timeleft 假设交叉点已经过去，因此切换到具有“阈值”级别的下一个交叉值，如果有的话。最佳实践应该是使用预测作为普通问题诊断的补充，而不是替代。（例如，一个简单的触发器 <code>{host:item.timeleft(1h,,X)} < 1h</code> 当项目值接近X时可能进入问题状态，然后一旦达到值X就突然恢复。如果问题是项目值低于X，请使用：<code>{host:item.last()} < X or {host:item.timeleft(1h,,X)} < 1h</code>如果问题是项目值高于X，请使用：<code>{host:item.last()} > X or {host:item.timeleft(1h,,X)} < 1h</code></note>`

2.4 适合功能

默认的 fit 是线性 功能。但是如果您的监控系统更复杂，您可以选择更多的选择。

```
^ fit ^ x = f(t) ^
|linear |x = a + bt |
|polynomial1) |x = a0 + a1t + a2t2 + ... + antn |
|exponential |x = aexp(bt) |
|logarithmic |x = a + blog(t) |
|power |x = atb |
```

2.5 模式

(**forecast only**)

每次触发功能被评估时，它都会从指定的历史时段获得数据，并将其指定给数据。因此，如果数据略有不同，拟合函数将略有不同。如果我们在将来在指定时间内简单地计算拟合函数的值，那么您将不会在现在和将来的那个时刻之间预测分析项目的行为。对于某些“fit”选项（如多项式），未来的简单值可能会产生误导。

```
^ 模式 ^ 预测 结果 ^
|value |f(now + time) |
|max |maxnow <= t <= now + time f(t) |
|min |minnow <= t <= now + time f(t) |
|delta |max - min |
|avg |average of f(t) (now <= t <= now + time') according to definition |
```

3 细节

为了避免大量的计算，我们将指定时间段内的第一个值的时间戳值加上1 ns作为新的零时间（当前时期为10阶，时间平方为 10^{18} ，双精度为约 10^{-16} ）。添加1 ns以提供对数和功率拟合的所有正时间值，其涉及计算 $\log(t)$ 。时间偏移不影响线性，多项式，指数（除了更容易和更精确的计算），但改变对数和功能函数的形状。

4 潜在错误

在这种情况下，函数返回-1：

- 指定评估期不含数据；
- 数学运算的结果没有定义²⁾；
- 数值问题(不幸的是，对于一些输入数据范围和双精度浮点格式的精度变得不足)³⁾。

如果选择合适不好描述提供的数据或只有太少的数据用于准确预测，就不会有警告或错误被标记。

5 示例和处理错误

要在主机上的可用磁盘空间用尽时收到警告，可以使用如下触发器表达式：

```
1. {host:vfs.fs.size[/,free].timeleft(1h,,0)}<1h
```

但是，错误代码-1可能会发挥作用，并将您的触发器置于问题状态。一般来说，这是很好的，因为你发现一个警告，你的预测不能正常工作，你应该更彻底地看看他们，以找出原因。但有时它是坏的，因为-1可以简单地意味着没有关于最后一小时内获得的主机可用磁盘空间的数据。但是，如果您收到太多正值警报，则应考虑使用更复杂的触发器表达式⁴⁾：

```
1. {host:vfs.fs.size[/,free].timeleft(1h,,0)}<1h and {host:vfs.fs.size[/,free].timeleft(1h,,0)}>-1
```

forecast的情况可能会更加负责.首先, -1可能会也可能不会将触发器置于问题状态, 具体取决于您是否具有表达式:

```
1. {host:item.forecast(...)}<...
```

或者

```
1. {host:item.forecast(...)}>...
```

此外, 如果项目值为负值, -1可能是有效的预测。但这种情况实际发生的可能性很小, (参见运算符=如何 运作).添加

```
1. ... or {host:item.forecast(...)}=-1
```

or

```
1. ... and {host:item.forecast(...)}<>-1
```

if you want or don't want to treat -1 as a problem respectively.

参见

- [Predictive trigger functions \(pdf\)](#) on zabbix.org

1)

Polynomial degree can be from 1 to 6, *polynomial1* is equivalent to *linear*. However, use higher degree polynomials [with caution](#). If the evaluation period contains less points than needed to determine polynomial coefficients, polynomial degree will be lowered (e.g *polynomial5* is requested, but there are only 4 points, therefore *polynomial3* will be fitted).

2)

比如将指数 or 功率 函数计入log()item值.如果数据包含零或负数, 您将收到错误 因为log() 仅限于正值。

3)

For *linear*, *exponential*, *logarithmic* and *power* fits all necessary calculations can be written explicitly. For *polynomial* only *value* can be calculated without any additional steps. Calculating *avg* involves computing polynomial antiderivative (analytically). Computing *max*, *min* and *delta* involves computing polynomial derivative (analytically) and finding its roots (numerically). Solving $f(t) = 0$ involves finding polynomial roots (numerically).

4)

但在这种情况下, -1可能会导致您的触发器从问题状态恢复。要完全保护使用:

```
1. {host:vfs.fs.size[/,free].timeleft(1h,,0)}<1h and ({TRIGGER.VALUE}=0 and
{host:vfs.fs.size[/,free].timeleft(1h,,0)}<>-1 or {TRIGGER.VALUE}=1)
```

9 事件标签

概述

在Zabbix中可以自定义事件标签。在触发器级别上定义事件标签。在事件标签定义后，相应的新事件将被标记为事件标签数据。

在拥有自定义事件标签的情况下，可以变得更加灵活。例如，可以基于事件标签定义动作。

事件标签示例

```
1. MySQL, Service:MySQL, Services, Services:Customer, Applications, Application:Java, Priority:High
```

用例

此功能的一些用例如下：

- 识别日志文件中的问题并单独关闭它们
 - 在日志触发器中定义标签，该标签将使用[事件关联](#)]：选择OK事件仅关闭匹配事件并选择匹配的标签的选项；[查看使用标签创建的问题事件，并单独关闭](#)。 - 用它来过滤通知 在触发器级别上定义标签以标记不同标签的事件；在操作条件中使用标签过滤，只接收与标签数据匹配的事件。 - 查看前端的事件标签信息 在触发器级别上定义标签以标记不同标签的事件；在//监控// -> //异常//中查看此问题。 - 从项目值中提取的信息作为标签值 在标签值中使用{{ITEM.VALUE}}.regsub() 宏；在//监控// -> //异常//查看标签值，作为从item值提取的数据。 - 在通知中更好地识别问题 在触发器级别定义标签；在问题通知中使用{EVENT.TAGS}宏；更容易识别通知所属的应用程序/服务。 - 通过使用模板级别的标签来简化配置任务 在模板触发器级别上定义标签；从模板触发器创建的所有触发器上查看这些标签。 - 使用低级别发现的标签创建触发器 (LLD) 在触发器原型上定义标签；在标签名称或值中使用LLD宏； * 从触发器原型创建的所有触发器上查看这些标签== 配置 ==事件标签在触发器配置中定义。 可以为触发器，模板触发器和触发器原型定义事件标签。
 

宏支持

`{ITEM.VALUE}` 和 `{ITEM.LASTVALUE}` 宏可用于填充标签名称或标签值。

标签名称/值支持[用户宏](#) 和用户宏上下文。用户宏上下文可能包括低级发现宏。

低级发现宏可用于触发器原型中的标签名称/值。

`{EVENT.TAGS}` 和 `{EVENT.RECOVERY.TAGS}` 宏可用于基于触发器的通知，并将它们解析为以逗号分隔的事件标签或恢复事件标签列表。

子字符串提取

支持子字符串提取来填充标签名称或标签值，使用新的 [宏功能](#) - 将正则表达式应用于`{ITEM.VALUE}`宏获取的值

```
1. {{ITEM.VALUE}}.regsub(pattern, output)]
```

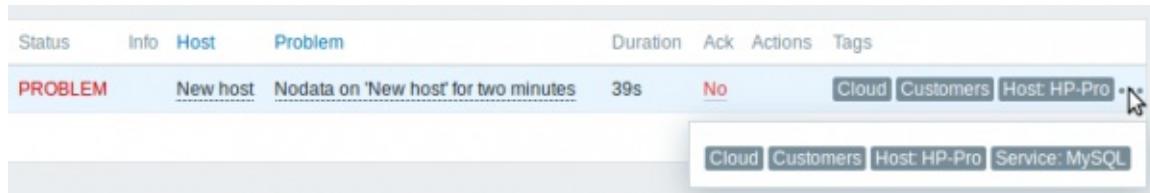
```
2. {{ITEM.VALUE}}.iregsub(pattern, output)}
```

标记名称和值在宏解析后的长度超过255个字符时将被剪切为255个字符。

查看事件标签

事件标签（如果已定义）可以在新的事件中看到：

- Monitoring → Problems
- Monitoring → Problems → Event details



只显示前三个标签条目。如果有三个以上的标签条目，则由三个点表示。如果您将鼠标悬停在这三个点上，则所有标签条目将显示在弹出窗口中。

性能和存储效果

预计事件标签的使用可能具有以下效果：

- 由于在事件标签表中创建新记录，因此事件处理速度将会更慢。
- 与从模板继承的触发器的操作相关的操作将更慢，因为必须在触发器标签表中为每个标签创建一个记录。因此，触发器的创建，更新和删除将会更慢。
- 由于触发器标签，配置缓存的同步速度会更慢。
- 事件标签的存储将需要额外的磁盘空间，这可以与现有触发器和事件表的大小相当。精确的存储要求取决于每个触发器和事件创建的标签数量。

4 事件

概述

在Zabbix可以生成以下几种类型的事件：

- trigger events - 触发器事件，当触发器改变他的状态时 (*OK*→*PROBLEM*→*OK*)；
- discovery events - 发现事件，当主机或服务被检测到；
- auto registration events - 自动注册事件，当主动的agents被自动注册到server时；
- internal events - 内部事件，当监控项item/自动发现规则low-level discovery rule变得不受支持或触发器进入了一个未知状态。

从Zabbix2.2版本开始支持内部事件。

事件是以时间戳的，并可以发送电子邮件等动作得得基础。

要查看前端事件的详细信息，点击*Monitoring* → *Problems*。那里你可以点击事件的日期和事件来查看事件的详细信息。

关于更多的可供参考信息，请查看：

- [trigger events](#)
- [other event sources](#)

1 触发器事件生成

概述

触发器状态的变化是事件最常见和最重要的来源。每次触发器的状态改变时，都会生成一个事件。该事件包含了触发器状态变更的详细信息、发生时间以及信息的状态。

触发器会创建两种类型的事件：问题（Problem）和正常（OK）。

问题事件（Problem event）

在以下情况下，一个问题事件（Problem event）将被创建：

- 当触发器状态为正常（OK）时，触发器表达式的计算结果为TRUE。
- 如果为触发器启用了多重问题事件生成，那么每次触发器表达式计算结果为TRUE。

正常（OK）事件

一个正常事件（OK event）关闭关联的问题事件（Problem event），可由以下三个部分创建：

- 触发器 - 基于“正常事件迭代（OK event generation）”和“正常事件关闭（OK event closes）”的设置；
- 关联项事件；
- 任务管理器 - 当事件被[手动关闭](#)。

触发器

触发器有“事件成功迭代（OK event generation）”的设置，用来控制如何生成正常事件（OK event）：

- 表达式 - 当触表达式的计算结果为FALSE的时候，触发器在问题（Problem）状态中生成一个正常事件（OK event）。这是一个最简单的设置，为默认启动。
- 恢复表达式 - 当表达式的计算结果为FALSE，并且恢复表达式的计算结果为TURE的时候，会为问题（Problem）状态的触发器生成一个正常事件（OK event）。如果触发器的恢复条件和问题标准不同，则可以使用此设置。
- 无 - 正常事件从来不生成。这个可以和多重问题事件生成一起结合使用，以便在某事件发生时可以更简单的发送通知。

此外，触发器有“事件成功关闭（OK event closes）”的设置，用来控制哪些问题事件（Problem events）被关闭：

- 所有问题 - 正常事件（OK event）将关闭触发器创建的所有打开的问题；
- 所有问题如果标记的值匹配 - 正常事件（OK event）将关闭触发器创建的打开的问题，并且至少有一个匹配的标记值。标记由“匹配”触发器设置标记定义。如果没有问题事件（Problem event）关闭，那么正常事件（OK event）将不会生成。这通常被称为触发级事件关联。

事件关联

事件关联（也被称为全局事件关联）是一种设置自定义事件关闭（导致正常事件生成）的规则。

这个规则定义了新的问题事件如何于现有的问题事件配对，并通过生成相应的正常事件来关闭新的事件或匹配事件。

但是，必须仔细地配置事件关联，因为它可能会对事件处理性能造成负面影响，或者如果配置不当，则会关闭比预期更多的事件（在最坏的情况下可能会关闭所有的问题事件）。以下是几个关于配置的小提示：

- 通过控制事件（与旧事件配对的事件）设置唯一的标记来相关的范围，并使用“新的事件标记（new event tag）”来关联条件；
- 不要忘记在使用“过去的事件标记”操作时添加基于过去事件的条件，否则可能会关闭所有现有的问题；
- 避免在使用不同关联配置时使用通用的标记名称。

任务管理器

如果允许在触发器中启用“允许手动关闭”，那么可以手动关闭触发器生成的问题事件。这在事件确认期间的界面中完成。这个事件并不是直接关闭，而是创建一个“关闭事件”的任务，任务管理器很快会处理它。任务管理器将会生成一个相应的正常事件，并且问题事件将会关闭。

2 手动关闭问题事件

概述

当触发器的状态从“问题（Problem）”变成“正常（OK）”时，问题事件通常会自动解决，但是很难判断是通过触发器表达式的方式解决。在这种情况下，就需要手动解决问题。

例如，*syslog*可能会报告一些内核参数需要调整以获得最佳性能，在这种情况下，问题报告给Linux管理员，它们会修复它，然后手动关闭此问题。

只有在触发器选项中启用“允许手动关闭”选项，问题事件才可以被手动关闭。

当一个问题事件是“手动关闭”时，Zabbix会为Zabbix Server生成了一个新的内部任务，然后任务管理器进程执行这个任务，并生成正常事件。

手动关闭问题事件并不意味着底层的触发器将永远不会再次进入“问题”状态。当触发器中包含的任何监控项有新数据达到时，将重新评估整个表达式，并可能会再次生成问题。

配置

需要两步来手动关闭问题事件。

触发器配置

在触发器的配置页面上，启用允许手动关闭（Allow manual close）选项。

Allow manual close

事件确认

如果已启用允许手动关闭的触发器出现问题，你可以进入该触发器的“确认事件”屏幕，并手动关闭该问题。

要关闭这个问题，可以在确认事件屏幕查看关闭问题（Close problem）选项，并点击确认（Acknowledge）。

Event acknowledgements

Message Fixed, closing.

History	Time	User	Message	User action
Acknowledge	<input checked="" type="radio"/> Only selected event <input type="radio"/> Selected and all unacknowledged PROBLEM events 13 events <input type="radio"/> Selected and all unacknowledged events 25 events			

Close problem

Acknowledge **Cancel**

该请求通过Zabbix server处理，通常需要几秒才能关闭问题。在此期间，该问题在前端页面的监测中 (*Monitoring*) → 问题 (*Problems*) 显示的状态为关闭中 (*CLOSING*)。

验证

下面的方式可以验证该问题是否被手动关闭：

- 通过监测中 (*Monitoring*) → 问题 (*Problems*) 页面查看事件的详细信息：
- 通过在提供此信息的通知消息中使用宏 {EVENT.ACK.HISTORY} 来验证。

3 其他事件来源

1 发现事件

Zabbix定期扫描网络发现规则中定义的IP范围，可以为每个规则单独配置检查频率。一旦发现主机或服务，就会生成一个发现事件（或多个事件）。

Zabbix可以生成以下事件：

事件	描述
Service Up	每当Zabbix检测到活跃的服务。
Service Down	每当Zabbix无法检测到服务。
Host Up	如果活跃的服务中至少有一个IP。
Host Down	如果所有的服务都没有响应。
Service Discovered	如果服务在维护时间之后恢复或者第一次被发现。
Service Lost	如果服务在运行后丢失。
Host Discovered	如果主机在维护时间滞后恢复或者第一次被发现。
Host Lost	如果主机在运行后丢失。

2 主动式客户端自动发现事件

主动式客户端自动注册会在Zabbix创建事件。

如果配置了自动注册，当以前未知的主动式客户端向服务器发起请求时，服务器将会自动注册该主动式客户端。服务器使用主动式客户端请求的IP地址和端口，添加到自动注册的主机里。

关于自动注册更多的信息，请查阅[自动注册](#)页面。

3 内部事件

在下面的情境下，就会发生内部事件：

- 监控项的状态从“正常”变为“不支持的”；
- 监控项的状态从“不支持的”变为“正常”；
- 自动发现规则的状态从“正常”变为“不支持的”；
- 自动发现规则的状态从“不支持的”变为“正常”；
- 触发器的状态从“正常”变为“未知的”；
- 触发器的状态从“未知的”变为“正常”。

从Zabbix2.2开始支持内部事件。引入内部事件的目的是允许在发生任何内部事件时通知用户，例如，一个监控项的状态变为不支持的，并停止采集数据。

5 事件关联

概述

通常，在Zabbix中正常事件会关闭所有的问题事件，但在某些情况下需要更的细致的方法。例如，当监控日志文件时，在日志文件中想要发现某些问题，并将它们单独关闭，而不是一起关闭。

当触发器配置页面的“多重问题事件生成”选项为启用的情况下，通常适用于日志监控、主动采集（trap）处理等。

在Zabbix中，可以根据[事件标签](#)关联问题事件。标签用于提取值并为问题事件创建标识。利用这一点，问题也可以根据匹配的标签进行关闭。

换言之，相同的触发器可以创建由事件标签标识的不同事件。因此，可以单独地标识问题事件，并基于事件标签地标识单独关闭。

事件关联可以被定义在：

- 触发器配置 - 可以使用一个触发器来将问题与解决方案相关联。
- 全局 - 可以使用全局关联规则从不同地触发器/轮询方法将问题与解决问题相关联。

工作原理

在日志监控中，可能会遇到下面类似地输出：

```
1. Line1: Application 1 stopped
2. Line2: Application 2 stopped
3. Line3: Application 1 was restarted
4. Line4: Application 2 was restarted
```

事件关联地想法是将从“Line1”的问题事件到“Line3”的恢复事件，和从“Line2”的问题事件到“Line4”的恢复事件相匹配，并能逐个关闭这些问题：

```
1. Line1: Application 1 stopped
2. Line3: Application 1 was restarted #problem from Line 1 closed
3.
4. Line2: Application 2 stopped
5. Line4: Application 2 was restarted #problem from Line 2 closed
```

为此，需要将通过标签将这些事件相关联，例如，可以标识为“Application 1”和“Application 2”。这个过程也可以将正则表达式应用于日志中来提取标签的值。然后，当事件创建时，他们分别给标识为“Application 1”和“Application 2”，并且问题可以与解决方法相匹配

配置

在触发器的配置界面配置事件关联：

- 转到触发器的[配置](#)界面；

Trigger Dependencies

Name	Log trigger2														
Severity	Not classified	Information	Warning	Average	High	<input type="button" value="Add"/>									
Problem expression	{Zabbix server:log[/var /log/messages,Stopping Stopped shutting Started Apache Service Session].str(Stopping)}=1 or {Zabbix server:log[/var /log/messages,Stopping Stopped shutting Started Apache Service Session].str(shutting)}=1														
<input type="button" value="Expression constructor"/>															
OK event generation	<input checked="" type="radio"/> Expression <input type="radio"/> Recovery expression <input type="radio"/> None														
Recovery expression	{Zabbix server:log[/var /log/messages,Stopping Stopped shutting Started Apache Service Session].str(Started)}=1														
<input type="button" value="Expression constructor"/>															
PROBLEM event generation mode	<input type="radio"/> Single <input checked="" type="radio"/> Multiple														
OK event closes	<input type="radio"/> All problems <input checked="" type="radio"/> All problems if tag values match														
Tag for matching	Application														
Tags	<table border="1"> <tr> <td>Application</td> <td><code>{{ITEM.VALUE}}.iregsub("(</code></td> <td><input type="button" value="Remove"/></td> </tr> <tr> <td>Application</td> <td><code>{{ITEM.VALUE}}.iregsub("(</code></td> <td><input type="button" value="Remove"/></td> </tr> <tr> <td>Application</td> <td><code>{{ITEM.VALUE}}.iregsub("%</code></td> <td><input type="button" value="Remove"/></td> </tr> </table> <input type="button" value="Add"/>						Application	<code>{{ITEM.VALUE}}.iregsub("(</code>	<input type="button" value="Remove"/>	Application	<code>{{ITEM.VALUE}}.iregsub("(</code>	<input type="button" value="Remove"/>	Application	<code>{{ITEM.VALUE}}.iregsub("%</code>	<input type="button" value="Remove"/>
Application	<code>{{ITEM.VALUE}}.iregsub("(</code>	<input type="button" value="Remove"/>													
Application	<code>{{ITEM.VALUE}}.iregsub("(</code>	<input type="button" value="Remove"/>													
Application	<code>{{ITEM.VALUE}}.iregsub("%</code>	<input type="button" value="Remove"/>													
Allow manual close	<input checked="" type="checkbox"/>														
URL															
Description															
Enabled	<input checked="" type="checkbox"/>														
<input type="button" value="Add"/> <input type="button" value="Cancel"/>															

- 选择“问题事件生成模式”的“多重”选项；

- 选择“正常事件关闭”的“如果标签值配置的所有问题”
- 输入事件匹配的标签名称；
- 从日志中提取标签的值以配置**事件标签**

如果配置成功，能偶看到标记的“application”的问题事件，并与监测中 → 问题页面看到结果相匹配

Problems										Export to CSV	
Time	Severity	Recovery time	Status	Info	Host	Problem	Duration	Ack	Actions	Tags	
08:38:18	High		08:38:18 RESOLVED		Zabbix server	Service Apache stopped	0	No		Service: Apache Webserver	

当为不相关的问题创建相似的事件标签时，有一些警告信息是关于配置错误的：

- 当由两个applications向相同的日志文件写入故障和恢复信息，用户通过在标签中使用单独的正则表达式来提取标签的名称。例如“application A”和来自宏{ITEM.VALUE}的“application B”（当消息格式不同时），然而，如果和正则表达式不匹配的话，可能会无法按照计划工作。不匹配的正则表达式将生成空的标签值，并且在问题和正常事件中的单个空标签值足以关联它们。因此，来自“application A”的恢复消息可能会意外地关闭来自“application B”地错误消息。
 - 实际上标签和标签的值只有在触发器触发时才会显示。如果所使用的正则表达式无效的话，则会使用默认的字段“UNKNOWN”进行替换。如果错过了标签值“UNKNOWN”的初始问题事件，那么可能会出现与标签值“UNKNOWN”的后续正常事件，并有可能导致关闭不应该关闭的问题事件。
- 如果用户使用没有宏功能的宏{ITEM.VALUE}作为标签值，则会有255个字符串的限制。当日志消息很长，并且前面255个字符串是不明确的话，就有可能导致类似的事件标签用于不相关的问题上。

配置全局关联

在略微不同的情况下，可能会有不同的触发器来触发问题和解决问题。例如，日志触发器可能会报告应用程序有问题，而轮询触发器可能会报告应用程序还处于运行状态。

利用事件标签，可以将日志触发器标记为“Status: Down”，而轮询触发器将标记为“Status: Up”。然而，在全局关联规则中，可以关联这些触发器并将动作的操作分配给此关联。例如关闭旧事件或关闭新事件。

配置事件的全局关联规则：

- 在前端点击*Configuration*，转到*Event correlation*页面；
- 在右上角点击“Create correlation”（或者在已有的关联名称上编辑）；
- 输入关联规则的参数。

Correlation Operations

Name	Close old events		
Type of calculation	And	A and (B and D) and E	
Conditions	Label	Name	Action
	A	Old event tag <i>Application</i> = new event tag <i>Application</i>	Remove
	B	Old event tag <i>Application</i> = ABC	Remove
	D	Old event tag <i>State</i> = Down	Remove
	E	New event tag <i>State</i> = Up	Remove
New condition	New event tag value	tag	= value
	Add		
Description	Close old events for Application ABC if an event with "State=Up" happens.		
Enabled	<input checked="" type="checkbox"/>		
	Add	Cancel	

- 选择关联规则的操作。

Correlation Operations

Operations	Details	Action
	Close old events	Remove
New operation	Close new event	
	Add	

6 可视化

1 图形

概述

随着大量的监控数据被采集到Zabbix中，如果用户可以以可视化的表现形式来查看发生了什么事情，那么和仅仅只有数字的表现形式比起来则更加轻松。

以下是进行图形设置的地方。图形可以一目了然地掌握数据的流向并关联问题，发现某件事情开始，或在某件事情可能变成问题事件时进行报告。

Zabbix为用户提供了如下几种图形：

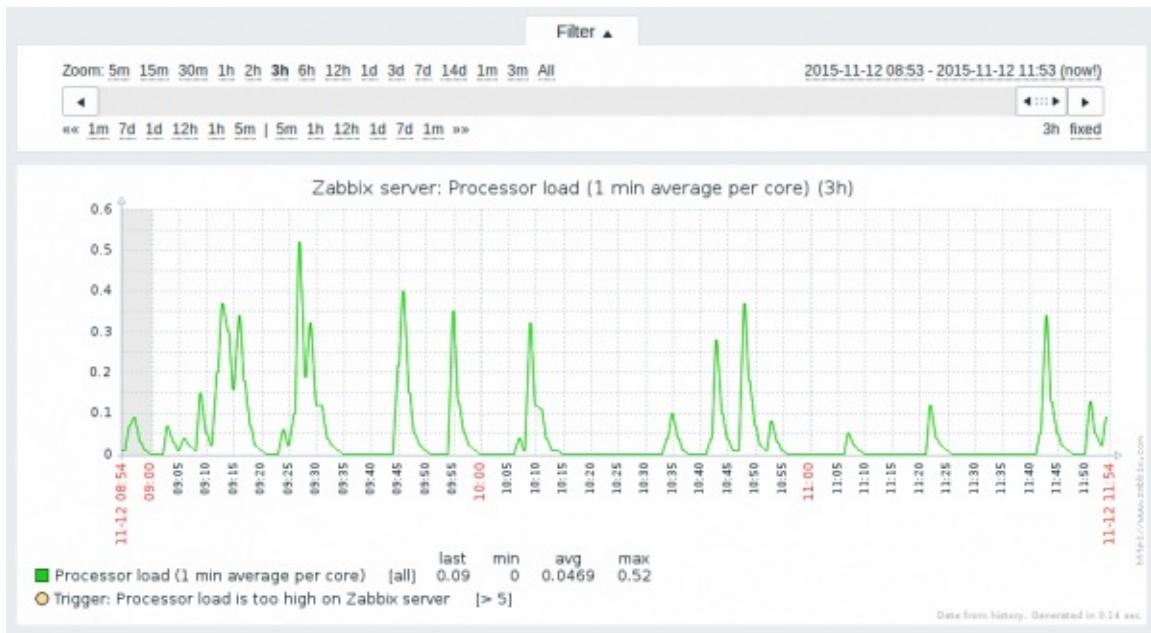
- 监控项数据的内置简单图形[simple graphs](#)；
- 可能创建更复杂的自定义图形[customised graphs](#)
- 在最新数据中，可以利用特定图形[ad-hoc graphs](#)快速访问几个监控项的数据比较

1 简单图形

Overview

Zabbix提供了简单图形，用来可视化显示监控项采集到的数据。对于用户而言，并不需要进行配置就可以查看简单图形。这是由Zabbix免费提供的。

通过*Monitoring → Latest data*点击各自监控项的图形链接，就可以展示图形。



时间段选择器

注意图形上方的时间段选择器。它允许你可以轻松选择所需的时间段。

时间段选择器中的滑动快可以来回拖动，以及缩放，使之更有效地改变展示的时间段。左侧的链接允许选择一些常用的预定义时间段（在滑动区域的上方），并点击时间段的链接来回移动（滑动区域的下方）。通过右侧的时间链接，点击可以弹出日历设置特定的开始/结束时间。

在右下角的**fixed/dynamic**链接具有以下效果：

- 控制在日历弹出窗口中更改开始/结束时间时，是否时间段保持不变；
- 当选择`fixed`时，点击左下方的时间段链接（`< 1m 7d 1d 12h 1h 5m | 5m 1h 12h 1d 7d 1m >`）将会移动滑块，而不会改变其滑块的尺寸，时间段会随着时间向前或向后移动，并不会改变时间段；当选择`dynamic`时，点击左下方的时间段链接将会随着向左向右的时间段链接而左右放大滑块。
- 当选择`fixed`时，点击滑块上较大的`<`和`>`按钮将移动滑块，而不会改变其滑块的尺寸；而当选择`dynamic`时，点击`<`按钮将随左右方向而向左或向右放大滑块，每次点击都以1天为单位左右移动。

控制选择时间段展示的另一种方法就是按住鼠标左键选择图形中想要展示的区域，该区域将高亮显示，当松开鼠标左键时，图形将放大显示到刚才选中的高亮区域。

数值类型的监控项可以使用简单图形。对于文本类型的监控项，可以使用监测中 → 最新数据中的历史记录链接。

最新数据和历史数据

对于最新的数据，通过每个收到的值绘制连接成一条单线。只要有且至少有一个可用于一个值的水平像素，就会绘制单线。

对于历史数据，将绘制连接成三条线，深绿色的线显示平均值，而深粉色和浅绿色则显示该时间点的最大值和最小值，最大值和最小值的中间部分用黄色背景填充。

工作时间（工作日）在图形中显示为白色背景，当非工作时间将显示为灰色（前端使用的主题为默认的深蓝时）。



简单图形会自动显示工作时间，而显示[自定义图形](#)需要用户配置。如果图形显示超过3个月的数据，那么将不显示工作时间。

生成历史/趋势数据图形

图形是基于[历史](#)和[趋势](#)数据生成的。在图形下方的灰色标题表明了数据来自哪里。

下面几个因素将影响是使用历史数据还是趋势数据：

- 较老的监控项历史数据。例如，监控项的历史数据只保留14天。在这个时候，如果查看14天以后的数据，那么会从趋势数据绘制图形。
- 图形中的数据拥挤。如果图形的水平像素超过 $3600/16$ ，Zabbix会使用趋势数据（即使监控项的历史数据在同一时期仍然可用）。
- 如果禁用趋势数据，并且监控项的历史数据在同一时期是可用的，那么图形将使用监控项的历史数据来构建。这从Zabbix2.2.1开始受支持（在Zabbix2.2.1之前，如果禁用了趋势数据，即使监控项的历史数据可用，那么只会显示一段空白的图形）。

切换到原始值

在页面的右上角的下拉菜单允许从简单图形切换到值/最近的500个值。这可以用于查看构建图形的数值。

这里的值表现的是原始的值，即没有使用单位或后期处理的值。然而，值映射是被应用了的。

已知的问题

查看图形 [已知的问题](#)。

2 自定义图表

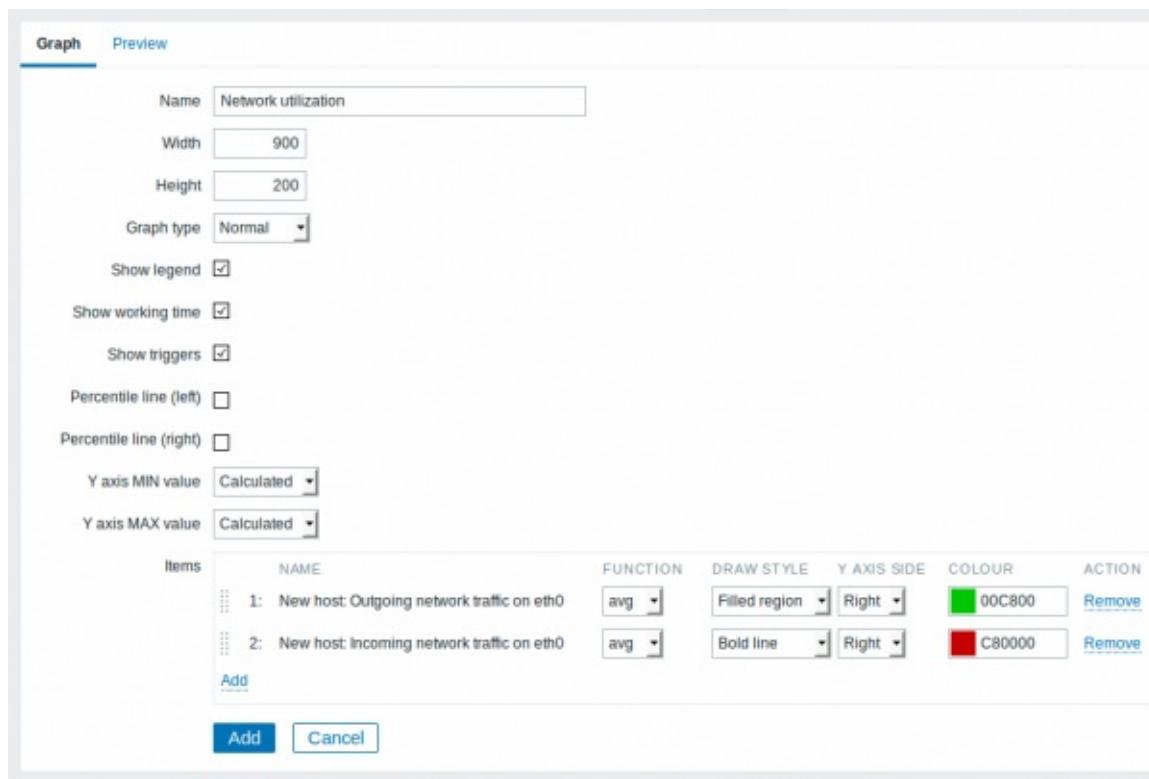
概述

自定义图表，顾名思义，就是提供定制的功能。虽然简单图形对于查看单个监控项的数据很适用，但它们并不提供配置功能。因此，如果想要更改图形的样式、线条的显示方式或是比较多个监控项。例如，单个图形中显示接收和发送的流量，就需要自定义图形。自定义图形是手动配置的。可以为单个主机、多个主机、单个模板创建自定义图形。

配置自定义图表

按照以下步骤创建自定义图形：

- 在前端页面点击 *Configuration* → *Hosts* (或 *Templates*)；
- 点击所要创建图形的主机或模板旁的 *Graphs*；
- 在图形屏幕上点击 *Create graph*；
- 编辑图形的属性。



图形的属性

参数	描述
名称	图形名称（唯一的）。从Zabbix 2.2开始，在名称中可以使用监控项的值，通过使用标准 <code>{host:key.func(param)}</code> 参数的简单宏。在这个宏中只支持 <code>avg</code> 、 <code>last</code> 、 <code>max</code> 、 <code>min</code> 这些以秒为参数的函数。宏 <code>{HOST.HOST<1-9>}</code> 支持在这个宏中使用，在图形中引入第一、第二、第三等主机。
宽	图形的宽度，以像素为单位（仅用于预览饼图pie/爆炸exploded图形）。
高	图形的高度，以像素为单位。

图形类型	图形类型:正常 - 正常的图形, 图形显示为线条层积 - 叠层图, 通过填充区域展示饼图 - 饼图爆炸 - “爆炸”的饼图, 部分显示为切出的饼图
查看图例	设置图形图例的展示。
查看工作时间	如果选中, 非工作时间将显示为灰色的背景。这个选项不适用于饼图和爆炸饼图。
查看触发器	如果选中, 触发器在背景中显示为红色的线。这个选项不适用于饼图和爆炸饼图。
百分比线(左)	在左边的Y轴显示百分比。例如, 百分比线的参数设置为95%, 那么百分比线将位于95%的值所在的水平, 并显示为亮绿色的线。这个选项只适用于正常的图形。
百分比线(右)	在右边的Y轴显示百分比。例如, 百分比线的参数设置为95%, 那么百分比线将位于95%的值所在的水平, 并显示为亮红色的线。这个选项只适用于正常的图形。
Y轴的最小值	Y轴的最小值: 可计算的 - 自动计算Y轴最小值固定的 - Y轴的最小值是固定的。这个选项不适用于饼图和爆炸饼图。监控项 - 监控项的最后一个值是将成为最小值。
Y轴的最大值	Y轴的最大值: 可计算的 - 自动计算Y轴的最大值固定的 - Y轴的最大值是固定的。这个选项不适用于饼图和爆炸饼图。监控项 - 监控项的最后一个值将成为最大值。
3D视图	启用3D风格。这个选项只适用于爆炸饼图。
监控项	监控项, 构成图形的数据来源。

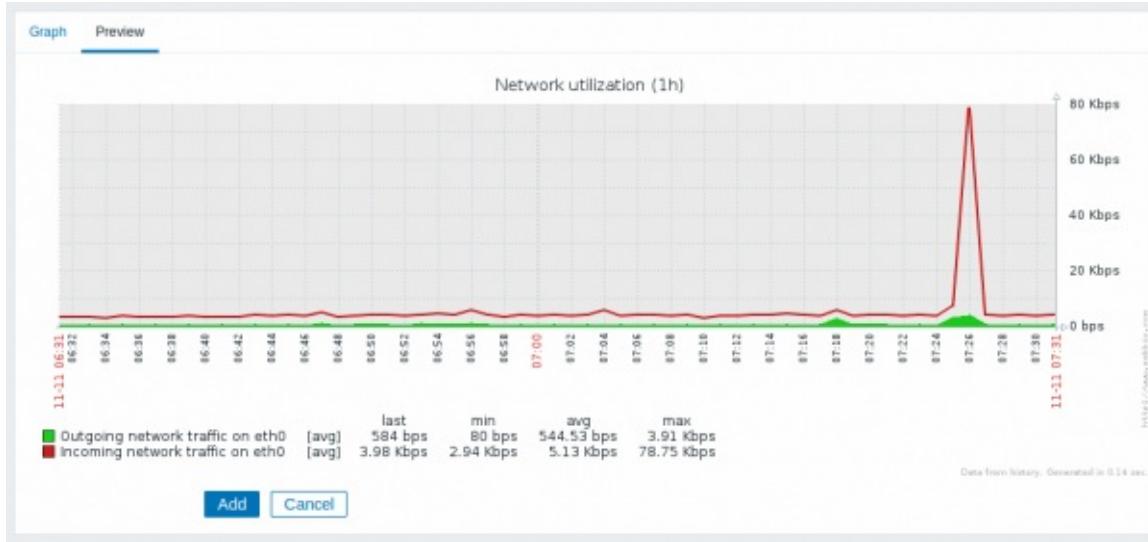
配置图表项

添加在图形中展示数据的监控项，在前端的图形页面点击“监控项”的“添加”选项，选择相应的监控项，并设置监控项数据展示的属性。监控项展示的属性：

参数	描述
排序次序(0~100)	绘制顺序。将首先处理顺序为0的监控项。可用来绘制线条或区域填充后面(或前面)另一个。\\可以拖动监控项头部的图标上下拖动来设置它们的先后顺序。
名称	监控项的名称, 将其数据用来展示。
类型	类型(此项只适用于饼图和爆炸饼图): 简单 - 将监控项的值按照比例显示。图形总数 - 监控项的值占满整个饼图。请注意:“图形总数”监控项的着色只有在不被“成比例的”监控项占用的范围内可见。
功能	当一个监控项中有多个值时, 将显示为什么值: 所有 - 所有的值(最小、平均值和最大)最小 - 仅显示最小值平均 - 仅显示平均值最大 - 仅显示最大值
绘图风格	绘图风格(此选项只适用于正常图形; 对于层积的图形始终显示为填满的区域): 线条 - 绘制线条填满的区域 - 绘制为填满的区域粗线 - 绘制为粗线点 - 绘制为点虚线 - 绘制为虚线
Y轴的位置	Y轴将处于左边还是右边。
颜色	RGB颜色显示为十六进制字符。

图形预览

在预览的选项卡中, 将显示配置图形的预览, 以便可以立即查看创建的内容。



需要注意的是：预览并不会显示任何模板监控项的数据。



在本例中，请注意显示触发器水准和触发器信息的粗体虚线

3个触发器是在图例中显示的触发器的上限。如果图形高度设置为小于120像素，那么图例中将不会显示触发器的信息。

3 特设图形

概述

虽然一个[简单图形](#)非常适合查阅一个监控项的数据，同时[自定义图形](#)提供了定制的选项，但是两者都不允许快速创建多个监控项数据的比较图形，工作量小且没有维护。

为了解决这个问题，从Zabbix 2.4开始就可以快速创建多个监控项的特设图形。

配置

创建特设图形，需要执行以下步骤：

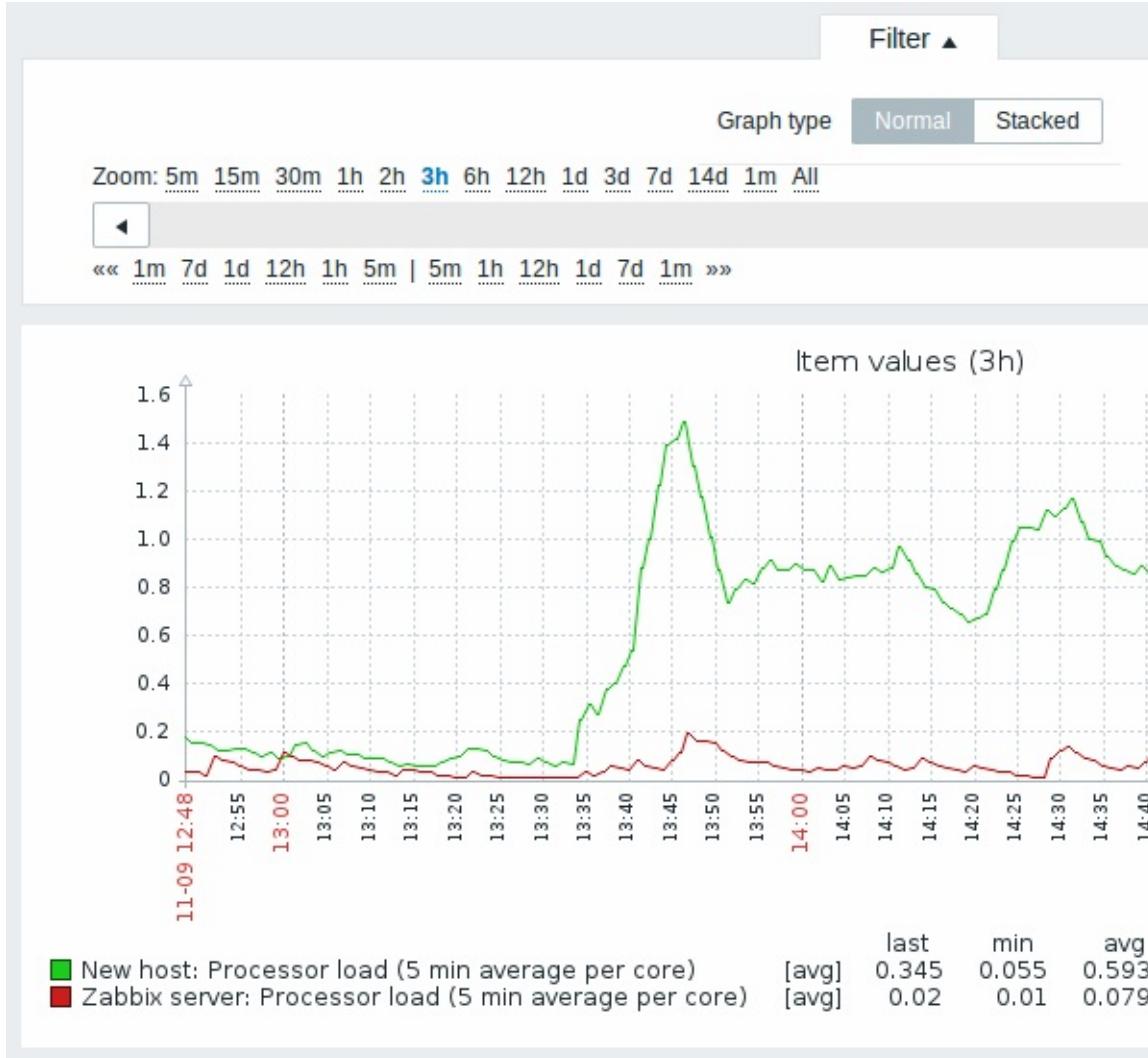
- 跳转至前端页面的监测中 → 最新数据；
- 使用过滤器选择要展示的监控项；
- 选中想要绘制的监控项的复选框；
- 点击显示堆叠数据图或显示数据图按钮。

The screenshot shows the 'Latest data' section of the Zabbix frontend. At the top, there are filters for 'Host groups' (set to 'Discovered hosts'), 'Name' (set to '(5 min average)'), and checkboxes for 'Show items without data' (checked) and 'Show details' (unchecked). Below the filters is a table of monitoring items:

HOST	NAME	LAST CHECK	LAST VALUE	CHANGE	
Zabbix server	CPU (1 item)				
<input checked="" type="checkbox"/>	Processor load (5 min average per core)	2015-08-20 10:31:15	0.07	-0.01	Graph
New host	CPU (1 item)				
<input checked="" type="checkbox"/>	Processor load (5 min average per core)	2015-08-20 10:30:43	0.67	+0.07	Graph

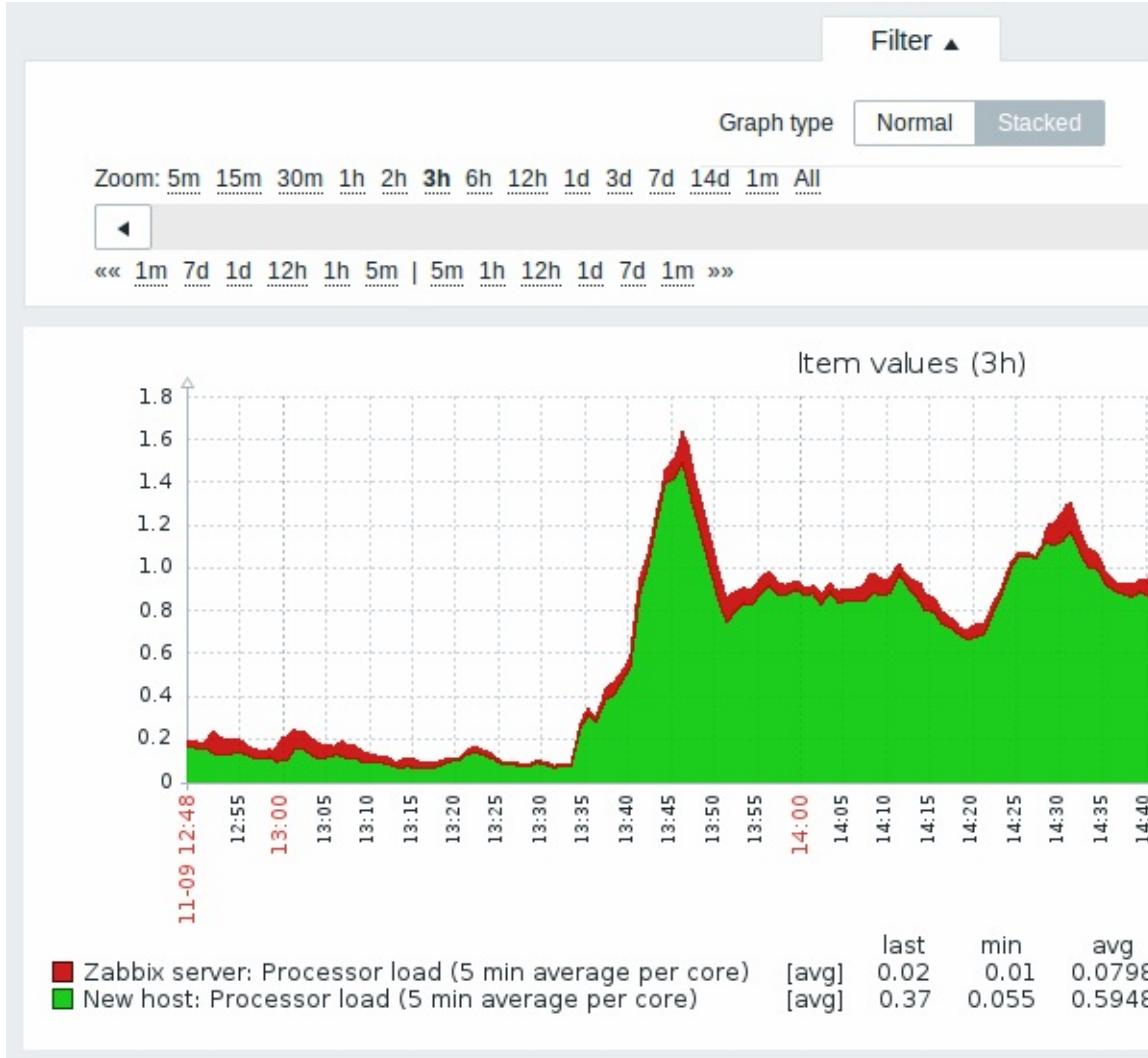
At the bottom of the table, there are buttons for 'Display stacked graph' and 'Display graph'. The 'Display graph' button is highlighted with a blue border.

创建的图形显示为：



值得注意的是，为避免在图形中显示太多的线条，只显示每个监控项的平均值（最大/最小值的线条不显示）。触发器和触发器信息也不显示在图形中。

在创建的图形窗口中，可以使用时间段选择器，并可以从“正常”切换至“层积”的图形风格。



2 拓扑图

概述

如果有一个网络环境，作为运维人员，想要了解其基础设施的状况。为此，可以在Zabbix中创建网络拓扑图。

所有用户都可以创建网络拓扑图。这个拓扑图可以是公开的（对所有用户可用）或私人的（对选定的用户可用）。

开始[配置拓扑图](#)。

1 配置拓扑图

概述

在Zabbix中配置拓扑图，首先需要拓扑图，并定义关于拓扑图的常规参数，然后就可以使用图像等元素填充实际拓扑图。

你可以使用主机、主机组、触发器、图像或其他拓扑图元素填充拓扑图。

图标用来表示拓扑图元素。你可以定义与图标一起显示的信息，并设置以特殊方式显示最近的问题。与此同时，你可以链接图标并定义在此链接上显示的信息。

你可以通过点击图标来添加可访问的自定义URLs。因此，你可以将主机图标链接到主机属性，或将拓扑图图标链接到其他拓扑图。

通过 *Monitoring → Maps* 对拓扑图进行管理，可以对其进行配置、管理和浏览。在监控视图中，你可以点击图标，并利用一些脚本和URLs的链接。

在Zabbix中，所有的用户（包括非管理员用户）可以创建网络拓扑图。拓扑图拥有其所有者，该所有者表示是谁创建了这个拓扑图。

拓扑图可以是被公开的，也可以是私有的。公开的拓扑图对所有用户是可见的，然而，用户必须具有读取所有拓扑图元素的权限才能查看拓扑图。要向拓扑图添加元素的话，用户还必须至少具有对拓扑图的可读权限。

私有的拓扑图只允许它的所有者访问。私有拓扑图可以通过其所有者共享到其他的用户和用户组。常规（非超级管理员）用户只能与他们所属的组和用户共享。私有拓扑图对所有者和拓扑图被共享的用户将是可见的，只要他们他们拥有对所有地图元素的读取权限即可。管理员级别的用户，同样拥有所有拓扑图元素的读取权限，就可以查看和编辑私有地图，而不用管所有者或属于所有者共享的用户。

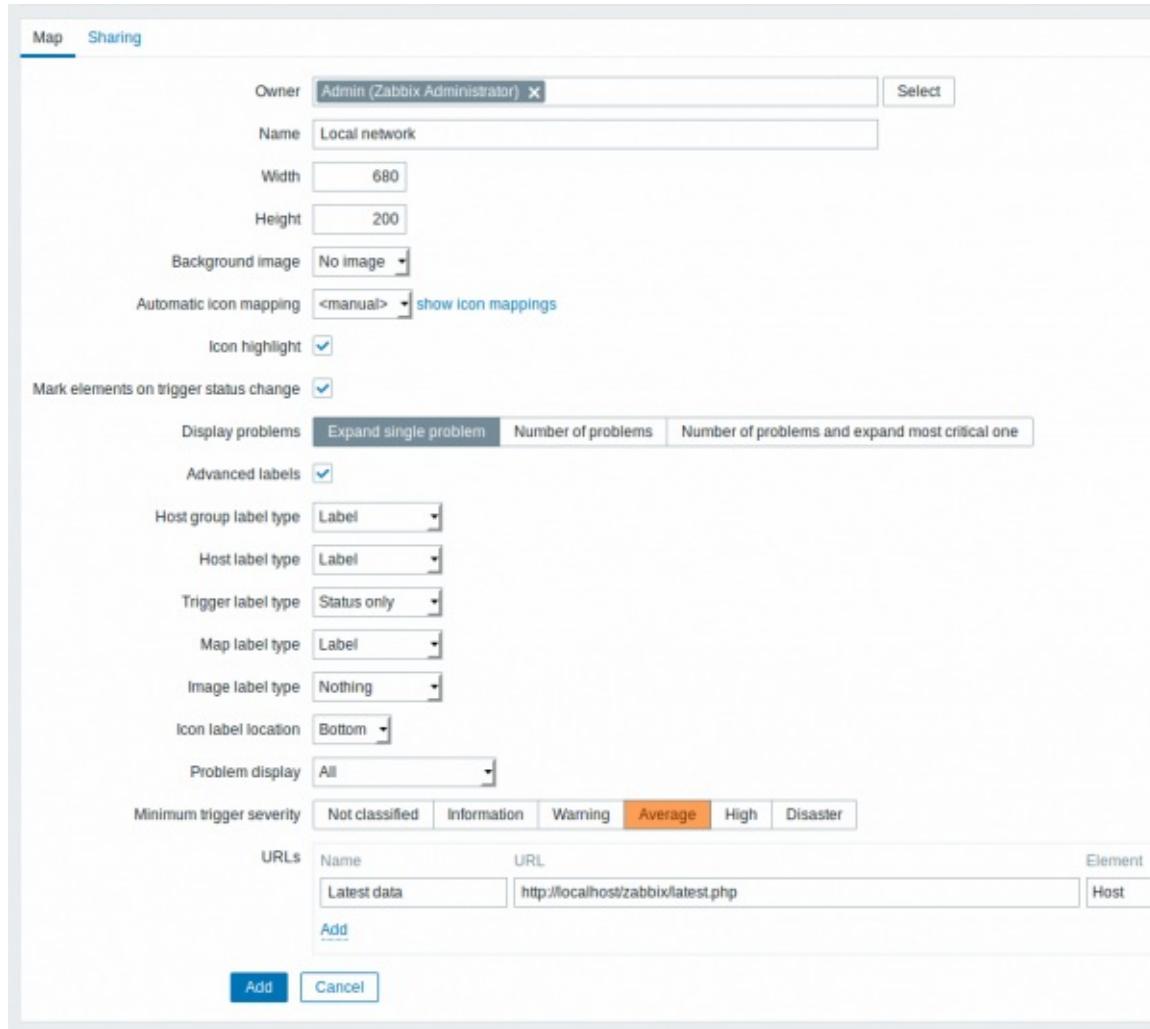
创建一个拓扑图

通过以下步骤创建一个拓扑：

- 在Zabbix前端页面打开*Monitoring → Maps*；
- 点击所有拓扑图来查看；
- 在右上角点击创建拓扑图。

你还可以在现有拓扑图的配置页面的使用*Clone* 和 *Full clone*按钮，创建一个新的拓扑图。点击*Clone*按钮将保留原拓扑图的常规布局属性，但是没有元素。而*Full clone*不仅保留原拓扑图的常规布局属性，并且会保留原拓扑图的所有元素。

在 拓扑图 标签页面包含一些常规的拓扑图属性：



常规拓扑图属性为：

参数	描述
所有者	该拓扑图所有者的名称。
名称	唯一的拓扑图名称。
宽	拓扑图的宽度，以像素为单位。
高	拓扑图的高度，以像素为单位。
背景图片	使用背景图片：没有图片 - 没有背景图片（默认为白色的背景）图片 - 选择一张图片来作为背景图片。不执行图片缩放。你可以使用地图或其他图片来优化拓扑图。
图标自动映射	你可以设置为使用图标自动映射，在Administration → General → Icon mapping进行配置，图标映射允许某些图标与某些主机的库存字段进行映射。
图标高亮	如果你选中此框，那么图标将会高亮显示。具有活动触发器的元素将于严重等级的触发器一同显示为相同颜色的圆形背景。此外，如果所有的问题被确认，那么会在圆形周围显示一条加粗的绿色线条。状态为“禁用”和“在维护中”的元素将会分别获得灰色的正方形背景和橙色的正方形背景。
触发器状态变更的标记元素	触发器状态（最近的问题或解决方案）的最近变更状况将会在元素图标的三边，且三边没有标签的标记（向内侧的红色三角形）上高亮显示。标记将显示30分钟。
展开问题详情	如果拓扑图上的元素（主机、主机组或其他拓扑图）存在一个问题，则此选项将会控制问题（触发器）显示为具体名称还是问题计数。如果选中此选项，那么将使用问题名称。
高级标签	如果选中此框，你将能够定义为单独的元素类型定义单独的标签类型。

图标标签类型	图标可使用的标签类型为：标签 - 图标标签IP 地址 - IP 地址元素名称 - 元素名称（例如主机名称）只有状态 - 只有状态（正常OK 和 问题PROBLEM）无 - 没有标签可显示。
图标标签位置	与图标相关的标签位置：底部 - 在图标的下面左边 - 在左边右边 - 在右边顶部 - 做图标的上面
问题显示	问题计数显示为：所有的 - 将显示完整的问题数分开的 - 未确认的问题数将显示为分开的若干个总问题数。unacknowledged problem count will be displayed separated as a number of the total problem count仅未确认的 - 仅显示未确认的问题数
最小的触发器严重程度级别	低于最小的触发器严重程度级别的问题将不会被显示在拓扑图中。例如，如果选择严重级别，那么“信息”和“未分类”触发器严重程度级别的变更将不会在拓扑图中体现。这个参数从Zabbix2.2后开始支持。
URLs	可以定义每个元素的URLs（带有标签）。当用户在拓扑图的查看模式下点击元素时，它们将会显示为链接。这些宏可以在拓扑图URLs中使用：{MAP.ID}、{HOSTGROUP.ID}、{HOST.ID}、{TRIGGER.ID}

在分享标签页包含拓扑图类型以及权限为私人的拓扑图的共享选项（用户组、用户）：

Type	Private	Public	
List of user group shares	User groups Network administrators	Permissions Read-only Read-write	Action Remove
List of user shares	Users Admin (Zabbix Administrator)	Permissions Read-only Read-write	Action Remove
<input type="button" value="Add"/> <input type="button" value="Cancel"/>			

参数	描述
类型	选择拓扑图的类型：私人的 - 拓扑图只对选定的用户组和用户可见。公开的 - 拓扑图对所有人可见。
共享用户组的列表	选择拓扑图允许访问的用户组。你可以允许给予只读还是读写权限。
共享用户的列表	选择拓扑图允许访问的用户。你可以给予只读还是读写权限。

当你点击添加以保存这个拓扑图时，你已经创建了一个其名称、尺寸和某些选择参数都为空的拓扑图。现在你需要添加一些元素。为此，在拓扑图列表中点击构造函数来打开编辑区域。

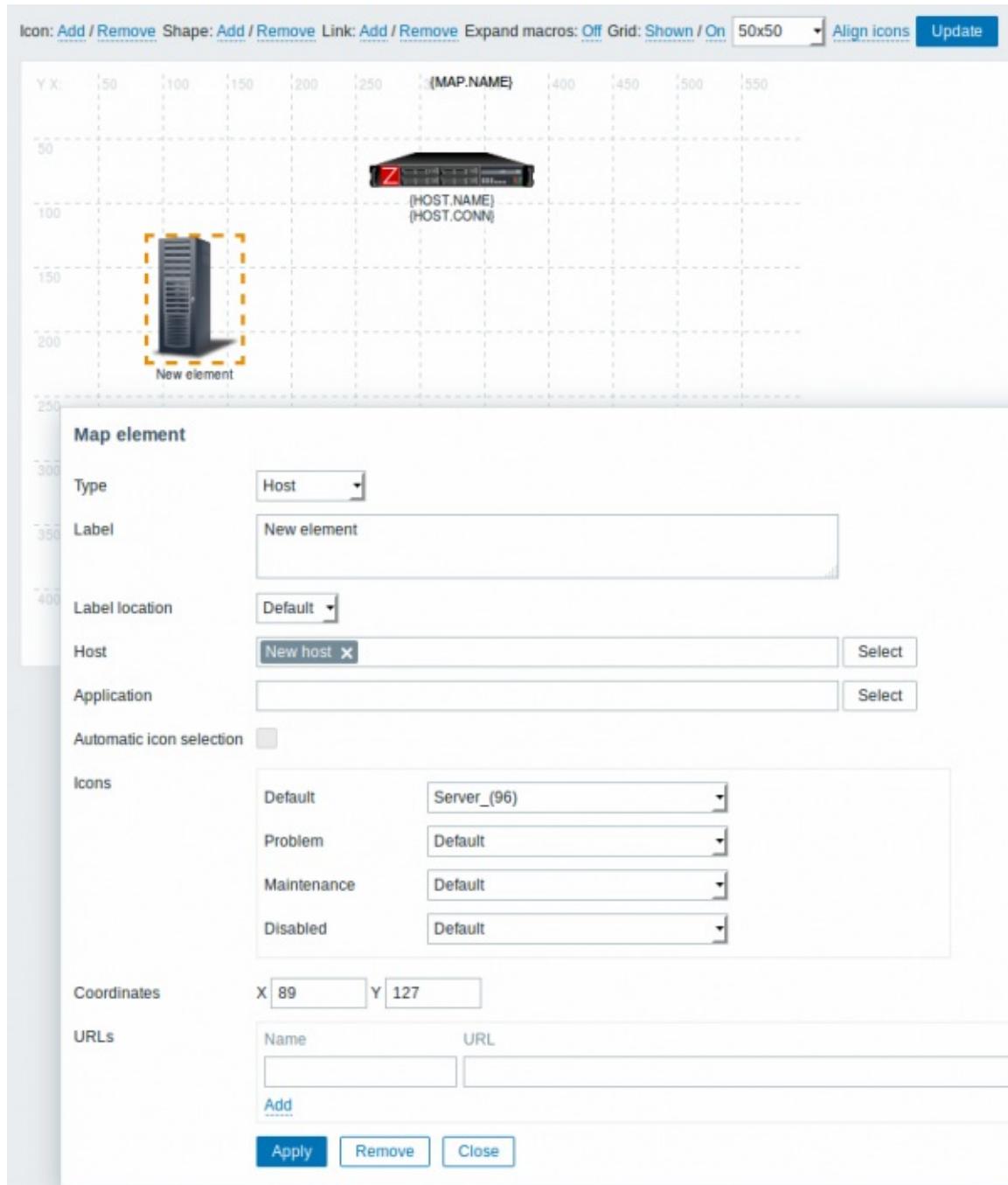
添加元素

添加元素，点击图标旁边的添加链接。新的元素将出现在拓扑图的左上角，将其拖到任何你想要的地方。

值得注意的是，“网格”选项为“开”的话，元素将始终与网格对齐（你可以从下拉菜单选择各种不同的网格尺寸，也可以隐藏/显示网格）。如果你想要将元素放到任何地方而不需要对齐，可以将此选项设置为“关”。（随机的元素可以稍

后通过对齐图标按钮达到与网格对齐)

现在你已经添加了一些元素，你可能需要通过给每个元素定义名称来区分它们，这时候，可以点击元素图标，之后出现一个表单，你可以在表单上设置元素的类型、设置元素的名称、选择不同的图标等。



拓扑图元素属性：

参数	描述
类型	元素的类型：主机 - 图标表示的是所选主机的所有触发器的状态；拓扑图 - 图标表示的是拓扑图所有元素的状态；触发器 - 图标表示的是单个触发器的状态；i主机组 - 图标表示的是所选主机组里的所有主机触发器的状态； 图像 - 一个图标，不会链接到任何资源。
标签	图标的标签，可以为任何字段。 标签中可以使用宏和多行字符串。
标签位置	与图标相关联的标签位置：默认 - 拓扑图的默认标签位置底部 - 在图标之下左边 - 在左边右边 - 在右边顶部 - 在图标之上
主机	如果元素类型为“主机”的话，请输入主机。该字段是自动完成的，所以在开始键入主机名称的时候，将

主机	在下拉菜单提供与之匹配的主机。向下滚动选择主机。与此可以通过点击“x”来删除所选主机。
拓扑图	选择拓扑图，如果选择的元素类型为“拓扑图”的话。
触发器	选择触发器，如果选择的元素类型为“触发器”的话。
主机组	如果选择的元素类型为“主机组”的话，请输入主机组。该字段是自动完成的，所以在开始键入主机组的时候，将在下拉菜单提供与之匹配的主机组。向下滚动选择主机组。与此可以通过点击“x”来删除所选的主机组。
应用集	你可以选择一个应用集，允许仅显示属于应用集的触发器的问题。该字段在主机和主机组的元素类型上是可用的。该参数从Zabbix 2.4.0后开始支持。
图标自动选择	在这种情况下，将使用图标映射来确定要使用的图标。
图标	在这种情况下：默认、问题、维护、禁用，你可以为元素选择显示不同的图标。
X轴坐标	拓扑图元素的X轴坐标。
Y轴左边	拓扑图元素的Y轴坐标。
URLS	可以为元素设置元素特定的URLs。当用户在拓扑图查看模式下点击元素时，这些将显示为链接。如果元素具有自己的链接，并且定义了类型为拓扑图级别的URLs，那么它们会被合并到同一菜单中。可以在拓扑图元素URLs中使用的宏为：{MAP.ID}、{HOSTGROUP.ID}、{HOST.ID}、{TRIGGER.ID}

<重点注意的>添加的元素不会自动保存。如果你离开当前页面，那么所有更改将会丢失。

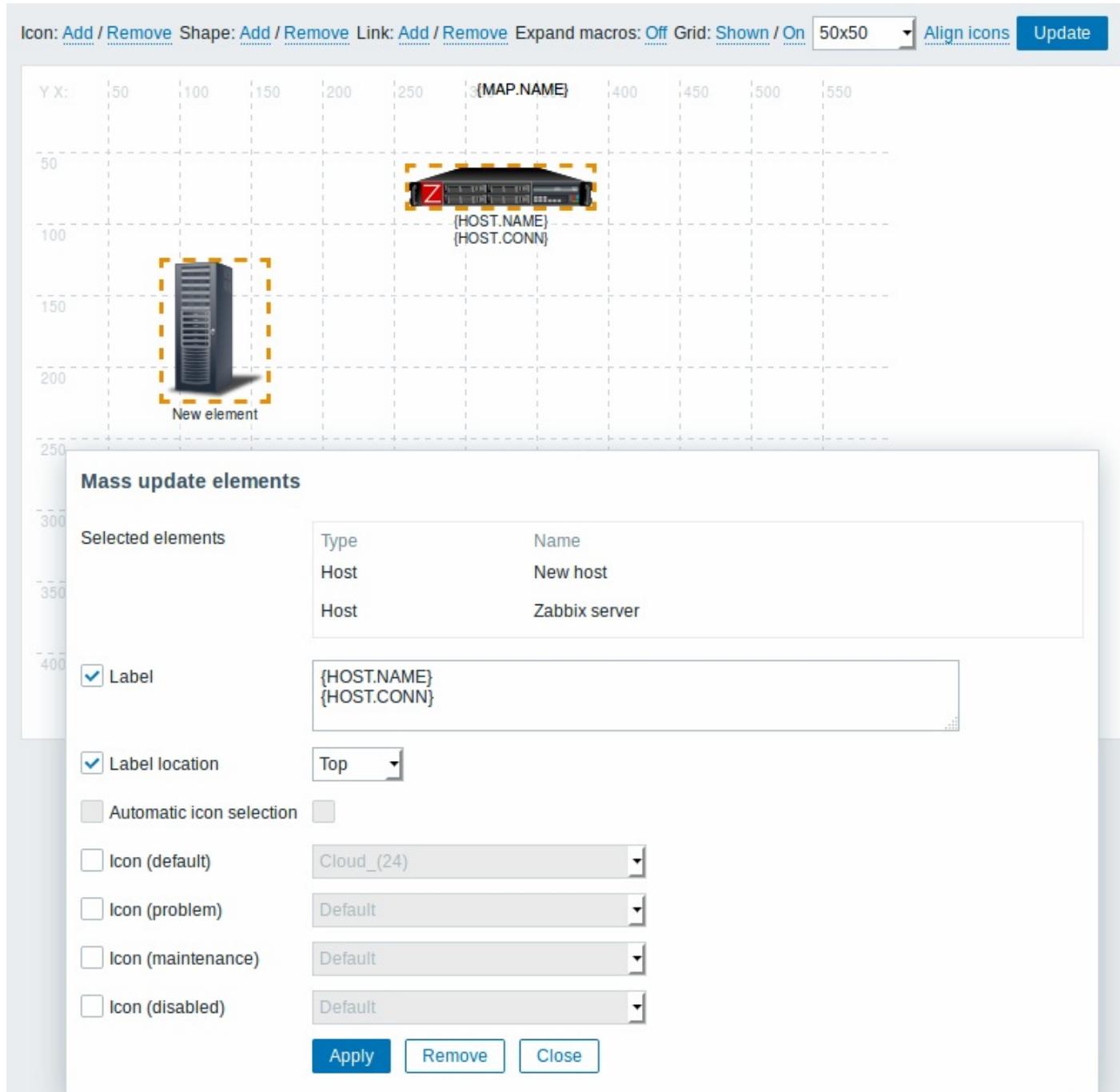
因此，最好点击右上角的“更新”按钮。一旦点击，所有更改将会被保存，无论你在下面弹出的串口选择什么。每个拓扑图也保存选定的网格选项。</note>

选择元素

要选择元素，需要选中一个元素，然后按住Ctrl键来选择另一个元素。

你可以通过在可编辑区域中鼠标单击后鼠标拖动显示的矩形框来选择多个元素（Zabbix 2.0后可用）。

一旦选择多个元素后，元素属性窗口将切换为批量更新元素模式，以便你可以一次更改所选元素的属性。为此，可以使用该多选框标记属性，并为其输入一个新值。同样的，你可以在此使用宏（例如，{HOST.NAME}表示元素标签）。



链接元素

一旦你在拓扑图上放置了一些元素，那么是时候开始链接它们。要链接两个元素，你必须首先选中它们，当元素被选中时，点击“链接”旁边的“添加”进行链接。

当链接被创建后，单个元素表现在一个附加的链接部分，点击编辑进行编辑链接属性。

Icon: Add / Remove Shape: Add / Remove Link: Add / Remove Expand macros: Off Grid: Shown / On 50x50 Align icons Update

Map element

Type	Host								
Label	New element								
Label location	Default								
Host	New host <input type="button" value="X"/>								
Application	<input type="button" value="Select"/>								
Automatic icon selection	<input type="checkbox"/>								
Icons	<table border="1"> <tbody> <tr> <td>Default</td> <td>Server_(96)</td> </tr> <tr> <td>Problem</td> <td>Default</td> </tr> <tr> <td>Maintenance</td> <td>Default</td> </tr> <tr> <td>Disabled</td> <td>Default</td> </tr> </tbody> </table>	Default	Server_(96)	Problem	Default	Maintenance	Default	Disabled	Default
Default	Server_(96)								
Problem	Default								
Maintenance	Default								
Disabled	Default								
Coordinates	X 89 Y 77								
URLs	<table border="1"> <thead> <tr> <th>Name</th> <th>URL</th> </tr> </thead> <tbody> <tr> <td><input type="text"/></td> <td><input type="text"/></td> </tr> </tbody> </table> <p>Add <input type="button" value="Apply"/> <input type="button" value="Remove"/> <input type="button" value="Close"/></p>	Name	URL	<input type="text"/>	<input type="text"/>				
Name	URL								
<input type="text"/>	<input type="text"/>								
Links	<table border="1"> <thead> <tr> <th>Element name</th> <th>Link indicators</th> <th>Action</th> </tr> </thead> <tbody> <tr> <td>Zabbix server</td> <td><input type="button" value="Edit"/></td> <td><input type="button" value="Edit"/></td> </tr> </tbody> </table>	Element name	Link indicators	Action	Zabbix server	<input type="button" value="Edit"/>	<input type="button" value="Edit"/>		
Element name	Link indicators	Action							
Zabbix server	<input type="button" value="Edit"/>	<input type="button" value="Edit"/>							
Label	<input type="text" value="100Mbps"/>								
Connect to	<input type="button" value="Zabbix server"/>								
Type (OK)	<input type="button" value="Bold line"/>								
Colour (OK)	<input type="color" value="#00CC00"/>								
Link indicators	<table border="1"> <thead> <tr> <th>Trigger</th> <th>Type</th> <th>Colour</th> <th>Action</th> </tr> </thead> <tbody> <tr> <td><input type="button" value="Add"/></td> <td><input type="button" value="Edit"/></td> <td><input type="button" value="Edit"/></td> <td><input type="button" value="Edit"/></td> </tr> </tbody> </table> <p><input type="button" value="Apply"/> <input type="button" value="Remove"/> <input type="button" value="Close"/></p>	Trigger	Type	Colour	Action	<input type="button" value="Add"/>	<input type="button" value="Edit"/>	<input type="button" value="Edit"/>	<input type="button" value="Edit"/>
Trigger	Type	Colour	Action						
<input type="button" value="Add"/>	<input type="button" value="Edit"/>	<input type="button" value="Edit"/>	<input type="button" value="Edit"/>						

链接属性：

参数	描述
标签	将在链接顶部呈现的标签。在这个字段支持这个宏 <code>{host:key.func(param)}</code> ，但只能使用“avg”、“last”、“min”和“max”这些触发器函数，这些函数以秒为单位。
连接到	链接链接到的元素。
类型 (OK)	默认链接的样式：线 - 单条线粗线 - 粗线点 - 点虚线 - 虚线
颜色 (OK)	默认的链接的颜色
链接指示器	与链接相链接的触发器列表。在这种情况下，触发器的状态状态为“问题”，则其样式将应用于该链接。

2 链接指示器

概述

你可以为网络拓扑图中的元素之间的[链接](#)分配一些触发器。当这些触发器状况为“问题”状态时，可以在链接上体现出来。

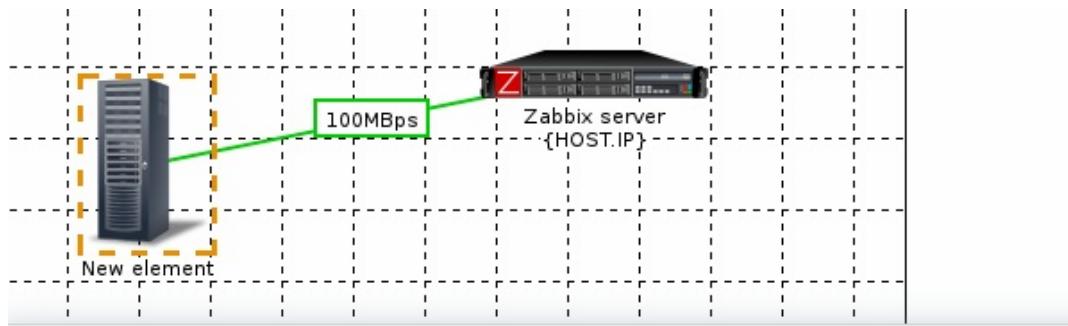
当你配置一个链接，你设置为默认的链接类型和颜色。当你分配触发器到一个链接时，你可以为这些触发器分配不同的链接类型和颜色。

如果这些触发器中任何一个进入“问题”状态，它们的链接样式和颜色将显示在链接上。也许你的默认链接为绿线。现在，在“问题”状态的触发器中，你的链接可能会变成粗体红色（如果你已经定义了它）。

配置

要将触发器分配到链接指示器，需要执行以下步骤：

- 选择一个拓扑图元素；
- 在“批量更新元素”的链接属性的相应链接里，点击编辑；
- 在链接指示器旁边点击添加，选择一个或多个触发器。

**Map element**

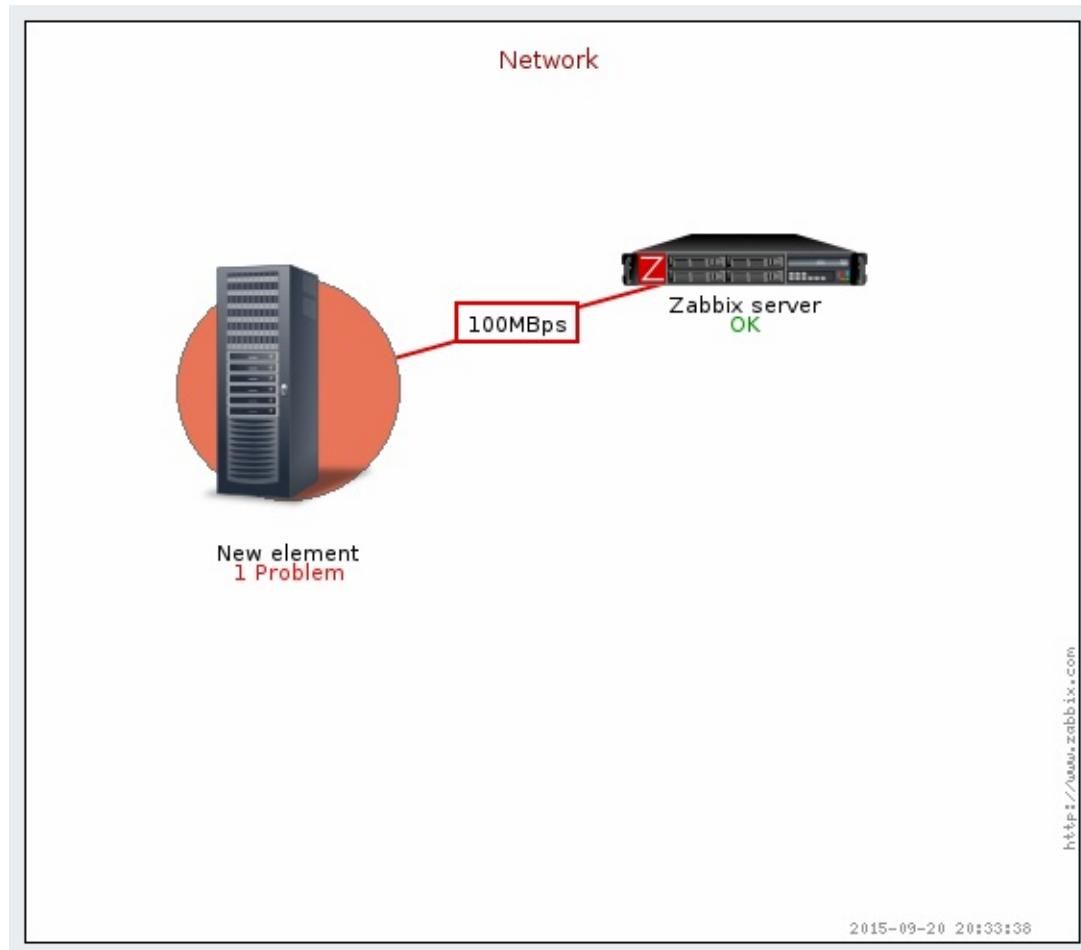
Type	<input type="button" value="Host"/>								
Label	<input type="text" value="New element"/>								
Label location	<input type="button" value="Default"/>								
Host	<input type="button" value="New host"/> <input type="button" value="Select"/>								
Application	<input type="button" value="Select"/>								
Automatic icon selection	<input type="checkbox"/>								
Icons	<table border="0"> <tr> <td>Default</td> <td><input type="button" value="Server_(96)"/></td> </tr> <tr> <td>Problem</td> <td><input type="button" value="Server_(128)"/></td> </tr> <tr> <td>Maintenance</td> <td><input type="button" value="Server_(24)"/></td> </tr> <tr> <td>Disabled</td> <td><input type="button" value="Default"/></td> </tr> </table>	Default	<input type="button" value="Server_(96)"/>	Problem	<input type="button" value="Server_(128)"/>	Maintenance	<input type="button" value="Server_(24)"/>	Disabled	<input type="button" value="Default"/>
Default	<input type="button" value="Server_(96)"/>								
Problem	<input type="button" value="Server_(128)"/>								
Maintenance	<input type="button" value="Server_(24)"/>								
Disabled	<input type="button" value="Default"/>								
Coordinates	X <input type="text" value="89"/> Y <input type="text" value="127"/>								
URLs	<table border="0"> <tr> <th>NAME</th> <th>URL</th> </tr> <tr> <td><input type="text"/></td> <td><input type="text"/></td> </tr> </table> <p>Add</p> <p><input type="button" value="Apply"/> <input type="button" value="Remove"/> <input type="button" value="Close"/></p>	NAME	URL	<input type="text"/>	<input type="text"/>				
NAME	URL								
<input type="text"/>	<input type="text"/>								
Links	<table border="0"> <thead> <tr> <th>ELEMENT NAME</th> <th>LINK INDICATORS</th> </tr> </thead> <tbody> <tr> <td>Zabbix server</td> <td>New host: Zabbix agent on New host is unreachable for 5 minutes [i]</td> </tr> </tbody> </table>	ELEMENT NAME	LINK INDICATORS	Zabbix server	New host: Zabbix agent on New host is unreachable for 5 minutes [i]				
ELEMENT NAME	LINK INDICATORS								
Zabbix server	New host: Zabbix agent on New host is unreachable for 5 minutes [i]								
Label	<input type="text" value="100MBps"/>								
Connect to	<input type="button" value="Zabbix server"/>								
Type (OK)	<input type="button" value="Bold line"/>								
Colour (OK)	<input type="color" value="#00CC00"/>								
Link indicators	<table border="0"> <thead> <tr> <th>TRIGGER</th> <th>TYPE</th> <th>COL</th> </tr> </thead> <tbody> <tr> <td>New host: Zabbix agent on New host is unreachable for 5 minutes</td> <td><input type="button" value="Line"/></td> <td><input type="color" value="red"/> D</td> </tr> </tbody> </table> <p>Add</p> <p><input type="button" value="Apply"/> <input type="button" value="Remove"/> <input type="button" value="Close"/></p>	TRIGGER	TYPE	COL	New host: Zabbix agent on New host is unreachable for 5 minutes	<input type="button" value="Line"/>	<input type="color" value="red"/> D		
TRIGGER	TYPE	COL							
New host: Zabbix agent on New host is unreachable for 5 minutes	<input type="button" value="Line"/>	<input type="color" value="red"/> D							

添加的触发器可以在链接指示器列表中看到。

你可以直接从列表中为每个触发器设置链接的颜色和类型，完成后，点击应用，关闭当前的“批量更新元素”后，点击更新以保存拓扑图的更改。

显示

在*Monitoring → Maps*中，如果触发器进入“问题”状态，那么在链接上就会显示相应的颜色。



如果多个触发器进入“问题”状态，则严重程度最高的将决定链接的颜色和样式。如果具有相同严重性的多个触发器都分配到同一个拓扑图链接，那么ID最最低的触发器将优先显示。

3 聚合图形

概述

在Zabbix的聚合图形页面上，你可以把从各种来源的信息聚合到一起，以便在单个屏幕上快速查看。基本上聚合图形是一个表格，你选择每个表格中有多少个单元格以及单元格中要显示的元素。 可以显示以下元素：

- 简单图形；
- 简单图形原型；
- 用户定义的自定义图形；
- 自定义图形原型；
- 拓扑图；
- 其他聚合图形；
- 纯文本信息；
- 服务器信息（概述）；
- 主机信息（概述）；
- 触发器信息（概述）；
- 主机/主机组问题（触发器的状态）；
- 系统状态；
- 数据概述；
- 时钟；
- 事件历史；
- 最近的动作历史；
- URL（从另一个位置获取的数据）。

聚合图形在监测中 → [聚合图形](#)中进行管理，可以对其进行配置，管理和查看。它们也可以添加到监测中 → [仪表盘](#)的“收藏夹”部分。

要配置屏幕，你必须首先通过定义其常规属性来创建它，然后在单元格中添加单个元素。

Zabbix 的所有用户（包括非管理员用户）都可以创建聚合图形。聚合图形有一个所有者 - 创建它们的用户。

聚合图形可以被公开或私有。所有用户都可以看到公开的聚合图形。

私有的聚合图形只对其所有者可见。所有者可以向其他用户和用户组共享私有的聚合图形。常规（非超级管理员）用

户只能与他们所属的组和用户共享。只要他们拥有所有聚合图形中元素的读取权限，私有屏幕将对其所有者和聚合图形共享的用户可见。只要管理员级用户对所有聚合图形中的元素都具有读取权限，就可以查看和编辑私有聚合图形，而不管所有者或所有者属于共享用户列表。

对于公开和私有的聚合图形，用户必须至少具有所有聚合图形中元素的读取权限才能看到屏幕。要向聚合图形中添加元素，用户还必须至少具有对其的读取权限。

创建一个聚合图形

按照以下步骤创建聚合图形：

- 在 Zabbix 前端跳转到 监测中 → 聚合图形；
- 跳转到所有聚合图形页面；
- 点击创建聚合图形。

聚合图形标签页包含常规聚合图形属性：



给你的屏幕一个唯一的名称，并设置列数（垂直单元格）和行数（水平单元格）。

分享标签页包含聚合图形类型以及专用聚合图形的共享选项（用户组，用户）：



参数	描述
所有者	选择聚合图形的所有者。
类型	选择聚合图形的类型：私有 - 聚合图形只有对选定的用户组和用户可见。公开 - 聚合图形对所有人可见。
用户组共享列表	选择可访问聚合图形的用户组。你可以赋予只读或读写权限。
用户共享列表	选择可访问聚合图形的用户。你可以赋予只读或读写权限。

点击添加保存聚合图形。

添加元素

要向屏幕添加元素，请单击列表中的聚合图形名称旁边的构造函数。

在打开的新页面上，您可能只会看到名为 [更改](#) 的链接。单击这个链接将打开一个新页面，您可以在此页面设置每个单元格中显示的内容。

在现有的聚合图形上，单击现有元素以打开表单，您可以设置要显示的内容。



聚合图形元素的属性：

参数	描述
资源	在单元格中显示的信息：动作日志 - 近期的动作日志时钟 - 数字或模拟时钟显示当前服务器或本地时间。数据概述 - 一组主机的最新数据图形 - 单一的自定义图形图形原型 - 自动发现 (LLD) 规则的自定义图形。事件历史 - 最新的事件。主机组问题 - 由主机组过滤的触发器状态（包括不含事件的触发器，从Zabbix 2.2开始）主机信息 - 高级主机的相关信息主机问题 - 由主机过滤的触发器的状态（包括不含事件的触发器，自Zabbix 2.2起），从Zabbix 2.2开始）拓扑图 - 单一的拓扑图纯文本 - 纯文本数据聚合图形 - 聚合图形（一个聚合图形内可能包含其他聚合图形）简单图形 - 单一的简单图形简单图形原型 - 基于自动发现 (LLD) 生成监控项的简单图形。Zabbix的状态 - 关于 Zabbix 服务器的高级信息系统状态 - 展示系统状态（类似于仪表盘）触发器信息 - 高级触发器的相关信息。触发器概述 - s主机组的触发器状态。URL - 包含来自外部资源的内容。
水平对齐	可能的值：居中**左侧**右侧
垂直对齐	可能的值：居中**顶部**底部
列跨度	将单元格扩展到多个列，与HTML列跨越的方式相同。
行宽度	将单元格扩展到多行，与HTML行跨越的方式相同。

注意表格两边的“ + ”和“ - ”控件。

点击表格上方的“ + ”将会添加一列。 点击表格下方的“ - ”将删除一列。

点击表格左侧的“ + ”将会添加一行。 点击表格右侧的“ - ”将会删除一行。

如果图形高度设置为小于120像素，则在图形中将不会显示任何图例。

动态元素

对于某些元素，有一个额外的选项称为动态监控项。 首先选中此框并不会改变任何东西。

然而，一旦你去监测中 - >聚合图形，你可能会意识到，现在你有额外的下拉列表选择主机。 因此，你有一个聚合图形，其中一些元素显示相同的信息，而其他元素根据当前选择的主机显示信息。

这样做的好处是，你不需要创建额外的屏幕，只因为你希望看到包含来自各种主机的数据的相同图形。

动态监控项选项适用于以下几个聚合图形选项：

- 图形（自定义图形）；
- 图形原型；
- 简单图形；
- 简单图形原型；

- 纯文本；
- URL。

点击一个动态图形就可以查看它的全部师徒；尽管目前仅使用默认主机支持的自定义图形和图形原型（即主机在下拉列表中未选中）。当下拉菜单中选择另一个主机时，使用该主机的监控项数据创建动态图，并且生成的图形不可点击。

动态 URL 元素不会显示在监测中 - > 聚合图形中，除非选择了主机。如果没有选定的主机，则只显示“无主机选择”的消息。

1 聚合图形元素

概述

本章节列出了可用的 [聚合图形](#) 元素，并提供聚合图形元素配置的详细信息。

1 动作日志

在动作日志元素中，您可以显示动作操作（通知，远程命令）的详细信息。它从管理 -> 审计中复制信息。

在“资源”中选择动作日志进行配置：



你可以设置以下特定选项：

展示行数	设置在聚合图形单元格中显示多少行动作日志。
以其排序	以其排序：时间（降序或升序）类型（降序或升序）状态（降序或升序）接收者（降序或升序）。

2 时钟

在时钟元素中，你可以展示本地、服务器或特定的主机时间。

在“资源”中选择时钟进行配置：



你可以设置以下特定选项：

时间类型	选择本地、服务器或特定主机时间。
监控项	选择显示时间的监控项。显示主机时间，可以使用 <code>system.localtime[local]</code> 监控项。这个监控项必须存在于主机中。\\此字段仅在选择主机时间时是可用的。
宽	选择时钟的宽度。
高	选择时钟的高度。

3 数据概述

在数据概述元素中，你可以展示一组主机的最新数据，它从监测中 -> 概述复制信息。（当在“概述”中选择数据类型时）。

在“资源”中选择数据概述进行配置：



你可以设置以下特定选项：

主机组	选择主机组。
应用集	输入应用集的名称。
主机位置	选择主机位置 - 左侧或顶部。

4 图形

在图形元素中，你可以展示单一的自定义图形。

在“资源”中选择图形进行配置：



你可以设置以下特定选项：

图形	选择要展示的图形。
宽	选择展示图形的宽度。
高	选择展示图形的高度。
动态监控项	设置展示不同数据的图形，取决于选中的主机中。

5 图形原型

在图形原型元素中，你可以展示从自动发现（LLD）规则的自定义图形。

在“资源”中选择图形原型进行配置：



你可以设置以下特定选项：

图形原型	选择展示的图形原型。
最大列数	在聚合图形单元格中应该展示多少列生成的图形。当有多个自动发现（LLD）生成的图形时，此选项是有用的。
宽	选择图形的宽度。
高	选择图形的高度。
动态监控系	设置展示不同数据的图形，取决于选中的主机中。

6 事件历史

在事件历史元素中，你可以展示最新的事件。

在“资源”中选择事件历史进行配置：



你可以设置以下特定选项：

展示行数	设置在聚合图形单元格中展示多少行事件历史。
------	-----------------------

7 主机组问题

在主机组问题元素中，你可以展示通过主机组过滤的触发器状态。这个类似于从[仪表盘](#)展示的最近的20个问题。

在“资源”中选择主机组问题进行配置：



你可以设置以下特定选项：

主机组	选择主机组。
展示行数	设置在聚合图形单元格中展示多少行触发器状态。
触发器以其排序	从下拉菜单中的最后改变（降序）、严重等级（降序）或主机（升序）选择触发器的排序方式。

8 主机信息

在主机信息元素中，你可以展示关于主机可用性的高级信息。

在“资源”中选择“主机信息”进行配置：



你可以设置以下特定选项：

主机组	选择主机组。
风格	选择垂直或水平显示。

9 主机问题

在主机问题元素中，你可以展示通过主机过滤的触发器状态。这个类似于从[仪表盘](#)展示的最近的20个问题。

在“资源”中选择“主机问题”进行配置：



你可以设置以下特定选项：

主机	选择主机。
展示行数	设置在聚合图形单元格中展示多少行触发器状态。
触发器以其排序	从下拉菜单中的最后改变（降序）、严重等级（降序）或主机（升序）选择触发器的排序方式。

10 拓扑图

在拓扑图元素中，你可以展示配置好的网络拓扑图。

在“资源”中选择拓扑图进行配置：



你可以设置以下特定选项：

拓扑图	选择要展示的拓扑图。
-----	------------

11 纯文本

在纯文本元素中，你可以在纯文本中展示最近的监控项数据。

在“资源”中选择纯文本进行配置：



你可以设置以下特定选项：

监控项	选择一个监控项。
展示行数	设置在聚合图形单元格中展示多少行最近数据。
通过文本还是 <code>HTML</code> 展示	设置通过文本还是 <code>HTML</code> 展示。
动态监控项	设置展示不同数据的图形，取决于选中的主机中。

12 聚合图形

在聚合图形元素中，你可以展示其他聚合图形。一个聚合图形里可能包含其他聚合图形。

在“资源”中选择聚合图形进行配置：



你可以设置以下选项：

聚合图形	选择要展示的聚合图形。
------	-------------

13 简单图形

在简单图形元素中你可以展示单个简单图形。

在“资源”中选择简单图形进行配置：



你可以设置以下特定选项：

监控项	为简单图形选择监控项。
宽	选择图形的宽度。
高度	选择图形的高度。
动态监控项	设置展示不同数据的图形，取决于选中的主机中。

14 简单图形原型

在简单图形原型元素中，你可以展示基于通过自动发现生成监控项的简单图形。

在“资源”中选择简单图形原型进行配置：



你可以设置以下特定选项：

监控项原型	为简单图形选择监控项原型。
最大列数	在聚合图形单元格中应该展示多少列生成的图形。在有许多自动发现 (LLD) 生成的图形是，此选项是有用的。
宽	选择图形的宽度。
高	选择图形的高度。
动态监控项	设置展示不同数据的图形，取决于选中的主机中。

15 Zabbix状态

在Zabbix状态元素中，你可以展示 Zabbix 和 Zabbix 服务器的高级信息。

在“资源”中选择Zabbix 信息进行配置。



16 系统状态

在这个元素中，你可以展示类似于仪表盘控件中系统状态。

在“资源”中选择系统状态进行配置：



17 触发器信息

在触发器信息元素中，你可以展示关于触发器状态的高级信息。

在“资源”中选择触发器信息进行配置：



你可以设置以下特定选项：

主机组	选择一个或多个主机组。
风格	选择垂直或水平显示。

18 触发器概述

在触发器概述元素中，你可以为一组主机展示触发器状态。从监测中 → 概述中复制信息（当在概述中选择触发器类型）。

在“资源”中选择触发器概述进行配置：



你可以设置以下特定选项：

主机组	选择主机组。
应用集	输入应用集名称。
主机位置	选择主机的位置 - 左侧或顶部。

19 URL

在 URL 元素中，你可以从外部资源展示 URL 内容。

在“资源”中选择URL进行配置：



你可以设置以下特定选项：

<u>URL</u>	输入要展示的 <u>URL</u> 。
宽	窗口展示的宽度。
高	串口展示的高度。
动态监控项	设置展示不同 <u>URL</u> 内容，取决于选中的主机中。

如果通过 HTPPS 访问 Zabbix 前端，那么浏览器可能不会加载包含在聚合图形中的 HTTP 页面（当使用 URL 元素时）。

4 幻灯片演示

概述

在幻灯片演示中，你可以配置多个[聚合图形](#)以设定的间隔逐个显示。

有时候，你可能想要在一些配置好的聚合图形之间进行切换。虽然这样可以手动完成，但这样做一次两次后就会变得非常乏味。因此，就可以通过幻灯片演示来自动完成此项工作。

Zabbix 的所有用户（包括非管理员用户）都可以创建幻灯片演示。幻灯片演示拥有所有者 - 这个所有者就是哪个用户创建了它们。

幻灯片演示可以选择公开或私有。公开的幻灯片演示允许所有用户访问，然而，用户必须拥有对幻灯片演示的元素（聚合图形）读取权限才能看到。要向幻灯片演示添加聚合图形，用户仍必须拥有对聚合图形的读取权限。

私有的幻灯片演示仅对其所有者可见。私有的幻灯片演示可以通过其所有者分享给其他用户或用户组。常规（非超级管理员）用户只能与他们所属的组和用户共享。只要他们拥有对所有包含的聚合图形的读取权限，其所有者和用户就可以看到私有幻灯片演示。管理员级别的用户只要拥有对所有包含的聚合图形的读取权限，就可以看到并编辑私人幻灯片演示，无论是所有者还是属于共享用户列表。

配置

通过以下步骤创建一个幻灯片演示：

- 在 Zabbix 前端页面跳转到 监测中→ 聚合图形；
- 在右上角的下拉菜单选择幻灯片演示；
- 跳转到所有的幻灯片演示；
- 点击 创建幻灯片播放。

“幻灯片”标签也包含常规幻灯片演示属性：



参数	描述
所有者	选择幻灯片演示的所有者。指定所有者时强制性的。
名称	幻灯片演示的唯一名称。
默认延迟 (秒)	默认情况下多长时间展示一个聚合图形，然后跳转到下一个聚合图形。以秒为单位。
幻灯片	要选择的聚合图形列表。点击添加选择聚合图形。聚合图形上的向上/向下箭头可以按照显示的顺序上下拖动聚合图形。如果在幻灯片上只显示一个聚合图形，那么直接创建包含该图形的聚合图形添加到此即可。
聚合图形	聚合图形名称。

延迟	聚合图形显示多长时间的自定义值，以秒为单位。如果此项设置为 0，那么将使用默认延迟。
动作	点击移除从幻灯片演示中移除该聚合图形。

本示例中的幻灯片演示由以下顺序显示的两个聚合图形组成：

Zabbix server → 展示30秒 → Zabbix server2 → 展示15秒 → Zabbix server → 展示30秒 → Zabbix server2 → 依次循环。

“分享”标签页包含幻灯片演示的类型以及私有幻灯片的共享选项（用户组和用户）：



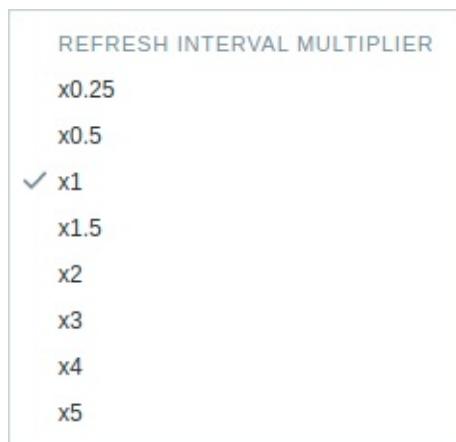
参数	描述
类型	选择幻灯片演示的类型：私有 - 幻灯片演示仅对选定的用户组和用户可见。公开 - 幻灯片演示对所有人可见。
用户组共享的列表	选择幻灯片可访问的用户组。你可以设置只读或读写权限。
用户共享的列表	选择幻灯片可访问的用户。你可以设置只读或读写权限。

点击添加保存幻灯片演示。

延迟

可以在监测中 - >聚合图形中查看准备就绪的幻灯片，然后从下拉列表中选择幻灯片演示，并单击幻灯片演示名称。

使用下拉列表旁边的菜单选项，您可以通过选择刷新时间倍数来加快或减慢显示速度：



如果延迟时间少于5秒（通过输入延迟小于5秒或使用滑动延迟倍数），则将使用5秒的最小延迟。

7 模板

概述

A template is a set of entities that can be conveniently applied to multiple hosts. 模板是可以方便地应用于多个主机的一组实体。

The entities may be: 实体可以是:

- items
- 监控项
- triggers
- 触发器
- graphs
- 图形
- applications
- 应用
- screens (*since Zabbix 2.0*)
- 聚合图形 (自Zabbix 2.0起)
- low-level discovery rules (*since Zabbix 2.0*)
- 自动发现规则 (自Zabbix 2.0起)
- web scenarios (*since Zabbix 2.2*)
- web场景 (自Zabbix 2.0起)

As many hosts in real life are identical or fairly similar so it naturally follows that the set of entities (items, triggers, graphs,...) you have created for one host, may be useful for many. Of course, you could copy them to each new host, but that would be a lot of manual work. Instead, with templates you can copy them to one template and then apply the template to as many hosts as needed. 由于现实生活中的许多主机是相同或类似的，所以，您为一个主机创建的一组实体（项目，触发器，图形，...）可能对许多人有用。当然，您可以将它们复制到每个新的主机上，但需要费很大功夫。相反，使用模板，您可以将它们复制到一个模板，然后根据需要将模板应用于尽可能多的主机。

When a template is linked to a host, all entities (items, triggers, graphs,...) of the template are added to the host. Templates are assigned to each individual host directly (and not to a host group). 当模板链接到主机时，模板的所有实体（项目，触发器，图形，...）都将添加到主机。模板直接分配给每个单独的主机（而不是主机组）。

Templates are often used to group entities for particular services or applications (like Apache, MySQL, PostgreSQL, Postfix...) and then applied to hosts running those services. 模板通常用于为特定服务或应用程序（如Apache, MySQL, PostgreSQL, Postfix ...）分组实体，然后应用于运行这些服务的主机。

Another benefit of using templates is when something has to be changed for all the hosts. Changing something on the template level once will propagate the change to all the linked hosts. 使用模板的另一个好处是当所有主机都需要更改时。只需要在模板上更改某些内容将会将更改应用到所有链接的主机。

Thus, the use of templates is an excellent way of reducing one's workload and streamlining the Zabbix configuration. 因此，使用模板是减少工作量并简化Zabbix配置的好方法。

Proceed to [creating and configuring a template](#). 继续[创建和配置模板](#)。

1 配置模板

概述

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

配置模板需要首先通过定义一些参数来创建模板，然后添加实体（项目，触发器，图形等）。

创建模板

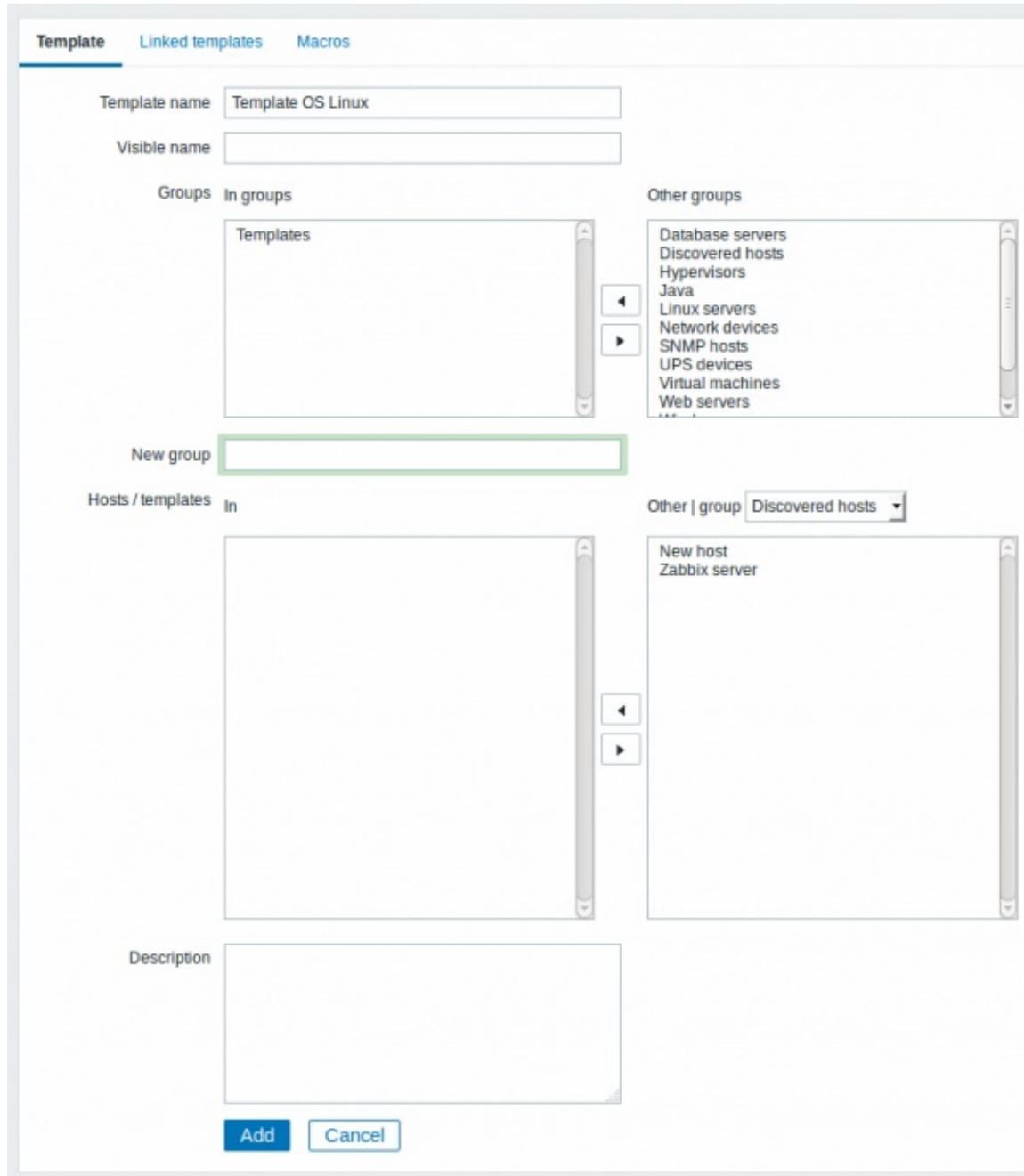
To create a template, do the following:

要创建模板，请执行以下操作：

- 转到配置→模板
- 点击创建模板
- 编辑模板属性

The **Template** tab contains general template attributes.

模板选项卡包含常规模板属性。



模板属性：

参数	描述
模板名称	唯一的模板名称。
可见名称	如果你设置了这个名字，那么它将是列表，地图等中可见的。
群组	模板所属的主机/模板组。
新的群组	可以创建一个新组来保存模板。\\如果为空忽略。
主机/模板	应用模板的主机/模板列表。
描述	输入模板说明。

The **Linked templates** tab allows you to link one or more “nested” templates to this template. All entities (items, triggers, graphs etc.) will be inherited from the linked templates.

链接的模板选项卡允许您将一个或多个“嵌套”模板链接到此模板。所有实体（项目，触发器，图表等）将从链接的模板继承。

To link a new template, start typing in the *Link new templates* field until a list of templates corresponding to the entered letter(s) appear. Scroll down to select. When all templates to be linked are selected, click on *Add*.

要链接新的模板，请开始输入链接指示器字段，直到出现与输入的字母对应的模板列表。向下滚动选择。当选择要链接的所有模板时，单击添加。

To unlink a template, use one of the two options in the *Linked templates* block:

要取消链接模板，请使用链接的模板模块中的两个选项之一：

- 取消链接 - 取消链接模板，但保留其项目，触发器和图形
- 取消链接并清理 - 取消链接模板并删除其所有项目，触发器和图形

The **Macros** tab allows you to define template-level [user macros](#). You may also view here macros from linked templates and global macros if you select the *Inherited and template macros* option. That is where all defined user macros for the template are displayed with the value they resolve to as well as their origin.

宏选项卡允许您定义模板级[用户宏](#)。如果选择了继承模板的宏选项，则还可以从链接的模板和全局宏中查看宏。在这里，模板的所有定义的用户宏都显示了它们所决定的值以及它们的起源。

MACRO	EFFECTIVE VALUE	TEMPLATE VALUE	GLOBAL VALUE
[\$NESTED_TEMPLATE_MACRO]	→ 53689	Change ← Template2: "53689"	
[\$SNMP_COMMUNITY]	→ public	Change	← "public"
[\$TEMPLATE_MACRO]	→ 25864	Remove	

For convenience, links to respective templates and global macro configuration are provided. It is also possible to edit a nested template/global macro on the template level, effectively creating a copy of the macro on the template.

为方便起见，提供了相应模板和全局宏配置的链接。也可以在模板级别上编辑嵌套模板/全局宏，有效地创建模板上宏的副本。

按钮：

Add	添加模板。添加的模板应该出现在列表中。
Update	更新现有模板的属性。

Clone	根据当前模板的属性创建另一个模板，包括从链接模板继承的实体（项目，触发器等）
Full clone	基于当前模板的属性创建另一个模板，包括从链接的模板继承并直接附加到当前模板的实体（项目，触发器等）。
Delete	删除模板；模板（项目，触发器等）的实体与链接的主机保留。
Delete and clear	从链接的主机中删除模板及其所有实体。
Cancel	取消编辑模板属性。

With a template created, it is time to add some entities to it.

创建一个模板，开始添加一些实体。

Items have to be added to a template first. Triggers and graphs cannot be added without the corresponding item.

项目必须首先添加到模板中。如果没有相应的项目，则无法添加触发器和图形。

添加监控项，触发器，图形

To add items to the template, do the following:

要向模板添监控项，请执行以下操作：

- 转到配置→主机（或模板）
- Click on *Items* in the row of the required host/template
- 单击所需主机/模板行中的监控项
- Mark the checkboxes of items you want add to the template
- 标记要添加到模板的项目的复选框
- Click on *Copy* below the item list
- 点击项目列表下面的复制
- Select the template (or group of templates) the items should be copied to and click on *Copy*
- 选择要复制的项目的模板（或模板组），然后单击复制

All the selected items should be copied to the template.

所有选定的监控项都应该被复制到模板中。

Adding triggers and graphs is done in similar fashion (from the list of triggers and

graphs respectively), again, keeping in mind that they can only be added if the required items are added first.

添加触发器和图形以类似的方式完成（分别从触发器和图形列表），请记住，只有在首先添加所需项目时，才能添加它们。

添加聚合图形

To add screens to a template in *Configuration → Templates*, do the following:

要在配置→模板中向屏幕添加聚合图形，请执行以下操作：

- Click on *Screens* in the row of the template
- 点击模板行中的聚合图形
- Configure a screen following the usual method of [configuring screens](#)
- 按照通常的配置聚合图形的方法[配置聚合图形](#)

The elements that can be included in a template screen are: simple graph, custom graph, clock, plain text, [URL](#).

可以包含在模板聚合图形中的元素有：简单图形，自定义图形，时钟，纯文本，URL。

配置自动发现规则

See the [low-level discovery](#) section of the manual.

请参阅手册的[自动发现](#)部分。

添加Web场景

To add web scenarios to a template in *Configuration → Templates*, do the following:

要将配置→模板中的 Web 场景添加到模板，请执行以下操作：

- 点击模板行中的Web
- Configure a web scenario following the usual method of [configuring web scenarios](#)
- 按照通常的Web方案配置方式[配置Web场景](#)

2 链接/取消链接

概述

Linking is a process whereby templates are applied to hosts, whereas unlinking removes the association with the template from a host.

链接是将模板应用于主机的过程，而取消链接将从主机中删除与模板的关联。

模板直接链接到各个主机，而不是主机组。只需将模板添加到主机组就不会链接到主机组。主机组仅用于主机和模板的逻辑分组。

链接模板

要将模板链接到主机，请执行以下操作：

- 转到配置→主机
- 单击所需的主机并切换到模板选项卡
- 点击链接指示器旁边的选择
- 在弹出窗口中选择一个或多个模板
- 单击主机属性窗体中的添加/更新

The host will now have all the entities (items, triggers, graphs, etc) of the template.

主机现在将拥有模板的所有实体（项目，触发器，图形等）。

如果在那些模板中有相同监控项的项，如链接到相同的主机将失败。并且作为触发器和图形使用项目，如果使用相同的项目键，它们也不能从多个模板链接到单个主机。

当从模板添加实体（监控项，触发器，图表等）时：

- 主机上以前存在的相同实体被更新为模板的实体
- 添加模板中的实体
- any directly linked entities that, prior to template linkage, existed only on the host remain untouched
- 在模板连接之前，只存在于主机上的任何直接链接的实体保持不变

In the lists, all entities from the template now are prefixed by the template name, indicating that these belong to the particular template. The template name itself (in grey text) is a link allowing to access the list of those entities on the template level.

在列表中，模板中的所有实体都以模板名称为前缀，表示这些属于特定模板。模板名称本身（灰色文本）是允许访问模板级别上这些实体列表的链接。

If some entity (item, trigger, graph etc.) is not prefixed by the template name, it means that it existed on the host before and was not added by the template.

如果某个实体（监控项，触发器，图表等）未被模板名称前缀，则表示该模板存在于主机之前，并未被模板添加。

实体唯一性标准

When adding entities (items, triggers, graphs etc.) from a template it is important to know what of those entities already exist on the host and need to be updated and what entities differ. The uniqueness criteria for deciding upon the sameness/difference are:

从模板中添加实体（监控项，触发器，图表等）时，重要的是要知道这些实体已经存在于主机上并需要更新，哪些实体有所不同。决定同一性/差异的唯一性标准是：

- for items - the item key
• 用于监控项 - 项目键
- for triggers - trigger name and expression
• 用于触发器 - 触发器名称和表达式
- for custom graphs - graph name and its items
• 用于自定义图形 - 图形名称及其项目
- for applications - application name
• 用于应用集 - 应用集名称

将模板链接到多个主机

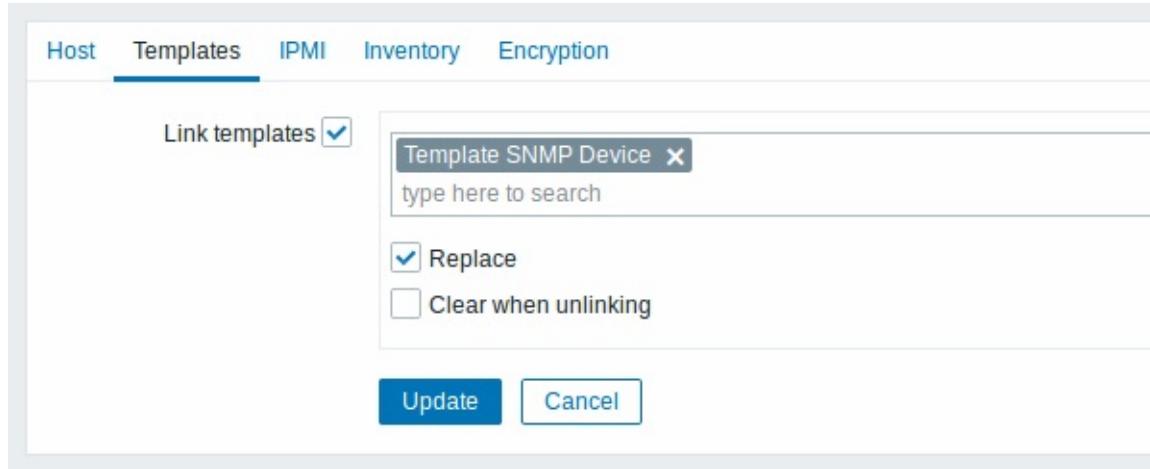
There are some ways of mass-applying templates (to many hosts at once):有一些批量应用模板的方法（对许多主机一次搞定）：

- To link a template to many hosts, in *Configuration → Templates*, click on the template, then select hosts from the respective group in the *Other* box, click on « and update the template.
- 要将模板链接到许多主机，请在配置→模板中单击模板，然后从其他主机框中的相应组中选择主机，然后单击“更新模板。”

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

反之亦然，如果您在“收件箱”中选择链接的主机，请单击»并更新模板，从该模板中取消链接模板（主机仍将从模板继承监控项，触发器，图表等）。

- To update template linkage of many hosts, in *Configuration* → *Hosts* select some hosts by marking their checkboxes, then click on **Mass update** below the list and then in the *Templates* tab select to link additional templates:
- 要更新许多主机的模板链接，在配置→主机中通过标记其复选框来选择一些主机，然后单击列表下方的**批量更新**，然后在模板选项卡中选择链接其他模板：



Select *Link templates* and start typing the template name in the auto-complete field until a dropdown appears offering the matching templates. Just scroll down to select the template to link.

选择链接模板，并在自动完成字段中开始输入模板名称，直到出现一个提供匹配模板的下拉列表。只需向下滚动即可选择要链接的模板。

The *Replace* option will allow to link a new template while unlinking any template that was linked to the hosts before. The *Clear when unlinking* option will allow to not only unlink any previously linked templates, but also remove all elements inherited from them (items, triggers, etc.).

在替换选项将允许同时取消关联之前被连接到主机的任何模板链接一个新的模板。在取消链接并清理选项将允许不仅取消链接任何以前链接的模板，但也从中移除（监控项，触发器等）继承了所有元素。

Zabbix提供了大量预定义的模板。您可以使用这些作为参考，但请注意在生产中不改变使用它们，因为它们可能包含太多监控项，并且太频繁地轮询数据。如果你喜欢它们，确保它们以适和你的需求。

编辑链接实体

If you try to edit an item or trigger that was linked from the template, you may realize that many key options are disabled for editing. This makes sense as the idea of templates is that things are edited in one-touch manner on the template level. However, you still can, for example, enable/disable an item on the individual host and set the update interval, history length and some other parameters.

如果你尝试编辑从模板链接的监控项或触发器，你可能会意识到许多关键选项被禁用以进行编辑。这是有道理的，因为模板的想法是在模板级别上以一触式方式编辑事物。但是，你仍然可以启用/禁用单个主机上的监控项，并设置更新间隔，历史长度和其他一些参数。

If you want to edit the entity fully, you have to edit it on the template level (template level shortcut is displayed in the form name), keeping in mind that these changes will affect all hosts that have this template linked to them.

如果要完全编辑实体，则必须在模板级别进行编辑（模板级快捷方式以表单名称显示），请注意，这些更改将影响所有与此模板链接的主机。

取消链接模板

要从主机中取消链接模板，请执行以下操作：

- 转到配置→主机
- Click on the required host and switch to the *Templates* tab
- 单击所需的主机并切换到模板选项卡
- 单击取消链接或取消链接并清理模板旁边以取消链接
- 单击主机属性窗体中的更新

Choosing the *Unlink* option will simply remove association with the template, while leaving all its entities (items, triggers, graphs etc.) with the host.

选择取消链接选项将简单地删除与模板的关联，同时将其所有实体（监控项，触发器，图形等）与主机保持一致。

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

选择取消链接并清理选项将删除与模板及其所有实体（监控项，触发器，图表等）的关联。

3 嵌套

概述

Nesting is a way of one template encompassing one or more other templates.

嵌套是一种包含一个或多个其他模板的方式

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

因为将各个模板实体分离出来用于各种服务，应用程序等都是有意义的，所以您可能会得到相当多的模板，所有这些模板都可能需要链接到相当多的主机。为了简化图片，可以在一个“嵌套”模板中将一些模板链接在一起。

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

嵌套的好处在于，您仅需将一个模板链接到主机，并且主机将自动继承链接的模板的所有实体。

配置嵌套模板

如果要链接一些模板，首先可以使用现有模板或新模板，然后：

- 打开模板属性窗体
- Look for the *Linked templates* tab
- 查找链接的模板选项卡
- Click on *Select* to select templates in the popup window
- 单击选择以在弹出窗口中选择模板
- Click on *Add* to list selected templates
- 单击添加以列出所选模板
- 单击模板属性窗体中的添加/更新

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

现在，模板应该具有所链接的模板的所有实体（监控项，触发器，自定义图表等）

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

要取消链接任何链接的模板，以相同的形式使用取消链接或取消链接并清理按钮，然后单击更新。

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

选择取消链接选项将简单地删除与其他模板的关联，而不删除其所有实体（监控项，触发器，图形等）

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

选择取消链接并清理选项将删除与其他模板及其所有实体（监控项，触发器，图表等）的关联。

权限问题

- You may have a setup where an Admin level user has *Read-write* access to some Template A while not having *Read-write* access to Template B that holds Template A in a nested setup. In this case, an item created on Template A, while inherited by the hosts of Template A, **will not** be inherited by the hosts of Template B. Thus, creating a trigger for such an item will fail altogether, because of missing corresponding items on hosts of Template B.
- 您可以设置一个设置，其中一个管理级用户对某个模板A具有读写访问，而不具有读写访问模板B，该模板B在一个嵌套的设置中保存模板A。在这种情况下，在模板A上创建的项，而由模板A的宿主继承，将不会被模板B的主机继承，因此，创建这样一个项的触发器将完全失败，因为在模板B的主机上缺少相应的项。

8 事件通知

概述

假设我们已经配置了一些项目和触发器，并且由于触发器改变状态，现在正在发生一些事件，之后要考虑的就是 actions。

首先，我们不想一直盯着触发器或事件列表。最好是在发生比较严重的事情（如异常）时，接收到通知。另外，当发生问题时，我们希望看到所有有关的人都知道。

这就是为什么发送通知是Zabbix提供的主要操作之一， 可以定义在特定事件中通知到谁以及什么时间通知。

为了能够发送和接收Zabbix的通知，您必须：

- 定义一些[media](#)
- 配置[action](#) 向指定的media发送消息

Actions由*conditions* 和 *operations*组成。基本上，当条件满足时，执行操作。两个主要操作是发送消息（通知）并执行远程命令。

对于发现和自动注册创建的事件，可以使用一些其他操作。 包括添加或删除主机，链接模板等

1 Media类型

概述

媒介是用于在Zabbix中发送通知和警报的传送通道。

您可以配置多种媒介类型：

- [E-mail](#)
- [SMS](#)
- [Jabber](#)
- [EZ Texting](#)
- [自定义警报提示](#)

1 E-mail

概述

要将电子邮件配置为邮件的传递通道，您需要将电子邮件配置为媒介类型，并为用户分配特定的地址。

配置

要将电子邮件配置为媒介类型：

- 在 管理 -> 媒体类型中设置
- 点击创建媒介类型（或者点击预定义媒介类型的列表中的*E-mail*）。

Media type Options

Name	Email
Type	Email
SMTP server	mail.company.com
SMTP server port	25
SMTP helo	company.com
SMTP email	Zabbix info <zabbix@company.com>
Connection security	None STARTTLS SSL/TLS
SSL verify peer	<input type="checkbox"/>
SSL verify host	<input type="checkbox"/>
Authentication	None Username and password
Username	[empty]
Password	[empty]
Enabled	<input checked="" type="checkbox"/>
Add Cancel	

媒介类型属性：

参数	说明
Name	媒介类型的名称
Type	选择 <i>Email</i>
SMTP server	设置SMTP服务器来处理传出的消息。

<i>SMTP server port</i>	设置SMTP服务器端口来处理传出的消息。Zabbix 3.0版本之后支持此选项。
<i>SMTP helo</i>	设置正确的SMTP helo值，通常是域名。
<i>SMTP email</i>	此处输入的地址将被用作发送消息的 From 地址。在Zabbix-HQ < >中添加发件人显示名称（如“Zabbix-HQ”）Zabbix 2.2版本之后，支持实际的电子邮件地址。与RFC 5322允许的相比，Zabbix电子邮件中的显示名称有一些限制，如示例所示：有效的例子： （只有电子邮件地址，不需要使用尖括号）Zabbix HQ < >（尖括号中是显示名称和电子邮件地址）无效示例：Zabbix HQ （显示名称存在，但电子邮件地址没有尖括号）"Zabbix\@\\<H(comment)Q\>" < >（虽然RFC 5322有效，但Zabbix电子邮件中不支持引用的对和注释）
<i>Connection security</i>	选择连接安全级别： None - 不要使用 CURLOPT_USE_SSL 选项 STARTTLS - 使用有 CURLUSESSLALL 值的 CURLOPT_USE_SSL 选项 SSL/TLS - 是否使用 CURLOPT_USE_SSL 是可选的。支持此选项 Zabbix 3.0之后可以。
<i>_SSL verify peer</i>	选中该复选框以验证SMTP服务器的SSL证书。“SSLCALocation”服务器配置指令的值应该放在 CURLOPT_CAPATH 中以进行证书验证设置 CURL 选项 CURLOPT_SSL_VERIFYPEER 。Zabbix 3.0之后支持此选项。
<i>SSL verify host</i>	标记该复选框以验证SMTP服务器证书的公用名称字段或主题备用名称字段是否匹配。设置 CURL 选项 CURLOPT_SSL_VERIFYHOST 。Zabbix 3.0之后支持此选项。
<i>Authentication</i>	选择认证级别： None - 没有设置 CURL 选项 Normal password - CURLOPT_LOGIN_OPTIONS 在“AUTH=PLAIN”中设置 Zabbix 3.0之后。支持此选项
<i>Username</i>	认证中使用的用户名。设置 CURLOPT_USERNAME 的值。Zabbix 3.0之后。支持此选项
<i>Password</i>	认证中使用的密码。设置 CURLOPT_PASSWORD 的值。Zabbix 3.0之后。支持此选项
<i>Enabled</i>	标记该复选框以启用媒体类型。

要使SMTP验证选项可用，Zabbix服务器应使用curl 7.20.0或更高版本的 - with-libcurl </ nowiki> [[:manual/installation/install#configurethe_sources|编译]] 选项进行编译。</note>== 用户媒介 == 要为用户分配一个特定的地址：在 //管理 -> 用户// 中进行设置 打开用户属性窗体 * 在媒介选项卡中，单击 //Add//{{:manual:config:user_email2.png|}} 用户媒介属性：^参数^说明^//Type// | 选择//Email//为媒介类型 ||//Send to// | 指定发送消息的电子邮件地址。 添加收件人显示名称(//<nowiki>Some User <[\[email protected\]](#)> 如上截图中的“Some User”)以及Zabbix 2.2 版本之后支持的实际电子邮件地址。请参阅媒体类型属性 [SMTP电子邮件](#) 描述中显示名称和电子邮件地址的示例和限制。
 ||When active | 您可以限制邮件发送的时间，例如仅限工作日 (1-5, 09: 00-18: 00)。 \ 格式的描述请参见 [时间段格式](#) 页面 ||Use if severity | 标记您要接收通知的触发严重性的复选框。 Note 对于非触发事件，使用默认严重性 ('未分类')，因此如果要接收非触发事件的通知，请将其保留。 ||Status | 用户媒介的状态 **Enabled** - 正在使用. **Disabled** - 没有被使用. | _

2 SMS

概述

Zabbix支持使用连接到Zabbix服务器的串行端口的串行GSM调制解调器发送SMS消息。

确保：

- 串行设备的速度（在Linux下通常为`/dev/ttys0`）与GSM调制解调器的速度相匹配。 Zabbix没有设置串行链路的速度。 它使用默认设置。
- 'zabbix'用户对串行设备有读/写访问权。 运行命令`ls -l /dev/ttys0`来查看串口设备的当前权限。
- GSM调制解调器输入PIN码，并在电源复位后保留PIN码。 或者，您可以在SIM卡上禁用PIN。 可以通过在终端软件（如Unix minicom或Windows超级终端）中发出命令`AT+CPIN="NNNN"`（NNNN是您的PIN号，引号必须存在）来输入PIN。

Zabbix已经使用这些GSM调制解调器进行了测试：

- Siemens MC35
- Teltonika ModemCOM/G10

要将SMS配置为邮件的传送通道，还需要将SMS配置为媒体类型，并为用户输入相应的电话号码。

配置

要将SMS配置为媒介类型：

- 进入管理 - >媒介类型
- 点击 创建媒介类型（或者点击预先定义的媒介类型列表中的 `SMS`）.

媒介类型属性：

参数	说明
<code>Description</code>	媒介类型的名称
<code>Type</code>	选择 <code>SMS</code> 为媒介类型
<code>GSM modem</code>	设置GSM调制解调器的串行设备名称。

用户媒介

要为用户分配电话号码：

- 进入 管理 - >用户
- 打开用户属性窗体
- 在媒介选项卡中，单击 `Add`

用户媒介属性：

参数	说明
Type	选择SMS作为媒介类型
Send to	指定要发送消息的电话号码。
When active	您可以限制邮件发送的时间，例如仅限工作日（1-5, 09: 00-18: 00）。\格式描述请参见 时间段规则 页面
Use if severity	标记您要接收通知的触发严重性的复选框。
Status	媒介的状态 Enabled - 使用中. Disabled - 禁用.

3 Jabber

概述

Zabbix支持发送Jabber消息。

发送通知时，Zabbix首先尝试查找Jabber SRV记录，如果失败，则会使用该域的地址记录。在Jabber SRV记录中，选择具有最高优先级和最大权重的记录。如果失败，则不会尝试其他记录。

要将Jabber配置为消息的传递通道，您需要将Jabber配置为媒体类型，并为用户输入相应的地址。

配置

将Jabber配置为媒介类型：

- 进入管理 - >媒介类型
- 点击 创建媒介类型（或者在预定义的媒体类型列表中单击 *Jabber*）

媒介类型属性：

参数	说明
<i>Description</i>	媒介类型的名称
<i>Type</i>	选择 <i>Jabber</i> 为媒介类型
<i>Jabber identifier</i>	输入 <i>Jabber</i> 标识符。
<i>Password</i>	输入 <i>Jabber</i> 密码

用户媒介

要为用户分配Jabber地址：

- 进入 管理 - >用户
- 打开用户属性窗体
- 在媒介标签中，点击/Add用户媒介属性：^参数^说明^|_Type |选择 *Jabber* 作为媒介类型。||Send to |指定发送消息的地址。||When active |您可以限制邮件发送的时间，例如仅限工作日（1-5, 09: 00-18: 00）。格式说明参见[时间段规格](#) 页面||Use if severity |标记您要接收通知的触发严重性的复选框。||Status |用户媒介的状态**Enabled** - 使用中.**Disabled** - 禁用。|_

4 Ez Texting

概述

您可以使用[Zabbix技术合作伙伴](#) Ez Texting发送消息。

要将Ez Texting配置为消息的传递通道，您需要将Ez Texting配置为媒介类型，并为用户分配收件人标识。

配置

要将Ez Texting配置为媒介类型：

- 进入 管理 - >媒介类型
- 点击 创建媒介类型

Media types

Name	Ez Texting
Type	Ez Texting https://app.eztexting.com
Username	
Password	
Message text limit	USA (160 characters)
Enabled	<input checked="" type="checkbox"/>
Add Cancel	

媒介类型属性：

参数	说明
Description	媒介类型的名称
Type	选择 Ez Texting 作为类型。
Username	输入Ez Texting用户名。
Password	输入Ez Texting密码。
Message text limit	选择消息文本限制。 USA (160 characters)**Canada (136 characters)**

用户媒介

要向用户分配Ez短信收件人标识：

- 进入 管理 - >用户

- 打开用户属性窗体
- 在媒介标签中，点击*Add*

用户媒介属性：

参数	说明
Type	选择Ez Texting媒介类型
Send to	指定发送消息的收件人。
When active	您可以限制邮件发送的时间，例如仅限工作日（1-5, 09: 00-18: 00）。格式说明请参见 时间段规格
Use if severity	标记您要接收通知的触发严重性的复选框。
Status	用户媒介状态 Enabled - 使用中. Disabled - 禁用.

5 自定义警报提示

概述

如果您对发送警报的现有媒体类型不满意，则可以使用其他方式来执行此操作。 您可以创建一个将以您的方式处理通知的脚本。

警报脚本在Zabbix服务器上执行。 这些脚本位于服务器[配置文件](#)中定义的目录中**AlertScriptsPath**。

这是一个示例警报脚本：

```
1. #!/bin/bash
2.
3. to=$1
4. subject=$2
5. body=$3
6.
7. cat <<EOF | mail -s "$subject" "$to"
8. $body
9. EOF
```

从版本3.4开始，Zabbix检查执行的命令和脚本的退出代码。任何与 **0** 不同的退出代码都被视为[命令执行](#)错误。在这种情况下，Zabbix会尝试重复执行失败。

环境变量不会为脚本保留或创建，因此它们应该被明确处理。

配置

将自定义警报文本配置为媒介类型：

- 进入 管理 ->媒介类型
- 点击创建媒介类型

Name: Script

Type: Script

Script name: notification.sh

PARAMETER	ACTION
{ALERT.SENDTO}	Remove
{ALERT.SUBJECT}	Remove
{ALERT.MESSAGE}	Remove
	Add

Enabled

媒介类型属性：

参数	说明
Name	输入媒介类型的名称。
Type	选择 <i>Script</i> 作为媒介类型
Script name	输入脚本的名称。
Script parameters	向脚本添加命令行参数。{ALERT.SENDTO}, {ALERT.SUBJECT} 和 {ALERT.MESSAGE} 在脚本参数中是支持的Zabbix 3.0支持自定义脚本参数。

用户媒介

要为用户分配自定义警报提示符：

- 进入 管理 -> 用户
- 打开用户属性窗体
- 在媒介选项卡中，单击 *Add*

用户媒介属性：

参数	说明
Type	选择自定义的alertcripts媒介类型。
Send to	指定收件人接收警报。
When active	您可以限制执行警示标记的时间，例如，仅限工作日（1-5, 09: 00-18: 00）。\格式说明 参见 时间段规格 页面
Use if severity	标记要激活警示标记的触发严重性的复选框。
Status	用户媒介的状态 Enabled - 使用中. Disabled - 禁用.

2 Actions

概述

如果您希望由于事件而发生某些操作（例如发送通知），则需要配置操作。

可以根据所有支持的类型的事件来定义操作：

- 触发事件 - 当trigger的状态从*OK* 转到 *PROBLEM* 或者转回时
- 发现事件 - 发生网络发现时
- 自动注册事件 - 当新的活动代理自动注册
- 内部事件 - 当项目不受支持或触发器进入未知状态

配置动作

要配置操作，请执行以下操作：

- 进入 配置 - >操作
- 从*Event source*下拉单中选择所需的来源
- 点击 创建 *action*
- 命名*action*
- 选择进行操作的条件
- 选择 *operations* 来执行
- 选择*recovery operations*来执行

一般动作属性：

Actions

Action Operations Recovery operations

Name	Report problems to Zabbix administrators	
Type of calculation	And/Or	A and B
Conditions	Label	Name
	A	Maintenance status not in <i>maintenance</i>
	B	Host group = <i>Zabbix servers</i>
New condition	Host group	= type here to search
	Add	
Enabled	<input checked="" type="checkbox"/>	
	Add	Cancel

参数	说明
Name	action名称
Type of calculation	选择评估 type_of_calculation选项 作为行动条件 (有多个条件) : And - 必须满足所有条件 Or - 如果满足一个条件就足够了 And/Or - 两者的组合: AND与不同的条件类型和 OR具有相同的条件类型 Custom expression - 用于评估操作条件的用户定义的计算公式。
Conditions	行动条件清单。
New condition	选择一个新的动作 条件 并且点击Add .
Enabled	选中该复选框以启用该操作。 否则将被禁用。

1 条件

概述

只有在事件与定义的条件匹配的情况下才执行操作。 配置[action](#)时设置条件。

可以为基于触发的动作设置以下条件：

条件类型	支持的操作	说明
<i>Application</i>	=like not like	指定要排除的应用程序或应用程序。 = - 事件属于与指定应用程序链接的项目的触发器。 like - 事件属于与包含字符串的应用程序链接的项目的触发器。 .not like - 事件属于链接到不包含字符串的应用程序的项目的触发器。
<i>Host group</i>	=<>	指定要排除的主机组或主机组。 = - 事件属于此主机组。 <> - 事件不属于此主机组。指定父主机组隐含地选择所有嵌套的主机组。要仅指定父组，必须使用 <> 运算符另外设置所有嵌套组。
<i>Template</i>	=<>	指定要排除的模板或模板。 = - 属于从此模板继承的触发器的事件。 <> - 不属于从此模板继承的触发器的事件。
<i>Host</i>	=<>	指定要排除的主机或主机。 = - 属于这个主机的事件。 <> - 不属于这个主机的事件。
<i>Tag</i>	= <>like not like	指定事件标记或要排除的事件标记。 = - 含有该标记的事件。 <> - 不含该标记的事件。 like - 标签中包含此字符串的事件。 .not like - 标签中不包含此字符串的事件
<i>Tag value</i>	= <>like not like	指定事件标签和值组合或要排除的标签和值组合。 = - 包含该值和标签的事件。 <> - 不包含该值和标签的事件。 like - 值和标签中包含该字符串的事件。 .not like - 值和标签中不包含该字符串的事件
<i>Trigger</i>	=<>	指定触发器或要排除的触发器。 = - 由该触发器产生的事件。 <> - 除了这一个，由任何其他触发器生成的事件。
<i>Trigger name</i>	like not like	在触发器名称中指定一个字符串或要排除的字符串。 like - 事件由触发器生成，在名称中包含此字符串。区分大小写。 .not like - 触发器名称中不包含该字符串。区分大小写。 <i>Note:</i> 输入的值将与所有宏扩展的触发器名称进行比较。
<i>Trigger severity</i>	=<>>=<=	指定触发严重性。 = - 等于触发严重性。 <> - 不等于触发严重性。 => - 大于或等于触发严重性。 <= - 小于或等于触发严重性。
<i>Time period</i>	in not in	指定时间段或要排除的时间段。 in - 事件时间在该时间段内。 .not in - 事件时间不在该时间段内。 格式描述参见 Time period specification 页面。
<i>Maintenance status</i>	in not in	指定主机进行维护或不进行维护。 in - 主机处于维护模式。 .not in - 主机不在维护模式。 <i>Note:</i> 如果触发表达式中涉及到多个主机，则至少有一个主机不在维护模式下，条件匹配。

当触发器的新动作被创建时，它会自动获得一个条件：“维护状态=不在维护”。有了这种情况，维护中的主机不会发送通知。 用户可以删除此条件。

可以为基于发现的事件设置以下条件：

条件类型	支持的操作	说明
<i>Host IP</i>	=<>	指定要发现的主机的IP地址范围或要排除的范围。 = - 主机IP在该范围内。 <> - 主机IP不在该范围内。它可能有以下格式：单IP： 192.168.1.33IP地址范围： 192.168.1.1-10.1-254IP mask： 192.168.4.0/24List： 192.168.1.1-254, 192.168.2.1-

		100, 192.168.2.200, 192.168.4.0/24\自Zabbix 3.0.0起就提供列表格式的空格。
Service type	=>	指定已发现服务的服务类型或者要排除的服务类型。= - 匹配发现的服务。<> - 与发现的服务不匹配。可用服务类型: SSH, LDAP, SMTP, FTP, HTTP, HTTPS (<i>available since Zabbix 2.2 version</i>), POP, NNTP, IMAP, TCP, Zabbix agent, SNMPv1 agent, SNMPv2 agent, SNMPv3 agent, ICMP ping, telnet (<i>available since Zabbix 2.2 version</i>)。
Service port	=>	指定发现的服务或的TCP端口范围或者要排除的TCP端口范围。= - 服务端口在该范围内。<> - 服务端口不在该范围内
Discovery rule	=>	指定发现规则或要排除的发现规则。= - 使用这个发现规则。<> - 使用除此之外的任何其他发现规则。
Discovery check	=>	指定discovery check或要排除的discovery check= - 使用这个discovery check.<> - 使用除此之外的其他任何discovery check
Discovery object	=	指定发现的对象。= - 等于发现的对象(设备或服务)。
Discovery status	=	Up - 匹配'Host Up' 和 'Service Up' 事件 Down - 匹配'Host Down' 和 'Service Down' 事件 Discovered - 匹配 'Host Discovered' 和 'Service Discovered' 事件 Lost - 匹配 'Host Lost' 和 'Service Lost' 事件
Uptime/Downtime	>=<	Uptime for 'Host Up' and 'Service Up' events. Downtime for 'Host Down' and 'Service Down' events.>= - 大于或者等于。参数以秒为单位给出。<= - 小于或等于。参数以秒为单位给出。
Received value	=>>= <=like not like	指定从代理接收的值(Zabbix, SNMP)。区分大小写字符串比较。如果为规则配置了多个Zabbix代理或SNMP检查，则检查所有的Zabbix代理或SNMP检查(每个检查生成与所有条件匹配的新事件)。= - 等于该值。<> - 不等于该值。>= - 大于或者等于该值。<= - 小于或者等于该值。 like - 包含子串。参数作为字符串给出。 not like - 不包含子串。参数作为字符串给出。
Proxy	=>	指定代理或要排除的代理。= - 使用这个代理。<> - 使用除此之外的任何其他代理。

基于活动代理自动注册的动作可以设置以下条件:

条件类型	支持的操作	说明
Host metadata	like not like	指定主机元数据或要排除的主机元数据。 like - 主机元数据包含字符串。 not like - 主机元数据不包含字符串。可以在 代理配置文件 中指定主机元数据。
Host name	like not like	指定主机名或要排除的主机名。 like - 包含字符串的主机名。 not like - 不包含字符串的主机名
Proxy	=>	指定代理或要排除的代理。= - 使用这个代理。<> - 使用除此之外的任何其他代理。

可以根据内部事件为动作设置以下条件:

条件类型	支持的操作	说明
Application	=like not like	指定应用程序或要排除的应用程序。= - 属于与指定应用程序链接的项的事件。 like - 属于与包含字符串的应用程序链接的项的事件。 not like - 属于不包含字符串的应用程序链接的项的事件。

<i>Event type</i>	=	Item in "not supported" state - 匹配监控项从“正常”到“不支持”状态的事件 Low-level discovery rule in "not supported" state - 匹配低级发现规则从“正常”到“不支持”状态的事件 Trigger in "unknown" state - 匹配触发从“正常”到“未知”状态的事件
<i>Host group</i>	=<>	指定主机组或要排除的主机组。= - 属于此主机组的事件。<> - 不属于此主机组的事件。
<i>Template</i>	=<>	指定模板或要排除的模板。= - 属于从此模板继承的监控项/触发器/低级发现规则的事件。<> - 不属于从此模板继承的监控项/触发器/低级发现规则的事件。
<i>Host</i>	=<>	指定主机或要排除的主机。= - 属于这个主机的事件。<> - 不属于这个主机的事件。

运算类型

计算条件的以下选项可用：

- **And** - 必须满足所有条件

请注意，当它们被选为“Trigger =”“条件时，在几个触发器之间不允许使用“And”计算。 操作只能根据一个触发事件执行。

- **Or** - 如果满足一个条件就足够了
- **And/Or** - 两者的组合：AND与不同的条件类型和OR具有相同的条件类型，例如：

•

*Host group = Oracle servers**Host group = MySQL servers**Trigger name like 'Database is down'**Trigger name like 'Database is unavailable'*

计算为

(*Host group = Oracle servers or Host group = MySQL servers*)**and**(*Trigger name like 'Database is down' or Trigger name like 'Database is unavailable'*)

- **Custom expression** - a user-defined calculation formula for evaluating action conditions. It must include all conditions (represented as uppercase letters A, B, C, ...) and may include spaces, tabs, brackets (), **and** (case sensitive), **or** (case sensitive).

While the previous example with **And/Or** would be represented as (A or B) and (C or D), in a custom expression you may as well have multiple other ways of calculation:

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

由于删除对象，被禁用的actions

如果某个操作条件/操作中使用的某个对象（主机，模板，触发器等）被删除，则会删除条件/操作，禁用该操作以避免操作的错误执行。 该操作可以由用户重新启用。

当删除以下项目是会发生这种情况：

- 主机组（“机组”条件，特定机组上的“远程命令”操作）；
- 主机（“主机”条件，特定主机上的“远程命令”操作）；
- 模板（“模板”条件，“链接到模板”和“与模板的链接”操作）；
- 触发器（“触发”条件）；
- 发现规则（使用“发现规则”和“发现检查”条件时）；
- 代理（“代理”条件）。

Note: 如果远程命令有许多目标主机，并且我们删除其中的一个，则只有该主机将从目标列表中删除，操作本身将保留。但是，如果它是唯一的主机，操作也将被删除。“链接到模板”和“取消与模板的链接”操作也是一样。

删除“发送消息”操作中使用的用户或用户组时，操作不会被禁用。

2 操作

概述

您可以为所有事件定义以下操作：

- 发送信息
- 执行远程命令（包括 IPMI）

对于发现事件，还有其他操作可用：

- 添加主机
- 删除主机
- 启用主机
- 禁用主机
- 添加到群组
- 从组中删除
- 链接到模板
- 取消与模板的链接
- 设置主机库存模式

自动注册事件的附加操作有：

- 添加主机
- 禁用主机
- 添加到群组
- 链接到模板
- 设置主机库存模式

配置操作

要配置操作，进入action [配置](#) 中的 [操作](#) 选项卡，然后单击操作块中的 *New*。编辑操作步骤，然后单击 *Add* 添加到 操作列表中。

操作属性：

Action Operations Recovery operations Acknowledgement operations

Default operation step duration 1h

Default subject Problem: {TRIGGER.NAME}

Default message Problem started at {EVENT.TIME} on {EVENT.DATE}
Problem name: {TRIGGER.NAME}
Host: {HOST.NAME}
Severity: {TRIGGER.SEVERITY}

Original problem ID: {EVENT.ID}
(TRIGGER.URL)

Pause operations while in maintenance

Operations	Steps	Details	Start in	Duration	Action
1	Send message to user groups:	Zabbix administrators via Email	Immediately	Default	Edit Remove
3	Send message to user groups:	Managers via SMS	02:00:00	Default	Edit Remove
4	Run remote commands on current host		03:00:00	Default	Edit Remove

Operation details

Steps 3 - 3 (0 - infinitely)

Step duration 0 (0 - use action default)

Operation type Send message

Send to User groups	User group	Action
Managers		Remove
Add		

Send to Users	User	Action
Add		

Send only to SMS

Default message

Conditions	Label	Name	Action
A		Event acknowledged = Not Ack	Remove
New			

[Update](#) [Cancel](#)

[Add](#) [Cancel](#)

参数	说明
默认操作步长	一个操作步骤默认持续时间（最少60秒）。例如，一小时的持续时间意味着如果执行操作，则在下一步之前将经过一小时。
Default subject	默认消息主题为通知。 主题可能包含 宏 。
Default message	通知的默认消息。 消息可能包含 宏 。
Pause operations while in maintenance	标记此复选框以延长维护期间的操作。如果取消选中此复选框，即使在维护期间，操作也将立即执行。Zabbix 3.2.0之后支持此选项。
Operations	显示操作操作，具体如下： Steps - 分配操作的升级步骤 Details - 操作类型及其收件人/目标。自Zabbix 2.2以来，操作列表还显示了发送消息中使用的媒体类型（电子邮件，SMS，Jabber等）以及通知收件人的名称和姓氏（在别名之后的括号中） Start in - 执行操作后的事件多长时间 Duration (sec) - 显示步长。 如果步骤使用默认持续时间，则显示Default如果使用自定义持续时间，则显示时间。 Action - 显示用于编辑和删除操作的链接。要配置新操作，请单击New.

<i>Operation details</i>	此块用于配置操作的详细信息。	
	<i>Steps</i>	在 升级 计划表中选择步骤分配操作： From - 从这一步开始执行 To - 执行到此步骤 ($\theta=\text{infinity}$, 执行不会受到限制)
<i>Step duration</i>	这些步骤的自定义持续时间 (θ = 使用默认步骤持续时间)。几个操作可以分配到同一步骤。如果这些操作具有不同的步长定义，则考虑最短的步骤并将其应用于该步骤。	
<i>Operation type</i>	所有事件都有两种操作类型： Send message - 发送消息给用户 Remote command - 执行远程命令更多的操作可用于发现和基于自动注册的事件(见上文)。	
	操作类型： 发信息	
<i>Send to user groups</i>	点击 <i>Add</i> 选择要发送消息的用户组。用户组必须至少具有“读取” 权限 以获得主机的通知	
<i>Send to users</i>	点击 <i>Add</i> 选择要发送消息的用户。用户组必须至少具有“读取” 权限 以获得主机的通知	
<i>Send only to</i>	发送消息到所有定义的媒体类型或只选一个。	
<i>Default message</i>	如果选择，将使用默认消息(见上文)。	
<i>Subject</i>	自定义消息的主题。主题可能包含宏。	
<i>Message</i>	自定义消息。消息可能包含宏。	
操作类型： 远程命令		
<i>Target list</i>	选择要执行命令的目标： Current host - 命令在导致问题事件的触发器的主机上执行。如果触发器中有多个主机，则此选项将无法正常工作。 Host - 选择主机以执行命令。 Host group - 选择主机组以执行命令。指定父主机组隐含地选择所有嵌套的主机组。因此，远程命令也将在嵌套组的主机上执行。主机上的命令只能执行一次，即使主机与多次匹配(例如来自多个主机组，单独地和从主机组匹配)。如果在Zabbix服务器上执行命令，目标列表是无意义的。在这种情况下选择更多目标只会导致服务器上执行的命令更多次。请注意，对于全局脚本，目标选择也取决于全局脚本 配置 中设置的 主机组 。	
<i>Type</i>	选择命令类型： IPMI - 执行IPMI命令 Custom script - 执行一组自定义的命令 SSH - 执行SSH命令 Telnet - 执行Telnet命令 Global script - 执行管理 - >脚本中定义的全局脚本之一。	
<i>Execute on</i>	在Zabbix服务器或Zabbix代理上执行自定义脚本。要在代理上执行脚本，它必须是 配置为 允许来自服务器的远程命令。如果选择“自定义脚本”作为类型，则该字段可用。	
<i>Commands</i>	输入命令。支持的宏将根据导致事件的触发表达式进行解析。例如，主机宏将解析为主机的触发器表达式(而不是目标列表)。	
	<i>Conditions</i>	执行操作的条件： Not ack - 只有当事件未被确认时 Ack - 只有事件被确认时。



1 发送消息

概述

发送消息是通知人们遇到问题的最佳方式之一。 这就是为什么它是Zabbix提供的主要行动之一。

配置

为了能够发送和接收Zabbix的通知，您必须：

- 定义[media](#)发送消息
- 配置[动作操作](#) 向一个定义的媒体发送消息

Zabbix仅向生成该事件的主机至少具有“读取”权限的用户发送通知。 必须至少有一个触发表达式的主机可访问。

您可以配置使用 [升级](#)发送消息的自定义场景。

要成功接收和阅读Zabbix的电子邮件，电子邮件服务器/客户端必须支持标准的“SMTP / MIME电子邮件”格式，因为Zabbix发送UTF-8数据（如果主题仅包含ASCII字符，则不是UTF-8编码）。消息的主题和主体是base64编码，遵循“SMTP / MIME电子邮件”格式标准。

跟踪消息

您可以在监控 ->问题中查看发送的消息的状态。在Actions column 您可以看到有关所采取actions的汇总信息。在那里绿色的数字表示发送的消息，红色的 - 失败的消息。 进行中 表示启动了一个动作。 失败 通知没有成功执行任何操作。

如果您点击活动时间查看活动详细信息，您还将看到 消息动作 块包含由于事件发送（或未发送）的消息的详细信息。在报表 ->动作日志 您将看到为配置操作的那些事件所采取的所有操作的详细信息。

2 远程命令

概述

使用远程命令，您可以定义在某些情况下，监视的主机上会自动执行某个预定义的命令。

因此，远程命令是智能主动监控的强大机制

在功能最明显的用途中，您可以尝试：

- 如果没有响应，则自动重新启动某些应用程序（Web服务器，中间件，CRM）
- 如果不响应请求，请使用IPMI“reboot”命令重新启动一些远程服务器
- 如果磁盘空间不足，可自动释放磁盘空间（删除较旧的文件，清理/ tmp）
- 根据CPU负载，将VM从一个物理盒移植到另一个物理盒
- 在CPU（磁盘，内存，任何资源）不足的情况下，将新节点添加到云环境中

配置远程命令的操作类似于发送消息的操作，唯一的区别是Zabbix将执行命令而不是发送消息。

不支持在Zabbix代理监视的Zabbix代理上执行远程命令，因此需要从Zabbix服务器到代理的命令才能直接连接。

远程命令限制为255个字符。可以通过将多个命令放置在新行上来执行多个命令。远程命令可能包含宏。由Zabbix服务器执行的远程命令如[命令执行](#)中所述执行，包括退出代码检查。

即使目标主机处于维护状态，也会执行远程命令。

以下教程提供了有关如何设置远程命令的分步说明。

配置

在Zabbix代理（自定义脚本）上执行的那些远程命令必须首先在相应的命令中启用[zabbix_agentd.conf](#)。

确保 **EnableRemoteCommands** 参数设置为 **1** 并取消注释。如果更改此参数，请重新启动代理守护程序。

远程命令不适用于活动的Zabbix代理。

然后，在配置新的动作时进入配置 - >操作：

- 定义适当的条件。在此示例中，设置在Apache应用程序之一的任何灾难问题时激活该操作：

- 在操作选项卡中，选择远程命令操作类型
- 选择远程命令类型 (IPMI, 自定义脚本, SSH, Telnet, 全局脚本)
- 输入远程命令

例如：

```
1. sudo /etc/init.d/apache restart
```

在这种情况下，Zabbix将尝试重新启动Apache进程。 使用此命令，确保该命令在Zabbix代理上执行（点击Zabbix代理按钮执行）。

Note the use of **sudo** - 默认情况下，Zabbix用户没有权限重新启动系统服务。有关如何配置 **sudo** 的提示，请参见下文。

Zabbix代理应在远程主机上运行并接受传入连接。Zabbix代理在后台执行命令。

Zabbix代理程序上的远程命令由系统无延迟执行。运行[, nowait]键，不检查执行结果。在Zabbix服务器上，远程命令是在zabbix_server的TrapperTimeout参数中设置的超时执行的。conf文件被 [检查](#) 以执行结果。

访问权限

确保'zabbix'用户具有已配置命令的执行权限。可能有兴趣使用 **sudo** 来访问特权命令。要配置访问，请以root身份执行：

```
1. # visudo
```

可以在 **sudoers**文件中使用的行：

```
1. # allows 'zabbix' user to run all commands without password.
2. zabbix ALL=NOPASSWD: ALL
```

```
1. # allows 'zabbix' user to restart apache without password.
2. zabbix ALL=NOPASSWD: /etc/init.d/apache restart
```

在某些系统上`sudoers` 文件将阻止非本地用户执行命令。 若要修改，在`/etc/sudoers`中添加**requiretty**注释。

具有多个接口的远程命令

如果目标系统具有所选类型的多个接口（Zabbix代理或IPMI），则将在默认接口上执行远程命令。

可以通过SSH和Telnet使用除Zabbix代理之外的其他界面执行远程命令。 可用的使用界面按以下顺序选择：

- Zabbix agent default interface
- SNMP default interface
- JMX default interface
- IPMI default interface

IPMI远程命令

对于IPMI远程命令，应使用以下语法：

```
1. <command> [<value>]
```

where

- - one of IPMI commands without spaces
- - 'on', 'off' or any unsigned integer. is an optional parameter.

示例

示例 1

在一定条件下重新启动Windows。

为了在Zabbix检测到问题时自动重新启动Windows，请定义以下操作：

参数	说明
Operation type	'Remote command'
Type	'Custom script'
Command	c:\windows\system32\shutdown.exe -r -f

示例 2

使用IPMI控制重新启动主机。

参数	说明
Operation type	'Remote command'
Type	'IPMI'
Command	reset

示例 3

使用IPMI控制关闭主机电源。

参数	说明
Operation type	'Remote command'
Type	'IPMI'
Command	power off

3 附加操作

概述

对于发现事件，还有其他操作可用：

- 添加主机
- 删除主机
- 启用主机
- 禁用主机
- 添加到群组
- 从组中删除
- 链接到模板
- 取消与模板的链接
- 设置主机库存模式

自动注册事件的附加操作有：

- 添加主机
- 禁用主机
- 添加到群组
- 链接到模板
- 设置主机库存模式

添加主机

主机在发现过程中被添加，一旦主机被发现，而不是发现过程结束。

由于网络发现可能需要一些时间，因为许多不可用的主机/服务有耐心和使用合理的IP范围是可取的。

添加主机时，其名称由标准 `gethostbyname` 函数决定。如果可以解析主机，则使用解析名称。如果没有，则使用IP地址。此外，如果IPv6地址必须用于主机名，则所有“：“（冒号）将被替换为“_”（下划线），因为主机名中不允许冒号。

如果执行代理发现，当前主机名查找仍然发生在Zabbix服务器上。

如果一个主机已经存在于与新发现的Zabbix配置中相同名称的主机上，1.8之前的Zabbix版本将添加具有相同名称的另一个主机。Zabbix 1.8.1和更高版本将 `_ N` 添加到主机名，其中 `N` 是增加数字，从2开始。

4 在信息中使用宏

概述

在消息主题和消息文本中，您可以使用宏来更有效的问题报告。

由Zabbix提供支持[完整的宏列表](#)。

示例

这里的例子说明了如何在消息中使用宏。

示例 1

Message subject:

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

收到消息后，消息主题将被替换为：

```
1. zabbix.zabbix.com服务器上的处理器负载太高: PROBLEM
```

示例 2

Message:

```
1. Processor load is: {zabbix.zabbix.com:system.cpu.load[,avg1].last()}
```

When you receive the message, the message will be replaced by something like:

```
1. Processor load is: 1.45
```

示例E 3

Message:

```
1. Latest value: {{HOST.HOST}:{ITEM.KEY}.last()}
2. MAX for 15 minutes: {{HOST.HOST}:{ITEM.KEY}.max(900)}
3. MIN for 15 minutes: {{HOST.HOST}:{ITEM.KEY}.min(900)}
```

收到消息时，消息将被替换为：

```
1. Latest value: 1.45
2. MAX for 15 minutes: 2.33
3. MIN for 15 minutes: 1.01
```

示例 4

Message:

```
1. http://<server_ip_or_name>/zabbix/events.php?triggerid={TRIGGER.ID}&filter_set=1
```

When you receive the message, it will contain a link to all events of the problem trigger.

示例 5

Informing about values from several hosts in a trigger expression.

Message:

```
1. Trigger: {TRIGGER.NAME}
2. Trigger expression: {TRIGGER.EXPRESSION}
3.
4. 1. Item value on {HOST.NAME1}: {ITEM.VALUE1} ({ITEM.NAME1})
5. 2. Item value on {HOST.NAME2}: {ITEM.VALUE2} ({ITEM.NAME2})
```

When you receive the message, the message will be replaced by something like:

```
1. Trigger: Processor load is too high on a local host
2. Trigger expression: {Myhost:system.cpu.load[percpu,avg1].last()}>5 | {Myotherhost:system.cpu.load[percpu,avg1].last()}>5
3.
4. 1. Item value on Myhost: 0.83 (Processor load (1 min average per core))
5. 2. Item value on Myotherhost: 5.125 (Processor load (1 min average per core))
```

示例6

Receiving details of both the problem event and recovery event in a [recovery](#) message:

Message:

```
1. Problem:
2.
3. Event ID: {EVENT.ID}
4. Event value: {EVENT.VALUE}
5. Event status: {EVENT.STATUS}
6. Event time: {EVENT.TIME}
7. Event date: {EVENT.DATE}
8. Event age: {EVENT.AGE}
9. Event acknowledgement: {EVENT.ACK.STATUS}
10. Event acknowledgement history: {EVENT.ACK.HISTORY}
11.
12. Recovery:
13.
14. Event ID: {EVENT.RECOVERY.ID}
15. Event value: {EVENT.RECOVERY.VALUE}
16. Event status: {EVENT.RECOVERY.STATUS}
17. Event time: {EVENT.RECOVERY.TIME}
```

```
18. Event date: {EVENT.RECOVERY.DATE}
```

When you receive the message, the macros will be replaced by something like:

```
1. Problem:  
2.  
3. Event ID: 21874  
4. Event value: 1  
5. Event status: PROBLEM  
6. Event time: 13:04:30  
7. Event date: 2014.01.02  
8. Event age: 5m  
9. Event acknowledgement: Yes  
10. Event acknowledgement history: 2014.01.02 13:05:51 "John Smith (Admin)"  
11. -acknowledged-  
12.  
13. Recovery:  
14.  
15. Event ID: 21896  
16. Event value: 0  
17. Event status: OK  
18. Event time: 13:10:07  
19. Event date: 2014.01.02
```

Zabbix 2.2.0之后支持原始问题事件和恢复事件的单独通知宏。

3 恢复操作

概述

恢复操作允许在问题解决时通知您。

恢复操作支持消息和远程命令。恢复操作不支持升级 - 所有操作都分配到一个步骤。

使用场景

恢复操作的一些用例如下：

- 通知所有通知有关问题的用户
 - 选择“发送恢复消息”作为操作类型
- 恢复时有多个操作：发送通知并执行远程命令
 - 添加发送消息和执行命令的操作类型
- 在外部帮助台/票务系统中打开机票，并在问题解决时将其关闭
 - 创建一个与帮助台系统通信的外部脚本
 - 创建一个操作，该操作具有执行此脚本的操作，从而打开一张票据
 - 恢复操作，使用其他参数执行此脚本并关闭故障单
 - 使用{EVENT.ID}宏来引用原始问题

配置恢复操作

配置恢复操作：

- 进入action配置中的恢复操作标签
- 点击操作块中的*New*
- 编辑操作详情并且点击 *Add*

以添加几个操作。

恢复操作属性：

参数	说明
<i>Default subject</i>	恢复通知的默认消息主题。 主题可能包含 宏 。
<i>Default message</i>	恢复通知的默认消息。 消息可能包含 宏 。
<i>Operations</i>	恢复操作详细信息显示。要配置新的恢复操作，请单击 New 。
<i>Operation details</i>	此块用于配置恢复操作的详细信息。
<i>Operation type</i>	有三种操作类型可用于恢复事件： Send recovery message - 所有在问题事件通知的用户发送恢复消息 Send message - 发送恢复信息给指定的用户 Remote command - 执行远程命令
操作类型：发送恢复信息	
<i>Default message</i>	如果选择，将使用默认消息（见上文）。
<i>Subject</i>	自定义消息的主题。 主题可能包含宏。
<i>Message</i>	自定义消息。 消息可能包含宏。

操作类型：发信息	
发送到用户组 <i>Send to users</i>	点击 Add 选择要发送恢复消息的用户组。用户组必须至少具有“读取”权限向主机通知。
<i>Send only to</i>	将恢复消息发送到所有定义的媒体类型或仅选定的媒体类型。
<i>Default message</i>	如果选择，将使用默认消息（见上文）。
<i>Subject</i>	自定义消息的主题。主题可能包含宏。
<i>Message</i>	自定义消息。消息可能包含宏。
Operation type: 远程命令 <i>Type</i>	选择命令类型： IPMI - 执行IPMI命令 Custom script - 执行一组自定义的命令。您可以选择在Zabbix代理或Zabbix服务器上执行该命令。 SSH - 执行SSH命令 Telnet - 执行Telnet命令 Global script - 执行管理 - >脚本其中定义的全局脚本之一。
<i>Execute on</i>	在Zabbix代理或Zabbix服务器上执行命令。
<i>Commands</i>	输入命令。



4 Escalations

概述

通过Escalations，您可以创建发送通知或执行远程命令的自定义场景。

实际应用中，这意味着：

- 用户可以立即收到新问题通知
- 通知可以重复，直到问题解决
- 发送通知可以延时
- 通知可以升级到另一个“较高”的用户组
- 可以立即执行远程命令，或者长时间不解决问题

操作会根据升级步骤进行升级。每一步都有一段时间。

您可以定义默认持续时间和单个步骤的自定义持续时间。一个升级步骤的最短持续时间为60秒。

您可以从任何步骤开始执行操作，例如发送通知或执行命令。第一步是立即采取行动。如果要延迟操作，可以将其分配给稍后的步骤。对于每个步骤，可以定义几个操作。

升级步骤的数量不受限制。

[配置操作](#)是即可定义Escalations。Escalations仅对问题操作支持，而不是恢复。

Escalations的其他方面

让我们考虑如果一个操作包含几个升级步骤，在不同的情况下会发生什么。

情况	运行
在发送初始问题通知后，所涉及的主机进入维护状态	取决于action配置中设置的在维护期间暂停操作，所有剩余的升级步骤都由维护期间或延迟引起的延迟执行。维护期不能取消操作。
在时间段操作条件中定义的时间段在发送初始通知后结束	执行所有剩余的升级步骤。时间段条件不能停止操作；它对于何时启动/未启动操作而不是操作具有效果。
维护过程中出现问题，维护结束后继续（未解决）	取决于action配置中在维护期间暂停操作的设置，所有升级步骤都可以从维护结束或立即执行。
在无数据维护期间会出现问题，并在维护结束后继续（未解决）	在执行所有升级步骤之前，必须等待触发器触发。
不同的升级紧随其后并重叠	每个新的升级的执行取代先前的升级，但是至少一个升级步骤总是在以前的升级中执行。在针对触发器的每个事件评估创建的事件的操作中，此行为都是相关的。
在升级过程中（如正在发送的消息），基于任何类型的事件： - 该操作被禁用 - 该事件被删除 基于触发事件： - 触发器被禁用或删除 - 主机或项目被禁用 基于关于触发器的内部事件： - 触发器被禁用或删除 基于关于项目/低	发送正在发送的消息，然后再发送一条关于升级的消息。后续消息将在邮件正文的开头有取消文本（注意：取消邮件已取消）命名原因（例如，注意：取消升级：动作'<动作名称>'禁用）。通过这种方式，收件人被通知升级被取消，不再执行任何步骤。此消息将发送给接收通知的所有人员。取消

级发现规则的内部事件： - 该项目被禁用或删除 - 主机被禁用	的原因也记录到服务器日志文件中(从Debug Level 3=Warning)开始。
在升级过程中 (如发送消息)，删除该操作	不再发送消息。 信息被记录到服务器日志文件 (从Debug Level 3=Warning)开始，例如： <code>escalation cancelled: action id:334 deleted</code>

Escalation 示例

示例 e 1

Sending a repeated notification once every 30 minutes (5 times in total) to a 'MySQL Administrators' group. To configure:

- in Operations tab, set the *Default operation step duration* to '1800' seconds (30 minutes)
- Set the escalation steps to be *From '1' To '5'*
- Select the 'MySQL Administrators' group as recipients of the message

Operations	Steps Details	Start in	Duration
1 - 5 Send message to user groups: MySQL Administrators via all media. Immediately Default	New		

Notifications will be sent at 0:00, 0:30, 1:00, 1:30, 2:00 hours after the problem starts (unless, of course, the problem is resolved sooner).

If the problem is resolved and a recovery message is configured, it will be sent to those who received at least one problem message within this escalation scenario.

If the trigger that generated an active escalation is disabled, Zabbix sends an informative message about it to all those that have already received notifications.

示例 2

Sending a delayed notification about a long-standing problem. To configure:

- In Operations tab, set the *Default operation step duration* to '36000' seconds (10 hours)
- Set the escalation steps to be *From '2' To '2'*

Default operation step duration

Default subject

Default message

```
Problem started at {EVENT.TIME} on {EVENT.DATE}
Problem name: {TRIGGER.NAME}
Host: {HOST.NAME}
Severity: {TRIGGER.SEVERITY}

Original problem ID: {EVENT.ID}
{TRIGGER.URL}
```

Pause operations while in maintenance

Operations

Steps	Details	Start in	Duration
2	Send message to users: IT Management (Andrew Head) via Email	10:00:00	Default
New			

A notification will only be sent at Step 2 of the escalation scenario, or 10 hours after the problem starts.

You can customize the message text to something like 'The problem is more than 10 hours old'.

示例 3

Escalating the problem to the Boss.

In the first example above we configured periodical sending of messages to MySQL administrators. In this case, the administrators will get four messages before the problem will be escalated to the Database manager. Note that the manager will get a message only in case the problem is not acknowledged yet, supposedly no one is working on it.

The screenshot shows the 'Operations' tab of the Zabbix Escalation configuration. At the top, there are fields for 'Default operation step duration' (set to 30m), 'Default subject' (Problem: {TRIGGER.NAME}), and 'Default message' (containing placeholders for event time, date, host, severity, original problem ID, and trigger URL). A checkbox 'Pause operations while in maintenance' is checked. Below this, the 'Operations' section lists two steps: 'Send message to user groups: MySQL Administrators via Email' (immediately, default duration) and 'Send message to users: Database manager (Andrew Head) via Email' (02:00:00, default duration). The 'Operation details' section includes fields for 'Steps' (5 - 5 (0 - infinitely)), 'Step duration' (0), 'Operation type' (Send message), and 'Send to User groups' (User group, Add button). It also includes 'Send to Users' (User, Database manager (Andrew Head), Add button) and 'Send only to' (Email dropdown). A 'Default message' section contains a subject (Unacknowledged problem: {TRIGGER.NAME}) and a message body with placeholders. Finally, a 'Conditions' section shows a condition 'Event acknowledged = Not Ack' labeled 'A'.

Note the use of {ESC.HISTORY} macro in the message. The macro will contain information about all previously executed steps on this escalation, such as notifications sent and commands executed.

示例 4

A more complex scenario. After multiple messages to MySQL administrators and escalation to the manager, Zabbix will try to restart the MySQL database. It will happen if the problem exists for 2:30 hours and it hasn't been acknowledged.

If the problem still exists, after another 30 minutes Zabbix will send a message to all guest users.

If this does not help, after another hour Zabbix will reboot server with the MySQL database (second remote command) using IPMI commands.

The screenshot shows the 'Operations' tab of an escalation configuration. It includes fields for 'Default operation step duration' (30m), 'Default subject' (Problem: {TRIGGER.NAME}), and a 'Default message' box containing problem details. A checkbox 'Pause operations while in maintenance' is checked. Below is a table of operations:

Operations	Steps Details	Start in	Duration	Action
1 - 0	Send message to user groups: MySQL Administrators via Email	Immediately	Default	Edit Remove
5	Send message to users: Database manager (Andrew Head) via Email	02:00:00	Default	Edit Remove
6	Run remote commands on current host	02:30:00	Default	Edit Remove
7	Send message to user groups: Guests via all media	03:00:00	Default	Edit Remove
9	Run remote commands on current host	04:00:00	Default	Edit Remove
New				

示例 5

An escalation with several operations assigned to one step and custom intervals used. The default operation step duration is 30 minutes.

The screenshot shows the 'Operations' tab of an escalation configuration, identical to Example 4 but with different operation details. The 'Default message' box contains problem details. A checkbox 'Pause operations while in maintenance' is checked. Below is a table of operations:

Operations	Steps Details	Start in	Duration	Action
1 - 4	Send message to user groups: MySQL Administrators via Email	Immediately	Default	Edit Remove
5 - 6	Send message to users: Database manager (Andrew Head) via Email	02:00:00	1h	Edit Remove
5 - 7	Send message to user groups: Zabbix administrators via Email	02:00:00	10m	Edit Remove
11	Send message to user groups: Guests via Email	04:00:00	Default	Edit Remove

通知将发送如下：

- to MySQL administrators at 0:00, 0:30, 1:00, 1:30 after the problem starts
- to Database manager at 2:00 and 2:10 (and not at 3:00; seeing that steps 5 and 6 overlap with the next operation, the shorter custom step duration of 600 seconds in the next operation overrides the longer step duration of 3600 seconds tried to set here)
- to Zabbix administrators at 2:00, 2:10, 2:20 after the problem starts (the custom step duration of 600 seconds working)
- to guest users at 4:00 hours after the problem start (the default step duration of 30 minutes returning between steps 8 and 11)

3 接收不受支持的项目的通知

概述

Zabbix 2.2之后支持接收不支持的项目的通知。

它是Zabbix内部事件概念的一部分，允许用户在这些场合获得通知。内部事件反映了状态的变化：

- 当监控项从“正常”变成“不支持”（和返回）
- 当触发器从“正常”改为“未知”（和返回）
- 当低级发现规则从“正常”到“不支持”（和返回）

本节介绍了当项目不受支持时接收通知的操作方法。

配置

总体来说，设置通知的过程应该与在Zabbix中设置警报类似。

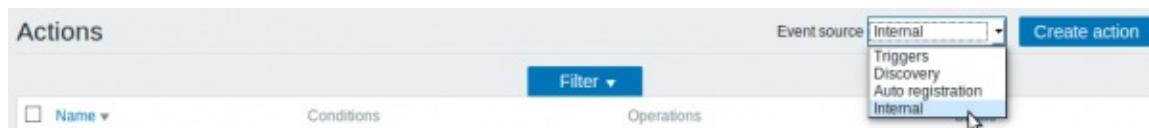
步骤 1

配置 [一些媒介](#)，例如电子邮件，短信或Jabber，用于通知。请参阅手册的相应章节执行此任务。

为了通知内部事件，使用默认严重性（'未分类'），因此如果要接收内部事件的通知，请在[配置用户媒介](#)时选中。

步骤 2

进入配置 - >操作 选择 内部 作为事件来源.点击右上角创建action 开一个动作配置表单。



步骤 3

在动作选项卡中输入操作的名称。然后在新条件块中选择 事件类型，选择项目处于“不支持”状态 作为值.

The screenshot shows the 'Actions' configuration interface. The 'Action' tab is selected. The 'Name' field contains 'Report not supported items'. Under the 'Conditions' section, there is one condition labeled 'A' with the description 'Event type = Item in "not supported" state'. Below this, under 'New condition', there is a dropdown menu for 'Event type' with the value 'Item in "not supported" state' selected. A tooltip for this dropdown lists four options: 'Item in "not supported" state', 'Low-level discovery rule in "not supported" state', 'Trigger in "unknown" state', and another partially visible option. The 'Enabled' checkbox is checked. At the bottom are 'Add' and 'Cancel' buttons.

不要忘记点击 *Add* 来实际列出条件块中的条件。

步骤 4

在操作选项卡中，输入问题消息的主题/内容。

点击 操作 模块中的 *New*，并且选择消息的一些接收者（用户组/用户）和用于传送的媒体类型（或“全部”）。

Actions

Action Operations Recovery operations

Default operation step duration (minimum 60 seconds)

Default subject

Default message
Host: {HOST.NAME}
Item: {ITEM.NAME}
Item key: {ITEM.KEY}
State: {ITEM.STATE}
Problem event: {EVENT.ID}
So far: {ESC.HISTORY}

Operations Steps Details S

1 - 2 **Send message to user groups:** Zabbix administrators via Email Ir

Operation details

Steps - (0 - infinitely)

Step duration (minimum 60 seconds, 0 - use action)

Operation type Send message

Send to User groups	User group	Action
	Zabbix administrators	Remove
	Add	
Send to Users	User	Action
	Add	
Send only to	<input type="button" value="Email"/>	
Default message	<input checked="" type="checkbox"/>	
Update Cancel		

点击操作细节块中的Add，添加 操作 模块中实际所包含的操作。

如果您希望收到多个通知，请设置操作步骤持续时间（发送消息之间的间隔）并添加其他操作。

步骤 5

恢复操作选项卡允许在项目恢复到正常状态时配置恢复通知。

输入恢复信息的主题/内容。

点击 操作 模块中的 New，并且选择消息的一些接收者（用户组/用户）和用于传送的媒体类型（或“全部”）。

Actions

Action Operations Recovery operations

Default subject	{ITEM.STATE}: {HOST.NAME}: {ITEM.NAME}
Default message	Host: {HOST.NAME} Item: {ITEM.NAME} Item key: {ITEM.KEY} State: {ITEM.STATE} Recovery event: {EVENT.RECOVERY.ID}
Operations	Details Notify all who received any messages regarding the problem before
Operation details	Operation type: Send recovery message ▾ Default message: <input checked="" type="checkbox"/> Update Cancel
Add Cancel	

点击操作细节块中的*Add*, 添加 操作 模块中实际所包含的操作.

步骤 6

完成后, 单击表单下方的“添加**”按钮。

就这样, 你完成了! 现在, 如果某些项目不受支持, 您可以期待收到Zabbix的第一个通知。

9 宏

概述

Zabbix支持许多在多种情况下使用的宏。宏是一个变量，由如下特殊语法标识：

1. `{MACRO}`

根据在上下文中， 宏解析为一个特殊的值。

有效地使用宏可以节省时间，并使Zabbix变地更加高效。

在一个的典型用途中，宏可以用于模板中。因此，模板的触发器可能命名为“Processor load is too high on {HOST.NAME}”。当这个模板应用与主机（如 Zabbix Server ）时，并且当触发器展示在监控页面上时，触发器的名称讲解析为“Processor load is too high on Zabbix server”。

宏可以在监控项键值参数中使用。宏只能用在监控项键值参数的一部分中，例如

`item.key[server_{HOST.HOST}_local]`。双引号参数不是必须的，因为Zabbix将处理任何模糊不清的特殊参数（如果这些参数存在于已解析的宏中）。

详细请查阅：

- 受支持的宏的完整列表；
- 宏 函数；
- 如何配置 用户宏。

1 宏函数

概述

宏函数能提供自定义宏值的功能。

有时候宏可能会解析为一个不一定易于使用的值。它可能很长，或包含你想提取的一个特殊感兴趣的子字符串。这在宏函数中是可以使用的。

宏函数的语法为：

```
1. {<macro>.<func>(<params>)}
```

其中：

- 这个参数为要定义的宏（例如 {ITEM.VALUE}）；
- 要应用的函数；
- 以逗号分隔的函数参数列表。如果他们以 空格 (空格), " 或者包含) , , 这些符号开始，则参数必须要引用。

例如：

```
1. {{ITEM.VALUE}}.regsub(pattern, output)}
```

受支持的宏函数

函数	描述	参数	受支持于
regsub (<pattern>, <output>)	通过正则表达式匹配提取的子字符串（区分大小写）。	pattern - 匹配的正则表达式 output - 输出的选项。 \1 - \9 占位符支持被正则表达式匹配的组 placeholders are supported for captured groups. 如果参数 pattern 是一个不正确的正则表达式，那么将返回 “UNKNOWN”。	{ITEM.VALUE} {ITEM.LASTVALUE}
iregsub (<pattern>, <output>)	通过正则表达式匹配提取的子字符串（区分大小写）。	pattern - 匹配的正则表达式 output - 输出得选项 \1 - \9 placeholders are supported for captured groups如果参数 pattern 是一个不正确的正则表达式，那么将返回 “UNKNOWN”。	{ITEM.VALUE} {ITEM.LASTVALUE}

如果在受支持的位置使用函数，但是应用于不支持宏函数得宏， 那么宏的计算结果为“UNKNOWN”。

如果在不支持宏函数的位置将宏函数应用于宏，则忽略该函数。

示例

关于宏函数可用于自定义宏值的方法，在下面的示例中说明，其中包含的“log line”作为接收值：

接收值	宏	输出
123Log line	<code>{{ITEM.VALUE}}.regsub(^[0-9]+, Problem)}</code>	Problem
123 Log line	<code>{{ITEM.VALUE}}.regsub("^([0-9]+)", "Problem")}</code>	Problem
123 Log line	<code>\$_ITEM.VALUE}.regsub("^([0-9]+)", "Problem ID: \1")}</code>	Problem ID: 123
Log line	<code>{{ITEM.VALUE}}.regsub(".", "Problem ID: \1")}</code>	Problem ID:
MySQL crashed errno 123	<code>\$_ITEM.VALUE}.regsub("^([A-Z]+)([0-9]+)", "Problem ID: \1_\2 ")}</code>	Problem ID: MySQL_123?
123 Log line	<code>\$_ITEM.VALUE}.regsub("([1-9]+", "Problem ID: \1")}</code>	UNKNOWN (invalid regular expression)



2 用户宏

概述

除了[支持开箱即用的宏](#)之外，Zabbix 还支持更灵活的用户宏。

用户宏可以在全局、模板和主机级别进行定义。这些宏具有一个特殊的语法：

1. `{$MACRO}`

用户宏可被用于：

- 监控项名称；
- 监控项键值参数；
- 触发器名称和描述；
- 触发器表达式参数和常量(详细查阅下文的 [示例](#))
- 许多其他位置 (详细查阅 [位置支持的宏](#))

宏名称中允许使用以下字符：**A-Z** , **0-9** , **_** , **.**。

Zabbix 根据以下优先级解析宏：

- 主机级别的宏 (首先检查)；
- 为主机的第一级别模板定义的宏 (即，直接链接到主机的模板)，按照模板 ID 来排序；
- 为主机的第二级别模板定义的宏，按照模板 ID 来排序；
- 为主机的第三级别模板定义的宏，按照模板ID来排序，等；
- 全局宏 (最后检查)。

换言之，如果一个主机不存在一个宏， Zabbix 将会尝试在级别递增的主机模板中找到它，如果仍然找不到，那么将使用全局宏 (如果全局宏存在的话)。

如果 Zabbix不能找到宏， 那么宏将不能被解析。

如果要定义用户宏，请转到Zabbix的前端页面的如下位置：

- 对于全局宏，请访问 管理 → 常规 → 右上角下拉菜单选择 “宏” ；
- 对于主机和模板级别的宏，请打开主机或模板属性并查看 宏 标签页面。

如果在模板的监控项或触发器使用用户宏，建议将该宏添加到模板，即使它被定义在全局级别上。这样的话，将模板导出至XML文件中，之后在其他系统中导入，那么在其他系统中使用也将达到预期的使用效果。

全局和主机宏的常用案例

- 利用具有主机特定属性的模板：密码、端口号、文件名称、正则表达式等；
- 运用全局宏进行全局的一键配置更改或微调。

示例

示例 1

在 “Status of SSH daemon” 监控项键值中使用主机级别的宏：

```
net.tcp.service[ssh,,{$SSH_PORT}]
```

该监控项可以分配给多个主机，前提是在这些主机上定义了 `{$SSH_PORT}` 的值。

示例 2

在 “CPU load is too high” 触发器上使用主机级别的宏：

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

这样的触发器将会在模板上创建，而不会在单个主机中编辑。

如果要使用数值作为函数参数（例如，`max(#3)`），则在宏定义中要包含井号（hash mark）例如：`SOME_PERIOD`
 $\Rightarrow \#3$

示例 3

在“CPU load is too high”触发器中使用了两个宏：

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

请注意，宏可以用作触发器函数的参数，在这个示例中为 `min()`。

在触发器表达式中，如果引用参数或者常量，则用户宏将会解析。如果引用主机、监控项键值、函数、操作或其他触发器表达式的话，他们将不会解析。

用户宏上下文

An optional context can be used in user macros, allowing to override the default value with context-specific one. 可以在用户宏中使用可选上下文，允许使用特定的上下文来重写默认的值。

User macros with context have a similar syntax: 具有上下文的用户宏具有类似的语法：

```
1. {$MACRO:context}
```

Macro context is a simple text value. The common use case for macro contexts would be using a low-level discovery `macro value` as a user macro context. For example, a trigger prototype could be defined for mounted file system discovery to use a different low space limit depending on the mount points or file system types. 宏上下文是一个简单的文本值。宏上下文的常见使用案例是使用自动发现宏值作为用户宏上下文。例如，根据文件系统的挂载点或文件系统类型，可以为挂载的文件系统自动发现定义自动发现触发器原型以使用不同的可使用空间阈值。

Only low-level discovery macros are supported in macro contexts. Any other macros are ignored and treated as plain text. 在宏上下文中只支持自动发现的宏。任何其他宏都将被忽略，并视为纯文本。

Technically, macro context is specified using rules similar to `item key` parameters, except macro context is not parsed as several parameters if there is a `,` character: 从技术上讲，宏上下文是使用类似于[监控项键值](#)参数的规则来指定的，除非有一个字符“，”，否则宏上下文是不被解析为几个参数：

- Macro context must be quoted with `"` if the context contains a `}` character or starts with a `"` character. Quotes inside quoted macros must be escaped with the `\` character. The `\` character itself is not escaped, which means it's impossible to have a quoted macro ending with the `\` character - the macro `{$MACRO:"a:\b\c\"}` is invalid.
- 宏的内容必须引用'“”，如果宏内容包含有'}'的字符或者从一个'“'字符. 那在引号中的引号必须使用'\\'字符进行转义从而不改变宏的内容，这也说明宏被引用后的字符'“'没有被转义则不能使用，宏 `{$MACRO:"a:\b\c\"}` 是无效的。
- The leading spaces in context are ignored, the trailing spaces are not. For example `{$MACRO:A}` is the same as `{$MACRO: A}` , but not `{$MACRO:A }` .
- 宏内容中前面的空格会被忽略，后面的空格不会忽略. 例如： `{$MACRO:A}` 和 `{$MACRO: A}` 相同，但不同于 `{$MACRO:A }` .
- All spaces before leading quotes and after trailing quotes are ignored, but all spaces inside quotes are not. Macros `{$MACRO:"A"}` , `{$MACRO: "A"}` , `{$MACRO:"A" }` and `{$MACRO: "A" }` are the same, but macros `{$MACRO:"A"}` and `{$MACRO:" A "}` are not.
- 宏变量中引号外面的空格可以被忽略，但是引号里面的空格不会忽略. 宏 `{$MACRO:"A"}` , `{$MACRO: "A"}` , `{$MACRO:"A" }` 和 `{$MACRO: "A "}` 相同，但是跟宏 `{$MACRO:"A"}` 和 `{$MACRO:" A "}` 不同。

The following macros are all equivalent, because they have the same context:

`{$MACRO:A}` , `{$MACRO: A}` and `{$MACRO:"A"}` . This is in contrast with item keys, where `key[a]` , `key[a]` and `key["a"]` are the same semantically, but different for uniqueness purposes.

以下的宏是一样的，因为它们表示的内容一样： `{$MACRO:A}` , `{$MACRO: A}` 和 `{$MACRO:"A"}` . 这与监控项相反， `key[a]` , `key[a]` 和 `key["a"]` 在语法上相同，但唯一性不同。

When context macros are processed, Zabbix looks up the macro with its context. If a macro with this context is not defined by host or linked templates, and it is not a global macro with context, then the macro without context is searched for.

当宏在使用时，zabbix会查看宏的背景，如果宏没有在主机或者模板上关联，或者有没有在全局宏中定义，那么宏不会被使用。

See [usage example](#) of macro context in a disk space trigger prototype and take limitation clause into consideration.

参考[usage example](#)章节中磁盘触发器原型关于宏变量的使用示例。

3 自动发现 (LLD) 宏

概述

有一种[自动发现 \(LLD\)](#) 函数中使用的宏类型为：

```
1. {#MACRO}
```

它是一个在LLD规则中使用的宏，并返回文件系统名称、网络接口和 SNMP OIDs。

这些宏可以用于创建监控项、触发器和图形原型。然后，当发现真实的文件系统、网络接口等，这些宏将被替换为真实的值，并且以这些值来创建真实的监控项、触发器和图形。

这些宏还用于在虚拟机[自动发现](#)中创建主机和主机组原型。

受支持的位置

LLD 宏可以用在：

- 用于监控项原型中：
 - names
 - key parameters
 - units
 - SNMP OIDs
 - IPMI sensor fields
 - calculated item formulas
 - SSH and Telnet scripts
 - database monitoring SQL queries
 - descriptions (从 2.2.0 开始支持)
- 用于触发器原型中：
 - names
 - expressions
 - URLs (从 3.0.0 开始支持)
 - descriptions (从 2.2.0 开始支持)
 - event tag names and values (从 3.2.0 开始支持)

- 用于图形原型中:
 - names
- 用于主机原型中 (从 2.2.0 开始支持):
 - names
 - visible names
 - host group prototype names
 - (详细查阅 [全部列表](#))

在上述所有位置, LLD 宏都可以在用户宏上下文中使用。

一些自动发现 (LLD) 宏在 Zabbix 中是已经预先内置的, 例如 {#FSNAME}、{#FSTYPE}、{#IFNAME}、{#SNMPINDEX}、{#SNMPVALUE} 这些宏。然而, 当你在创建[自定义](#)自动发现规则的时候, 遵守这些宏名称不是强制性的。所以, 你可以使用任何其他的 LLD 宏名称并引用该名称。

10 用户和用户组

概述

Zabbix 中的所有用户都通过 Web 前端去访问 Zabbix 应用程序。并为每个用户分配唯一的登陆名和密码。

所有用户的密码都被加密并储存于 Zabbix 数据库中。用户不能使用其用户名和密码直接登陆到 UNIX 服务器中，除非他们也被因此建立在 UNIX 中。可以使用 SSL 来保护 Web 服务器和用户浏览器之间的通讯。

使用一个灵活的 [用户权限架构](#) 可以限制和区分对以下内容的访问权限：

- 管理 Zabbix 前端的功能；
- 主机组中监视的主机。

最初 Zabbix 安装后有两个预先定义好的用户“Admin”和“guest”。其中，“guest”用户是未经验证身份的用户。在你使用“Admin”登陆前，你是“guest”用户。继续在 Zabbix 中[配置用户](#)。

1 配置用户

概述

根据以下步骤来配置一个用户：

- 在 Zabbix 前端页面跳转到 管理 → 用户；
- 在当前页面点击创建用户（或在用户名中编辑现有的用户）；
- 在窗口中编辑用户属性。

常规属性

在 用户 标签页包含常规用户属性：

The screenshot shows the 'User' configuration page in the Zabbix frontend. The 'User' tab is selected. The form contains the following fields:

- Alias:** Admin
- Name:** Zabbix
- Surname:** Administrator
- Groups:** Zabbix administrators (with a delete button) - A search input field 'type here to search' is also present.
- Password:** Change password
- Language:** English (en_US)
- Theme:** System default
- Auto-login:**
- Auto-logout:** 15m
- Refresh:** 30s
- Rows per page:** 50
- URL (after login):** (empty input field)

At the bottom are three buttons: **Update** (blue), **Delete**, and **Cancel**.

参数	描述
别名	唯一的用户名，用作登陆名。
名字	用户的名字（可选的）。如果此项不为空的话，则在确认信息和通知收件人信息中可见。
姓氏	用户的姓氏（可选的）。如果此项不为空的话，则在确认信息和通知收件人信息中可见。
	输入用户密码的两个字段。With an existing password, contains a Password

密码	button, clicking on which opens the password fields.
用户组	用户所属 用户组 的列表。 所属的用户组决定用户可以 访问 的主机组和主机。 点击添加进行添加用户组。
语言	Zabbix 前端的预言。 PHP扩展插件 <code>gettext</code> 是翻译所必须的。
主机	定义了前端的样式：系统默认 - 使用默认的系统设置蓝 - 标准的蓝色主题深色 - 另一种深色主题
自动登录	如果你希望Zabbix记住登录的信息并自动登录30天，请启用此选项。此选项需要用到浏览器的cookies。
自动登出 (最少90秒)	勾选此选项以设置用户在不活跃时间(最少90秒)后自动退出登录。
刷新 (秒)	设置图形、聚合图形、文本数据等的刷新速率。可以设置为0即禁止刷新。
每页行数	设置每个页面显示的行数
URL (登录后)	通过设置一个 URL ，当你登录 Zabbix 后，可以跳转到此 URL 。例如，设置为触发器的状态页面。

报警媒介

报警媒介标签页包含用户定义的所有报警媒介。报警媒介用于发送通知。点击添加将报警媒介分配给用户。

关于配置报警媒介类型详细的信息，请参阅[报警媒介类型](#)。

权限

权限标签页包含以下信息：

- 用户类型 (Zabbix User, Zabbix Admin, Zabbix Super Admin)。 用户不能改变自己的用户类型。
- 用户可以访问的主机组。默认情况下，“Zabbix User”和“Zabbix Admin”用户无权访问任何的主机组和主机。若要获得访问权限，需要将他们定义到访问相应主机组和主机的用户组中。

关于详细信息，请参阅[用户权限](#) 页面。

2 权限

概述

你可以定义相应的用户类型，然后通过将无特权用户包含在具有访问主机组数据权限的用户组中来区分 Zabbix 中的用户权限。

用户类型

用户类型定义了对前端管理菜单的访问级别以及对主机组数据的默认访问权限。

用户类型	描述
Zabbix 用户	用户可以访问“监测中”菜单页面。 默认情况下，用户无权访问任何资源。 必须明确分配对主机组的任何权限。
Zabbix 管理员	用户可以访问“监测中和配置”菜单页面。 默认情况下，用户无权访问任何主机组。 必须明确给出对主机组的任何权限。
Zabbix 超级管理员	用户可以访问所有内容：监测中、配置和管理菜单页面。 用户对所有主机组具有读写访问权限。 权限不能通过拒绝对特定主机组的访问来撤销。

主机组权限

只准许主机组级别的[用户组](#)访问 Zabbix 中的任何主机数据。

这意味着个人用户不能被直接授予对主机（或主机组）的访问权限。 只能通过被授予访问包含主机的主机组的用户组的一部分来授予对主机的访问权限。

3 用户组

概述

用户组可以为组用户组织目的和对数据分配权限。对于主机组的监控数据权限只能分配给用户组，而不是个人用户。

将一组用户和另一组用户的可用信息单独分离开，这样做通常会更有意义。因为这样可以通过用户进行分组，然后将不同的权限分配给主机组来实现。

一个用户可以属于任何数量的组。

配置

通过以下步骤配置用户组：

- 在 Zabbix 前端跳转到管理 → 用户组；
- 点击创建用户组（或者在用户组名上编辑现有的用户组）；
- 在表单中编辑用户组属性。

“用户组”标签页包含以下常规的用户组属性：

参数	描述
组名	唯一的组名。
用户	在组中的...这个方框内包含当前组内用户的列表。要将其他用户添加到此组中，请在其他组这个方框下选择相应的用户，并点击«按钮进行添加。
前端	如何对组内用户进行身份验证。系统默认 - 使用默认的验证方式Internal - 使用 Zabbix 验证。如

访问	如果设置了HTTP 验证，则忽略此项。停用的 - 被禁止访问 Zabbix GUI。a
已启用	用户组和组成员的状态。已选中 - 用户组和用户被启用。未选中 - 用户组和用户被禁用。
调试模式	选中此框将会激活用户的调试模式。

权限标签页允许你指定用户组访问主机组（和主机组内主机）数据：

Host group	Permissions
All groups	None
Clouds	Read-write Read Deny None
Discovered hosts	Read-write Read Deny None
Europe/Latvia/Riga/Zabbix servers (including subgroups)	Read-write Read Deny None
HQ	Read-write Read Deny None
HQ SNMP hosts (including subgroups)	Read-write Read Deny None
Linux servers (including subgroups)	Read-write Read Deny None
Network devices	Read-write Read Deny None
Templates	Read-write Read Deny None

type here to search Select Read-write Read Deny None
 Include subgroups
[Add](#)

主机组的当前权限显示在权限方框内。

如果主机组的当前权限由所有嵌套主机组继承，则由主机组名称后面的括号中的包含的子组文本指示。

你可以更改对主机组的访问级别：

- 读写 - 对主机组具有读写权限；
- 只读 - 对主机组具有只读权限；
- 拒绝 - 拒绝对主机组的访问；
- 无 - 不设置任何权限。

使用下面的选择字段选择主机组和对它们的访问级别（请注意，如果组已经在列表中，则选择无将从列表中删除主机组）。如果要包括嵌套主机组，请选中“包含子组”复选框。该字段是自动完成的，因此开始键入主机组的名称将提供匹配组的下拉列表。如果你希望查看所有主机组，请单击选择按钮。

来自多个用户组的主机访问

用户可以属于任意数量的用户组。这些组对主机可能具有不同的访问权限。

因此，重要的是要知道非特权用户将能够访问哪些主机。例如，让我们考虑如何在用户组A和B中的用户的各种情况下对“主机 X”（在主机组1中）的访问将受到影响。

- 如果“用户组 A ”只有“主机组 1 ”的只读权限，但“用户组 B ”拥有“主机组 1 ”的 读写权限，则这个用户将获得对“主机 X ”的读写权限。

“读写” 权限要优先于从 Zabbix 2.2 开始的“只读”权限。

- 在与上述相同的情况下，如果“主机组2”中的“主机 X ”同时拒绝“用户组 A ”或“用户组 B ”，那么“主机 X ”的访问将不可用，尽管“主机组 1 ”有读写权限。
- 如果“用户组 A ”没有定义权限，同时“用户组 B ”具有对“主机组 1 ”的读写权限，那么用户将获得对“主机 X ”的读写访问。
- 如果“用户组 A ”具有对“主机组 1 ”的拒绝权限，同时“用户组 B ”具有对“主机组 1 ”的读写权限，则用户访问“主机 X ”将被拒绝。

其他细节

- 如果一个具有对主机具有读写权限的管理级别用户无法访问Templates主机组，则具有读写访问主机的管理级用户将无法链接/取消链接模板。 使用只读访问Templates主机组，他将能够链接/取消链接到主机的模板，但是，模板列表中不会看到任何模板，也不能在其他地方使用模板。
- 具有只读访问主机的管理级用户将不会在配置页面的主机列表中看到主机；但是，在IT服务配置中可以访问主机触发器。
- 只要地图为空或只有图像，任何非Zabbix超级管理员用户（包括“guest”）都可以看到网络地图。 当主机、主机组或触发器被添加到地图时，权限被遵守。 这同样适用于屏幕和幻灯片。 无论权限如何，用户将看到任何没有直接或间接链接到主机的对象。 Any non-Zabbix Super Admin user (including 'guest') can see network maps as long as the map is empty or has only images. When hosts, host groups or triggers are added to the map, permissions are respected. The same applies to screens and slideshows as well. The users, regardless of permissions, will see any objects that are not directly or indirectly linked to hosts.

8. 服务监控

总览

服务监控(services monitoring)旨在帮助那些想要高级(业务)基础设施的监控的人。在许多情况下，我们关注的不是底层细节，比如磁盘空间不足、CPU 负载高等。我们关注的是IT部门提供的可用性的服务。我们还对确定IT基础设施薄弱的地方，IT各种服务级协定(SLA)，现有的IT基础设施的结构，以及其他的信息感兴趣

Zabbix 服务监控(services)对提到的问题提出了解决方案。

服务(services)是分层表示监控数据。

下面来看一个简单服务的例子：

```
1. IT Service
2. |
3. | -Workstations
4. | |
5. | | -Workstation1
6. | |
7. | | -Workstation2
8. |
9. | -Servers
```

该结构的每个节点都具有属性状态。根据选择算法进行状态计算并传播到上层节点。服务(services)最底层的服务是触发器。该节点的状态依赖于触发器的状态。

注意，触发器不分类或信息的严重程度不影响SLA计算。

配置

配置服务(services)，请访问：配置(Configuration)→服务(services)。

在该界面，您可以创建一个分层次的监控结构。最高的父节点服务是 'root'。您可以通过添加低级服务节点和各个节点服务创建下层层次结构。

Service	Action
root	Add child
▼ Servers	Add child
Server 1	Add child Delete
Server 2	Add child Delete
Server 3	Add child Delete
Server 4	Add child Delete
Server 5	Add child Delete

点击 *Add child* 添加服务(services)。可以单击其名称编辑一个现有的服务。您可以通过弹出的表单编辑服务属性。

配置一个服务(services)

服务选项卡包含通用服务属性：

Service	Dependencies	Time
Name	Server 1	
Parent service	SLA by service	Change
Status calculation algorithm	Problem, if at least one child has a problem	
Calculate SLA, acceptable SLA (in %)	<input type="checkbox"/> 99.9000	
Trigger	New host: Zabbix agent on New host is unreachable	Select
Sort order (0->999)	0	
Update Delete Cancel		

参数	说明
名称(<i>Name</i>)	服务的名字。
上层服务(<i>Parent service</i>)	父节点。
状态计算算法(<i>Status calculation algorithm</i>)	服务状态计算方法:不计算 - 不计算节点状态问题, 如果至少一个子节点有一个问题 - 只要一个子节点有异常, 该节点就异常。问题, 如果所有的子节点都有问题 - 当且仅当所有子节点都有异常, 该节点才异常。
计算SLA(<i>Calculate SLA</i>)	是否计算SLA的百分比。
可接受的SLA(%)	

计)(Acceptable SLA)	这个服务SLA百分比是可以接受的，用于报表。
触发器(Trigger)	选择关联的触发器: None - 没有关联的触发器触发器名称 - 连接到触发器，节点的状态取决于触发器状态在最底层的服务必须依赖触发器。(否则节点状态会显示不对。) \当触发被链接，其链接以前的状态是不计数的。
排序(Sort order)	显示的顺序，数字小的优先

依赖关系(Dependencies)选项卡可以看到该服务的所有子节点。单击Add单添加一个之前配置过的服务。

Depends on	SERVICES	SOFT	TRIGGER
Server 2		<input type="checkbox"/>	
Server 3		<input checked="" type="checkbox"/>	
Server 4		<input type="checkbox"/>	<input checked="" type="checkbox"/>
Add			

[Update](#) [Delete](#) [Cancel](#)

硬依赖和软依赖

服务可用性可以取决于若干其它服务，而不是仅仅一个。第一选项是直接添加为子节点。

然而，如果一些服务已经加入到其他的服务树，它不能被移动到这里作为子节点。如何创建？答案是“软”连接。添加服务并勾选 Soft 复选框。这样的服务可以留在原来的位置在树上，还依赖于其他服务。“软链接”的服务在服务树中显示是灰色的。另外，如果一个服务只有一个“软链接”的子节点，就可以删除此服务，而不用删除软链接的子节点。

时间(Time)选项卡用于设置服务的工作时间。

Service times	TYPE	INTERVAL	NOTE
No times defined. Work 24x7.			

New service time

Period type:

From: Time: :

Till: Time: :

[Add](#)

[Update](#) [Delete](#) [Cancel](#)

参数	说明
服务时间 (Service times)	默认地，所有的服务都将24x7x365操作。如果例外需要，添加新的服务。
新服务时间 (New service time)	服务时间:工作时间(Uptime) - 服务正常运行时间维护时间(Downtime) - 维护时间状态的时段内不会计算SLA百分比。一个时间-停机(One-time downtime) - 一次性的维护时间。维护时间状态的时段内不会计算SLA百分比。添加相应的时间。注意：服务时间仅影响其配置的服务。因此，父节点服务不会考虑在子节点服务上配置的服务时间（除非在父节点服务上配置相应的服务时间）。在前端计算服务状态和SLA时，会考虑服务时间。然而，不管服务时间如何，关于服务可用性的信息会被连续地插入到数据库中。

前端显示

服务(services)监控，去 [监控中\(Monitoring\) -> 服务\(services\)](#) .

9. Web 监控

概况

你可以使用 Zabbix 检查几个网站可用性方面。

如果要使用 Web 检测功能，必须在 [编译](#) Zabbix 的时候加入 curl(libcurl) 的支持。

要使用 Web 监控，您需要定义 web 场景。Web 场景包括一个或多个 HTTP 请求或“步骤”。Zabbix 服务器根据预定义的命令周期性的执行这些步骤。

从 Zabbix2.2 开始，Web 场景和 Items, Triggers 等一样，是依附在 Hosts/Templates 的。这意味着 web 场景也可以创建一个模板，然后应用于多个主机。

所有的 web 场景会收集下列数据：

- 整个场景中所有步骤的平均下载速度
- 失败的步骤数量
- 最后一次错误信息

对于 web 场景的所有步骤，都会收集下列数据：

- 平均下载速度
- 响应时间
- HTTP 状态码

更多详情，请参见 [web 监控项](#)。

执行 web 场景收集的数据保存在数据库中。数据自动用于图形、触发器和通知。

Zabbix 还支持获取 [HTML](#) 内容中是否存在设置的字符串。还可以模拟登陆动作和模拟鼠标单击。

Zabbix web 监控同时支持 HTTP 和 HTTPS。当运行 web 场景时，Zabbix 将选择跟踪重定向（请参见下面的选择跟踪重定向）。重定向硬编码的最大数量为 10（使用 curl 选项 `CURLOPT_MAXREDIRS`）。在执行 web 场景时，所有 Cookie 都会保存。

web 监控使用 HTTPS 协议请参阅 [已知问题](#)

配置 Web 场景

配置 web 场景：

- 转到：配置 (*Configuration*)->主机（或者 模板）
- 点击主机 (host)/ 模板 (template) 行中的 *Web*

- 点击右上角 创建 web 场景（或点击场景名字进行编辑现有的场景）
- 在场景的表单中输入参数

场景选项卡允许您配置此 Web 场景的通用参数。

Scenarios

Name: Availability of google

Application: Web checks

New application: Web checks

Update interval: 1m

Attempts: 1

Agent: Firefox 33.0 (Linux)

HTTP proxy: http://[user[:password]@]proxy.example.com[:port]

Variables	Name	Value	Remove
	name	= value	Remove
	Add		

Headers	Name	Value	Remove
	name	= value	Remove
	Add		

Enabled:

Add Cancel

场景参数：

参数	说明
主机 (Host)	场景所属的主机名或模板的名字。
名称 (Name)	唯一的场景名称。Zabbix 2.2 开始，这个名字支持用户宏和 {HOST.} 宏。
应用 (Application)	选择一个场景属于的应用。Web 场景监控项在 监测中 (Monitoring)–最新数据 (Latest data) 栏中将会分组在选择的应用中。
新的应用 (New application)	对场景创建一个新的应用。
更新间隔 (Update interval) (秒)	执行场景时间间隔，以秒为单位。
重试次数	尝试执行 web 场景中步骤的次数。对于网络问题（超时，没有连接，等等）Zabbix 可以多次重复执行步骤。这个数字对场景中的所有步骤都会生效。尝试次数最大可以设置为

(Attempts)	10，默认值为 1。注意：Zabbix 不会因为一个错误的响应代码或者期望的字符串没有出现就会触发这个重试。Zabbix 2.2 开始支持此参数。
代理 (Agent)	选择一个客户端。zabbix 会模拟选择的浏览器，当一个网站对不同的浏览器返回不同的内容的时候是非常有用的。zabbix 2.2 开始，这块可以使用用户自定义宏。
HTTP 代理 (HTTP proxy)	您可以指定要使用一个 HTTP 代理，使用格式 <code>http://[username[:password]@]proxy.mycompany.com[:port]</code> 默认使用 1080 端口。如果指定，代理将覆盖代理相关联的环境变量，比如 <code>httpProxy</code> 。 如果没有指定，那么代理将不会覆盖代理相关的环境变量。输入的值是通过“是 (as is)”，不需要进行完整性检查。你也可以输入 SOCKS 代理地址。如果您指定了错误的协议，连接会失败，项目将成为不受支持的。没有指定的协议，代理将被视为一个 HTTP 代理。注意：HTTP 代理仅支持简单身份验证。此字段中可以使用用户宏。 _Zabbix 2.2 开始支持此参数。
变量 (Variables)	可以在场景中的步骤 (URL, POST 变量) 中使用变量。它们具有以下格式： <code>{macro1}=value1{macro2}=value2{macro3}=regex:<regular expression></code> 例如： <code>{username}=Alexei{password}=kj3h5kJ34bd{hostid}=regex:hostid is ([0-9]+)</code> 然后可以在 {username}, {password} 和 {hostid} 的步骤中引用宏。 Zabbix 将自动将其替换为实际值。请注意，使用 <code>regex:</code> 的变量：需要一个步骤来获取正则表达式的值，因此提取的值只能应用于后续步骤。如果值部分以 <code>regex:</code> 开头，那么它之后的部分将被视为正则表达式，将搜索网页，如果找到，则将匹配存储在变量中。 注意，必须存在至少一个子组，以便可以提取匹配的值。Zabbix 2.2 开始支持变量中的正则表达式匹配。Zabbix 2.2 开始， <code>{HOST.}</code> 宏 和用户宏可以在此字段中使用。在查询字段或提交表单数据时，变量会自动进行 URL 编码，但使用 raw 方式提交数据或者直接在 URL 中使用时，必须手动进行 URL 编码
HTTP 头 (Headers)	执行请求时将发送的自定义的 HTTP headers。应使用与在 HTTP 协议中出现的语法相同的语法列出标题，可选地使用 <code>CURLOPT_HTTPHEADER</code> CURL 选项支持的一些其他功能。例如： <code>Accept-Charset=utf-8Accept-Language=en-USContent-Type=application/xml; charset=utf-8</code> 用户宏和 <code>{HOST.*}</code> 宏 和可以在此字段中使用。从 Zabbix 2.4 开始支持指定自定义头。
启用 (Enabled)	如果选中此复选框，则此场景处于启用状态，否则禁用。

注意，当编辑一个现有的场景时，会出现两个额外的按钮：

Clone	基于现有的场景的属性创建另一个场景。
Clear history and trends	删除场景的历史记录和趋势数据。这将使服务器在删除数据后立即执行场景。

如果 `HTTP proxy` 字段留空，使用 HTTP 代理的另一种方法是设置代理相关的环境变量。

对于 HTTP 检查 - 为 Zabbix 服务器用户设置 `http_proxy` 环境变量。例如，

`http_proxy=http://proxy_ip:proxy_port` .

对于 HTTPS 检查 - 设置 `HTTPS_PROXY` 环境变量。例如，

`HTTPS_PROXY=http://proxy_ip:proxy_port` . 通过运行 shell 命令可以获得更多详细信息：`# man curl` .

“步骤”选项卡允许您配置 Web 场景步骤。要添加 Web 场景步骤，请在 步骤 (Steps) 单击 添加 (Add)。

Steps	Name	Timeout	URL	Required	Status codes	Action
1: Home		15 sec	http://www.google.com	200		Remove
2: About		15 sec	http://www.google.com/intl/en/about	200		Remove

配置步骤

Name: Home

URL: http://www.google.com

Query fields

Name	Value
name	value

Add **Remove**

Post type: Form data (selected)

Post fields

Name	Value
name	value

Add **Remove**

Variables

Name	Value
name	value

Add **Remove**

Headers

Name	Value
name	value

Add **Remove**

Follow redirects:

Retrieve only headers:

Timeout: 15s

Required string:

Required status codes: 200

步骤参数：

参数	说明
名称 (Name)	唯一步骤名称。Zabbix 2.2 开始，该名称可以支持用户宏和 {HOST.*} 宏。
网址 (URL)	

1 Web 监控项

概述

在创建 Web 场景时，会自动添加一些新监控项以进行监控。

场景监控项

创建场景后，Zabbix 会自动添加以下监控项进行监控，将它们链接到所选应用程序。

监控项	说明
场景 <Scenario> 的 下载速度	此监控项将收集有关整个场景的下载速度（每秒字节数）的信息，即所有步骤的平均值。监控项 key: web.test.in[Scenario,,bps] 类型: Numeric(float)
场景 <Scenario> 的 失败步骤	此监控项将显示场景上失败的步骤的编号。如果所有步骤成功执行，则返回 0。监控项 key: web.test.fail[Scenario] 类型: Numeric(unsigned)
场景 <Scenario> 的 最后一个错误消 息	此监控项返回场景的最后一个错误消息文本。仅当场景具有失败步骤时，才会存储新值。如果所有步骤都正常，则不会收集新值。监控项 key: web.test.error[Scenario] 类型: Character

使用实际场景名称而不是“Scenario”。

添加的 Web 监控项将保留 30 天历史记录和 90 天趋势记录。

如果场景名称以双引号开头或包含逗号或方括号，则它将在监控项键中正确引用。在其他情况下，不会执行额外的引用。

这些监控项可用于创建触发器和定义通知条件。

例子 1

要创建“Web 场景失败”触发器，可以定义触发器表达式：

```
1. {host:web.test.fail[Scenario].last()}>>0
```

确保将“Scenario”替换为场景的真实名称。

例子 2

要在触发器名称中创建具有有用问题描述的“Web 场景失败”触发器，可以使用名称定义触发器：

```
1. Web scenario "Scenario" failed: {ITEM.VALUE}
```

和触发器表达式：

```
1. {host:web.test.error[Scenario].strlen()}>0 and {host:web.test.fail[Scenario].min()}>0
```

确保将“Scenario”替换为场景的真实名称。

例子 3

要创建“Web application is slow”触发器，可以定义一个触发器表达式：

```
1. {host:web.test.in[Scenario,,bps].last()}<10000
```

确保将“Scenario”替换为场景的真实名称。

场景步骤项

一旦创建步骤，Zabbix 会自动添加以下监控项进行监控，将它们链接到所选应用程序。

监控项	说明
场景 <Scenario> 中步骤 <Step> 的 下载速度	此监控项将收集关于步骤的下载速度（字节每秒）的信息。 监控项 key: web.test.in[Scenario,Step,bps] 类型: Numeric(float)
场景<Scenario> 中此步骤<Step>的 响应时间	此监控项将收集有关步骤的响应时间的信息（以秒为单位）。响应时间从请求开始计时，直到所有信息传输完毕。 监控项 key: web.test.time[Scenario,Step,resp] 类型: Numeric(float)
场景 <Scenario> 的步骤 <Step> 的 响应代码	此监控项将收集步骤的响应代码。 监控项 key: web.test.rspcode[Scenario,Step] 类型: Numeric(unsigned)

将分别使用实际场景和步骤名称而不是“Scenario”和“Step”。

添加的 Web 监控项将保留 30 天历史记录和 90 天趋势记录。

如果场景名称以双引号开头或包含逗号或方括号，则它将在监控项键中正确引用。在其他情况下，不会引用

这些监控项可用于创建触发器和定义通知条件。例如，要创建一个“Zabbix GUI 登录太慢”触发器，可以定义一个

```
1. {zabbix:web.test.time[ZABBIX GUI,Login,resp].last()}>3
```

2 场景示例

概述

本节介绍了如何使用 Web 监控的示例。

我们使用 Zabbix Web 监控来监控 Zabbix 的 Web 界面。我们想知道它是否可用、是否正常工作以及响应速度。为此，我们还必须使用我们的用户名和密码登录下。

场景

第 1 步

创建新的 Web 场景。

我们将添加一个场景来监控 Zabbix 的 Web 界面。该场景将执行多个步骤。

转到 配置 (Configuration) → 主机 (Hosts)，选择一个主机，然后在该主机行中单击 Web。然后单击 Create scenario.

Scenario Steps Authentication

Name: Zabbix frontend

Application: (dropdown)

New application: Zabbix frontend

Update interval: 1m

Attempts: 1

Agent: Firefox 33.0 (Linux)

HTTP proxy: http://[user[:password]@]proxy.example.com[:port]

Variables:

Name	Value
{user}	⇒ Admin
{password}	⇒ zabbix

Add

Headers:

Name	Value
name	⇒ value

Add

Enabled:

Add Cancel

在新的场景中，我们将场景命名为 `Zabbix frontend`，并为其创建一个新的 `Zabbix frontend` 应用 (application)。

注意，我们还将创建两个变量：`{user}`和`{password}`。

第 2 步

定义场景的步骤

单击 `Steps` 选项卡中的 `Add` 按钮添加单独的步骤。

Web 场景步骤 1

我们首先检查第一页响应是否正确，返回 HTTP 响应代码 200，并包含文本“Zabbix SIA”。

The screenshot shows the Zabbix configuration interface for creating a 'Web' step. The 'Name' field is set to 'First page'. The 'URL' field contains 'http://localhost/zabbix/index.php'. The 'Query fields' section has one entry: 'name = value'. The 'Post' section has one entry: 'name = value'. The 'Variables' section has one entry: 'name = value'. The 'Headers' section has one entry: 'name = value'. Under 'Follow redirects', the checkbox is checked. The 'Timeout' is set to 15. The 'Required string' is 'Zabbix SIA'. The 'Required status codes' is '200'. At the bottom are 'Add' and 'Cancel' buttons.

完成配置步骤后，单击 `Add`。

Web 场景步骤 2

我们继续登录 Zabbix 前端，我们通过我们在场景级别`{user}`和`{password}`上定义的宏（变量）来实现。

Name

URL

Query fields

Name	Value
<input type="text" value="name"/>	= <input type="text" value="value"/> <input type="button" value="Remove"/>

[Add](#)

Post

Name	Value
<input type="text" value="name"/>	= <input type="text" value="{user}"/> <input type="button" value="Remove"/>
<input type="text" value="password"/>	= <input type="text" value="{password}"/> <input type="button" value="Remove"/>
<input type="text" value="enter"/>	= <input type="text" value="Sign in"/> <input type="button" value="Remove"/>

[Add](#)

Variables

Name	Value
<input type="text" value="{sid}"/>	= <input type="text" value="!gex:name='sid' value='([0-9a-zA-Z]{16})'"/> <input type="button" value="Remove"/>

[Add](#)

Headers

Name	Value
<input type="text" value="name"/>	= <input type="text" value="value"/> <input type="button" value="Remove"/>

[Add](#)

Follow redirects

Retrieve only headers

Timeout

Required string

Required status codes

注意，Zabbix 前端在登录时使用 JavaScript 重定向，因此首先我们必须登录，只有在下一步的步骤中，我们才能检查登录功能。此外，登录步骤必须使用完整的 URL 以获取 `index.php` 文件

还要注意我们如何使用正则表达式的变量语法获取 `{sid}` 变量（会话 ID）的内容：`<?nowiki>?regex : name = "sid" value = "([0-9a-zA-Z]{16})" </?nowiki>`。步骤 4 中会使用此变量。

Web 场景步骤 3

登录后，我们现在应该验证一下是否登陆成功。为此，我们检查一个仅在登录后可见的字符串 - 例如 **Administration**（管理）。

Name

URL

Query fields

Name	Value
<input type="text" value="name"/>	<input type="text" value="value"/> <input type="button" value="Remove"/>

[Add](#)

Post

<input checked="" type="button" value="Form data"/> <input type="button" value="Raw data"/>	
Name	Value
<input type="text" value="name"/>	<input type="text" value="value"/> <input type="button" value="Remove"/>

[Add](#)

Variables

Name	Value
<input type="text" value="name"/>	<input type="text" value="value"/> <input type="button" value="Remove"/>

[Add](#)

Headers

Name	Value
<input type="text" value="name"/>	<input type="text" value="value"/> <input type="button" value="Remove"/>

[Add](#)

Follow redirects

Retrieve only headers

Timeout

Required string

Required status codes

Web 场景步骤 4

现在我们已经验证了前端是可访问的，我们可以登录并检索登录的内容，我们也应该注销 - 否则 Zabbix 数据库将被大量的开放会话记录所污染。

Name

URL Parse

Query fields

Name	Value
reconnect	= 1 Remove
sid	= {sid} Remove

[Add](#)

Post

[Form data](#) [Raw data](#)

Name	Value
name	= value Remove

[Add](#)

Variables

Name	Value
name	= value Remove

[Add](#)

Headers

Name	Value
name	= value Remove

[Add](#)

Follow redirects

Retrieve only headers

Timeout

Required string

Required status codes

Web 场景步骤 5

我们可以通过查找用户名字符串来检查我们是否已经注销了。

Name: Check logout

URL: http://localhost/zabbix/index.php

Query fields

Name	Value
name	= value

Add

Post

Form data (selected)

Name	Value
name	= value

Add

Variables

Name	Value
name	= value

Add

Headers

Name	Value
name	= value

Add

Follow redirects:

Retrieve only headers:

Timeout: 15

Required string: Username

Required status codes: 200

完成步骤配置

Web 场景步骤的完整配置应如下所示：

Scenario Steps Authentication						
Steps	Name	Timeout	URL	Required	Status codes	
1:	First page	15 sec	http://localhost/zabbix/index.php	Zabbix SIA	200	
2:	Log in	15 sec	http://localhost/zabbix/index.php		200	
3:	Check login	15 sec	http://localhost/zabbix/index.php	Administration	200	
4:	Log out	15 sec	http://localhost/zabbix/index.php		200	
5:	Check logout	15 sec	http://localhost/zabbix/index.php	Username	200	
Add						

第 3 步

保存 Web 监控场景。

通过以下方式查看场景 Monitoring → Web：

Web monitoring

Host	Name	Number of steps	Last check	Status
Zabbix server	Zabbix frontend	5	2017-03-24 08:32:50	OK

Displaying 1 of 1 found

单击场景名称以查看更详细的统计信息：

Details of web scenario: Zabbix frontend

Step	Speed	Response time	Response code	Status
First page	33.6 Kbps	92.3ms	200	OK
Log in	48.16 Kbps	168.4ms	200	OK
Check login	27.02 Kbps	300.2ms	200	OK
Log out	13.07 Kbps	237.2ms	200	OK
Check logout	33.02 Kbps	93.9ms	200	OK
TOTAL		892ms		OK

Zoom: 5m 15m 30m 1h 2h 3h 6h 12h 1d 3d All

2017-03-29 08:16 - 2017-03-29 09:16 (now!)

Filter ▲

Zabbix frontend: Speed (1h)

Download speed for step "First page" of scenario "Zabbix frontend". [avg] last min avg max 33.6 Kbps 13.15 Kbps 27.54 Kbps 33.9 Kbps
 Download speed for step "Log in" of scenario "Zabbix frontend". [avg] last min avg max 48.16 Kbps 26.94 Kbps 41.85 Kbps 48.86 Kbps
 Download speed for step "Check login" of scenario "Zabbix frontend". [avg] last min avg max 27.02 Kbps 27.02 Kbps 41.95 Kbps 49.41 Kbps
 Download speed for step "Log out" of scenario "Zabbix frontend". [avg] last min avg max 13.07 Kbps 9.22 Kbps 16.62 Kbps 19.34 Kbps
 Download speed for step "Check logout" of scenario "Zabbix frontend". [avg] last min avg max 33.02 Kbps 20.62 Kbps 33.29 Kbps 40.25 Kbps

Data from history. Generated in 0.00 sec.

Zabbix frontend: Response time (1h)

Response time for step "First page" of scenario "Zabbix frontend". [avg] last min avg max 92.3ms 91.4ms 117.3ms 235.7ms
 Response time for step "Log in" of scenario "Zabbix frontend". [avg] last min avg max 168.4ms 166ms 196.62ms 301ms
 Response time for step "Check login" of scenario "Zabbix frontend". [avg] last min avg max 300.2ms 164.1ms 196.82ms 300.2ms
 Response time for step "Log out" of scenario "Zabbix frontend". [avg] last min avg max 237.2ms 160.3ms 189.92ms 336ms
 Response time for step "Check logout" of scenario "Zabbix frontend". [avg] last min avg max 93.9ms 77ms 94.64ms 150.3ms

Data from history. Generated in 0.00 sec.

10. 虚拟机监控

概述

从 Zabbix 2.2.0 开始支持对 VMware 的监控。

Zabbix 可以使用 low-level discovery 自动发现 VMware hypervisors 和虚拟机，并根据事先定义的主机原型，为这些虚拟机建立 Host，添加监控。

Zabbix 中默认提供了几个模板，可以直接用来监控 VMware vCenter 或 ESX hypervisor。

支持 VMware vCenter 或 vSphere 版本最低为 4.1。

细节

虚拟机监控分两个步骤完成。首先，Zabbix 是通过 `vmware collector` 进程来监控虚拟机。这些进程通过 SOAP 协议从 VMware Web 服务获取必要的信息，对其进行预处理并存储到 Zabbix server 共享内存中。然后，zabbix pollers 通过 zabbix 简单检查 [VMware keys](#) 来检索这些数据。

从 Zabbix 2.4.4 开始，收集的数据分为两种类型：VMware 配置数据和 VMware 性能数据。这两种类型都由 `vmware collectors` 进程独立收集。因此，建议启用比受监控的 VMware 服务更多的收集器。否则，检索 VMware 性能统计信息可能会由于检索 VMware 配置数据而延迟（比较大型的环境，需要一段时间）。

目前基于 VMware 性能统计信息只有数据存储，网络接口和磁盘设备统计信息和自定义性能计数器项。

配置

要使虚拟机监控正常工作，[编译](#) Zabbix 时应加上 `-with-libxml2` 和 `-with-libcurl` 编译选项。

以下配置文件参数可用于调整虚拟机监控：

- **StartVMwareCollectors** - vmware 收集器实例的数量。此值取决于要监控的 VMware 服务的数量。在大多数情况下，这应该是： $servicenum < StartVMwareCollectors < (servicenum * 2)$ 其中 `servicenum` 是 VMware 服务的数量。例如：如果您有 1 个 VMware 服务要将 `StartVMwareCollectors` 设置为 2，那么如果您有 3 个 VMware 服务，请将其设置为 5。请注意，在大多数情况下，此值不应小于 2，不应大于 VMware 数量的 2 倍服务。还要记住，此值还取决于 VMware 环境大小和 `VMwareFrequency` 和 `VMwarePerfFrequency` 配置参数（请参阅下文）。
- **VMwareCacheSize**
- **VMwareFrequency**
- **VMwarePerfFrequency**
- **VMwareTimeout**

有关更多详细信息，请参阅 [server](#) and [proxy](#) 的配置文件页面。

自动发现

Zabbix 可以使用 low-level discovery 规则自动发现 VMware hypervisors 和虚拟机。

Name	Discover VMware hypervisors							
Type	Simple check							
Key	vmware.hv.discovery[\$URL]							
User name	{\$USERNAME}							
Password	{\$PASSWORD}							
Update interval (in sec)	3600							
Custom intervals	<table border="1"> <thead> <tr> <th>TYPE</th> <th>INTERVAL</th> <th>PERIOD</th> </tr> </thead> <tbody> <tr> <td>Flexible</td> <td>Scheduling</td> <td>50</td> <td>1-7,000</td> </tr> </tbody> </table>	TYPE	INTERVAL	PERIOD	Flexible	Scheduling	50	1-7,000
TYPE	INTERVAL	PERIOD						
Flexible	Scheduling	50	1-7,000					
Add								
Keep lost resources period (in days)	30							
Description	Discovery of hypervisors.							
Enabled	<input checked="" type="checkbox"/>							

上面截图中的发现规则键是 `vmware.hv.discovery[$URL]`。

主机原型

可以使用 low-level discovery 规则创建主机原型。当自动发现虚拟机时，这些原型成为真正的主机。原型在被发现之前，除了来自链接模板的那些，不能有自己的监控项和触发器。发现的主机将属于现有主机，并将采根据主机的 IP 进行主机配置。

The screenshot shows the 'Discovery rules' section of the Zabbix interface. At the top, there are tabs for 'All templates / Template Virt VMware', 'Applications 3', 'Items 3', 'Triggers', 'Graphs', 'Screens', and 'Discovery'. The 'Discovery' tab is selected. Below the tabs, there is a header row with columns for 'NAME ▲', 'ITEMS', 'TRIGGERS', 'GRAPHS', and 'HOSTS'. Three items are listed under the 'NAME' column:

- Discover VMware clusters (Item prototypes 1)
- Discover VMware hypervisors (Item prototypes)
- Discover VMware VMs (Item prototypes)

Each item row also includes 'Trigger prototypes', 'Graph prototypes', and 'Host prototypes' columns.

在主机原型配置中，LLD 宏用于主机名，可见名称和主机组原型字段。与现有主机组，模板链接和加密链接 (encryption) 等选项。

The screenshot shows the 'Host' configuration dialog. It has tabs for 'Host', 'Groups', 'Templates', 'Host inventory', and 'Encryption'. The 'Host' tab is selected. There are two input fields: 'Host name' containing '#HV.UUID' and 'Visible name' containing '#HV.NAME'. A checkbox labeled 'Create enabled' is checked. At the bottom are 'Add' and 'Cancel' buttons.

如果选中 *Create enabled*，则主机将添加为启用状态。如果未选中，将添加主机，但处于禁用状态。

在主机列表中，自动发现的主机将创建它们的发现规则的名称作为前缀。可以手动删除发现的主机。发现的主机也将根据发现规则的 丢失保留周期 *Keep lost resources period* (以天为单位) 值自动删除。除了启用 / 禁用主机和主机清单外，大多数配置选项都是只读的。发现的主机不能有自己的主机原型。

准备使用的模板

Zabbix 中默认提供了几个现成的模板，用于监控 VMware vCenter 或 ESX hypervisor。

这些模板包含事先定义的 LLD 规则以及用于监视虚拟安装的内置检查。

请注意，“Template Virt VMware”模板应用于 VMware vCenter 和 ESX hypervisor 监控。“Template Virt VMware Hypervisor”和“Template Virt VMware Guest”模板由自动发现使用，通常设置为自动链接到主机。

Templates

TEMPLATES	APPLICATIONS	ITEMS	TRIGGERS	GRAPHS	SCREENS	DI
Template Virt VMware Hypervisor	Applications 6	Items 19	Triggers	Graphs	Screens	Dis
Template Virt VMware Guest	Applications 8	Items 17	Triggers	Graphs	Screens	Dis
Template Virt VMware	Applications 3	Items 3	Triggers	Graphs	Screens	Dis

如果您的服务器从 2.2 之前的版本升级并且没有此类模板，您可以手动导入，从社区页面下载 [官方模板](#)。但是，这些模板依赖于 `VMware VirtualMachinePowerState` 和 `VMware` 状态值 映射，因此有必要首先创建这些值映射（使用 [SQL 脚本](#)，手动或从 XML 导入）

主机配置

要使用 VMware 简单检查，主机必须定义以下用户宏：

- `{$URL}` - VMware 服务 (vCenter or ESX hypervisor) SDK URL (<https://servername/sdk>)。
- `{$USERNAME}` - VMware 服务用户名
- `{$PASSWORD}` - VMware 服务`{$ USERNAME}`用户密码

例子

以下示例演示如何在 Zabbix 上快速设置 VMware 监控：

- 编译 zabbix server 时添加依赖项 (`-with-libxml2` 和 `-with-libcurl`)
- 将 Zabbix server 配置文件中的 `StartVMwareCollectors` 选项设置为 1 或更多
- 创建新主机
- 设置 VMware 身份验证所需的主机宏：

Host Macros

MACRO	VALUE
{\$PASSWORD}	⇒ <password>
{\$URL}	⇒ <url>
{\$USERNAME}	⇒ <username>

Add Cancel

- 将 VMware 服务模板链接到主机：

Linked templates

NAME	ACTION
Template Virt VMware	Unlink

Link new templates

type here to search Select

Add Cancel

- 单击 Add 按钮保存主机

扩展日志

使用调试级别 5 进行详细调试时，VMware 收集器收集的数据会记录到日志中。此级别可以在 `server` 和 `proxy` 配置文件中设置，或使用运行时控制选项（`-R log_level_increase="vmware collector,N"`，其中 N 是过程编号）。以下示例说明如果已设置调试级别为 4，扩展日志如何启动：

```
1. 提高所有 vmware 收集器的日志级别：
2. shell> zabbix_server -R log_level_increase="vmware collector"
```

```
1. 提高第二个 vmware 收集器的日志级别：
2. shell> zabbix_server -R log_level_increase="vmware collector,2"
```

10. 虚拟机监控

如果不需要对 VMware 收集器数据进行扩展日志，可以使用 `-R log_level_decrease` 选项进行停止。

虚拟机 discovery 关键字段

下表列出了虚拟机 discovery 键返回的内容。

项目键	字段	检索内容
描述		
vmware.cluster.discovery		
对于 VMware 集群的 discovery 。	{#CLUSTER.ID}	集群 ID。
{#CLUSTER.NAME}	集群名称。	
vmware.hv.discovery		
执行 hypervisor 的 discovery 。	{#HV.UUID}	唯一的 hypervisor 的 ID
{#HV.ID}	Hypervisor 的 ID (由 HostSystem 管理)	
{#HV.NAME}	Hypervisor 的名字	
{#CLUSTER.NAME}	群集名称，可能为空。	
{#DATACENTER.NAME}	数据中心名称。	
vmware.hv.datastore.discovery		
执行 hypervisor 数据存储库的 discovery 。请注意，多个 hypervisor 可以使用相同的数据存储 (datastore)。	{#DATASTORE}	数据存储名称。
vmware.vm.discovery		
执行虚拟机的 discovery 。	{#VM.UUID}	唯一虚拟机 ID。
{#VM.ID}	虚拟机 ID (由 VirtualMachine 管理)。	
{#VM.NAME}	虚拟机名。	
{#HV.NAME}	Hypervisor 名称。	
{#CLUSTER.NAME}	群集名称，可能为空。	
{#DATACENTER.NAME}	数据中心名称。	
vmware.vm.net.if.discovery		
执行虚拟机网络接口的 discovery 。	{#IFNAME}	网络接口名称。
vmware.vm.vfs.dev.discovery		
执行虚拟机磁盘设备的 discovery 。	{#DISKNAME}	磁盘设备名称。

vmware.vm.vfs.fs.discovery		
执行虚拟机文件系统的 discovery 。	{#FSNAME}	文件系统名称。



11. 维护

概述

您可以在Zabbix中为主机和主机组定义维护周期。有两种维护类型 - 继续对目标进行监控数据的收集和停止对目标进行监控数据的收集。

在“使用数据收集(with data collection)“的维护期间，触发器照常处理，并且在需要时创建事件。但是，如果在动作(Action)中配置中选择了处于维护期间的暂停操作(Maintenance status = not in “maintenance”)选项，则维护中的主机会暂停进行下步操作。在这种情况下，只要维护周期持续，将会忽略包括发送通知或远程命令的步骤。

例如，如果动作步骤计划在一个服务异常开始后的0, 30和60分钟，并且在异常出现后在10分钟至40分钟有半小时长的维护，则步骤2和步骤3将在半小时后执行，或在60分钟和90分钟(问题仍然存在)。类似地，如果在维护期间出现问题，动作操作将在维护周期之后开始

要在维护期间正常(无延迟)接收问题通知，必须在动作(action)配置中的选项中取消选中暂停操作(Maintenance status = not in “maintenance”)。

如果有一个主机(在触发器表达式中使用)不在维护模式，Zabbix将发送问题通知。

在维护期间，Zabbix服务器必须运行。定时器(Timer)进程负责在每分钟0秒时将主机状态切换到维护中或从维护状态中去除。代理(proxy)将始终收集数据，而不管维护类型(包括“无数据”维护)。如果设置了“无数据收集”，则服务器稍后将忽略从代理上报的数据。

当“无数据(no data)”维护结束时，在下一个检查之前，触发器不会触发nodata()函数。

如果在主机处于维护期间添加日志监控项，并且维护结束，则将仅收集维护结束后的日志文件条目。

为了确保定期维护周期(每天，每周，每月)的按照预期的时间执行，需要对Zabbix的所有部分使用通用时区。

配置

配置维护周期：

- 转到：配置(Configuration) → 维护(Maintenance)
- 单击 创建维护期(或现有维护期的名称)

维护(Maintenance)选项卡包含常规维护期属性：

Maintenance Periods Hosts & Groups

Name: Maintenance period

Maintenance type: With data collection (selected)

Active since: 2015 - 01 - 01 00 : 00

Active till: 2017 - 01 - 01 00 : 00

Description: We break and fix things at this time.

Add Cancel

参数	说明
名称(<i>Name</i>)	维护期的名称。
维护类型(<i>Maintenance type</i>)	可以设置两种类型的维护：使用数据收集(With data collection) - 服务器将在维护期间收集数据，触发器会正常工作 无数据收集(No data collection) - 服务器在维护期间不会收集数据
启动时间(<i>Active since</i>)	自启动的日期和时间变为维护状态。 注意：单独设置此时间不会激活维护期；为此转到期间(<i>Periods</i>)选项卡。
结束时间(<i>Active till</i>)	维护状态直到结束时间停止。
说明(<i>Description</i>)	维护周期说明。

期间(**Periods**)选项卡允许您定义维护进行的准确日期和时间。单击 新建(*New*) 打开一个维护期间窗体，您可以在其中定义时间 - 每日，每周，每月或一次性维护。

Periods	PERIOD TYPE	SCHEDULE	PERIOD
	Weekly	At 15:00 on every Friday of every week	1h

Maintenance period

Period type: Weekly

Every week(s): 1

Day of week:

- Monday
- Tuesday
- Wednesday
- Thursday
- Friday
- Saturday
- Sunday

At (hour:minute): 15 : 0

Maintenance period length: 0 Days 1 Hours 0 Minute

[Add](#) [Cancel](#)

对话框中具有 **每天/每周** 参数，默認為1. 将其设置为2将使每两天或每两周进行一次维护等。开始日期或周是根据开始时间(*Active since*) 设置的。

例如，将 开始时间(*Active since*) 设置为2013-09-06 12:00以及每两天在23:00有1个小时的维护周期，这样的话，第一个维护期从2013-09-06在23:00开始，而第二个维修期将于2013-09-08 23:00开始。或者，相同的开始时间(*Active since*)，每两天在01:00的一个小时长的维护时间，第一维护期将从2013-09-08在01:00开始，第二维护期在2013-09-10在01:00执行。

主机和组(**Hosts & Groups**)选项卡允许您对选择的主机和主机组进行维护。

Maintenance Periods Hosts & Groups

Hosts in maintenance

In maintenance

New host

Zabbix server

Other hosts | Group

Groups in maintenance

In maintenance

Zabbix servers

Other groups

Discovered hosts SNMP hosts

Update Clone Delete Cancel

指定父主机组将隐式选择所有嵌套的主机组。因此，也将在嵌套组的主机上执行维护。

界面显示

主机名旁边的橙色扳手图标表示此主机正在进行维护。通过以下方式查看，监视(*Monitoring*) → 仪表板(*Dashboard*)，监视(*Monitoring*) → 触发器(*Triggers*) 和 清单(*Inventory*) → 主机(*Hosts*) → 主机详细信息



当鼠标指针位于图标上方时，将显示维护详细信息。

仪表板中主机在维护中状态是否显示可以在仪表板过滤功能中进行设置。

此外，维护中的主机显示的是橙色背景，通过以下方式查看，在 监控(*Monitoring*) → 地图(*Maps*) 和 配置(*Configuration*) → 主机(*Hosts*) 中，其状态显示为“在维护(*In maintenance*)”。

12. 正则表达式

概述

Zabbix支持POSIX 正则表达式

在Zabbix中有两种方法使用正则表达式

- 手动输入正则表达式
- 使用在Zabbix中创建的全局正则表达式

正则表达式

你可以在受支持位置中手动输入正则表达式。请注意，表达式不能以@开头，因为该符号在Zabbix中用于引用全局正则表达式。

全局正则表达式

有一个高级编辑器用于在Zabbix前端中创建和测试复杂的正则表达式。

一旦以这种方式创建了正则表达式，它可以在前端的多个地方通过引用其名称（前缀为@）来使用，例如 @*mycustomregexp*

创建全局正则表达式

- 进入：管理 (Administration) → 一般 (General)
- 从下拉列表中选择正则表达式 (Regular expressions)
- 点击新的正则表达式 (New regular expression)

表达式选项卡允许设置正则表达式名称和添加子表达式。

Name	EXPRESSION TYPE	EXPRESSION	DELIMITER	CASE SENSITIVE	ACTION
Result is FALSE	▼	^lo\$	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	Remove
Result is FALSE	▼	^Software Loopback Interface	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	Remove

[Add](#) [Cancel](#)

参数	说明
名称 (Name)	设置正则表达式名称。允许任何Unicode字符。

表达式 (Expressions)	单击表达式区域中的添加 (Add) 以添加新的子表达式。	
	表达式类型 (Expression type)	选择表达式类型:字符串已包含 (Character string included) - 匹配子字符串包含任何字符串 (Any character string included) - 匹配逗号分隔列表中的任何子字符串字符串未包含 (Character string not included) - 匹配除了子字符串之外的任何字符串结果为真 (Result is TRUE) - 匹配正则表达式结果为假 (Result is FALSE) - 不匹配正则表达式
表达式 (Expression)	输入子字符串/正则表达式。	

自Zabbix 2.4.0开始，表达式中的正斜杠作为字符处理，而不是分隔符。这样可以保存包含斜杠的表达式，而以前它会产生错误。

Zabbix中的自定义正则表达式名称可能包含逗号，空格等。在引用时可能导致错误解释的情况下（例如，监控项键的参数中的逗号），整个引用可以放在引号中，如下所示：“@我的自定义regexp为purpose1, purpose2”。\正则表达式名称不能在其他位置引用（例如，在LLD规则属性中）。

自定义正则表达式可能由多个子表达式组成，并且可以通过提供测试字符串在测试选项卡中进行测试

Result	Expression type	Expression	Result
Result is FALSE	^Software Loopback Interface		TRUE
Result is FALSE	^(In)?[Ll]oop[Bb]ack[0-9_.]*\$		TRUE
Result is FALSE	^NULL[0-9.]*\$		TRUE
Result is FALSE	^[Ll]o[0-9.]*\$		FALSE
Result is FALSE	^[\$s]ystem\$		TRUE
Result is FALSE	^Nu[0-9.]*\$		TRUE
Combined result			FALSE

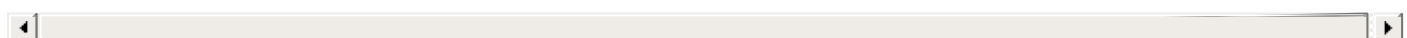
结果显示每个子表达式的状态和自定义表达式的状态

正则表达式支持位置

位置	正则表达式支持	全局正则表达式支持	注释
宏函数			
	regsub()	是	否
iregsub()			模式 (pattern) 参数

触发器函数				
	count()	是	是	模式 (pattern) 参数, 如果 (操作员) operator 参数是 regexp or iregexp
log.eventid()	模式 (pattern) 参数			
iregexp()				
regexp()				
自动发现				
	是	否	过滤 (Filter) 字段	
Web 监测				
	是	否	变量以 regex: 为前缀 Required string 字段的变量	
Zabbix 代理 (agent) 项				
	eventlog[]	是	是	regexp, severity, source, eventid 参数
log[]	regexp 参数			
log.count[]				
logrt[]	是/否	regexp 参数支持两老师, file_regexp 参数只支持 non-global 表达式		
logrt.count[]				
proc.cpu.util[]	否	cmdline 参数		
proc.mem[]				
proc.num[]				
sensor[]	device 和 sensor 参数在 Linux 2.4			
system.hw.macaddr[]	interface 参数			
system.sw.packages[]	package 参数			
vfs.dir.size[]	regex_incl 和 regex_excl			

	参数
vfs.file.regexp[]	regexp 参数
vfs.file.regmatch[]	
web.page.regexp[]	
SNMP 捕获	
	snmptrap[] 是 是 regexp 参数
图标映射 (Icon mapping)	是 是 <i>Expression</i> 字段



13. 事件确认

概述

Zabbix中的问题事件可以由用户确认。

如果用户获得有关问题事件的通知，可以访问Zabbix前端，从事件导航到确认屏幕并确认问题。当他们确认时，他们可以输入他们的评论，说他们正在努力，或者他们可能会写一些相关的描述。

这样，如果其他系统用户同样的问题，他们会立即看到是否已经被确认并且到目前为止的评论。

以这种方式，可以以更协调的方式进行解决与多个系统用户的问题的工作流程。

在定义 [动作操作 \(action operations\)](#) 时确认状态也使用。例如，您可以定义仅当某个事件未被确认一段时间后才将通知发送到较高级别的管理员

要确认事件，用户至少必须具有对相应触发器的读取权限。

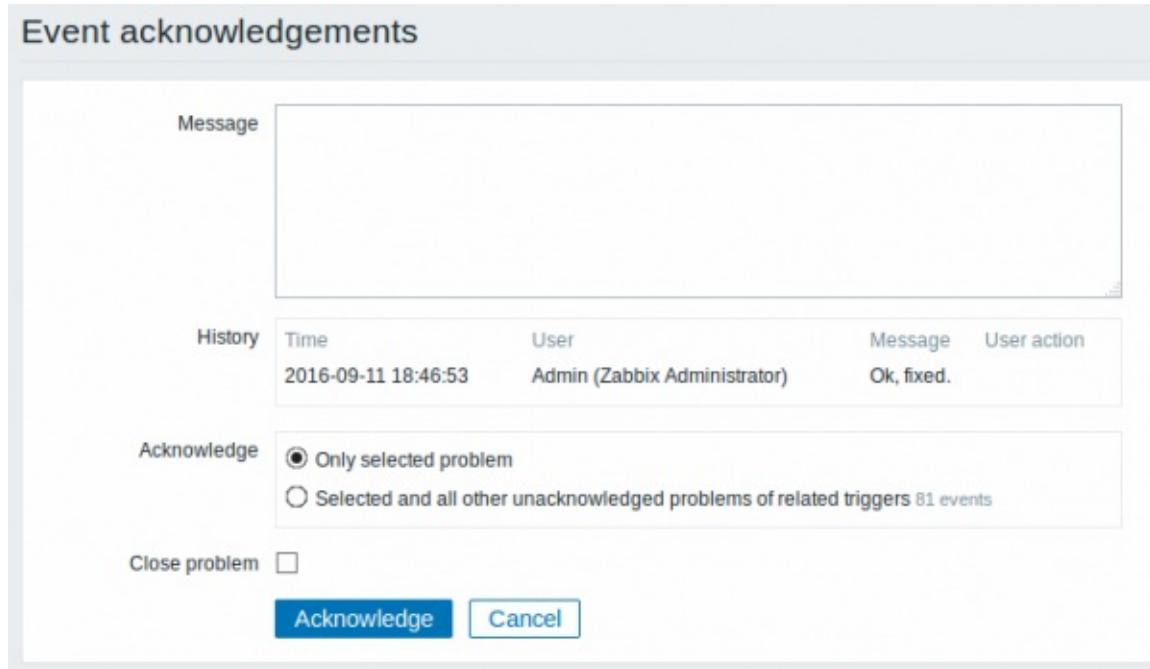
确认屏幕

问题确认状态显示在：

- 监测中 → 仪表板（最近20个问题 和 系统状态 小部件）
- 监测中 → 问题
- 监测中 → 问题 → 事件详情
- 监测中 → 概述（选择触发器）
- 监测中 → 触发器
- 监测中 → 聚合图形（使用 主机群组问题，主机问题，系统状态 和 触发器概览 资源）

确认列包含“是”或“否”，分别表示确认或未确认的问题。“是”也可以有一个数字，表示到目前为止该问题的意见的数量。

“是”和“否”都是链接。点击它们将进入确认屏幕。



要确认问题，请输入您的评论，然后单击确认。 您可以选择确认所选择的事件或所选事件以及触发器的所有其他未确认的问题。

任何先前的问题评论都会显示在消息区域的下方。

管理 → 常规中可以打开/关闭前台的事件确认。 关闭时，除了操作操作中的操作条件外，与确认相关的控件被隐藏。 此外，当打开/关闭确认影响前端时，它仍然可以通过API。

手动关闭问题

您可以通过检查关闭问题选项通过确认屏幕[手动关闭](#)问题。 如果在[触发器配置](#)中选中了允许手动关闭选项，则可以以这种方式解决问题。

显示

通过单击监视 → 问题中的事件时间可以在事件详细信息中完整显示确认信息。

基于确认信息，可以配置问题计数在仪表板或地图中的显示方式。 为此，您必须在[映射配置](#)和[仪表板过滤器](#)。可以将所有问题计数，未确认的问题数量显示为仅与总数或未确认的问题数量分开。

确认状态显示在监视 → 触发器中。 在那里，确认状态也与触发器过滤选项一起使用。 您可以通过未确认的触发器或触发器过滤未知的最后一个事件。

14. 配置导出/导入

概述

Zabbix导出/导入功能使得可以在一个Zabbix系统与另一个Zabbix系统之间交换各种配置实体。

此功能的典型用例： 共享模板或网络地图 - Zabbix用户可以共享其配置参数 在`share.zabbix.com` 上共享网络场景 - 通过Web方案导出模板并上传到`share.zabbix.com`。 然后其他人可以下载模板并将XML导入Zabbix。

* 与第三方工具集成 - 通用的XML格式使得第三方工具和应用程序的集成和数据导入/导出成为可能。

什么可以导出/导入

可导出/导入的对象有：

- [主机组](#) (仅限Zabbix API)
- [模板](#)
- [主机](#)
- [拓扑](#)
- 图片
- [聚合图形](#)
- 值映射

导出格式

数据可以使用Zabbix 网页前端或[Zabbix API](#)导出。 支持导出的格式有：

- XML - 在前端
- XML or JSON - 在Zabbix API

关于导出的详细信息

- 所有支持的元素都导出到一个文件中。
- 不导出从链接模板继承的主机和模板实体（项目，触发器，图形，发现规则）。 在导出时，对主机级别上的这些实体所做的任何更改（例如更改的项目间隔，修改的正则表达式或添加的原型到低级发现规则）都将丢失；导入时，所有来自链接模板的实体将按照原始链接的模板重新创建。
- 由低级别发现创建的实体和依赖于它们的任何实体不会导出。 例如，为LLD规则生成的项目创建的触发器将不被导出。

关于导入的详细信息

- 第一次遇到错误时停止导入。

- 在图像导入期间更新现有图像时，将忽略“imagetype”字段，即无法通过导入更改图像类型。
- 使用“删除缺失”选项导入主机/模板时，导入的XML文件中不存在的主机/模板宏也将被删除。
- items, triggers, graphs, host/template applications, discoveryRules, itemPrototypes, triggerPrototypes, graphPrototypes的空标签无意义，即与缺少的标签相同。其他标签，例如item applications是有意义的，即空标签表示没有项目的应用，缺少的标签装置不更新应用。
- 导入支持XML和JSON，导入文件必须具有正确的文件扩展名：XML使用.xml 和JSON使用.json。
- 有关支持的XML版本，请参阅 [兼容性信息](#)。

XML 基本格式

```
1. <?xml version="1.0" encoding="UTF-8"?>
2. <zabbix_export>
3.   <version>3.4</version>
4.   <date>2016-10-04T06:20:11Z</date>
5. </zabbix_export>
```

```
1. <?xml version="1.0" encoding="UTF-8"?>
```

XML文档的默认开始。

```
1. <zabbix_export>
```

Zabbix XML导出的根元素。

```
1. <version>3.4</version>
```

导出版本。

```
1. <date>2016-10-04T06:20:11Z</date>
```

以ISO 8601长格式创建导出时的日期。

其他标签依赖于导出的对象。

1 主机组

在前端主机组只能通过主机或模板[导出](#)。当导出主机或模板时，它所属的所有组都将自动导出。

API允许独立于主机或模板导出主机组。

```
1. <groups>
2.   <group>
3.     <name>Zabbix servers</name>
4.   </group>
5. </groups>
```

groups/group

参数	类型	说明	详细
name	<i>string</i>	组名.	

2 模板

概述

模板的[导出](#)包含许多相关对象和对象关系。

模板导出包含：

- 链接的主机组
- 模板数据
- 链接到其他模板
- 连接到主机组
- 直接链接的应用集
- 直接链接的监控项
- 直接链接触发器
- 直接链接图形
- 直接链接聚合图形
- 直接链接发现规则与所有原型
- 直接链接Web场景
- 值映射

导出

导出模板，按以下步骤：

- 转到：配置 → 模板
- 选中需要导出模板的复选框
- 在列表正文点击导出

The screenshot shows the 'Templates' section of the Zabbix configuration interface. A single template, 'Template App MySQL', is selected and highlighted with a yellow background. The 'Name' dropdown menu is open, showing 'Name ▾'. To the right, the text 'Applications' is displayed above the selected template. Below the list are three buttons: 'Export' (highlighted with a blue border), 'Delete', and 'Delete and clear'. A status message '1 selected' is shown to the left of the buttons.

所选模板将导出的本地XML文件默认名称为`zabbix_export_templates.xml`。

导入

导入模板，按以下步骤：

- 转到：配置 → 模板
- 点击右边的*Import*
- 选择需要导入的文件
- 在导入规则中选中需要的选项
- 点击导入

Rules	Update existing	Create new	Delete missing
Groups	<input checked="" type="checkbox"/>		
Hosts	<input type="checkbox"/>	<input type="checkbox"/>	
Templates	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	
Template screens	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>
Template linkage		<input checked="" type="checkbox"/>	
Applications		<input checked="" type="checkbox"/>	<input type="checkbox"/>
Items	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>
Discovery rules	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>
Triggers	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>
Graphs	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>
Web scenarios	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>
Screens	<input type="checkbox"/>	<input type="checkbox"/>	
Maps	<input type="checkbox"/>	<input type="checkbox"/>	
Images	<input type="checkbox"/>	<input type="checkbox"/>	
Value mappings	<input type="checkbox"/>	<input checked="" type="checkbox"/>	

Import **Cancel**

导入后在前端显示成功或失败的信息

导入规则：

规则	说明
更新现有的	将使用从导入文件获取的数据更新现有元素。 否则不会更新。
创建新的	导入将使用导入文件中的数据添加新元素。 否则不会添加它们。
删除失败	导入将删除导入文件中不存在的现有元素。 否则不会删除它们。

导出格式

```

1. <?xml version="1.0" encoding="UTF-8"?>
2. <zabbix_export>
3.   <version>3.4</version>
4.   <date>2016-12-28T09:18:27Z</date>
5.   <groups>
6.     <group>
7.       <name>Templates</name>
8.     </group>
9.   </groups>
10.  <templates>
11.    <template>

```

```

12.      <template>Template App MySQL</template>
13.      <name>Template App MySQL</name>
14.      <description/>
15.      <groups>
16.          <group>
17.              <name>Templates</name>
18.          </group>
19.      </groups>
20.      <applications>
21.          <application>
22.              <name>MySQL</name>
23.          </application>
24.      </applications>
25.      <items>
26.          <item>
27.              <name>MySQL status</name>
28.              <type>0</type>
29.              <snmp_community/>
30.              <snmp_oid/>
31.              <key>mysql.ping</key>
32.              <delay>60</delay>
33.              <history>7</history>
34.              <trends>365</trends>
35.              <status>0</status>
36.              <value_type>3</value_type>
37.              <allowed_hosts/>
38.              <units/>
39.              <snmpv3_contextname/>
40.              <snmpv3_securityname/>
41.              <snmpv3_securitylevel>0</snmpv3_securitylevel>
42.              <snmpv3_authprotocol>0</snmpv3_authprotocol>
43.              <snmpv3_authpassphrase/>
44.              <snmpv3_privprotocol>0</snmpv3_privprotocol>
45.              <snmpv3_privpassphrase/>
46.              <formula>1</formula>
47.              <delay_flex/>
48.              <params/>
49.              <ipmi_sensor/>
50.              <authtype>0</authtype>
51.              <username/>
52.              <password/>
53.              <publickey/>
54.              <privatekey/>
55.              <port/>
56.              <description>It requires user parameter mysql.ping, which is defined in
      userparameter_mysql.conf.&#13;
57.      &#13;;
58.      0 - MySQL server is down&#13;;
59.      1 - MySQL server is up</description>
60.              <inventory_link>0</inventory_link>
61.              <applications>
62.                  <application>
63.                      <name>MySQL</name>
64.                  </application>

```

```
65.             </applications>
66.             <valuemap>
67.                 <name>Service state</name>
68.             </valuemap>
69.             <logtimefmt/>
70.             <preprocessing/>
71.         </item>
72.         <item>
73.             <name>MySQL insert operations per second</name>
74.             <type>0</type>
75.             <snmp_community/>
76.             <snmp_oid/>
77.             <key>mysql.status[Com_insert]</key>
78.             <delay>60</delay>
79.             <history>7</history>
80.             <trends>365</trends>
81.             <status>0</status>
82.             <value_type>0</value_type>
83.             <allowed_hosts/>
84.             <units>qps</units>
85.             <snmpv3_contextname/>
86.             <snmpv3_securityname/>
87.             <snmpv3_securitylevel>0</snmpv3_securitylevel>
88.             <snmpv3_authprotocol>0</snmpv3_authprotocol>
89.             <snmpv3_authpassphrase/>
90.             <snmpv3_privprotocol>0</snmpv3_privprotocol>
91.             <snmpv3_privpassphrase/>
92.             <formula>1</formula>
93.             <delay_flex/>
94.             <params/>
95.             <ipmi_sensor/>
96.             <authtype>0</authtype>
97.             <username/>
98.             <password/>
99.             <publickey/>
100.            <privatekey/>
101.            <port/>
102.            <description>It requires user parameter mysql.status[*], which is defined in
    userparameter_mysql.conf.</description>
103.            <inventory_link>0</inventory_link>
104.            <applications>
105.                <application>
106.                    <name>MySQL</name>
107.                </application>
108.            </applications>
109.            <valuemap/>
110.            <logtimefmt/>
111.            <preprocessing>
112.                <step>
113.                    <type>10</type>
114.                <params/>
115.                </step>
116.            </preprocessing>
117.        </item>
```

```
118.      <item>
119.          <name>MySQL queries per second</name>
120.          <type>0</type>
121.          <snmp_community/>
122.          <snmp_oid/>
123.          <key>mysql.status[Questions]</key>
124.          <delay>60</delay>
125.          <history>7</history>
126.          <trends>365</trends>
127.          <status>0</status>
128.          <value_type>0</value_type>
129.          <allowed_hosts/>
130.          <units>qps</units>
131.          <snmpv3_contextname/>
132.          <snmpv3_securityname/>
133.          <snmpv3_securitylevel>0</snmpv3_securitylevel>
134.          <snmpv3_authprotocol>0</snmpv3_authprotocol>
135.          <snmpv3_authpassphrase/>
136.          <snmpv3_privprotocol>0</snmpv3_privprotocol>
137.          <snmpv3_privpassphrase/>
138.          <formula>1</formula>
139.          <delay_flex/>
140.          <params/>
141.          <ipmi_sensor/>
142.          <authtype>0</authtype>
143.          <username/>
144.          <password/>
145.          <publickey/>
146.          <privatekey/>
147.          <port/>
148.          <description>It requires user parameter mysql.status[*], which is defined in
    userparameter_mysql.conf.</description>
149.          <inventory_link>0</inventory_link>
150.          <applications>
151.              <application>
152.                  <name>MySQL</name>
153.                  </application>
154.              </applications>
155.              <valuemap/>
156.              <logtimefmt/>
157.              <preprocessing>
158.                  <step>
159.                      <type>10</type>
160.                      <params/>
161.                  </step>
162.              </preprocessing>
163.          </item>
164.      </items>
165.      <discovery_rules/>
166.      <httptests/>
167.      <macros/>
168.      <templates/>
169.      <screens>
170.          <screen>
```

```
171.         <name>MySQL performance</name>
172.         <hsize>2</hsize>
173.         <vsize>1</vsize>
174.         <screen_items>
175.             <screen_item>
176.                 <resourcetype>0</resourcetype>
177.                 <width>500</width>
178.                 <height>200</height>
179.                 <x>0</x>
180.                 <y>0</y>
181.                 <colspan>1</colspan>
182.                 <rowspan>1</rowspan>
183.                 <elements>0</elements>
184.                 <valign>1</valign>
185.                 <halign>0</halign>
186.                 <style>0</style>
187.                 <url/>
188.                 <dynamic>0</dynamic>
189.                 <sort_triggers>0</sort_triggers>
190.             <resource>
191.                 <name>MySQL operations</name>
192.                 <host>Template App MySQL</host>
193.             </resource>
194.             <max_columns>3</max_columns>
195.             <application/>
196.         </screen_item>
197.         <screen_item>
198.             <resourcetype>0</resourcetype>
199.             <width>500</width>
200.             <height>270</height>
201.             <x>1</x>
202.             <y>0</y>
203.             <colspan>1</colspan>
204.             <rowspan>1</rowspan>
205.             <elements>0</elements>
206.             <valign>1</valign>
207.             <halign>0</halign>
208.             <style>0</style>
209.             <url/>
210.             <dynamic>0</dynamic>
211.             <sort_triggers>0</sort_triggers>
212.             <resource>
213.                 <name>MySQL bandwidth</name>
214.                 <host>Template App MySQL</host>
215.             </resource>
216.             <max_columns>3</max_columns>
217.             <application/>
218.         </screen_item>
219.     </screen_items>
220. </screen>
221. </screens>
222. </template>
223. </templates>
```

```

224. <triggers>
225.   <trigger>
226.     <expression>{Template App MySQL:mysql.ping.last(0)}=0</expression>
227.     <recovery_mode>0</recovery_mode>
228.     <recovery_expression/>
229.     <name>MySQL is down</name>
230.     <correlation_mode>0</correlation_mode>
231.     <correlation_tag/>
232.     <url/>
233.     <status>0</status>
234.     <priority>2</priority>
235.     <description/>
236.     <type>0</type>
237.     <manual_close>0</manual_close>
238.     <dependencies/>
239.     <tags/>
240.   </trigger>
241. </triggers>
242. <graphs>
243.   <graph>
244.     <name>MySQL operations</name>
245.     <width>900</width>
246.     <height>200</height>
247.     <yaxismin>0.0000</yaxismin>
248.     <yaxismax>100.0000</yaxismax>
249.     <show_work_period>1</show_work_period>
250.     <show_triggers>1</show_triggers>
251.     <type>0</type>
252.     <show_legend>1</show_legend>
253.     <show_3d>0</show_3d>
254.     <percent_left>0.0000</percent_left>
255.     <percent_right>0.0000</percent_right>
256.     <ymin_type_1>0</ymin_type_1>
257.     <ymax_type_1>0</ymax_type_1>
258.     <ymin_item_1>0</ymin_item_1>
259.     <ymax_item_1>0</ymax_item_1>
260.     <graph_items>
261.       <graph_item>
262.         <sortorder>0</sortorder>
263.         <drawtype>0</drawtype>
264.         <color>C8C800</color>
265.         <yaxisside>0</yaxisside>
266.         <calc_fnc>2</calc_fnc>
267.         <type>0</type>
268.         <item>
269.           <host>Template App MySQL</host>
270.           <key>mysql.status[Com_begin]</key>
271.         </item>
272.       </graph_item>
273.       <graph_item>
274.         <sortorder>1</sortorder>
275.         <drawtype>0</drawtype>
276.         <color>006400</color>

```

```
277.             <yaxisside>0</yaxisside>
278.             <calc_fnc>2</calc_fnc>
279.             <type>0</type>
280.             <item>
281.                 <host>Template App MySQL</host>
282.                 <key>mysql.status[Com_commit]</key>
283.             </item>
284.         </graph_item>
285.         <graph_item>
286.             <sortorder>2</sortorder>
287.             <drawtype>0</drawtype>
288.             <color>C80000</color>
289.             <yaxisside>0</yaxisside>
290.             <calc_fnc>2</calc_fnc>
291.             <type>0</type>
292.             <item>
293.                 <host>Template App MySQL</host>
294.                 <key>mysql.status[Com_delete]</key>
295.             </item>
296.         </graph_item>
297.         <graph_item>
298.             <sortorder>3</sortorder>
299.             <drawtype>0</drawtype>
300.             <color>0000EE</color>
301.             <yaxisside>0</yaxisside>
302.             <calc_fnc>2</calc_fnc>
303.             <type>0</type>
304.             <item>
305.                 <host>Template App MySQL</host>
306.                 <key>mysql.status[Com_insert]</key>
307.             </item>
308.         </graph_item>
309.         <graph_item>
310.             <sortorder>4</sortorder>
311.             <drawtype>0</drawtype>
312.             <color>640000</color>
313.             <yaxisside>0</yaxisside>
314.             <calc_fnc>2</calc_fnc>
315.             <type>0</type>
316.             <item>
317.                 <host>Template App MySQL</host>
318.                 <key>mysql.status[Com_rollback]</key>
319.             </item>
320.         </graph_item>
321.         <graph_item>
322.             <sortorder>5</sortorder>
323.             <drawtype>0</drawtype>
324.             <color>00C800</color>
325.             <yaxisside>0</yaxisside>
326.             <calc_fnc>2</calc_fnc>
327.             <type>0</type>
328.             <item>
329.                 <host>Template App MySQL</host>
```

```

330.           <key>mysql.status[Com_select]</key>
331.           </item>
332.       </graph_item>
333.       <graph_item>
334.           <sortorder>6</sortorder>
335.           <drawtype>0</drawtype>
336.           <color>C800C8</color>
337.           <yaxisside>0</yaxisside>
338.           <calc_fnc>2</calc_fnc>
339.           <type>0</type>
340.       <item>
341.           <host>Template App MySQL</host>
342.           <key>mysql.status[Com_update]</key>
343.       </item>
344.   </graph_item>
345. </graph_items>
346. </graph>
347. </graphs>
348. <value_maps>
349.     <value_map>
350.         <name>Service state</name>
351.         <mappings>
352.             <mapping>
353.                 <value>0</value>
354.                 <newvalue>Down</newvalue>
355.             </mapping>
356.             <mapping>
357.                 <value>1</value>
358.                 <newvalue>Up</newvalue>
359.             </mapping>
360.         </mappings>
361.     </value_map>
362. </value_maps>
363. </zabbix_export>

```

元素标签

元素标签值在下表中说明。

Templates 标签

元素	元素属性	类型	范围	说明
templates				模板根元素
template				单个模板
	template	string		唯一的模板名称。
	name	string		可见的模板名称。
	description	text		模板说明
groups				主机组根元素
group				单个主机组

	name	string	唯一的组名称
applications			模板应用集的根元素
application			单个模板应用集
	name		应用集名称
macros			模板用户宏的根元素。
macro			单个模板用户宏
	name		模板用户宏名称
	value		模板用户宏值
templates			链接的模板的根元素
template			单个模板
	name	string	模板名称

Template item 标签

元素	元素属性	类型	范围	说明
items				监控项的根元素
item				单个监控项
	name	string		监控项名称
	type	integer	0 - Zabbix 客户端 - SNMPv1 客户端 - Zabbix采集器 3 - 简单检查 4 - SNMPv2 客户端 5 - Zabbix内部 - SNMPv3 客户端 7 - Zabbix客户端(主动式) 8 - Zabbix整合 9 - HTTP 测试(Web监控场步骤) 10 - 外部检查 11 - 数据库监控 12 - IPMI客户端 13 - SSH 客户端 14 - TELNET客户端 15 - 可计算的 16 - JMX agent代理程序 17 - SNMP trap	监控项类型
	snmpcommunity	string		如果“类型”为1, SNMP团体名称
	snmp_oid	string		SNMP 对象ID
	key	string		监控项的键
	delay	integer		监控项收集的常数(以秒为单位)
	history	integer		保存历史的天数。
	trends	integer		保持趋势的天数。
	status	integer	0 - 启用 1 - 禁用	监控项状态
			0 - 浮点数 1 - 字符	

	<code>value_type</code>	<code>integer</code>	- 日志3 - 数字(无正负)4 - 文本	接收值的类型
	<code>allowed_hosts</code>	<code>string</code>		如果'type'为2,发送数据的主机的 址(逗号分隔)。
	<code>units</code>	<code>string</code>		返回值单位(bps B)。
	<code>snmpv3_contextname</code>	<code>string</code>		SNMPV3上下文名
	<code>snmpv3_securityname</code>	<code>string</code>		SNMPV3安全名称
	<code>snmpv3_securitylevel</code>	<code>integer</code>	0 - noAuthNoPriv1 - authNoPriv2 - authPriv	SNMPV3 安装级别
	<code>snmpv3_authprotocol</code>	<code>integer</code>	0 - MD51 - SHA	SNMPV3 认证协议
	<code>snmpv3_authpassphrase</code>	<code>string</code>		SNMPV3 认证密钥
	<code>snmpv3_privprotocol</code>	<code>integer</code>	0 - DES1 - AES	SNMPV3 隐私协议
	<code>snmpv3_privpassphrase</code>	<code>string</code>		SNMPV3 隐私密钥
	<code>formula</code>	<code>string</code>		如 果“multiplier” 则计算接收值的 乘积。
	<code>delay_flex</code>	<code>string</code>		监控项数据采集更 频率。
	<code>params</code>	<code>text</code>		如果“type”为 13,14, 则“执行 本”的名称如 果'type'为11, 则“SQL query” 如果'type'为15 则“Formula”字 符串。
	<code>ipmi_sensor</code>	<code>string</code>		如果'type'为12 IPMI 传感器ID
	<code>authtype</code>	<code>integer</code>	0 - password1 - key	'type'为13, 验 证类型
	<code>username</code>	<code>string</code>		'type' 为 11,13,14, 用户 名
	<code>password</code>	<code>string</code>		'type' 为 11,13,14, 密码
	<code>publickey</code>	<code>string</code>		'type' 为 13, 文件的名称
	<code>privatekey</code>	<code>string</code>		'type' 为 13, 文件的名称
	<code>port</code>	<code>string</code>		监控项的自定义端 口
	<code>description</code>	<code>text</code>		监控项说明
	<code>inventory_link</code>	<code>integer</code>	0 - 没有链接数字_ - “host_inventory”表 中的字段数	使用监控项值填 充字段。
	<code>logtimefmt</code>	<code>string</code>		日志条目中的时 间格式。仅用于日志 项。

	interface_ref	string		参考主机接口。
value_map				Value map.
	name	string		值映射监控项的名
applications				应用集的根元素
application				单个应用集
	name			应用集名称
preprocessing				监控项值预处理
step				单个监控项值预处 骤
	type	integer	1 - 自定义乘数 2 - 右边修剪 3 - 左边修剪 4 - 两边修剪 5 - 正则表达式匹配 6 - 布尔值转十进制 7 - 八进制转十进制 8 - 十六进制转十进制 9 - 计算为(接收值 - 先前值) 10 - 计算为(收到的值 - 上一个值) / (上次检查的时间)	监控项值预处理步 类型。
	params	string		监控项值预处理步 参数。

模板低级发现规则标签

元素	元素属性	类型	范围	说明
discoveryrules				低级别发现规则的根元素
discovery_rule				单个低级别发现规则
	对于大多数元素标签值，请参阅常规监控项的元素标签值。下面仅描述特定于低级发现规则的标签。			
	lifetime	string	0 - 3650	由LLD规则发现丢失资源的保留天数。
filter				单个过滤器
	evaltype	integer	0 - 与/或逻辑 1 - 与逻辑 2 - 或逻辑 3 - 自定义公式	用于检查低级发现规则过滤条件的逻辑。
	formula	string		过滤条件的自定义计算公式。

	conditions			过滤条件的根元素
condition				单个过滤条件
	macro	string		低级发现的宏名称
	value	string		过滤器值：正则表达式或全局正则表达式
	operator	integer		.
	formulaid	character		过滤器条件ID。用于定制计算公式
item_prototypes				item_prototypes的根元素
item_prototype				单个item_prototype。
	对于大多数元素标签值，请参阅常规监控项的元素标签值。下面仅描述特定于itemprototypes的标签。			
application_prototypes				应用集原型的根元素。
application_prototype				单个应用集原型
	name			应用集原型名称

模板触发器标签

元素	元素属性	类型	范围	说明
triggers				触发器的根元素
trigger				单个触发器
	expression	string		触发器表达式
	recovery_mode	integer	0 - 表达式1 - 恢复表达式2 - 无	生成OK事件的基础。
	recovery_expression	string		触发器恢复表达式。
	name	string		触发器名字
	correlation_mode	integer	0 - 无事件关联1 - 通过标签的事件关联	关联模式
	correlation_tag	string		用于事件相关的标签名称
	url	string		触发器

	url	string		URL。
	status	integer	0 - 启用1 - 禁用	触发器状态
	priority	integer	0 - 不分类1 - 信息2 - 警告 3 - 一般严重4 - 严重5 - 灾难	触发严重性。
	description	text		触发器说明
	type	integer	0 - 单问题事件1 - 多问题事件	事件生成类型。
	manual_close	integer	0 - 不允许1 - 允许	手动关闭问题事件。
dependencies				依赖关系的根元素。
dependency				单个依赖关系
	name	string		依赖触发器名称。
	expression	string		依赖触发器表达式
	recovery_expression	string		依赖触发恢复表达式。
tags				事件标签的根元素
tag				单个事件标签
	tag	string		标签名称
	value	string		标签值

模板图形标签

元素	元素属性	类型	范围	说明
graphs				图形的根元素
graph				单个图形
	name	string		图形名称
	width	integer		图形宽度，以像素为单位。用于预览和饼图/分裂式饼图。
	height	integer		图形高度，以像素为单位。用于预览和饼图/分裂式饼图。
	yaxismin	double		如果“ymin_type_1”为1，Y轴的最小值。

yaxismax	<code>double</code>		如果“ymax_type_1”为1, Y轴的最大值。
show_work_period	<code>integer</code>	0 - 否1 - 是	如果“type”为0,1, 则突出显示非工作时间。
show_triggers	<code>integer</code>	0 - 否1 - 是	如果'type'为0,1, 则显示简单的触发值作为一行。
type	<code>integer</code>	0 - 正常1 - 层积的2 - 饼图3 - 分裂的4 - 3D饼图5 - 3D分裂式饼图	图形类型
show_legend	<code>integer</code>	0 - 否1 - 是	显示图例
show_3d	<code>integer</code>	0 - 2D1 - 3D	如果'type'为2,3, 启用3D风格
percent_left	<code>double</code>		如果“type”为0, 则显示左轴的百分位数线。
percent_right	<code>double</code>		如果“type”为0, 则显示右轴的百分位数线。
ymin_type_1	<code>integer</code>	0 - 可计算的1 - 固定的2 - 选择监控项的最新值	如果'type'为0,1, Y轴的最小值。
ymax_type_1	<code>integer</code>	0 - 可计算的1 - 固定的2 - 选择监控项的最新值	如果'type'为0,1, Y轴的最大值。
ymin_item_1	<code>string</code>	空或监控项详细信息	如果'ymin_type_1'为2, 监控项详细信息
ymax_item_1	<code>string</code>	空或监控项详细信息	如果'ymax_type_1'为2, 监控项详细信息

graph_items				图形监控项的根元素
graph_item				单个图形监控项
	sortorder	<code>integer</code>		绘制顺序。 较小的值首先绘制。 可用于绘制线条或区域后面(或前面)的另一个。
	drawtype	<code>integer</code>	0 - 线1 - 填满的区域2 - 粗线3 - 点4 - 虚线	如果图形'type'为0, 绘制风格
	color	<code>string</code>		元素颜色(6位, 十六进制)
	yaxisside	<code>integer</code>	0 - 左轴1 - 右轴	如果图形'type'是0,1, 元素所属的Y轴位置(左或右)
	calc_fnc	<code>integer</code>	1 - 最小2 - 平均4 - 最大7 - 所有(如果图形'type'为0, 最小, 平均和最大)9 - 最后(如果图形'type'不为0,1)	如果一个监控项存在多个值, 则绘制数据。
	type	<code>integer</code>	1 - 监控项的值成比例地表示在饼图2 - 监控项的值代表整个饼图(图形概括)	饼图/分解图的绘制类型。

item				单个监控项
	host	string		监控项主机
	key	string		监控项键

模板Web场景标签

元素	元素属性	类型	范围	说明
httpTests				Web场景的根元素
httpTest				单个Web场景
	name	string		Web场景名称
	delay	integer		执行Web场景的频率，以秒为单位。
	attempts	integer	1-10	执行Web方案步骤的尝试次数。
	agent	string		客户代理。 Zabbix将假装成为所选的浏览器。 同的浏览器返回不同的内容时，这很有用。
	http_proxy	string		指定要使用的HTTP代理。 使用格式： <code>http://[username[:password]@]proxy.mycompany</code>
	variables	text		在场景步骤中可以使用的场景级变量（宏）列表。
	headers	text		执行请求时将发送的HTTP头。
	status	integer	0 - 启用 1 - 禁用	Web场景状态
	authentication	integer	0 - 无 1 - 基本 2 - NTLM	验证方式
	http_user	string		验证用户名
	http_password	string		用户名指定的验证密码
	verify_peer	integer	0 - 否 1 - 是	验证Web服务器的SSL证书。
	verify_host	integer	0 - 否 1 - 是	验证Web服务器证书的公用名称字段或主题备用名匹配。
	ssl_cert_file	string		用于客户端身份验证的SSL证书文件的名称。
	ssl_key_file	string		用于客户端身份验证的SSL私钥文件的名称。
	ssl_key_password	string		SSL私钥文件密码。
steps				Web场景步骤的根元素
step				单个Web场景步骤
	name	string		Web场景步骤名称
	url	string		监控的URL
	posts	text		'Post'变量列表。

	variables	text		应在此步骤之后应用级变量（宏）的列表。如果有'regex: '前缀，则根据'regex: '前缀之后式模式从该步骤返回的数据中提取其值
	headers	text		执行请求时将发送的HTTP头。
	follow_redirects	integer	0 - 否 1 - 是	遵循HTTP重定向
	retrieve_mode	integer	0 - 内容 1 - 只有头	HTTP响应检索模式。
	timeout	integer		步骤执行的超时时间，以秒为单位
	required	string		必填字符串。如果空，则忽略。
	status_codes	string		以逗号分隔的已接受的状态代码列表。如果空，如：200-201, 210-299



3 主机

概述

主机[导出](#)带很多相关对象和对象关系。

主机导出包含：

- 链接的主机组
- 主机数据
- 模板联动
- 主机组联动
- 主机接口
- 直接链接的应用集
- 直接链接的监控项
- 直接链接触发器
- 直接链接图形
- 直接链接发现规则与所有原型
- 直接链接Web场景
- 主机宏
- 主机库存数据
- 值映射

When a host is imported and updated, it can only be linked to additional templates and never be unlinked from any.当主机导入和更新，它只能被链接到其他模板，从来没有从任何断开链接。

导出

要导出主机，请执行以下操作

- 转到：配置 → 主机
- 选择要导出的主机的复选框
- 点击列表下方的导出

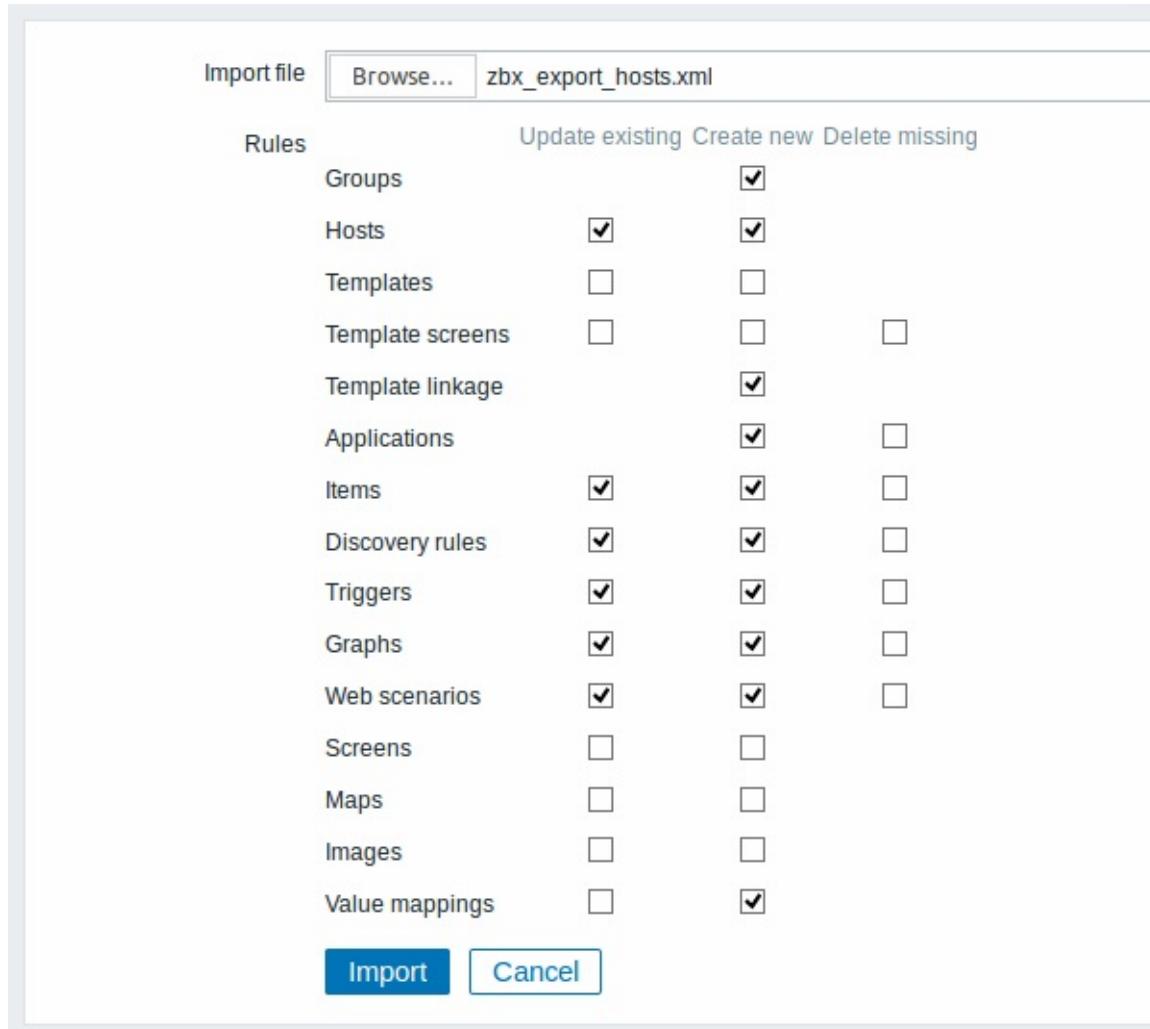
The screenshot shows the 'Hosts' section of the Zabbix interface. At the top, there's a header with tabs: Applications, Items, Triggers, Graphs, Discovery, and View. Below the header, a table lists two hosts: 'Zabbix server' and 'Zabbix host'. Both hosts have their checkboxes checked. The 'Zabbix server' row shows 12 Applications, 79 Items, 46 Triggers, 12 Graphs, and 3 Discoveries. The 'Zabbix host' row shows 10 Applications, 43 Items, 21 Triggers, 10 Graphs, and 2 Discoveries. At the bottom of the table, there are five buttons: 'Enable', 'Disable', 'Export' (which has a cursor icon over it), 'Mass update', and 'Delete'.

Selected hosts are exported to a local XML file with default name `zabbix_export_hosts.xml`. 被选择的主机导出到本地XML文件，默认名字为`zabbix_export_hosts.xml`。

导入

要导入主机，请执行以下操作

- 转到：配置 → 主机
- 点击右边的导入
- 选择要导入的文件
- 在导入规则里选择需要的选项
- 点击导入



导入的成功或失败消息将显示在前端。

导入规则：

规则	说明
更新现有的	将使用从导入文件获取的数据更新现有元素。 否则不会更新。
创建新的	导入将使用导入文件中的数据添加新元素。 否则不会添加它们。
删除失败	导入将删除导入文件中不存在的现有元素。 否则不会删除它们。

导出格式

```

1. <?xml version="1.0" encoding="UTF-8"?>
2. <zabbix_export>
3.   <version>3.4</version>
4.   <date>2016-12-20T08:26:43Z</date>
5.   <hosts>
6.     <host>
7.       <host>Zabbix server</host>
8.       <name>Zabbix server</name>
9.       <description>Zabbix monitoring server.</description>
10.      <proxy/>
11.      <status>0</status>

```

```
12.      <ipmi_authtype>-1</ipmi_authtype>
13.      <ipmi_privilege>2</ipmi_privilege>
14.      <ipmi_username/>
15.      <ipmi_password/>
16.      <templates/>
17.      <groups>
18.          <group>
19.              <name>Zabbix servers</name>
20.          </group>
21.      </groups>
22.      <interfaces>
23.          <interface>
24.              <default>1</default>
25.              <type>1</type>
26.              <useip>1</useip>
27.              <ip>127.0.0.1</ip>
28.              <dns/>
29.              <port>20001</port>
30.              <interface_ref>if1</interface_ref>
31.          </interface>
32.      </interfaces>
33.      <applications>
34.          <application>
35.              <name>Memory</name>
36.          </application>
37.          <application>
38.              <name>Zabbix agent</name>
39.          </application>
40.      </applications>
41.      <items>
42.          <item>
43.              <name>Agent ping</name>
44.              <type>0</type>
45.              <snmp_community/>
46.              <snmp_oid/>
47.              <key>agent.ping</key>
48.              <delay>60</delay>
49.              <history>7</history>
50.              <trends>365</trends>
51.              <status>0</status>
52.              <value_type>3</value_type>
53.              <allowed_hosts/>
54.              <units/>
55.              <snmpv3_securityname/>
56.              <snmpv3_securitylevel>0</snmpv3_securitylevel>
57.              <snmpv3_authpassphrase/>
58.              <snmpv3_privpassphrase/>
59.              <formula>1</formula>
60.              <delay_flex/>
61.              <params/>
62.              <ipmi_sensor/>
63.              <authtype>0</authtype>
64.              <username/>
```

```
65.          <password/>
66.          <publickey/>
67.          <privatekey/>
68.          <port/>
69.          <description>The agent always returns 1 for this item. It could be used in combination
  with nodata() for availability check.</description>
70.          <inventory_link>0</inventory_link>
71.          <applications>
72.              <application>
73.                  <name>Zabbix agent</name>
74.              </application>
75.          </applications>
76.          <valuemap>
77.              <name>Zabbix agent ping status</name>
78.          </valuemap>
79.          <logtimefmt/>
80.          <preprocessing/>
81.          <interface_ref>if1</interface_ref>
82.      </item>
83.      <item>
84.          <name>Available memory</name>
85.          <type>0</type>
86.          <snmp_community/>
87.          <snmp_oid/>
88.          <key>vm.memory.size[available]</key>
89.          <delay>60</delay>
90.          <history>7</history>
91.          <trends>365</trends>
92.          <status>0</status>
93.          <value_type>3</value_type>
94.          <allowed_hosts/>
95.          <units>B</units>
96.          <snmpv3_securityname/>
97.          <snmpv3_securitylevel>0</snmpv3_securitylevel>
98.          <snmpv3_authpassphrase/>
99.          <snmpv3_privpassphrase/>
100.         <formula>1</formula>
101.         <delay_flex/>
102.         <params/>
103.         <ipmi_sensor/>
104.         <authtype>0</authtype>
105.         <username/>
106.         <password/>
107.         <publickey/>
108.         <privatekey/>
109.         <port/>
110.         <description>Available memory is defined as free+cached+buffers memory.</description>
111.         <inventory_link>0</inventory_link>
112.         <applications>
113.             <application>
114.                 <name>Memory</name>
115.             </application>
116.         </applications>
117.         <valuemap/>
```

```
118.          <logtimefmt/>
119.          <preprocessing/>
120.          <interface_ref>if1</interface_ref>
121.        </item>
122.      </items>
123.    <discovery_rules>
124.      <discovery_rule>
125.        <name>Mounted filesystem discovery</name>
126.        <type>0</type>
127.        <snmp_community/>
128.        <snmp_oid/>
129.        <key>vfs.fs.discovery</key>
130.        <delay>3600</delay>
131.        <status>0</status>
132.        <allowed_hosts/>
133.        <snmpv3_securityname/>
134.        <snmpv3_securitylevel>0</snmpv3_securitylevel>
135.        <snmpv3_authpassphrase/>
136.        <snmpv3_privpassphrase/>
137.        <delay_flex/>
138.        <params/>
139.        <ipmi_sensor/>
140.        <authtype>0</authtype>
141.        <username/>
142.        <password/>
143.        <publickey/>
144.        <privatekey/>
145.        <port/>
146.        <filter>{#FSTYPE}:@File systems for discovery</filter>
147.        <lifetime>30</lifetime>
148.        <description>Discovery of file systems of different types as defined in global regular
expression "File systems for discovery".</description>
149.      <item_prototypes>
150.        <item_prototype>
151.          <name>Free disk space on $1</name>
152.          <type>0</type>
153.          <snmp_community/>
154.          <snmp_oid/>
155.          <key>vfs.fs.size[{#FSNAME},free]</key>
156.          <delay>60</delay>
157.          <history>7</history>
158.          <trends>365</trends>
159.          <status>0</status>
160.          <value_type>3</value_type>
161.          <allowed_hosts/>
162.          <units>B</units>
163.          <snmpv3_securityname/>
164.          <snmpv3_securitylevel>0</snmpv3_securitylevel>
165.          <snmpv3_authpassphrase/>
166.          <snmpv3_privpassphrase/>
167.          <formula>1</formula>
168.          <delay_flex/>
169.          <params/>
170.          <ipmi_sensor/>
```

```
171.             <authtype>0</authtype>
172.             <username/>
173.             <password/>
174.             <publickey/>
175.             <privatekey/>
176.             <port/>
177.             <description/>
178.             <inventory_link>0</inventory_link>
179.             <applications>
180.                 <application>
181.                     <name>Filesystems</name>
182.                 </application>
183.             </applications>
184.             <valuemap/>
185.             <logtimefmt/>
186.             <preprocessing/>
187.             <application_prototypes>
188.                 <application_prototype>
189.                     <name>{#FSNAME}</name>
190.                 </application_prototype>
191.             </application_prototypes>
192.             <interface_ref>if1</interface_ref>
193.         </item_prototype>
194.     </item_prototypes>
195.     <trigger_prototypes>
196.         <trigger_prototype>
197.             <expression>{Zabbix server
2:vfs.fs.size[{#FSNAME},pfree].last()<20</expression>
198.                 <name>Free disk space is less than 20% on volume {#FSNAME}</name>
199.                 <url/>
200.                 <status>0</status>
201.                 <priority>2</priority>
202.                 <description/>
203.                 <type>0</type>
204.             </trigger_prototype>
205.         </trigger_prototypes>
206.         <graph_prototypes>
207.             <graph_prototype>
208.                 <name>Disk space usage {#FSNAME}</name>
209.                 <width>600</width>
210.                 <height>340</height>
211.                 <yaxismin>0.0000</yaxismin>
212.                 <yaxismax>0.0000</yaxismax>
213.                 <show_work_period>0</show_work_period>
214.                 <show_triggers>0</show_triggers>
215.                 <type>2</type>
216.                 <show_legend>1</show_legend>
217.                 <show_3d>1</show_3d>
218.                 <percent_left>0.0000</percent_left>
219.                 <percent_right>0.0000</percent_right>
220.                 <ymin_type_1>0</ymin_type_1>
221.                 <ymax_type_1>0</ymax_type_1>
222.                 <ymin_item_1>0</ymin_item_1>
223.                 <ymax_item_1>0</ymax_item_1>
```

```
224.             <graph_items>
225.                 <graph_item>
226.                     <sortorder>0</sortorder>
227.                     <drawtype>0</drawtype>
228.                     <color>C80000</color>
229.                     <yaxisside>0</yaxisside>
230.                     <calc_fnc>2</calc_fnc>
231.                     <type>2</type>
232.                     <item>
233.                         <host>Zabbix server 2</host>
234.                         <key>vfs.fs.size[{#FSNAME},total]</key>
235.                     </item>
236.                 </graph_item>
237.             <graph_item>
238.                 <sortorder>1</sortorder>
239.                 <drawtype>0</drawtype>
240.                 <color>00C800</color>
241.                 <yaxisside>0</yaxisside>
242.                 <calc_fnc>2</calc_fnc>
243.                 <type>0</type>
244.                 <item>
245.                     <host>Zabbix server 2</host>
246.                     <key>vfs.fs.size[{#FSNAME},free]</key>
247.                 </item>
248.             </graph_item>
249.         </graph_items>
250.     </graph_prototype>
251. </graph_prototypes>
252.     <interface_ref>if1</interface_ref>
253. </discovery_rule>
254. </discovery_rules>
255. <httptests>
256.     <httptest>
257.         <name>Zabbix</name>
258.         <application/>
259.         <delay>60</delay>
260.         <attempts>1</attempts>
261.         <agent>Mozilla/5.0 (Windows NT 6.1; WOW64; Trident/7.0; rv:11.0)</agent>
262.         <http_proxy/>
263.         <variables/>
264.         <headers/>
265.         <status>0</status>
266.         <authentication>0</authentication>
267.         <http_user/>
268.         <http_password/>
269.         <verify_peer>0</verify_peer>
270.         <verify_host>0</verify_host>
271.         <ssl_cert_file/>
272.         <ssl_key_file/>
273.         <ssl_key_password/>
274.         <steps>
275.             <step>
276.                 <name>Main page</name>
```

```

277.           <url>https://zabbix.com</url>
278.           <posts/>
279.           <variables/>
280.           <headers/>
281.           <follow_redirects>1</follow_redirects>
282.           <retrieve_mode>0</retrieve_mode>
283.           <timeout>60</timeout>
284.           <required/>
285.           <status_codes>200</status_codes>
286.       </step>
287.   </steps>
288. </httptest>
289. </httptests>
290. <macros>
291.   <macro>
292.     <macro>{$M1}</macro>
293.     <value>m1</value>
294.   </macro>
295.   <macro>
296.     <macro>{$M2}</macro>
297.     <value>m2</value>
298.   </macro>
299. </macros>
300. <inventory/>
301. </host>
302. </hosts>
303. <value_maps>
304.   <value_map>
305.     <name>Zabbix agent ping status</name>
306.     <Mappings>
307.       <mapping>
308.         <value>1</value>
309.         <newvalue>Up</newvalue>
310.       </mapping>
311.     </Mappings>
312.   </value_map>
313. </value_maps>
314. </zabbix_export>

```

元素标签

元素标签值在下表中说明。

主机标签

元素	元素属性	类型	范围	说明
groups				组的根元素
group				单个组
	name	string		唯一的组名称。

hosts				主机的根元素
host				单个主机
	host	string		唯一的主机名称
	name	string		可见的名称
	description	text		主机说明
	status	integer	0 - 监视的1 - 未监视	主机状态
	ipmi_authtype	integer	-1 - 默认0 - 无1 - MD22 - MD54 - straight5 - OEM6 - RMCP+	IPMI会话认证类型。
	ipmi_privilege	integer	1 - 回调2 - 用户3 - 操作者4 - 管理者5 - OEM	IPMI 优先权层级
	ipmi_username	string		IPMI用户名称
	ipmi_password	string		IPMI密码
	tls_connect	integer	1 - 非加密2 - 带PSK的TLS4 - 带证书的TLS	传出连接类型。
	tls_accept	integer	1 - 非加密2 - 带PSK的TLS3 - 非加密和带PSK的TLS4 - 带证书的TLS5 - 非加密和带证书的TLS6 - 带PSK的TLS 或 证书7 - 非加密 和 带PSK的TLS 或 证书	传入连接类型。
	tls_issuer	string		允许的代理(Agent) / 代理(Proxy)证书颁发者。
	tls_subject	string		允许的代理(Agent) / 代理(Proxy)证书主题。
	tls_psk_identity	string		PSK身份字符串。
	tls_psk	string		PSK值字符串。
proxy				代理服务器(Proxy)
	name	string		监视主机的Proxy代理名称(如果有)。
templates				链接模板的根元素
template		string		单个模板

		<code>string</code>		
interfaces				生机接口的根元素
interface				单个接口
	default	<code>integer</code>	0 - 将要1 - 主要（默认）	接口状态。一个主机只能有一个主接口。
	type	<code>integer</code>	0 - 未知1 - Zabbix客户端 2 - SNMP 3 - IPMI 4 - JMX	接口类型
	useip	<code>integer</code>	0 - 使用DNS名称1 - 使用IP地址	用于连接主机的接口。
	ip	<code>string</code>		IP地址可以是IPv4或IPv6。
	dns	<code>string</code>		DNS名称。
	port	<code>string</code>		端口号。
	bulk	<code>integer</code>	0 - 禁用1 - 启用	使用批量请求SNMP。
	interface_ref	<code>string</code>		要在监控项中使用的接口参考名称。
applications				应用集的根元素
application				单个应用集
	name			应用集应名称
macros				宏的根元素
macro				单个宏
	name			用户宏名称
	value			用户宏值

主机监控项标签

元素	元素属性	类型	范围	说明
items				监控项的根元素
item				单个监控项
	name	<code>string</code>		监控项名称
			0 - Zabbix客户端 - SNMPv1 客户端 - Zabbix 采集器 3 - 简单检查 4 - SNMPv2 客户端 5 - Zabbix内部 6	

	<code>type</code>	<code>integer</code>	Zabbix客户端(主动式)8 - Zabbix整合9 - HTTP 测试 (web 监控场景步骤)10 - 外部检查11 - 数据库监控12 - IPMI客户端13 - SSH 客户端14 - TELNET客户端15 - 可计算的16 - JMX agent代理程序17 - SNMP trap	监控项类型
	<code>snmpcommunity</code>	<code>string</code>		SNMP community name if 'type' 1, 4.
	<code>snmp_oid</code>	<code>string</code>		SNMP 对象 ID.
	<code>key</code>	<code>string</code>		监控项键
	<code>delay</code>	<code>integer</code>		监控项收集的常数(以秒为单位)。
	<code>history</code>	<code>integer</code>		保存历史的天数。
	<code>trends</code>	<code>integer</code>		保存趋势的天数。
	<code>status</code>	<code>integer</code>	0 - 启用1 - 禁用	监控项状态
	<code>value_type</code>	<code>integer</code>	0 - 浮点数1 - 字符2 - 日志3 - 数字(无正负)4 - 文本	接收值类型。
	<code>allowed_hosts</code>	<code>string</code>		如果 'type' 为2, 发送数据的主机的地址(逗号分隔)。
	<code>units</code>	<code>string</code>		返回值单位 (bps)。
	<code>snmpv3_contextname</code>	<code>string</code>		SNMPv3上下文名
	<code>snmpv3_securityname</code>	<code>string</code>		SNMPv3安全名称
	<code>snmpv3_securitylevel</code>	<code>integer</code>	0 - 非加密1 - authNoPriv2 - authPriv	SNMPv3 加密级别
	<code>snmpv3_authprotocol</code>	<code>integer</code>	0 - MD51 - SHA	SNMPv3 认证协议
	<code>snmpv3_authpassphrase</code>	<code>string</code>		SNMPv3 认证密码。
	<code>snmpv3_privprotocol</code>	<code>integer</code>	0 - DES1 - AES	SNMPv3 隐私协议
	<code>snmpv3_privpassphrase</code>	<code>string</code>		SNMPv3 SNMPv3 密码。
	<code>formula</code>	<code>string</code>		如果 "multiplier" 用于计算接收值的值
	<code>delay_flex</code>	<code>string</code>		监控项数据采集频率。
	<code>params</code>	<code>text</code>		如果 'type' 为 13, 14, "执行脚本名称" 如果 'type' 11, "SQL查询" 如果 'type' 为 1,

				如果 'type' 为 1 , "公式"字段
	ipmi_sensor	string		如果 'type' 为 12, IPMI传感器
	authtype	integer	0 - 密码1 - 键	如果 'type' 为 认证类型
	username	string		如果 'type' 为 11,13,14, 用户
	password	string		如果 'type' 为 11,13,14, 密码
	publickey	string		如果 'type' 为 公钥文件的名称
	privatekey	string		如果 'type' 为 私钥文件的名称
	port	string		监控项自定义端口
	description	text		监控项说明
	inventory_link	integer	0 - 非链接数字_ - "host_inventory"表中的字段数	使用监控项值填入 存字段。
	logtimefmt	string		日志条目中的时间格式。仅用于日志项。
	interface_ref	string		参考主机接口。
value map				值映射。
	name	string		用于监控项的值映射名称。
applications				应用集根元素。
application				单个应用集。
	name			应用集名称。
preprocessing				监控项值预处理。
step				监控项值预处理步
	type	integer	1 - 自定义乘数2 - 右修剪3 - 左修剪4 - 两边修剪5 - 正则表达式匹配6 - 布尔值转十进制7 - 八进制转十进制8 - 十六进制转十进制9 - 差量；计算为(接收值 - 先前值)10 - 差量每秒速率；计算为(接收值 - 上一个值) / (上次检查的时间)	监控项值预处理步型。
	params	string		监控项值预处理步参数。

主机自动发现规则标签

元素	元素属性	类型	范围	说明
discoveryrules				自动发现规则的根元素
discovery_rule				单个自动发现规则
	对于大多数元素标签值，请参阅常规监控项的元素标签值。下面仅描述特定于自动发现规则的标签。			
	lifetime	string	0 - 3650	保持由LLD规则发现的资源丢失的天数。
filter				单个过滤器
	evaltype	integer	0 - 和/或逻辑 1 - 和逻辑2 - 或逻辑 3 - 自定义表达式	用于检查自动发现规则过滤条件的逻辑。
	formula	string		过滤条件的自定义计算公式。
	conditions			过滤条件的根元素
condition				单个过滤条件
	macro	string		自动发现宏名称
	value	string		过滤器值：正则表达式或全局正则表达式。
	operator	integer		.
	formulaid	character		过滤条件ID。用于定制计算公式。
item_prototypes				监控项原型 (item_prototypes) 的根元素
item_prototype				单个监控项原型 (item_prototypes)
	对于大多数元素标签值，请参阅常规监控项的元素标签值。下面仅描述特定于监控项原型 (itemprototypes) 的标签。			
application_prototypes				应用集原型的根元素
application_prototype				单个应用集原型

	name			应用集原型名称
--	------	--	--	---------

主机触发器标签

元素	元素属性	类型	范围	说明
triggers				触发器根元素
trigger				单个触发器
	expression	string		触发器表达式
	recovery_mode	integer	0 - 表达式 1 - 恢复表达式 2 - 无	生成OK事件的基础
	recovery_expression	string		触发器恢复表达式
	name	string		触发器名称
	correlation_mode	integer	0 - 无事件相关 1 - 事件相关的标签	相关模式
	correlation_tag	string		用于事件相关的标签名称。
	url	string		触发器URL
	status	integer	0 - 启用 1 - 禁用	触发器状态
	priority	integer	0 - 未分类 1 - 信息 2 - 警告 3 - 一般严重 4 - 严重 5 - 灾难	触发严重性
	description	text		触发器描述
	type	integer	0 - 单个问题事件 1 - 多重问题事件	事件生成类型
	manual_close	integer	0 - 不允许 1 - 允许	手动关闭问题事件。
dependencies				依赖关系的根元素
dependency				单个依赖关系
	name	string		依赖关系触发器名称
	expression	string		依赖关系触发表达式
	recovery_expression	string		依赖关系触发表达式
tags				事件标签的根元素
tag				单个事件标签
	tag	string		标签名称
	value	string		标签值

主机图形标签

元素	元素属性	类型	范围	说明
graphs				图形的根元素
graph				单个图形
	name	string		图形名称
	width	integer		图形宽度, 以像素为单位。用于预览和饼图/爆炸图。
	height	integer		图形高度, 以像素为单位。用于预览和饼图/爆炸图。
	yaxismin	double		如果“ymin_type_1”为1, Y轴的最小值。
	yaxismax	double		如果“ymax_type_1”为1, Y轴的最大值。
	show_work_period	integer	0 - 否1 - 是	如果“type”为0, 1, 则突出显示非工作时间。
	show_triggers	integer	0 - 否1 - 是	如果'type'为1, 显示简单的触发值作为一行
	type	integer	0 - 正常1 - 层积的2 - 饼图3 - 分裂的4 - 3D饼图5 - 3D分裂式饼图	图形类型
	show_legend	integer	0 - 否1 - 是	显示图例
	show_3d	integer	0 - 2D1 - 3D	如果'type'为2、3, 启用3D风格
	percent_left	double		如果'type'为0, 显示左轴的百分位线
	percent_right	double		如果'type'为0, 显示右轴的百分位线
	ymin_type_1	integer	0 - 可计算的1 - 固定的2 - 所选监控项的最后一个值	如果'type'为0、1, Y轴的最小值
	ymax_type_1	integer	0 - 可计算的1 - 固定的2 - 所选监控项的最后一个值	如果'type'为0、1, Y轴的最大值
	ymin_item_1	string	空或监控项细节	如果'ymin_type_1'为2, 监控项细节
	ymax_item_1	string	空或监控项细节	如果'ymax_type_1'为2, 监控项细节
graph_items				图形监控项的根元素
graph_item				单个图形监控项

	sortorder	integer		绘制顺序。较小的值首先绘制。可用于绘制线条或区域后面(或前面)的另一个。
	drawtype	integer	0 - single line 1 - filled region 2 - bold line 3 - dotted line 4 - dashed line	如果图形'type'为0, 绘图风格
	color	string		元素颜色(6位十六进制)
	yaxisside	integer	0 - 左侧 1 - 右侧	如果图形'type'为0, 元素所属的Y轴位置(左或右)
	calc_fnc	integer	1 - 最小 2 - 平均 4 - 最大 7 - 所有(如果图形'type'为0, 最小, 平均和最大) 9 - 最新(如果图形'type'不为0, 1)	如果一个监控项存在多个值, 则绘制数据。
	type	integer	1 - 该监控项的值成比例地表示在饼图 2 - 监控项的值代表整个饼图(图形总和)	饼图/分裂式饼图的绘图风格。
item				单个监控项
	host	string		监控项主机
	key	string		监控项键

主机Web监测标签

元素	元素属性	类型	范围	说明
httptests				Web监测的根元素。
httptest				单个Web监测
	name	string		Web监测名称
	delay	integer		执行Web监测的频率, 以秒为单位。
	attempts	integer	1-10	执行Web监测步骤的尝试次数。
	agent	string		客户代理。Zabbix将假装成为所选的浏览器。不同的浏览器返回不同的内容时, 这很有用。
	http_proxy	string		使用以下格式指定要使用的HTTP代理: <code>http://[username[:password]@]proxy.mycompany.com:8080</code>
	variables	text		在场景步骤中可以使用的场景级变量(宏)列表
	headers	text		执行请求时将发送的HTTP头。
	status	integer	0 - 启用 1 - 禁用	Web监测状态
			0 - none 1 - 基	

			基础的2 - NTLM	
	http_user	string		验证用户名。
	http_password	string		指定用户名的验证密码。
	verify_peer	integer	0 - 否1 - 是	验证Web服务器的SSL证书。
	verify_host	integer	0 - 否1 - 是	验证Web服务器证书的公用名称字段或主题备用匹配。
	ssl_cert_file	string		用于客户端身份验证的SSL证书文件的名称。
	ssl_key_file	string		用于客户端身份验证的SSL私钥文件的名称。
	ssl_key_password	string		SSL私钥文件密码。
steps				Web监测步骤的根元素。
step				单个Web监测步骤
	name	string		Web监测步骤名称
	url	string		监控的URL
	posts	text		'Post' 变量列表
	variables	text		此步骤后应用的步级变量（宏）列表。如果带有'regex: '前缀，则根据'regex: '前缀之模式模式从该步骤返回的数据中提取其值
	headers	text		执行请求时将发送的HTTP头。
	follow_redirects	integer	0 - 否1 - 是	遵循HTTP重定向
	retrieve_mode	integer	0 - 内容1 - 仅 头信息	HTTP响应检索模式。
	timeout	integer		超时执行步骤，以秒为单位。
	required	string		必填字符串。如果空，则忽略。
	status_codes	string		以逗号分隔的已接受的状态代码列表。如果空例如： 200-201, 210-299



4 网络拓扑图

概述

网络拓扑图 [导出](#) 包含：

- 所有相关图像
- 拓扑图结构 - 所有拓扑图设置，全部包含其设置的元素，拓扑图链接和拓扑图链接状态指示器

未导出的是主机组，主机，触发器，其他拓扑图或可能与导出的拓扑图相关的任何其他元素。因此，如果拓扑图引用的至少一个元素丢失，则导入它将失败。

Zabbix 1.8.2 开始支持网络拓扑图导出/导入。

导出

要导出网络拓扑图，请执行以下操作：

- 转到：监测中 - > 拓扑图
- 标记要导出的网络拓扑图的复选框
- 点击列表下面的导出

<input type="checkbox"/>	Name	Width	Height
<input checked="" type="checkbox"/>	Network	590	400
<input type="checkbox"/>	Offices	700	550
<input type="checkbox"/>	User map	800	600

1 selected [Export](#) [Delete](#)

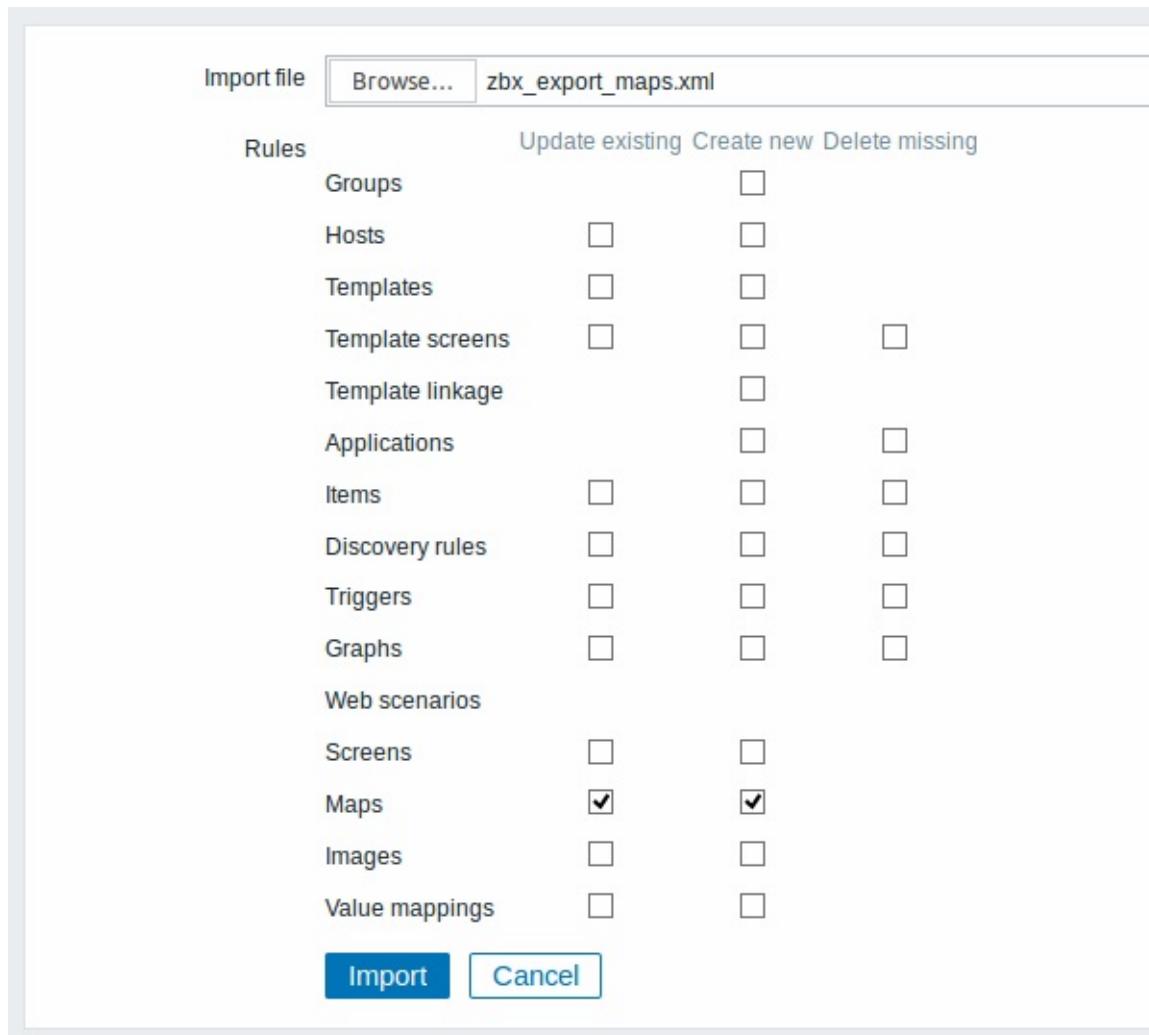
所选拓扑图导出为默认名称为 `zabbix_export_maps.xml` 的本地 XML 文件。

导入

要导入网络拓扑图，请执行以下操作：

- 转到：监测中 - > 拓扑图
- 点击右边的 导入
- 选择导入文件
- 在导入规则中标记所需的选项

- 点击导入



导入的成功或失败消息将显示在前端。

导入规则：

规则	说明
更新已存在	将使用从导入文件获取的数据更新现有拓扑图。 否则不会更新。
创建新的	导入将使用导入文件中的数据添加新的拓扑图。 否则不会添加它们。

如果您取消选中两个拓扑图选项，并检查图像的相应选项，则只会导入图像。 图像导入仅适用于Zabbix Super Admin用户。

如果替换现有的图像，它将影响使用此图像的所有拓扑图。

导出格式

导出一个小的网络拓扑图与三个元素，他们的图像和它们之间的一些链接。 请注意，图像被截断以节省空间

```

1. <?xml version="1.0" encoding="UTF-8"?>
2. <zabbix_export>
3.   <version>3.4</version>
4.   <date>2016-10-05T08:16:20Z</date>

```

```

5.    <images>
6.        <image>
7.            <name>Server_(64)</name>
8.            <imagetype>1</imagetype>
9.            <encodedImage>iVBOR...SuQmCC</encodedImage>
10.       </image>
11.       <image>
12.           <name>Workstation_(64)</name>
13.           <imagetype>1</imagetype>
14.           <encodedImage>iVBOR...SuQmCC</encodedImage>
15.       </image>
16.       <image>
17.           <name>Zabbix_server_3D_(96)</name>
18.           <imagetype>1</imagetype>
19.           <encodedImage>iVBOR...ggg==</encodedImage>
20.       </image>
21.   </images>
22.   <maps>
23.       <map>
24.           <name>Network</name>
25.           <width>590</width>
26.           <height>400</height>
27.           <label_type>0</label_type>
28.           <label_location>0</label_location>
29.           <highlight>1</highlight>
30.           <expandproblem>0</expandproblem>
31.           <markelements>1</markelements>
32.           <show_unack>0</show_unack>
33.           <severity_min>2</severity_min>
34.           <grid_size>40</grid_size>
35.           <grid_show>1</grid_show>
36.           <grid_align>1</grid_align>
37.           <label_format>0</label_format>
38.           <label_type_host>2</label_type_host>
39.           <label_type_hostgroup>2</label_type_hostgroup>
40.           <label_type_trigger>2</label_type_trigger>
41.           <label_type_map>2</label_type_map>
42.           <label_type_image>2</label_type_image>
43.           <label_string_host/>
44.           <label_string_hostgroup/>
45.           <label_string_trigger/>
46.           <label_string_map/>
47.           <label_string_image/>
48.           <expand_macros>0</expand_macros>
49.           <background/>
50.           <iconmap/>
51.           <urls/>
52.           <selements>
53.               <selement>
54.                   <elementtype>0</elementtype>
55.                   <label>Host 1</label>
56.                   <label_location>-1</label_location>
57.                   <x>476</x>

```

```
58.          <y>28</y>
59.          <elementsubtype>0</elementsubtype>
60.          <areatype>0</areatype>
61.          <width>200</width>
62.          <height>200</height>
63.          <viewtype>0</viewtype>
64.          <use_iconmap>0</use_iconmap>
65.          <selementid>8</selementid>
66.          <element>
67.              <host>Discovered host</host>
68.          </element>
69.          <icon_off>
70.              <name>Server_(64)</name>
71.          </icon_off>
72.          <icon_on/>
73.          <icon_disabled/>
74.          <icon_maintenance/>
75.          <application/>
76.          <urls/>
77.      </selement>
78.      <selement>
79.          <elementtype>0</elementtype>
80.          <label>Zabbix server</label>
81.          <label_location>-1</label_location>
82.          <x>252</x>
83.          <y>50</y>
84.          <elementsubtype>0</elementsubtype>
85.          <areatype>0</areatype>
86.          <width>200</width>
87.          <height>200</height>
88.          <viewtype>0</viewtype>
89.          <use_iconmap>0</use_iconmap>
90.          <selementid>6</selementid>
91.          <element>
92.              <host>Zabbix server</host>
93.          </element>
94.          <icon_off>
95.              <name>Zabbix_server_3D_(96)</name>
96.          </icon_off>
97.          <icon_on/>
98.          <icon_disabled/>
99.          <icon_maintenance/>
100.         <application/>
101.         <urls/>
102.     </selement>
103.     <selement>
104.         <elementtype>0</elementtype>
105.         <label>New host</label>
106.         <label_location>-1</label_location>
107.         <x>308</x>
108.         <y>230</y>
109.         <elementsubtype>0</elementsubtype>
110.         <areatype>0</areatype>
```

```

111.           <width>200</width>
112.           <height>200</height>
113.           <viewtype>0</viewtype>
114.           <use_iconmap>0</use_iconmap>
115.           <selementid>7</selementid>
116.           <element>
117.             <host>Zabbix host</host>
118.           </element>
119.           <icon_off>
120.             <name>Workstation_(64)</name>
121.           </icon_off>
122.           <icon_on/>
123.           <icon_disabled/>
124.           <icon_maintenance/>
125.           <application/>
126.           <urls/>
127.         </selement>
128.       </selements>
129.       <links>
130.         <link>
131.           <drawtype>0</drawtype>
132.           <color>008800</color>
133.           <label/>
134.           <selementid1>6</selementid1>
135.           <selementid2>8</selementid2>
136.           <linktriggers/>
137.         </link>
138.         <link>
139.           <drawtype>2</drawtype>
140.           <color>00CC00</color>
141.           <label>100Mbps</label>
142.           <selementid1>7</selementid1>
143.           <selementid2>6</selementid2>
144.           <linktriggers>
145.             <linktrigger>
146.               <drawtype>0</drawtype>
147.               <color>DD0000</color>
148.               <trigger>
149.                 <description>Zabbix agent on {HOST.NAME} is unreachable for 5
 minutes</description>
150.               <expression>{Zabbix host:agent.ping.nodata(5m)}=1</expression>
151.               <recovery_expression/>
152.             </trigger>
153.           </linktrigger>
154.         </linktriggers>
155.       </link>
156.     </links>
157.   </map>
158. </maps>
159. </zabbix_export>

```

元素标签

元素标签值在下表中说明。

元素	元素属性	类型	范围	描述
images				图片的根元素。
image				单个图片
	name	string		唯一的图片名称。
	imagetype	integer	1 - 图标2 - 背景	图片类型
	encodedImage			Base64编码图片
maps				拓扑图的根元素
map				单个拓扑图
	name	string		唯一的拓扑图名称
	width	integer		拓扑图宽度，以像素为单位。
	height	integer		拓扑图高度，以像素为单位。
	label_type	integer	0 - 标签1 - 主机IP地址2 - 元素名称3 - 仅状态4 - 没有	拓扑图元素标签类型
	label_location	integer	0 - 底部1 - 左侧2 - 右侧3 - 顶部	默认情况下拓扑图元素标签位置
			0 - 否1	为活动触发器和主机状态启用图

			-是	出显示。
	expandproblem	integer	0 - 否 1 - 是	显示具有单个问题的元素的问题器。
	markelements	integer	0 - 否 1 - 是	突出显示最近更改其状态的拓扑素。
	show_unack	integer	0 - 计数所有问题 1 - 计算未确认的问题 2 - 分别计算确认和未确认的问题	问题显示。
	severity_min	integer	0 - 未分类 1 - 信息 2 - 警告 3 - 一般严重 4 - 严重 5 - 灾难	默认情况下，在拓扑图上显示的触发严重性。
			20,	

	grid_size	integer	20, 40, 50, 75 或 100	如果“grid_show=1”，拓扑图的单元格大小（以像素为单位）
	grid_show	integer	0 - 是1 - 否	在拓扑图配置中显示网格。
	grid_align	integer	0 - 是1 - 否	在拓扑图配置中自动对齐图标。
	label_format	integer	0 - 否1 - 是	使用高级标签配置。
	label_type_host	integer	0 - 标签1 - 主机IP地址2 - 元素名称3 - 仅状态4 - 无5 - 自定义标签	如果“label_format=1”，显示主机标签
	label_type_hostgroup	integer	0 - 标签2 - 元素名称3 - 仅状态4 - 无5 - 自定义	如果“label_format=1”，显示主机组标签

			签	
	label_type_trigger	integer	0 - 标签2 - 元素名称3 - 仅状态4 - 无5 - 自定义标签	如果“label_format=1”，显示触发标签
	label_type_map	integer	0 - 标签2 - 元素名称3 - 仅状态4 - 无5 - 自定义标签	如果“label_format=1”，显示拓扑图标签
	label_type_image	integer	0 - 标签2 - 元素名称4 - 无5 - 自定义标签	如果“label_format=1”，显示图片标签
	label_string_host	string		如果“label_type_host=5”，元素的自定义标签
	label_string_hostgroup	string		如果“label_type_hostgroup=5”，主机组元素的自定义标签

				主机组元素的自定义标签
	label_string_trigger	string		如果“label_type_trigger=触发元素的自定义标签”
	label_string_map	string		如果“label_type_map=5”，图元素的自定义标签
	label_string_image	string		如果“label_type_image=5”像元素的自定义标签
	expand_macros	integer	0 - 否 1 - 是	在拓扑图配置中展开标签中的宏
	background	id		如果“imagetype=2”，背景图ID（如果有）
	iconmap	id		图标映射的ID（如果有）。
urls				
url				单个URL。
	name	string		链接名称
	url	string		链接 URL。
	elementtype	integer	0 - 主机1 - 拓扑图2 - 触发器3 - 主机组4 - 图片	Map item type the link belongs to.
selements				
selement				单个拓扑图元素
	elementtype	integer	0 - 主机1 - 拓扑图2 - 触发器3 - 主机组4	Map element type.

			片	
	label	string		图标标签。
	label_location	integer	-1 - 使用拓扑图默认0 - 底部1 - 左侧2 - 右侧3 - 顶部	
	x	integer		X轴位置。
	y	integer		Y轴位置。
	elements subtype	integer	0 - - 单个主机组1 - 所有主机组	如果“elementtype=3”，元素型
	areatype	integer	0 - - 和整个拓扑图一样1 - 自定义大小	如果“elements subtype=1”， 大小
	width	integer		如果 “areatype=1”，区域宽
	height	integer		如果 “areatype=1”，区域高
			0 - - 均匀地	

	viewtype	integer	地 放 在 该 区 域	如果“elements_subtype=1”， 布局算法
	use_iconmap	integer	0 - 否1 - 是	为此元素使用图标映射。 只有在 拓扑图级别激活了图标映射功能才 有意义。
	selementid	id		唯一元素记录ID。
	application	string		应用名称过滤器 如果给出了应 用程序名称，则只有属于给定应用程 序触发器的问题才会显示在拓扑图
element				在拓扑图上表示的单个Zabbix 元素 (拓扑图, 主机组, 主机, 等)
	host			
icon_off				当元素处于“确定”状态时使用 的图片。
icon_on				当元素处于“问题”状态时使用 的图片。
icon_disabled				元素被禁用时使用的图片。
icon_maintenance				元素在维护中使用的图片。
	name	string		唯一图片名称
links				
link				拓扑图元素之间的单独链接。
	drawtype	integer	0 - 线2 - 粗 线3 - 点4 - 虚 线	链接风格。
	color	string		链接颜色 (6位, 十六进制)。
	label	string		链接标签
	selementid1	id		要连接的一个元素的ID。
	selementid2	id		要连接的另一个元素的ID。
linktriggers				
linktrigger				单个链接状态指示器。
	drawtype	integer	0 - 线2 - 粗 线3 - 点4	触发器处于“问题”状态时的链接 方式。

			虚线	
	color	string		当触发器处于“问题”状态时，键色（6位，十六进制）。
trigger				用于指示链接状态的触发器。
	description	string		触发器名称。
	expression	string		触发表达式。
	recovery_expression	string		触发恢复表达式。



5 聚合图形

概述

聚合图形 [导出](#) 包含聚合图形结构 - 所有聚合图形设置以及所有聚合图形元素及其配置。

聚合图形本身（如主机，主机组或任何其他数据）中的任何内容都不会导出。因此，如果聚合图形引用的至少一个元素丢失，则导入它将失败。

导出

要导出聚合图形，请执行以下操作：

- 转到：监测中 - > 聚合图形
- 标记要导出的聚合图形的复选框
- 点击列表的下面导出

Name	Dimension (cols x rows)
Servers	2 x 3
Zabbix server	2 x 3
Zabbix server2	3 x 3

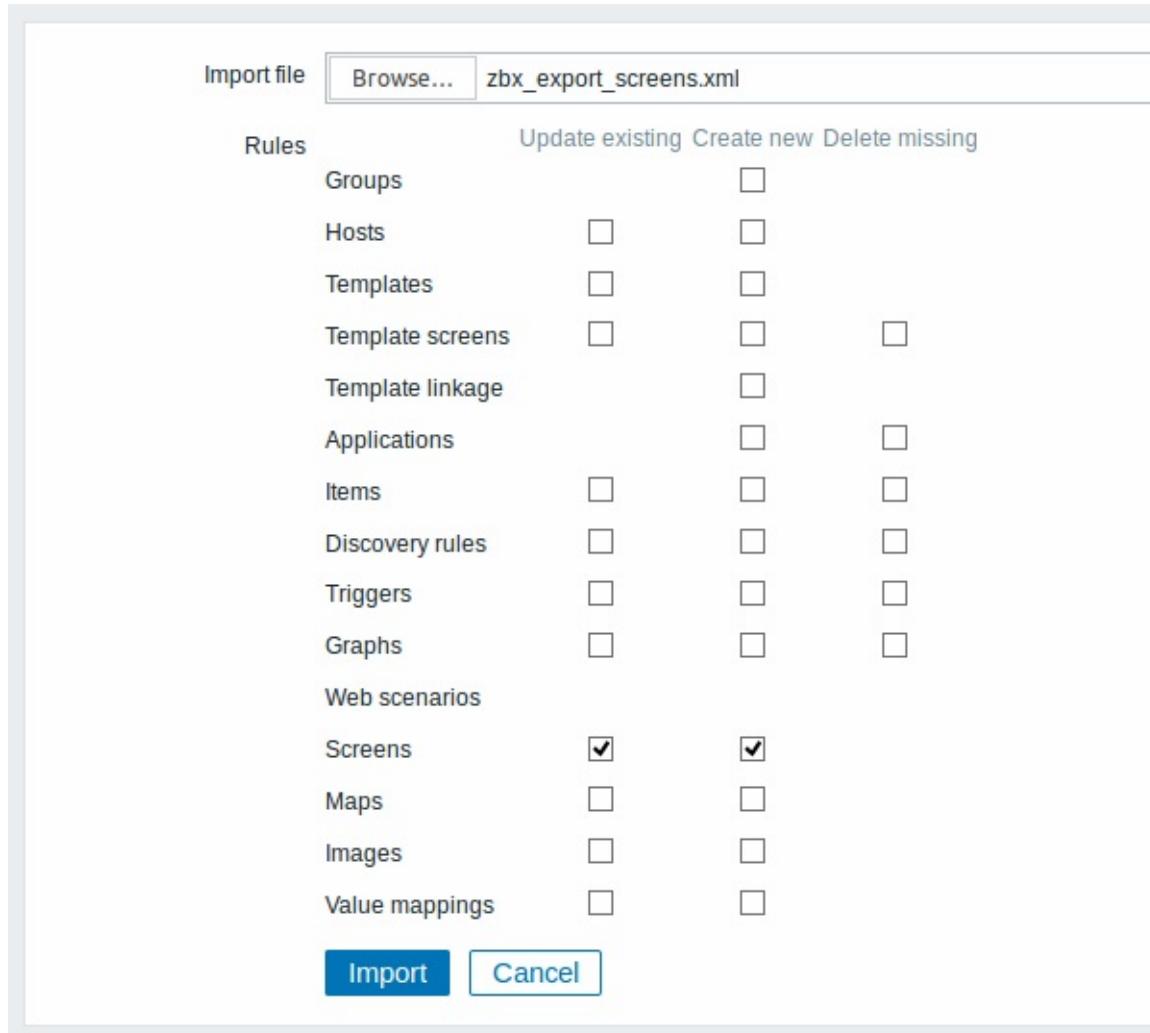
1 selected **Export** Delete

所选聚合图形将导出为默认名称为`zabbix_export_screens.xml`的本地XML文件。

导入

要导入聚合图形，请执行以下操作：

- 转到：监测中 - > 聚合图形
- 点击右侧的 导入
- 选择导入文件
- 在导入规则中标记所需的选项
- 点击 导入



导入的成功或失败消息将显示在前端。

导入规则：

规则	说明
更新已存在	将使用从导入文件获取的数据更新现有聚合图形。 否则不会更新。
创建新的	导入将使用导入文件中的数据添加新聚合图形。 否则不会添加它们。

导出格式

导出一个小聚合图形，两个图形占据聚合图形的第一行。

```

1. <?xml version="1.0" encoding="UTF-8"?>
2. <zabbix_export>
3.   <version>3.4</version>
4.   <date>2016-10-07T08:02:40Z</date>
5.   <screens>
6.     <screen>
7.       <name>Zabbix server</name>
8.       <hsize>2</hsize>
9.       <vsize>3</vsize>
10.      <screen_items>
11.        <screen_item>
```

```
12.          <resourcetype>0</resourcetype>
13.          <width>300</width>
14.          <height>80</height>
15.          <x>0</x>
16.          <y>0</y>
17.          <colspan>1</colspan>
18.          <rowspan>1</rowspan>
19.          <elements>0</elements>
20.          <valign>0</valign>
21.          <halign>0</halign>
22.          <style>0</style>
23.          <url/>
24.          <dynamic>1</dynamic>
25.          <sort_triggers>0</sort_triggers>
26.          <resource>
27.              <name>CPU load</name>
28.              <host>Zabbix host</host>
29.          </resource>
30.          <max_columns>3</max_columns>
31.          <application/>
32.      </screen_item>
33.      <screen_item>
34.          <resourcetype>0</resourcetype>
35.          <width>300</width>
36.          <height>80</height>
37.          <x>1</x>
38.          <y>0</y>
39.          <colspan>1</colspan>
40.          <rowspan>1</rowspan>
41.          <elements>0</elements>
42.          <valign>0</valign>
43.          <halign>0</halign>
44.          <style>0</style>
45.          <url/>
46.          <dynamic>1</dynamic>
47.          <sort_triggers>0</sort_triggers>
48.          <resource>
49.              <name>CPU utilization</name>
50.              <host>Zabbix host</host>
51.          </resource>
52.          <max_columns>3</max_columns>
53.          <application/>
54.      </screen_item>
55.  </screen_items>
56. </screen>
57. </screens>
58. </zabbix_export>
```

元素标签

元素标签值在下表中说明。

元素	元素性情	类型	范围	说明
screens				
screen				
	name	string		唯一聚合图形名称。
	hsize	integer		水平尺寸，列数。
	vsize	integer		垂直尺寸，行数。
screen_items				
screen_item				
	resourcetype	integer	0 - 图形1 - 简单图形2 - 拓扑图3 - 纯文本4 - 主机信息5 - 触发器信息6 - 服务器信息7 - 时钟8 - 聚合图形9 - 触发器概览10 - 数据概览11 URL12 - 动作日志13 - 事件历史14 - 主机群组问题15 - 系统状态16 - 主机问题19 - 简单图形原型20 - 图形原型	资源类型。
	width	integer		如果 'resourcetype' 为 0, 1, 7, 11, 19 or 20, 聚合图形项目的宽度 (以像素为单位)
	height	integer		如果 'resourcetype' 为 0, 1, 7, 11, 19 or 20, 聚合图形项目的高度 (以像素为单位)
	x	integer		聚合图形上的X坐标, 从左到右。'0'表示从第一列开始。
	y	integer		聚合图形上的Y坐标, 从上到下。'0'表示从第一行开始。
	colspan	integer		聚合图形项目跨越的列数。
	rowspan	integer		聚合图形项目跨越的数字或行。

	elements	integer		如果 'resourcetype' 是 3, 12, 13, 14 or 16, 聚合图形上显示的行数
	valign	integer	0 - 居中 (默认)1 - 顶部2 - 底部	垂直对齐。
	halign	integer	0 - 居中 (默认)1 - 左侧2 - 右侧	水平对齐。
	style	integer	0 - 纯文本1 - HTML	如果 'resourcetype' 为 3, 显示聚合图形项目的选项
integer	0 - 本地时间1 - 服务器时间2 - 主机时间	如果 'resourcetype' 为 7, 显示聚合图形项目的选项		
integer	0 - 水平1 - 垂直	如果 'resourcetype' 为 4, 5, 显示聚合图形项目的选项		
integer	0 - 左边1 - 顶部	如果 'resourcetype' 为 9, 10, 显示聚合图形项目的选项		
	url	string		如果 'resourcetype' 为 11, 链接 URL
	dynamic	integer	0 - 否1 - 是	如果 'resourcetype' 为 0, 1, 3, 19 或 20, 动态监控项
	sort_triggers	integer	0 - 最后更改(降序)1 - 严重性(降序)2 - 主机(升序)	如果 'resourcetype' 为 14, 16, 对触发器进行排序的选项
integer	3 - 时间(升序) 4 - 时间(降序) 5 - 类型(升序) 6 - 类型(降序) 7 - 状态(升序) 8 - 状态(降序) 9 - 重试(升序) 10 - 重试(降序) 11 - 接收者(升序) 12 - 接收者(降序)	如果 'resourcetype' 为 12, 对触发器进行排序的选项		
	max_columns	integer		如果 'resourcetype' 为 19 or 20, 应在聚合图形单元格中显示生成的图表列数当有许多LLD生成的图表时很有用。
				如果

	application	string		'resourcetype' 为 9 or 10, 按应用程序名称过滤
resource				
	name	string		资源名称
	host	string		资源主机

15. 发现

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

1 网络发现

概述

Zabbix提供了有效和非常灵活的网络自动发现功能。

当网络发现正确设置后你可以：

- 加快Zabbix部署
- 简化管理
- 无需过多管理就能在快速变化的环境中使用Zabbix

Zabbix网络发现基于以下信息：

- IP范围
- 可用的外部服务 (FTP, SSH, WEB, POP3, IMAP, TCP等)
- 来自 zabbix agent 的信息 (仅支持未加密模式)
- 来自 snmp agent 的信息

不支持：

- 发现网络拓扑

网络发现由两个阶段组成：发现 (discovery) 和动作 (actions)。

发现

Zabbix定期扫描[网络发现规则](#)中定义的IP范围，并为每个规则单独配置检查的频次。

请注意，一个发现规则始终由单一发现进程处理，IP范围主机不会被分拆到多个发现进程处理。

每个规则中都定义了一组需要检测的服务。

发现检查与其他检查独立处理。如果任何检查未找到服务（或失败），则仍会处理其他检查

网络发现模块每次检测到 service 和 host(IP)都会生成一个 discovery 事件

事件	条件
<i>Service Discovered</i>	服务首次被发现或者由 'down' 变 'up'
<i>Service Up</i>	服务持续 'up'
<i>Service Lost</i>	服务由 'up' 变 'down'
<i>Service Down</i>	服务持续 'down'
<i>Host Discovered</i>	在主机的所有服务都 'down' 之后，至少一个服务是 'up'。
<i>Host Up</i>	主机至少有一个服务是 'up' 状态

<i>Host Lost</i>	主机的所有服务在至少一个是 'up' 之后全部是 'down'。
<i>Host Down</i>	所有服务都持续 'down'

动作

zabbix 所有 [动作](#) 都是基于发现事件, 例如:

- 发送通知
- 添加/删除主机
- 启用/禁用主机
- 添加主机到组
- 从组中删除主机
- 将主机链接到模板/从模板中取消链接
- 执行远程脚本命令

基于事件的网络发现动作, 可以根据设备类型、IP地址、状态、运行时间/停机时间等进行配置, 查看动作[操作](#)和[条件](#)页面。

创建主机

A host is added if the *Add host* operation is selected. A host is also added, even if the *Add host* operation is missing, if you select operations resulting in actions on a host. Such operations are:如果在[动作→操作](#)选择添加主机操作, 那么主机会被添加, 即使添加主机操作未被执行, 通过下列的操作仍然可以添加主机, 这样的操作是:

- enable host
- 启用主机
- disable host
- 禁用主机
- add host to a host group
- 添加主机到主机组
- link template to a host
- 将主机链接到模板

When adding hosts, a host name is the result of reverse DNS lookup or IP address if reverse lookup fails. Lookup is performed from the Zabbix server or Zabbix proxy, depending on which is doing the discovery. If lookup fails on the proxy, it is not retried on the server. If the host with such a name already exists, the next host would get _2 appended to the name, then _3 and so on.当添加主机时, 如果反向查找失败, 那么主

机名就是DNS反向查找的结果或者是IP地址。查找是从Zabbix服务器或Zabbix代理执行的，具体取决于自动发现的执行。如果在Zabbix代理上查找失败，则不会在Zabbix服务器上重试。如果具有相同名称的主机已经存在，那么下一个主机将会把_2附加在主机名后，依次附加_3等。

Created hosts are added to the *Discovered hosts* group (by default, configurable in *Administration* → *General* → *Other*). If you wish hosts to be added to another group, add a *Remove from host groups* operation (specifying “*Discovered hosts*”) and also add an *Add to host groups* operation (specifying another host group), because a host must belong to a host group. 创建的主机会被添加到主机群组中的*Discovered hosts*下（默认情况下，在管理→一般→其他可以进行配置），如果希望将主机添加到另一个主机群组中，可以从动作→操作选择添加一个从主机群组中删除的操作类型（需要指定“*Discovered hosts*”），当然也可以选择添加到主机群组的操作类型（需要指定其他的主机群组），因为主机必须属于主机群组。

If a host already exists with the discovered IP address, a new host is not created. However, if the discovery action contains operations (link template, add to host group, etc), they are performed on the existing host. 如果主机已经存在，且自动发现中同时存在已发现的IP地址，那么将不会创建新的主机，但是，如果自动发现的操作包含（链接模板，添加到主机群组等），则会在已经存在的主机上执行相应的操作。

Host removal

移除主机

Since Zabbix 2.4.0, hosts created by a network discovery rule are deleted automatically if a discovered entity is not in the rule's IP range any more. Hosts are deleted immediately. 从Zabbix 2.4.0开始，如果已发现的实体不在自动发现规则的IP范围内，则由网络发现规则创建的主机将会被自动删除。主机将立即删除

Interface creation when adding hosts

添加主机时的创建接口

When hosts are added as a result of network discovery, they get interfaces created according to these rules: 当网络自动发现的主机被添加时，它们是根据以下规律来创建的

- the services detected - for example, if an SNMP check succeeded, an SNMP interface will be created
- 检测到服务 - 例如，如果SNMP检查成功，那么将会创建一个SNMP接口；
- if a host responded both to Zabbix agent and SNMP requests, both types of interfaces will be created
- 如果主机响应Zabbix agent和SNMP的请求，那么这两种类型的接口都会被创建；
- if uniqueness criteria are Zabbix agent or SNMP-returned data, the first interface found for a host will be created as the default one. Other IP addresses will be added as additional interfaces.
- 如果唯一性准则是Zabbix agent键值或是SNMP OID返回的数据，这第一个接口发现的主机将会被创建，而这个接口将会被作为默认接口，其他IP地址将会作为附加接口被添加。

- if a host responded to agent checks only, it will be created with an agent interface only. If it would start responding to SNMP later, additional SNMP interfaces would be added.
- 如果主机只响应agent检查，则只能使用agent接口来创建。如果稍后开始响应SNMP的检查，那么将添加SNMP接口为附加接口。
- if 3 separate hosts were initially created, having been discovered by the “IP” uniqueness criteria, and then the discovery rule is modified so that hosts A, B and C have identical uniqueness criteria result, B and C are created as additional interfaces for A, the first host. The individual hosts B and C remain. In *Monitoring → Discovery* the added interfaces will be displayed in the “Discovered device” column, in black font and indented, but the “Monitored host” column will only display A, the first created host. “Uptime/Downtime” is not measured for IPs that are considered to be additional interfaces.
- 如果最初创建了3个独立的主机，他们都被自动发现的唯一性准则“IP”发现，然后修改自动发现规则，为了使A、B和C自动发现的唯一性准则结果是相同的，那么接口B和C作为接口A的附加接口来创建第一个主机。主机B和C作为个体主机仍然存在。在监控中 → 自动发现中，添加的接口将以黑色字体和缩进形式显示在“已发现的设备”这一列中，但在“已监测的主机”这一列将只显示第一个创建的主机A。由于被认为附加接口的IP，所以不测量主机B和C的“在线时间/断线时间”。

配置网络发现规则

概述

配置Zabbix的网络发现规则去发现主机和服务：

- 首先配置 (*Configuration*) → 自动发现 (*Discovery*)
- 单击创建发现规则 (*Create rule*) (或在自动发现规则名称上编辑现有规则)
- 编辑自动发现规则属性

规则属性



参数	描述
名称 (<i>Name</i>)	规则名称，唯一。 例如：“Local network”.
通过代理发现 (<i>Discovery by proxy</i>)	谁执行当前发现规则： no proxy - Zabbix server 执行发现< proxy name > - 指定的 proxy执行
IP范围 (<i>IP range</i>)	发现规则中的IP地址范围。 格式如下：单个IP: 192.168.1.33 IP段: 192.168.1-10.1-255。 范围受限于覆盖地址的总数（小于64K）。子网掩码: 192.168.4.0/24 支持的子网掩码: IPv4: /16 - /30 IPv6: /112 - /128 IP列表: 192.168.1.1-255, 192.168.2.1-100, 192.168.2.200, 192.168.4.0/24 从Zabbix 3.0.0起，此字段支持空格，表格和多行。
延迟 (<i>Delay</i>) (秒)	此参数定义Zabbix将执行规则的频率。规则执行完毕之后，要多久才执行下一次。
检查 (<i>Checks</i>)	Zabbix将使用这个检查列表进行发现。支持的checks: SSH, LDAP, SMTP, FTP, HTTP, HTTPS, POP, NNTP, IMAP, TCP, Telnet, Zabbix agent, SNMPv1 agent, SNMPv2 agent, SNMPv3 agent, ICMP ping. 基于协议的发现使用 net.tcp.service[] 功能测试每个主机，但不包括查询SNMP OID的SNMP协议。通过在未加密模式下查询监控项(item)来测试Zabbix agent。有关更多详情，请参阅 agent items 。“端口”参数可以是以下之一：单端口: 22 端口段: 22-45 端口列表: 22-45, 55, 60-70
设备唯一标识 (<i>Device uniqueness criteria</i>)	唯一标准如下： IP地址 - 使用 IP 地址作为设备唯一性标识，不处理多IP设备。如果具有相同IP的设备已经存在，则将认为已经发现，并且不会添加新的主机。 发现检查类型 - 使用 SNMP 或者 Zabbix agent 的 check 作为唯一标识。
启用 (<i>Enabled</i>)	是否启用当前规则。

修改代理(proxy)设置

从Zabbix 2.2.0起，不同的代理发现的主机被认为是不同的主机。虽然这允许在不同子网使用相同的IP段执行发现，但是对已监测子网改变代理复杂，是因为代理的变化也必须应用于所有发现的主机。在发现规则中替换代理的步骤如下：

- 禁用发现规则
- 同步代理配置
- 替换发现规则中的代理
- 替换由此规则发现的所有主机的代理
- 启用发现规则

真实使用场景

例如我们设置IP段为192.168.1.1-192.168.1.254的网络发现规则。

在我们的例子中，我们需要：

- 发现有Zabbix agent运行的主机
- 每10分钟执行一次
- 如果主机正常运行时间超过1小时，添加主机
- 如果主机停机时间超过24小时，删除主机
- 将Linux主机添加到“Linux servers”组
- 将Windows主机添加到“Windows servers”组
- 链接模板 *Template OS Linux* 到Linux主机
- 链接模板 *Template OS Windows* 到Windows主机

步骤1

首先给我们的IP段定义网络发现规则。



Zabbix试图通过连接Zabbix agents并获取`system.uname`键值来发现IP段为192.168.1.1-192.168.1.254中的主机。根据不同键值来对应不同的操作系统的不同操作。例如将Windows服务器链接到Template OS Windows，将Linux服务器链接到Template OS Linux。

规则将每10分钟(600秒)执行一次。

当规则添加后，Zabbix将自动执行发现规则并生成基于发现的事件做后续处理。

步骤2

定义[动作\(action\)](#) 将所发现的Linux服务器添加到相应的组/模板



如果发生以下情况，动作(action)将被激活：

- “Zabbix agent”服务是“up”
- system.uname(规则中定义的Zabbix agent值)包含“Linux”more
- 正常运行时间为1小时 (3600秒) 或更长时间



该动作(action)将执行以下操作：

- 将发现的主机添加到“Linux servers”组 (如果以前未添加主机，也添加主机)
- 链接主机到“Template OS Linux”模板。Zabbix将自动开始使用“Template OS Linux”模板中的项目和触发器来监控主机。

步骤3

定义动作(action) 将所发现的Windows服务器添加到相应的组/模板



步骤4

定义动作删除失联主机



如果“Zabbix agent”服务 'down' 超过24小时 (86400秒)，服务器将被删除。

2 自动注册

概述

活动的Zabbix agent可以自动注册到服务器进行监控。这种方式无需在服务器上手动配置它们。

当以前未知的active agent要求检查时，会发生自动注册。

该功能可能非常方便自动监控新的Cloud节点。一旦在Cloud Zabbix中有一个新节点，Zabbix将自动启动主机监控，并进行性能和可用性数据的收集。

Active agent自动注册还支持对被添加的主机进行被动检查的监控。当active agent要求检查时，提供它配置文件中定义的“ListenIP”或“ListenPort”配置参数，这些参数将发送到服务器。（如果指定了多个IP地址，则第一个将被发送到服务器。）

服务器在添加新的自动注册主机时，使用接收到的IP地址和端口配置agent。如果没有接收到IP地址值，则使用传入连接的IP地址。如果没有接收到端口值，则使用10050。

配置

指定服务器

在agent[配置文件](#)中指定Zabbix server - zabbix_agentd.conf

```
1. ServerActive=10.0.0.1
```

如果你没有在zabbixagentd.conf中特别定义了`_Hostname`，则服务器将使用agent的系统主机名命名主机。Linux中的系统主机名可以通过运行`'hostname'`命令获得。

修改配置后重启agent

active agent自动注册动作

当服务器从agent收到自动注册请求时，它会调用一个[动作](#)。事件源“自动注册”的操作必须配置为agent自动注册。

设置[网络发现](#)不需要使active agents自动注册。

在Zabbix页面，转到配置→动作，选择自动注册为事件源，然后单击创建操作：

- 在“动作”选项卡，定义 Action 名称
- 可选指定条件。如果要使用“主机元数据”条件，请参阅下一节。
- 在“操作”选项卡中，添加“添加主机”，“添加到主机组”（例如，发现的主机），“链接到模板”等。

如果只能主动监视（例如Zabbix服务器被防火墙不允许访问的主机）的自动注册主机，则可能需要创建一个特定的模板，如`Template_Linux-active`以链接到

使用主机元数据

当agent程序向服务器发送自动注册请求时，会发送其主机名。在某些情况下（例如，Amazon云端节点），Zabbix服务器的主机名不足以区分发现的主机。主机元数据可将其他信息从agent发送到服务器。

主机元数据在agent配置文件 - `zabbix_agentd.conf` 中配置。在配置文件中指定主机元数据有两种方式：

1. `HostMetadata`
2. `HostMetadataItem`

请参阅上面链接中的选项说明。

每当活动agent刷新主动检查到服务器的请求时，都会进行自动注册尝试。请求的延迟在agent的 `RefreshActiveChecks` 参数中指定。第一个请求在agent重新启动后立即发送。

例1

使用主机元数据来区分Linux和Windows主机。

假设你希望主机由Zabbix server自动注册。首先你的网络上有active Zabbix agents（请参阅上面的“配置”部分），其次你的网络上有Windows主机和Linux主机，再次Zabbix页面可以使用“Template OS Linux”和“Template OS Windows”模板，然后在主机注册时，你才能将Linux / Windows模板应用于正在注册的主机。默认情况下，只有主机名在自动注册时才会发送到服务器，这可能还不够。为了确保将正确的模板应用于主机，您应该使用主机元数据。

Agent配置

首先要做的是配置agents。添加下一行agent配置文件：

1. `HostMetadataItem=system.uname`

这样才能确保主机元数据将包含“Linux”或“Windows”，具体取决于运行agent的主机。主机元数据示例如下：

1. `Linux: Linux server3 3.2.0-4-686-pae #1 SMP Debian 3.2.41-2 i686 GNU/Linux`
2. `Windows: Windows WIN-0PXGGSTYNHO 6.0.6001 Windows Server 2008 Service Pack 1 Intel IA-32`

对配置文件进行任何更改后，请重新启动agent。

前端配置

现在你需要在前端创建2个动作。第一个动作：

- 名称：Linux主机自动注册
- 条件：主机元数据似 `Linux`
- 操作：链接到模板：Template OS Linux

在这种情况下，您可以跳过“添加主机”操作。链接到模板需要首先添加主机，服务器会自动执行

第二个动作：

- 名称：Windows主机自动注册

- 主机元数据似 Windows
- 操作：链接到模板：Template OS Windows

例2

使用主机元数据我们可以用于区分各个主机。

Agent配置

将下一行添加到代理配置文件：

```
1. HostMetadata=Linux      21df83bf21bf0be663090bb8d4128558ab9b95fba66a6dbf834f8b91ae5e08ae
```

其中“Linux”是一个平台，其余的字符串是一些难以猜测的秘密文本。

对配置文件进行任何更改后，请勿忘记重新启动代理。

前端配置

在前端创建一个动作，使用上面提到的复杂密码来禁止不需要的主机：

- 名称：自动注册动作Linux
- 条件：
 - 计算类型：AND
 - 条件 (A)：主机元数据 似 Linux
 - 条件 (B)：主机元数据 似
21df83bf21bf0be663090bb8d4128558ab9b95fba66a6dbf834f8b91ae5e08ae
- 操作
 - 向用户发送消息：通过Admin发送所有媒体
 - 添加到主机组：Linux servers
 - 链接到模板：Template OS Linux

请注意，这种方法本身并没有提供强大的保护，因为数据是以纯文本形式传输的。

3 自动发现 (LLD)

概述

自动发现 (LLD) 提供了一种在计算机上为不同实体自动创建监控项、触发器和图形的方法。例如，Zabbix可以在你的机器上自动开始监控文件系统或网络接口，而无需为每个文件系统或网络接口手动创建监控项。此外，可以配置 Zabbix 根据定期执行发现后的得到实际结果，来移除不需要的监控项。

在Zabbix中，支持六种类型的发现项目：

- 系统文件的发现；
- 网络接口的发现；
- CPU和CPU内核的发现
- SNMP OID的发现
- 使用ODBC SQL查询的发现
- Windows服务的发现

用户可以自己定义发现类型，只要它们遵循特定的JSON协议。

发现过程的一般架构如下。

首先，用户在“配置”→“模板”→“发现”列中创建一个发现规则。发现规则包括（1）发现必要实体（例如，文件系统或网络接口）的项目和（2）应该根据该项目的值创建的监控项、触发器和图形的原型

发现必要实体的项目就像其他地方所看到的常规项目：服务器向该项目的值询问Zabbix agent（或者该项目的任何类型的设置），agent以文本值进行响应。区别在于agent响应的值应该包含特定JSON格式的发现实体的列表。这种格式的自定义检查者发现的细节才是最重要的，因为返回值必须包含宏→值对。例如，项目“net.if.discovery”可能会返回两对键值：“{#IFNAME}”→“lo”和“{#IFNAME}”→“eth0”。

Zabbix agent版本2.0支持自动发现项目“vfs.fs.discovery”和“net.if.discovery”。从Zabbix agent版本2.4起支持发现项目“system.cpu.discovery”。从Zabbix server和proxy版本2.0起支持发现SNMP OID。从Zabbix server和proxy版本3.0起支持使用ODBC SQL查询的发现。

在使用IBM DB2数据库运行的Zabbix proxy上，自动发现规则的返回值限制为2048字节。此限制不适用于Zabbix server，因为返回值不会被存储在数据库中。

这些宏用于名称，键值和其他原型字段中，然后用接收到的值为每个发现的实体创建实际的监控项、触发器、图形甚至主机。请参阅使用LLD宏的[选项](#)的完整列表。

当服务器接收到发现项目的值时，它会查看宏→值对，每对都根据原型生成实际监控项、触发器和图形。在上面的“net.if.discovery”示例中，服务器将生成环路接口“lo”的一组监控项、触发器和图表，另一组用于界面“eth0”。

以下部分将详细说明上述过程，并作为一个指导上述类型的所有发现。最后一节描述了发现项目的JSON格式，并给出了文件系统发现实现的Perl脚本的示例。

3.1 文件系统的发现

要配置文件系统的发现，请执行以下操作：

- 转到：配置 → 模板
- 在一个合适的模板的行点击发现

The screenshot shows the 'Templates' page in Zabbix. At the top, there is a header with tabs: Applications, Items, Triggers, Graphs, Screens, and Discovery. Below the header, a table lists templates. The first row is 'Template OS Linux' with counts: Applications 10, Items 32, Triggers 15, Graphs 5, Screens 1, and Discovery 2.

- 单击屏幕右上角的创建发现规则
- 填写以下详细信息。

发现规则选项卡包含常规发现规则属性：

The screenshot shows the 'Discovery rule' configuration form. The 'Discovery rule' tab is selected. The form fields include:

- Name: Mounted filesystem discovery
- Type: Zabbix agent
- Key: vfs.fs.discovery
- Update interval: 1h
- Custom intervals: A table with one row showing Type: Flexible, Interval: 50s, and Period: 1-7,00:00-24:00. An 'Add' button is available for more rows.
- Keep lost resources period: 30d
- Description: Discovery of file systems of different types as defined in global regular expression "File systems for discovery".
- Enabled: checked

At the bottom are 'Add' and 'Cancel' buttons.

参数	描述
名称	发现规则名称。

类型	执行发现的检查类型；可以是 Zabbix agent或Zabbix agent (主动) 文件系统发现。
键值	许多平台上的Zabbix agent程序内置了“vfs.fs.discovery”键值的项目（有关详细信息，请参阅 支持的项目键列表 ），并将返回一个JSON，其中包含计算机上存在的文件系统列表及其类型。
数据更新间隔(秒)	此字段设置Zabbix执行发现的频率。一开始，当你只是设置文件系统发现时，您可能希望将其设置为段间隔时间，但一旦发现它可以将其设置为30分钟或更长时间，因为文件系统通常不会更改。注意：如果设置为“0”，则不会轮询该项。但是，如果灵活间隔也存在非零值，则在灵活间隔持续时间内将轮询该项。
自定义时间间隔	您可以创建用于检查项目的自定义规则：灵活 - 创建更新间隔（不同频次的间隔）的调度 - 创建自定义轮询调度。有关详细信息，请参阅 自定义时间间隔 。从Zabbix 3.0.0起支持调度
保留失去的资源期间(天)	该字段允许你设置发现的实体将被发现状态变为“不再支持”（最多3650天）后将被保留（不会被删除）的天数。注意：如果设置为“0”，将立即删除实体。不建议使用“0”，因为错误地编辑过滤器可能会在实体中删除所有的历史数据。
描述	输入说明文字。
已启用	如果选中，该规则将被执行。

过滤器选项卡包含发现规则过滤器定义：

Discovery rule Filters

Type of calculation: And/Or A or (B and C) ...

Filters	LabelMacro	Regular expression
A	[#FSTYPE]	matches @File systems for discovery
B	[#MACRO]	matches regular expression
Add		

Add Cancel

参数	描述
计算方式	计算过滤器的可用选项如下：与 - 所有过滤器满足；或 - 只需一个过滤器满足；与/或 - 不同的宏名称用与，相同的宏名称用或；自定义表达式 - 提供定义自定义计算的过滤器的。该公式必须包括列表中的所有过滤器。限于255个符号
过滤器	<p>过滤器可用于仅为特定文件系统生成实际监控项，触发器和图形。它支持POSIX扩展正则表达式。例如，如果你只对C :, D :, 和E: 文件系统感兴趣，则可以将{#FSNAME}放入“宏”和“^C ^D ^E”正则表达式到“正则表达式”文本字段。也可以使用{#FSTYPE}宏（例如“^ext ^reiserfs”）的文件系统类型以及使用{#FSDEVICETYPE}宏（例如“fixed”）的驱动器类型（仅由Windows agent支持）进行过滤。你可以在“正则表达式”字段中输入正则表达式或引用全局正则表达式。为了测试正则表达式，你可以使用“grep -E”，例如：</p> <pre style="margin-left: 20px;">1. for f in ext2 nfs reiserfs smbfs; do echo \$f grep -E '^ext ^reiserfs' echo "SKIP: \$f"; done</pre>

从Zabbix **3.0.0**起支持Windows上的宏{#FSDRIVETYPE}。Zabbix **2.4.0**起支持定义多个过滤器。注意，如果响应中缺少过滤器中的某些宏，则找到的实体将被忽略。.

如果要正确发现不同的文件系统名称，则必须将MySQL中的Zabbix数据库创建为区分大小写。

发现规则历史记录不被保留。

创建规则后，转到该规则的项目，然后点击“创建监控项原型”创建项目原型。请注意在需要文件系统名称时使用宏{#FSNAME}的宏。当发现规则被处理时，该宏将被替换为发现的文件系统。

Item prototype [Preprocessing](#)

Name	Free disk space on \$1 (percentage)		
Type	Zabbix agent		
Key	vfs.fs.size[{#FSNAME},pfree]		
Type of information	Numeric (float)		
Units	%		
Update interval	1m		
Custom intervals	Type	Interval	Period
	Flexible	Scheduling	50s
			1-7,00:00-24:00
	Add		
History storage period	1w		
Trend storage period	365d		
Show value	As is show value mappings		
New application			
Applications	<ul style="list-style-type: none"> -None- CPU Filesystems General Memory Network interfaces OS Performance Processes Security ... 		
New application prototype	Application_{#FSNAME}		
Application prototypes	<ul style="list-style-type: none"> -None- 		
Description			
Create enabled	<input checked="" type="checkbox"/>		

监控项原型特有的属性：

参数	描述
----	----

新应用原型	您可以定义一个新的应用原型。在应用原型中，你可以使用自动发现 (LLD) 宏，在发现后，将用实际值替换创建特定于发现实体的应用。有关更多具体信息，请参阅 应用发现说明 。
应用原型	从现有应用原型中选择。
创建已启用	如果选中，项目将被添加到启用状态。如果未选中，该项目将被添加到已发现的实体，但处于禁用状态。

我们可以为我们感兴趣的每个文件系统度量创建几个项目原型：

Item prototypes

All templates / Template OS Linux Discovery list / Mounted filesystem discovery Item prototypes

<input type="checkbox"/> NAME ▲	KEY	INTERVAL
<input type="checkbox"/> Free disk space on {#FSNAME}	vfs.fs.size[{#FSNAME},free]	1m
<input type="checkbox"/> Free disk space on {#FSNAME} (percentage)	vfs.fs.size[{#FSNAME},pfree]	1m
<input type="checkbox"/> Free inodes on {#FSNAME} (percentage)	vfs.fs.inode[{#FSNAME},pfree]	1m
<input type="checkbox"/> Total disk space on {#FSNAME}	vfs.fs.size[{#FSNAME},total]	1h
<input type="checkbox"/> Used disk space on {#FSNAME}	vfs.fs.size[{#FSNAME},used]	1m

然后，我们以类似的方式创建触发器原型：

Trigger prototype **Dependencies**

Name	Free disk space is less than 20% on volume {#FSNAME}						
Severity	Not classified	Information	Warning	Average	High	<input type="button" value="..."/>	
Expression	{Template OS Linux:vfs.fs.size[{#FSNAME},pfree].last(0)}<20						
Expression constructor							
OK event generation	<input checked="" type="radio"/> Expression	<input type="radio"/> Recovery expression	<input type="radio"/> None				
PROBLEM event generation mode	<input checked="" type="radio"/> Single	<input type="radio"/> Multiple					
OK event closes	<input checked="" type="radio"/> All problems	<input type="radio"/> All problems if tag values match					
Tags	<input type="text" value="tag"/> <input type="text" value="value"/> <input type="button" value="Add"/>						
Allow manual close	<input type="checkbox"/>						
URL	<input type="text"/>						
Description	<input type="text"/>						
Create enabled	<input checked="" type="checkbox"/>						

触发原型特有的属性：

参数		描述
创建启用	如果选中，触发器将被添加到启用状态。如果未选中，触发器将被添加到已发现的实体，但处于禁用状态。	

当从原型创建真实触发器时，对表达式中使用什么常量（在我们的示例中为 '20'）是比较灵活的。了解[具有上下文的用户宏](#)可以实现这种灵活性。

在依赖关系选项卡，也可以定义触发器原型之间的[依赖关系](#)（自Zabbix 3.0起支持）。触发器原型可以依赖于来自相同自动发现 (LLD) 规则另一个触发器原型或常规触发器。触发器原型也可不依赖于不同的LLD规则触发器原型或常规触发器的产生的触发器原型。。主机触发器原型不能依赖于模板的触发器。

Trigger prototypes

All templates / Template OS Linux Discovery list / Mounted filesystem discovery Item prototypes 5

SEVERITY	NAME	EXPRESSION
Warning	Free disk space is less than 20% on volume {#FSNAME}	{Template OS}
Warning	Free inodes is less than 20% on volume {#FSNAME}	{Template OS}

我们也可以创建图形原型：

Graph prototype

Preview

Name	Disk space usage {#FSNAME}						
Width	600						
Height	340						
Graph type	Pie						
Show legend	<input checked="" type="checkbox"/>						
3D view	<input checked="" type="checkbox"/>						
Items	<table border="1"> <thead> <tr> <th>Name</th> <th>Type</th> </tr> </thead> <tbody> <tr> <td>1: Template OS Linux: Total disk space on {#FSNAME}</td> <td>Graph</td> </tr> <tr> <td>2: Template OS Linux: Free disk space on {#FSNAME}</td> <td>Simple</td> </tr> </tbody> </table>	Name	Type	1: Template OS Linux: Total disk space on {#FSNAME}	Graph	2: Template OS Linux: Free disk space on {#FSNAME}	Simple
Name	Type						
1: Template OS Linux: Total disk space on {#FSNAME}	Graph						
2: Template OS Linux: Free disk space on {#FSNAME}	Simple						
Add	Add prototype						

Graph prototypes

All templates / Template OS Linux Discovery list / Mounted filesystem discovery Item prototypes 5

NAME	WIDTH
Disk space usage {#FSNAME}	600

最后，我们创建了一个发现规则，如下图所示。它有五个监控项目原型，两个触发器原型和一个图形原型。

Discovery rules

All templates / Template OS Linux Applications 10 Items 32 Triggers 15 Graphs 5 Screens 1

NAME	ITEMS	TRIGGERS	GRAPHS	H
Mounted filesystem discovery	Item prototypes 5	Trigger prototypes 2	Graph prototypes 1	H

注意：有关配置主机模板，请参阅虚拟机监控中有关[主机模板](#)配置的部分。

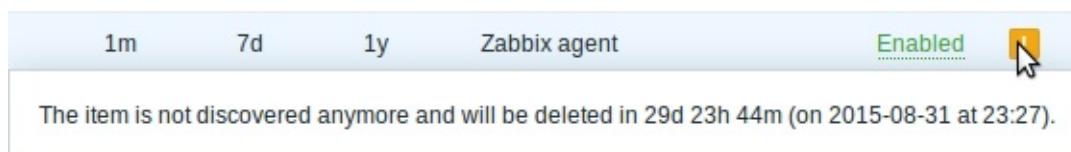
下面的屏幕截图说明了主机配置中发现的监控项，触发器和图形的样子。发现的实体前缀有橙色链接到他们来自的发现规则。

<input type="checkbox"/> Wizard	Name	Triggers	Key ▲
<input type="checkbox"/>	Mounted filesystem discovery: Free inodes on / (percentage)	1	vfs.fs.inod
<input type="checkbox"/>	Mounted filesystem discovery: Free disk space on /		vfs.fs.size
<input type="checkbox"/>	Mounted filesystem discovery: Free disk space on / (percentage)	1	vfs.fs.size
<input type="checkbox"/>	Mounted filesystem discovery: Total disk space on /		vfs.fs.size
<input type="checkbox"/>	Mounted filesystem discovery: Used disk space on /		vfs.fs.size

请注意，如果已经存在具有相同唯一性条件的现有实体，例如具有相同键值或具有相同名称的图形的项目，则不会创建发现的实体。

如果发现的实体（文件系统，接口等）停止发现（或不再通过过滤器），则由自动发现规则（LLD）创建的项目（类似地，触发器和图形）将被自动删除。这时，监控项，触发器和图表将在保留失去的资源期间字段中定义的日期过去后被删除。

当发现的实体变为“不再支持”时，项目列表中将显示生命周期指示符。将鼠标指针移动到其上，并显示一条消息，指示在删除项目之前剩下多少天。。



The item is not discovered anymore and will be deleted in 29d 23h 44m (on 2015-08-31 at 23:27).

如果实体被标记为删除，但未在预期时间被删除（禁用的发现规则或项目主机），则在下次发现规则被处理时，它们将被删除。

标记为删除的其他实体的实体，如果在发现规则级别上更改，则不会更新。例如，如果基于LLD的触发器标记为要删除的项目，则它们将不会更新。

<input type="checkbox"/> Severity	Name ▲	Expression
<input checked="" type="checkbox"/> Warning	Mounted filesystem discovery: Free disk space is less than 20% on volume /	{Zabbix serv
<input checked="" type="checkbox"/> Warning	Mounted filesystem discovery: Free inodes is less than 20% on volume /	{Zabbix serv

3.2 网络接口的发现

网络接口的发现与文件系统的发现完全相同，只是你使用发现规则的键值是“`net.if.discovery`”而不是“`vfs.fs.discovery`”，并使用宏`{#IFNAME}`而不是`{#FSNAME}`。

你可能希望基于“`net.if.discovery`”创建的监控项原型示

例：“`net.if.in[{#IFNAME},bytes]`”，“`net.if.out[{#IFNAME},bytes]`”。

有关过滤器的更多信息，请参阅[上文](#)。

3.3 CPU和CPU内核的发现

CPU和CPU内核的发现以与网络接口发现类似的方式完成相同，除了发现规则的键值是“`system.cpu.discovery`”之外。此发现键返回两个宏 - `{#CPU.NUMBER}`和`{#CPU.STATUS}`分别标识CPU序号和状态。要注意，实际物理处理器，内核和超线程之间不能做出明确的区分。在Linux, UNIX和BSD系统上的`{#CPU.STATUS}`返回处理器的状态，可以是“`online`”还是“`offline`”。在Windows系统上，同一个宏可能表示第三个值 - “`unknown`” - 这表示处理器已被检测到，但尚未收集到任何信息

CPU发现依赖于agent的收集器进程去收集和获取数据。如果agent的测试（`-t`）命令行不起作用，这将返回一个`NOT_SUPPORTED`状态以及附带的消息，表明收集器进程尚未启动。

可以基于CPU发现规则创建的监控项原型包括例如：“`system.cpu.util[{#CPU.NUMBER}, <type>, <mode>]`”或“`system.hw.cpu[{#CPU.NUMBER}, <info>]`”。

3.4 SNMP OID的发现

例如，我们将在交换机上执行SNMP发现。首先，进入“配置”→“模板”。

要编辑模板的发现规则，请单击“自动发现”列中的链接。

然后，按“创建发现规则”，并在下面的屏幕截图中填写表单与详细信息。

与文件系统和网络接口发现不同，项目不一定必须具有“`snmp.discovery`”键值 - SNMP agent的项目类型就足够了。

要发现的OID在SNMP OID字段中以下列格式定义： `discovery[#{MACRO1}, oid1, #{MACRO2}, oid2, ...]`

其中`{#MACRO1}`, `{#MACRO2}` ...是有效的lld宏名称和`oid1`, `oid2...` 是能够为这些宏生成有意义的值的OID。已发现OID索引的内置宏`{#SNMPINDEX}`将应用于发现的实体。发现的实体按`{#SNMPINDEX}`宏值分组。

为了理解清楚，在我们的交换机上执行几个`snmpwalk`:

```

1. $ snmpwalk -v 2c -c public 192.168.1.1 IF-MIB::ifDescr
2. IF-MIB::ifDescr.1 = STRING: WAN
3. IF-MIB::ifDescr.2 = STRING: LAN1
4. IF-MIB::ifDescr.3 = STRING: LAN2
5.
6. $ snmpwalk -v 2c -c public 192.168.1.1 IF-MIB::ifPhysAddress
7. IF-MIB::ifPhysAddress.1 = STRING: 8:0:27:90:7a:75
8. IF-MIB::ifPhysAddress.2 = STRING: 8:0:27:90:7a:76
9. IF-MIB::ifPhysAddress.3 = STRING: 8:0:27:2b:af:9e

```

并将SNMP OID设置为： `discovery[#{IFDESCR}, ifDescr, #{IFPHYSADDRESS}, ifPhysAddress]`

现在，这个规则会发现设置为`{#IFDESCR}`宏实体WAN, LAN1和LAN2, `{#IFPHYSADDRESS}`宏设置为`8: 0: 27: 90: 7A: 75`, `8: 0: 27: 90: 7A: 76`和`8: 0: 27: 2B: AF: 9E`, `{#SNMPINDEX}`宏设定为所发现的OID索引1, 2和3:

```

1. {
2.   "data": [
3.     {
4.       "{#SNMPINDEX}": "1",
5.       "{#IFDESCR}": "WAN",
6.       "{#IFPHYSADDRESS}": "8:0:27:90:7a:75"
7.     },
8.     {
9.       "{#SNMPINDEX}": "2",
10.      "{#IFDESCR}": "LAN1",
11.      "{#IFPHYSADDRESS}": "8:0:27:90:7a:76"
12.    },
13.    {
14.      "{#SNMPINDEX}": "3",
15.      "{#IFDESCR}": "LAN2",
16.      "{#IFPHYSADDRESS}": "8:0:27:2b:af:9e"
17.    }
18.  ]
19. }

```

如果一个实体没有指定的OID，则该实体将忽略相应的宏。例如，如果我们有以下数据：

```

1. ifDescr.1 "Interface #1"
2. ifDescr.2 "Interface #2"
3. ifDescr.4 "Interface #4"

```

```
4.  
5. ifAlias.1 "eth0"  
6. ifAlias.2 "eth1"  
7. ifAlias.3 "eth2"  
8. ifAlias.5 "eth4"
```

那么在这种情况下，SNMP发现 `discovery[{"#IFDESCR"}, ifDescr, {"#IFALIAS"}, ifAlias]` 将返回以下结构：

```
1. {  
2.     "data": [  
3.         {  
4.             "#SNMPINDEX": 1,  
5.             "#IFDESCR": "Interface #1",  
6.             "#IFALIAS": "eth0"  
7.         },  
8.         {  
9.             "#SNMPINDEX": 2,  
10.            "#IFDESCR": "Interface #2",  
11.            "#IFALIAS": "eth1"  
12.        },  
13.        {  
14.            "#SNMPINDEX": 3,  
15.            "#IFALIAS": "eth2"  
16.        },  
17.        {  
18.            "#SNMPINDEX": 4,  
19.            "#IFDESCR": "Interface #4"  
20.        },  
21.        {  
22.            "#SNMPINDEX": 5,  
23.            "#IFALIAS": "eth4"  
24.        }  
25.    ]  
26. }
```

Discovery rule [Filters](#)

Name	Network interfaces		
Type	SNMPv2 agent		
Key	ifDescr		
SNMP OID	discovery[[#IFDESCR],IF-MIB::ifDescr]		
SNMP community	{\$SNMP_COMMUNITY}		
Port			
Update interval	1h		
Custom intervals	Type	Interval	Period
	<input checked="" type="radio"/> Flexible	<input type="radio"/> Scheduling	50s
			1-7,00:00-24:00
	Add		
Keep lost resources period	30d		
Description	<p>You may also consider using IF-MIB::ifType or IF-MIB::ifAlias for discovery depending on your filtering needs.</p> <p>{\$SNMP_COMMUNITY} is a global macro.</p>		
Enabled	<input checked="" type="checkbox"/>		
	Add	Cancel	

以下屏幕截图显示了我们如何在监控项原型中使用这些宏：

Item prototype Preprocessing

Name	Incoming traffic on interface \$1		
Type	SNMPv2 agent		
Key	ifInOctets[{#IFDESCR}]		
SNMP OID	IF-MIB::ifInOctets.{#SNMPINDEX}		
SNMP community	{\${SNMP_COMMUNITY}}		
Port			
Type of information	Numeric (unsigned)		
Units	bps		
Update interval	1m		
Custom intervals	Type	Interval	Period
	Flexible	Scheduling	50s 1-7,00:00-24:00
	Add		
History storage period	1w		
Trend storage period	365d		
Show value	As is	show value mappings	
New application			

再次，根据需要创建尽可能多的监控项原型：

Item prototypes

[All templates / Template SNMP Interfaces](#) [Discovery list / Network interfaces](#) [Item prototypes 8](#)

<input type="checkbox"/> NAME ▲	KEY	INTERVAL	HI
<input type="checkbox"/> Admin status of interface {#IFDESCR}	ifAdminStatus[{#IFDESCR}]	1m	7d
<input type="checkbox"/> Alias of interface {#IFDESCR}	ifAlias[{#IFDESCR}]	1h	7d
<input type="checkbox"/> Description of interface {#IFDESCR}	ifDescr[{#IFDESCR}]	1h	7d
<input type="checkbox"/> Inbound errors on interface {#IFDESCR}	ifInErrors[{#IFDESCR}]	1m	7d
<input type="checkbox"/> Incoming traffic on interface {#IFDESCR}	ifInOctets[{#IFDESCR}]	1m	7d
<input type="checkbox"/> Operational status of interface {#IFDESCR}	ifOperStatus[{#IFDESCR}]	1m	7d
<input type="checkbox"/> Outbound errors on interface {#IFDESCR}	ifOutErrors[{#IFDESCR}]	1m	7d
<input type="checkbox"/> Outgoing traffic on interface {#IFDESCR}	ifOutOctets[{#IFDESCR}]	1m	7d

以及触发原型：

Trigger prototype Dependencies

Name	Operational status was changed on {HOST.NAME} int						
Severity	Not classified	Information	Warning	Average	High	Critical	
Expression	<div style="border: 1px solid #ccc; padding: 5px;"> {Template SNMP Interfaces:ifOperStatus[#{#IFDESCR}].diff(0)}=1 </div> <input type="button" value="Add"/>						
Expression constructor							
OK event generation	Expression	Recovery expression	None				
PROBLEM event generation mode	Single	Multiple					
OK event closes	All problems	All problems if tag values match					
Tags	<input type="text" value="tag"/> <input type="text" value="value"/> <input type="button" value="Remove"/> <input type="button" value="Add"/>						
Allow manual close	<input type="checkbox"/>						
URL	<input type="text"/>						
Description	<div style="border: 1px solid #ccc; height: 100px; margin-top: 10px;"></div>						
Create enabled	<input checked="" type="checkbox"/>						

Trigger prototypes

All templates / Template SNMP Interfaces Discovery list / Network interfaces Item prototypes 8

<input type="checkbox"/> SEVERITY	NAME	▲	EXPR
<input type="checkbox"/> Information	Operational status was changed on {HOST.NAME} interface {#IFDESCR}	{Template	

和图形原型：

Graph prototype Preview

Name	Traffic on interface {#IFDESCR}		
Width	900		
Height	200		
Graph type	Normal ▾		
Show legend	<input checked="" type="checkbox"/>		
Show working time	<input checked="" type="checkbox"/>		
Show triggers	<input checked="" type="checkbox"/>		
Percentile line (left)	<input type="checkbox"/>		
Percentile line (right)	<input type="checkbox"/>		
Y axis MIN value	Calculated ▾		
Y axis MAX value	Calculated ▾		
Items	Name	Function	Draw style
	1: Template SNMP Interfaces: Incoming traffic on interface {#IFDESCR}	avg ▾	Gradient
	2: Template SNMP Interfaces: Outgoing traffic on interface {#IFDESCR}	avg ▾	Gradient
Add Add prototype			

Graph prototypes

All templates / Template SNMP Interfaces Discovery list / Network interfaces Item prototypes 8 1

<input type="checkbox"/> NAME ▾	WIDTH
Traffic on interface {#SNMPVALUE}	900

我们的发现规则摘要：

Discovery rules

All templates / Template SNMP Interfaces Applications 1 Items 1 Triggers Graphs Screens

<input type="checkbox"/> NAME ▾	ITEMS	TRIGGERS	GRAPHS	HISTORY
Network interfaces	Item prototypes 8	Trigger prototypes 1	Graph prototypes 1	History 0

当服务器运行时，它将根据SNMP发现规则返回的值创建实际监控项，触发器和图形。在主机配置中，它们的前缀是橙色链接到它们来自的发现规则。

Items

All hosts / Switch1 Enabled ZBX SNMP JMX IPMI Applications 1 Items 241 Triggers 30 Groups Filter ▾

<input type="checkbox"/> Wizard	Name	Triggers	Key ▲
<input type="checkbox"/>	Network interfaces: Admin status of interface 1		ifAdminStatus[1]
<input type="checkbox"/>	Network interfaces: Admin status of interface 2		ifAdminStatus[2]
<input type="checkbox"/>	Network interfaces: Admin status of interface 3		ifAdminStatus[3]
<input type="checkbox"/>	Network interfaces: Admin status of interface 4		ifAdminStatus[4]

Triggers

All hosts / Switch1 Enabled ZBX SNMP JMX IPMI Applications 1 Items 241 Triggers 30 Groups Filter ▾

<input type="checkbox"/> Severity	Name ▲	Exp
<input checked="" type="checkbox"/> Information	Network interfaces: Operational status was changed on {HOST.NAME} interface 1	{proc}
<input checked="" type="checkbox"/> Information	Network interfaces: Operational status was changed on {HOST.NAME} interface 2	{proc}
<input checked="" type="checkbox"/> Information	Network interfaces: Operational status was changed on {HOST.NAME} interface 3	{proc}
<input checked="" type="checkbox"/> Information	Network interfaces: Operational status was changed on {HOST.NAME} interface 4	{proc}

Graphs

All hosts / Switch1 Enabled ZBX SNMP JMX IPMI Applications 1 Items 241 Triggers 30 Groups Group all Filter ▾

<input type="checkbox"/> Name ▲	
<input type="checkbox"/>	Network interfaces: Traffic on interface 1
<input type="checkbox"/>	Network interfaces: Traffic on interface 2
<input type="checkbox"/>	Network interfaces: Traffic on interface 3
<input type="checkbox"/>	Network interfaces: Traffic on interface 4

3.5 ODBC SQL查询的发现

这种类型的发现使用SQL查询完成，其结果自动转换为适合于自动发现 (LLD) 的JSON对象。使用“数据库监控”类型的项目执行SQL查询。因此，ODBC监控页面上的说明都适用于“数据库监控”发现规则，唯一的区别是应该使用“db.odbc.discovery[<description>,<dsn>]”键代替“db.odbc.select[<description>,<dsn>]”。

举例来说明SQL查询如何转换为JSON，我们可以通过在Zabbix数据库上执行ODBC查询来执行Zabbix proxies 自动发现 (LLD)。这对于自动创建“zabbix[proxy,<name>,lastaccess]” 内部项目 来监视哪些proxies是存活

的很有用。

让我们从发现规则配置开始：

Type	Interval	Period
Flexible	50s	1-7,00:00-24:00

这里，对Zabbix数据库的执行查询用于选择所有Zabbix proxies以及它们正在监视的主机数量。例如，可以使用主机数量来过滤掉空 proxies：

```

1. mysql> SELECT h1.host, COUNT(h2.host) AS count FROM hosts h1 LEFT JOIN hosts h2 ON h1.hostid = h2.proxy_hostid
      WHERE h1.status IN (5, 6) GROUP BY h1.host;
2. +-----+-----+
3. | host   | count  |
4. +-----+-----+
5. | Japan 1 |      5 |
6. | Japan 2 |     12 |
7. | Latvia  |      3 |
8. +-----+-----+
9. 3 rows in set (0.01 sec)

```

通过“db.odbc.discovery []”项目的内部工作，此查询的结果将自动转换为以下 JSON：

```

1. {
2.     "data": [
3.         {
4.             "{#HOST}": "Japan 1",
5.             "{#COUNT}": "5"
6.         },
7.         {
8.             "{#HOST}": "Japan 2",
9.             "{#COUNT}": "12"
10.        },
11.        {
12.            "{#HOST}": "Latvia",
13.            "{#COUNT}": "3"
14.        }
15.    ]
16. }
```

It can be seen that column names become macro names and selected rows become the values of these macros.

可以看出，列名称成为宏名称，选定的行将成为这些宏的值。

如果将列名称变换为宏名称不明显，建议在上述示例中使用像“COUNT(h2.host) AS count”这样的列别名。

如果列名称无法转换为有效的宏名称，则不支持发现规则，错误消息将详细列出违规列号。如果需要其他帮助，获取的列名称在Zabbix服务器日志文件中的DebugLevel = 4下提供：

```

1. $ grep db.odbc.discovery /tmp/zabbix_server.log
2. ...
3. 23876:20150114:153410.856 In db_odbc_discovery() query:'SELECT h1.host, COUNT(h2.host) FROM hosts h1 LEFT
   JOIN hosts h2 ON h1.hostid = h2.proxy_hostid WHERE h1.status IN (5, 6) GROUP BY h1.host;'
4. 23876:20150114:153410.860 db_odbc_discovery() column[1]:'host'
5. 23876:20150114:153410.860 db_odbc_discovery() column[2]:'COUNT(h2.host)'
6. 23876:20150114:153410.860 End of db_odbc_discovery():NOTSUPPORTED
7. 23876:20150114:153410.860 Item [Zabbix server:db.odbc.discovery[proxies,{\$DSN}]] error: Cannot convert column
   #2 name to macro.
```

现在我们了解SQL查询如何转换为JSON对象，我们可以在项目原型中使用{#HOST}宏：

Item prototype Preprocessing

Name	Last acces time of proxy (#HOST)						
Type	Zabbix internal						
Key	zabbix[proxy,#HOST,lastaccess]						
Type of information	Numeric (unsigned)						
Units	unixtime						
Update interval	60s						
Custom intervals	<table border="1"> <thead> <tr> <th>Type</th> <th>Interval</th> <th>Period</th> </tr> </thead> <tbody> <tr> <td>Flexible</td> <td>50s</td> <td>1-7,00:00-24:00</td> </tr> </tbody> </table> Add	Type	Interval	Period	Flexible	50s	1-7,00:00-24:00
Type	Interval	Period					
Flexible	50s	1-7,00:00-24:00					
History storage period	90d						
Trend storage period	365d						
Show value	As is show value mappings						

执行发现后，将为每个proxy创建一个监控项：

Items

All hosts / Zabbix server 1	Enabled	ZBX	SNMP	JMX	IPMI	Applications 12	Items 70	Triggers	Filter ▾
<input type="checkbox"/> Wizard	Name							Triggers	Key ▲
<input type="checkbox"/>	Proxy discovery: Last access time of proxy Japan1							zabbix[proxy,Japan1,lastaccess]	
<input type="checkbox"/>	Proxy discovery: Last access time of proxy Japan2							zabbix[proxy,Japan2,lastaccess]	
<input type="checkbox"/>	Proxy discovery: Last access time of proxy Latvia							zabbix[proxy,Latvia,lastaccess]	

3.6 Windows服务的发现

Windows服务发现的方式与文件系统的发现相同。在发现规则中使用的关键是“service.discovery”，并且支持以下宏用于过滤器和监控项/触发器/图形原型：

1. {#SERVICE.NAME}
2. {#SERVICE.DISPLAYNAME}
3. {#SERVICE.DESCRIPTION}
4. {#SERVICE.STATE}
5. {#SERVICE.STATENAME}
6. {#SERVICE.PATH}
7. {#SERVICE.USER}
8. {#SERVICE.STARTUP}

```
9. {#SERVICE.STARTUPNAME}
```

基于Windows服务发现，你可以创建一个监控项原型，如“service.info[#{SERVICE.NAME},<param>]”），其中param接受以下值：state, displayname, path, user, startup 或 description。例如，要获取服务的显示名称，您应该使用“service.info[#{SERVICE.NAME},displayname]”项目。如果没有指定param值（“service.info[#{SERVICE.NAME}]”），则使用默认参param态。

{ # SERVICE.STATE}和{ # SERVICE.STATENAME}宏返回相同的内容，但{ # SERVICE.STATE}返回数值（0-7），而{ # SERVICE.STATENAME}返回文字（running, paused, start pending, pause pending, continue pending, stop pending, stopped or unknown）。{ # SERVICE.STARTUP}和{ # SERVICE.STARTUPNAME}也是如此，其中一个返回数字值（0-4），而另一个文本（automatic, automatic delayed, manual, disabled, unknown）。

3.7 为同一项目设置多个LLD规则

从Zabbix agent版本3.2，可以使用zabbix_agentd.conf文件中的“Alias”参数来更改自动发现项目键值，以便为同一项目配置多个LLD规则。

3.8 创建自定义LLD规则

也可以创建完全自定义的LLD规则，发现任何类型的实体 - 例如数据库服务器上的数据库。

为此，应该创建一个返回JSON的自定义项目，指定找到的对象以及可选的一些属性。每个实体的宏数量不受限制 - 而内置的发现规则返回一个或两个宏（例如，两个用于文件系统发现）。

下面举例说明JSON格式。假设我们运行一个旧的Zabbix 1.8 agent（不支持“vfs.fs.discovery”），但是我们仍然需要发现文件系统。这是一个用于Linux的简单Perl脚本，用于发现挂载的文件系统并输出JSON，其中包含文件系统名称和类型。使用它的一种方式是具有键值“vfs.fs.discovery_perl”的参数：

```
1. #!/usr/bin/perl
2.
3. $first = 1;
4.
5. print "{\n";
6. print "\t\"data\":[\n\n";
7.
8. for (`cat /proc/mounts`)
9. {
10.     ($fsname, $fstype) = m/\S+ (\S+) (\S+)/;
11.
12.     print "\t,\n" if not $first;
13.     $first = 0;
14.
15.     print "\t{\n";
16.     print "\t\t\"#FSNAME\":\"$fsname\",";
17.     print "\t\t\"#FSTYPE\":\"$fstype\"\n";
18.     print "\t}\n";
19. }
20.
21. print "\n\t]\n";
```

```
22. print "}\n";
```

对于LLD宏名允许的符号为 **0-9 , A-Z , _ , .**名称中不支持小写字母。

其输出的示例（为了清楚起见重新格式化）如下所示。用于自定义发现检查的JSON必须遵循相同的格式。

```
1. {
2.   "data": [
3.
4.     { "#FSNAME":"/", "#FSTYPE":"rootfs" },
5.     { "#FSNAME":"/sys", "#FSTYPE":"sysfs" },
6.     { "#FSNAME":"/proc", "#FSTYPE":"proc" },
7.     { "#FSNAME":"/dev", "#FSTYPE":"devtmpfs" },
8.     { "#FSNAME":"/dev/pts", "#FSTYPE":"devpts" },
9.     { "#FSNAME":"/lib/init/rw", "#FSTYPE":"tmpfs" },
10.    { "#FSNAME":"/dev/shm", "#FSTYPE":"tmpfs" },
11.    { "#FSNAME":"/home", "#FSTYPE":"ext3" },
12.    { "#FSNAME":"/tmp", "#FSTYPE":"ext3" },
13.    { "#FSNAME":"/usr", "#FSTYPE":"ext3" },
14.    { "#FSNAME":"/var", "#FSTYPE":"ext3" },
15.    { "#FSNAME":"/sys/fs/fuse/connections", "#FSTYPE":"fusectl" }
16.
17.  ]
18. }
```

然后，在发现规则的“过滤器”字段中，我们可以将“`{#FSTYPE}`”指定为宏，将 “`rootfs|ext3`”指定为正则表达式。

你不一定使用具有自定义LLD规则的宏名称`FSNAME/FSTYPE`，你可以随意使用任何名称。

3.9 在用户宏上下文中使用LLD宏

具有上下文的用户宏可用于在触发器表达式中实现更灵活的阈值。可以在用户宏级别上定义不同的阈值，然后根据发现的上下文使用触发器常量。当宏中使用的自动发现 (LLD) 宏被解析为真实值时，会出现在发现的上下文。

为了说明我们可以从例子中使用上述数据和假设下面的文件系统将被发现：`/`，`/home`，`/tmp`，`/usr`，`/var`。

我们可以为主机定义一个可用磁盘空间触发器原型，其中阈值由具有上下文的用户宏表示：

```
{host:vfs.fs.size[{#FSNAME}, pfree].last()}<${LOW_SPACE_LIMIT:"{#FSNAME}"}
```

然后添加用户宏：

- `LOW_SPACE_LIMIT` **10**
- `LOW_SPACE_LIMIT:/home` **20**
- `LOW_SPACE_LIMIT:/tmp` **50**

现在，一旦文件系统被发现，事件将被告知是否产生`/`，`/usr`以及`/var`文件系统具有小于**10%** 的可用磁盘空间，该`/home`文件系统-小于 **20%** 的可用磁盘空间或，`/tmp`的文件系统-小于**50%**的可用磁盘空间。

触发功能参数中的用户宏上下文内不支持LLD宏。

关于自动发现(LLD)的注意事项

应用发现

应用原型支持LLD宏。

一个应用原型可以被相同发现规则的几个监控项原型使用。

如果创建的应用原型未被任何项目原型使用，它将自动从“应用原型”列表中删除。

像其他发现的实体一样，应用遵循发现规则（“保留失去的资源期间”设置）中定义的生命周期 - 在指定的天数未被发现后，将被删除。

如果一个应用不再被发现，所有发现的项目将自动从中删除，即使应用本身由于“失去的资源期间”设置而尚未被删除。

由一个发现规则定义的应用原型不能发现相同的应用。在这种情况下，只有第一个原型发现将成功，其余的将报告相应的LLD错误。只有在不同发现规则中定义的应用原型才能够发现相同的应用。

16. 分布式监控

概述

Zabbix通过Zabbix [proxies](#)为IT基础设施提供有效和可用的分布式监控

代理(proxies)可用于代替Zabbix server本地收集数据，然后将数据报告给服务器。

Proxy 特征

当选择使用/不使用proxy时，必须考虑几个注意事项。

	Proxy
轻量级 (<i>Lightweight</i>)	Yes
图形界面 (<i>GUI</i>)	No
独立工作 (<i>Works independently</i>)	Yes
易于维护 (<i>Easy maintenance</i>)	Yes
自动生成数据库 (<i>Automatic DB creation</i>) ¹	Yes
本地管理 (<i>Local administration</i>)	No
准备嵌入式硬件 (<i>Ready for embedded hardware</i>)	Yes
单向TCP连接 (<i>One way TCP connections</i>)	Yes
集中配置 (<i>Centralised configuration</i>)	Yes
生成通知 (<i>Generates notifications</i>)	No

[1] 自动数据库创建功能仅适用于SQLite。其他数据库需要[手动设置](#)。

1 代理

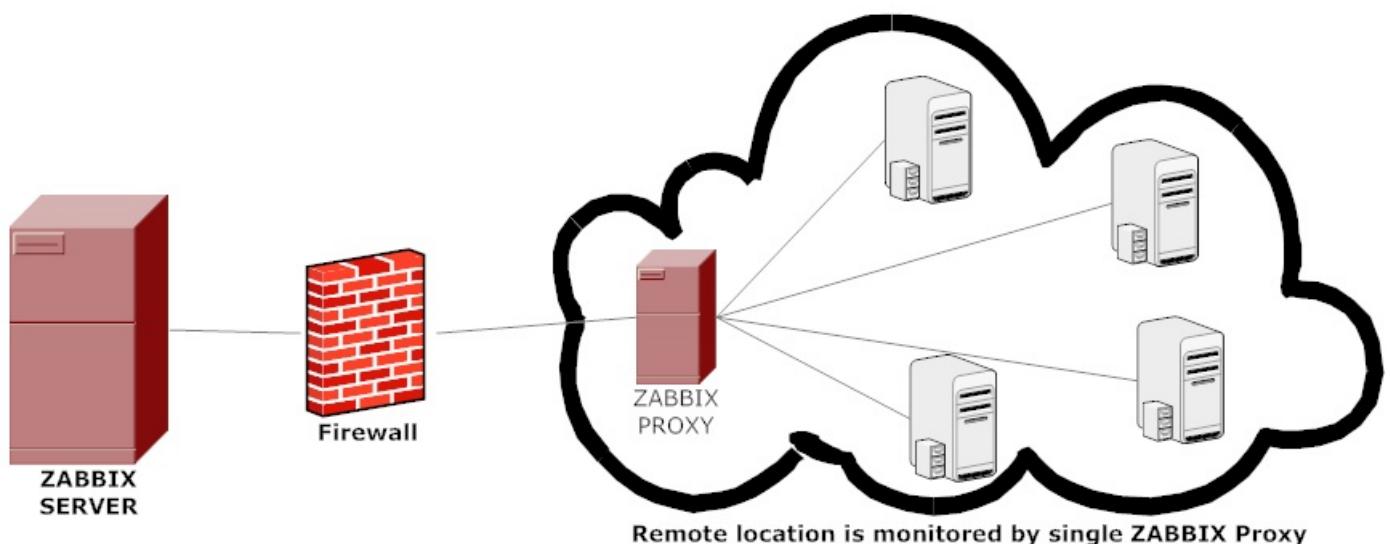
概述

zabbix proxy 可以代替 zabbix server 收集性能和可用性数据, 然后把数据汇报给 zabbix server, 并且在一定程度上分担了zabbix server 的压力.

此外, 当所有agents和proxies报告给一个Zabbix server并且所有数据都集中收集时, 使用proxy是实现集中式和分布式监控的最简单方法。

zabbix proxy 使用场景:

- 监控远程区域设备
- 监控本地网络不稳定区域
- 当 zabbix 监控上千设备时, 使用它来减轻 server 的压力
- 简化分布式监控的维护



zabbix proxy 仅仅需要一条 tcp 连接到 zabbix server, 所以防火墙上仅仅需要加上一条规则即可。

zabbix proxy 数据库必须和 server 分开, 否则数据会被破坏。

proxy 收集到数据之后, 首先将数据缓存在本地, 然后在一定得时间之后传递给 zabbix server, 这样就不会因为服务器的任何临时通信问题而丢失数据。这个时间由 [proxy配置文件](#) 中参数 `ProxyLocalBuffer` 和 `ProxyOfflineBuffer` 决定。

注意从Zabbix server数据库直接更新最新配置的proxy可能会比Zabbix server新, 而Zabbix server的配置由于 [CacheUpdateFrequency](#) 的原因而无法快速更新。因此, proxy收集发送Zabbix server数据可能会被忽略。

zabbix proxy 是一个数据收集器, 它不计算触发器、不处理事件、不发送报警。有关proxy功能的概述, 如下表:

--	--

功能	proxy支持(yes/no)	
项目 (Items)		
Zabbix agent checks (active)	Yes ¹	
Simple checks	Yes	
Trapper items	Yes	
SNMP checks	Yes	
SNMP traps	Yes	
IPMI checks	Yes	
JMX checks	Yes	
日志文件监控 (Log file monitoring)	Yes	
内部检查 (Internal checks)	Yes	
SSH checks	Yes	
Telnet checks	Yes	
外部检查 (External checks)	Yes	
内置web监控 (Built-in web monitoring)	Yes	
网络发现(Network discovery)	Yes	
自动发现 (Low-level discovery)	Yes	
触发器计算 (Calculating triggers)	No	
处理事件 (Processing events)	No	
发送报警 (Sending alerts)	No	
远程命令 (Remote commands)	No	

[1] 使用 agent active 模式,一定要记住在 agent 的配置文件参数 **ServerActive** 加上 proxy 的 IP 地址。

配置

一旦[安装并配置](#)了一个proxy，我们便可以在zabbix管理站点配置它了。

添加 proxies

要在Zabbix前端配置代理：

- 转到：管理→agent代理程序
- 单击创建代理



参数	描述
agent代理程	proxy名称。它必须与proxy配置文件中的Hostname参数中的名称相同。

序名称	proxy名称。它必须与proxy配置文件中的Hostname参数中的名称相同。
代理模式 (Proxy mode)	选择proxy模式 Active - the proxy will connect to the Zabbix server and request configuration data Passive - Zabbix server connects to the proxy Note that without encrypted communications (sensitive) proxy configuration data may become available to parties having access to the Zabbix server trapper port when using an active proxy. This is possible because anyone may pretend to be an active proxy and request configuration data if authentication does not take place.
系统代理程序模式	选择proxy模式主动式 - proxy将连接到Zabbix server并请求配置数据被动式 - Zabbix server连接到代理proxy注意, 当使用active proxy, 未加密通信(敏感) proxy配置数据可用于访问Zabbix server trapper端口。如果不进行身份验证, 任何人都可以伪装成active proxy并请求配置数据。
Hosts	Add hosts to be monitored by the proxy. Hosts already monitored by another proxy are greyed out in the <i>Other hosts</i> selection.
主机	添加要由proxy监视的主机。被另一个proxy监视的主机在其他主机选项中显示为灰色。
Description	Enter the proxy description.
描述	输入proxy描述。

该 加密选项卡允许你与proxy的加密连接。

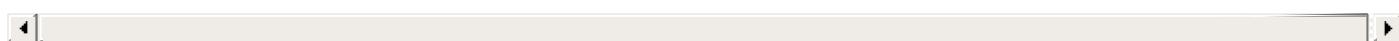
参数	描述
连接代理	服务器如何连接到被动代理: 非加密(默认), 共享秘钥(PSK)或证书。
从代理连接	从active proxy中选择允许的连接类型。可以同时选择几种连接类型(用于测试和切换到其他连接类型)。默认为“无加密”。
发行者	允许颁发证书。证书首先通过CA(认证机构)验证。如果CA有效, 则由CA签名, 这时可以使用发行者字段来进一步限制允许的CA。该字段是可选的, 如果你的Zabbix安装使用多个CA的证书, 则使用该字段。
主体	允许的证书。证书首先通过CA验证。如果它有效, 由CA签名, 这时主体字段可以用于仅允许一个主体字符串值。如果此字段为空, 则接受CA签名的任何有效证书。
共享密钥一致性	共享密钥身份字符串
共享密钥(PSK)	共享密钥(16进制)。如果Zabbix使用mbed TLS(PolarSSL)库, 最大长度为64位十六进制(32字节PSK); 如果Zabbix使用GnuTLS或OpenSSL库, 最大长度为512位十六进制数(256字节PSK)。示例: 1f87b595725ac58dd977beef14b97461a7c1045b9a1c963065002c5473194952

主机配置

您可以使用由agent代理程序监测字段指定[主机配置](#)窗体中的proxy监视单个主机。



主机[批量更新](#)是指定主机由proxy监视的另一种方式。



17. 加密

概述

Zabbix支持使用传输层安全 (TLS) 协议v.1.2在Zabbix server, Zabbix proxy, Zabbix agent, zabbix_sender和zabbix_get程序之间的加密通信。从Zabbix 3.0开始支持加密，支持基于证书和共享秘钥加密。

加密是可选的，可以针对各个组件进行配置（例如，一些proxies和agents可以配置为与服务器一起使用基于证书的加密，而其他可以使用共享秘钥加密，剩下的可以像以前一样继续使用未加密的通信）。

服务器（代理）可以为不同的主机使用不同的加密配置。

Zabbix守护程序使用一个监听端口进行加密和未加密的传入连接。添加加密不需要在防火墙上打开新端口。

限制

- 私钥以明文形式存储在Zabbix组件启动期间可读的文件中。
- 共享密钥在Zabbix前端输入，并以纯文本形式存储在Zabbix数据库中。
- 内置加密不保护通讯：
 - 在运行Zabbix前端的Web服务器和用户Web浏览器之间
 - 在Zabbix前端和Zabbix服务器之间，
 - Zabbix服务器（代理）和Zabbix数据库之间。
- 目前，每个加密的连接都会打开一个完整的TLS握手，不会实现会话缓存和故障单。
- 根据网络延迟，添加加密会增加检查和操作的时间。例如，如果分组延迟为100ms，则打开TCP连接并发送未加密的请求大约需要200ms。加密约1000毫秒添加建立TLS连接。可能需要增加超时，否则一些监控项和动作在agent运行远程脚本时，使用加密会失败，而不加密则成功。
- 网络发现不支持加密。通过网络发现执行的Zabbix agent检查将是未加密的，如果Zabbix agent配置为拒绝未加密的连接，那么这种检查将不会成功。

用加密支持编译Zabbix

为了支持加密Zabbix必须编译并链接到三个加密库之一：

- *mbed TLS* (以前的*PolarSSL*) (版本1.3.9及更高版本1.3.x)。*mbed TLS* 2.x当前不支持，它不是1.3分支的替代替代，Zabbix将不会使用*mbed TLS* 2.x进行编译。
- *GnuTLS* (3.1.18版)
- *OpenSSL* (1.0.1版)

通过指定“configure”脚本的选项来选择库：

- `--with-mbedtls[=DIR]`
- `--with-gnutls[=DIR]`
- `--with-openssl[=DIR]`

例如，要使用OpenSSL配置服务器和agent代理的源，可以使用以下内容：

```
1. ./configure --enable-server --enable-agent --with-mysql --enable-ipv6 --with-net-snmp --with-libcurl --with-libxml2 --with-openssl
```

可以使用不同的加密库（例如具有OpenSSL的服务器，具有GnuTLS的agent代理）来编译不同的Zabbix组件。

如果您计划使用共享密钥（PSK），请考虑使用PSKs在Zabbix组件中使用GnuTLS或mbed TLS库。GnuTLS和mbed TLS库支持具有Perfect Forward Secrecy的PSK密码。OpenSSL库（版本1.0.1, 1.0.2c）支持PSK，但可用的PSK密码套件不提供完美转发保密。

连接加密管理

Zabbix中的连接可以使用：

- 非加密（默认）
- RSA certificate-based encryption
- 基于RSA证书的加密
- PSK-based encryption
- 基于PSK的加密

有两个重要参数用于为Zabbix组件之间的连接指定加密：

- `TLSConnect`
- `TLSAccept`

`TLSConnect` 指定要使用什么加密传出连接和可以采取3个中的一个（unencrypted, PSK, certificate）。`TLSConnect` 用于Zabbix proxy的配置文件（在主动模式下，仅指定与服务器的连接）和Zabbix agentd（用于主动检查）。在的zabbix前端的 `TLSConnect` 等效物是连接主机在字段配置→主机→<一些主机>→加密选项卡和连接代理字段中管理→代理→<一些代理>→加密选项卡。如果配置的连接加密类型失败，则不会尝试其他加密类型。

`TLSAccept` 指定允许进入连接的连接类型。连接类型：unencrypted, PSK, certificate。可以指定一个或多个值。`TLSAccept` 用于Zabbix proxy的配置文件（在被动模式下，仅指定来自服务器的连接）和Zabbix agentd（用于被动检查）。在的zabbix前端的 `TLSAccept` 等效物是从主机连接在字段配置→主机→<一些主机>→加密选项卡和从连接代理在字段管理→代理→<一些代理>→加密选项卡。

通常，您仅为传入加密配置一种类型的加密。但您可能希望切换加密类型，例如从加密到基于证书的最小停机时间和回滚可能性。要实现这一点，您可以将TLSAccept=unencrypted, cert在agentd配置文件中设置并重新启动Zabbix

agent。然后，您可以 `zabbixget` 使用证书测试与agent的连接。如果一切正常，你可以重新配置加密中的zabbix前端，agent配置→主机→<某些主机>→加密设置选项卡连接主机_到“证书”。当服务器配置缓存被更新（如果主机正在通过proxy进行监视时，proxy配置被更新），则与该agent的连接将被加密。如果一切正常工作，您可以 `TLSAccept=cert` 在agent配置文件中设置并重新启动Zabbix agent。现在agent将只接受加密的基于证书的连接。未加密和基于PSK的连接将被拒绝

以类似的方式，它可以在服务器和proxy上运行。如果在Zabbix前端主机配置中连接设置为“证书”，则只能从agent（主动检查）和 `zabbix_sender`（trapper项目）接受基于证书的加密连接。

很可能您将配置传入和传出连接使用相同的加密类型或根本不加密。但从技术上讲，可以非对称地进行配置，例如基于传入和基于PSK的出口连接的基于证书的加密。

有关概述，每个主机的加密配置将显示在Zabbix前端配置→主机右侧的代理加密列中。配置显示示例：

例	连接到主机	允许从主机连接	拒绝从主机连接
<code>NONE</code>	未加密	未加密	加密证书和基于PSK的证书
<code>CERT</code> <code>NONE</code> <code>PSK</code> <code>CERT</code>	加密，基于证书	基于加密证书	未加密和基于PSK的
<code>PSK</code> <code>NONE</code> <code>PSK</code> <code>CERT</code>	加密，基于PSK的	加密PSK为主	未加密和基于证书
<code>PSK</code> <code>NONE</code> <code>PSK</code> <code>CERT</code>	加密，基于PSK的	未加密和基于PSK的加密	以证书为基础
<code>CERT</code> <code>NONE</code> <code>PSK</code> <code>CERT</code>	加密，基于证书	未加密，PSK或基于证书的加密	-

默认是未加密的连接。必须单独为每个主机和代理配置加密。

zabbix_get和zabbix_sender加密

请参阅man-pages `zabbix_get` 和 `zabbix_sender` 以使用加密。

密码套件

在Zabbix启动期间内部配置了密码套件，并且依赖于加密库，目前它们不是用户可配置的。

按照从高到低顺序的库类型配置密码：

密码库	
<code>mbedtls TLS (PolarSSL) 1.3.9</code>	TLS-ECDHE-RSA-WITH-AES-128-GCM-SHA256TLS-ECDHE-RSA-WITH-AES-128-CBC-SHA256TLS-RSA-WITH-AES-128-CBC-SHA
<code>GnuTLS 3.1.18</code>	TLSECDHE_RSA_AES_128_GCM_SHA256TLS_ECDHE_RSA_AES_128_CBC_SHA256TLS_ECDHE
<code>_OpenSSL 1.0.2c</code>	ECDHE-RSA-AES128-GCM-SHA256ECDHE-RSA-AES128-SHA256ECDHE-RSA-AES128-SHAEE
<code>OpenSSL 1.1.0</code>	ECDHE-RSA-AES128-GCM-SHA256ECDHE-RSA-AES128-SHA256ECDHE-RSA-AES128-SHAEE

密码套件使用证书：

	TLS服务器		
TLS客户端	<i>mbed TLS (PolarSSL)</i>	<i>GnuTLS</i>	<i>OpenSSL 1.0.2</i>
<i>mbed TLS (PolarSSL)</i>	TLS-ECDHE-RSA-WITH-AES-128-GCM-SHA256	TLS-ECDHE-RSA-WITH-AES-128-GCM-SHA256	TLS-ECDHE-RSA-WITH-AES-128-GCM-SHA256
<i>GnuTLS</i>	TLS-ECDHE-RSA-WITH-AES-128-GCM-SHA256	TLS-ECDHE-RSA-WITH-AES-128-GCM-SHA256	TLS-ECDHE-RSA-WITH-AES-128-GCM-SHA256
<i>OpenSSL 1.0.2</i>	TLS-ECDHE-RSA-WITH-AES-128-GCM-SHA256	TLS-ECDHE-RSA-WITH-AES-128-GCM-SHA256	TLS-ECDHE-RSA-WITH-AES-128-GCM-SHA256

密码套件使用PSK：

	TLS服务器		
TLS客户端	<i>mbed TLS (PolarSSL)</i>	<i>GnuTLS</i>	<i>OpenSSL 1.0.2</i>
<i>mbed TLS (PolarSSL)</i>	TLS-ECDHE-PSK-WITH-AES-128-CBC-SHA256	TLS-ECDHE-PSK-WITH-AES-128-CBC-SHA256	TLS-PSK-WITH-AES-128-CBC-SHA
<i>GnuTLS</i>	TLS-ECDHE-PSK-WITH-AES-128-CBC-SHA256	TLS-ECDHE-PSK-WITH-AES-128-CBC-SHA256	TLS-PSK-WITH-AES-128-CBC-SHA
<i>OpenSSL 1.0.2</i>	TLS-PSK-WITH-AES-128-CBC-SHA	TLS-PSK-WITH-AES-128-CBC-SHA	TLS-PSK-WITH-AES-128-CBC-SHA



1 使用证书

概述

Zabbix可以使用PEM格式的RSA证书，由公共或内部认证机构（CA）签名。根据预先配置的CA证书进行证书验证。不支持自签名证书。可以选择使用证书撤销列表（CRL）。每个Zabbix组件只能配置一个证书。

有关如何设置和操作内部CA的更多信息，如何生成证书请求并签名，如何撤销证书，您可以找到许多在线操作，例如[OpenSSL PKI Tutorial v1.1](#)。

仔细考虑和测试证书扩展 - 请参阅[使用X.509 v3证书扩展的限制](#)。

证书配置参数

参数	Mandatory	描述
<i>TLSCAFfile</i>		包含用于对等证书验证的顶级CA证书的文件的完整路径名。在具有多个成员的证书链的情况下，它们必须被排序：较低级别的CA证书，然后是较高级别的CA证书。来自多个CA的证书可以包含在单个文件中。
<i>TLSCLRFfile</i>		包含证书吊销列表的文件的完整路径名。见 证书吊销清单(CRL) 中的注释。 Certificate Revocation Lists (CRL) 。
<i>TLSCertFile</i>		包含证书（证书链）的文件的完整路径名。设置此文件的访问权限 - 它必须只能由Zabbix用户读取。在具有多个成员的证书链的情况下，必须首先对其进行排序：服务器，proxy或agent证书，其次是较低级别的CA证书，然后是较高级别CA的证书。
<i>TLSKeyFile</i>	*	包含私钥的文件的完整路径名。设置此文件的访问权限 - 它必须只能由Zabbix用户读取。
<i>TLSServerCertIssuer</i>		允许的服务器证书发行者(issuer)。
<i>TLSServerCertSubject</i>		允许的服务器证书主体(subject)。

在Zabbix server上配置证书

- 为了验证对等证书，Zabbix server必须具有使用其顶级自签名根CA证书的文件访问权限。例如，如果我们期望来自两个独立根CA的证书，我们可以将其证书放入文件中 `/home/zabbix/zabbix_ca_file` :

```

1. Certificate:
2. Data:
3.      Version: 3 (0x2)
4.      Serial Number: 1 (0x1)
5.      Signature Algorithm: sha1WithRSAEncryption
6.      Issuer: DC=com, DC=zabbix, O=Zabbix SIA, OU=Development group, CN=Root1 CA
7.      ...
8.      Subject: DC=com, DC=zabbix, O=Zabbix SIA, OU=Development group, CN=Root1 CA
9.      Subject Public Key Info:
10.         Public Key Algorithm: rsaEncryption
11.         Public-Key: (2048 bit)
12.         ...
13. X509v3 extensions:

```

1 使用证书

```
14.          X509v3 Key Usage: critical
15.              Certificate Sign, CRL Sign
16.          X509v3 Basic Constraints: critical
17.              CA:TRUE
18.      ...
19.  -----BEGIN CERTIFICATE-----
20. MIID2jCCAsKgAwIBAgIBATANBgkqhkiG9w0BAQUFADB+MRMwEQYKCZImiZPyLGQB
21. ...
22. 9wEzdN8uTrqoyU78gi12npLj08LegRKjb5hFTVm0
23. -----END CERTIFICATE-----
24. Certificate:
25.     Data:
26.         Version: 3 (0x2)
27.         Serial Number: 1 (0x1)
28.     Signature Algorithm: sha1WithRSAEncryption
29.     Issuer: DC=com, DC=zabbix, O=Zabbix SIA, OU=Development group, CN=Root2 CA
30.     ...
31.     Subject: DC=com, DC=zabbix, O=Zabbix SIA, OU=Development group, CN=Root2 CA
32.     Subject Public Key Info:
33.         Public Key Algorithm: rsaEncryption
34.         Public-Key: (2048 bit)
35.     ...
36.     X509v3 extensions:
37.         X509v3 Key Usage: critical
38.             Certificate Sign, CRL Sign
39.         X509v3 Basic Constraints: critical
40.             CA:TRUE
41.     ...
42.  -----BEGIN CERTIFICATE-----
43. MIID3DCCAsSgAwIBAgIBATANBgkqhkiG9w0BAQUFADB/MRMwEQYKCZImiZPyLGQB
44. ...
45. vdGNYoSfvu41GQAR5Vj5FnRJRzv5XQ0Z3B6894GY1zY=
46. -----END CERTIFICATE-----
```

2. 将Zabbix服务器证书链放入文件中，例如 `/home/zabbix/zabbix_server.crt` :

```
1. Certificate:
2.     Data:
3.         Version: 3 (0x2)
4.         Serial Number: 1 (0x1)
5.     Signature Algorithm: sha1WithRSAEncryption
6.     Issuer: DC=com, DC=zabbix, O=Zabbix SIA, OU=Development group, CN=Signing CA
7.     ...
8.     Subject: DC=com, DC=zabbix, O=Zabbix SIA, OU=Development group, CN=Zabbix server
9.     Subject Public Key Info:
10.        Public Key Algorithm: rsaEncryption
11.        Public-Key: (2048 bit)
12.    ...
13.    X509v3 extensions:
14.        X509v3 Key Usage: critical
15.            Digital Signature, Key Encipherment
16.        X509v3 Basic Constraints:
```

```

17.          CA:FALSE
18.          ...
19.  -----BEGIN CERTIFICATE-----
20.  MIIECDCCAvCgAwIBAgIBATANBgkqhkiG9w0BAQUFADCBgTETMBEGCgmSJomT8ixk
21.  ...
22.  h02u1GHiy46GI+xfR3LsPwFKlkTaaLaL/6aaQ==
23.  -----END CERTIFICATE-----
24. Certificate:
25.   Data:
26.     Version: 3 (0x2)
27.     Serial Number: 2 (0x2)
28.   Signature Algorithm: sha1WithRSAEncryption
29.     Issuer: DC=com, DC=zabbix, O=Zabbix SIA, OU=Development group, CN=Root1 CA
30.   ...
31.     Subject: DC=com, DC=zabbix, O=Zabbix SIA, OU=Development group, CN=Signing CA
32.   Subject Public Key Info:
33.     Public Key Algorithm: rsaEncryption
34.     Public-Key: (2048 bit)
35.   ...
36.   X509v3 extensions:
37.     X509v3 Key Usage: critical
38.       Certificate Sign, CRL Sign
39.     X509v3 Basic Constraints: critical
40.       CA:TRUE, pathlen:0
41.   ...
42.  -----BEGIN CERTIFICATE-----
43.  MIID4TCCAsmgAwIBAgIBAjANBgkqhkiG9w0BAQUFADB+MRMwEQYKCZImiZPyLGQB
44.  ...
45.  dyCeWnvL7u5sd6ffo8iRny0QzbHKmQt/wUtcVIvWXdMIFJM0Hw==
46.  -----END CERTIFICATE-----

```

Here the first is Zabbix server certificate, followed by intermediate CA certificate.

这里首先是Zabbix server证书，其次是中间CA证书。

3. 将Zabbix server私钥放入文件中，例如 `/home/zabbix/zabbix_server.key`：

```

1. -----BEGIN PRIVATE KEY-----
2. MIIEwAIBADANBgkqhkiG9w0BAQEFAASCBKowggSmAgEAAoIBAQCTIXIJoVnNXD1
3. ...
4. IJLkhbybBYEf47MLhffWa7XvZTY=
5. -----END PRIVATE KEY-----

```

4. 在Zabbix server配置文件中编辑TLS参数，如下所示：

```

1. TLSCAFile=/home/zabbix/zabbix_ca_file
2. TLSCertFile=/home/zabbix/zabbix_server.crt
3. TLSKeyFile=/home/zabbix/zabbix_server.key

```

为Zabbix proxy配置基于证书的加密

1. 使用顶级CA证书, proxy证书(链)和私钥准备文件, 如在 [Zabbix server上配置证书](#) 中所述。编辑参数 `TLSCAFile`, `TLSCertFile`, `TLSKeyFile` 在proxy配置相应。

2. 对于proxy 代理编辑 `TLSConnect` 参数:

```
1. TLSConnect=cert
```

对于被动proxy编辑 `TLSAccept` 参数:

```
1. TLSAccept=cert
```

3. 现在你有一个基于证书的最小proxy配置。您可能希望通过设置 `TLSServerCertIssuer` 和 `TLSServerCertSubject` 参数来提高proxy安全性(请参阅[限制允许的证书发行者和主体](#))。

4. 在最终的proxy配置文件中, TLS参数可能如下所示:

```
1. TLSConnect=cert
2. TLSAccept=cert
3. TLSCAFile=/home/zabbix/zabbix_ca_file
4. TLSServerCertIssuer=CN=Signing CA,OU=Development group,O=Zabbix SIA,DC=zabbix,DC=com
5. TLSServerCertSubject=CN=Zabbix server,OU=Development group,O=Zabbix SIA,DC=zabbix,DC=com
6. TLSCertFile=/home/zabbix/zabbix_proxy.crt
7. TLSKeyFile=/home/zabbix/zabbix_proxy.key
```

1. 在Zabbix前端配置此proxy的加密:

- 转到: 管理→agent代理程序(proxies)
- 选择代理, 然后单击加密选项卡

在下面的示例中, 发行者(Issuer)和主体(fields)字段填写 - 请参阅

[\[manual:encryption/using_certificates#restricting_allowed_certificate_issuer_and_subject | 限制允许的证书发行者和主体\]](#)为什么以及如何使用这些字段。

对于主动proxy

Proxy		Encryption	
Connections to proxy		No encryption	PSK
Connections from proxy		<input type="checkbox"/> No encryption <input type="checkbox"/> PSK <input checked="" type="checkbox"/> Certificate	
Issuer	CN=Signing CA,OU=Development group,O=Zabbix SIA,DC=zabbix,DC=com		
Subject	CN=Zabbix proxy,OU=Development group,O=Zabbix SIA,DC=zabbix,DC=com		
		Update	Clone
		Delete	Cancel

对于被动proxy

Proxy Encryption

Connections to proxy: No encryption, PSK, Certificate

Connections from proxy: No encryption, PSK, Certificate

Issuer: CN=Signing CA,OU=Development group,O=Zabbix SIA,DC=zabbix,DC=com

Subject: CN=Zabbix proxy,OU=Development group,O=Zabbix SIA,DC=zabbix,DC=com

Update **Clone** **Delete** **Cancel**

为Zabbix agent配置基于证书的加密

1. 使用顶级CA证书，代理证书（链）和私钥准备文件，如在[Zabbix server配置证书](#)中所述。编辑参数 `TLSCAFile` , `TLSCertFile` , `TLSKeyFile` 在agent配置相应。

1. For active checks edit `TLSConnect` parameter:

2. 对于主动检查编辑 `TLSConnect` 参数：

```
1. TLSConnect=cert
```

对于被动检查编辑 `TLSAccept` 参数：

```
1. TLSAccept=cert
```

3. 现在，您有一个基于证书的最小agent配置。您可能希望通过设置 `TLSServerCertIssuer` 和 `TLSServerCertSubject` 参数提高agent安全性。（请参阅[限制允许的证书发行者和主体](#)）。

4. 在最终agent配置文件中，TLS参数可能如下所示：

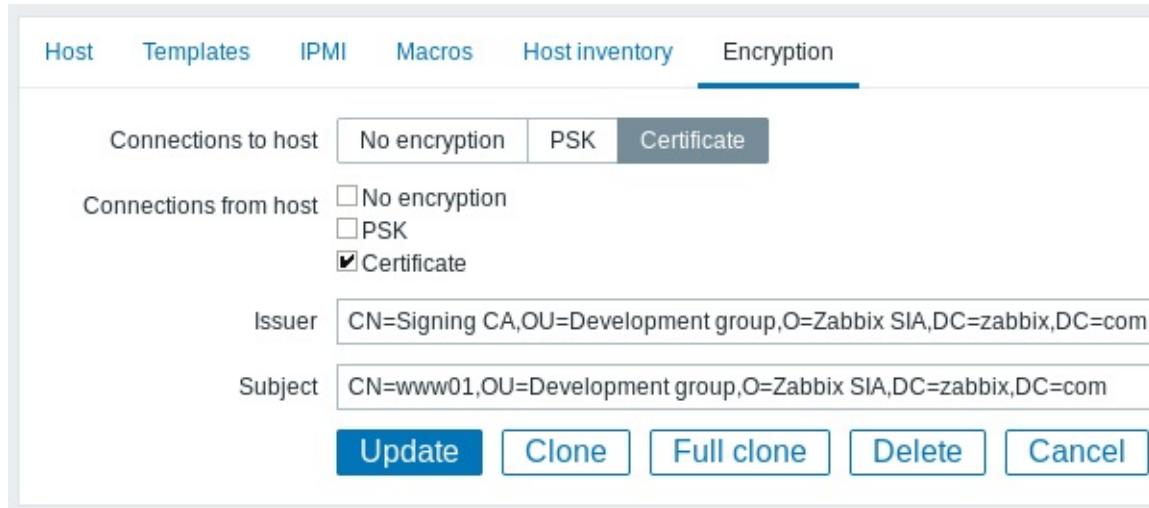
```
1. TLSConnect=cert
2. TLSAccept=cert
3. TLSCAFile=/home/zabbix/zabbix_ca_file
4. TLSServerCertIssuer=CN=Signing CA, OU=Development group, O=Zabbix SIA, DC=zabbix, DC=com
5. TLSServerCertSubject=CN=Zabbix proxy, OU=Development group, O=Zabbix SIA, DC=zabbix, DC=com
6. TLSCertFile=/home/zabbix/zabbix_agentd.crt
7. TLSKeyFile=/home/zabbix/zabbix_agentd.key
```

（例如，假设主机是通过proxy监视的，因此是proxy证书主体。）

5. 在Zabbix前端为此agent配置加密：

- Go to: Configuration → Hosts
- 转到：配置→主机
- Select host and click on **Encryption** tab
- 选择主机，然后单击加密选项卡

在下面的示例中，发行者和主体字段填写 - 请参阅[限制允许的证书发行者和主体](#)为什么以及如何使用这些字段。



限制允许的证书发行者和主体

当两个Zabbix组件（例如服务器和agent）建立TLS连接时，他们会检查对方的证书。如果对等证书由受信任的CA（具有预先配置的顶级证书 `TLS CA file`）签名，则是有效的和尚未过期，并通过一些其他检查，则可以进行通信。在最简单的情况下，不会检查证书发行者和主体。

这是一个风险 - 任何拥有有效证书的人都可以冒充任何人（例如，主机证书可以用来模拟服务器）。在通过专门的内部CA签署证书的小型环境中，这可能是可以接受的，并且冒充的风险较低。

如果您你的顶级CA用于发布不应被Zabbix接受的其他证书，或者你想降低冒充风险，您可以通过指定其发行者(Issuer)和主体(Subject)字符串来限制允许的证书。

例如，您可以在Zabbix proxy配置文件中写：

```
1. TLSServerCertIssuer=CN=Signing CA, OU=Development group, O=Zabbix SIA, DC=zabbix, DC=com
2. TLSServerCertSubject=CN=Zabbix server, OU=Development group, O=Zabbix SIA, DC=zabbix, DC=com
```

通过这些设置，主动proxy将不会与证书中具有不同发行者或主体字符串的Zabbix server通信，被动proxy将不接受来自此类服务器的请求。

有关发行者或主体字符串匹配的几个注释：

- 独立检查发行者和主体字符串。两者都是可选的。
- 允许使用UTF-8字符。
- 未指定的字符串意味着任何字符串都被接受。

- 字符串按“原样”比较，它们必须完全一致才能匹配。
- 匹配中不支持通配符和正则表达式。
- 只有[RFC 4514轻量级目录访问协议 \(LDAP\)](#)的一些要求：实现了可分辨名称的字符串表示：
 - 转义字符'''' (U+0022), '+' U+002B, ',' U+002C, ';' U+003B, '<' U+003C, '>' U+003E, '\' U+005C 在字符串中的任何地方。
 - 字符串开头处的转义字符空格(' ' U+0020) 或数字符号 ('#' U+0023)。
 - 字符串末尾的转义字符空间(' ' U+0020)。
- 如果遇到空字符(U+0000) ([RFC 4514](#)允许)，则匹配失败。
- 要求[RFC 4517轻量级目录访问协议 \(LDAP\)](#)：句法和匹配规则和 [RFC 4518轻量级目录访问协议 \(LDAP\)](#)：国际化字符串准备，由于所需的工作量不支持。

发行者和主体字符串和格式的字段顺序很重要！Zabbix遵循 [RFC 4514](#)建议，并使用“反向”字段顺序。

反向顺序可以举例说明：

```
1. TLSServerCertIssuer=CN=Signing CA,OU=Development group,O=Zabbix SIA,DC=zabbix,DC=com
2. TLSServerCertSubject=CN=Zabbix proxy,OU=Development group,O=Zabbix SIA,DC=zabbix,DC=com
```

请注意，它以低位 (CN) 开始，进入中级 (OU, O) 并以顶级 (DC) 字段结束。

默认情况下，*OpenSSL*将以“正常”的顺序显示证书发行者和主体字段，具体取决于使用的其他选项：

```
1. $ openssl x509 -noout -in /home/zabbix/zabbix_proxy.crt -issuer -subject
2. issuer= /DC=com/DC=zabbix/O=Zabbix SIA/OU=Development group/CN=Signing CA
3. subject= /DC=com/DC=zabbix/O=Zabbix SIA/OU=Development group/CN=Zabbix proxy
4.
5. $ openssl x509 -noout -text -in /home/zabbix/zabbix_proxy.crt
6. Certificate:
7. ...
8.     Issuer: DC=com, DC=zabbix, O=Zabbix SIA, OU=Development group, CN=Signing CA
9. ...
10.    Subject: DC=com, DC=zabbix, O=Zabbix SIA, OU=Development group, CN=Zabbix proxy
```

这里发行者和主体字符串从顶级 (DC) 开始，以低级 (CN) 字段结尾，空格和字段分隔符取决于所使用的选项。这些值都不会匹配Zabbix发行者 (Issuer) 和主体 (Subject) 字段！

要获得适当的发行者和主体字符串可用于Zabbix使用特殊选项调用\OpenSSL `-nameopt`
`esc_2253,esc_ctrl,utf8,dump_nostr,dump_unknown,dump_der,sep_comma_plus,dn_rev,sname` :

```
1. $ openssl x509 -noout -issuer -subject \
2.           -nameopt esc_2253,esc_ctrl,utf8,dump_nostr,dump_unknown,dump_der,sep_comma_plus,dn_rev,sname \
3.           -in /home/zabbix/zabbix_proxy.crt
4. issuer= CN=Signing CA,OU=Development group,O=Zabbix SIA,DC=zabbix,DC=com
5. subject= CN=Zabbix proxy,OU=Development group,O=Zabbix SIA,DC=zabbix,DC=com
```

现在字符串字段是反序，字段是逗号分隔的，可以在Zabbix配置文件和前端使用。

使用X.509 v3证书扩展的限制

- 主体备用名称 (*subjectAltName*) 扩展名。 Zabbix不支持来自*subjectAltName*扩展名的替代主体名称（如IP地址，电子邮件地址）。只能在Zabbix中检查“主体”字段的值（请参阅[限制允许的证书发行者和主体](#)）。如果证书使用*subjectAltName*扩展名，那么结果取决于加密工具包的特定组合。Zabbix组件被编译（可能工作或不工作，Zabbix可能拒绝接受来自对等体的证书）
- 扩展密钥使用扩展。如果使用，则通常需要*clientAuth* (TLS WWW客户端身份验证) 和*serverAuth* (TLS WWW服务器身份验证)。例如，被动检查的zabbix agent是作为TLS服务器，所以*serverAuth*必须在agent证书设置。对于主动检查agent证书需要*clientAuth*进行设置。*GnuTLS*在违规使用情况下发出警告，但允许通信进行。
- 名称限制扩展。并不是所有的加密工具包都支持它。此扩展可能会阻止Zabbix加载CA证书，此部分被标记为关键(*critical*) (取决于特定的加密工具包)。

证书撤销清单 (CRL)

如果证书受到威胁，CA可以通过在CRL中包含来撤销证书。可以在server器，proxy和agent的配置文件中配置CRL `TLSCRLFile`。例如：

```
1. TLSCRLFile=/home/zabbix/zabbix_crl_file
```

where `zabbix_crl_file` may contain CRLs from several CAs and look like:

那里 `zabbix_crl_file` 可能包含几个CA的CRL，如下所示

```
1. -----BEGIN X509 CRL-----
2. MIIB/QIBATANBgkqhkiG9w0BAQUFADCBgTETMBEGCgmSJomT8ixkARkWA2Nv
3. ...
4. treZeUPjb7LSmZ3K2hpZN7So0ZcAoHQ3Gwd9npuctg=
5. -----END X509 CRL-----
6. -----BEGIN X509 CRL-----
7. MIIB+TCB4gIBATANBgkqhkiG9w0BAQUFADB/MRMwEQYKCZImiZPyLGQBGRDY29t
8. ...
9. CAEebS2CND3ShBedZ8YSil5906JvaDP61lR51Ns=
10. -----END X509 CRL-----
```

CRL文件仅在Zabbix启动时加载。CRL更新需要重新启动。

如果使用[OpenSSL](#)编译Zabbix组件，要使用CRL，则证书链中的每个顶级和中级CA都必须具有相应的CRL（可以为空）`TLSCRLFile`。

使用CRL扩展的限制

- 权限密钥标识符扩展。 具有相同名称的CA的CRL在*MBEDTLS* (*PolarSSL*) 的情况下可能不起作用，即使使用“权限密钥标识符”扩展。

2 使用共享密钥

概述

Zabbix中的每个共享密钥（PSK）实际上是一对：

- 非秘密PSK identity（共享密钥一致性）字符串，
- 秘密PSK字符串值。

PSK identity（共享密钥一致性）字符串是非空的UTF-8字符串。

例如，“PSK ID 001 Zabbix agentd”。这是一个独特的名称，由Zabbix组件引用该特定的PSK。不要将敏感信息放在PSK identity（共享密钥一致性）字符串中 - 它通过网络未加密传输。

PSK值很难猜出十六进制数字的字符串，例

如“e560cb0d918d26d31b4f642181f5f570ad89a390931102e5391d08327ba434e9”。

大小限制

在Zabbix中有PSK identity（共享密钥一致性）和PSK值的大小限制，在某些情况下，加密库可以有下限：

组件	PSK identity最大大小	PSK值最小大小	PSK值最大大小
Zabbix	128 UTF-8 characters	128-bit (16-byte PSK, entered as 32 hexadecimal digits)	2048-bit (256-byte PSK, entered as 512 hexadecimal digits)
Zabbix	128个UTF-8字符	128位 (16字节PSK, 输入32位十六进制数字)	2048位 (256字节PSK, 输入512个十六进制数字)
GnuTLS	128 bytes (may include UTF-8 characters)	-	2048-bit (256-byte PSK, entered as 512 hexadecimal digits)
GnuTLS	128字节 (可能包括UTF-8字符)	-	2048位 (256字节PSK, 输入512个十六进制数字)
mbed TLS (PolarSSL)	128 UTF-8 characters	-	256-bit (default limit) (32-byte PSK, entered as 64 hexadecimal digits)
mbed TLS (PolarSSL)	128个UTF-8字符	-	256位 (默认限制) (32字节PSK, 以64位十六进制数字输入)
OpenSSL	127 bytes (may include UTF-8 characters)	-	2048-bit (256-byte PSK, entered as 512 hexadecimal digits)
OpenSSL	127字节 (可能包括UTF-8字符)	-	2048位 (256字节PSK, 输入512个十六进制数字)

Zabbix前端允许配置多达128个字符的长的PSK identity（共享密钥一致性）字符串和2048位长的PSK，而不管使用的加密库。如果某些Zabbix组件支持较低限制，则用户有责任为这些组件配置PSK identity（共享密钥一致性）和PSK值。超出长度限制会导致Zabbix组件之间的通信故障。

在Zabbix server使用PSK连接到agent之前，服务器将查找数据库中为该agent配置的PSK identity（共享密钥一致性）和PSK值（实际上在配置缓存中）。agent在收到连接后，从其配置文件中读取PSK identity（共享密钥一致性）和PSK值。如果双方具有相同的PSK identity（共享密钥一致性）字符串和PSK值，则连接可能会成功。

用户有责任确保没有两个具有相同PSK identity（共享密钥一致性）字符串但不同值的PSK。否则可能会导致使用PSK和PSK identity（共享密钥一致性）字符串的Zabbix组件之间的通信被中断。

生成PSK

例如，可以使用以下命令生成256位（32字节）PSK：

- with OpenSSL:
- 使用OpenSSL:

```
1. $ openssl rand -hex 32
2. af8ced32dfe8714e548694e2d29e1a14ba6fa13f216cb35c19d0feb1084b0429
```

- 使用GnuTLS:

```
1. $ psktool -u psk_identity -p database.psk -s 32
2. Generating a random key for user 'psk_identity'
3. Key stored to database.psk
4.
5. $ cat database.psk
6. psk_identity:9b8eafedfaae00cece62e85d5f4792c7d9c9bcc851b23216a1d300311cc4f7cb
7.
```

请注意，上面“psktool”命令产生PSK值的数据库文件。Zabbix只需要PSK文件中的PSK，因此应该从文件中删除'psk_identity:'。

配置PSK进行服务器和agent通信（示例）

在agent主机上，将PSK值写入文件，例如/home/zabbix/zabbix_agentd.psk。该文件必须在第一个文本字符串中包含PSK，例如：

```
1. 1f87b595725ac58dd977beef14b97461a7c1045b9a1c963065002c5473194952
```

设置PSK文件的访问权限 - 它必须只能由Zabbix用户读取。

在agent配置文件zabbix_agentd.conf中编辑TLS参数，例如set：

```
1. TLSConnect=psk
2. TLSAccept=psk
3. TLSPSKFile=/home/zabbix/zabbix_agentd.psk
4. TLSPSKIdentity=PSK_001
```

agent将连接到服务器（主动检查）并接受来自服务器和 `zabbix_get` 使用PSK连接。PSK身份将是“PSK_001”。

重新启动agent。现在可以使用zabbix_get例如：

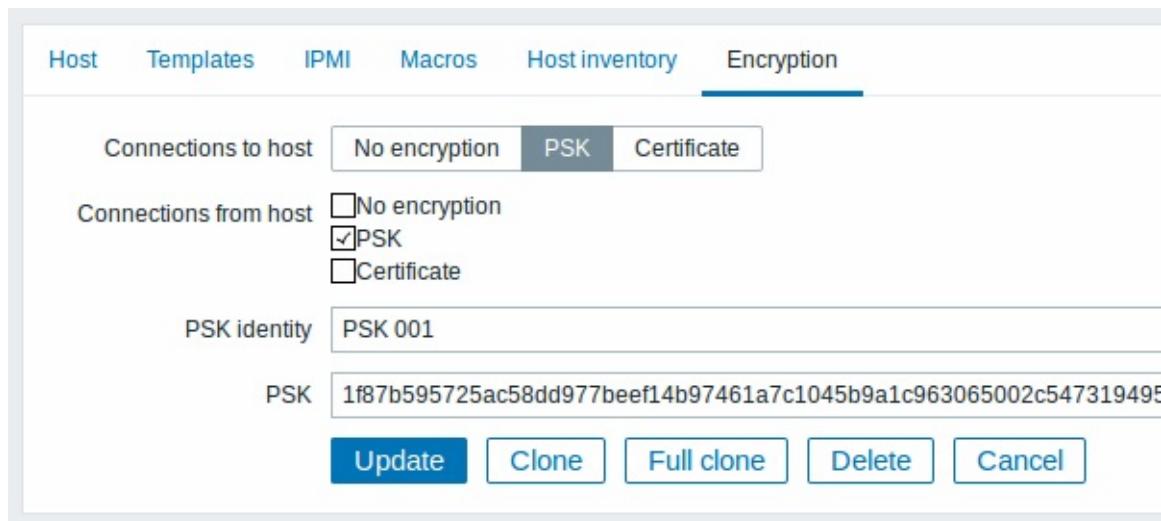
```
1. $ zabbix_get -s 127.0.0.1 -k "system.cpu.load[all,avg1]" --tls-connect=psk \
2.      --tls-psk-identity="PSK 001" --tls-psk-file=/home/zabbix/zabbix_agentd.psk
```

(为了最大限度地减少停机时间看看如何改变连接方式[连接加密管理](#))。

在Zabbix前端为此agen配置PSK加密：

- 转到：配置→主机
- 选择主机，然后单击加密选项卡

例如：



当配置缓存与数据库同步时，新连接将使用PSK。检查服务器和agent日志文件以获取错误消息。

为服务器 - 主动proxy通信配置PSK (示例)

在proxy上，将PSK值写入文件，例如/home/zabbix/zabbix_proxy.psk。该文件必须在第一个文本字符串中包含PSK，例如：

```
1. e560cb0d918d26d31b4f642181f5f570ad89a390931102e5391d08327ba434e9
```

设置PSK文件的访问权限 - 它必须只能由Zabbix用户读取。

在proxy配置文件zabbix_proxy.conf中编辑TLS参数，例如set：

```
1. TLSConnect=psk
2. TLSPSKFile=/home/zabbix/zabbix_proxy.psk
3. TLSPSKIdentity=PSK 002
```

proxy将使用PSK连接到服务器。PSK identity (共享密钥一致性) 将是“PSK 002”。

(为了最大限度地减少停机时间看看如何改变连接方式[连接加密管理t](#))。

在Zabbix前端配置此proxy的PSK。转到管理–agent代理程序，选择代理，转到“加密”选项卡。在“从代理连接”勾选PSK。将“PSK identity (共享密钥一致性)”字段填上“PSK 002”，“共享密钥 (PSK)”字段填上“e560cb0d918d26d31b4f642181f5f570ad89a390931102e5391d08327ba434e9”。点击“更新”。

重新启动proxy。它将开始使用基于PSK的加密连接到服务器。检查服务器和proxy日志文件以获取错误消息。

对于被动proxy，该过程非常相似。唯一的区别 - TLSAccept=psk在proxy配置文件中设置并在Zabbix前端设置“连接代理” PSK。

3 故障排除

一般建议

- 从理解开始，哪个组件充当TLS客户端，哪个组件充当问题的TLS服务器。Zabbix server, proxies and agents，取决于它们之间的交互，都可以作为TLS服务器和客户端。例如，Zabbix server连接到agent进行被动检查，充当TLS客户端。该agent是TLS服务器的角色。Zabbix agent，请求从proxy的主动检查列表，充当TLS客户端。prox服务器是TLS服务器。`zabbix_get` 并且 `zabbix_sender` 程序始终作为TLS客户端。
 - Zabbix使用相互验证。每一方验证对方，并可拒绝连接。例如，如果agent的证书无效，则连接到agent的Zabbix server可以立即关闭连接。反之亦然 - 如果服务器不被agent信任，Zabbix agent可关闭来自服务器的连接。
- 检查双方的日志文件 - 在TLS客户端和TLS服务器中。拒绝连接的一方可能会记录为什么被拒绝的准确理由。其他方面经常报告相当普遍的错误（例如“Connection closed by peer”，“connection was non-properly terminated”）。
- 有时配置错误的加密会导致混淆的错误消息，而不会指向真正的原因。在下面的小节中，我们尝试提供一个（简单的）的消息收集和可能有助于故障排除的可能原因。请注意，不同的加密工具包（OpenSSL，GnuTLS，mbed TLS（PolarSSL））在相同的问题情况下经常产生不同的错误消息。有时错误消息甚至依赖于两端的密码工具包的特定组合。

1 连接类型或权限问题

服务器配置为与agent程序连接，但agent仅接受未加密的连接

在服务器或proxy日志（带有`mbed TLS (PolarSSL) 1.3.11`）

```
1. Get value from agent failed: ssl_handshake(): SSL - The connection indicated an EOF
```

在服务器或proxy日志中（使用`GnuTLS 3.3.16`）

```
1. Get value from agent failed: zbx_tls_connect(): gnutls_handshake() failed: \
2. -110 The TLS connection was non-properly terminated.
```

在服务器或proxy日志中（使用`OpenSSL 1.0.2c`）

```
1. Get value from agent failed: TCP connection successful, cannot establish TLS to [[127.0.0.1]:10050]: \
2. Connection closed by peer. Check allowed connection types and access rights
```

一方连接证书，但另一方只接受PSK，反之亦然

在任意日志中（使用`mbed TLS (PolarSSL)`）：

```
1. failed to accept an incoming connection: from 127.0.0.1: ssl_handshake():\
2. SSL - The server has no ciphersuites in common with the client
```

在任意日志中（使用`GnuTLS`）：

```
1. failed to accept an incoming connection: from 127.0.0.1: zbx_tls_accept(): gnutls_handshake() failed:\
2. -21 Could not negotiate a supported cipher suite.
```

在任意日志中（使用`OpenSSL 1.0.2c`）：

```
1. failed to accept an incoming connection: from 127.0.0.1: TLS handshake returned error code 1:\ 
2. file .\ssl\s3_srvr.c line 1411: error:1408A0C1:SSL routines:ssl3_get_client_hello:no shared cipher:\ 
3. TLS write fatal alert "handshake failure"
```

2 证书问题

OpenSSL与CRL一起使用，对于证书链中的某些CA，其CRL不包含在“TLSCRLFile”中

TLS服务器日志在`mbed TLS (PolarSSL)`和`OpenSSL`对等(peer)的情况下：

```
1. failed to accept an incoming connection: from 127.0.0.1: TLS handshake with 127.0.0.1 returned error code 1: \
2.     file s3_srvr.c line 3251: error:14089086: SSL routines:ssl3_get_client_certificate:certificate verify
    failed: \
3.     TLS write fatal alert "unknown CA"
```

在TLS服务器日志在`GnuTLS`对等(peer)的情况下：

```
1. failed to accept an incoming connection: from 127.0.0.1: TLS handshake with 127.0.0.1 returned error code 1: \
2.     file rsa_pk1.c line 103: error:0407006A: rsa routines:RSA_padding_check_PKCS1_type_1:\
3.     block type is not 01 file rsa_eay.c line 705: error:04067072: rsa routines:RSA_EAY_PUBLIC_DECRYPT:paddin
```

服务器运行期间CRL过期或到期

`OpenSSL`, 在服务器日志中：

- 到期前：

```
1. cannot connect to proxy "proxy-openssl-1.0.1e": TCP successful, cannot establish TLS to [[127.0.0.1]:20004]:\
2.     SSL_connect() returned SSL_ERROR_SSL: file s3_clnt.c line 1253: error:14090086:\
3.     SSL routines:ssl3_get_server_certificate:certificate verify failed:\
4.     TLS write fatal alert "certificate revoked"
```

- 过期后：

```
1. cannot connect to proxy "proxy-openssl-1.0.1e": TCP successful, cannot establish TLS to [[127.0.0.1]:20004]:\
2.     SSL_connect() returned SSL_ERROR_SSL: file s3_clnt.c line 1253: error:14090086:\
3.     SSL routines:ssl3_get_server_certificate:certificate verify failed:\
4.     TLS write fatal alert "certificate expired"
```

这里的意思是，使用有效的CRL，撤销的证书将被报告为“已撤销证书”。当CRL过期时，错误消息更改为“证书已过期”，这是相当误导的。

`GnuTLS`, 在服务器日志中：

- 过期前后相同：

```
1. cannot connect to proxy "proxy-openssl-1.0.1e": TCP successful, cannot establish TLS to [[127.0.0.1]:20004]:\
2.     invalid peer certificate: The certificate is NOT trusted. The certificate chain is revoked.
```

mbed TLS (PolarSSL), 在服务器日志中:

- 到期前:

```
1. cannot connect to proxy "proxy-openssl-1.0.1e": TCP successful, cannot establish TLS to [[127.0.0.1]:20004]:\n2.     invalid peer certificate: revoked
```

- 过期后:

```
1. cannot connect to proxy "proxy-openssl-1.0.1e": TCP successful, cannot establish TLS to [[127.0.0.1]:20004]:\n2.     invalid peer certificate: revoked, CRL expired
```

3 PSK问题

PSK包含奇数个十六进制数字

Proxy或agent不启动， proxy或agent日志中的消息：

```
1. invalid PSK in file "/home/zabbix/zabbix_proxy.psk"
```

长度超过128字节的PSK identity (共享密钥一致性) 字符串传递给GnuTLS

在TLS客户端日志中

```
1. gnutls_handshake() failed: -110 The TLS connection was non-properly terminated.
```

在TLS服务器端日志中。

```
1. gnutls_handshake() failed: -90 The SRP username supplied is illegal.
```

超过32个字节的PSK传递到mbed TLS (PolarSSL)

在任何Zabbix日志中：

```
1. ssl_set_psk(): SSL - Bad input parameters to function
```

18. Web界面

请使用侧边导航栏来访问此章节的内容。

概览

为了从任何地方和任何平台轻松访问Zabbix，提供了基于Web的界面。

同时尝试访问同一主机不同的端口上的两个Zabbix前端安装，将会失败。登录第二个会终止第一个会话，依此类推。

1 前端部分

请使用侧边导航栏来访问此章节的内容。

1 监控中

概述

监视菜单全部是关于显示数据的。无论Zabbix被设置为收集、显示以及操作什么信息，都将在监控菜单各部分显示出来。

1 仪表板

1 仪表板

概述

监测中 → 仪表板 页面与普通汽车的中控仪表板类似，用于显示所有重要信息的汇总。

The screenshot shows the Zabbix Dashboard interface. At the top, there's a navigation bar with links for Monitoring, Inventory, Reports, Configuration, Administration, and a search bar. Below that is a sub-navigation bar for Dashboard, Problems, Overview, Web, Latendata, Triggers, Graphs, Screens, Maps, Discovery, and Services. A central toolbar has buttons for 'Add widget', 'Save changes', and 'Cancel'.

My CPU section: Contains two line graphs. The left graph is titled 'New host: CPU load (1h)' and the right is 'New host: Processor load (5 min average per core) (1h)'. Both graphs show data from history over the last 24 hours. Below each graph is a legend and some statistics (e.g., last, min, avg, max values).

Problems section: A table listing current problems. It includes columns for Time, Recovery time, Status, Info, Host, Problem + Severity, Duration, Ack, and Action. Two entries are shown:

- 2017-07-28 09:34: PROBLEM New host Free disk space is less than 20% on volume /
- 2017-07-27 12:47:30 PROBLEM Zabbix server Lack of free swap space of Zabbix server

Text item section: A table showing timestamped values for text items. Two entries are listed:

- 2017-08-02 09:34: Linux martin-HP-Pro-3900-Small-Form-Factor-PC 4.4.0-87-generic #119~14.04.1-080334
- 2017-08-02 08:03:44 Ubuntu SMP Tue Jul 18 14:51:32 UTC 2017 x86_64

Host status section: A table showing host groups and their status. It includes columns for Host group, Without problems, With problems, and Total.

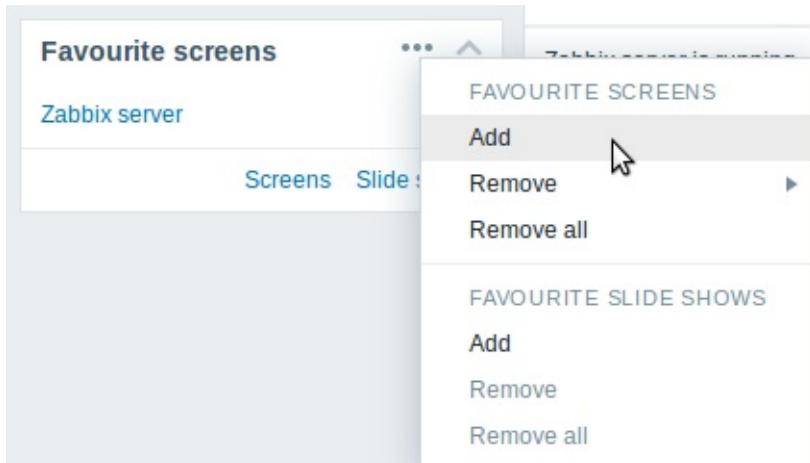
Host group	Without problems	With problems	Total
Discovered hosts	1	1	1
Zabbix servers	1	1	1

System status section: A table showing system status across host groups. It includes columns for Host group, Disaster, High, Average, Warning, Information, and Not classified.

Host group	Disaster	High	Average	Warning	Information	Not classified
Discovered hosts	1					

收藏夹

在仪表板中，一些窗口被称为收藏夹。你可以将你最需要的图形、自定义图形、聚合图形、幻灯片演示以及拓扑图的超链接添加到对应收藏夹。举个例子，当你需要添加一个新的聚合图形到聚合图形收藏夹。点击窗口部件中的菜单按钮，选择添加按钮。就像下面这张截图一样。



接下来会弹出一个窗口，显示已经存在的聚合图形，你只需要勾选的图形，然后单击下方的选择按钮，你所选的聚合图形会以超链接的形式显示在收藏夹中。

状态窗口

在本节第一张截图中，还有一些状态窗口， - Zabbix状态、系统状态、主机状态、最近20个问题、Web监测、发现状态，它们分别显示了各自角度的汇总数据。

就和你在截图中看到的一样，状态窗口和收藏夹窗口最多可以排放三列。此外，所有窗口都可以自由移动。只要长按窗口标题栏不放，就可以将这个窗口放在仪表板内任何你想要放的位置。

仪表板过滤器

单击标题栏中的  便可以访问仪表板过滤器 .

Dashboard

Dashboard filter **Enabled**

Host groups **Selected**

Show selected groups **Select**

Hide selected groups **Select**

Hosts Show hosts in maintenance

Triggers with severity Not classified
 Information
 Warning
 Average
 High
 Disaster

Trigger name

Problem display **All** **Separated** **Unacknowledged only**

Update **Cancel**

通过启用过滤器，你可以对显示在仪表板中的对主机和触发器进行限制，也可以对如何显示异常计数进行定义。

参数	描述
仪表板过滤器 <u>Dashboard filter</u>	单击此链接以启用/禁用仪表板过滤器。
主机组 <u>Host groups</u>	选择要显示的主机群组:所有(All) - 所有主机群组选择的(Selected) - 选定的主机群组。
查看选择的群组 <u>Show</u>	当主机群组字段为 选择的(<i>Selected</i>)，则该字段可用。输入要显示的主机群组。此字段可以自动完成，因此开始键入主机群组名称时，将提供匹配组的下拉列表。指定一个父主机

<code>selected groups</code>	群组，隐式选择全部嵌套主机组。来自这些主机组的主机数据将显示在仪表板中。如果没有输入主机组，则将显示所有的主机组。
<code>隐藏已选定机组 (Hide selected groups)</code>	当主机群组字段为 选择的(<i>Selected</i>)，则该字段可用。输入要隐藏的主机组。此字段可以自动完成，因此开始键入主机群组名称时，将提供匹配组的下拉列表。指定一个父主机群组，隐式选择全部嵌套主机组。注意 来自这些主机群组的主机数据将不会显示在仪表板中。例如，主机 001, 002, 003 A 可能在A组中，而主机002, 003同时也在B组中。如果我们选择同时显示A组且隐藏B组，只有主机001的数据会显示在仪表板中。
<code>主机 (Hosts)</code>	勾选是否查看维修中主机，以在仪表板中显示。
<code>有严重性的触发器 (Triggers with severity)</code>	勾选触发器严重等级，以在仪表板中显示。
<code>触发器名称 (Trigger name like)</code>	用此字符串，对在系统状态、主机状态以及最后20个问题窗口中显示的触发器汇总进行限制。
<code>问题显示 (Problem display)</code>	问题统计有以下选项：全部(All) - 显示全部问题统计分开的(Separated) - 未确认异常计数将会显示为，分离成若干总异常计数仅未确认的(Unacknowledged only) - 只显示未确认异常数据



如果仪表板过滤被启用，会显示一个有绿色指示灯的过滤图标：

主机菜单

在最后20个问题窗口单击主机，会调出主机菜单。包括自定义脚本、最新数据、触发器、主机资产记录、图形和主机聚合图形的链接。

The screenshot shows the 'Problems' section of the Zabbix web interface. A context menu is open over a row for a host named 'New host'. The menu items are:

- SCRIPTS
- Detect operating system
- Ping
- Traceroute
- GO TO
- Host inventory
- Latest data
- Triggers
- Graphs
- Host screens

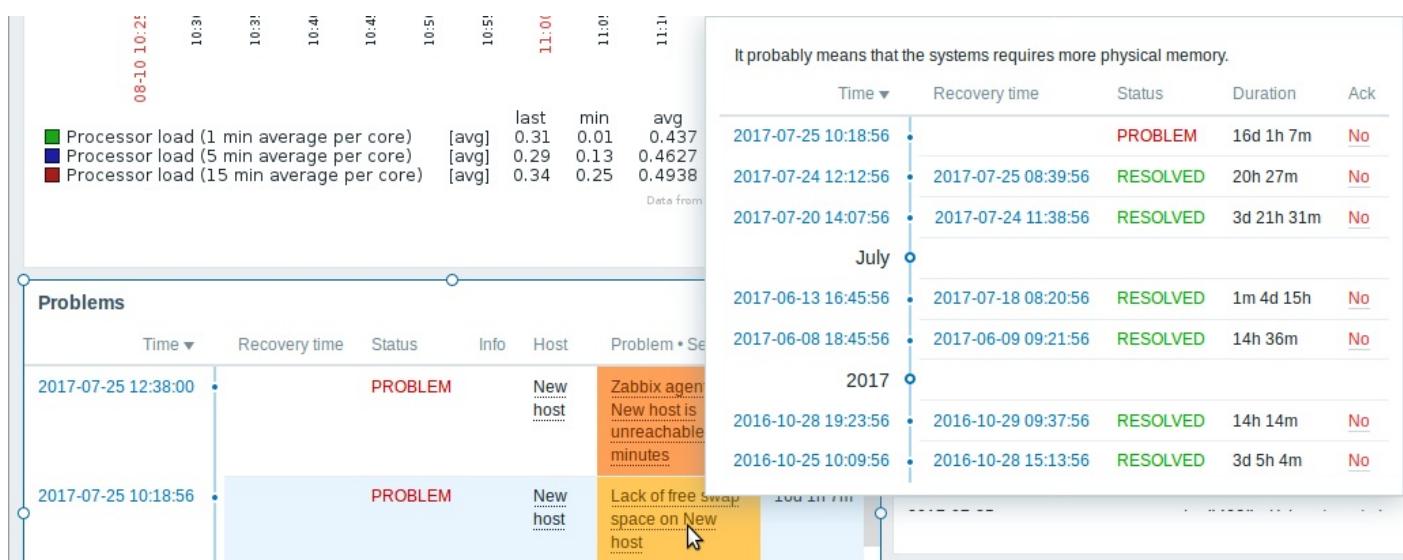
单击WEB前端中其他位置中的主机名，也可以访问此主机菜单。

- 监控中(Monitoring) → 问题(Problems)
- 监控中(Monitoring) → 问题(Problems) → 事件详情(Event details)

- 监控中(Monitoring) → 概述(Overview)1)
- 监控中(Monitoring) → 最后数据(Latest data)
- 监控中(Monitoring) → 触发器(Triggers)
- 监控中(Monitoring) → 汇聚图形(Screens) (在主机问题以及主机群组窗口中)
- 监控中(Monitoring) → 拓扑图(Maps)
- 报表(Reports) → 触发器TOP 100 (Triggers top 100)

触发器事件弹出式菜单

点击 近20个问题窗口中的问题 (issue)，会调出触发器事件弹出式菜单。它包括该事件的列表，事先定义好的触发器描述和可点击的URL。



1)

主机位置选择为左侧

2 问题

概述

在监测中→问题中，你可看到当前存在什么问题。问题指处在“问题”状态下的触发器。

Problems								Export to CSV		
Time	Severity	Recovery time	Status	Info	Host	Problem	Duration	Ack	Actions	Tags
01:36:53	Warning	01:36:53	RESOLVED	Zabbix server 1	Log trigger2		0	No	Done 2	Application Application: Apache
01:22:49	Not classified	01:26:19	RESOLVED	New host	CPU load too high on 'New host' for 3 minutes	3m 30s	No		Cloud CPU Local	+++
01:04:47	Warning	01:15:47	RESOLVED	New host	Disk I/O is overloaded on New host	11m	No			
01:04:19	Not classified	01:14:49	RESOLVED	New host	CPU load too high on 'New host' for 3 minutes	10m 30s	No		Cloud CPU Local	+++
Yesterday										
2016-06-21 10:18:54	Warning		PROBLEM	Zabbix server 1	Lack of free swap space on Zabbix server 1	2m 22d 15h	Yes 1			
June										
2016-04-05 11:18:08	Warning		PROBLEM	New host	Free disk space is less than 20% on volume /	5m 9d 14h	No			
2016-04-05 11:18:08	Warning		PROBLEM	New host	Free disk space is less than 20%	5m 9d 14h	No			
Displaying 7 of 7 found										

列	描述
时间 (Time)	显示问题开始时间。
严重等级 (Severity)	显示异常严重等级。问题严重等级取决于其触发器的严重等级。触发器严重等级的颜色用作单元背景色。已处理过的问题，其背景颜色是绿色。
恢复时间 (Recovery time)	显示问题恢复时间
状态 (状态)	显示问题状态被显示为：问题(Problem) - 未解决问题已恢复(Resolved) - 近期已解决问题。你可通过使用过滤器来隐藏近期已解决问题新解决的和近期解决的问题会闪烁30分钟。已解决问题共显示30分钟。触发器显示时间的配置在管理 → 通用 → 触发器显示选项(Trigger displaying options)。
主机 (Host)	显示问题主机。
问题 (Problem)	显示问题名称问题名称取决于其触发器的问题名称。
持续时间 (Duration)	显示问题持续时间
确认 (Ack)	显示问题确认状态：是(Yes) - 绿色字体表明问题已确认。如果一项问题的所有事件都已被确认，则此项问题被认为已被确认。不(No) - 红色链接表明有未被确认的事件。如果您单击该链接，将切到一个批量确认屏幕，该触发器的所有问题可以立即确认。如果管理→一般中选择了启用事件确认，此列将会显示。
动作 (Actions)	显示 动作(Action)状态：进行中(In progress) - 正在进行动作 完成(Done) - 动作已完成错误(Failures) - 动作失败对问题采取的动作（例如发送或执行远程命令的通知）也会被显示。
标签 (Tags)	显示时间标签(Event tags) (如果存在)。

使用过滤器

您可以使用过滤器只显示你感兴趣的问题。过滤器位于目录上方。

参数	描述
显示__ (Show)	按问题状态进行筛选：最近的问题(Recent problems) - 显示未解决以及近期已解决异常（默认）)问题(Problems) - 显示未解决的问题历史记录(History) - 显示所有事件的历史记录
主机群组__ (Host Group)	按一个或多个主机群组筛选：指定一个父主机群组，指定一个父主机群组，隐式选择全部嵌套主机群组。
主机__ (Host)	按一个或多个主机进行筛选
应用集__ (Application)	按应用集名称筛选
触发器__ (Trigger)	按一个或多个触发器筛选
问题__ (Problem)	按问题名称筛选
最小触发器严重等级__ (Minimum trigger severity)	按最小触发器严重等级筛选
小于一定时长__ (Age less than)	按异常存在时长筛选
主机资产记录__ (Host inventory)	按资产记录类型和值进行筛选
标签__ (Tags)	按事件标签名称和值进行筛选
显示维护中的主机异常__ (Show hosts in maintenance)	标记复选框，以显示维护中的主机异常。
仅显示未确认的异常__ (Show unacknowledged only)	标记复选框，仅显示未确认的异常。
显示细节__ (Show details)	标记复选框，以显示异常的潜在触发器表达式。

查看详细信息

监测中→问题，异常开始和恢复的时间都有链接，单击链接可以打开更多事件细节。

Event details							
Event source details		Acknowledgements					
Host	Zabbix host	Time User Message User action					
Trigger	Disk I/O is overloaded on Zabbix host	No data found.					
Severity	Warning						
Problem expression	{Zabbix host system.cpu.util[iowait].avg(5m)}>20						
Recovery expression							
Event generation	Normal	No data found.					
Allow manual close	No						
Disabled	No						
Event details							
Event	Disk I/O is overloaded on Zabbix host	Time Recovery time Status Age Duration Ack Actions					
Time	2016-10-25 07:48:47	2016-10-25 07:49:47	RESOLVED	1m 4d 10h	1m	No	
Acknowledged	No	2016-10-25 07:46:47	2016-10-25 07:47:47	RESOLVED	1m 4d 10h	1m	No
Resolved by	Trigger	2016-10-24 10:51:47	2016-10-24 11:09:47	RESOLVED	1m 5d 7h	18m	No
Tags		2016-09-12 17:52:47	2016-09-12 17:55:47	RESOLVED	2m 17d	3m	No
		2016-09-12 01:04:47	2016-09-12 01:15:47	RESOLVED	2m 17d 17h	11m	No
		2016-09-11 21:04:47	2016-09-11 21:09:47	RESOLVED	2m 17d 21h	5m	No
		2016-09-07 07:38:47	2016-09-07 07:42:47	RESOLVED	2m 22d 10h	4m	No
		2016-09-06 07:41:47	2016-09-06 07:45:47	RESOLVED	2m 23d 10h	4m	No

3 概述

概述

监测中→概述。提供了总览触发器状态一个平台，或者将不同主机的数据放在一起进行比较。 可用下列显示的选项：

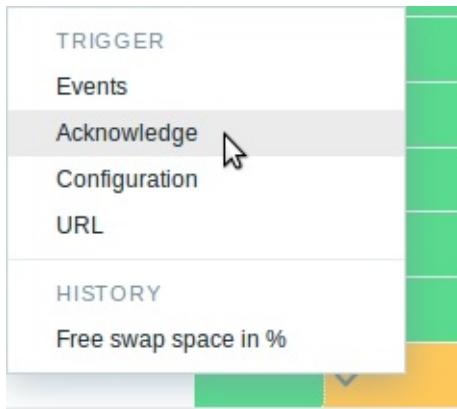
- 在主机群组下拉菜单中选择全部或指定主机群组
- 在类型下拉菜单中选择要显示的信息类型（触发器或数据）
- 在主机位置下拉菜单中选择主机名位于表格顶端或表格左侧显示

触发器概述

当在类型下拉菜单中选中触发器，将会出现与下列截图类似的触发器概览界面。如图，两个本地主机的触发器状态以色块呈现（颜色取决于触发器状态）显示：

The screenshot shows the Zabbix 'Overview' page for triggers. At the top, there are several filter options: 'Group: all', 'Type: Triggers', 'Hosts location: Top', and buttons for 'Select', 'Add', 'Remove', 'Filter', and 'Reset'. Below these filters, there are dropdowns for 'Triggers status' (Any), 'Acknowledge status' (Any), 'Minimum trigger severity' (Not classified), and a date range 'Age less than' set to 14 days. There is also a checkbox for 'Show hosts in maintenance' which is checked. On the left side, under the 'TRIGGERS' section, a list of trigger descriptions is shown, such as '/etc/passwd has been changed on {HOST.NAME}', 'Configured max number of opened files is too low on {HOST.NAME}', etc. To the right of the triggers, there is a grid representing two hosts. The first host, labeled 'NEW HOST', has a green background with a blue arrow pointing down. The second host, labeled 'ZABBIX SERVER', has a green background with a yellow arrow pointing up. A small checkmark is visible in the bottom right corner of the Zabbix Server's grid cell.

注意，近期触发器更改（最近三十分钟内）会显示为闪烁的色块。蓝色上下箭头表示触发器有依赖性。在鼠标经过时，会显示依赖性的细节。单击触发器色块，会出现一个菜单，为触发器相关的事件、配置、确认、URL或是一些预定图形或最新值提供了链接。



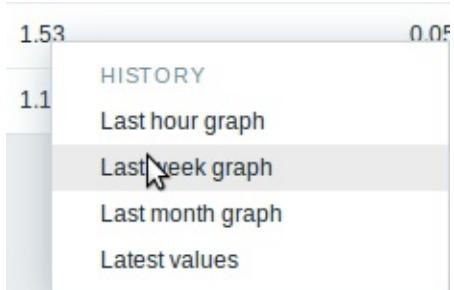
数据概述

当在类型下拉菜单中选中数据，将会出现与下列截图类似的数据概览界面。如图，显示了两个本地主机的性能监控项的数据。

Overview		
	Group	Type
Filter ▲		
Filter by application	Performance	Select
	Filter	Reset
ITEMS		
Context switches per second	4.55 Ksp/s	169 sp/s
CPU idle time	0 %	86.72 %
CPU interrupt time	0 %	0 %
CPU iowait time	0 %	1.02 %
CPU nice time	0.0083 %	0 %
CPU softirq time	0.02 %	0.26 %
CPU steal time	0 %	0 %
CPU system time	85 %	4.4 %
CPU user time	15.22 %	7.15 %
Interrupts per second	3.3 Kips	74 ips
Processor load (1 min average per core)	2.38	0.03
Processor load (5 min average per core)	1.79	0.06
Processor load (15 min average per core)	1.07	0.06

默认情况下，只显示最近24小时内下降的值。此限制已被引入，其目的是提高大量最新数据的初始加载时间。也可以通过更改 在 `include/defines.inc.php` 中的 `ZBX_HISTORY_PERIOD` 常量(constant) 来更改此限制。

单击一段数据，会出现一个菜单，提供了一些预定义图形或最新值的链接。



4 Web监测

概述

在监测中→web监测部分，用于展示web监控(web scenarios)的当前信息

Web monitoring				
HOST	NAME ▲	NUMBER OF STEPS	LAST CHECK	STATUS
Zabbix server	Zabbix frontend	5	2016-01-02 16:45:32	OK
Displaying 1 of 1 found				

注意：被禁用主机的名称在主机下拉菜单和列表中，都以红色显示。从 Zabbix2.2.0 开始，被禁用主机的数据保持可用。

默认情况下，只显示在最近24小时内显示的值。为了提高网页监控大量页面的初始加载时间，引入了这一限制。也可以通过在 `include/defines.inc.php` 中改变 `ZBX_HISTORY_PERIOD` 定义(constant) 的值来更改限制

web监测列表中的的名称，链接到关于其更详细的统计数据：



5 最新数据

概述

监测→最新数据 可以用来查看监控项收集的最新值，以及访问各种项目图表。

您第一次打开此页时，不会显示任何内容。

The screenshot shows the 'Latest data' interface. At the top, there are search fields for 'Host groups', 'Hosts', and 'Application', each with a 'Select' button. Below these are two checkboxes: 'Show items without data' (unchecked) and 'Show details' (unchecked). There are 'Filter' and 'Reset' buttons. The main area has a table header with columns: HOST, NAME, LAST CHECK, LAST VALUE, and CHANGE. A message at the bottom says 'Specify some filter condition to see the values.'

HOST	NAME	LAST CHECK	LAST VALUE	CHANGE
Specify some filter condition to see the values.				

您需要在筛选器中进行选择，如主机组、主机、应用集或监控项名称来访问相应数据。

The screenshot shows the 'Latest data' interface after filtering by 'Zabbix server'. The table now displays 13 CPU-related items. Each item has a checkbox, a name, a last check timestamp, a last value, and a change percentage. There are also 'Graph' links for each item. The table header remains the same: HOST, NAME, LAST CHECK, LAST VALUE, and CHANGE.

HOST	NAME	LAST CHECK	LAST VALUE	CHANGE
Zabbix server	CPU (13 items)			
	Processor load (15 min average per core)	2016-01-02 19:16:13	0.05	Graph
	Processor load (5 min average per core)	2016-01-02 19:24:15	0.08	-0.02 Graph
	Processor load (1 min average per core)	2016-01-02 19:24:14	0.09	-0.16 Graph
	Interrupts per second	2016-01-02 19:24:52	128 ips	+52 ips Graph
	CPU user time	2016-01-02 19:24:24	4.24 %	+4.03 % Graph
	CPU system time	2016-01-02 19:24:23	2.63 %	+1.76 % Graph
	CPU steal time	2016-01-02 19:24:22	0 %	Graph
	CPU softirq time	2016-01-02 19:24:21	0.2 %	+0.11 % Graph
	CPU nice time	2016-01-02 19:24:20	0 %	Graph
	CPU iowait time	2016-01-02 19:24:19	0.51 %	-0.77 % Graph
	CPU interrupt time	2016-01-02 19:24:18	0 %	Graph
	CPU idle time	2016-01-02 19:24:17	93.94 %	-1.74 % Graph
	Context switches per second	2016-01-02 19:24:56	217 sps	+15 sps Graph
New host	CPU (13 items)			
Zabbix server	Filesystems (5 items)			
New host	Filesystems (5 items)			
Zabbix server	General (5 items)			

在显示列表中，在主机及相关应用前单击  以显示该主机和应用集的最新值。您可以展开所有主机和所有应用集，

通过在标题行中单击  来显示所有项目。

注意：被禁用主机的名称显示为红色。从Zabbix2.2.0起，被禁用主机数据（包括图表和项目值列表）在最新数据页可以被访问。

本页面的列表针对监控项展示以下列：监控项名称、最近检查记录、最后一个值、更改量(Change)以及一个跳转到项目值的简单图表/历史记录的链接。

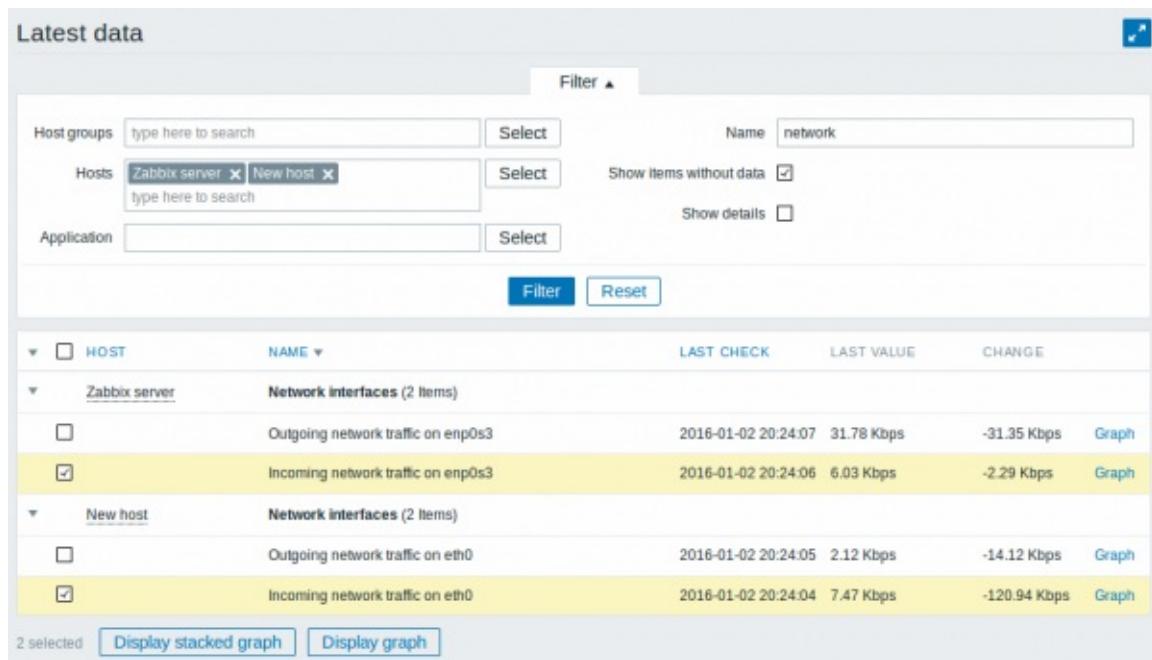
默认情况下，只显示在最近24小时内出现的值。为了提高大量最新数据的初始加载时间，引入这一限制。也可以通过在include/defines.inc.php 中改变ZBX_HISTORY_PERIOD 定义(constant) 值来更改限制.

使用筛选器

您可以使用筛选器只显示您感兴趣的监控项。筛选器链接位于表格上方中部。您可以使用它来过滤主机群组、主机、应用集、取自监控项名称中的字符串；还可以选择是否显示没有收集到数据的项目（查看无资料项目）。

指定一个父主机组，隐式选择所有嵌套的主机组。

显示详细信息会增加显示监控项相关的以下项目：该监控项的键值、间隔设置、历史记录及趋势的保存时间设置、监控项的类型和监控项的错误（良好/不支持）等详细信息，同时键值是一个链接到监控项配置的超链接。



HOST	NAME	LAST CHECK	LAST VALUE	CHANGE
Zabbix server	Network interfaces (2 items)			
<input type="checkbox"/>	Outgoing network traffic on enp0s3	2016-01-02 20:24:07	31.78 Kbps	-31.35 Kbps Graph
<input checked="" type="checkbox"/>	Incoming network traffic on enp0s3	2016-01-02 20:24:06	6.03 Kbps	-2.29 Kbps Graph
New host	Network interfaces (2 items)			
<input type="checkbox"/>	Outgoing network traffic on eth0	2016-01-02 20:24:05	2.12 Kbps	-14.12 Kbps Graph
<input checked="" type="checkbox"/>	Incoming network traffic on eth0	2016-01-02 20:24:04	7.47 Kbps	-120.94 Kbps Graph

默认情况下，会显示没有数据的项目，但不显示详细内容。

比较项目图表

可以在第二列的复选框选择几个监控，然后用简单图形或堆叠图比较它们的数据。选择感兴趣的监控项，然后单击表下所需图形的按钮，即可查看图形。

链接到值的历史/简单图形

提供最新值列表中的最后一列：

- 一个 历史链接 链接（用于所有文本项）-链接到的列表（values/ 500个最新values）显示前一个项目值的历史记录。
- 一 图表 链接（用于所有数字项）-链接到一个简单图形(simple graph)。 图形被调用出来后，从右上角的下拉框也可以切换为显示值(values)或最新500个值(500 latest values)。

Zabbix server: Processor load (1 min average per core)

Values As plain text

Filter ▲

Zoom: 5m 15m 30m 1h 2h 3h 6h 12h 1d 3d 7d 14d 1m All

2015-11-03 05:09 - 2015-11-03 11:09 (now!)

6h fixed

TIMESTAMP	VALUE
2015-11-03 11:09:14	0.01
2015-11-03 11:08:14	0.03
2015-11-03 11:07:14	0.09
2015-11-03 11:06:14	0.24
2015-11-03 11:05:14	0.04
2015-11-03 11:04:14	0.11
2015-11-03 11:03:14	0
2015-11-03 11:02:14	0.01
2015-11-03 11:01:14	0.02
2015-11-03 11:00:14	0.05

此列表中显示的是“原始的”值，即指未经处理的指。

注意：所显示的值的总量由搜索限制和筛选结果参数的值定义，这两个值可以在[管理\(Administration\) -> 一般\(General\)](#)中设置。

6 触发器

概述

监测→触发器 显示触发器状态。

The screenshot shows the Zabbix 'Triggers' page. At the top, there are several filter options: 'Trigger status' (Any, Recent problems, Problems), 'Acknowledge status' (Any), 'Events' (Show all (7 days)), 'Minimum trigger severity' (Not classified), 'Age less than' (14 days), 'Name' (empty), 'Application' (empty), 'Host inventory' (Type empty), and checkboxes for 'Show hosts in maintenance' and 'Show details'. Below the filters is a table of triggers:

Severity	Status	Info	Time	Recovery time	Age	Duration	Ack	Host	Name	Description
Not classified	OK		2016-11-29 13:56:49		15m 39s		No	102 New host	CPU load too high on 'New host' for 3 minutes	Add
	RESOLVED		2016-11-29 13:46:49	2016-11-29 13:56:49	25m 39s	25m 39s	No			
	RESOLVED		2016-11-29 13:26:49	2016-11-29 13:32:49	45m 39s	20m	No			
	RESOLVED		2016-11-29 12:51:49	2016-11-29 13:25:19	1h 20m 39s	35m	No			
	RESOLVED		2016-11-29 12:41:19	2016-11-29 12:45:19	1h 31m 9s	10m 30s	No			
	RESOLVED		2016-11-29 12:32:49	2016-11-29 12:36:49	1h 39m 39s	8m 30s	No			
Not classified	PROBLEM		2016-11-29 12:06:55		2h 5m 33s		No	1 New host	SSH service is unavailable	Add
Warning	PROBLEM		2016-11-29 12:06:55		2h 5m 33s		No	1 New host	Free disk space is less than 20% on volume /	

At the bottom right of the table, it says 'Displaying 3 of 3 found'.

列	描述
严重度 (Severity)	显示触发器严重度 严重度的颜色用作问题触发器的单元格背景。正常的触发器，背景为绿色。
状态 (Status)	显示触发器状态 - 正常(OK)或者问题(Problem) 默认情况下，近期更改过状态的触发器会闪烁30分钟。此外，即使过滤器设置为只显示问题，近期状态更改为正常的触发器也会显示30分钟。 文本颜色和闪烁选项可在 管理 → 通用 → 触发器显示选项 中进行配置。
信息 (Info)	带有问号的灰色图标表示有一些相关信息要显示。 如果您将鼠标指针移到上面，消息就会显示出来。
时间 (Time)	对于触发器 - 将显示上次触发状态更改的日期和时间。 对于触发事件 - 将显示触发器状态更改为“问题(PROBLEM)”时的日期和时间。
恢复时间 (Recovery time)	对于触发事件 - 将显示触发器状态更改为“正常”时的日期和时间 点开触发器条目前的▶ 来查看触发器事件，显示恢复时间。 如果为筛选器中的事件选择“显示全部”或“显示确认”选项，则可以看到触发器事件。
历时 (Age)	显示触发状态上次更改的至今的时间。
持续时间 (Duration)	显示问题状态的持续时间。 持续时间指触发器事件的持续时间。
已确认 (Acknowledged)	显示触发器确认状态： Yes - 绿色文本说明触发器已确认。如果所有事件都已确认，则认为触发器已被确认。 No - 红色链接代表未被确认事件。如果您单击该链接，将被带到一个批量确认页面，该触发器的所有事件都可以立即确认。 Note: 如果您只想确认单个事件，去检测中 → 问题页面。 No events - 如果触发器没有事件，Zabbix 2.0.4支持显示此字符串，在此之前，这些触发器都显示为已确认。

主机__ (Host)	显示触发器主机。它也是一个调用用户自定义脚本、最新数据、主机资产记录、图形和主机聚合图形的链接。
名称__ (Name)	显示触发器名称。它也是一个调用触发器事件列表、触发器配置以及监控数据简单图形的链接。如果在触发器配置中定义了链接，则链接列表也可能包含自定义触发器URL。
描述__ (Description)	触发器描述链接。由低级发现创建的触发器，不添加描述。

使用筛选器

您可以使用筛选器只显示您感兴趣的项目。筛选器位于表格上方。

默认情况下，只显示“近期问题(Recent problem)”状态中的触发器，包括问题触发器和近期才更改为“正常(OK)”状态的触发器。您也可以选择显示“问题(Problem)”状态下的触发器（仅问题触发器）或“任何(Any)”状态下的触发器。

请注意，如果您选择“任何(Any)”，那么在大型设备上处理的数据量可能是压倒性的，页面可能需要很长时间来加载。 您可以通过 `?filter_rst=1` 重置筛选器来替代URL参数，以此解决这个问题。

批量编辑选项

列表下方的按钮提供了一个批量编辑选项：

- 批量确认 - 确认所选触发器

要使用此选项，需标记相应触发器前的复选框，并点击批量确认。

7 图形

概述

监测中 → 图形 可显示已配置的任何[自定义图形\(custom graph\)](#)。



要显示一个图形，按顺序在页面右上的群组、主机及图形三个下拉菜单中选择需要显示的图形。

注意：在主机下拉菜单中，被禁用主机以红色突出显示。。从Zabbix2.2.0起，被禁用主机图形也可以从本页可以访问。

时段选择器

图形上方的筛选器部分包含一个时段选择器，方便您轻松选择所需时段。选择器中的选择滑块可以在时间轴上来回拖动、调整大小，高效地更改所选的需要显示时段。左侧在滑块区域之上，有一些常用链接用来选择的滑块大小。左侧滑块区域下方同样有一些可供选择的链接，它们用于的将滑块在时间轴上向前/向后移动（或在右下为“动态的”时，调整滑块大小）。右侧滑块区域上的链接用于调用一个日历控件，其作用与左上部的链接一样，用于设置指定的开始/结束时间。滑块区域右下的“固定的” / “动态的”切换链接具有以下作用：

- 当使用右上日历控件更改 开始 / 结束 时间，对时段是否保持不变，进行控制。
- 当“固定的”时，使用左下部的链接（« 1m 7d 1d 12h 1h 5m | 5m 1h 12h 1d 7d 1m »）将移动滑块，而不改变其大小。在“动态的”情况下，“<”和“>”将在各个方向上增大滑块。应用控件将会在各个方向上增大滑块。
- 当“固定的”时，按下时间轴两端较大的“<”和“>”按钮将移动滑块不改变它的大小。在“动态的”情况下，“<”和“>”将在各个方向上增大滑块。滑块会按其大小变化。举个例子，如果滑块大小是一个月，它将移动一个月（“固定的”）或者增大1天“动态的”。

控制显示时间的另一种方法是用鼠标左键在图形中长按，使图形中出现一个区域高亮，一旦鼠标左键释放，图形将放大到高亮区域显示。

控制

标题栏中有三个可用按钮：

-  - 在[仪表板\(Dashboard\)](#)中的收藏部件中添加图表
-  - 将图表显示，重设到显示最后一小时数据的原始设置。
-  - 使用完整的浏览器窗口来显示图表。

8 聚合图形

概述

在“监控中 - >聚合图形”部分，您可以配置，管理和查看Zabbix 聚合图形(*screens*) 以及 幻灯片演示(*slide shows*)。

当您打开此部分时，您将看到您访问的最后一个聚合图形/幻灯片演示或您可以访问的所有实体的列表。聚合图形/幻灯片演示列表可以按名称过滤。

Zabbix 3.0之后的版本中，所有的聚合图形/幻灯片演示可以是公共的或私人的。公共的可用于所有用户，而私有的用户只能由其所有者和实体共享的用户访问。

使用标题栏中的下拉菜单在聚合图形/幻灯片演示之间切换。

聚合图形列表

NAME	DIMENSION (COLS X ROWS)	ACTIONS
Offices	2 x 10	Properties Constructor
Servers	2 x 3	Properties Constructor
Zabbix server	2 x 2	Properties Constructor
Zabbix server2	3 x 3	Properties Constructor

Displaying 4 of 4 found

显示数据：

列	描述
Name	聚合图形的名称。点击名称 查看 对应的聚合图形。
Dimensions	聚合图形 的列数和行数。
Actions	有两个动作可用：属性 - 编辑一般 聚合图形 的 属性 (名称和尺寸)构造函数 - 访问网格化的聚合图形 元素 来做编辑

要 [创建新的聚合图形](#)，点击右上角的创建屏幕按钮。要从XML文件导入屏幕，请单击右上角的导入按钮。导入聚合图形的用户将被设置为其所有者。

批量编辑选项

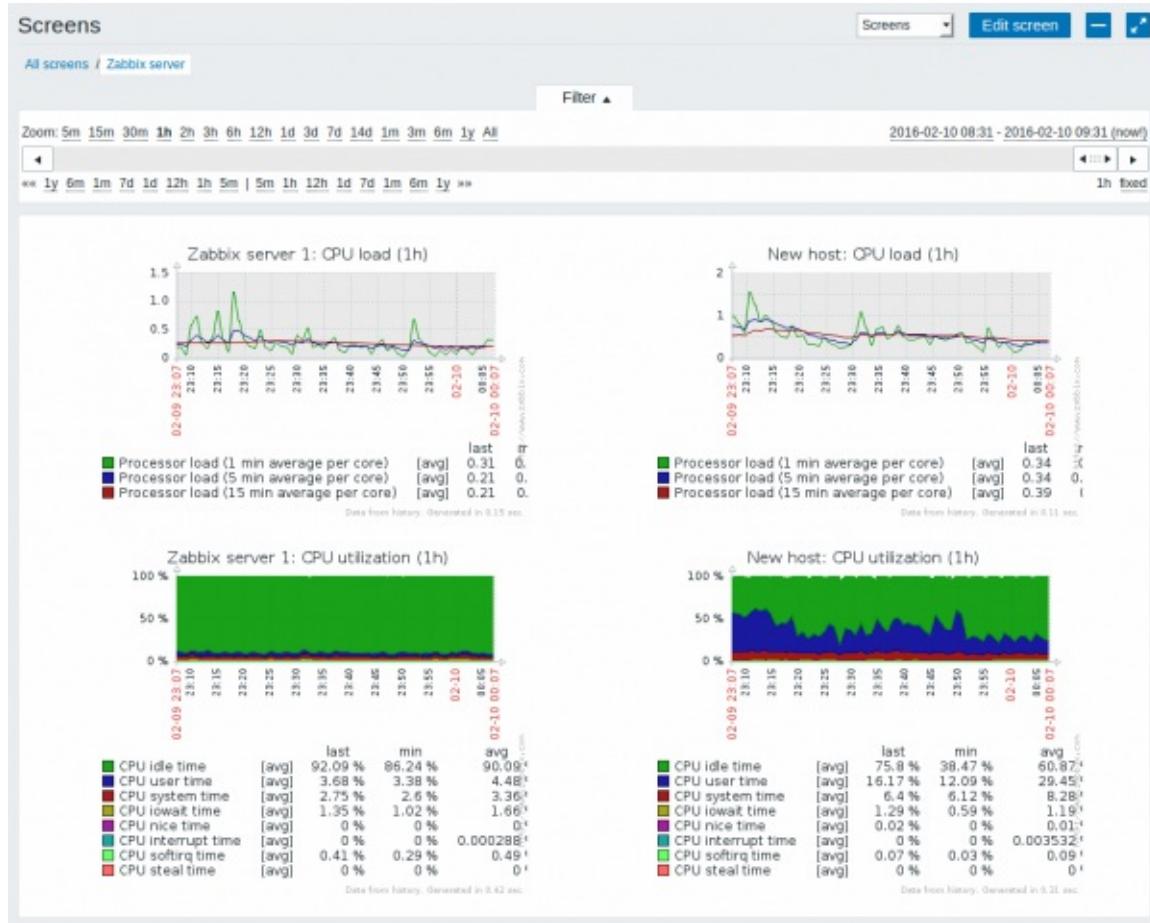
列表下方的按钮提供了一些批量编辑选项：

- *Export* - 将/聚合图形导出到XML文件
- *Delete* - 删除聚合图形,

要使用这些选项，请在相应聚合图形之前标记复选框，然后单击所需的按钮。

查看聚合图形

要查看聚合图形，请在所有聚合图形列表中单击其名称。



时间段选择器

聚合图形上方的过滤器部分包含时间段选择器。它允许您轻松选择所需的时间段，影响图形中显示的数据等。

控件

标题栏中有三个控制按钮：

- Edit screen - 去聚合图形构造函数编辑聚合图形
- + - 将聚合图形添加到仪表板收藏夹小部件中
- ⤢ - 使用浏览器窗口全屏显示聚合图形

幻灯片演示列表

使用标题栏中的下拉菜单从聚合图形切换到幻灯片演示。

NAME	DELAY	NUMBER OF SLIDES	ACTIONS
Zabbix administrators	30s	2	Properties
Zabbix administrators2	30s	4	Properties

Displaying 2 of 2 found

显示数据：

列	描述
Name	幻灯片演示的名称。点击名称 查看幻灯片演示 。
Delay	显示一个幻灯片演示的默认持续时间。
Number of slides	显示中幻灯片演示的数量。
Actions	只有一个操作可做 属性 - 编辑幻灯片演示 属性

要[创建](#)幻灯片演示，点击右上角的创建幻灯片放映按钮。

批量编辑选项

列表下方的按钮提供了一个批量编辑选项：

- *Delete* - 删除幻灯片演示

要使用此选项，请在相应幻灯片显示之前标记复选框，然后单击 *Delete*。

查看幻灯片演示

要查看幻灯片演示，请在所有幻灯片演示列表中单击其名称。

控件

标题栏中有四个控制按钮：

- [Edit slide show](#) - 幻灯片演示属性编辑
- + - 将幻灯片演示添加至[仪表板\(Dashboard\)](#)收藏小部件
- - 浏览器全屏显示幻灯片演示
- - 放慢或者加快幻灯片演示播放速度

引用聚合图形

聚合图形可以由 'elementid' 和 'screenname' GET参数引用。 例如，

```
1. http://zabbix/zabbix/screens.php?screenname=Zabbix%20server
```

将使用该名称打开聚合图形 (Zabbix服务器)。

如果同时指定“elementid” (聚合图形ID) 和“screenname” (聚合图形名)，则“screenname” (聚合图形名) 具有较高的优先级。

9 拓扑图

概述

在监控 ->拓扑图部分，您可以配置，管理和查看 [拓扑图](#)。

当您打开此部分时，您将看到您访问的最后一张拓扑图或您可以访问的所有拓扑图的列表。 拓扑图列表可以按名称过滤。

自Zabbix 3.0以来，所有拓扑图都可以是公共的或私有的。 所有用户都可以使用公共拓扑图，而私人拓扑图只能由其所有者和拓扑图共享的用户访问。

拓扑图列表

<input type="checkbox"/> NAME	WIDTH	HEIGHT	ACTIONS
Network	590	400	Properties Constructor
Offices	700	550	Properties Constructor
User map	800	600	Properties Constructor

Displaying 3 of 3 found

显示数据：

列	描述
<i>Name</i>	拓扑图的名称。点击名称 浏览 对应的拓扑图。
<i>Width</i>	显示拓扑图宽度
<i>Height</i>	显示拓扑图的高度
<i>Actions</i>	两项操作可做：属性 - 编辑拓扑图整体 属性架构 - 访问网格化的 拓扑图元素 来编辑

配置新的拓扑图，点击右上角的创建拓扑图按钮。要从XML文件导入拓扑图，请单击右上角的导入按钮。 导入拓扑图的用户将被设置为其所有者。

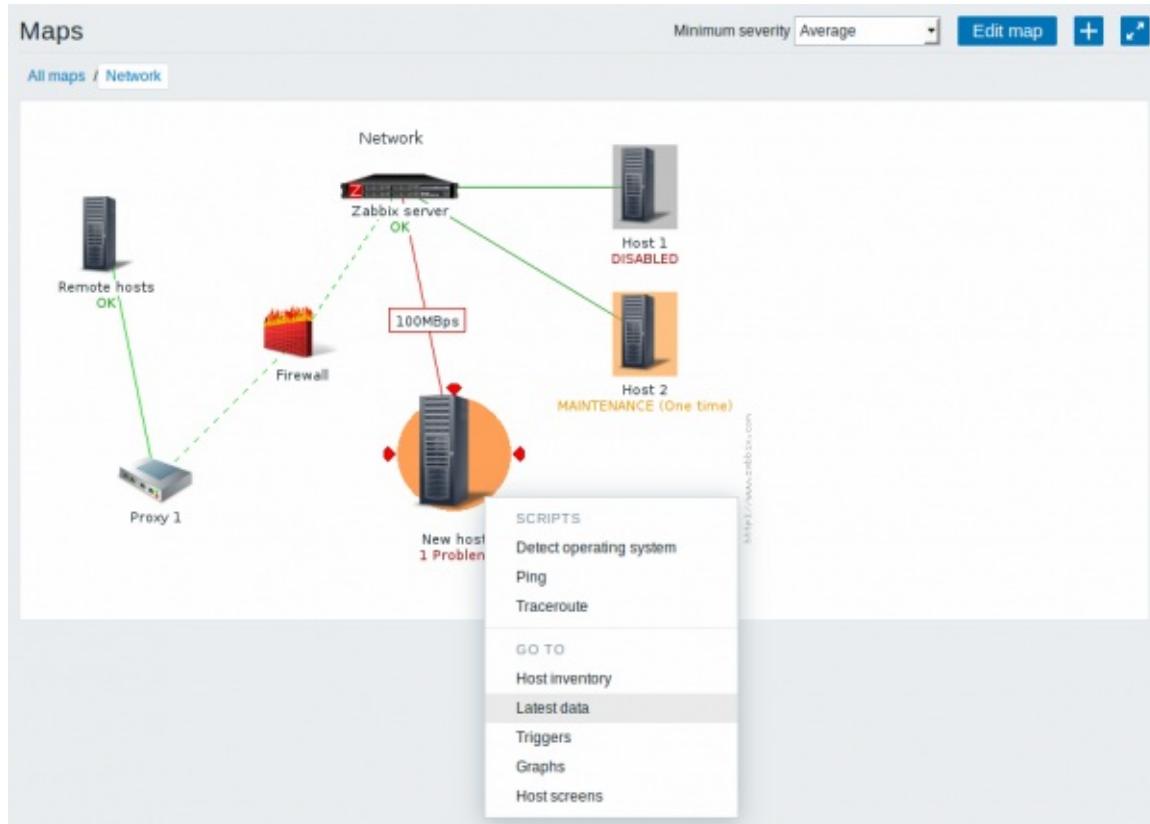
列表下方的两个按钮提供了一些批量编辑选项：

- *Export* - 将拓扑图导出为XML文件
- *Delete* - 删除拓扑图

要使用这些选项，请在各个拓扑图之前标记复选框，然后单击所需的按钮。

查看拓扑图

要查看拓扑图，请在所有拓扑图列表中单击其名称。



您可以使用拓扑图标标题栏中的下拉列表来选择要显示的问题触发器的最低严重性级别。标记为 *default* 的严重性是映射配置中设置的级别。如果拓扑图包含子拓扑图，则导航到子拓扑图将保留较高级别的拓扑图严重性。

图标突出显示

如果一个拓扑图元素处于问题状态，则以圆圈突出显示。圆的填充颜色对应于问题触发器的严重性颜色。所选严重性级别以上的问题只会与元素一起显示。如果所有问题都得到承认，则会显示圆圈周围的粗绿色边框。另外，如果一个主机在 [维修](#) 状态，则突出显示橙色的填充方块，禁用（未监视）主机以灰色突出显示，填充方块和着重显示只有在图标突出显示复选框被标记在拓扑图的[配置](#)中时，才会显示。

最近更改的标记

向内指向元素周围的红色三角形表示最近的触发状态变化 - 最近30分钟内发生的最近的触发状态更改。如果触发器状态上的标记元素更改复选框在拓扑图[配置](#)中标记，则会显示这些三角形。

链接

点击拓扑图元素会打开一些包含一些可用链接的菜单。

控件

标题栏中有三个控制按钮：

- Edit map** - 转到拓扑图构造器来编辑地图内容
- +** - 将拓扑图添加到仪表板中的收藏小部件中

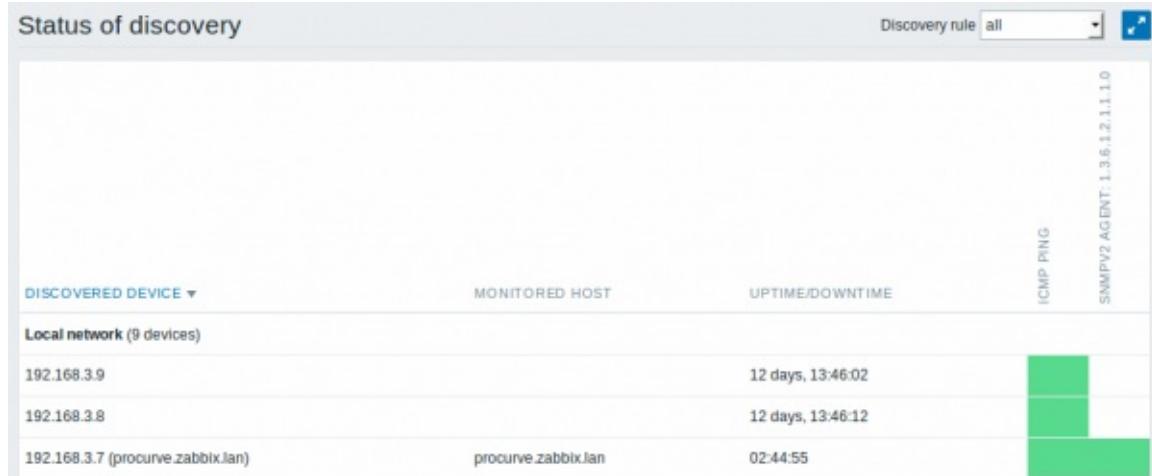


- 浏览器全屏显示拓扑图

10 自动发现

概述

在“监控中 ->自动发现”部分中，显示了[网络发现](#)的部分结果。发现的设备按发现规则排序。



如果设备已被监控，主机名将列在监控的主机列中，并且在上次发现后发现或丢失的设备的持续时间显示在正常运行/停机时间列中。

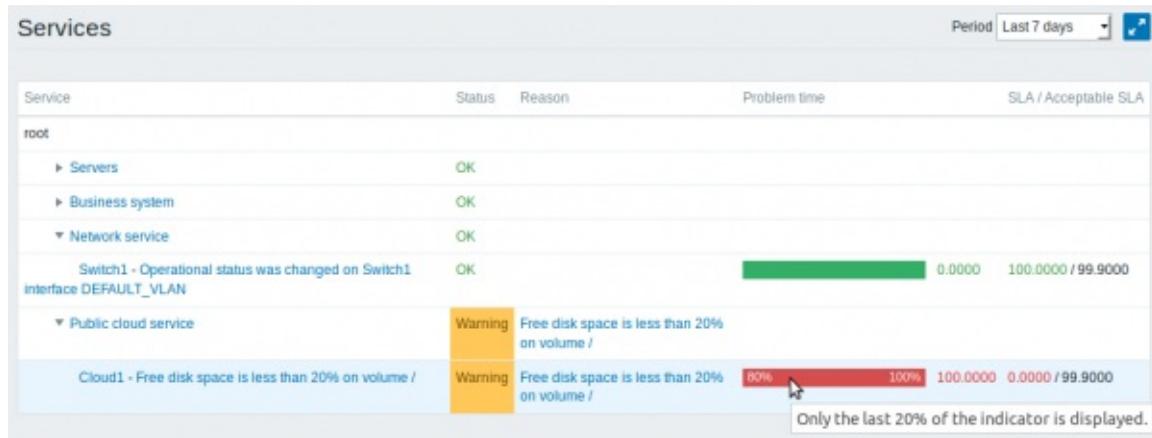
之后，显示每个发现的设备的各个服务状态的列。每个单元格的工具提示将显示单独的服务正常运行时间或停机时间。

只有在至少一台设备上找到的那些服务才会有一列显示其状态的列。

11 IT服务

概述

在“检测中 - > IT服务”部分中，显示IT 服务的状态。

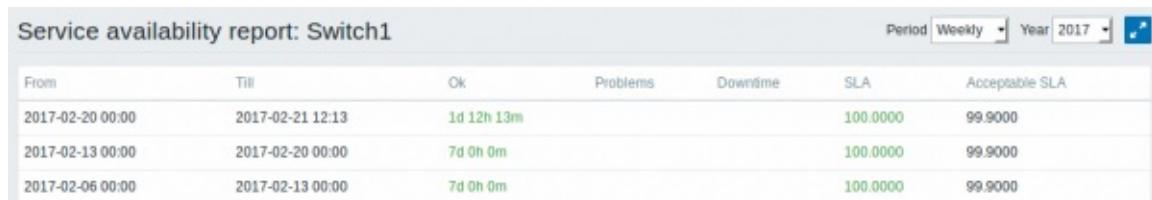


显示现有IT服务的列表以及其状态和SLA的数据。从右上角的下拉菜单中，您可以选择所需的显示周期。

显示数据：

参数	描述
Service	服务名称。
Status	服务状态:OK - 正常(触发颜色和严重性) - 表示一个问题及其严重性
Reason	表示问题的原因(如果有)。
Problem time	显示SLA栏。绿/红比表示可用性/问题的比例。显示栏显示SLA的最后20% (从80%到100%)。该栏包含可用性数据图表的链接。
SLA/可接受的SLA	显示当前SLA /预期SLA值。如果当前值低于可接受水平，则显示为红色。

您还可以单击服务名称以访问 IT服务可用性报告。



在这里，您可以在更长的时间内按日/每周/每月/每年评估IT服务可用性数据

2 库存

概述

“库存”菜单具有部分，根据所选参数概述主机库存数据，以及查看主机库存明细的能力。

1 概述

概述

库存 - >概述部分提供了有关[主机盘点](#)数据概述的方法。

要显示的概述，请选择一个主机组（或所有组）和显示数据的库存字段。将显示与所选字段的每个条目相对应的主机数量。

Host inventory overview		Group	all	Grouping by	Type
TYPE		HOST COUNT	▼		
Zabbix server		1			
Workstation		1			
Switch		1			

概述的完整性取决于主机维护多少库存信息。

主机数列中的数字是链接；它们导致这些主机在主机库存表中被过滤掉。

Host inventory							Group	all
							Filter	▲
							Field	Type
							exact	Zabbix server
HOST	GROUP	NAME	TYPE	OS	SERIAL NUMBER	TAG	MAC ADDRESS	A
Zabbix server	Discovered hosts, Zabbix servers	linux-qvvt	Zabbix server	Linux	linux-qvvt 3.11.10-21-default #1 SMP Mon Jul 21 15:28:46 U			
Displaying 1 of 1 found								

2 主机

概述

在库存 -> 主机部分显示主机的**库存量数据**。从右上角的下拉列表中选择一个组，以显示该组中的主机的库存量数据。您还可以通过任何清单字段过滤主机，仅显示您感兴趣的主机。

The screenshot shows a table titled 'Host inventory' with one row of data. The columns are labeled: HOST, GROUP, NAME, TYPE, OS, SERIAL NUMBER, TAG, and MAC ADDRESS. The data row is: Zabbix server, Discovered hosts, Zabbix servers, linux-qvvt, Zabbix server, Linux linux-qvvt 3.11.10-21-default #1 SMP Mon Jul 21 15:28:46 U. A filter bar at the top has 'Field' set to 'Type' and 'exact' value 'Zabbix server'. Buttons for 'Filter' and 'Reset' are present.

要显示所有主机清

单，在组下拉列表中选择“全部”，清除过滤器中的比较字段，然后按“过滤器”。虽然表中只显示了一些关键的库存字段，但您也可以查看该主机的所有可用库存信息。为此，请单击第一列中的主机名。== 库存详情==概述选项卡包含有关主机的一些一般信息以及预定义脚本的链接，最新的监视数据和主机配置选项：

The screenshot shows the 'Details' tab of the host configuration interface for 'Zabbix server_1'. It includes sections for Agent interfaces (IP address 192.168.3.220, Port 10050), SNMP interfaces (IP address 127.0.0.1, Port 161), OS (Linux linux-qvvt 3.11.10-21-default #1 SMP Mon Jul 21 15:28:46 U), Description (Added on 2015-07-28), Monitoring (Web, Latest data, Triggers, Problems, Graphs, Screens), Configuration (Host, Applications 13, Items 81, Triggers 47, Graphs 12, Discovery 3, Web 1), and a 'Cancel' button.

详细信息选项卡包含

主机的所有可用库存明细：

The screenshot shows a 'Details' tab selected in a Zabbix host configuration interface. The host is identified as a 'Zabbix server' named 'linux-qvvt'. It runs Linux version 3.11.10-21-default. The system was built on Mon Jul 21 15:28:46 UTC 2014. The MAC address is listed as A [enp0s3] 08:00:27:62:c4:53. The host is located at 'Head Office' in 'Riga'. A 'Cancel' button is visible at the bottom.

Type Zabbix server

Name linux-qvvt

OS Linux linux-qvvt 3.11.10-21-default #1 SMP Mon Jul 21 15:28:46 U

OS (Full details) Linux version 3.11.10-21-default (geeko@buildhost) (gcc version 4.8.1 20130909 [gcc-4_8-branch revision 202388] (SUSE Linux)) #1 SMP Mon Jul 21 15:28:46 UTC 2014 (9a9565d)

MAC address A [enp0s3] 08:00:27:62:c4:53, [enp0s3] 08:00:27:62:c4:53

Location Head Office

Site city Riga

[Cancel](#)

库存数据的完整性取

决于与主机保持多少库存信息。 如果没有维护信息，则“详细信息”禁用。

3 报告

概述

“报告”菜单包含几个部分，其中包含各种预定义和用户可定制的报告，重点是显示Zabbix的状态，触发器和收集数据等参数的概述。

1 Zabbix的状态

概述

在报表中 - > Zabbix的状态显示关键系统数据的摘要。

Status of Zabbix		
PARAMETER	VALUE	DETAILS
Zabbix server is running	Yes	localhost:10051
Number of hosts (enabled/disabled/templates)	45	3 / 0 / 42
Number of items (enabled/disabled/not supported)	113	106 / 2 / 5
Number of triggers (enabled/disabled [problem/ok])	60	60 / 0 [2 / 58]
Number of users (online)	3	2
Required server performance, new values per second	1.55	

此报告也显示为[仪表板](#) 中的小部件

显示数据

参数	值	详述
Zabbix 服务器 正在运行	Zabbix服务器的状态: Yes - 服务器正在运行 No - 服务器没有运行 <i>Note:</i> 为了确保Web前端知道服务器正在运行, 服务器上必须至少有一个Trapper进程 (<code>zabbix_server.conf</code> 文件中的 <code>StartTrappers</code> 参数)	Zabbix服务器的位置和端口。
主机数 量	显示配置的主机总数。\\模板也算作主 机类型。	受监控主机数量/未监控主机/模板数。
监控项 的个数	显示监控项总数, 仅指的是分配来启用 主机的项目。	受监控/禁用/不受支持的项目数。
触发数 量	显示触发总数。 只有分配给启用的主 机和根据启用的项目的触发计数。	启用/禁用触发器。 [触发问题/ ok状态.]
用户数 量	显示配置的用户总数。	N在线人数。
所需 的服 务器 性 能, 每秒新 的值	显示Zabbix服务器每秒处理的新值的 预期数量。	所需服务器性能 一个估计值, 可以作为指导。要精确处 理的数值, 请使用' <code>zabbix [wcache, values, all]`'\\计算中包含受监控主机的启用项目。日志项目 被计为每个项目更新间隔的一个值. 定期间隔值被计数; 灵活和调度间隔值不是。“节点”维护期间的计算未进行 调整。 trapper项目不计算在内。</code>

2 可用性报告 |

概述

在报告 -> 可用性报告中，您可以看到每个触发器在问题/状态中的时间比例。 显示每个状态的时间百分比。

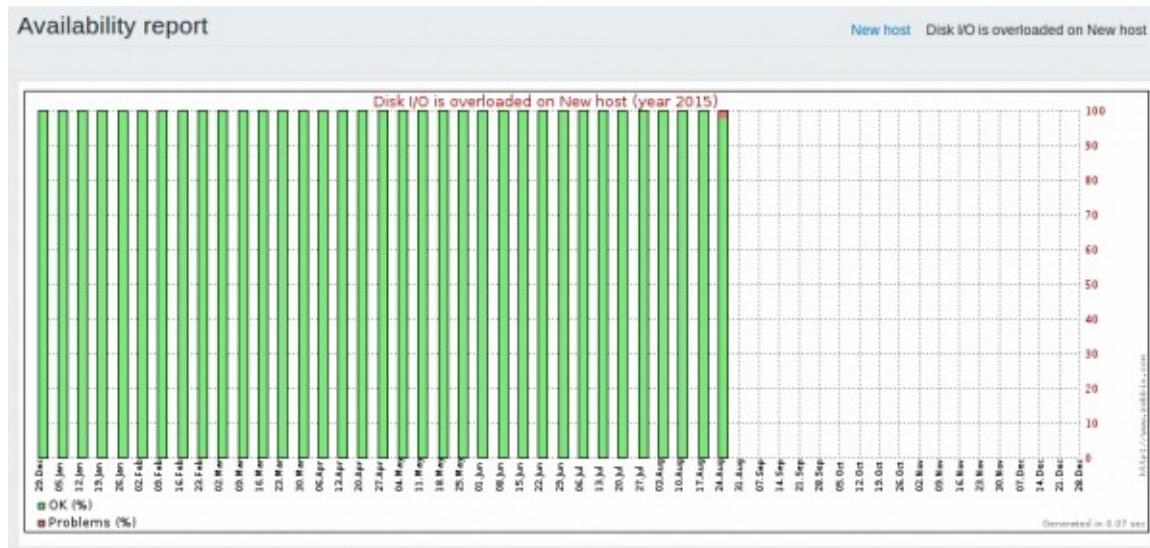
因此，很容易确定系统上各种元素的可用性情况。

NAME	PROBLEMS	OK	GRAPH
/etc/passwd has been changed on Zabbix server	100.0000%	Show	
Configured max number of opened files is too low on Zabbix server	100.0000%	Show	
Configured max number of processes is too low on Zabbix server	100.0000%	Show	
Disk I/O is overloaded on Zabbix server	100.0000%	Show	
Free disk space is less than 20%	100.0000%	Show	
Free disk space is less than 20% on volume /	100.0000%	Show	
Free inodes is less than 20% on volume /	100.0000%	Show	
Host information was changed on Zabbix server	100.0000%	Show	
Host name of zabbix-agentd was changed on Zabbix server	100.0000%	Show	
Hostname was changed on Zabbix server	100.0000%	Show	
Lack of available memory on server Zabbix server	100.0000%	Show	
Lack of free swap space on Zabbix server	100.0000%	Show	

从右上角的下拉菜单中，您可以选择选择模式 - 是否显示主机或属于模板的触发器。然后在过滤器中，您可以将选择范围缩小到所需的选项和时间段。

HOST	NAME	PROBLEMS	OK	GRAPH
New host	Disk I/O is overloaded on New host	0.3472%	99.6528%	Show
Zabbix server	Disk I/O is overloaded on Zabbix server	100.0000%		Show

触发器的名称是指向该触发器的最新事件的链接。点击“图形”列中的显示显示一个条形图，其中可用性信息以条形格式显示，表示当前年份过去一周的每个条。



绿色部分代表OK时间，红色表示问题时间。

3 触发器前100

概述

在报表中 - >触发器前100 您可以看到在评估期间最多次更改其状态的触发器，按状态更改次数排序。

100 busiest triggers				
Filter ▲ Host groups <input type="text" value="type here to search"/> <input type="button" value="Select"/> From <input type="text" value="2015"/> - <input type="text" value="01"/> - <input type="text" value="01"/> <input type="text" value="00"/> : <input type="text" value="00"/> <input type="button" value="..."/> Hosts <input type="text" value="type here to search"/> <input type="button" value="Select"/> Till <input type="text" value="2016"/> - <input type="text" value="01"/> - <input type="text" value="01"/> <input type="text" value="00"/> : <input type="text" value="00"/> <input type="button" value="..."/> Severity <input type="checkbox"/> Not classified <input checked="" type="checkbox"/> Warning <input checked="" type="checkbox"/> High <input type="checkbox"/> Information <input checked="" type="checkbox"/> Average <input checked="" type="checkbox"/> Disaster Today Yesterday Current week Current month Current year Last week Last month Last year				
HOST	trigger	severity	number of status changes	
New host	Disk I/O is overloaded on New host	Warning	75	
Zabbix server 1	Zabbix discoverer processes more than 75% busy	Average	71	
New host	Too many processes on New host	Warning	23	
Zabbix server 1	Disk I/O is overloaded on Zabbix server 1	Warning	11	
Zabbix server 1	Zabbix http poller processes more than 75% busy	Average	5	
New host	/etc/passwd has been changed on New host	Warning	3	
New host	CPU load too high on 'New host' for two minutes	High	3	
New host	Processor load is over 2 on New host	Warning	3	
Zabbix server 1	/etc/passwd has been changed on Zabbix server 1	Warning	1	
Zabbix server 1	Free disk spa	Trigger		
	Events			
	Configuration			
New host	Free disk spa	History		
New host	Free disk spa			
Zabbix server 1	Free disk spa			
	Checksum of /etc/passwd			

主机和触发器列条目

都是提供一些有用选项的链接： 对主机来说 - 链接到用户定义的脚本，最新数据，库存，主图的图形和屏幕 对触发器来说 - 链接到最新的事件，触发器配置表单和一个简单的图表 使用过滤器您可以使用过滤器来显示主机组，主机，触发器严重性，预定义时间段或自定义时间段的触发器。指定父主机组隐含地选择所有嵌套的主机组。

4 审计

概述

在 报告 ->审核 部分，用户可以查看在前端所做更改的记录。

审计日志

在此屏幕中，可以看到在前端进行的各种更改的审核日志。你可以使用过滤器，位于审计日志 栏之下， 缩小用户，活动类型，受影响资源和时间段的记录。

The screenshot shows the 'Audit log' interface with the following details:

- Filter Bar:** Includes fields for User (with a 'Select' button), Action (set to 'Update'), and Resource (set to 'All'). Buttons for 'Filter' and 'Reset' are also present.
- Time Range:** Set to '2015-10-04 11:14 - 2015-11-03 11:14 (now)'.
- Zoom Options:** Includes links for 5m, 15m, 30m, 1h, 2h, 3h, 6h, 12h, 1d, 3d, 7d, 14d, 1m, 3m, All, and a date range selector from ** 1m to ** 1m.
- Table Headers:** TIME, USER, IP, RESOURCE, ACTION, ID, DESCRIPTION, DETAILS.
- Data Rows:**
 - 2015-11-03 10:47:21 Admin 192.168.3.31 Map Updated 0 Name [Network]
 - 2015-11-03 10:46:43 Admin 192.168.3.31 Map Updated 0 Name [Network]
 - 2015-11-03 10:40:55 Admin 192.168.3.31 Map Updated 0 Name [Network]
 - 2015-10-22 10:52:45 Admin 192.168.3.31 Trigger Updated 13605 Lack of free swap space on {HOSTNAME}.
Comments: It probably means that the system requires more physical memory.
http://www.forensicswiki.org/wiki/Physical_memory => It probably means that the system requires more physical memory.

显示数据：

列	描述
Time	审计记录的时间戳 .
User	活动用户。
IP	在活动中使用的IP。
Resource	将显示受影响的资源。
Action	活动类型显示 - 登录, 注销, 添加, 更新, 删除, 启用 或者 禁止 .
ID	显示受影响资源的ID。
Description	显示资源的描述。
Details	显示执行活动的详细信息。

5 动作日志

概述

在此屏幕中显示操作中执行的操作（通知，远程命令）的详细信息。

你可以使用过滤器，位于动作日志栏之下，to narrow down the records by recipient of e-mail and time period. 通过电子邮件和时间段收件人缩小记录。

The screenshot shows the 'Action log' screen in Zabbix. At the top, there is a search bar labeled 'Recipient' with a 'Select' button, and a date range selector from '2018-04-29 14:24' to '2018-05-31 14:24 [now]'. Below this are filters for time intervals (Zoom: 5m, 15m, 30m, 1h, 2h, 6h, 12h, 1d, 3d, 7d, 14d, 1m, 3m, 6m, All) and a navigation bar with arrows and page numbers (1m, 2d, fixed).

Time	Action	Type	Recipient	Message	Status	Info
2018-05-23 11:35:58	Report problems to Zabbix administrators	Email	Admin (Zabbix Administrator) Martins.Valkovskis@zabbix.com	Subject: Problem: My host has just been restarted Message: Problem started at 11:35:56 on 2018-05-23 Problem name: My host has just been restarted Host: My host Severity: Information Original problem ID: 126377	Sent	
2018-05-23 11:35:37	Report problems to Zabbix administrators	Email	Admin (Zabbix Administrator) Martins.Valkovskis@zabbix.com	Subject: Resolved: Too many processes on My host Message: Problem has been resolved at 11:35:35 on 2018-05-23 Problem name: Too many processes on My host Host: My host Severity: Warning Original problem ID: 126311	Sent	

显示数据：

列	描述
<i>Time</i>	操作时间戳
<i>Action</i>	显示导致operation的action名称。Zabbix 2.4.0之后的版本可以显示动作名称。
<i>Type</i>	显示操作类型 - 邮件 或者 命令 .
<i>Recipient(s)</i>	显示用户别名，姓名（括号中）和通知收件人的电子邮件地址。Zabbix 2.4.0之后的版本可以显示显示用户别名，姓名
<i>Message</i>	将显示消息/远程命令的内容。
<i>Status</i>	显示操作状态：进行中 - actions正在进行中\对于正在进行的actions，显示剩余的重试次数 - 服务器将尝试发送通知的剩余次数。已发送 - 通知已发送已执行 - 命令已执行未发送 - action并未结束
<i>Info</i>	显示有关操作执行的错误信息（如果有）。

6 通知

概述

在 报告 ->通知 栏中， 将显示发送给每个用户的通知数量的报告。

从右上角的下拉菜单中，您可以选择媒体类型（或全部），周期（每天/周/月/年的数据）和发送通知的年份。

Notifications		Media type	all	Period	Daily	Year	2016
DAY		ADMIN (ZABBIX ADMINISTRATOR)		GUEST		USER (NEW USER)	
2016-01-01							
2016-01-02		6 (6/0/0/0)					
2016-01-03		2 (2/0/0/0)					
2016-01-04		10 (10/0/0/0)					
2016-01-05		24 (24/0/0/0)					
2016-01-06		10 (10/0/0/0)					
2016-01-07		6 (6/0/0/0)					
2016-01-08		4 (4/0/0/0)					

每列显示每个系统用户的总计数。

4 配置

概述

配置菜单包含用于设置主要Zabbix功能的部分，例如主机和主机组，数据收集，数据阈值，发送问题通知，创建数据可视化等。

1 主机组

概述

在 配置 -> 主机组菜单下，用户可以配置和维护主机组。 主机组可以同时包含模板和主机。

有一个现有主机组及其详细信息的列表。 您可以按名称搜索和过滤主机组。

	Name	Hosts	Templates	Members	Info
<input type="checkbox"/>	Zabbix servers	Hosts 1	Templates	Zabbix server 1	
<input type="checkbox"/>	Windows servers	Hosts 1	Templates	Win server 2008	
<input type="checkbox"/>	Web servers	Hosts 1	Templates	Apache	
<input type="checkbox"/>	Virtual machines	Hosts 1	Templates	VMware	
<input type="checkbox"/>	UPS devices	Hosts	Templates		
New template, ODBC discovery, Template1, Template2, Template App FTP Service, Template App HTTP Service, Template App HTTPS Service, Template App IMAP Service, Template App LDAP Service, Template App MySQL, Template App NNTT Service, Template App NTP Service, Template App POP Service, Template App SMTP Service, Template App SSH Service, Template App Telnet Service, Template App Zabbix Agent, Template App Zabbix Proxy, Template App Zabbix Server, Template ICMP Ping, Template IPMI Intel SR1530, Template IPMI Intel SR1630, Template JMX Generic, Template JMX Tomcat, Template OS AIX, Template OS FreeBSD, Template OS HP-UX, Template OS Linux, Template OS Linux_b, Template OS Mac OS X, Template OS OpenBSD, Template OS Solaris, Template OS Windows, Template SNMP Device, Template SNMP Disks, Template SNMP Generic, Template SNMP Interfaces, Template SNMP Interfaces_Orig, Template SNMP OS Linux, Template SNMP OS Windows, Template SNMP Processors, Template Virt VMware, Template Virt VMware Guest, Template Virt VMware Hypervisor					
<input type="checkbox"/>	Templates	Hosts	Templates 44		
<input type="checkbox"/>	SNMP hosts	Hosts 2	Templates	Switch1, Switch2	

显示数据：

列	描述
Name	主机组的名称。点击组名打开主机组。 配置表单 。
Hosts	主机组中主机的数量（以灰色显示）。在主机的整个列表中，单击“主机”将过滤掉属于该组的那些。
Templates	组中的模板数（以灰色显示）。在模板的整个列表中，单击“模板”将过滤掉属于该组的那些。
Members	组构成的名称，模板名称以灰色显示，受监视的主机名以蓝色和非监视主机名称显示为红色。单击名称将打开模板/主机配置表单。
Info	显示有关主机组的错误信息（如果有）。

批量编辑选项

列表下方的按钮提供了一些批量编辑选项：

- *Enable hosts* - 将组中所有主机的状态更改为“监视”
- *Disable hosts* - 将组中所有主机的状态更改为“未监视”
- *Delete* - 删除主机组

要使用这些选项，请在相应的主机组之前标记复选框，然后单击所需的按钮。

2 模板

概述

在配置 ->模板部分，用户可以配置和维护模板。

显示现有模板及其详细信息的列表如下：

Templates								Group	all	Create template	Import
<input type="button" value="Filter"/> <input type="text" value="Name like"/> <input type="button" value="Filter"/> <input type="button" value="Reset"/>											
	Templates	Applications	Items	Triggers	Graphs	Screens	Discovery	Web	Linked templates	Linked to	
<input type="checkbox"/>	Template Virt VMware Hypervisor	Applications 6	Items 10	Triggers 3	Graphs 2	Screens 2	Discovery 5	Web			
<input type="checkbox"/>	Template Virt VMware Guest	Applications 8	Items 17	Triggers 3	Graphs 2	Screens 2	Discovery 5	Web			
<input type="checkbox"/>	Template Virt VMware	Applications 3	Items 3	Triggers 2	Graphs 2	Screens 2	Discovery 3	Web			
<input type="checkbox"/>	Template SNMP Processors	Applications 1	Items 1	Triggers 1	Graphs 1	Screens 1	Discovery 1	Web		Template SNMP OS Linux, Template SNMP OS Windows	
<input type="checkbox"/>	Template SNMP OS Windows	Applications 4	Items 6	Triggers 2	Graphs 2	Screens 2	Discovery 3	Web		Template SNMP Disks, Template SNMP Generic, Template SNMP Interfaces_Orig, Template SNMP Processors	
<input type="checkbox"/>	Template SNMP OS Linux	Applications 4	Items 6	Triggers 2	Graphs 2	Screens 2	Discovery 3	Web		Template SNMP Disks, Template SNMP Generic, Template SNMP Interfaces_Orig, Template SNMP Processors	
<input type="checkbox"/>	Template SNMP Interfaces_Orig	Applications 1	Items 1	Triggers 1	Graphs 1	Screens 1	Discovery 1	Web		Switch1, Template SNMP Device, Template SNMP OS Linux, Template SNMP OS Windows	

从标题栏中的右侧的下拉列表中，您可以选择是显示所有模板还是仅显示属于组的模板。您也可以按名称搜索和过滤模板。

显示数据：

列	描述
模板	模板名称，单击模板名称打开模板 配置表单 。
实体（应用程序，项目，触发器，图形，屏幕，发现，Web）	模板中各个实体的数量（以灰色显示）。单击实体名称将在该实体的整个列表中过滤掉属于该模板的那些实体。
链接的模板	链接到特定模板的模板，在嵌套设置中，模板将共享所链接模板的所有实体。
链接到 模板链接到的主机和模板 .	

点击右上角创建模板 按钮配置新的模板。点击右上角导入按钮，从XML 文件导入模板。

批量编辑选项

列表下方的按钮提供了一些批量编辑选项：

- *Export* - 将模板导出到XML文件
- *Delete* - 删除模板，将其链接的实体（项目，触发器等）与主机保持同步
- *Delete and clear* - 从主机中删除模板及其链接的实体

要使用这些选项，请在相应模板之前标记复选框，然后单击所需的按钮。

3 主机

概览

在配置 -> 主机中，用户可以配置和维护主机。

有一个显示现有主机及其详细信息的列表。

从右边的下拉菜单中有 主机栏，您可以选择是显示所有主机还是仅显示属于一个特定组的主机。

Hosts											Group	all	Create host	Import
	NAME	APPLICATIONS	ITEMS	TRIGGERS	GRAPHS	DISCOVERY	WEB	INTERFACE	TEMPLATES	STATUS	AVAILABILITY	INFO	AGENT	ENCRYPTION
<input type="checkbox"/>	Zabbix server	Applications 12	Items 71	Triggers 44	Graphs 12	Discovery 2	Web 1	192.168.3.194:10050	Template1 (Template) Template App Zabbix Server, Template OS Linux (Template App Zabbix Agent)	Enabled	ZBX SNMP JMX IPMI	NONE		
<input type="checkbox"/>	procure.zabbix.lan	Applications 1	Items 1	Triggers	Graphs	Discovery 1	Web	192.168.3.7:161	Template SNMP Interfaces	Enabled	ZBX SNMP JMX IPMI	NONE		
<input type="checkbox"/>	New host	Applications 10	Items 42	Triggers 18	Graphs 9	Discovery 2	Web	192.168.3.31:32050	Template OS Linux (Template App Zabbix Agent)	Enabled	ZBX SNMP JMX IPMI	NONE		

显示数据：

列	描述
名称	主机名称，单击主机名打开主机配置表。
元素（应用程序，监控项，触发器，图形，发现，Web）	单击元素名称将显示主机的项目，触发器等。各个元素的数量以灰色显示。
界面	显示主机的主界面。
模板	显示与主机链接的模板。如果链接的模板中包含其他模板，那么它们将显示在括号中，以逗号分隔。单击模板名称将打开其配置表单。
状态	显示主机状态 - 启用 或是 禁止。点击状态可以更改。
可用性	显示主机的可用性。四个图标各自表示支持的接口（Zabbix代理，SNMP，IPMI，JMX）。界面的当前状态由相应的颜色显示：绿色 - 可用红色 - 不可用（在鼠标悬停时，显示无法访问接口的原因的详细信息）灰色 - 知或未配置
代理加密	显示与主机连接的加密状态：None - 没有加密PSK - 使用预共享密钥Cert - 使用证书
Info	显示有关主机的错误信息（如果有）。

点击右上角创建主机 按钮配置一个新的主机。点击右上角 导入 按钮从XML文件导入主机。

批量编辑选项

列表下方的按钮提供了一些批量编辑选项：

- *Enable* - 将主机状态更改为 已监控
- *Disable* - 将主机状态更改为 未被监控

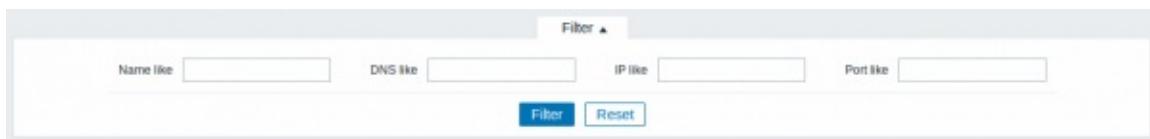
- *Export* - 将主机导出为XML文件
- *Mass update* - 多个主机[多个属性更新。](#)
- *Delete* - 删除主机

要使用这些选项，请在相应的主机之前标记复选框，然后单击所需的按钮。

过滤器

由于该列表可能包含很多主机，可能需要过滤出您真正需要的主机。

[过滤器](#) 链接在主机列表之上。如果您点击它，则可以使用过滤器，您可以通过名称，DNS，IP或端口号过滤主机。



阅读主机可用性

主机可用性图标反映了Zabbix服务器上的当前主机接口状态。因此，在前台：

- 如果禁用主机，可用性图标将不会立即变为灰色（未知状态），因为服务器必须首先同步配置更改；
- 如果启用主机，则可用性图标将不会立即变为绿色（可用），因为服务器必须同步配置更改并开始首先轮询主机。

未知主机状态

Zabbix服务器将主机可用性图标设置为相应代理接口（Zabbix，SNMP，IMP，JMX）的灰色（未知状态），如果：

- 界面上没有启用的项目（它们被删除或禁用）；
- 主机已禁用；
- 主机被设置为由代理监视，不同的代理或由服务器监视，如果它被代理监视；
- 主机由看起来处于脱机状态的代理进行监控（在最大心跳间隔（1小时）内没有从代理收到更新）。
-

请参阅有关主机的更多细节[unreachability](#)。

1 应用

概览

可以从配置 ->模板中访问模板的应用程序列表，然后单击相应模板的应用程序。

可以在 配置 ->主机中访问主机的应用程序列表，然后单击相应主机的应用程序。

显示现有应用程序的列表：

The screenshot shows the 'Applications' page in the Zabbix interface. At the top, there are dropdown menus for 'Group' (set to 'all') and 'Host' (set to 'Zabbix server'), and a 'Create application' button. Below these are navigation links: 'All hosts / Zabbix server', 'Enabled', 'ZBX', 'SNMP', 'JMX', 'IPMI', 'Applications 12', 'Items 71', 'Triggers 44', 'Graphs 12', 'Discovery rules 2', and 'Web scenarios 1'. The main content area displays a table of applications:

APPLICATION	ITEMS	INFO
Template OS Linux: CPU	Items 13	
Template OS Linux: Filesystems	Items 5	
Template OS Linux: General	Items 5	
Template OS Linux: Memory	Items 5	
Template OS Linux: Network interfaces	Items 2	
Template OS Linux: OS	Items 8	
Template OS Linux: Performance	Items 13	
Template OS Linux: Processes	Items 2	
Template OS Linux: Security	Items 2	
Template App Zabbix Agent: Zabbix agent	Items 3	
Zabbix frontend	Items	
Template App Zabbix Server: Zabbix server	Items 30	

显示数据：

列	描述
应用	应用的名称，显示为直接创建的应用程序的蓝色链接。单击应用程序名称链接将打开该应用程序配置表。如果主机应用程序属于模板，则模板名称将以灰色链接的形式显示在应用程序名称之前。单击模板链接将打开模板级别的应用程序列表。
项	单击项目以查看应用程序中包含的项目。项目数量显示为灰色。
Info	显示有关应用程序的错误信息（如果有）。

点击 右上角创建应用程序按钮配置新的应用程序。

批量编辑选项

列表下方的按钮提供了一些批量编辑选项：

- *Enable* - 将应用状态更改为 启用
- *Disable* - 将应用状态更改为 禁用
- *Delete* - 删除应用

要使用这些选项，请在各个应用程序之前标记复选框，然后单击所需的按钮。

2 Items项

概览

可以从配置 ->模板中访问模板的项目列表，然后单击相应模板的项目。

在配置 ->主机中可以访问主机的项目列表，然后单击相应主机的项目。

现有项目的列表：

Items									Create item					
		All hosts / Zabbix server 1	Enabled	Zabbix	SNMP	IMIX	IPMI	Applications 12	Items 77	Triggers 44	Graphs 12	Discovery rules 3	Web scenarios 1	Filter ▾
Wizard	Name	Triggers	Key	Interval	History	Trends	Type	Applications	Status	Info				
<input type="checkbox"/>	Template App Zabbix Agent: Host name of zabbix-agent is running	Triggers 1	agent.hostname	1h	7d	365d	Zabbix agent	Zabbix agent	Enabled					
<input type="checkbox"/>	Template App Zabbix Agent: Version of zabbix-agent(d) is running	Triggers 1	agent.version	1h	7d	365d	Zabbix agent	Zabbix agent	Enabled					
<input type="checkbox"/>	Template OS Linux: Maximum number of opened files	Triggers 1	kernel.maxfiles	1h	7d	365d	Zabbix agent	OS	Enabled					
<input type="checkbox"/>	Template OS Linux: Maximum number of processes	Triggers 1	kernel.maxproc	1h	7d	365d	Zabbix agent	OS	Enabled					
<input type="checkbox"/>	Network interface discovery: Incoming network traffic on enp0s3		netif.in[enp0s3]	1m	7d	365d	Zabbix agent	Network interfaces	Enabled					
<input type="checkbox"/>	Network interface discovery: Outgoing network traffic on enp0s3		netif.out[enp0s3]	1m	7d	365d	Zabbix agent	Network interfaces	Enabled					
<input type="checkbox"/>	Template OS Linux: Number of running processes	Triggers 1	proc.num[_run]	1m	7d	365d	Zabbix agent	Processes	Enabled					
<input type="checkbox"/>	Template OS Linux: Number of processes	Triggers 1	proc.num[0]	1m	7d	365d	Zabbix agent	Processes	Enabled					
<input type="checkbox"/>	Template OS Linux: Host boottime		system.boottime	10m	7d	365d	Zabbix agent	General, OS	Enabled					

显示数据：

列	描述
Wizard	向导图标是指向向导的链接，用于根据项目创建触发器。
Name	项目的名称，显示为项目详细信息的蓝色链接。单击项目名称链接将打开该项目 配置表 。如果主机项目属于模板，模板名称将显示在项目名称之前，作为灰色链接。单击模板链接将打开模板级别的项目列表。如果项目是从项目原型创建的，则其名称前面是低级别的发现规则名称，以橙色显示。单击发现规则名称将打开项目原型列表。
触发器	将鼠标移动到触发器上将显示一个信息框，显示与该项目相关联的触发器。触发器的数量以灰色显示。
Key	显示项目键。
间隔	显示检查频率。
历史	将显示项目数据记录将保留多少天。
趋势	将显示将保留多少天的项目趋势记录。
类型	显示项目类型（Zabbix代理，SNMP代理，简单检查等）。
应用程序	项目应用程序显示。
状态	显示项目状态 - 启用，禁用 或者 不支持。通过点击状态，您可以更改它 - 从启用到禁用（反之亦然）；从不支持到禁用（反之亦然）。
Info	如果一切正常，此列中不会显示图标。如果有错误，将显示带有十字架的红色方形图标。将鼠标移动到图标上方，您将看到带有错误描述的工具提示。

点击右上角 **创建项目** 按钮配置新项目。

批量编辑选项

列表下方的按钮提供了一些批量编辑选项：

- *Enable* - 将项目状态更改为 启用
- *Disable* - 将项目状态更改为 禁用
- *Clear history* - 删除项目的历史记录和趋势数据
- *Copy* - 将项目复制到其他主机或模板
- *Mass update* - 多项目多属性 同时升级
- *Delete* - 删除项

要使用这些选项，请在相应项目之前标记复选框，然后单击所需的按钮。

过滤器

由于列表可能包含很多项目，可能需要过滤出您真正需要的项目。

过滤器 链接在列表上方可用。 如果您点击它，则可以使用过滤器，您可以通过多个属性过滤项目。

The screenshot shows the Zabbix item list interface with a detailed filter configuration. At the top, there's a 'Filter' button and several dropdown menus for filtering by Host group, Host, Application, Name like, Key like, Type, Type of information, State, Status, Triggers, and Template. Below the filter bar, there are several sections: APPLICATIONS (CPU, Filesystems, General, Memory, Network Interfaces, OS, Performance, Processes, Security, Zabbix agent, Zabbix server), TYPES (Zabbix agent, Zabbix internal, Zabbix trapper), STATUS (Enabled, Disabled), STATE (Normal, Not supported), TEMPLATE (Not templated items, Templated items), WITH TRIGGERS (Without triggers, With triggers, the latter is selected), HISTORY (Interval: 10, 60, 300, 600, 3600), and INTERVAL (Value: 60). The 'With triggers' section is highlighted.

在过滤器下方的子滤波器 below the filter 提供进一步的过滤选项（已经过虑的数据）。 您可以选择具有公共参数值的项目组。如果单击一个组，它将突出显示，只有具有此参数值的项目保留在列表中。

3 触发器

概述

可以从 配置 -> 模板中访问模板的触发器列表，然后单击相应模板的触发器。

可以从配置 -> 主机访问主机的触发器列表，然后单击相应主机的触发器。

Triggers					
	Severity	Name	Expression	Status	Info
<input type="checkbox"/>	Warning	Template OS Linux: Jetpassword has been changed on [HOSTNAME]	[Zabbix server]!{file cksum[etc/pwdfile.dif][0]}>0	Enabled	
<input type="checkbox"/>	Information	Template OS Linux: Configured max number of opened files is too low on [HOSTNAME]	[Zabbix server]!{kernel.maxfiles.last[0]}<3024	Enabled	
<input type="checkbox"/>	Information	Template OS Linux: Configured max number of processes is too low on [HOSTNAME]	[Zabbix server]!{kernel.maxproc.last[0]}<256	Enabled	
<input type="checkbox"/>	Warning	Template OS Linux: Disk I/O is overloaded on [HOSTNAME]	[Zabbix server]!{system.cpu.util[jowait].avg5m}>20	Enabled	
<input type="checkbox"/>	Warning	Mounted filesystem discovery: Free disk space is less than 20% on volume /	[Zabbix server]!{fs.size[pfree].last[0]}<20	Enabled	
<input type="checkbox"/>	Warning	Mounted filesystem discovery: Free inodes is less than 20% on volume /	[Zabbix server]!{fs.inode[pfree].last[0]}<20	Enabled	
<input type="checkbox"/>	Information	Template OS Linux: Host information was changed on [HOSTNAME]	[Zabbix server]!{system.hostname.diff[0]}>0	Enabled	
<input type="checkbox"/>	Information	Template App Zabbix Agent: Host name of zabbix-agent was changed on [HOSTNAME]	[Zabbix server]!{agent.hostname.diff[0]}>0	Enabled	
<input type="checkbox"/>	Information	Template OS Linux: Hostname was changed on [HOSTNAME]	[Zabbix server]!{system.hostname.diff[0]}>0	Enabled	
<input type="checkbox"/>	Average	Template OS Linux: Lack of available memory on server [HOSTNAME]	[Zabbix server]!{vm.memory.size[available].last[0]}<20M	Enabled	

显示数据：

列	描述
<i>Severity</i>	触发器的严重性由名称和单元格背景颜色显示。
<i>Name</i>	触发器的名称，显示为蓝色链接以触发细节。单击触发器名称链接将打开触发器配置表。如果主机触发器属于模板，则模板名称将在触发器名称之前显示为灰色链接。单击模板链接将打开模板级别的触发器列表。如果触发器是从触发器原型创建的，则其名称前面是低级别的发现规则名称，以橙色显示。单击发现规则名称将打开触发器原型列表。
<i>Expression</i>	显示触发表达式。表达式的host-item部分显示为链接，链接到项目配置表单。
<i>Status</i>	显示触发状态 - 启用，禁用 或者未知。通过点击状态，您可以更改它 - 从启用到禁用（反之亦可）；从未知到已禁用（反之亦可）。
<i>Info</i>	如果一切正常，此列中不会显示图标。如果有错误，将显示带有十字架的红色方形图标。将鼠标移动到图标上方，您将看到带有错误描述的工具提示。

点击右上角创建触发器配置新的触发器。

批量编辑选项

列表下方的按钮提供了一些批量编辑选项：

- *Enable* - 将触发状态更改为 启用
- *Disable* - 将触发状态更改为 禁用
- *Copy* - 将触发器复制到其他主机或模板

- *Mass update* - 一次更新多个触发器的几个属性
- *Delete* - 删除触发器

要使用这些选项，请在相应的触发器之前标记复选框，然后单击所需的按钮。

4 图

概述

可以从 Configuration → Templates (配置→模板) 访问模板的自定义图列表，然后单击相应模板的 Graphs (图)。

可以从 Configuration → Hosts (配置→主机) 访问主机的自定义图列表，然后单击相应主机的图。

显示现有图的列表。.

Graphs			
	Group	Host	Create graph
All hosts / Zabbix server 1	Enabled	ZBX	SNMP JMX IPMI
	Applications	Items	Triggers
	Graphs	Discovery rules	Web scenarios
<input type="checkbox"/> Name		Width	Height
<input type="checkbox"/> Template OS Linux: CPU jumps		900	200
<input type="checkbox"/> Template OS Linux: CPU load		900	200
<input type="checkbox"/> Template OS Linux: CPU utilization		900	200
<input type="checkbox"/> Mounted filesystem discovery: Disk space usage /		600	340
			Pie

显示数据：

列	描述
Name	自定义图的名称，显示图细节的蓝色链接。点击图名的链接来打开图 configuration form。如果主机图属于模板，则模板名称将在图名称之前，以灰色链接显示。单击模板链接打开模板级的图列表。如果图是从图原型创建的，则其名称前面是低级别发现规则名，并以橙色显示。单击发现规则名将打开图原型列表
Width	图显示的宽度
Height	图显示的长度
Graph type	图显示的类型 - Normal, (正常图形), Stacked, (叠加图形), Pie (饼状图形) 或者 Exploded. (分解饼状图形) .

配置新的图，可以点击顶部右上角的 Create graph 按钮。

批量编辑选项

列表下面的按键会提供一些批量编辑选项：

- *Copy* - 将图复制到其他主机或模板上。
- *Delete* - 删除图

要使用这些选项，请在各个图之前标记复选框，然后单击所需的按钮

5 发现规则

概述

从 Configuration → Templates（配置→模板）访问模板的低级别发现规则，随后点击相应模板的Discovery（发现）。

从 Configuration → Hosts（配置→ 主机）访问主机的低级别发现规则列表，随后点击相应主机的Discovery（发现）。

显示现有的低级别发现规则列表：

Discovery rules							
All hosts / Zabbix server Enabled		ITEMS	TRIGGERS	GRAPHS	HOSTS	KEY	INTERVAL
	NAME	ITEMS	TRIGGERS	GRAPHS	HOSTS	KEY	INTERVAL
<input type="checkbox"/>	Template OS Linux: Mounted filesystem discovery	Item prototypes 5	Trigger prototypes 2	Graph prototypes 1	Host prototypes	vfs.fs.discovery	1h
<input type="checkbox"/>	Template OS Linux: Network interface discovery	Item prototypes 2	Trigger prototypes	Graph prototypes 1	Host prototypes	netif.discovery	1h

显示数据：

列	描述
Name	规则名，用蓝色链接来显示。点击规则名打开低级别发现规则。 configuration form . 如果发现规则属于模板，模板名将以灰色链接，显示在规则名前面。点击模板链接将会在模板级打开规则列表。
Items	显示监控项原型列表的链接现有监控项原型的数量用灰色来显示。
Triggers	显示触发器原型列表的链接。现有触发器原型的数量用灰色来显示。
Graphs	显示图原型列表的链接。现有图原型的数量用灰色来显示。
Hosts	显示主机原型列表的链接。现有主机原型的数量用灰色来显示。
Key	显示用于发现的监控项值。
Interval	显示执行发现的频率。
Type	显示用于发现的监控项类型 (Zabbix agent, SNMP agent, 等).
Status	D显示发现规则的状态 - Enabled, (可用) Disabled (不可用) or Not supported. (不支持) 通过点击状态可以改变它。从可用到不可用(反之亦然)；从不支持到不可用(反之亦然)
Info	如果一切顺利，没有任何图标显示在这一列。但要有错误的话，如果有错误，将显示带有十字的红色方形图标。 将鼠标移动到图标上方，您将看到带有错误描述的提示

配置新的低等级发现规则，可以点击顶部右上角的 Create discovery rule 按钮。and corner.

批量编辑选项

列表下面的按键会提供一些批量编辑选项：

- *Enable* - 将低等级发现规则的状态改为 Enabled 可用
- *Disable* - 将低等级发现规则的状态改为 Disabled 禁用。

- *Delete* - 删除低等级发现规则。

要使用这些选项，请在各个发现规则之前标记复选框，然后单击所需的按钮

6 Web场景

概述

从 Configuration → Templates （配置→ 模板）访问模板的web场景列表，随后点击相应模板的Web。

从 Configuration → Hosts （配置→ 主机）访问主机的web场景列表，随后点击相应主机的Web。

显示现有的web场景。从 Scenarios 栏中的右下方的下拉列表中，您可以选择是显示所有Web场景或仅显示属于一个特定组和主机的场景。此外，您可以选择隐藏已禁用的方案（或再次显示），方法是单击相应的链接。

NAME	NUMBER OF STEPS	UPDATE INTERVAL	ATTEMPTS	AUTHENTICATION	HTTP PROXY	APPLICATION	STATUS	INFO
Zabbix frontend	5	3m	1	None	No	Zabbix frontend	Enabled	

显示数据：

列	描述
Name	Web场景名称。点击web场景名称来打开web场景。 configuration form .
Number of steps	场景里包含的步骤数。
Update interval	场景多久执行一次。
Attempts	执行web场景步骤的尝试有多少次
Authentication	显示验证方法- Basic, NTLM or None.
HTTP proxy	显示 HTTP proxy 或者在不应用的情况下选择 'No'。
Application	Web场景应用已被显示
Status	

通过点击状态来改变它。

Info	如果一切顺利，没有任何图标显示在这一列。但要有错误的话，如果有错误，将显示带有十字的红色方形图标。将鼠标移动到图标上方，您将看到带有错误描述的提示
------	---

配置新的web场景，可以点击顶部右上角的 Create web scenario 按钮。

批量编辑选项

列表下面的按键会提供一些批量编辑选项：

- *Enable* - 改变场景的状态至 Enabled 可用
- *Disable* - 改变场景的状态至 Disabled 不可用
- *Clear history* - 为场景清楚历史和趋势数据。
- *Delete* - 删除web场景。

要使用这些选项，请在各个web场景之前标记复选框，然后单击所需的按钮

4 维护

概述

在 Configuration → Maintenance 里，用户可以为主机维护和配置维护时段。

现有的维护时段和其细节的列表展示从 Maintenance periods 中的右侧的下拉列表中，您可以选择是显示所有维护周期或是仅显示属于一个特定组的维护时段。

Maintenance periods					
Create maintenance period Group all Filter ▾					
	Name	Type	Active since	Active till	State
<input type="checkbox"/>	Weekly maintenance	With data collection	2015-01-01 00:00	2017-01-01 00:00	Active
<input type="checkbox"/>	One time	With data collection	2015-12-22 14:35	2016-03-01 00:00	Expired

Displaying 2 of 2 found

显示数据：

列	描述
Name	维护时段的名称。点击维护时段名称打开维护时段。 configuration form .
Type	显示维护时段的类型： With data collection 或 No data collection
Active since	执行维护时段的开始时间和数据。
Active till	执行维护时段的结束时间和数据
State	维护时段的状态： Approaching - 将会被激活。 Active - 已激活。 Expired - 不再激活
Description	显示维护时段的描述。

Name, Type, Active since 和 Active till 可以按照升序/降序的方式排列。为了排序，请点击列名。

To 配置新的维护时段，点击顶部右上角的 Create maintenance period 按钮。

批量编辑选项

列表下面的按键会提供一些批量编辑选项：

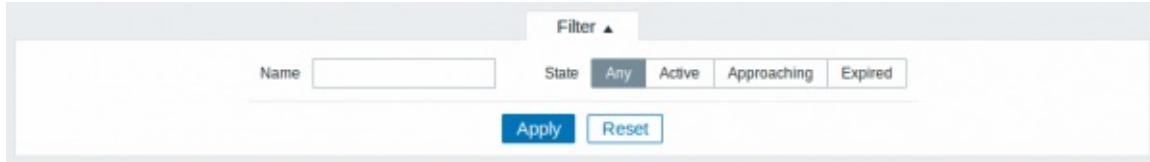
- **Delete** - 删除维护时段。

要使用这些选项，请在各个维护时段之前标记复选框，然后单击所需的按钮

过滤器

当一个列表包含多个维护时段时，可以使用过滤器功能找到真正您想要的。

Filter 链接在维护时段列表下可用。如果您点击它，过滤器就会可用，您可以用名称和状态进行过滤。



5 动作

概述

在 Configuration → Actions 里，用户可以维护和配置动作。显示的操作是分配给所选事件源（触发器，发现，自动注册）的操作。显示现有的动作和它们的描述。显示的操作是分配给所选事件源（触发器，发现，自动注册）的操作。现有的动作和其细节的列表展示。

要查看分配给不同事件源的操作，请将源从下拉菜单改到 Actions 栏中的右键。

显示数据：

列	描述
Name	动作名称。点击动作名称来打开动作。 configuration form .
Conditions	显示动作条件。
Operations	显示动作操作。从Zabbix 2.2开始，操作列表还显示通知收件人用于通知的媒介类型（电子邮件，短信，Jabber等）以及名字和姓氏（在别名之后的括号中）。
Status	显示动作状态。 - <i>Enabled</i> 或者 <i>Disabled</i> . 通过点击状态来修改它。参见 Escalations 获取更多细节，如在升级过程中，动作被禁用该怎么办。

配置新的动作，点击顶部右上角的 Create action 按钮。

批量编辑选项

列表下面的按键会提供一些批量编辑选项：

- *Enable* - 改变动作的状态至 *Enabled* 可用
- *Disable* - 改变动作的状态至 *Disabled* 不可用
- *Delete* - 删除动作

要使用这些选项，请在各个动作之前标记复选框，然后单击所需的按钮

过滤器

因为该列表可能包含多个动作，您可能需要过滤器来找到您真正需要的。

动作列表上面的 Filter (过滤器) 连接是可用的。如果您点击它，则可以使用过滤器，您可以通过名称和状态过滤操作。

Filter ▲

Name	<input type="text" value="admin"/>	Status	Any	Enabled	Disabled
<button>Apply</button> <button>Reset</button>					

6 事件关联

概述

在配置 ->事件关联中，用户可以配置和维护Zabbix事件的全局关联规则。

Name	Conditions	Operations	Status
<input type="checkbox"/> Close old event	New event tag Application = ABC Old event tag Application = ABC	Close old events	Enabled

Displaying 1 of 1 found

显示数据：

列	描述
Name	关联规则的名称。单击关联规则名称打开规则 配置表 。
Conditions	显示关联规则条件。
Operations	显示关联规则操作。
Status	显示关联规则状态 - 启用 或者禁用. 点击状态可以更改。

点击右上角 建立关联 配置新的关联规则。

批量编辑选项

列表下方的按钮提供了一些批量编辑选项：

- *Enable* - 将相关联状态更改为启用
- *Disable* - 将相关联状态更改为 禁用
- *Delete* - 删除关联规则

要使用这些选项，请在相应的关联规则之前标记复选框，然后单击所需的按钮。

过滤器

由于列表可能包含多个关联规则，可能需要过滤出您真正需要的那些。

过滤器 链接在相关规则列表上方可用。 如果您点击它，则可以使用过滤器，您可以通过名称和状态过滤关联规则。

7 发现

概述

在 配置 ->发现中用户可以配置和维护发现规则。

显示现有发现规则及其详细信息的列表。

The screenshot shows a table titled "Discovery rules". It has columns for Name, IP range, Delay, Checks, and Status. One row is visible: "Local network" with IP range 192.168.3.1-254, delay 1h, checks ICMP ping, SNMPv2 agent, Zabbix agent, and status Enabled. A blue button at the top right says "Create discovery rule". A filter dropdown is also present.

Discovery rules				
<input type="button" value="Create discovery rule"/> Filter ▾				
Name	IP range	Delay	Checks	Status
Local network	192.168.3.1-254	1h	ICMP ping, SNMPv2 agent, Zabbix agent	Enabled

Displaying 1 of 1 found.

显示数据：

列	描述
名称	发现规则的名称。 单击发现规则名称将打开发现规则 配置表 。
IP范围	显示用于网络扫描的IP地址范围。
延迟	显示执行发现的频率。
检查	显示用于发现的检查类型。
状态	显示动作状态 - 启用 或者 禁止.\点击状态可以更改。

点击右上角 创建发现规则 按钮配置新的发现规则。

批量编辑选项

列表下方的按钮提供了一些批量编辑选项：

- 启用 - 将发现规则状态更改为 用
- 禁用 - 将发现规则状态更改为 禁用
- 删除 - 删发现规则

要使用这些选项，请在相应的发现规则之前标记复选框，然后单击所需的按钮。

过滤器

由于列表可能包含许多发现规则，可能需要过滤出您真正需要的那些。

过滤器 链接在发现规则列表之上。 如果您点击它，则可以使用过滤器，您可以通过名称和状态过滤发现规则。

The screenshot shows a filter interface with a "Filter ▾" button. Below it are input fields for "Name" and "Status" (with options "Any", "Enabled", and "Disabled"). At the bottom are "Apply" and "Reset" buttons.

8 IT服务

概述

在配置 - > *IT服务* 中用户可以配置和维护 *IT服务* 层次结构。

第一次打开此部分时，它只包含一个 *root* 入口。

您可以将其用作构建受监视基础架构层次结构的起点。点击 *Add child* 添加服务，然后在您添加的服务下添加其他服务。

SERVICE	ACTION	STATUS CALCULATION	TRIGGER
root	Add child		
▼ SLA by service	Add child	Problem, if all children have problems	
Server 1	Add child Delete	Problem, if at least one child has a problem	
Server 2	Add child Delete	Problem, if at least one child has a problem	
Server 3	Add child Delete	Problem, if at least one child has a problem	
Server 4	Add child Delete	Problem, if at least one child has a problem	
Server 5	Add child Delete	Problem, if at least one child has a problem	

有关添加服务的详细信息，请参阅 [IT 服务](#) 模块。

5 管理

概览

管理菜单用于Zabbix的管理功能。此菜单仅适用于[Super Administrators](#)类型的用户。

1 常规设置

概述

管理 ->常规 部分包含多个用于设置前端相关默认值和自定义Zabbix的屏幕。

右侧的下拉菜单允许您在不同的配置屏幕之间切换。



1 GUI

此屏幕提供了与前端相关的默认值的定制。



配置参数：

参数	描述
默认主题	没有在其个人资料中设置特定的用户的默认主题。
下拉第一个入口	不管元素选择下拉列表中的第一个条目是 全部 或是 无. 并且勾选了记住所选项, 当导航到另一个页面时, 下拉列表中的最后一个选定的元素将被记住 (而不是默认值)。
限制搜索和过滤结果	将在Web界面列表中显示的元素 (行) 的最大数量, 例如, 在监控 ->触发器或者 配置 ->主机. 注意: 如果设置为例如“50”, 则前50个元素将仅显示在所有受影响的前端列表中。如果一些列表包含五十多个元素, 那么它的指示将是“+” “显示 50 + 发现中的1到50”. 另外, 如果使用过滤, 并且仍然有超过50个匹配, 则只显示前50个。
最大元素数在表格内显示	对于单个表格单元格中显示的条目, 将不再显示此处配置的条目。
启用事件确认	此参数定义在Zabbix界面中是否激活了事件确认
显示不早于..的时间于(天数)	此参数定义在“触发器状态”屏幕中显示多少天事件。 默认为7天。
每个触发器显示的最大事件计数	触发状态屏幕中每个触发器的最大事件数。 默认值为100。
如果Zabbix服务器关闭, 则显示警告	如果无法访问Zabbix服务器 (可能会关闭), 此参数将使浏览器窗口中显示警报消息。 即使用户向下滚动页面, 邮件仍然可见。 如果鼠标移过它, 该信息将被暂时隐藏以显示下面的内容。 Zabbix 2.0.1之后的版本支持此参数。

1 常规设置

2 Housekeeper

家是由Zabbix服务器执行的定期流程。 该过程消除用户删除的过时信息和信息。



在本节中，内容任务可以单独启用或禁用每个任务：事件和警报/ IT服务/审核/用户会话/历史/趋势。如果启用了家政管理，可以设置管理员被删除之前数据记录将保留多少天。

对于历史和趋势，还有其他选择： 覆盖项目历史记录周期 以及 覆盖项目趋势期。此选项允许全局设置项目历史/趋势将保留多少天，在这种情况下，覆盖[项目配置](#)中保留历史记录/保留趋势字段中各个项目的值集。

即使禁用内部管家，也可以覆盖历史/趋势存储期。因此，当使用外部管家时，可以使用历史记录数据存储期间字段设置历史存储期。

恢复默认值 按钮可以恢复所做的任何更改。

3 图片

图像部分显示Zabbix中可用的所有图像。 图像存储在数据库中。



类型 下拉菜单允许您在图标和背景图像之间切换：

- 标用于显示[网络图](#) 元素
- 背景用作网络图的背景图像

添加图像

您可以通过点击右上角创建图标或者创建背景按钮添加自己的图像。



图像属性：

参数	描述
名称	图像的唯一名称。
上传	从本地系统中选择要上传到Zabbix的文件 (PNG, JPEG)。

上传文件的最大大小受ZBX_MAX_IMAGE_SIZE值的限制，为1024×1024字节或1 MB。如果图像大小接近1 MB，“max_allowed_packet”的MySQL配置参数的默认值为1MB，则图像的上传可能会失败。在这种情况下，增加[max_allowed_packet](#) 参数。

4 图标映射

本部分允许使用某些图标创建某些主机的映射。 主机清单字段信息用于创建映射。

然后可以使用映射[网络地图配置](#) 自动为匹配的主机分配适当的图标。

创建一个新的图标图，点击右上角的 [创建图标地图](#)。



配置参数：

参数	描述
名称	图标地图的唯一名称。
映射	映射列表。 映射顺序决定哪一个优先级。 您可以使用拖放方式在列表上下移动映射。
库存字段	将要查找一个匹配的主机库存字段。
表达式	描述匹配的正则表达式。
图标	如果找到表达式的匹配，则使用图标。
默认	要使用的默认图标。

5 正则表达式

此部分允许创建可在前端的多个位置使用的自定义正则表达式。参见[正则表达式](#) 细节。

6 宏

本节允许定义系统范围的宏。



更多细节，参见[用户宏](#)。

7 值映射

本部分允许管理对于Zabbix前端中输入数据的可读表示有用的值映射。

NAME	VALUE MAP	USED IN ITEMS
Zabbix agent ping status	1 → Up 0 → Running 1 → Paused 2 → Start pending 3 → Pause pending 4 → Continue pending 5 → Stop pending 6 → Stopped 7 → Unknown 255 → No such service	
Windows service state	0 → poweredOff 1 → poweredOn 2 → suspended	Yes
VMware VirtualMachinePowerState	0 → gray 1 → green 2 → yellow 3 → red	Yes
VMware status	1 → up 2 → down 3 → testing 4 → unknown 5 → dormant 6 → notPresent 7 → lowerLayerDown	Yes
SNMP interface status (ifOperStatus)		

更多细节，参见[值映射](#)。

8 工作时间

工作时间是系统范围的参数，用于定义工作时间。 工作时间显示为图形中的白色背景，而非工作时间显示为灰色。



时间格式描述请参见[时间段规格](#) 页面。

9 触发严重级

此部分允许自定义[触发严重级名称和颜色](#)



此部分允许自定义触发严重性您可以输入新的名称和颜色代码，或单击颜色以从提供的调色板中选择另一个。

更多信息，请参见 [自定义触发严重级](#)页面。

10 触发显示选项

此部分允许自定义触发状态在前端中的显示方式。



确认/未确认事件的颜色可以自定义并启用或禁用闪烁。

此外，可以定制显示OK触发的时间段和触发状态更改时的闪烁时间。 最大值为86400秒（24小时）。

11 其他参数

此部分允许配置其他前端参数。



参数	描述
刷新不支持项（以秒为单位）	由于用户参数错误或代理商不支持某些项目，某些项目可能会不受支持。Zabbix可以配置为定期使不受支持的项目处于活动状态。Zabbix将在此处设置N秒钟激活不受支持的项目。如果设置为0，则自动激活将被禁用。代理检查每10分钟不支持的项目。这是不可配置的代理。
发现主机组	被 网络发现 和 agent自动注册 发现的主机 将自动放置在主机组中，此处选择。
默认主机库存模式	主机库存默认 模式 。 每当由服务器或前端创建新的主机或主机原型时都是可循的，除非在主机发现/自动注册中被设置主机库存模式 选项覆盖。
数据库关闭消息的用户组	用户组发送报警信息或“无”。Zabbix服务器的可用性取决于后端数据库的可用性。如果没有数据库，它不能工作。 Database watchdog ，一个特殊的Zabbix服务器进程，会在遇到灾难时对选定的用户进行报警。如果数据库关闭，watchdog将使用所有配置的用户媒体条目向此处设置的用户组发送通知。Zabbix服务器不会停止；它将等到数据库再次重新继续处理。Note：直到Zabbix版本1.8.2数据库watchdog才支持MySQL。 自1.8.2以来，它支持所有数据库。
<i>Log unmatched SNMP traps</i>	如果没有找到相应的SNMP接口。查看日志 SNMP trap 。

2 Proxies

概述

在 Administration → Proxies 里，[分布式监控](#)可以在Zabbix前端进行配置。

Proxies

显示现有proxy列表及其详细信息

Proxies							Create proxy
<input type="checkbox"/> Name Mode Encryption Last seen (age) Host count Item count Required performance (vps)							Hosts
<input type="checkbox"/> Remote proxy Active NONE CERT Never							8 42 0.52 Apache, Discovered host, JB One, MySQL, New host, Private, VMware, Win server 2008
Displaying 1 of 1 found							

显示的信息：

Column	描述
<i>Name</i>	Proxy名称。点击proxy名可以打开当前proxy 配置表单 .
<i>Mode</i>	显示Proxy的模式 - Active 或者 Passive.
<i>Encryption</i>	显示来自proxy的连接的加密状态:None - 不加密PSK - 使用PSK方式Cert - 使用证书
<i>Last seen (age)</i>	显示sever上次看到agent的时间
<i>Host count</i>	显示被proxy监控的host数量
<i>Item count</i>	显示被proxy监控的监控项的数量。
<i>Required performance (vps)</i>	显示所需的proxy性能 (每秒需要收集的值的数量)。
<i>Hosts</i>	列出由proxy监控的所有主机。 单击主机名将打开主机配置表单。

配置新的proxy，请单击顶部右上角的 Create proxy 按键。

批量编辑选项

列表下面的按键会提供一些批量编辑选项：

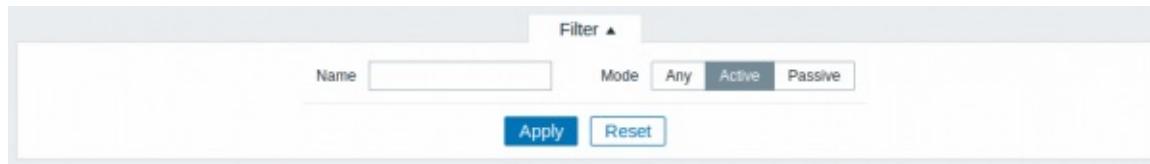
- *Enable hosts* - 将被proxy监控的host的状态改为 Monitored (监控)
- *Disable hosts* - 将被proxy监控的host的状态改为 Not monitored (不监控)
- *Delete* - 删除proxy

要使用这些选项，请在各个proxy之前标记复选框，然后单击您需要的按键。

过滤器

因为列表中可能包含许多proxy，所以可能需要通过过滤得到您需要的内容。

Filter过滤器 链接位于agent列表之上。 如果您点击它，则可以使用过滤器，您可以通过名称和模式过滤proxy。



3 身份认证

概述

在 Administration → Authentication 中，可以改变Zabbix用户身份认证方法。可用的方法为：内部认证（internal），LDAP和HTTP认证。



默认情况下，使用内部Zabbix认证。要更改的话，请点击认需要选择的证方法按钮，然后按 Update更新

Internal

使用内部Zabbix认证。

LDAP

外部LDAP认证可用于检查用户名和密码。请注意，该用户也必须存在于Zabbix中，但是它的Zabbix密码将不会被使用

Zabbix LDAP验证至少要与Microsoft Active Directory和OpenLDAP一起工作



配置参数：

参数	描述
<i>LDAP host</i>	LDAP服务器名称。例如： ldap://ldap.zabbix.com 安全LDAP服务器使用 ldaps 协议。 ldaps://ldap.zabbix.com
<i>Port</i>	LDAP服务器接口， 默认为389。 \安全 LDAP连接端口号一般为636。
<i>Base DN</i>	寻找账户的基本路径： ou=Users, ou=system (for OpenLDAP), DC=company, DC=com (for Microsoft Active Directory)
<i>Search attribute</i>	用户搜索的LDAP 账户属性： uid (for OpenLDAP), sAMAccountName (for Microsoft Active Directory)
<i>Bind DN</i>	通过LDAP服务器进行绑定和搜索的LDAP帐户，例如： uid=ldapsearch, ou=system (for OpenLDAP), CN=ldap_search, OU=user_group, DC=company, DC=com (for Microsoft Active Directory) Required, 匿名绑定目前不支持。
<i>_Bind password</i>	通过LDAP服务器进行绑定和搜索的LDAP账户密码。
<i>Test authentication</i>	测试部分的标题
<i>Login</i>	测试用户名(当前Zabbix前端登录的). 用户名必须在LDAP服务器上存在。 . 如果无法验证测试用户， Zabbix将不会激活LDAP身份验证。
<i>User password</i>	测试用户的LDAP密码。

建议创建一个单独的LDAP帐户（绑定DN），以LDAP中的最小权限执行绑定和搜索，而不使用真正的用户帐户（用于登录Zabbix前端）。这种方法提供更多的安全性，并且用户在LDAP服务器中更改密码时，不需要更改 Bind password 绑定密码。在上表中， ldap_search 是帐号名。

某些用户组仍然可以由Zabbix授权。 这些组必须具有内部的[前端访问](#)设置为内部认证) 将被Apache授权, 而不是由Zabbix授权!

HTTP

可以使用基于Apache (HTTP) 的身份验证来检查用户名和密码。 请注意, 用户也必须存在于Zabbix中, 但是它的Zabbix密码将不会被使用。

小心! 确保Apache身份验证已配置并正常工作, 然后再打开它。

<note>在Apache身份认证验证的情况下, 所有用户 (即使 [前端访问](#) 设置为内部认证) 将被Apache授权, 而不是由Zabbix授权!

4 用户组

概述

在 Administration → User groups 中，维护系统中的用户组

用户组

显示现有用户组及其详细信息的列表。

User groups						Create user group
		Members	Frontend access	Debug mode	Status	
<input type="checkbox"/>	Name ▾	#				
<input type="checkbox"/>	Zabbix administrators	Users 1	Admin (Zabbix Administrator)	System default	Enabled	Enabled
<input type="checkbox"/>	Windows administrators	Users		System default	Disabled	Enabled
<input type="checkbox"/>	Security specialists	Users 1	user (New User)	System default	Disabled	Enabled
<input type="checkbox"/>	No access to the frontend	Users		Disabled	Disabled	Enabled
<input type="checkbox"/>	Network administrators	Users		System default	Disabled	Enabled
<input type="checkbox"/>	MySQL Administrators	Users		System default	Disabled	Enabled
<input type="checkbox"/>	Linux administrators	Users		System default	Disabled	Enabled
<input type="checkbox"/>	IT management	Users		System default	Disabled	Enabled
<input type="checkbox"/>	Helpdesk	Users		System default	Disabled	Enabled
<input type="checkbox"/>	Guests	Users 1	guest	System default	Disabled	Enabled
<input type="checkbox"/>	Enabled debug mode	Users		System default	Enabled	Enabled
<input type="checkbox"/>	Disabled	Users		System default	Disabled	Disabled
<input type="checkbox"/>	Database manager	Users		System default	Disabled	Enabled

Displaying 13 of 13 found

显示的数据：

Column	描述
<i>Name</i>	用户组的名称。单击该用户组名来打开用户组的 配置列表 。
<i>#</i>	该组中用户数量。单击 <code>Users</code> 将会显示过滤出的对应的用户。
<i>Members</i>	用户组中独立用户的别名(括号内会有名字和姓氏) 单击别名将会打开用户配置列表，来自禁用组的用户会以红色显示。
<i>Frontend access</i>	显示前端访问级别： System default - Zabbix, LDAP 或 HTTP身份认证；取决于选择的身份验证的方法 Internal - 方法 Disabled - 禁止用户进行前端访问。单击当前用户级别可以改变用户访问级别。
<i>Debug mode</i>	显示Debug模式的状态 <code>Enabled</code> 或 <code>Disabled</code> 。通过点击状态可以改变它。
<i>Status</i>	显示用户组状态。- <code>Enabled</code> 或 <code>Disabled</code> 。通过点击状态可以改变它。

配置新的用户组，点击顶部右上角的 `Create user group`。

批量编辑选项

列表下面的按键会提供一些批量编辑选项：

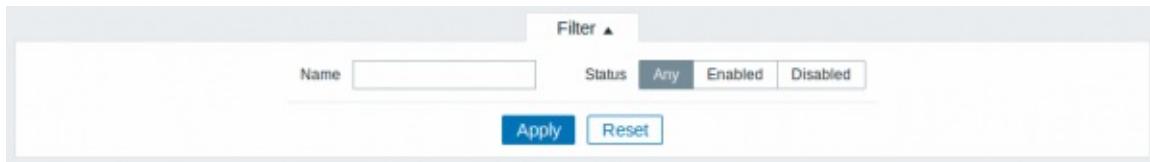
- *Enable* - 将用户组状态改为 Enabled
- *Disable* - 将用户组状态改为 Disabled
- *Enable debug mode* - 激活该用户组的debug模式
- *Disable debug mode* - 禁用该用户组的debug模式
- *Delete* - 删除用户组

要使用这些选项，请在各个用户组之前标记复选框，然后单击您需要的按键。

过滤器

因为列表中可能包含许多用户组，所以可能需要通过过滤得到您需要的内容。

[Filter过滤器](#) 链接位于用户组列表之上。 如果您点击它，则可以使用过滤器，您可以通过名称和状态过滤用户组。



5 用户

概述

在 Administration → Users 中，维护系统中的用户。

用户

显示现有用户及其详细信息的列表。

Users										User group	All	Create user
	Alias	Name	Surname	User type	Groups	Is online?	Login	Frontend access	Debug mode	Status		
<input type="checkbox"/>	Admin	Zabbix	Administrator	Zabbix Super Admin	Zabbix administrators	Yes (2016-07-01 02:05:18)	Ok	System default	Enabled	Enabled		
<input type="checkbox"/>	guest			Zabbix User	Guests	Yes (2016-07-01 02:05:13)	Ok	System default	Disabled	Enabled		
<input type="checkbox"/>	user	New	User	Zabbix Admin	Security specialists	No (2016-02-29 09:57:19)	Ok	System default	Disabled	Enabled		

Displaying 3 of 3 found

从 Users 栏中的右侧的下拉列表中，您可以选择是显示所有用户或是只显示属于一个特定组的用户。

显示的数据：

列	描述
Alias	用户的别名, 用于登录Zabbix。点击该别名打开用户 配置表单 。
Name	用户的名字。
Surname	用户的姓氏。
User type	显示用户类型 - Zabbix Super Admin, (Zabbix超级管理员) Zabbix Admin (Zabbix管理员) 或者 Zabbix User (Zabbix用户)。
Groups	显示用户类型 - Zabbix Super Admin, (Zabbix超级管理员) Zabbix Admin (Zabbix管理员) 或者 Zabbix User (Zabbix用户)。
Is online?	显示用户在线状态- Yes 或 No. 用户最后活动时间显示在括号内。
Login	显示用户的登录状态。 - Ok or Blocked. 用户会因连续五次失败登录而被暂时锁定。点击 Blocked 您可以解除该用户的锁定。
Frontend access	显示前端访问级别。 - System default, Internal 或 Disabled, 这取决于整个用户组的设置。
Debug mode	显示Debug模式 - Enabled 或 Disabled, 这取决于整个用户组的设置。
Status	显示用户状态 - Enabled 或 Disabled, 这取决于整个用户组的设置。

配置新的用户，点击顶部右上角的 Create user

批量编辑选项

列表下面的按键会提供一些批量编辑选项：

- *Unblock* - 重新启用被阻止用户对系统的访问

- *Delete* - 删除用户

要使用这些选项，请在各个用户之前标记复选框，然后单击您需要的按键

过滤器

因为列表中可能包含许多用户，所以可能需要通过过滤得到您需要的内容。

Filter过滤器 链接位于用户列表之上。 如果您点击它，则可以使用过滤器，您可以通过别名，名字，姓氏和用户类型过滤用户。

The screenshot shows a user filtering interface with the following elements:

- Search fields: Alias, Name, Surname.
- User type dropdown: Any, Zabbix User, Zabbix Admin, Zabbix Super Admin.
- Action buttons: Apply, Reset.

6 媒介类型

概述

在 Administration → Media types 部分，用户可以配置和维护媒介类型信息。

媒介类型信息包含使用媒介作为通知的传送通道的一般说明。具体细节，比如发送通知的个人电子邮件地址与个人用户保持一致。

显示现有媒介类型及其详细信息的列表。

Media types					Create media type
<input type="checkbox"/> Name ▲ Type Status Used in actions				Details	
				Filter ▾	
<input type="checkbox"/>	Email	Email	Enabled	Report log problem: agent stopped, Report problems to Zabbix administrators, Report problems to Zabbix administrators II	SMTP server: "mail.zabbix.com", SMTP helo: "zabbix.com", SMTP email: "monitoring.info@zabbix.com"
<input type="checkbox"/>	Jabber	Jabber	Enabled		Jabber identifier: "jabber@company.com"
<input type="checkbox"/>	Script	Script	Enabled		Script name: "script.sh"
<input type="checkbox"/>	SMS	SMS	Enabled		GSM modem: "/dev/ttyS0"

Displaying 4 of 4 found

显示的信息：

列	描述
Name	媒介类型名称。单击名称打开媒介类型 配置表单 。
Type	显示媒介类型（电子邮件，短信等）
Status	显示媒介类型状态 - Enabled or Disabled. 您可用过单击来改变其状态。
Used in actions	将显示直接使用媒介类型的所有动作（仅在 Send only to 下拉菜单中选择）。单击动作名称打开动作配置表单。
Details	显示媒介类型的详细信息。

配置新的媒介类型，点击顶部右上角的 Create media type

批量编辑选项

列表下面的按键会提供一些批量编辑选项：

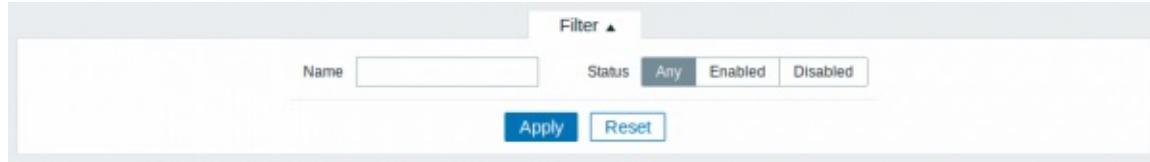
- *Enable* - 将媒介类型状态改为 Enabled
- *Disable* - 将媒介类型状态改为 Disabled
- *Delete* - 删除媒介类型

要使用这些选项，请在各个媒介类型之前标记复选框，然后单击您需要的按键。

过滤器

因为列表中可能包含许多媒介类型，所以可能需要通过过滤得到您需要的内容。

Filter过滤器 链接位于媒介类型列表之上。 如果您点击它，则可以使用过滤器，您可以通过名称和状态过滤媒介类型。



7 脚本

概述

在 Administration → Scripts 中，可以配置和维护用户定义的全局脚本。

这些脚本取决于设置的用户权限，之后可以通过单击主机上各个前端位置(Dashboard, Problems, Latest data, Status of triggers, Maps)便可执行，同时也可以用作行动操作来运行。脚本在Zabbix sever或agent上执行。显示现有脚本及其详细信息的列表

Scripts						
Create script Filter ▾						
Name	Type	Execute on	Commands	User group	Host group	Host access
Detect operating system	Script	Server	sudo /usr/bin/hmap -O {HOST.CONN}	Zabbix administrators	All	Read
Ping	Script	Server	/bin/ping -c 3 {HOST.CONN}	All	All	Read
Traceroute	Script	Server	/usr/bin/traceroute {HOST.CONN}	All	All	Read

Displaying 3 of 3 found

显示的数据：

列	描述
Name	脚本名。点击该脚本名打开脚本 配置表格...
Type	显示脚本类型- Script 或者 IPMI 命令
Execute on	显示脚本执行在Zabbix sever或者agent上。
Commands	显示在脚本中执行的所有命令。
User group	显示该脚本可用的用户组(或者 All 针对所有用户组)。
Host group	将显示该脚本可用的主机组(或者 All 针对所有主机组)。
Host access	显示主机组的权限级别 Read 或者 Write. 只有具备所需权限级别的用户才能访问执行脚本。

配置新的脚本，请单击顶部右上角的 Create script 按键。

批量编辑选项

列表下面的按键会提供一些批量编辑选项：

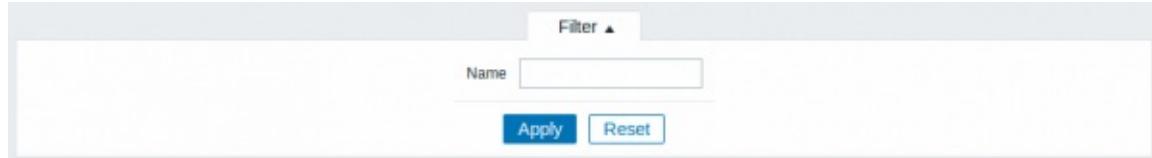
- **Delete** -删除脚本

要使用这个选项，请在各个脚本之前标记复选框，然后单击 Delete.

过滤器

因为列表中可能包含许多脚本，所以可能需要通过过滤得到您需要的内容。

Filter过滤器 链接位于脚本列表之上。 如果您点击它，则可以使用过滤器，您可以通过名称过滤脚本。



配置全局脚本

Name: Detect operating system

Type: IPMI Script

Execute on: Zabbix agent Zabbix server (proxy) Zabbix server

Commands: sudo /usr/bin/nmap -O {HOST.CONN}

Description:

User group: Zabbix administrators

Host group: Selected

Required host permissions: Read Write

Enable confirmation:

Confirmation text:

脚本属性：

参数	描述
Name	脚本的唯一名称。从Zabbix 2.2起，名称可以以所需的路径为前缀，例如 Default/，将脚本放入相应的目录。通过监控部分中的菜单访问脚本时，将根据给定的目录进行组织。脚本不能与现有目录名称相同（反之亦然）。脚本名称在其目录中必须是唯一的。未转义的脚本名称具有唯一性验证，即“Ping”和“\ Ping”无法添加到同一文件夹中。单个反斜杠在其后直接转义任何符号。例如，字符'/'和'\'可以用反斜杠转义，即\ /或\。
Type	点击相应的按键，来选择脚本类型 - IPMI command 或者 Script .
Execute on	单击对应的按键来在Zabbix server或agent上执行脚本。从Zabbix 2.0版本起，(在Zabbix agent 配置文件 中的EnableRemoteCommands参数中启用远程命令)，可以使用

	Zabbix agent 执行脚本的选项。
<i>Commands</i>	输入脚本执行命令的完整路径。命令中支持以下宏: {HOST.CONN}, {HOST.IP}, {HOST.DNS}, {HOST.HOST}, {HOST.NAME}。如果宏可能解析为具有空格的值(例如, host name), 不要忘记使用引号。从Zabbix 2.2起, 脚本命令开始支持 用户宏 。
<i>Description</i>	为脚本添加描述
<i>User group</i>	选择脚本可用的用户组 (All 是对所有的用户组)。
<i>Host group</i>	选择脚本可用的主机组 (All 是对所有主机组)。
<i>Required host permissions</i>	选择主机组的权限级别 – Read 或 Write。 只有具有所需权限级别的用户才能访问执行脚本。
<i>Enable confirmation</i>	在执行脚本之前选中复选框以显示确认消息。 对于潜在的危险操作 (如重新启动脚本) 可能需要很长的操作时间, 此功能因此会特别有用。
<i>Confirmation text</i>	使用复选框, 输入确认弹出窗口的自定义确认文本 (例如, Remote system 远程系统 将要重启, 您确定吗?)。 要查看文字的效果, 请点击该字段旁边的 Test confirmation 。 从Zabbix 2.2起, 确认文本将扩展到主机名宏 - {HOST.HOST}, {HOST.NAME}, 主机连接宏 - {HOST.IP}, {HOST.DNS}, {HOST.CONN} 和用户宏。 Note注意: 测试确认消息时, 宏不会被扩展。

脚本的执行和结果

由Zabbix sever运行的脚本由[命令执行](#) 部分中描述的顺序执行, 包括退出代码检查。 脚本结果将显示在运行脚本后显示在弹窗中。

*Note:*脚本的返回值是标准输出以及标准错误。

请参见下面的脚本和结果窗口示例:

```
1. uname
2. uname --non-existing-flag
3. /tmp/non_existing_script.sh
```



8 队列

概述

在 Administration → Queue 中，显示等待升级的监控项。

理想情况下，当您打开此部分时，应该都是“绿色”的，表示队列中没有任何监控项。如果所有监控项都没有延迟更新，则没有等待。但是，由于服务器性能匮乏，连接问题或proxy问题，有些监控项可能会延迟，并且在该区域中显示信息。有关详细信息，请参阅[Queue队列](#)。

队列仅在Zabbix服务器运行时可用。

从右上角的下拉菜单中您可以选择：

- 监控项类型队列概述
- Proxy队列概述
- 延时监控项列表

监控项类型概述

在此屏幕中，如果异常与一个或多个监控项类型相关，则可以轻松找到它。

ITEMS	5 SECONDS	10 SECONDS	30 SECONDS	1 MINUTE	5 MINUTES	MORE THAN 10 MINUTES	Overview
Zabbix agent	0	0	0	0	0	0	
Zabbix agent (active)	0	0	0	0	0	0	
Simple check	0	0	0	0	0	0	
SNMPv1 agent	0	0	0	0	0	0	
SNMPv2 agent	0	0	0	0	0	0	
SNMPv3 agent	0	0	0	0	0	0	
Zabbix internal	4	14	1	0	0	0	
Zabbix aggregate	0	0	0	0	0	0	

每行包含一个监控项类型。每列显示等待监控项的数量 - 等待5-10秒/ 10-30秒/ 30-60秒/ 1-5分钟/ 5-10分钟或超过10分钟。

Proxy概述

在此屏幕中，如果异常与agent或server之一相关，则可以轻松找到它。

PROXY	5 SECONDS	10 SECONDS	30 SECONDS	1 MINUTE	5 MINUTES	MORE THAN 10 MINUTES	Overview by proxy
Remote proxy	0	13	15	0	0	0	
Server	0	0	0	0	0	0	
Total: 2							

每行包含一个proxy，server在列表的最后一个。每列显示等待监控项的数量 - 等待5-10秒/ 10-30秒/ 30-60秒/ 1-5分钟/ 5-10分钟或超过10分钟。

等待监控项列表

在下屏中，每个等待监控项被列了出来。

Queue of items to be updated				Details
SCHEDULED CHECK	DELAYED BY	HOST	NAME	
2016-01-04 17:28:39	36s	Zabbix server 1	Zabbix busy discoverer processes, in %	
2016-01-04 17:28:40	35s	Zabbix server 1	Zabbix busy escalator processes, in %	
2016-01-04 17:28:41	34s	Zabbix server 1	Zabbix busy history syncer processes, in %	
2016-01-04 17:28:42	33s	Zabbix server 1	Zabbix busy housekeeper processes, in %	
2016-01-04 17:28:43	32s	Zabbix server 1	Zabbix busy http poller processes, in %	
2016-01-04 17:28:44	31s	Zabbix server 1	Zabbix busy icmp pinger processes, in %	

在主机列中，由proxy监视的主机以proxy名称为前缀（从Zabbix 2.4.0起）。

显示的数据：

列	描述
<i>Next check</i>	显示检查到期的时间。
<i>Delayed by</i>	显示延迟的长度。
<i>Host</i>	显示监控项的主机。
<i>Name</i>	显示等待监控项的名称。

2 用户资料

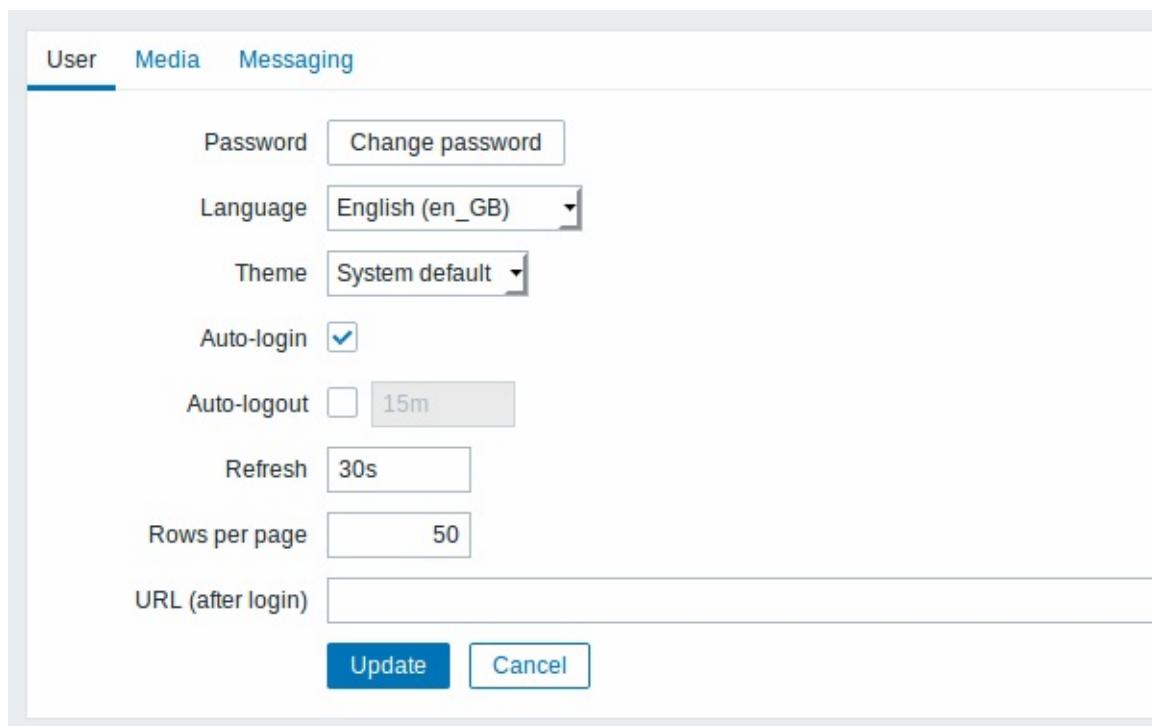
概述

在用户资料中，你可以自定义一些Zabbix的前端特性，比如：界面语言，主题颜色，列表中显示的行数等等。此改变只针对当前用户。

点击zabbix窗口右上角的  来访问用户信息

配置

User 选项卡允许您设置关于用户相关配置。



The screenshot shows the 'User' configuration page with the following settings:

- Language: English (en_GB)
- Theme: System default
- Auto-login: checked
- Auto-logout: 15m
- Refresh: 30s
- Rows per page: 50
- URL (after login): (empty)

At the bottom are 'Update' and 'Cancel' buttons.

参数	描述
<i>Password</i>	点击链接显示两个字段，来输入新的密码。
<i>Language</i>	选择您想要的界面语言。PHP的gettext扩展是翻译正常运作所必需的。
<i>Theme</i>	为您的资料选择一种特殊的颜色主题
<i>Auto-login</i>	选择复选框来标记自动登录，无需再次输入用户名和密码。
<i>Auto-logout (min 90 seconds)</i>	勾选了这个复选框后，您将在设定的秒数后自动注销(最少90秒)。但是该选项在会在Zabbix sever关闭，全局配置选项被启用，且Zabbix前端持续打开，例如，监控目录页面一直在后台进行信息刷新时，会无法正常工作。另外，自动退出Auto-logout会在登录勾选“30天记住我”这个选项后出去非激活状态。
<i>Refresh (in seconds)</i>	您可以设置监控目录下信息刷新的频率。只有Dashboard例外，它使用为自己的每个部件使用自己的刷新参数。
<i>Rows per</i>	

<i>page</i>	
<i>URL (after login)</i>	您可以设置在登录后显示的自定义 URL 不同于默认的 Monitoring → Dashboard 例如，它可以甚至成 Monitoring 的 URL → Triggers.

如果某些语言在用户资料中无法选择，则意味着它的区域设置未安装在Web服务器上。请参阅链接[link](#)，了解如何安装。

媒介**Media**选项卡允许您指定给用户以 `media` 细节，例如类型，应用的地址与何时使用它们发送通知。

The screenshot shows a user interface for managing media types. At the top, there are tabs: User, Media, and Messaging. The 'Media' tab is selected. Below the tabs is a table with the following columns: TYPE, SEND TO, WHEN ACTIVE, USE IF SEVERITY, and STATUS. There are two entries:

TYPE	SEND TO	WHEN ACTIVE	USE IF SEVERITY	STATUS
Email	user@company.com	1-5,09:00-18:00	NIWAHD	Enabled
Jabber	user@company.com	1-7,00:00-24:00	NIWAHD	Enabled

Below the table are two buttons: 'Add' (highlighted in blue) and 'Update' (blue button). To the right of the table is a 'Cancel' button.

只有 管理员级别admin level `admin level` 用户（管理员和超级管理员）可以更改他们自己的媒介细节。

可通过**Messaging**选项卡，设置全局通知`global notifications`。

参考

- 如何安装其他区域设定以便能够在用户资料中选择不可用的语言

1 全局通知

概述

全局通知是一种在Zabbix前端屏幕上显示当前正在发生的问题的方法。

没有全局通知，触发器状态 or *Dashboard* 页面，将不会显示任何有关当前正在发生的问题的信息。全局通知会显示这些信息，不论您在哪里。

全局通知涉及到信息的显示和播放声音。

配置

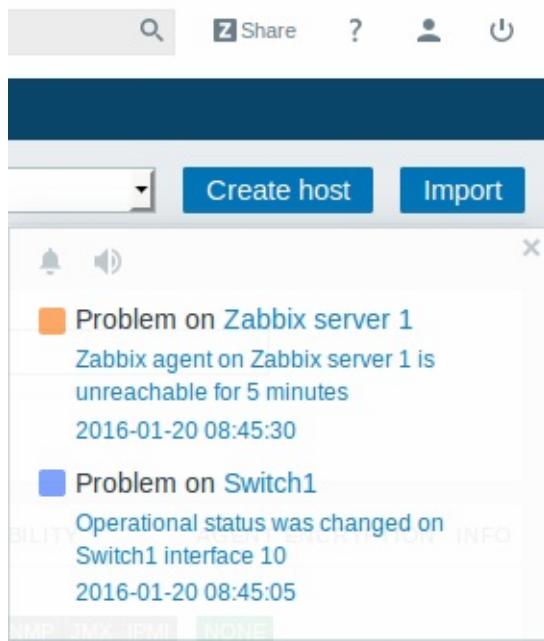
全局通知可以在每个用户的资料配置下的[Messaging](#)选项卡启用。

The screenshot shows the 'User' configuration screen with the 'Messaging' tab selected. Under 'Frontend messaging', the checkbox is checked. The 'Message timeout' is set to '60s'. The 'Play sound' dropdown is set to 'Once'. Below these, there is a section for 'Trigger severity' with seven entries, each with a checkbox, a dropdown for sound selection, and 'Play' and 'Stop' buttons. The entries are: Recovery (alarm_ok), Not classified (no_sound), Information (alarm_information), Warning (alarm_warning), Average (alarm_average), High (alarm_high), and Disaster (alarm_disaster). At the bottom are 'Update' and 'Cancel' buttons.

参数	描述
<i>Frontend messaging</i>	选中该复选框以启用全局通知。
<i>Message timeout</i>	您可以设置消息显示的时间。默认情况下，消息将在屏幕上显示60秒。
<i>Play sound</i>	您可以设置声音的播放长度。 Once - 声音完整播放一次。 10 seconds - 声音重复播放10秒。 Message timeout - 当消息显示时，声音一直播放。
<i>Trigger severity</i>	您可以设置全局通知和声音的触发器的严苛度。同时，您还可以针对不同的严苛度选择合适的声音。如果没有标记严苛度，那么就不会显示任何消息。而且，只有标记的严苛性才会显示恢复信息。因此，如果标记 <i>Recovery</i> 和 <i>Disaster</i> ，全局通知将会显示问题，以及灾难严苛度触发器的恢复。

全局信息显示

当消息到达时，它们显示在右侧的浮动区域中。 通过拖动区域标题可以垂直地重新放置此部分。



在这个区域内，一些控件是可用的：

- Snooze 键将会静音当前的警报音
- Mute/Unmute 键在播放与不播放警报音之间切换。

2 浏览器中的声音

概述

为了在Zabbix前端播放声音，*Frontend messaging* 必须在用户档案里的 *Messaging* 选项卡里被启用，并检查所有触发器的严重程度，同时声音也必须在全局通知的弹窗里被启用。

Zabbix前端的声音已经在以下Web浏览器版本中成功测试，且不需要其的配置：

- Linux上的Firefox 3.5.16
- Linux上的Opera 11.01
- Windows上的Google Chrome 9.0
- Windows上的Firefox 3.5.16
- Windows上的IE7
- Windows上的Opera v11.01
- Windows上的Chrome v9.0
- Windows上的Safari v5.0，但该浏览器需要安装 Quick Time Player。

附加要求

Firefox v 3.5.16

要在Firefox浏览器中播放wav文件，您可以使用以下应用：

- Windows Media Player
- Quick Time 插件.

然后，在 *Tools* (工具) → *Options* (选项) → *Applications* (应用程序)，中，“Wave sound (audio / wav)”中设置Windows Media Player播放这些文件

Safari 5.0

需要Quick Time Player.

IE浏览器

在 IE7和IE8中播放声音：

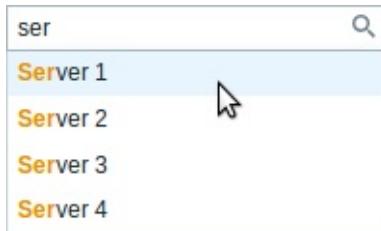
- 在 *Tools*工具 → *Internet*选项 → 高级 启用 在网页中播放声音s
- In *Tools*工具 → 插件管理... 启用 Windows Media Player
- 在Windows Media Player中， 工具 (Tools) → 选项 → 文件类型 启用 启用 Windows audio file (wav)在 Windows Media Player中， 工具 (Tools) → 选项标签 *Options tab*, “文件类型”只对“高级用

户”和“系统管理员”可用，比如，普通用户无权访问此选项卡，不会看到它。另外，如果 IE 在本地缓存文件夹(%userprofile%\Local Settings\Temporary Internet Files)中没有wav文件，那么声音文件在首次不会被播放。== 已知无法正常工作的 ==无法播放声音的浏览器：* 在Linux下的Opera 10.11

3 全局搜索

在Zabbix前端，可以搜索多种实体。实体搜索输入框在右上角。搜索可以通过摁 回车 键或者点击  搜索图标来开始搜索。

如果有一个以输入的字符串开头的主机，将显示一个下拉列表，列出所有这样的主机：



实体搜索

可以搜索这些实体及其属性：

- 主机
 - 主机名
 - 可见名
 - IP地址
 - DNS 名
- 模板
 - 名称
- H主机组
 - 名称

指定父主机组间接地选择所有嵌套的主机组

在搜索结果中，可以折叠每个单独的块。 每个块下面会，会有找到的实体数和显示的实体数，例如，例如 `Displaying 13 of 13 found.` (显示了已找到的13个实体中的13个)。 每个块中显示的条目数量限制为

The screenshot shows the Zabbix search interface with the query "Search: Zabbix server". The results are displayed in three sections:

- Hosts**: Shows one host named "Zabbix server" with IP 192.168.3.31 and DNS jmrc.zabbix.lan. It provides links to Latest data, Triggers, Problems, Graphs, Screens, Web, Applications, Items, Triggers, Graphs, Discovery, and Web.
- Host groups**: Shows one host group named "Zabbix servers" with a link to Latest data, Triggers, Problems, Graphs, Web, Hosts, and Templates.
- Templates**: Shows one template named "Template App Zabbix Server" with a link to Applications, Items, Triggers, Graphs, Screens, Discovery, and Web.

Each section includes a "Displaying 1 of 1 found" message at the bottom right.

100。

对于所有配置实体，找到的实体数会显示在实体名称的旁边。

勾选的主机显示为蓝色，禁用的主机显示为红色。如果一个主机/模板的主机名和查询字符串匹配却和可见名不同，那它将显示在可见名下方的括号里。

可用链接

对于找到的实体，下列链接均可用：

- 主机
 - 监控
 - 最新数据
 - 触发器
 - 异常
 - 图 (从 Zabbix 2.2起)
 - 主机屏幕
 - Web场景 (从 Zabbix 2.2起)
 - 配置
 - 主机属性
 - 应用
 - 监控项
 - 触发器
 - 图
 - 发现规则 (从 Zabbix 2.2起)

- Web场景(从 Zabbix 2.2起)
- 主机组
 - 监控
 - 最新数据
 - 触发器
 - 异常
 - 图 (从 Zabbix 2.2起)
 - Web场景 (从 Zabbix 2.2起)
 - 配置
 - 主机组属性
 - 主机组成员(主机和模板：独立链接从 Zabbix 2.2起)
- 模板
 - 配置
 - 模板属性
 - 应用
 - 监控项
 - 触发器
 - 图
 - 模板屏幕
 - 发现规则 (从 Zabbix 2.2起)
 - Web场景 (从 Zabbix 2.2起)

4 前端维护模式

概述

Zabbix web前端可以暂时禁用，以禁止访问它。这对于保护Zabbix数据库免受用户发起的任何更改非常有用，从而保护了数据库的完整性。

Zabbix数据库可以被停止，并且维护任务可以在Zabbix 前端在维护模式中进行。

来自指定IP地址的用户将能够在维护模式期间正常工作。

配置

为了启用维护模式，必须以取消注释的方法修改 `maintenance.inc.php` 文件（位于web服务器上的Zabbix HTML 文档目录的/`conf` 中）：

```

1. // 维护模式
2. define('ZBX_DENY_GUI_ACCESS', 1);
3.
4. // 一列包含IP地址的数组，它们可以连接到前端（可选）
5. $ZBX_GUI_ACCESS_IP_RANGE = array('127.0.0.1');
6.
7. // 警告屏幕上所显示的信息（可选）
8. $ZBX_GUI_ACCESS_MESSAGE = 'We are upgrading MySQL database till 15:00. Stay tuned...';

```

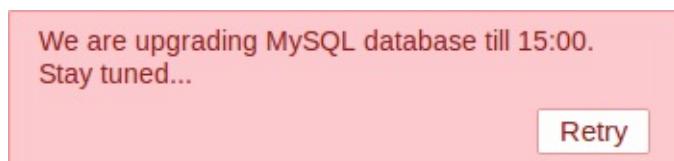
参数	详情
<code>ZBX_DENY_GUI_ACCESS</code>	打开维护模式：1 - 维护模式已打开，其他的数字表示未打开
<code>ZBX_GUI_ACCESS_IP_RANGE</code>	

例如：`array('192.168.1.1', '192.168.1.2')` |

<code>ZBX_GUI_ACCESS_MESSAGE</code>	一条由您定义的信息，用来通知用户正在进行维护（可选）
-------------------------------------	----------------------------

显示

下图显示了在维护模式下访问Zabbix前端的情况。屏幕每30秒刷新一次，以便在维护结束后，无需用户干预即可恢复正常状态。



在`ZBX_GUI_ACCESS_IP_RANGE`中定义的IP地址也可以一直访问前端。

5 页面参数

概述

大多数Zabbix Web界面页面支持各种HTTP GET参数，用于控制显示的内容。可以通过在URL之后指定 parameter=value (参数=值) 对成对来传递它们，通过问号 (?) 与URL分开，并通过&符号(&)彼此分隔。

触发器状态

Accessed as Monitoring → Triggers, 页面名 tr_status.php.

要设置过滤器，必须传递参数 filter_set=1 。 未指定的字段将重置为默认值。

一般参数

- groupid
- hostid
- fullscreen

页面特殊参数

- show_triggers - filter 选项 **Triggers status** (触发器状态), 1 - Recent problem (近期的异常), 2 - Any (任何异常), 3 - Problem (异常)
- show_events - filter 选项 **Events** (事件), 1 - Hide all (隐藏所有), 2 - See all (显示所有), 3 - Show unacknowledged (显示未确认的)
- ack_status - filter 选项 **Acknowledge status** (确认状态), 1 - Any (所有), 2 - With unacknowledged events (所有未确认状态), 3 - With last event unacknowledged (最后一个未确认事件)
- show_severity - filter 选项 **Min severity** (最小严苛度), 0-5 - corresponding severity (对应的严苛度)
- show_details - filter 选项 **Show details** (显示细节), 0 - do not show (不显示), 1 - show (显示)
- status_change_days - filter 选项 **Age less than** (时间小于), 以天计
- status_change - filter option **Age less than** (时间小于), 0 - disabled (禁用), 1 - enabled (可用) (status_change_days 可用)
- txt_select - filter option 0 - **disabled** (禁用), 1 - **enabled** (可用) (status_change_days 可用), 任意字符串
- application - filter option **Application** (应用), 任意字符串
- show_maintenance - filter option **Show hosts in maintenance** (显示在维护中的主机), 0 -

do not show hosts in maintenance (不显示在维护中的主机), 1 - show hosts in maintenance (显示在维护中的主机)

•

库存过滤器

从Zabbix 2.4.0开始，触发器可被库存过滤。这里的语法会相对复杂一下。库存的字段和值是添加在一个以0为底的索引条目中，例如：

```
1. inventory[0][field]=type_full  
2. inventory[0][value]=Virtual machine  
3. inventory[1][field]=os_full  
4. inventory[1][value]=Linux
```

这些必须是经过 URL编码，传递参数的值如下：

```
1. inventory%5B0%5D%5Bfield%5D=type_full  
2. inventory%5B0%5D%5Bvalue%5D=Virtual machine  
3. inventory%5B1%5D%5Bfield%5D=os_full  
4. inventory%5B1%5D%5Bvalue%5D=Linux
```

库存字段代码可以在 [[manual:api:reference:host:object#host_inventory|Zabbix API host 对象文档]] 中找到

触发器事件

访问一个可能对通知有用的特殊触发器事件，可以使用以下 URL

```
1. http://<server_ip_or_name>/zabbix/events.php?triggerid={TRIGGER.ID}&filter_set=1
```

6 定义

概述

虽然前端许多操作可以使用前端自带的配置功能，但自定义操作目前只能通过编辑定义文件来进行。这个文件 `defines.inc.php` 位于 Zabbix 前端程序 /include 目录

参数

该文件中的参数可能会让用户感兴趣：

- `ZBX_LOGIN_ATTEMPTS`

在登录锁定前，允许现有系统用户登录尝试失败的次数（参见 `ZBX_LOGIN_BLOCK`）。默认是五次。一旦超过了登录尝试次数的登录失败，每次额外的失败尝试都会导致登录锁定。仅用于[内部认证](#)。

- `ZBX_LOGIN_BLOCK`

在一系列不成功的登录尝试后阻止用户访问 Zabbix 前端的秒数（参见 `ZBX_LOGIN_ATTEMPTS`）。默认时间是30秒。仅用于[内部认证](#)。[internal](#)。

- `ZBX_PERIOD_DEFAULT`

默认图形周期，以秒计。默认为一个小时。

- `ZBX_MIN_PERIOD`

最小图形周期，以秒计。默认为一个小时。

- `ZBX_MAX_PERIOD`

最大图形周期，以秒计。从 Zabbix 1.6.7 开始默认为两年，之前版本默认为一年。

- `ZBX_HISTORY_PERIOD`

`Latest data, Web, Overview 和 Data overview` 页面中显示历史数据的最长期限。默认为86400秒（24 小时）。如果设置为0秒，则为无限期。

- `GRAPH_YAXIS_SIDE_DEFAULT`

在简单图形中，Y轴的默认位置，以及向自定义图形添加监控项时，下拉框的默认值。可能的值：0 – 左，1 – 右。默认：0

- `DEFAULT_LATEST_ISSUES_CNT`

控制 dashboard 的 `Last n issues` 小部件上显示的异常数。默认显示20个异常。

- `SCREEN_REFRESH_TIMEOUT` (available since 2.0.4)

用于屏幕，并定义屏幕元素更新的超时秒数。当启动更新，并超过了定义秒数且屏幕元素并未更新时，屏幕元素将变暗

默认: 30

- SCREEN_REFRESH_RESPONSIVENESS (从2.0.4版后可用)

用于屏幕，并定义关闭查询跳过的秒数。否则，如果屏幕元素处于更新状态，会跳过所有更新查询，直到收到响应。使用此参数后，可能会在N秒后发送另一个更新查询，而无需等待对第一个的响应。

默认: 10

- QUEUE_DETAIL_ITEM_COUNT

定义监控项队列的检索限制。从Zabbix 3.2.4起，可以设置比默认值高的值。默认: 500

- ZBX_SHOW_SQL_ERRORS (available since 3.4.0)

如设置true，将会在前端显示SQL错误；如果更改为false，开启了Debug模式，依然会将SQL错误显示给所有用户；Debug模式禁用，只有超级管理员可以看到SQL错误，其他用户只会看到：“SQL error. Please contact Zabbix administrator.”

默认: true

7 制作自己的主题

概述

默认情况下，Zabbix预置了许多主题。您还可以按照以下提供的步骤，制作自定义主题。如果您创作了一些很好的主题，请随时与Zabbix社区分享您的工作成果。

步骤 1

为了制作属于您自己的主题，您需要在 `styles/` 文件夹下创建一个CSS文件(例如：`custom-theme.css`)。您可以从不同的主题复制文件，并据此创建主题，或从头开始创作。

步骤2

您可以通过`Z::getThemes()`方法将您的主题添加到主题列表中。您可以通过覆盖`ZBase::getThemes()`方法来执行此操作。这可以通过在 `include/classes/core/Z.php`中的关闭括号之前添加以下代码：

```

1. public static function getThemes() {
2.     return array_merge(parent::getThemes(), array(
3.         'custom-theme' => _('Custom theme')
4.     ));
5. }
```

需要注意的是：您在第一对引号内指定的名称必须与没有扩展名的主题文件的名称相匹配。

添加多个主题，只需要将它们罗列在第一个主题下面即可，例如：

```

1. public static function getThemes() {
2.     return array_merge(parent::getThemes(), array(
3.         'custom-theme' => _('Custom theme'),
4.         'anothertheme' => _('Another theme'),
5.         'onemoretheme' => _('One more theme')
6.     ));
7. }
```

注意：除了最后的一个主题外，其他主题必须用逗号分隔。

为了改变图形颜色，必须在 `graph_theme` 数据库表格中添加该条目。

步骤3

激活新主题在Zabbix前端，您可以将此主题设置为默认主题或在用户资料改主题。

享受新的外观吧！

19. API

概述

Zabbix API允许你以编程方式检索和修改Zabbix的配置，并提供对历史数据的访问。它广泛用于：

- 创建新的应用程序以使用Zabbix；
- 将Zabbix与第三方软件集成；
- 自动执行常规任务。

Zabbix API是基于Web的API，作为Web前端的一部分提供。它使用JSON-RPC 2.0协议，这意味着两件事：

- 该API包含一组独立的方法；
- 客户端和API之间的请求和响应使用JSON格式进行编码。

有关协议和JSON的更多信息可以在[JSON-RPC 2.0 规范](#) 和 [JSON 格式主页](#)中找到。

结构

Zabbix API包含许多方法，这些方法都名义上分组为单组的API。每个方法执行一个特定任务。例如，方法

`host.create` 隶属于 `host` 这个API，用于创建新主机。历史上，API有时被称为“类”。

大多数API至少包含四种方法：`get`，`create`，`update` 和 `delete`，分别是检索，创建，更新和删除数据，但是某些API提供一套完全不同的一组方法。

执行请求

设置前端后，你就可以使用远程HTTP请求来调用API。为此，需要向 `apijsonrpc.php` 位于前端目录中的文件发送 `HTTP POST` 请求。例如，如果你的Zabbix前端安装在 `http://company.com/zabbix`，那么用HTTP请求来调用 `apiinfo.version` 方法就如下面这样：

```
1. POST http://company.com/zabbix/api_jsonrpc.php HTTP/1.1
2. Content-Type: application/json-rpc
3.
4. {"jsonrpc":"2.0","method":"apiinfo.version","id":1,"auth":null,"params":{}}
```

请求的 `Content-Type` 头部必须设置为以下值之一：`application/json-rpc`，`application/json` 或 `application/jsonrequest`。

你可以使用任何HTTP客户端或JSON-RPC测试工具手动执行API请求，但对于开发应用程序，我们建议使用[社区维护的程序库](#)。

示例

以下部分将详细介绍一些使用示例。

验证

在访问Zabbix中的任何数据之前，你需要登录并获取身份验证令牌。这可以使用该 `user.login` 方法完成。让我们假设你想要以标准Zabbix Admin用户身份登录。然后，你的JSON请求将如下所示：

```

1. {
2.     "jsonrpc": "2.0",
3.     "method": "user.login",
4.     "params": {
5.         "user": "Admin",
6.         "password": "zabbix"
7.     },
8.     "id": 1,
9.     "auth": null
10. }
```

让我们仔细看看请求对象。它具有以下属性：

- `jsonrpc` - API使用的JSON-RPC协议的版本； Zabbix API实现JSON-RPC版本2.0；
- `method` - 调用的API方法；
- `params` - 将被传递给API方法的参数；
- `id` - 请求的任意标识符；
- `auth` - 用户认证令牌；因为我们还没有一个，它的设置null。

如果你正确提供了凭据，API返回的响应将包含用户身份验证令牌：

```

1. {
2.     "jsonrpc": "2.0",
3.     "result": "0424bd59b807674191e7d77572075f33",
4.     "id": 1
5. }
```

响应对象又包含以下属性：

- `jsonrpc` - JSON-RPC协议的版本；
- `result` - 方法返回的数据；
- `id` - 相应请求的标识符。

检索主机

我们现在有一个有效的用户身份验证令牌，可以用来访问Zabbix中的数据。例如，让我们使用 `host.get` 方法检索所有已配置主机的ID，主机名和接口：

```

1. {
2.     "jsonrpc": "2.0",
3.     "method": "host.get",
4.     "params": {
5.         "output": [
6.             "hostid",
7.             "host"
8.         ],
9.         "selectInterfaces": [
10.             "interfaceid",
11.             "ip"
12.         ]
13.     },
14.     "id": 2,
15.     "auth": "0424bd59b807674191e7d77572075f33"
16. }

```

请注意，`auth` 属性现在设置为我们通过调用获得的身份验证令牌 `user.login`。

响应对象将包含有关主机的数据：

```

1. {
2.     "jsonrpc": "2.0",
3.     "result": [
4.         {
5.             "hostid": "10084",
6.             "host": "Zabbix server",
7.             "interfaces": [
8.                 {
9.                     "interfaceid": "1",
10.                    "ip": "127.0.0.1"
11.                }
12.            ]
13.        }
14.    ],
15.    "id": 2
16. }

```

出于性能原因，我们建议始终列出要检索的对象属性，并避免检索所有内容。

创建新监控项

让我们使用从上一个请求 `host.get` 中获得的数据在“Zabbix server”上创建一个新 [监控项](#)。这个可以通过使用方法 `item.create`：

```

1. {
2.     "jsonrpc": "2.0",
3.     "method": "item.create",
4.     "params": {
5.         "name": "Free disk space on $1",
6.         "key_": "vfs.fs.size[/home/joe/,free]"

```

```

7.     "hostid": "10084",
8.     "type": 0,
9.     "value_type": 3,
10.    "interfaceid": "1",
11.    "delay": 30
12. },
13. "auth": "0424bd59b807674191e7d77572075f33",
14. "id": 3
15. }

```

成功的响应将包含新创建监控项的ID，可用于在以后的请求中引用监控项：

```

1. {
2.   "jsonrpc": "2.0",
3.   "result": {
4.     "itemids": [
5.       "24759"
6.     ]
7.   },
8.   "id": 3
9. }

```

`item.create` 方法和其他的创建 (create) 方法，也可以接受对象的数组，并通过一次API调用中创建多个监控项。

创建多个触发器

因此，如果create方法接受数组，我们可以添加多个触发器，像这样：

```

1. {
2.   "jsonrpc": "2.0",
3.   "method": "trigger.create",
4.   "params": [
5.     {
6.       "description": "Processor load is too high on {HOST.NAME}",
7.       "expression": "{Linux server:system.cpu.load[percpu,avg1].last()}>5",
8.     },
9.     {
10.       "description": "Too many processes on {HOST.NAME}",
11.       "expression": "{Linux server:proc.num[].avg(5m)}>300",
12.     }
13.   ],
14.   "auth": "0424bd59b807674191e7d77572075f33",
15.   "id": 4
16. }

```

成功的响应将包含新创建的触发器的ID：

```

1. {
2.   "jsonrpc": "2.0",

```

```

3.     "result": {
4.         "triggerids": [
5.             "17369",
6.             "17370"
7.         ]
8.     },
9.     "id": 4
10. }
```

更新监控项

启用监控项，即将其状态设置为“0”：

```

1. {
2.     "jsonrpc": "2.0",
3.     "method": "item.update",
4.     "params": {
5.         "itemid": "10092",
6.         "status": 0
7.     },
8.     "auth": "0424bd59b807674191e7d77572075f33",
9.     "id": 5
10. }
```

成功的响应将包含已更新监控项的ID：

```

1. {
2.     "jsonrpc": "2.0",
3.     "result": {
4.         "itemids": [
5.             "10092"
6.         ]
7.     },
8.     "id": 5
9. }
```

`item.update` 方法以及其他更新方法也可以接受对象数组，并通过一次API调用更新多个监控项。

更新多个触发器

启用多个触发器，即将其状态设置为0：

```

1. {
2.     "jsonrpc": "2.0",
3.     "method": "trigger.update",
4.     "params": [
5.         {
6.             "triggerid": "13938",
7.             "status": 0
8.         },
9.     ]
```

```

9.      {
10.         "triggerid": "13939",
11.         "status": 0
12.     }
13. ],
14. "auth": "0424bd59b807674191e7d77572075f33",
15. "id": 6
16. }

```

成功的响应将包含更新的触发器的ID：

```

1. {
2.   "jsonrpc": "2.0",
3.   "result": {
4.     "triggerids": [
5.       "13938",
6.       "13939"
7.     ]
8.   },
9.   "id": 6
10. }

```

这是更新的首选方法。一些API方法 `host.massupdate` 允许编写更简单的代码，但不建议使用这些方法，因为它们将在未来的版本中删除。

错误处理

到那一点，我们试过的一切工作正常。但是，如果我们尝试对API调用不正确会发生什么？让我们尝试通过调用 `host.create` 创建另一个主机，但省略强制 `groups` 参数。

```

1. {
2.   "jsonrpc": "2.0",
3.   "method": "host.create",
4.   "params": {
5.     "host": "Linux server",
6.     "interfaces": [
7.       {
8.         "type": 1,
9.         "main": 1,
10.        "useip": 1,
11.        "ip": "192.168.3.1",
12.        "dns": "",
13.        "port": "10050"
14.      }
15.    ],
16.  },
17.  "id": 7,
18.  "auth": "0424bd59b807674191e7d77572075f33"
19. }

```

该响应将包含一条错误消息：

```

1. {
2.     "jsonrpc": "2.0",
3.     "error": {
4.         "code": -32602,
5.         "message": "Invalid params.",
6.         "data": "No groups for host \"Linux server\"."
7.     },
8.     "id": 7
9. }
```

如果发生错误，而不是 `result` 属性，响应对象将包含 `error` 具有以下数据的属性：

- `code` - 错误代码；
- `message` - 一个简短的错误摘要；
- `data` - 更详细的错误消息。

错误可能发生在不同的情况下，例如，使用不正确的输入值，会话超时或试图访问不存在的对象。你的应用程序应该能够正常处理这些类型的错误。

API版本

为了简化API版本控制，自Zabbix 2.0.4开始，API的版本与Zabbix本身的版本相匹配。你可以使用 `apiinfo.version` 方法查找你正在使用的API的版本。这对于调整应用程序以使用特定于版本的功能非常有用。

我们保证在主要版本内部具有向后兼容性。当在主要版本之间进行向后不兼容的更改时，我们通常将旧功能在下一个版本中保留为已弃用，并且仅在此后的版本中将其删除。有时，我们可能会删除主要版本之间的功能，而不提供任何向后兼容性。重要的是，你不要依赖任何弃用的功能，并尽快迁移到较新的替代品。

你可以在 [API更改日志中跟踪](#)对API所做的所有更改。

进一步阅读

你现在知道了足够开始使用 Zabbix API，但不要停在这里。为了进一步阅读，我们建议你查看 [可用的API列表](#)。

方法参考

本节提供了Zabbix API提供的功能的概述，并将帮助你找到可用的类和方法。

监控

Zabbix API允许你访问在监视期间收集的历史记录和其他数据。

历史

检索由Zabbix监控流程收集的历史值，用于呈现或进一步处理。

[历史API](#)

事件

检索触发器，网络发现和其他Zabbix系统生成的事件，以实现更灵活的情境管理或第三方工具集成。

[事件 API](#)

服务监控

检索有关任何服务的详细服务层可用性信息。

[服务SLA计算](#)

配置

Zabbix API允许您管理监控系统的配置。

主机和主机组

管理主机组，主机及其相关的一切，包括主机接口，主机宏和维护期。

[主机 API](#) | [主机组 API](#) | [主机接口 API](#) | [用户宏 API](#) | [维护 API](#)

监控项和应用

定义要监视的项目。创建或删除应用程序并为其分配项目。

[监控项 API](#) | [应用 API](#)

触发器

配置触发器以通知您系统中的问题。管理触发器依赖关系。

[触发器 API](#)

图表

编辑图形或单独的图形项目，以便更好地显示收集的数据。

[图表 API](#) | [图形项 API](#)

模板

管理模板并将其链接到主机或其他模板。

[模板 API](#)

导出和导入

导出和导入Zabbix配置数据以进行配置备份，迁移或大规模配置更新。

[配置 API](#)

低级发现

配置低级发现规则以及项目，触发器和图原型以监视动态实体。

[LLD规则API](#) | [项目原型API](#) | [触发器原型API](#) | [图原型API](#) | [主机原型API](#)

屏幕

分别编辑全局和模板级屏幕或每个屏幕项。

[屏幕API](#) | [屏幕项API](#) | [模板屏幕 API](#) | [模板屏幕项API](#)

动作和警报

定义动作和报警，以通知用户某些事件或自动执行远程命令。获取有关生成的警报及其接收者的信息。

[动作API](#) | [报警API](#)

服务

管理服务级别监视的服务，并检索有关任何服务的详细SLA信息。

[服务API](#)

地图

配置映射以创建IT基础架构的详细动态表示。

[地图API](#)

地图API

配置Web方案以监视Web应用程序和服务。

Web场景API

网络发现

管理网络级发现规则以自动查找和监控新主机。获得对所发现的服务和主机的信息的完全访问。

[发现规则API](#) | [发现检查API](#) | [发现主机API](#) | [发现服务API](#)

行政

使用Zabbix API，您可以更改监控系统的管理设置。

用户

添加有权访问Zabbix的用户，将其分配给用户组并授予权限。配置媒体类型和用户接收警报的方式。

[用户API](#) | [用户组API](#) | [媒介类型API](#) | [媒介API](#)

通用

更改某些全局配置选项。

[图标地图API](#) | [图像API](#) | [用户宏API](#)

代理

管理分布式监视设置中使用的代理。

[代理 API](#)

脚本

配置和执行脚本以帮助您完成日常任务。

[脚本API](#)

API信息

检索Zabbix API的版本，以便应用程序可以使用特定于版本的功能。

[API信息API](#)

Action 动作

这个类被设计用于动作。

对象引用：

- [动作](#)
- [动作条件](#)
- [动作操作](#)

可用方法：

- [action.create](#) - 创建新动作
- [action.delete](#) - 删除动作
- [action.get](#) - 检索动作
- [action.update](#) - 更新工作

> 动作对象

下面是动作API相关的对象。

动作

动作对象具有以下属性。

属性	类型	描述
actionid	string	(只读) 动作的IP。
esc_period(必需)	string	默认操作步骤持续时间。必须大于60秒。接受秒，带后缀的时间单位和用户宏。
eventsource(必需)	integer	(常量) 动作将处理的事件的类型。请参阅 事件 "source" 属性 以获取支持的事件类型列表。
name(必需)	string	动作的名称
def_longdata	string	异常消息文本。
def_shortdata	string	异常消息主题。
r_longdata	string	恢复消息文本。
r_shortdata	string	恢复消息主题。
status	integer	动作是启动还是禁用。取值：0 - (默认)启用；1 - 禁用。

动作操作

动作操作对象定义执行动作时执行的操作。它具有以下属性。

属性	类型	描述
operationid	string	(只读) 动作操作的ID。
operationtype(必需)	integer	操作类型 可能的值：0 - 发送消息；1 - 远程命令；2 - 添加主机；3 - 删除主机 4 - 添加到主机组；5 - 从主机组删除；6 - 链接到模板；7 - 取消与模板的关联；8 - 启用主机；9 - 禁用主机；10 - 设置主机库存模式。
actionid	string	该操作所属的操作的ID。
escperiod	string	升级步骤的持续时间（秒）。必须大于60秒。接受秒，时间单位后缀和用户宏。如果设置为0或0，将使用默认动作升级期。默认：0s.
esc_step_from	integer	步骤开始升级。默认：1.
esc_step_to	integer	逐步结束升级 默认：1.
evaltype	integer	操作条件评估方法。可能的值：0 - (默认) AND / OR; 1 - AND; 2 - OR.
opcommand	object	包含操作运行的命令的对象的对象。 下面详细描述操作命令对象 . 远程命令操作需要。
opcommand_grp	array	主机组运行远程命令。每个对象具有以下属性： <code>opcommand_grpid</code> - (string, 只读) 对象的ID; <code>operationid</code> - (string) 操作的ID; <code>groupid</code> - (string) 主机组的ID。如果 <code>opcommand_hst</code> 未

		设置，则需要远程命令操作。
opcommand_hst	array	主机运行远程命令。 每个对象具有以下属性： <code>opcommand_hstid</code> - (string, 只读) 对象的ID; <code>operationid</code> - (string) 操作的ID; <code>hostid</code> - (string) 主机的ID; 如果设置为0，则该命令将在当前主机上运行。 如果 <code>opcommand_grp</code> 未设置，则需要远程命令操作。
opconditions	array	用于触发动作的操作条件。 操作对象条件 下面详细描述操作 。
opgroup	array	主机组添加主机。 每个对象具有以下属性： <code>operationid</code> - (string) 操作的 ID; <code>groupid</code> - (string) 主机组的ID。 需要“添加到主机组”和“从主机组删除”操作。
opmessage	object	对象包含有关操作发送的消息的数据。 操作消息对象 下面详细描述操作 。 需要消息操作。
opmessage_grp	array	发送消息的用户组。 每个对象具有以下属性： <code>operationid</code> - (string) 操作的 ID; <code>usrgrp_id</code> - (string) 用户组的ID。 如果 <code>opmessage_usr</code> 未设置，则为消息操作所必需。
opmessage_usr	array	发送消息到用户。 每个对象具有以下属性： <code>operationid</code> - (string) 操作的 ID; <code>userid</code> - (string) 用户的ID。 如果 <code>opmessage_grp</code> 未设置，则为消息操作必需。
optemplate	array	将主机链接到的模板。 每个对象具有以下属性： <code>operationid</code> - (string) 操作的 ID; <code>templateid</code> - (string) 模板的ID。 需要“链接到模板”和“取消与模板的链接”操作。
opinventory	object	库存模式设置主机。 对象具有以下属性： <code>operationid</code> - (string) 操作的 ID; <code>inventory_mode</code> - (string) 库存模式。 需要设置主机库存模式”操作。

操作指令

操作命令对象包含有关操作将运行的命令的数据。

属性	类型	说明
<code>operationid</code>	string	(只读) 操作的ID。
<code>command</code>	string	命令运行。 输入IN(0,1,2,3)
<code>type</code> (必填)	integer	操作类型命令。 可能的值：0 - 自定义脚本； 1 - IPMI； 2 - SSH； 3 - Telnet 4 - 全局脚本。
<code>authtype</code>	integer	用于SSH命令的身份验证方法。 可能的值：0 - 密码； 1 - 公钥。 对于SSH命令是必需的。
<code>execute_on</code>	integer	将执行自定义脚本操作命令的目标。 可能的值：0 - Zabbix代理； 1 - Zabbix服务器。 自定义脚本命令必需。
<code>password</code>	string	用于具有密码认证和Telnet命令的SSH命令的密码。
<code>port</code>	string	用于SSH和Telnet命令的端口号。
<code>privatekey</code>	string	用于具有公钥认证的SSH命令的私钥文件的名称。 对于使用公钥身份验证的SSH命令，必需。
<code>publickey</code>	string	用于具有公钥认证的SSH命令的公钥文件名。 对于使用公钥身份验证的SSH命令，必需。
<code>scriptid</code>	string	用于全局脚本命令的脚本的ID。 全局脚本命令必需。
<code>username</code>	string	用于认证的用户名。 需要SSH和Telnet命令。

操作信息

操作消息对象包含有关操作将发送的消息的数据。

属性	类型	说明
operationid	string	(只读) 操作操作的ID。
defaultmsg	integer	是否使用默认动作消息文本和主题。 可能的值: 0 - (默认) _ 使用操作中的数据; 1 - 使用动作中的数据。
mediatypeid	string	将用于发送消息的媒体类型的ID。
integer	string	操作消息文本。
subject	string	操作信息主题。

动作操作条件

动作操作条件对象定义了执行当前操作必须满足的条件。 它具有以下属性。

属性	类型	说明
opconditionid	string	(readonly) 动作操作条件的ID
conditiontype \ (必填)	integer	条件类型 可能的值: 14 - 确认事件。
value \ (必填)	string	价值比较。
operationid	string	(readonly) 操作的ID。
operator	integer	条件运算符。 可能的值: 0 - (默认) =。

每个操作条件类型都支持以下运算符和值。

条件	条件名称	支持的运算符	预期值
14	Event acknowledged	=	是否确认事件。 可能的值: 0 - 未确认; 1 - 确认。

Action recovery operation

The action recovery operation object defines an operation that will be performed when a problem is resolved. Recovery operations are possible for trigger actions and internal actions. It has the following properties.

Property	Type	Description
operationid	string	(readonly) ID of the action operation.
operationtype(required)	integer	Type of operation. Possible values for trigger actions: 0 - send message; 1 - remote command; 11 - send recovery message. Possible values for internal actions: 0 - send message; 11 - send recovery message.
actionid	string	ID of the action that the recovery operation belongs to.
		Object containing the data about the command run by the recovery operation. The operation

		command object is described in detail below. Required for remote command operations.
opcommandgrp	array	Host groups to run remote commands on. Each object has the following properties: <code>opcommand_grpid</code> - (string, readonly) ID of the object; <code>operationid</code> - (string) ID of the operation; <code>groupid</code> - (string) ID of the host group. Required for remote command operations if <code>opcommand_hst</code> is not set.
opcommand_hst	array	Host to run remote commands on. Each object has the following properties: <code>opcommand_hstid</code> - (string, readonly) ID of the object; <code>operationid</code> - (string) ID of the operation; <code>hostid</code> - (string) ID of the host; if set to 0 the command will be run on the current host. Required for remote command operations if <code>opcommand_grp</code> is not set.
opmessage	object	Object containing the data about the message sent by the recovery operation. The operation message object is described in detail below. Required for message operations.
opmessage_grp	array	User groups to send messages to. Each object has the following properties: <code>operationid</code> - (string) ID of the operation; <code>usrgrpid</code> - (string) ID of the user group. Required for message operations if <code>opmessage_usr</code> is not set.
opmessage_usr	array	Users to send messages to. Each object has the following properties: <code>operationid</code> - (string) ID of the operation; <code>userid</code> - (string) ID of the user. Required for message operations if <code>opmessage_grp</code> is not set.

Action filter

The action filter object defines a set of conditions that must be met to perform the configured action operations. It has the following properties.

Property	Type	Description
<code>conditions</code> (required)	array	Set of filter conditions to use for filtering results.
<code>evaltype</code> (required)	integer	Filter condition evaluation method. Possible values: 0 - and/or; 1 - and; 2 - or; 3 - custom expression.
<code>evalformula</code>	string	(readonly) Generated expression that will be used for evaluating filter conditions. The expression contains IDs that reference specific filter conditions by its <code>formulaid</code> . The value of <code>eval_formula</code> is equal to the value of <code>formula</code> for filters with a custom expression.
<code>formula</code>	string	User-defined expression to be used for evaluating conditions of filters with a custom expression. The expression must contain IDs that reference specific filter conditions by its <code>formulaid</code> . The IDs used in the expression must exactly match the ones defined in the filter conditions: no condition can remain unused or omitted. Required

		condition can remain unused or omitted. Required for custom expression filters.
--	--	---

Action filter condition

The action filter condition object defines a specific condition that must be checked before running the action operations.

Property	Type	Description
conditionid	string	(readonly) ID of the action condition.
conditiontype(required)	integer	Type of condition. Possible values for trigger actions: 0 - host group; 1 - host; 2 - trigger; 3 - trigger name; 4 - trigger severity; 6 - time period; 13 - host template; 15 - application; 16 - maintenance status; 25 - event tag; 26 - event tag value. Possible values for discovery actions: 7 - host IP; 8 - discovered service type; 9 - discovered service port; 10 - discovery status; 11 - uptime or downtime duration; 12 - received value; 18 - discovery rule; 19 - discovery check; 20 - proxy; 21 - discovery object. Possible values for auto-registration actions: 20 - proxy; 22 - host name; 24 - host metadata. Possible values for internal actions: 0 - host group; 1 - host; 13 - host template; 15 - application; 23 - event type.
value(required)	string	Value to compare with.
value2	string	Secondary value to compare with. Required for trigger actions when condition type is 26.
actionid	string	(readonly) ID of the action that the condition belongs to.
formulaid	string	Arbitrary unique ID that is used to reference the condition from a custom expression. Can only contain capital-case letters. The ID must be defined by the user when modifying filter conditions, but will be generated anew when requesting them afterward.
operator	integer	Condition operator. Possible values: 0 - (default) =; 1 - <>; 2 - like; 3 - not like; 4 - in; 5 - >=; 6 - <=; 7 - not in.

To better understand how to use filters with various types of expressions, see examples on the [action.get](#) and [action.create](#) method pages.

The following operators and values are supported for each condition type.

Condition	Condition name	Supported operators	Expected value
0	Host group	=, <>	Host group ID.
1	Host	=, <>	Host ID.
2	Trigger	=, <>	Trigger ID.
	Trigger	like, not	

4	Trigger severity	=, <>, >=, <=	Trigger severity. Refer to the trigger "severity" property for a list of supported trigger severities.
5	Trigger value	=	Trigger value. Refer to the trigger "value" property for a list of supported trigger values.
6	Time period	in, not in	Time when the event was triggered as a time period .
7	Host IP	=, <>	One or several IP ranges to check separated by commas. Refer to the network discovery configuration section for more information on supported formats of IP ranges.
8	Discovered service type	=, <>	Type of discovered service. The type of service matches the type of the discovery check used to detect the service. Refer to the discovery check "type" property for a list of supported types.
9	Discovered service port	=, <>	One or several port ranges separated by commas.
10	Discovery status	=	Status of a discovered object. Possible values: 0 - host or service up; 1 - host or service down; 2 - host or service discovered; 3 - host or service lost.
11	Uptime or downtime duration	>=, <=	Time indicating how long has the discovered object been in the current status in seconds.
12	Received values	=, <>, >=, <=, like, not like	Value returned when performing a Zabbix agent, SNMPv1, SNMPv2 or SNMPv3 discovery check.
13	Host template	=, <>	Linked template ID.
15	Application	=, like, not like	Name of the application.
16	Maintenance status	in, not in	No value required: using the "in" operator means that the host must be in maintenance, "not in" - not in maintenance.
18	Discovery rule	=, <>	ID of the discovery rule.
19	Discovery check	=, <>	ID of the discovery check.
20	Proxy	=, <>	ID of the proxy.
21	Discovery object	=	Type of object that triggered the discovery event. Possible values: 1 - discovered host; 2 - discovered service.
22	Host name	like, not like	Host name.
23	Event type	=	Specific internal event. Possible values: 0 - item in "not supported" state; 1 - item in "normal" state; 2 - LLD rule in "not supported" state; 3 - LLD rule in "normal"

23	Event type	=	supported" state; 3 - LLD rule in "normal" state; 4 - trigger in "unknown" state; 5 - trigger in "normal" state.
24	Host metadata	like, not like	Metadata of the auto-registered host.
25	Tag	=, <>, like, not like	Event tag.
26	Tag value	=, <>, like, not like	Event tag value.

action.create

描述

```
object action.create(object/array actions)
```

此方法用来创建新的动作

参数

(object/array) 创建新动作

除此之外 standard action properties, 该方法接受以下参数。

参数	类型	描述
filter	object	这个动作的动作过滤器对象。
operations	array	为这个动作创建的动作操作。
recovery_operations	array	为这个动作创建恢复操作。
acknowledge_operations	array	为这个操作创建确认操作。

返回值

返回一个对象，其中包含 `actionids` 属性下已创建操作的 ID。 返回的 ID 的顺序与传递的操作的顺序相匹配。

示例

创建一个触发器动作

创建一个动作，当来自主机 “30045” 的触发器在其名称中出现 “memory” 进入问题状态时将运行该动作。 该操作必须首先向用户组 “7” 中的所有用户发送消息。 如果事件在 4 分钟内未得到解决，它将在组 “2” 中的所有主机上运行脚本 “3”。 在触发恢复时，它将通知所有收到有关此问题的消息的用户。 在触发确认时，带有自定义主题和正文的消息将发送给通过所有媒体类型留下确认和评论的所有人。

请求：

```

1. {
2.   "jsonrpc": "2.0",
3.   "method": "action.create",
4.   "params": {
5.     "name": "Trigger action",
6.     "eventsource": 0,
7.     "status": 0,
8.     "esc_period": "2m",
9.     "def_shortdata": "{TRIGGER.NAME}: {TRIGGER.STATUS}",
10.    "def_longdata": "{TRIGGER.NAME}: {TRIGGER.STATUS}\r\nLast value:
```

```

{ITEM.LASTVALUE}\r\n\r\n{TRIGGER.URL}",
11.    "filter": {
12.        "evaltype": 0,
13.        "conditions": [
14.            {
15.                "conditiontype": 1,
16.                "operator": 0,
17.                "value": "10084"
18.            },
19.            {
20.                "conditiontype": 3,
21.                "operator": 2,
22.                "value": "memory"
23.            }
24.        ]
25.    },
26.    "operations": [
27.        {
28.            "operationtype": 0,
29.            "esc_period": "0s",
30.            "esc_step_from": 1,
31.            "esc_step_to": 2,
32.            "evaltype": 0,
33.            "opmessage_grp": [
34.                {
35.                    "usrgrpid": "7"
36.                }
37.            ],
38.            "opmessage": {
39.                "default_msg": 1,
40.                "mediatypeid": "1"
41.            }
42.        },
43.        {
44.            "operationtype": 1,
45.            "esc_step_from": 3,
46.            "esc_step_to": 4,
47.            "evaltype": 0,
48.            "opconditions": [
49.                {
50.                    "conditiontype": 14,
51.                    "operator": 0,
52.                    "value": "0"
53.                }
54.            ],
55.            "opcommand_grp": [
56.                {
57.                    "groupid": "2"
58.                }
59.            ],
60.            "opcommand": {
61.                "type": 4,
62.                "scriptid": "3"

```

```

63.         }
64.     }
65. ],
66. "recovery_operations": [
67. {
68.     "operationtype": "11"
69. },
70. ],
71. "acknowledge_operations": [
72. {
73.     "operationtype": "12",
74.     "opmessage": {
75.         "message": "Custom acknowledge operation message body",
76.         "subject": "Custom acknowledge operation message subject"
77.     }
78. }
79. ]
80. },
81. "auth": "038e1d7b1735c6a5436ee9eae095879e",
82. "id": 1
83. }

```

响应：

```

1. {
2.     "jsonrpc": "2.0",
3.     "result": {
4.         "actionids": [
5.             "17"
6.         ]
7.     },
8.     "id": 1
9. }

```

创建一个发现动作

创建将已发现的主机链接到模板“30085”的操作。

请求：

```

1. {
2.     "jsonrpc": "2.0",
3.     "method": "action.create",
4.     "params": {
5.         "name": "Discovery action",
6.         "eventsources": 1,
7.         "status": 0,
8.         "esc_period": "0s",
9.         "filter": {
10.             "evaltype": 0,
11.             "conditions": [

```

```

12.      {
13.          "conditiontype": 21,
14.          "value": "1"
15.      },
16.      {
17.          "conditiontype": 10,
18.          "value": "2"
19.      }
20.  ]
21. },
22. "operations": [
23.     {
24.         "esc_step_from": 1,
25.         "esc_period": "0s",
26.         "optemplate": [
27.             {
28.                 "templateid": "10091"
29.             }
30.         ],
31.         "operationtype": 6,
32.         "esc_step_to": 1
33.     }
34.   ],
35. },
36. "auth": "038e1d7b1735c6a5436ee9eae095879e",
37. "id": 1
38. }
```

响应：

```

1. {
2.     "jsonrpc": "2.0",
3.     "result": {
4.         "actionids": [
5.             "18"
6.         ]
7.     },
8.     "id": 1
9. }
```

使用自定义表达式过滤器

创建将使用自定义筛选条件的触发器动作。 对于主机 “10084” 和 “10106”，操作必须为每个触发器发送一条消息，其严重性大于或等于 “警告”。 公式ID “A”， “B”和 “C”是任意选择的。

请求：

```

1. {
2.     "jsonrpc": "2.0",
3.     "method": "action.create",
4.     "params": {
```

```
5.         "name": "Trigger action",
6.         "eventsource": 0,
7.         "status": 0,
8.         "esc_period": "2m",
9.         "def_shortdata": "{TRIGGER.NAME}: {TRIGGER.STATUS}",
10.        "def_longdata": "{TRIGGER.NAME}: {TRIGGER.STATUS}\r\nLast value:
{ITEM.LASTVALUE}\r\n\r\n{TRIGGER.URL}",
11.        "filter": {
12.            "evaltype": 3,
13.            "formula": "A and (B or C)",
14.            "conditions": [
15.                {
16.                    "conditiontype": 4,
17.                    "operator": 5,
18.                    "value": "2",
19.                    "formulaid": "A"
20.                },
21.                {
22.                    "conditiontype": 1,
23.                    "operator": 0,
24.                    "value": "10084",
25.                    "formulaid": "B"
26.                },
27.                {
28.                    "conditiontype": 1,
29.                    "operator": 0,
30.                    "value": "10106",
31.                    "formulaid": "C"
32.                }
33.            ],
34.        },
35.        "operations": [
36.            {
37.                "operationtype": 0,
38.                "esc_period": "0s",
39.                "esc_step_from": 1,
40.                "esc_step_to": 2,
41.                "evaltype": 0,
42.                "opmessage_grp": [
43.                    {
44.                        "usrgrpid": "7"
45.                    }
46.                ],
47.                "opmessage": {
48.                    "default_msg": 1,
49.                    "mediatypeid": "1"
50.                }
51.            }
52.        ],
53.    },
54.    "auth": "038e1d7b1735c6a5436ee9eae095879e",
55.    "id": 1
56. }
```

响应：

```
1. {
2.     "jsonrpc": "2.0",
3.     "result": {
4.         "actionids": [
5.             "18"
6.         ]
7.     },
8.     "id": 1
9. }
```

参见

- [Action filter](#)
- [Action operation](#)

Source

`CAction::create()` in *frontends/php/include/classes/api/services/CAction.php*.

action.delete

Description

```
object action.delete(array actionIds)
```

This method allows to delete actions.

Parameters

(array) IDs of the actions to delete.

Return values

(object) Returns an object containing the IDs of the deleted actions under the `actionids` property.

Examples

Delete multiple actions

Delete two actions.

Request:

```
1. {
2.   "jsonrpc": "2.0",
3.   "method": "action.delete",
4.   "params": [
5.     "17",
6.     "18"
7.   ],
8.   "auth": "3a57200802b24cda67c4e4010b50c065",
9.   "id": 1
10. }
```

Response:

```
1. {
2.   "jsonrpc": "2.0",
3.   "result": {
4.     "actionids": [
5.       "17",
6.       "18"
7.     ]
8.   },
9.   "id": 1
}
```

action.delete

```
10. }
```

Source

CAction::delete() in *frontends/php/include/classes/api/services/CAction.php*.

action.get

Description

```
integer/array action.get(object parameters)
```

The method allows to retrieve actions according to the given parameters.

Parameters

(object) Parameters defining the desired output.

The method supports the following parameters.

Parameter	Type	Description
actionids	string/array	Return only actions with the given IDs.
groupids	string/array	Return only actions that use the given host groups in action conditions.
hostids	string/array	Return only actions that use the given hosts in action conditions.
triggerids	string/array	Return only actions that use the given triggers in action conditions.
mediatypeids	string/array	Return only actions that use the given media types to send messages.
usrgrpids	string/array	Return only actions that are configured to send messages to the given user groups.
userids	string/array	Return only actions that are configured to send messages to the given users.
scriptids	string/array	Return only actions that are configured to run the given scripts.
selectFilter	query	Returns the action filter in the <code>filter</code> property.
selectOperations	query	Return action operations in the <code>operations</code> property.
selectRecoveryOperations	query	Return action recovery operations in the <code>recoveryOperations</code> property.
sortfield	string/array	Sort the result by the given properties. Possible values are: <code>actionid</code> , <code>name</code> and <code>status</code> .
countOutput	flag	These parameters being common for all <code>get</code> methods are described in the reference commentary .
editable	boolean	
excludeSearch	flag	

filter	object
limit	integer
output	query
preservekeys	flag
search	object
searchByAny	boolean
searchWildcardsEnabled	boolean
sortorder	string/array
startSearch	flag

Return values

(integer/array) Returns either:

- an array of objects;
- the count of retrieved objects, if the `countOutput` parameter has been used.

Examples

Retrieve discovery actions

Retrieve all configured discovery actions together with action conditions and operations. The filter uses the “and” evaluation type, so the `formula` property is empty and `eval_formula` is generated automatically.

Request:

```

1. {
2.   "jsonrpc": "2.0",
3.   "method": "action.get",
4.   "params": {
5.     "output": "extend",
6.     "selectOperations": "extend",
7.     "selectRecoveryOperations": "extend",
8.     "selectFilter": "extend",
9.     "filter": {
10.       "eventsource": 1
11.     }
12.   },
13.   "auth": "038e1d7b1735c6a5436ee9eae095879e",
14.   "id": 1
15. }
```

Response:

```

1.  {
2.      "jsonrpc": "2.0",
3.      "result": [
4.          {
5.              "actionid": "2",
6.              "name": "Auto discovery. Linux servers.",
7.              "eventsources": "1",
8.              "status": "1",
9.              "esc_period": "0s",
10.             "def_shortdata": "",
11.             "def_longdata": "",
12.             "recovery_msg": "0",
13.             "r_shortdata": "",
14.             "r_longdata": "",
15.             "filter": {
16.                 "evaltype": "0",
17.                 "formula": "",
18.                 "conditions": [
19.                     {
20.                         "conditiontype": "10",
21.                         "operator": "0",
22.                         "value": "0",
23.                         "value2": "",
24.                         "formulaid": "B"
25.                     },
26.                     {
27.                         "conditiontype": "8",
28.                         "operator": "0",
29.                         "value": "9",
30.                         "value2": "",
31.                         "formulaid": "C"
32.                     },
33.                     {
34.                         "conditiontype": "12",
35.                         "operator": "2",
36.                         "value": "Linux",
37.                         "value2": "",
38.                         "formulaid": "A"
39.                     }
40.                 ],
41.                 "eval_formula": "A and B and C"
42.             },
43.             "operations": [
44.                 {
45.                     "operationid": "1",
46.                     "actionid": "2",
47.                     "operationtype": "6",
48.                     "esc_period": "0s",
49.                     "esc_step_from": "1",
50.                     "esc_step_to": "1",
51.                     "evaltype": "0",
52.                     "opconditions": [],
53.                     "optemplate": [

```

```

54.          {
55.              "operationid": "1",
56.              "templateid": "10001"
57.          }
58.      ],
59.  },
60.  {
61.      "operationid": "2",
62.      "actionid": "2",
63.      "operationtype": "4",
64.      "esc_period": "0s",
65.      "esc_step_from": "1",
66.      "esc_step_to": "1",
67.      "evaltype": "0",
68.      "opconditions": [],
69.      "opgroup": [
70.          {
71.              "operationid": "2",
72.              "groupid": "2"
73.          }
74.      ]
75.  }
76. ],
77. "recoveryOperations": [
78.     {
79.         "operationid": "585",
80.         "actionid": "2",
81.         "operationtype": "11",
82.         "evaltype": "0",
83.         "opconditions": [],
84.         "opmessage": {
85.             "operationid": "585",
86.             "default_msg": "1",
87.             "subject": "{TRIGGER.STATUS}: {TRIGGER.NAME}",
88.             "message": "Trigger: {TRIGGER.NAME}\r\nTrigger status: {TRIGGER.STATUS}\r\nTrigger severity: {TRIGGER.SEVERITY}\r\nTrigger URL: {TRIGGER.URL}\r\n\r\nItem values:\r\n1. {ITEM.NAME1} ({HOST.NAME1}:{ITEM.KEY1}): {ITEM.VALUE1}\r\n2. {ITEM.NAME2} ({HOST.NAME2}:{ITEM.KEY2}): {ITEM.VALUE2}\r\n3. {ITEM.NAME3} ({HOST.NAME3}:{ITEM.KEY3}): {ITEM.VALUE3}\r\nOriginal event ID: {EVENT.ID}",
89.             "mediatypeid": "0"
90.         }
91.     }
92. ],
93. },
94. ],
95. "id": 1
96. }

```

See also

- [Action filter](#)
- [Action operation](#)

action.get

Source

CAction::get() in *frontends/php/include/classes/api/services/CAction.php*.

action.update

Description

```
object action.update(object/array actions)
```

This method allows to update existing actions.

Parameters

(object/array) Action properties to be updated.

The `actionid` property must be defined for each action, all other properties are optional. Only the passed properties will be updated, all others will remain unchanged.

Additionally to the [standard action properties](#), the method accepts the following parameters.

Parameter	Type	Description
filter	object	Action filter object to replace the current filter.
operations	array	Action operations to replace existing operations.
recovery_operations	array	Action recovery operations to replace existing recovery operations.

Return values

(object) Returns an object containing the IDs of the updated actions under the `actionids` property.

Examples

Disable action

Disable action, that is, set its status to "1".

Request:

```

1. {
2.   "jsonrpc": "2.0",
3.   "method": "action.update",
4.   "params": {
5.     "actionid": "2",
6.     "status": "1"
7.   },
8.   "auth": "038e1d7b1735c6a5436ee9eae095879e",

```

```
9.      "id": 1
10. }
```

Response:

```
1. {
2.   "jsonrpc": "2.0",
3.   "result": {
4.     "actionids": [
5.       "2"
6.     ]
7.   },
8.   "id": 1
9. }
```

See also

- [Action filter](#)
- [Action operation](#)

Source

CAction::update() in *frontends/php/include/classes/api/services/CAction.php*.

API 信息

这个类用于检索 API 相关信息

相关方法：

- `apiinfo.version` - 获取 Zabbix API 版本

apiinfo.version

说明

```
string apiinfo.version(array)
```

该方法用于获取 Zabbix API 版本。

参数

此方法可用于未经身份验证的用户，必须在发送 JSON-RPC 请求中不加“auth”参数的情况下调用。

(array) 该方法接受一个空的数组。

返回值

(string) 返回 Zabbix API 的版本。

从 Zabbix 2.0.4 版本开始，API 的版本与 Zabbix 的版本相匹配。

范例

获取 API 版本

获取 Zabbix API 版本。

请求：

```
1. {
2.     "jsonrpc": "2.0",
3.     "method": "apiinfo.version",
4.     "params": [],
5.     "id": 1
6. }
```

响应：

```
1. {
2.     "jsonrpc": "2.0",
3.     "result": "2.4.0",
4.     "id": 1
5. }
```

来源

CAPIInfo::version() in *frontends/php/include/classes/api/services/CAPIInfo.php*.

Correlation 关联

This class is designed to work with correlations. 此类别应用于关联性使用

Object references: 对象引用:

- [Correlation](#)

Available methods: 可用的方法:

- `correlation.create` - creating new correlations 创建新的关联
- `correlation.delete` - deleting correlations 删除关联
- `correlation.get` - retrieving correlations 获取关联
- `correlation.update` - updating correlations 更新关联

> Correlation object 关联对象

The following objects are directly related to the [correlation API](#). 以下对象与“相关”API直接相关。

Correlation关联

The correlation object has the following properties. 相关对象具有以下属性。

属性	类型	说明
correlationid	string	(readonly) ID of the correlation. 关联的ID。
name (required)	string	Name of the correlation. 关联的名称。
description	string	Description of the correlation. 关联的说明
status	integer	Whether the correlation is enabled or disabled. 关联是否被启用 Possible values are: 可能的值是: 0 - (default)默认 enabled; 启用 1 - disabled. 禁用

Correlation operation 关联操作

The correlation operation object defines an operation that will be performed when a correlation is executed. It has the following properties. 相关操作对象定义当执行相关时将执行的操作。 它具有以下属性。

属性	类型	说明
type (required)	integer	Type of operation. 操作类型 Possible values: 可能的值是: 0 - close old events; 关闭旧事件 1 - close new event. 关闭新事件

Correlation filter关联筛选

The correlation filter object defines a set of conditions that must be met to perform the configured correlation operations. It has the following properties. 相关性过滤器对象定义了一组必须满足的条件来执行配置的相关操作。 它具有以下属性。

属性	类型	说明
evaltype (required)	integer	Filter condition evaluation method. 过滤条件评估方法。可能的值: 0 - and/or; 与/或 1 - and; 与 2 - or; 或 3 - custom expression. 自定义表达式
conditions (required)	array	Set of filter conditions to use for filtering results. 用于过滤结果的一组过滤条件。
evalformula	string	(readonly) Generated expression that will be used for evaluating filter conditions. The expression contains IDs that reference specific filter conditions by its formulaid . The value of eval_formula is equal to the value of formula for filters with a custom expression. 生成的表达式将用于评估过滤条件。该表达式包含通过其“公式”引用特定过滤条件的

		ID。“eval_formula”的值等于具有自定义表达式的过滤器的“公式”值。
formula	string	User-defined expression to be used for evaluating conditions of filters with a custom expression. The expression must contain IDs that reference specific filter conditions by its <code>formulaid</code> . The IDs used in the expression must exactly match the ones defined in the filter conditions: no condition can remain unused or omitted. 用于自定义表达式用于评估过滤条件的用户定义表达式。表达式必须包含通过其'公式'引用特定过滤条件的ID。表达式中使用的ID必须与过滤条件中定义的ID完全匹配：无条件可以保留未使用或省略。 Required for custom expression filters. 需要自定义表达式过滤器。

Correlation filter condition关联过滤条件

The correlation filter condition object defines a specific condition that must be checked before running the correlation operations. 相关滤波条件对象定义在运行相关运算之前必须检查的特定条件。

属性	类别	说明
<code>type(required)</code>	integer	Type of condition. 条件类型 Possible values: 可能的值: 0 - old event tag; 旧事件标签 1 - new event tag; 新事件标签 2 - new event host group; 新事件主机组; 3 - event tag pair; 事件标签对 4 - old event tag value; 旧事件标签值; 5 - new event tag value. 新的事件标签值。
tag	string	Event tag (old or new). Required when type of condition is: 0, 1, 4, 5. 事件标签(旧的或新的)。 条件类型为0, 1, 4, 5时需要。
groupid	string	Host group ID. Required when type of condition is: 2. 主机组ID。 条件类型是必需的: 2。
oldtag	string	Old event tag. Required when type of condition is: 3. 旧事件标签 条件类型为3时需要。
newtag	string	Old event tag. Required when type of condition is: 3. 旧事件标签 条件类型为3时需要。
value	string	Event tag (old or new) value. Required when type of condition is: 4, 5. 事件标签(旧或新)值。 条件类型: 4, 5时需要。
formulaid	string	Arbitrary unique ID that is used to reference the condition from a custom expression. Can only contain capital-case letters. The ID must be defined by the user when modifying filter conditions, but will be generated anew when requesting them afterward. 用于从自定义表达式引用条件的任意唯一ID。只能包含大写字母。修改过滤条件时，用户必须定义ID，但是以后请求时会重新生成。
operator	integer	Condition operator. 条件运算符。 Required when type of condition is: 2, 4, 5. 条件类型为2, 4, 5时需要。

To better understand how to use filters with various types of expressions, see examples on the `correlation.get` and `correlation.create` method pages. 要更好地了解如何使用具有各种类型表达式的过滤器，请参阅correlation.get和correlation.create方法页面上的示例。

The following operators and values are supported for each condition type. 每个条件类型都支持以下运算符和值。

持以下运算符和值。

条件	条件名称	Supported operators 支持的运算符	Expected value 期望值
2	Host group	=, <>	Host group ID. 主机组ID。
4	Old event tag value	=, <>, like, not like	string 字符串
5	New event tag value	=, <>, like, not like	string 字符串

correlation.create

Description 说明

```
object correlation.create(object/array correlations)
```

This method allows to create new correlations. 该方法允许创建新的关联

Parameters 参数

(object/array) Correlations to create. 关联创建。

Additionally to the [standard correlation properties](#), the method accepts the following parameters. 除了标准的相关属性，该方法接受以下参数。

参数	类型	说明
operations (required)	array	Correlation operations to create for the correlation. 创建关联的关联操作
filter (required)	object	Correlation filter object for the correlation. 关联的关联过滤对象

Return values 返回值

(object) Returns an object containing the IDs of the created correlations under the `correlationids` property. The order of the returned IDs matches the order of the passed correlations. 返回一个对象，该对象包含“相关性”属性下创建的关联的ID。返回的ID的顺序与通过的相关性的顺序相匹配。

Examples 例子

Create a new event tag correlation 创建一个新的事件标签关联

Create a correlation using evaluation method [AND/OR](#) with one condition and one operation. By default the correlation will be enabled. 使用评估方法“AND / OR”创建一个相关性，具有一个条件和一个操作。默认情况下，关联将被启用。

Request:

```

1. {
2.   "jsonrpc": "2.0",
3.   "method": "correlation.create",
4.   "params": {
5.     "name": "new event tag correlation",
6.     "filter": {
7.       "evaltype": 0,
8.       "conditions": [

```

```

9.          {
10.            "type": 1,
11.            "tag": "ok"
12.          }
13.        ]
14.      },
15.      "operations": [
16.        {
17.          "type": 0
18.        }
19.      ],
20.    },
21.    "auth": "343baad4f88b4106b9b5961e77437688",
22.    "id": 1
23.  }

```

Response:

```

1.  {
2.    "jsonrpc": "2.0",
3.    "result": {
4.      "correlationids": [
5.        "1"
6.      ]
7.    },
8.    "id": 1
9.  }

```

Using a custom expression filter 使用自定义表达式过滤器

Create a correlation that will use a custom filter condition. The formula IDs “A” or “B” have been chosen arbitrarily. Condition type will be “Host group” with operator “<>”. 创建一个使用自定义过滤条件的关联。公式ID“A”或“B”已被任意选择。条件类型将为“主机组”与操作员“<>”。

Request:

```

1.  {
2.    "jsonrpc": "2.0",
3.    "method": "correlation.create",
4.    "params": {
5.      "name": "new host group correlation",
6.      "description": "a custom description",
7.      "status": 0,
8.      "filter": {
9.        "evaltype": 3,
10.       "formula": "A or B",
11.       "conditions": [
12.         {
13.           "type": 2,
14.           "operator": 1,
15.           "formulaid": "A"

```

```

16.          },
17.          {
18.              "type": 2,
19.              "operator": 1,
20.              "formulaid": "B"
21.          }
22.      ],
23.  },
24.  "operations": [
25.      {
26.          "type": 1
27.      }
28.  ],
29. },
30. "auth": "343baad4f88b4106b9b5961e77437688",
31. "id": 1
32. }
```

Response:

```

1. {
2.     "jsonrpc": "2.0",
3.     "result": {
4.         "correlationids": [
5.             "2"
6.         ],
7.     },
8.     "id": 1
9. }
```

See also参见

- [Correlation filter](#)
- [Correlation operation](#)

Source来源

`CCorrelation::create()` in `frontends/php/include/classes/api/services/CCorrelation.php`.

correlation.delete

Description说明

```
object correlation.delete(array correlationids)
```

This method allows to delete correlations.该方法允许删除关联。

Parameters 参数

(array) IDs of the correlations to delete.要删除的关联的ID

Return values返回值

(object) Returns an object containing the IDs of the deleted correlations under the correlationids property.返回一个对象，该对象包含“相关性”属性下删除的关联的ID。

Example示例

Delete multiple correlations删除多个关联

Delete two correlations.删除两个关联

Request:

```
1. {
2.   "jsonrpc": "2.0",
3.   "method": "correlation.delete",
4.   "params": [
5.     "1",
6.     "2"
7.   ],
8.   "auth": "343baad4f88b4106b9b5961e77437688",
9.   "id": 1
10. }
```

Response:

```
1. {
2.   "jsonrpc": "2.0",
3.   "result": {
4.     "correlationids": [
5.       "1",
6.       "2"
7.     ]
8.   },
9.   "id": 1
}
```

correlation.delete

```
10. }
```

Source来源

CCorrelation::delete() in *frontends/php/include/classes/api/services/CCorrelation.php*.

correlation.get

Description

```
integer/array correlation.get(object parameters)
```

The method allows to retrieve correlations according to the given parameters. 该方法允许根据给定的参数获取关联。

Parameters 参数

(object) Parameters defining the desired output. 定义所需输出的参数。

The method supports the following parameters. 该方法支持以下参数。

参数	类型	描述
correlationids	string/array	Return only correlations with the given IDs. 仅返回给定ID的关联。
selectFilter	query	Returns the correlation filter in the <code>filter</code> property. 返回“filter”属性中的关联过滤器。
selectOperations	query	Return correlation operations in the <code>operations</code> property. 返回“操作”属性中的关联操作。
sortfield	string/array	Sort the result by the given properties. 按照给定的属性对结果进行排序 Possible values are: <code>correlationid</code> , <code>name</code> and <code>status</code> . 可能的值有: <code>correlationid</code> , <code>name</code> 和 <code>status</code> 。
countOutput	flag	These parameters being common for all <code>get</code> methods are described in the reference commentary . 这些参数对于所有的“get”方法都是常见的，在参考文献中有描述
editable	boolean	
excludeSearch	flag	
filter	object	
limit	integer	
output	query	
preservekeys	flag	
search	object	
searchByAny	boolean	
searchWildcardsEnabled	boolean	
sortorder	string/array	
startSearch	flag	

Return values 返回值

(integer/array) Returns either:

- an array of objects; 一组对象;
- the count of retrieved objects, if the `countOutput` parameter has been used. 如果已经使用“countOutput”参数，则检索到的对象的计数。

Examples 示例

Retrieve correlations 获取关联

Retrieve all configured correlations together with correlation conditions and operations. The filter uses the “and/or” evaluation type, so the `formula` property is empty and `eval_formula` is generated automatically. 获取所有配置的关联以及关联条件和操作。过滤器使用“和/或”评估类型，因此 `formula` 属性为空，“eval_formula”是自动生成的。

Request:

```

1. {
2.   "jsonrpc": "2.0",
3.   "method": "correlation.get",
4.   "params": {
5.     "output": "extend",
6.     "selectOperations": "extend",
7.     "selectFilter": "extend"
8.   },
9.   "auth": "343baad4f88b4106b9b5961e77437688",
10.  "id": 1
11. }
```

Response:

```

1. {
2.   "jsonrpc": "2.0",
3.   "result": [
4.     {
5.       "correlationid": "1",
6.       "name": "Correlation 1",
7.       "description": "",
8.       "status": "0",
9.       "filter": {
10.         "evaltype": "0",
11.         "formula": "",
12.         "conditions": [
13.           {
14.             "type": "3",
15.             "oldtag": "error",
16.             "newtag": "ok",

```

```
17.           "formulaid": "A"
18.       }
19.     ],
20.     "eval_formula": "A"
21.   },
22.   "operations": [
23.     {
24.       "type": "0"
25.     }
26.   ]
27. },
28. ],
29. "id": 1
30. }
```

See also 参见

- [Correlation filter](#)
- [Correlation operation](#)

Source 来源

CCorrelation::get() in *frontends/php/include/classes/api/services/CCorrelation.php*.

correlation.update

Description 说明

```
object correlation.update(object/array correlations)
```

This method allows to update existing correlations. 该方法允许更新现有的相关性。

Parameters 参数

(object/array) Correlation properties to be updated. 要更新的相关属性。

The `correlationid` property must be defined for each correlation, all other properties are optional. Only the passed properties will be updated, all others will remain unchanged. 必须为每个关联定义“相关性”属性，所有其他属性都是可选的。只有超时的属性将被更新，所有其他属性将保持不变。

Additionally to the `standard correlation properties`, the method accepts the following parameters. 除标准相关性外，该方法接受以下参数。

参数	类型	说明
filter	object	Correlation filter object to replace the current filter. 替换当前过滤器的关联过滤对象。
operations	array	Correlation operations to replace existing operations. 替换现有操作的关联操作。

Return values 返回值

(object) Returns an object containing the IDs of the updated correlations under the `correlationids` property. 返回一个对象，该对象包含“相关性”属性下更新的相关性的ID。

Examples示例

Disable correlation 禁用关联

Request:

```

1. {
2.   "jsonrpc": "2.0",
3.   "method": "correlation.update",
4.   "params": {
5.     "correlationid": "1",
6.     "status": "1"
7.   },
8.   "auth": "343baad4f88b4106b9b5961e77437688",
9.   "id": 1

```

```
10. }
```

Response:

```
1. {
2.   "jsonrpc": "2.0",
3.   "result": {
4.     "correlationids": [
5.       "1"
6.     ],
7.   },
8.   "id": 1
9. }
```

Replace conditions, but keep the evaluation method 替换条件，但保持评估方法

Request:

```
1. {
2.   "jsonrpc": "2.0",
3.   "method": "correlation.update",
4.   "params": {
5.     "correlationid": "1",
6.     "filter": {
7.       "conditions": [
8.         {
9.           "type": 3,
10.          "oldtag": "error",
11.          "newtag": "ok"
12.        }
13.      ]
14.    }
15.  },
16.  "auth": "343baad4f88b4106b9b5961e77437688",
17.  "id": 1
18. }
```

Response:

```
1. {
2.   "jsonrpc": "2.0",
3.   "result": {
4.     "correlationids": [
5.       "1"
6.     ],
7.   },
8.   "id": 1
9. }
```

See also参见

- [Correlation filter](#)
- [Correlation operation](#)

Source来源

`CCorrelation::update()` in *frontends/php/include/classes/api/services/CCorrelation.php*.

Graph item图形项

This class is designed to work with hosts. 这个类与主机

Object references对象引用:

- [Graph item图形监控项](#)

Available methods可用方法:

- `graphitem.get` - retrieving graph items检索图形项

> Graph item object 图形项目对象

The following objects are directly related to the `graphitem` API. 以下对象是 `graphitem` API 有直接关系

Graph item 图形项目

Graph items can only be modified via the `graph` API. 图形项目只能通过 `graph` API 修改。

The graph item object has the following properties. 图形项目对象拥有以下参数（属性）。

Property 参数	Type 类型	Description 描述
图形项目ID <code>gitemid</code>	string 字符串	(<i>readonly</i> 只读) ID of the graph item. 图形项目的ID。
颜色 <code>color</code> (required 必要)	string 字符串型	Graph item's draw color as a hexadecimal color code. 绘制图形项目的颜色，使用十六进制码表示
监控项ID <code>itemid</code> (required 必要)	string 字符串型	ID of the item. 监控项的ID。
计算方式 <code>calcfnc</code>	integer 整数型	Value of the item that will be displayed. 监控项的显示的值 Possible values: 可用值: 1 - minimum value 最小值; 2 - (default 默认) average value 平均值; 4 - maximum value 最大值; 7 - all values 所有值; 9 - last value 最后的值, used only by pie and exploded graphs. 仅用于饼图和分散饼图
绘图线型 <code>drawtype</code>	integer 整数型	Draw style of the graph item. 绘制图形项目的线型 Possible values: 可用值: 0 - (default 默认) line 实线; 1 - filled region 面积图(填满的区域); 2 - bold line 粗实线; 3 - dot 点; 4 - dashed line 虚线; 5 - gradient line (梯度线).
图形ID <code>graphid</code>	string 字符串	ID of the graph that the graph item belongs to. 图形项目所属图形的ID
排序码 <code>sortorder</code>	integer 整数型	Position of the item in the graph. 图形中项目的排序 Default: starts with 0 and increases by one with each entry. 默认从零开始，每增加一个加1.
类型 <code>type</code>	integer 整数型	Type of graph item. 图形类型 Possible values: 可用值: 0 - (default 默认) simple 简单图形; 2 - graph sum 汇总图形, used only by pie and exploded graphs. 仅用于饼图和分散饼图
Y轴位置 <code>yaxisside</code>	integer 整数型	Side of the graph where the graph item's Y scale will be drawn. 图形项目的Y轴画在图像的位置。 Possible values 可用值: 0 - (default 默认) left side 左侧; 1 - right side 右侧.

graphitem.get

Description 描述

`integer/array graphitem.get(object parameters)`

The method allows to retrieve graph items according to the given parameters. 此方法接受以下参数

Parameters 参数

`(object)` Parameters defining the desired output. `(关联数组)` 定义所需输出的参数

The method supports the following parameters. 此方法支持以下参数：

Parameter参数	Type类型	Description描述
图形项目ID gitemids	string/array 字符串型/数组	Return only graph items with the given IDs. 只返回指定ID的图形项目。
图形ID graphids	string/array 字符串型/数组	Return only graph items that belong to the given graphs. 只返回属于指定图形的图形项目。
监控项ID itemid	string/array 字符串型/数组	Return only graph items with the given item IDs. 只返回指定监控项ID的图形项目
类型 type	integer整数型	Return only graph items with the given type. 只返回指定类型的图形项目。Refer to the graph item object page for a list of supported graph item types. 关于支持的图形项目类型，参见 图形项目对象的列表
选择图形 selectGraphs	query 请求	Return the graph that the item belongs to as an array in the <code>graphs</code> property. 返回 <code>graphs</code> 参数正确的图像
排序条件 sortfield	string/array 字符串型/数组	Sort the result by the given properties. 按指定的参数排序Possible values are: <code>gitemid</code> . 可用值为: <code>gitemid</code>
countOutput	flag	These parameters being common for all <code>get</code> methods are described in detail in the reference commentary page . 这些参数对于所有get方法都是通用的，在附录1：参考中，查看详细描述。
editable	boolean	
limit	integer	
output	query	
preservekeys	flag	
sortorder	string/array	

Return values 返回值

`(integer/array)` Returns either: 返回整数或数组:

- an array of objects 返回一个对象数组;
- the count of retrieved objects, if the `countOutput` parameter has been used. 如果 `countOutput` 被使用, 返回检索对象的计数。

Examples示例

Retrieving graph items from a graph 从图形中检索图形项目

Retrieve all graph items used in a graph with additional information about the item and the host. 检索被图形使用的所有图形项目包含监控项和主机的额外信息

Request请求:

```

1. {
2.   "jsonrpc": "2.0",
3.   "method": "graphitem.get",
4.   "params": {
5.     "output": "extend",
6.     "graphids": "387"
7.   },
8.   "auth": "038e1d7b1735c6a5436ee9eae095879e",
9.   "id": 1
10. }
```

Response相应:

```

1. {
2.   "jsonrpc": "2.0",
3.   "result": [
4.     {
5.       "gitemid": "1242",
6.       "graphid": "387",
7.       "itemid": "22665",
8.       "drawtype": "1",
9.       "sortorder": "1",
10.      "color": "FF5555",
11.      "yaxisside": "0",
12.      "calc_fnc": "2",
13.      "type": "0",
14.      "key_": "system.cpu.util[,steal]",
15.      "hostid": "10001",
16.      "flags": "0",
17.      "host": "Template OS Linux"
18.    },
19.    {
20.      "gitemid": "1243",
21.      "graphid": "387",
22.      "itemid": "22668",
23.      "drawtype": "1",
24.      "sortorder": "2",
25.    }
26.  ]
27. }
```

```
25.         "color": "55FF55",
26.         "yaxisside": "0",
27.         "calc_fnc": "2",
28.         "type": "0",
29.         "key_": "system.cpu.util[,softirq]",
30.         "hostid": "10001",
31.         "flags": "0",
32.         "host": "Template OS Linux"
33.     },
34.     {
35.         "gitemid": "1244",
36.         "graphid": "387",
37.         "itemid": "22671",
38.         "drawtype": "1",
39.         "sortorder": "3",
40.         "color": "009999",
41.         "yaxisside": "0",
42.         "calc_fnc": "2",
43.         "type": "0",
44.         "key_": "system.cpu.util[,interrupt]",
45.         "hostid": "10001",
46.         "flags": "0",
47.         "host": "Template OS Linux"
48.     }
49. ],
50. "id": 1
51. }
```

See also 参见

- [Graph图形](#)

Source来源

`CGraphItem::get()` in *frontends/php/include/classes/api/services/CGraphItem.php*.

Graph prototype 原型图

This class is designed to work with graph prototypes. 这个类用于操作图片原型

Object references对象引用:

- [Graph prototype图形原型](#)

Available methods可用方法:

- `graphprototype.create` - creating new graph prototypes创建一个新的图形原型
- `graphprototype.delete` - deleting graph prototypes删除一个图形原型
- `graphprototype.get` - retrieving graph prototypes检索图形原型
- `graphprototype.update` - updating graph prototypes更新图形原型

> Graph prototype object 图形原型对象

The following objects are directly related to the `graphprototype` API. 以下对象与 `graphprototype` API 有直接关系。

Graph prototype 图形原型

The graph prototype object has the following properties. 图形原型对象有以下参数(属性)

Property 参数	Type 类型	Description 描述
图形ID <code>graphid</code>	<code>string</code> 字符串型	(<code>readonly</code> 只读) ID of the graph prototype. 图形原型的ID
高度 <code>height</code> (required 必要)	<code>integer</code> 整数型	Height of the graph prototype in pixels. 图形原型的高度(单位: 像素)
名称 <code>name</code> (required 必要)	<code>string</code> 字符串型	Name of the graph prototype. 图形原型的名称
宽度 <code>width</code> (required 必要)	<code>integer</code> 整数型	Width of the graph prototype in pixels. 图形原型的宽度(单位: 像素)
图形类型 <code>graphtype</code>	<code>integer</code> 整数型	Graph prototypes's layout type. 图形原型布局类型 Possible values 可用值: 0 - (<code>default</code> 默认) normal 常规; 1 - stacked 堆积图; 2 - pie 饼图; 3 - exploded 分散饼图.
百分比左 <code>percent_left</code>	<code>float</code> 浮点数型	Left percentile. 左侧百分比线 Default 默认值: 0.
百分比右 <code>percent_right</code>	<code>float</code> 浮点数型	Right percentile. 右侧百分比线 Default 默认值: 0.
3D展示 <code>show_3d</code>	<code>integer</code> 整数型	Whether to show discovered pie and exploded graphs in 3D. 是否使用3D形式展示被发现的饼图和分散饼图 Possible values 可用值: 0 - (<code>default</code> 默认) <code>show in 2D</code> 2D形式展示; 1 - <code>show in 3D</code> 3D形式展示.
图例显示 <code>show_legend</code>	<code>integer</code> 整数型	Whether to show the legend on the discovered graph. 是否在被发现的图形上显示图例 Possible values 可用值: 0 - <code>hide</code> 隐藏; 1 - (<code>default</code> 默认值) <code>show</code> 显示.
显示工作时间 <code>show_work_period</code>	<code>integer</code> 整数型	Whether to show the working time on the discovered graph. 是否在发现的图形上显示工作时间 Possible values 可用值: 0 - <code>hide</code> 隐藏; 1 - (<code>default</code> 默认值) <code>show</code> 显示.
模板ID <code>templateid</code>	<code>string</code> 字符串	(<code>readonly</code> 只读) ID of the parent template graph prototype. 图形圆形的父模板的ID
Y轴最大值 <code>yaxismax</code>	<code>float</code> 浮点数型	The fixed maximum value for the Y axis. Y轴的固定最大值
Y轴最小值 <code>yaxismin</code>	<code>float</code> 浮点数型	The fixed minimum value for the Y axis. Y轴的固定最小值
Y轴最大值监控项ID <code>ymax_itemid</code>	<code>string</code> 字符串	ID of the item that is used as the maximum value for the Y axis. 用于作为Y轴最大值的监控项ID
		Maximum value calculation method for the Y axis. Y轴最

ymax_type	整数型	大值的计算方式 Possible values可用值: 0 - (default默认值) _calculated计算的; 1 - fixed固定的; 2 - item监控项.
Y轴最小值监控项ID ymin_itemid	string 字符串型	ID of the item that is used as the minimum value for the Y axis. 用于作为Y轴最小值的监控项ID
Y轴最小值类型 ymin_type	integer 整数型	Minimum value calculation method for the Y axis. Y轴最小值的计算方式 Possible values可用值: 0 - (default默认值) _calculated计算的; 1 - fixed固定的; 2 - item监控项.

graphprototype.create

Description描述

```
object graphprototype.create(object/array graphPrototypes)
```

This method allows to create new graph prototypes.这个方法用于创建新的图形原型

Parameters参数

(object/array) Graph prototypes to create. (关联数组/数组) 创建图形原型

Additionally to the standard graph prototype properties, the method accepts the following parameters.此方法除了接受标准图形原型参数外，还接受以下参数：

Parameter参数	Type类型	Description描述
gitems (required 必要)	array 数组	Graph items to be created for the graph prototypes.创建到图形原型中的监控项 Graph items can reference both items and item prototypes, but at least one item prototype must be present.图形监控项能同时被监控项与监控项原型检索到，但必须至少有一个监控项原型。

Return values可用值

(object) Returns an object containing the IDs of the created graph prototypes under the `graphids` property. The order of the returned IDs matches the order of the passed graph prototypes.返回一个包含所创建的图形原型的ID的关联数组在 `graphids` 参数下。返回的ID的顺序与通过的图形原型项匹配。

Examples示例

Creating a graph prototype创建一个图形原型

Create a graph prototype with two items.创建一个含有两个监控项的图形原型

Request请求：

```

1. {
2.   "jsonrpc": "2.0",
3.   "method": "graphprototype.create",
4.   "params": {
5.     "name": "Disk space usage {#FSNAME}",
6.     "width": 900,
7.     "height": 200,
8.     "gitems": [
9.       {

```

```
10.         "itemid": "22828",
11.         "color": "00AA00"
12.     },
13.     {
14.         "itemid": "22829",
15.         "color": "3333FF"
16.     }
17. ],
18. },
19. "auth": "038e1d7b1735c6a5436ee9eae095879e",
20. "id": 1
21. }
```

Response响应：

```
1. {
2.     "jsonrpc": "2.0",
3.     "result": {
4.         "graphids": [
5.             "652"
6.         ]
7.     },
8.     "id": 1
9. }
```

See also参见

- [Graph item](#)

Source来源

CGraphPrototype::create() in
frontends/php/include/classes/api/services/CGraphPrototype.php.

graphprototype.delete

Description描述

```
object graphprototype.delete(array graphPrototypeIds)
```

This method allows to delete graph prototypes.此方法允许删除图形原型。

Parameters参数

(array) IDs of the graph prototypes to delete. (array) 需要删除的图形原型的ID。

Return values返回值

(object) Returns an object containing the IDs of the deleted graph prototypes under the graphids property.

(object) 返回一个包含被删除的图形原型的ID的关联数组在 graphids 参数下。

Examples示例

Deleting multiple graph prototypes删除多个图形原型

Delete two graph prototypes.删除连个图形原型。

Request请求:

```
1. {
2.   "jsonrpc": "2.0",
3.   "method": "graphprototype.delete",
4.   "params": [
5.     "652",
6.     "653"
7.   ],
8.   "auth": "3a57200802b24cda67c4e4010b50c065",
9.   "id": 1
10. }
```

Response响应:

```
1. {
2.   "jsonrpc": "2.0",
3.   "result": {
4.     "graphids": [
5.       "652",
6.       "653"
7.     ]
8.   }
9. }
```

graphprototype.delete

```
8.      },
9.      "id": 1
10. }
```

Source参见

CGraphPrototype::delete() in
frontends/php/include/classes/api/services/CGraphPrototype.php.

graphprototype.get

Description描述

```
integer/array graphprototype.get(object parameters)
```

The method allows to retrieve graph prototypes according to the given parameters. 此方法允许按照给定的参数检索图形原型。

Parameters参数

(object) Parameters defining the desired output. (object) 参数定义需要的输出

The method supports the following parameters. 此方法支持以下参数。

Parameter参数	Type类型	Description描述
发现ID discoveryids	string字符串型/array数组型	Return only graph prototypes that belong to the given discovery rules. 只返回属于指定自动发现规则的图形原型
图形ID graphids	string字符串型/array数组型	Return only graph prototypes with the given IDs. 只返回含有给定ID的图形原型
组ID groupids	string字符串型/array数组型	Return only graph prototypes that belong to hosts in the given host groups. 只返回属于指定主机组的主机的图形原型
主机ID hostids	string字符串型/array数组型	Return only graph prototypes that belong to the given hosts. 只返回属于指定主机的图形原型
继承的 inherited	boolean布尔型	If set to <code>true</code> return only graph prototypes inherited from a template. 如果设置此参数为 <code>true</code> ，只返回从一个模板继承的图形原型。
监控项ID itemids	string字符串型/array数组型	Return only graph prototypes that contain the given item prototypes. 只返回包含给定监控项参数的图形原型
模板化的 templated	boolean布尔型	If set to <code>true</code> return only graph prototypes that belong to templates. 如果此参数设置为 <code>true</code> ，只返回属于模板的图形原型。
模板ID templateids	string字符串型/array数组型	Return only graph prototypes that belong to the given templates. 只返回属于指定模板的图形原型。
selectDiscoveryRule	query	Return the LLD rule that the graph prototype belongs to in the <code>discoveryRule</code> property. 返回图形原型中 <code>discoveryRule</code> 参数内的低级发现规则
selectGraphItems	query	Return the graph items used in the graph prototype in the <code>gitems</code> property. 返回图形原型中 <code>gitems</code> 参数内被使用的图形项目
		Return the host groups that the graph

selectGroups	query	prototype belongs to in the <code>groups</code> property. 返回图形原型中 <code>groups</code> 参数内的主机组
selectHosts	query	Return the hosts that the graph prototype belongs to in the <code>hosts</code> property. 返回图形原型中 <code>hosts</code> 参数内的主机
selectItems	query	Return the items and item prototypes used in the graph prototype in the <code>items</code> property. 返回图形原型中 <code>itemss</code> 参数内的监控项
selectTemplates	query	Return the templates that the graph prototype belongs to in the <code>templates</code> property. 返回图形原型 <code>templatess</code> 参数内的模板
filter	object关联数组	Return only those results that exactly match the given filter. 只返回与给定过滤器完全匹配的结果。 \Accepts an array, where the keys are property names, and the values are either a single value or an array of values to match against. 接受一个数组，其中的键是属性名，值可以是一个值或者是一个匹配的值数组。 \Supports additional filters: 支持额外过滤器： <code>host</code> - technical name of the host that the graph belongs to; 图形所属主机的名称 <code>hostid</code> - ID of the host that the graph belongs to. 图形所属主机的ID
sortfield	string字符串型/array数组型	Sort the result by the given properties. 按指定的参数将结果排序。Possible values are: 可用值： <code>图形ID graphid</code> , <code>名称 name</code> and <code>图形类型 graphtype</code> .
countOutput	flag	These parameters being common for all <code>get</code> methods are described in detail in the reference commentary . 这些参数对于所有 <code>get</code> 方法都是通用的，在 附录1：参考 中，查看详细描述。
editable	boolean	
excludeSearch	flag	
limit	integer	
output	query	
preservekeys	flag	
search	object	
searchByAny	boolean	
searchWildcardsEnabled	boolean	
sortorder	string/array	
startSearch	flag	

Return values

(integer/array) Returns either:

- an array of objects;
- the count of retrieved objects, if the `countOutput` parameter has been used.

(整数型/数组型) 返回两个中的一个:

- 一个数组对象;
- 如果 `countOutput` 参数被使用, 将对检索到的对象进行计数。

Examples示例

Retrieving graph prototypes from a LLD rule从一个低级发现规则检索图形原型

Retrieve all graph prototypes from an LLD rule.从一个低级发现规则检索图形原型

Request请求:

```

1. {
2.   "jsonrpc": "2.0",
3.   "method": "graphprototype.get",
4.   "params": {
5.     "output": "extend",
6.     "discoveryids": "27426"
7.   },
8.   "auth": "038e1d7b1735c6a5436ee9eae095879e",
9.   "id": 1
10. }
```

Response响应:

```

1. {
2.   "jsonrpc": "2.0",
3.   "result": [
4.     {
5.       "graphid": "1017",
6.       "parent_itemid": "27426",
7.       "name": "Disk space usage {#FSNAME}",
8.       "width": "600",
9.       "height": "340",
10.      "yaxismin": "0.0000",
11.      "yaxismax": "0.0000",
12.      "templateid": "442",
13.      "show_work_period": "0",
14.      "show_triggers": "0",
15.      "graphtype": "2",
16.      "show_legend": "1",
17.      "show_3d": "1",
18.      "percent_left": "0.0000",
19.      "percent_right": "0.0000",
20.      "ymin_type": "0",
21.      "ymax_type": "0",
22.      "ymin_itemid": "0",
23.      "ymax_itemid": "0"
```

```
24.         }
25.     ],
26.     "id": 1
27. }
```

See also参见

- [Discovery rule](#)
- [Graph item](#)
- [Item](#)
- [Host](#)
- [Host group](#)
- [Template](#)

Source来源

CGraphPrototype::get() in
frontends/php/include/classes/api/services/CGraphPrototype.php.

graphprototype.update

Description描述

```
object graphprototype.update(object/array graphPrototypes)
```

This method allows to update existing graph prototypes.此方法允许更新已经存在的图形项目。

Parameters参数

(object/array) Graph prototype properties to be updated. (关联数组/数组) 需要更新的图形原型参数。

The `graphid` property must be defined for each graph prototype, all other properties are optional. Only the passed properties will be updated, all others will remain unchanged.必须定义每一个图形原型的 `graphid` 参数，所有其他的参数都是可选的。只有通过的参数将被更新，其他的参数会保持不变。Additionally to the standard graph prototype properties, the method accepts the following parameters.除了标准图形原型参数外，此方法还接受以下参数。

Parameter 参数	Type 类型	Description描述
gitems	array	Graph items to replace existing graph items.图形项目将替换已经存在的图形项。 If a graph item has the <code>gitemid</code> property defined it will be updated, otherwise a new graph item will be created.如果定义一个已经存在“gitemid”参数的图形项，那么“gitemid”参数将被更新，否则一个新的图形项将被创建。

Return values返回值

(object) Returns an object containing the IDs of the updated graph prototypes under the `graphids` property.返回一个关联数组，包含被更新的图片原型的ID参数。

Examples示例

Changing the size of a graph prototype改变一个图形原型的尺寸

Change the size of a graph prototype to 1100 to 400 pixels.将一个图形原型的尺寸修改成宽1100像素高400像素。

Request请求：

```

1. {
2.   "jsonrpc": "2.0",
3.   "method": "graphprototype.update",
4.   "params": {
5.     "graphid": "439",
6.     "width": 1100,

```

```
7.      "height": 400
8.    },
9.    "auth": "038e1d7b1735c6a5436ee9eae095879e",
10.   "id": 1
11. }
```

Response响应：

```
1. {
2.   "jsonrpc": "2.0",
3.   "result": {
4.     "graphids": [
5.       "439"
6.     ]
7.   },
8.   "id": 1
9. }
```

Source来源

CGraphPrototype::update() in
frontends/php/include/classes/api/services/CGraphPrototype.php.

Graph 图形

This class is designed to work with items. 这个类与“监控项”配合使用。

Object references: 对象引用:

- [Graph](#)

Available methods: 可用方法:

- `graph.create` - creating new graphs 创建新的图形
- `graph.delete` - deleting graphs 删除图形
- `graph.get` - retrieving graphs 检索图形
- `graph.update` - updating graphs 更新图形

> Graph object图形对象

The following objects are directly related to the `graph` API. 以下对象与“图形”API有关。

Graph

The graph object has the following properties.“graph”对象具有以下参数。

Property参数	Type类型	Description描述
图形ID graphid	string 字符串型	(readonly只读) ID of the graph. 图形的ID
高度 height (required 必要)	integer 整型	Height of the graph in pixels. 图形的高度(单位: 像素)
名称 name (required必 要)	string 字符串型	Name of the graph. 图形的名称
宽度 width (required必 要)	integer 整型	Width of the graph in pixels. 图形的宽度(单位: 像素)
标志 flags	integer 整型	(readonly只读) Origin of the graph. 图形的来源Possible values are: 可用值: 0 - (default默认值) a plain graph; 一幅简图; 4 - a discovered graph. 一副发现图
图形类型 graphtype	integer 整型	Graph's layout type. 图形类别。Possible values: 可用值: 0 - (default默认值) normal常规; 1 - stacked堆积图; 2 - pie饼图; 3 - exploded. 分散饼图
百分比左 percent_left	float浮 点型	Left percentile. 百分比线(左) Default: 默认值: 0.
百分比右 percentright	float浮 点型	Right percentile. 百分比线(右) Default: 默认值: 0.
3D展示 show_3d	integer 整型	Whether to show pie and exploded graphs in 3D. 是否以3D图形展示饼图和分散饼图 Possible values: 可用值: 0 - (default默认值) show in 2D; 以2D图展示 1 - show in 3D. 以3D图展示
显示图例 show_legend	integer 整型	Whether to show the legend on the graph. 是否在图形上显示图例Possible values: 可用值: 0 - hide; 隐藏 1 - (default默认值) show. 显示
显示工作时间 show_work_period	integer 整型	Whether to show the working time on the graph. 是否在图形上显示工作时间 Possible values: 可用值: 0 - hide; 隐藏 1 - (default默认值) show. 显示
模板ID templateid	string 字符串型	(readonly只读) ID of the parent template graph. 父模板图形ID
Y轴最大值 yaxismax	float浮 点型	The fixed maximum value for the Y axis. Y轴的固定最大值 Default: 默认值: 100
Y轴最小值 yaxismin	float浮 点型	The fixed minimum value for the Y axis. Y轴的固定最小值 Default: 默认值: 0
Y轴最大值监控项ID	string	ID of the item that is used as the maximum value for

ymax_itemid	字符串型	the Y axis.用于作为Y轴最大值的监控项ID
Y轴最大值类型 ymax_type	integer 整型	Maximum value calculation method for the Y axis.Y轴最大值的计算方式 Possible values可用值: 0 - (default默认值) _calculated; 可计算的 1 - fixed; 固定值 2 - item监控项.
Y轴最小值监控项ID ymin_itemid	string 字符串型	ID of the item that is used as the minimum value for the Y axis.用于作为Y轴最小值的监控项ID
Y轴最小值类型 ymin_type	integer 整型	Minimum value calculation method for the Y axis. Y轴最小值的计算方式Possible values可用值: 0 - (default默认值) _calculated可计算的; 1 - fixed固定值; 2 - item监控项.

graph.create

Description描述

```
object graph.create(object/array graphs)
```

This method allows to create new graphs.此方法允许创建新的图形。

Parameters参数

(object/array) Graphs to create. (对象/数组) 创建图形.

Additionally to the [standard graph properties](#), the method accepts the following parameters.除标准图形参数外，此方法还接受以下参数。

Parameter参数	Type类型	Description描述
gitems(required必要)	array数组	Graph items to be created for the graph.创建到图形中的监控项

Return values返回值

(object) Returns an object containing the IDs of the created graphs under the graphids property. The order of the returned IDs matches the order of the passed graphs. (关联数组) 返回一个对象包含，属于所创建的图形下的 graphids 参数的ID。返回的ID的顺序与成功创建的图形的顺序匹配。

Examples示例

Creating a graph创建一个图形

Create a graph with two items.创建一个有两个参数的图形

Request请求：

```

1. {
2.   "jsonrpc": "2.0",
3.   "method": "graph.create",
4.   "params": {
5.     "name": "MySQL bandwidth",
6.     "width": 900,
7.     "height": 200,
8.     "gitems": [
9.       {
10.         "itemid": "22828",
11.         "color": "#00AA00"
12.       },
13.       {

```

```
graph.create
```

```
14.         "itemid": "22829",
15.         "color": "3333FF"
16.     }
17.   ]
18. },
19. "auth": "038e1d7b1735c6a5436ee9eae095879e",
20. "id": 1
21. }
```

响应：

```
1. {
2.   "jsonrpc": "2.0",
3.   "result": {
4.     "graphids": [
5.       "652"
6.     ]
7.   },
8.   "id": 1
9. }
```

See also参见

- [Graph item](#)

Source来源

`CGraph::create()` in *frontends/php/include/classes/api/services/CGraph.php*.

graph.delete

Description描述

```
object graph.delete(array graphIds)
```

This method allows to delete graphs. 此方法允许删除图形。

Parameters参数

(array) IDs of the graphs to delete. (数组) 需要被删除的图形的ID。

Return values返回值

(object) Returns an object containing the IDs of the deleted graphs under the graphids property. (对象) 返回一个对象，含有被删除的图像的 graphids 参数的ID。

Examples示例

Deleting multiple graphs删除多个图形

Delete two graphs. 删除两个图形

Request请求：

```
1. {
2.   "jsonrpc": "2.0",
3.   "method": "graph.delete",
4.   "params": [
5.     "652",
6.     "653"
7.   ],
8.   "auth": "3a57200802b24cda67c4e4010b50c065",
9.   "id": 1
10. }
```

Response响应：

```
1. {
2.   "jsonrpc": "2.0",
3.   "result": {
4.     "graphids": [
5.       "652",
6.       "653"
7.     ]
8.   },
9.   "id": 1
}
```

```
graph.delete
```

```
10. }
```

Source来源

`CGraph::delete()` in `frontends/php/include/classes/api/services/CGraph.php`.

graph.get

Description描述

```
integer/array graph.get(object parameters)
```

The method allows to retrieve graphs according to the given parameters.此方法允许根据给定的参数返回图形

Parameters参数

(object) Parameters defining the desired output. (对象) 参数定义需要的输出。

The method supports the following parameters.此方法支持如下参数

Parameter参数	Type类型	Description描述
图形ID graphids	string/array 字符串型/数组型	Return only graphs with the given IDs.返回只含有指定ID的图形
主机组ID groupids	string/array 字符串型/数组型	Return only graphs that belong to hosts in the given host groups.返回只属于指定主机组的主机的图形。
模板ID templateids	string/array 字符串型/数组型	Return only graph that belong to the given templates.返回只属于指定模板的图形。
主机ID hostids	string/array 字符串型/数组型	Return only graphs that belong to the given hosts.返回只属于指定主机的图形。
监控项ID itemids	string/array 字符串型/数组型	Return only graphs that contain the given items.返回只包含指定监控项的图形。
模板化的 templated	boolean 布尔型	If set to <code>true</code> return only graphs that belong to templates.如果设置“true”返回只属于模板的图形。
继承的 inherited	boolean 布尔型	If set to <code>true</code> return only graphs inherited from a template.如果设置“true”只返回从模板继承的图形。
展开名称 expandName	flag 标记	Expand macros in the graph name.展开图形名称中的宏。
选择组 selectGroups	query 询问	Return the host groups that the graph belongs to in the <code>groups</code> property.返回图形的“groups”参数的主机组。
选择模板 selectTemplates	query 询问	Return the templates that the graph belongs to in the <code>templates</code> property.返回图形的“templates”参数的模板。
选择主机 selectHosts	query 询问	Return the hosts that the graph belongs to in the <code>hosts</code> property.返回图形的“hosts”参数的主机。

选择监控项 selectItems	query 询问	Return the items used in the graph in the <code>items</code> property. 返回图形的“items”参数的监控项。
选择图形发现 selectGraphDiscovery	query 询问	Return the graph discovery object in the <code>graphDiscovery</code> property. The graph discovery objects links the graph to a graph prototype from which it was created. 返回在 <code>graphDiscovery</code> 参数中图形发现对象。图型发现对象链接图形与创建父图形的参数。 It has the following properties: 它接受以下参数 <code>graphid</code> - (string) ID of the graph; 图形ID <code>graphid</code> - (string字符串型) 图形ID; <code>parent_graphid</code> - (string) ID of the graph prototype from which the graph has been created. 父图形ID <code>parent_graphid</code> - (string字符串型) 父图形的ID
选择图形项 selectGraphItems	query 询问	Return the graph items used in the graph in the <code>gitems</code> property. 返回图形的“gitems”参数的监控项。
选择发现规则 selectDiscoveryRule	query 询问	Return the low-level discovery rule that created the graph in the <code>discoveryRule</code> property. 返回创建图形的“discoverRule”参数中的低级发现规则。
过滤器 filter	object 关联数组	Return only those results that exactly match the given filter. 只返回与给定过滤器完全匹配的结果。Accepts an array, where the keys are property names, and the values are either a single value or an array of values to match against. 接受一个数组，其中的键是属性名，值可以是一个值或者是一个匹配的值数组。Supports additional filters: 支持额外过滤器： <code>host</code> - technical name of the host that the graph belongs to; 图形所属主机的名称 <code>hostid</code> - ID of the host that the graph belongs to. 图形所属主机的ID
排序域 sortfield	string 字符串型/array数组	Sort the result by the given properties. 按指定的参数将结果排序 Possible values are: 可用值 图形ID <code>graphid</code> , 名称 <code>name</code> and 图形类型 <code>graphtype</code> .
countOutput	flag	These parameters being common for all <code>get</code> methods are described in detail in the reference commentary page. 这些参数对于所有 <code>get</code> 方法都是通用的，在 附录1：参考 中，查看详细描述。
editable	boolean	
excludeSearch	flag	
limit	integer	
output	query	
preservekeys	flag	
search	object	
searchByAny	boolean	
searchWildcardsEnabled	boolean	
sortorder	string/array	
startSearch	flag	

Return values 返回值

(integer整数/array数组) Returns either返回两个中的一个:

- an array of objects;一个数组对象
- the count of retrieved objects, if the `countOutput` parameter has been used.如果 `countOutput` 参数被使用, 将对检索到的对象进行计数。

Examples示例

Retrieving graphs from hosts获取主机的所有图像

Retrieve all graphs from host "10107" and sort them by name. 获取所有主机"10107"所有图形并对他们按名称排列。

Request请求:

```

1. {
2.     "jsonrpc": "2.0",
3.     "method": "graph.get",
4.     "params": {
5.         "output": "extend",
6.         "hostids": 10107,
7.         "sortfield": "name"
8.     },
9.     "auth": "038e1d7b1735c6a5436ee9eae095879e",
10.    "id": 1
11. }
```

Response响应:

```

1. {
2.     "jsonrpc": "2.0",
3.     "result": [
4.         {
5.             "graphid": "612",
6.             "name": "CPU jumps",
7.             "width": "900",
8.             "height": "200",
9.             "yaxismin": "0.0000",
10.            "yaxismax": "100.0000",
11.            "templateid": "439",
12.            "show_work_period": "1",
13.            "show_triggers": "1",
14.            "graphtype": "0",
15.            "show_legend": "1",
16.            "show_3d": "0",
17.            "percent_left": "0.0000",
18.            "percent_right": "0.0000",
```

```
19.         "ymin_type": "0",
20.         "ymax_type": "0",
21.         "ymin_itemid": "0",
22.         "ymax_itemid": "0",
23.         "flags": "0"
24.     },
25.     {
26.         "graphid": "613",
27.         "name": "CPU load",
28.         "width": "900",
29.         "height": "200",
30.         "yaxismin": "0.0000",
31.         "yaxismax": "100.0000",
32.         "templateid": "433",
33.         "show_work_period": "1",
34.         "show_triggers": "1",
35.         "graphtype": "0",
36.         "show_legend": "1",
37.         "show_3d": "0",
38.         "percent_left": "0.0000",
39.         "percent_right": "0.0000",
40.         "ymin_type": "1",
41.         "ymax_type": "0",
42.         "ymin_itemid": "0",
43.         "ymax_itemid": "0",
44.         "flags": "0"
45.     },
46.     {
47.         "graphid": "614",
48.         "name": "CPU utilization",
49.         "width": "900",
50.         "height": "200",
51.         "yaxismin": "0.0000",
52.         "yaxismax": "100.0000",
53.         "templateid": "387",
54.         "show_work_period": "1",
55.         "show_triggers": "0",
56.         "graphtype": "1",
57.         "show_legend": "1",
58.         "show_3d": "0",
59.         "percent_left": "0.0000",
60.         "percent_right": "0.0000",
61.         "ymin_type": "1",
62.         "ymax_type": "1",
63.         "ymin_itemid": "0",
64.         "ymax_itemid": "0",
65.         "flags": "0"
66.     },
67.     {
68.         "graphid": "645",
69.         "name": "Disk space usage /",
70.         "width": "600",
71.         "height": "340",
```

```
graph.get
```

```
72.         "yaxismin": "0.0000",
73.         "yaxismax": "0.0000",
74.         "templateid": "0",
75.         "show_work_period": "0",
76.         "show_triggers": "0",
77.         "graphtype": "2",
78.         "show_legend": "1",
79.         "show_3d": "1",
80.         "percent_left": "0.0000",
81.         "percent_right": "0.0000",
82.         "ymin_type": "0",
83.         "ymax_type": "0",
84.         "ymin_itemid": "0",
85.         "ymax_itemid": "0",
86.         "flags": "4"
87.     }
88. ],
89. "id": 1
90. }
```

See also参见

- [Discovery rule](#)发现规则
- [Graph item](#)图形项
- [Item](#)监控项
- [Host](#)主机
- [Host group](#)主机组
- [Template](#)模板

Source来源

`CGraph::get()` in *frontends/php/include/classes/api/services/CGraph.php*.

graph.update

Description描述

`(object) graph.update(object/array graphs)`

This method allows to update existing graphs. 此方法允许更新已经存在的图形。

Parameters参数

`(object/array)` Graph properties to be updated. `(关联数组/数组)` 需要更新的图形参数。

The `graphid` property must be defined for each graph, all other properties are optional. Only the passed properties will be updated, all others will remain unchanged. 图形 `graphid` 参数必须为每一张图形都被定义，所有其他的参数都是可选的。只有传送的参数将被更新，其他的参数将保持不变。

Additionally to the `standard graph properties` the method accepts the following parameters. 标准图形参数外，此方法接受以下参数。

Parameter 参数	Type 类型	Description描述
图形项 <code>gitems</code>	array 数组	Graph items to replace existing graph items. If a graph item has the <code>gitemid</code> property defined it will be updated, otherwise a new graph item will be created. 图形项将替换现有的图形项。如果已经有一个图形项的“ <code>gitemid</code> ”参数被定义，那么它将被更新，否则将创建一个新的图形项。

Return values返回值

`(object)` Returns an object containing the IDs of the updated graphs under the `graphids` property. `(关联数组object)` 返回一个关联数组，包含更新图片的ID参数。

Examples示例

Setting the maximum for the Y scale设置Y轴最大值刻度

Set the the maximum of the Y scale to a fixed value of 100. 设置Y轴最大值刻度固定为100.

Request请求：

```

1. {
2.   "jsonrpc": "2.0",
3.   "method": "graph.update",
4.   "params": {
5.     "graphid": "439",
6.     "ymax_type": 1,

```

```
graph.update
```

```
7.     "yaxismax": 100
8. },
9.     "auth": "038e1d7b1735c6a5436ee9eae095879e",
10.    "id": 1
11. }
```

Response响应：

```
1. {
2.     "jsonrpc": "2.0",
3.     "result": {
4.         "graphids": [
5.             "439"
6.         ]
7.     },
8.     "id": 1
9. }
```

Source来源

CGraph::update() in *frontends/php/include/classes/api/services/CGraph.php*.

Host interface 主机接口

This class is designed to work with host interfaces. 此类旨在与主机接口配合使用。

Object references: 对象引用:

- [Host interface](#)

Available methods: 可用的方法:

- `hostinterface.create` - creating new host interfaces
- `hostinterface.delete` - deleting host interfaces
- `hostinterface.get` - retrieving host interfaces
- `hostinterface.massadd` - adding host interfaces to hosts
- `hostinterface.massremove` - removing host interfaces from hosts
- `hostinterface.replacehostinterfaces` - replacing host interfaces on a host
- `hostinterface.update` - updating host interfaces

> Host interface object主机接口对象

The following objects are directly related to the [hostinterface API](#). 以下对象与“hostinterface”API直接相关。

Host interface主机接口

The host interface object has the following properties. 56/5000主机接口对象具有以下属性。

Note that both IP and DNS are required. If you do not want to use DNS, set it to an empty string. 请注意，IP和DNS都是必需的。 如果您不想使用DNS，请将其设置为空字符串。

参数	类型	描述
interfaceid	string	(readonly) ID of the interface. 接口的ID。
dns(required)	string	DNS name used by the interface. 接口使用的DNS名称。 Can be empty if the connection is made via IP. 如果通过IP进行连接，则可以为空。
hostid(required)	string	ID of the host the interface belongs to. 接口所属主机的ID。
ip(required)	string	IP address used by the interface. 接口使用的IP地址。 Can be empty if the connection is made via DNS. 如果通过DNS进行连接，可以为空。
main(required)	integer	Whether the interface is used as default on the host. Only one interface of some type can be set as default on a host. 该接口是否在主机上用作默认接口。 主机上只能有一种类型的接口作为默认设置。 Possible values are: 可能的值是: 0 - not default; 不是默认值 1 - default. 默认
port(required)	string	Port number used by the interface. Can contain user macros. 接口使用的端口号。 可以包含用户宏。
type(required)	integer	Interface type. 接口类型 Possible values are: 可能的值是: 1 - agent; 2 - SNMP; 3 - IPMI; 4 - JMX.
useip(required)	integer	Whether the connection should be made via IP. 是否应通过IP进行连接。 Possible values are: 可能的值是: 0 - connect using host DNS name; 连接使用主机DNS名称; 1 - connect using host IP address for this host interface. 使用该主机接口的主机IP地址进行连接。
bulk	integer	Whether to use bulk SNMP requests. 是否使用批量SNMP请求。 Possible values are: 可能的值是: 0 - don't use bulk requests; 不要使用批量请求; 1 - (default) 默认 use bulk requests. 使用批量请求;

hostinterface.create

Description 说明

```
object hostinterface.create(object/array hostInterfaces)
```

This method allows to create new host interfaces. 该方法允许创建新的主机接口。

Parameters 参数

(object/array) Host interfaces to create. The method accepts host interfaces with the standard host interface properties. 主机接口创建。 该方法接受具有标准主机接口属性的主机接口

Return values 返回值

(object) Returns an object containing the IDs of the created host interfaces under the `interfaceids` property. The order of the returned IDs matches the order of the passed host interfaces. 返回一个包含“interfaceID”属性下创建的主机接口的ID的对象。返回的ID的顺序与传递的主机接口的顺序相匹配。

Examples 示例

Create a new interface 创建一个新的接口

Create a secondary IP agent interface on host "30052." 在主机“30052”上创建一个辅助IP代理接口。

Request:

```

1. {
2.   "jsonrpc": "2.0",
3.   "method": "hostinterface.create",
4.   "params": {
5.     "hostid": "30052",
6.     "dns": "",
7.     "ip": "127.0.0.1",
8.     "main": 0,
9.     "port": "10050",
10.    "type": 1,
11.    "useip": 1
12.  },
13.  "auth": "038e1d7b1735c6a5436ee9eae095879e",
14.  "id": 1
15. }
```

Response:

```
1.  {
2.      "jsonrpc": "2.0",
3.      "result": {
4.          "interfaceids": [
5.              "30062"
6.          ]
7.      },
8.      "id": 1
9. }
```

See also 参见

- [hostinterface.massadd](#)
- [host.massadd](#)

Source 来源

CHostInterface::create() in
frontends/php/include/classes/api/services/CHostInterface.php.

hostinterface.delete

Description 说明

```
object hostinterface.delete(array hostInterfaceIds)
```

This method allows to delete host interfaces.此方法允许删除主机接口。

Parameters 参数

(array) IDs of the host interfaces to delete.要删除的主机接口的ID。

Return values 返回值

(object) Returns an object containing the IDs of the deleted host interfaces under the `interfaceids` property.返回包含“interfaceids”属性下删除的主机接口的ID的对象。

Examples 示例

Delete a host interface 删除主机界面

Delete the host interface with ID 30062.删除ID为30062的主机接口。

Request:

```
1. {
2.   "jsonrpc": "2.0",
3.   "method": "hostinterface.delete",
4.   "params": [
5.     "30062"
6.   ],
7.   "auth": "3a57200802b24cda67c4e4010b50c065",
8.   "id": 1
9. }
```

Response:

```
1. {
2.   "jsonrpc": "2.0",
3.   "result": {
4.     "interfaceids": [
5.       "30062"
6.     ]
7.   },
8.   "id": 1
9. }
```

See also 参见

- [hostinterface.massremove](#)
- [host.massremove](#)

Source 来源

CHostInterface::delete() in
frontends/php/include/classes/api/services/CHostInterface.php.

hostinterface.get

Description 说明

```
integer/array hostinterface.get(object parameters)
```

The method allows to retrieve host interfaces according to the given parameters. 该方法允许根据给定的参数检索主机接口。

Parameters 参数

(object) Parameters defining the desired output. 定义所需输出的参数。

The method supports the following parameters. 该方法支持以下参数。

参数	类型	描述
hostids	string/array	Return only host interfaces used by the given hosts. 仅返回给定主机使用的主机接口。
interfaceids	string/array	Return only host interfaces with the given IDs. 只返回具有给定ID的主机接口。
itemidss	string/array	Return only host interfaces used by the given items. 仅返回给定项目使用的主机接口。
triggerids	string/array	Return only host interfaces used by items in the given triggers. 只返回给定触发器中项目使用的主机接口。
selectItems	query	Return the items that use the interface in the <code>items</code> property. 在“items”属性中返回使用界面的项目。 Supports <code>count</code> . 支持“计数”。
selectHosts	query	Return the host that uses the interface as an array in the <code>hosts</code> property. 在“hosts”属性中返回使用该接口作为数组的主机。
limitSelects	integer	Limits the number of records returned by subselects. 限制子选择返回的记录数。 Applies to the following subselects: 适用于以下子选项: <code>selectItems</code> .
sortfield	string/array	Sort the result by the given properties. 按照给定的属性对结果进行排序。 Possible values are: <code>interfaceid</code> , <code>dns</code> , <code>ip</code> . 可能的值为: <code>interfaceid</code> , <code>dns</code> , <code>ip</code> .
countOutput	flag	These parameters being common for all <code>get</code> methods are described in detail in the reference commentary page. 这些参数对于所有的“获取”方法是常见的，在参考评论页中有详细描述。
editable	boolean	
excludeSearch	flag	
filter	object	

limit	integer
nodeids	string/array
output	query
preservekeys	flag
search	object
searchByAny	boolean
searchWildcardsEnabled	boolean
sortorder	string/array
startSearch	flag

Return values 返回值

(integer/array) Returns either: 返回:

- an array of objects; 一组对象;
- the count of retrieved objects, if the `countOutput` parameter has been used. 如果已经使用“countOutput”参数，则检索到的对象的计数。

Examples 示例

Retrieve host interfaces 检索主机接口

Retrieve all data about the interfaces used by host “30057.” 检索主机“30057”使用的接口的所有数据。

Request:

```

1. {
2.   "jsonrpc": "2.0",
3.   "method": "hostinterface.get",
4.   "params": {
5.     "output": "extend",
6.     "hostids": "30057"
7.   },
8.   "auth": "038e1d7b1735c6a5436ee9eae095879e",
9.   "id": 1
10. }
```

Response:

```

1. {
2.   "jsonrpc": "2.0",
3.   "result": [
4.     {
5.       "interfaceid": "30050",
```

```

6.          "hostid": "30057",
7.          "main": "1",
8.          "type": "1",
9.          "useip": "1",
10.         "ip": "127.0.0.1",
11.         "dns": "",
12.         "port": "10050",
13.         "bulk": "1"
14.      },
15.      {
16.        "interfaceid": "30067",
17.        "hostid": "30057",
18.        "main": "0",
19.        "type": "1",
20.        "useip": "0",
21.        "ip": "",
22.        "dns": "localhost",
23.        "port": "10050",
24.        "bulk": "1"
25.      },
26.      {
27.        "interfaceid": "30068",
28.        "hostid": "30057",
29.        "main": "1",
30.        "type": "2",
31.        "useip": "1",
32.        "ip": "127.0.0.1",
33.        "dns": "",
34.        "port": "161",
35.        "bulk": "1"
36.      }
37.    ],
38.    "id": 1
39.  }

```

See also 参见

- [Host](#)
- [Item](#)

Source 来源

`CHostInterface::get() in
frontends/php/include/classes/api/services/CHostInterface.php.`

hostinterface.massadd

Description 说明

```
object hostinterface.massadd(object parameters)
```

This method allows to simultaneously add host interfaces to multiple hosts.该方法允许同时向多个主机添加主机接口。

Parameters 参数

(object) Parameters containing the host interfaces to be created on the given hosts.包含要在给定主机上创建的主机接口的参数。

The method accepts the following parameters.该方法接受以下参数。

参数	类型	描述
hosts (required)	object/array	Hosts to be updated. 要更新的主机。The hosts must have the <code>hostid</code> property defined. 主机必须定义“hostid”属性。
interfaces (required)	object/array	Host interfaces to create on the given hosts. 在给定的主机上创建主机接口。

Return values 返回值

(object) Returns an object containing the IDs of the created host interfaces under the `interfaceids` property.返回一个包含“interfaceID”属性下创建的主机接口的ID的对象。

Examples 示例

Creating interfaces 创建接口

Create an interface on two hosts.在两台主机上创建一个接口。

Request:

```

1. {
2.   "jsonrpc": "2.0",
3.   "method": "hostinterface.massadd",
4.   "params": {
5.     "hosts": [
6.       {
7.         "hostid": "30050"
8.       },
9.       {
10.        "hostid": "30052"
11.      }
12.    ]
13.  }
14. }
```

```

12.     ],
13.     "interfaces": {
14.         "dns": "",
15.         "ip": "127.0.0.1",
16.         "main": 0,
17.         "port": "10050",
18.         "type": 1,
19.         "useip": 1
20.     }
21. },
22. "auth": "038e1d7b1735c6a5436ee9eae095879e",
23. "id": 1
24. }
```

Response:

```

1. {
2.     "jsonrpc": "2.0",
3.     "result": {
4.         "interfaceids": [
5.             "30069",
6.             "30070"
7.         ]
8.     },
9.     "id": 1
10. }
```

See also 参见

- [hostinterface.create](#)
- [host.massadd](#)
- [Host](#)

Source 来源

`CHostInterface::massAdd()` in
frontends/php/include/classes/api/services/CHostInterface.php.

hostinterface.massremove

Description 说明

```
object hostinterface.massremove(object parameters)
```

This method allows to remove host interfaces from the given hosts. 该方法允许从给定的主机中删除主机接口。

Parameters 参数

(object) Parameters containing the IDs of the hosts to be updated and the interfaces to be removed. 含要更新的主机的ID和要删除的接口的参数。

参数	类型	描述
hostids(required)	string/array	IDs of the hosts to be updated. 要更新的主机的ID。
interfaces(required)	object/array	Host interfaces to remove from the given hosts. 要从给定主机中删除的主机接口。The host interface object must have the ip, dns and port properties defined 主机接口对象必须具有定义的ip, dns和port属性

Return values 返回值

(object) Returns an object containing the IDs of the deleted host interfaces under the `interfaceids` property. 返回包含“interfaceids”属性下删除的主机接口的ID的对象。

Examples 示例

Removing interfaces 删除接口

Remove the “127.0.0.1” SNMP interface from two hosts. 从两台主机中删除“127.0.0.1”SNMP界面。

Request:

```

1. {
2.   "jsonrpc": "2.0",
3.   "method": "hostinterface.massremove",
4.   "params": {
5.     "hostids": [
6.       "30050",
7.       "30052"
8.     ],
9.     "interfaces": {
10.       "dns": "",
11.       "ip": "127.0.0.1",

```

```
12.         "port": "161"
13.     }
14. },
15. "auth": "038e1d7b1735c6a5436ee9eae095879e",
16. "id": 1
17. }
```

Response:

```
1. {
2.     "jsonrpc": "2.0",
3.     "result": {
4.         "interfaceids": [
5.             "30069",
6.             "30070"
7.         ],
8.         "id": 1
9.     }
10. }
```

See also 参见

- [hostinterface.delete](#)
- [host.massremove](#)

Source 来源

CHostInterface::massRemove() in
frontends/php/include/classes/api/services/CHostInterface.php.

hostinterface.replacehostinterfaces

Description 说明

```
object hostinterface.replacehostinterfaces(object parameters)
```

This method allows to replace all host interfaces on a given host.此方法允许替换给定主机上的所有主机接口。

Parameters 参数

(object) Parameters containing the ID of the host to be updated and the new host interfaces.包含要更新的主机ID和新主机接口的参数。

参数	类型	描述
hostid (required)	string	ID of the host to be updated. 要更新的主机的ID。
interfaces (required)	object/array	Host interfaces to replace the current host interfaces with. 替换当前主机接口的主机接口。

Return values 返回值

(object) Returns an object containing the IDs of the created host interfaces under the **interfaceids** property.返回一个包含“interfaceID”属性下创建的主机接口的ID的对象。

Examples 示例

Replacing host interfaces 更换主机接口

Replace all host interfaces with a single agent interface.用单个代理接口替换所有主机接口。

Request:

```

1. {
2.   "jsonrpc": "2.0",
3.   "method": "hostinterface.replacehostinterfaces",
4.   "params": {
5.     "hostid": "30052",
6.     "interfaces": {
7.       "dns": "",
8.       "ip": "127.0.0.1",
9.       "main": 1,
10.      "port": "10050",
11.      "type": 1,
12.      "useip": 1
13.    }
14.  },

```

```
15.     "auth": "038e1d7b1735c6a5436ee9eae095879e",
16.     "id": 1
17. }
```

Response:

```
1. {
2.     "jsonrpc": "2.0",
3.     "result": {
4.         "interfaceids": [
5.             "30081"
6.         ]
7.     },
8.     "id": 1
9. }
```

See also 参见

- [host.update](#)
- [host.massupdate](#)

Source 来源

`CHostInterface::replaceHostInterfaces()` in
frontends/php/include/classes/api/services/CHostInterface.php.

hostinterface.update

Description 说明

```
object hostinterface.update(object/array hostInterfaces)
```

This method allows to update existing host interfaces.此方法允许更新现有的主机接口。

Parameters 参数

(object/array) Host interface properties to be updated.更新的主机接口属性。

The `interfaceid` property must be defined for each host interface, all other properties are optional. Only the given properties will be updated, all others will remain unchanged.必须为每个主机接口定义“interfaceid”属性，所有其他属性都是可选的。只有给定的属性将被更新，所有其他属性将保持不变。

Return values 返回值

(object) Returns an object containing the IDs of the updated host interfaces under the `interfaceids` property.返回一个包含“interfaceids”属性下更新的主机接口的ID的对象

Examples 示例

Changing a host interface port更改主机接口端口

Change the port of a host interface.更改主机接口的端口。

Request:

```

1. {
2.   "jsonrpc": "2.0",
3.   "method": "hostinterface.update",
4.   "params": {
5.     "interfaceid": "30048",
6.     "port": "30050"
7.   },
8.   "auth": "038e1d7b1735c6a5436ee9eae095879e",
9.   "id": 1
10. }
```

Response:

```

1. {
2.   "jsonrpc": "2.0",
3.   "result": {
```

```
4.         "interfaceids": [
5.             "30048"
6.         ],
7.     },
8.     "id": 1
9. }
```

Source 来源

CHostInterface::update() in
frontends/php/include/classes/api/services/CHostInterface.php.

Host prototype 主机原型

This class is designed to work with host prototypes. 该类用于与主机原型一起工作。

Object references: 对象引用:

- [Host prototype](#)
- [Host prototype inventory](#)
- [Group link](#)
- [Group prototype](#)

Available methods: 可用的方法:

- `hostprototype.create` - creating new host prototypes
- `hostprototype.delete` - deleting host prototypes
- `hostprototype.get` - retrieving host prototypes
- `hostprototype.update` - updating host prototypes

> Host prototype object

The following objects are directly related to the [hostprototype API](#). 以下对象与“hostprototype”API直接相关。

Host prototype 主机原型

The host prototype object has the following properties. 主机原型对象具有以下属性。

参数	类型	描述
hostid	string	(readonly) ID of the host prototype. 主机原型的ID。
host(required)	string	Technical name of the host prototype. 主机原型的技术名称。
name	string	Visible name of the host prototype. 主机原型的可见名称。Default: <code>host</code> property value. 默认值: <code>host</code> 属性值。
status	integer	Status of the host prototype. 主机原型的状态Possible values are: 可能的值是: 0 - (default默认) monitored host; 被监控的主机; 1 - unmonitored host. 不受监控的主机。
templateid	string	(readonly) ID of the parent template host prototype. 父模板主机原型的ID
tlsconnect	integer	Connections to host. 连接到主机。Possible values are: 可能的值是: 1 - (default默认) No encryption; 无加密 2 - PSK; 4 - certificate. 证书
tls_accept	integer	Connections from host. 主机连接。Possible bitmap values are: 可能的位图值为: 1 - (default默认) No encryption; 无加密 2 - PSK; 4 - certificate. 证书
tls_issuer	string	Certificate issuer. 证书发行者
tls_subject	string	证书类别
tls_psk_identity	string	PSK identity. Required if either <code>tls_connect</code> or <code>tls_accept</code> has PSK enabled. PSK身份 如果“ <code>tls_connect</code> ”或“ <code>tls_accept</code> ”启用了PSK，则必需。
tls_psk	string	The preshared key, at least 32 hex digits. Required if either <code>tls_connect</code> or <code>tls_accept</code> has PSK enabled. 预共享密钥，至少32位十六进制数字。如果“ <code>tls_connect</code> ”或“ <code>tls_accept</code> ”启用了PSK，则必需。

Host prototype inventory 最初的原型库存

The host prototype inventory object has the following properties. 主机原型库存对象具有以下属性

参数	类型	描述
inventorymode	integer	Host prototype inventory population mode. 主机原型库存人口模式。Possible values are: 可能的值是: -1 - disabled; 禁用 0 - (default) manual; 手动 1 - automatic 自动.

Group link组链接

The group link object links a host prototype with a host group and has the following properties.组链接对象将主机原型与主机组链接，并具有以下属性。

参数	类型	描述
groupprototypeid	string	(readonly) <i>ID of the group link.</i> 组链接的ID。
groupid (required)	string	<i>ID of the host group.</i> 主机组的ID。
hostid	string	(readonly) <i>ID of the host prototype</i> 主机原型的ID
templateid	string	(readonly)_ <i>ID of the parent template group link.</i> 父模板组链接的ID。

Group prototype组原型

The group prototype object defines a group that will be created for a discovered host and has the following properties.组原型对象定义将为已发现的主机创建的组，并具有以下属性。

参数	类型	描述
groupprototypeid	string	(readonly) <i>ID of the group prototype.</i> 组原型的ID。
name (required)	string	<i>Name of the group prototype.</i> 组原型的名称
hostid	string	(readonly) <i>ID of the host prototype</i> 主机原型的ID
templateid	string	(readonly)_ <i>ID of the parent template group prototype.</i> 父模板组原型的ID。

hostprototype.create

Description 说明

```
object hostprototype.create(object/array hostPrototypes)
```

This method allows to create new host prototypes.这种方法允许创建新的主机原型。

Parameters 参数

(object/array) Host prototypes to create.主机原型创建。

Additionally to the [standard host prototype properties](#), the method accepts the following parameters.除标准主机原型属性之外，该方法接受以下参数。

参数	类型	描述
groupLinks(required)	array	Group links to be created for the host prototype. 要为主机原型创建的组链接。
ruleid(required)	string	ID of the LLD rule that the host prototype belongs to. 主机原型所属的LLD规则的ID。
groupPrototypes	array	Group prototypes to be created for the host prototype. 将为主机原型创建的组原型。
inventory	object	Host prototype inventory properties.主机原型库存属性。
templates	object/array	Templates to be linked to the host prototype. 与主机原型链接的模板。The templates must have the <code>templateid</code> property defined.模板必须定义“templateid”属性。

Return values 返回值

(object) Returns an object containing the IDs of the created host prototypes under the `hostids` property. The order of the returned IDs matches the order of the passed host prototypes.返回包含“hostid”属性下创建的主机原型的ID的对象。 返回的ID的顺序与传递的主机原型的顺序相匹配。

Examples 示例

Creating a host prototype 创建主机原型

Create a host prototype “{#VM.NAME}” on LLD rule “23542” with a group prototype “{#HV.NAME}”. Link it to host group “2”.使用组原型“{#HV.NAME}”为LLD规则“23542”创建主机原型“{#VM.NAME}”。 将其链接到主机组“2”。

Request:

```

1. {
2.     "jsonrpc": "2.0",
3.     "method": "hostprototype.create",
4.     "params": {
5.         "host": "{#VM.NAME}",
6.         "ruleid": "23542",
7.         "groupLinks": [
8.             {
9.                 "groupid": "2"
10.            }
11.        ],
12.        "groupPrototypes": [
13.            {
14.                "name": "{#HV.NAME}"
15.            }
16.        ]
17.    },
18.    "auth": "038e1d7b1735c6a5436ee9eae095879e",
19.    "id": 1
20. }

```

Response:

```

1. {
2.     "jsonrpc": "2.0",
3.     "result": {
4.         "hostids": [
5.             "10103"
6.         ]
7.     },
8.     "id": 1
9. }

```

See also 参见

- [Group link](#)
- [Group prototype](#)
- [Host prototype inventory](#)

Source 来源

`CHostPrototype::create()` in
`frontends/php/include/classes/api/services/CHostPrototype.php`.

hostprototype.delete

Description 说明

```
object hostprototype.delete(array hostPrototypeIds)
```

This method allows to delete host prototypes. 该方法允许删除主机原型。

Parameters 参数

(array) IDs of the host prototypes to delete. 要删除的主机原型的ID。

Return values 返回值

(object) Returns an object containing the IDs of the deleted host prototypes under the hostids property. 返回一个包含“hostid”属性下删除的主机原型的ID的对象。

Examples 示例

Deleting multiple host prototypes 删除多个主机原型

Delete two host prototypes. 删除两个主机原型

Request:

```
1. {
2.   "jsonrpc": "2.0",
3.   "method": "hostprototype.delete",
4.   "params": [
5.     "10103",
6.     "10105"
7.   ],
8.   "auth": "3a57200802b24cda67c4e4010b50c065",
9.   "id": 1
10. }
```

Response:

```
1. {
2.   "jsonrpc": "2.0",
3.   "result": {
4.     "hostids": [
5.       "10103",
6.       "10105"
7.     ]
8.   },
9.   "id": 1
}
```

hostprototype.delete

```
10. }
```

Source 来源

CHostPrototype::delete() in
frontends/php/include/classes/api/services/CHostPrototype.php.

hostprototype.get

Description 说明

```
integer/array hostprototype.get(object parameters)
```

The method allows to retrieve host prototypes according to the given parameters. 该方法允许根据给定的参数检索主机原型。

Parameters 参数

(object) Parameters defining the desired output. 定义所需输出的参数。

The method supports the following parameters. 该方法支持以下参数。

参数	类型	描述
hostids	string/array	Return only host prototypes with the given IDs. 只返回具有给定ID的主机原型
discoveryids	string/array	Return only host prototype that belong to the given LLD rules. 仅返回属于给定LLD规则的主机原型。
inherited	boolean	If set to <code>true</code> return only items inherited from a template. 如果设置为“true”，只返回从模板继承的项目。
selectDiscoveryRule	query	Return the LLD rule that the host prototype belongs to in the <code>discoveryRule</code> property. 在“discoveryRule”属性中返回主机原型所属的LLD规则。
selectGroupLinks	query	Return the group links of the host prototype in the <code>groupLinks</code> property. 在“groupLinks”属性中返回主机原型的组链接。
selectGroupPrototypes	query	Return the group prototypes of the host prototype in the <code>groupPrototypes</code> property. 返回“groupPrototypes”属性中的主机原型的组原型。
selectInventory	query	Return the host prototype inventory in the <code>inventory</code> property. 在“库存”属性中返回主机原型库存。
selectParentHost	query	Return the host that the host prototype belongs to in the <code>parentHost</code> property. 在“parentHost”属性中返回主机原型所属的主机。
selectTemplates	query	Return the templates linked to the host prototype in the <code>templates</code> property. 返回在“模板”属性中链接到主机原型的模板。
sortfield	string/array	Sort the result by the given properties. 按照给定的属性对结果进行排序。Possible values are: <code>hostid</code> , <code>host</code> , <code>name</code> and <code>status</code> . 可能的值是: 'hostid', 'host', 'name' 和 'status'。

countOutput	flag	These parameters being common for all <code>get</code> methods are described in detail on the Generic Zabbix API information page.通用Zabbix API信息页面详细描述了所有“get”方法的这些参数。
editable	boolean	
excludeSearch	flag	
filter	object	
limit	integer	
output	query	
preservekeys	flag	
search	object	
searchByAny	boolean	
searchWildcardsEnabled	boolean	
sortorder	string/array	
startSearch	flag	

Return values 返回值

(integer/array) Returns either:返回:

- an array of objects;一组对象;
- the count of retrieved objects, if the `countOutput` parameter has been used.如果已经使用“countOutput”参数，则检索到的对象的计数。

Examples 示例

Retrieving host prototypes from an LLD rule 从LLD规则检索主机原型

Retrieve all host prototypes and their group links and group prototypes from an LLD rule.从LLD规则中检索所有主机原型及其组链接和组原型。

Request:

```

1. {
2.   "jsonrpc": "2.0",
3.   "method": "hostprototype.get",
4.   "params": {
5.     "output": "extend",
6.     "selectGroupLinks": "extend",
7.     "selectGroupPrototypes": "extend",
8.     "discoveryids": "23554"
9.   },
10.  "auth": "038e1d7b1735c6a5436ee9eae095879e",

```

hostprototype.get

```
11.     "id": 1
12. }
```

Response:

```
1. {
2.     "jsonrpc": "2.0",
3.     "result": [
4.         {
5.             "hostid": "10092",
6.             "host": "{#HV.UUID}",
7.             "status": "0",
8.             "name": "{#HV.NAME}",
9.             "templateid": "0",
10.            "tls_connect": "1",
11.            "tls_accept": "1",
12.            "tls_issuer": "",
13.            "tls_subject": "",
14.            "tls_psk_identity": "",
15.            "tls_psk": "",
16.            "groupLinks": [
17.                {
18.                    "group_prototypeid": "4",
19.                    "hostid": "10092",
20.                    "groupid": "7",
21.                    "templateid": "0"
22.                }
23.            ],
24.            "groupPrototypes": [
25.                {
26.                    "group_prototypeid": "7",
27.                    "hostid": "10092",
28.                    "name": "{#CLUSTER.NAME}",
29.                    "templateid": "0"
30.                }
31.            ]
32.        },
33.        ],
34.        "id": 1
35. }
```

See also 参见

- [Group link](#)
- [Group prototype](#)
- [Host prototype inventory](#)

Source 来源

hostprototype.get

```
CHostPrototype::get() in  
frontends/php/include/classes/api/services/CHostPrototype.php.
```

hostprototype.update

Description 说明

```
object hostprototype.update(object/array hostPrototypes)
```

This method allows to update existing host prototypes.此方法允许更新现有的主机原型。

Parameters 参数

(object/array) Host prototype properties to be updated.要更新的主机原型属性。

The `hostid` property must be defined for each host prototype, all other properties are optional. Only the passed properties will be updated, all others will remain unchanged.必须为每个主机原型定义“hostid”属性，所有其他属性都是可选的。只有过期的属性将被更新，所有其他属性将保持不变。

Additionally to the [standard host prototype properties](#), the method accepts the following parameters.除标准主机原型属性外，该方法还接受以下参数。

参数	类型	描述
groupLinks	array	Group links to replace the current group links on the host prototype.替换主机原型上当前组链接的组链接
groupPrototypes	array	Group prototypes to replace the existing group prototypes on the host prototype. 代替主机原型上现有组原型的组原型。
inventory	object	Host prototype inventory properties.主机原型库存属性。
templates	object/array	Templates to replace the currently linked templates.用于替换当前链接的模板的模板。The templates must have the <code>templateid</code> property defined.模板必须定义“templateid”属性。

Return values 返回值

(object) Returns an object containing the IDs of the updated host prototypes under the `hostids` property.返回包含“hostid”属性下更新的主机原型的ID的对象。

Examples 示例

Disabling a host prototype禁用主机原型

Disable a host prototype, that is, set its status to 1.禁用主机原型，即将其状态设置为1。

Request:

```

1. {
2.     "jsonrpc": "2.0",
3.     "method": "hostprototype.update",
4.     "params": {
5.         "hostid": "10092",
6.         "status": 1
7.     },
8.     "auth": "038e1d7b1735c6a5436ee9eae095879e",
9.     "id": 1
10. }

```

Response:

```

1. {
2.     "jsonrpc": "2.0",
3.     "result": {
4.         "hostids": [
5.             "10092"
6.         ]
7.     },
8.     "id": 1
9. }

```

See also 参见

- [Group link](#)
- [Group prototype](#)
- [Host prototype inventory](#)

Source 来源

`CHostPrototype::update()` in
frontends/php/include/classes/api/services/CHostPrototype.php.

Icon map 图标拓扑图

This class is designed to work with icon maps. 这个类用于配合使用图标拓扑图。

Object references: 对象引用:

- [Icon map](#)
- [Icon mapping](#)

Available methods: 相关方法:

- `iconmap.create` - create new icon maps 创建新的图标拓扑图
- `iconmap.delete` - delete icon maps 删除图标拓扑图
- `iconmap.get` - retrieve icon maps 获取图标拓扑图
- `iconmap.update` - update icon maps 更新图标拓扑图

> Icon map object图标拓扑图对象

The following objects are directly related to the [iconmap API](#). 以下是iconmap API的使用方法。

Icon map 图标拓扑图

The icon map object has the following properties. 图标拓扑图对象有以下属性。

属性	类型	说明
iconmapid	string	(readonly) ID of the icon map. (只读) 图标拓扑图ID。
default_iconid (required)	string	ID of the default icon. 默认图标的ID。
name (required)	string	Name of the icon map. 图标拓扑图的名称。

Icon mapping 图标映射

The icon mapping object defines a specific icon to be used for hosts with a certain inventory field value. It has the following properties. 图标映射对象定义了一个具体的图标，给具有特定资产清单字段值的主机使用。图标映射有以下属性。

属性	类型	说明
iconmappingid	string	(readonly) ID of the icon map. (只读) 图标拓扑图ID。
iconid (required)	string	ID of the icon used by the icon mapping. 被图标映射使用到的图标ID。
expression (required)	string	Expression to match the inventory field against. 使资产清单字段匹配的表达式。
inventory_link (required)	integer	ID of the host inventory field. 主机资产清单字段ID。 Refer to the host inventory object for a list of supported inventory fields. 参考host inventory object支持的资产清单字段列表。
iconmapid	string	(readonly) ID of the icon map that the icon mapping belongs to. (只读) 图标映射所属的图标拓扑图ID。
sortorder	integer	(readonly) Position of the icon mapping in the icon map. (只读) 在图标拓扑图中图标映射的位置。

iconmap.create

Description 说明

```
object iconmap.create(object/array iconMaps)
```

This method allows to create new icon maps.此方法允许创建新的图标拓扑图。

Parameters参数

(object/array) Icon maps to create.需要创建的图标拓扑图。

Additionally to the [standard icon map properties](#), the method accepts the following parameters.另外，对于标准图标拓扑图属性，此方法接受以下参数。

参数	类型	说明
mappings(required)	array	Icon mappings to be created for the icon map.为图标拓扑图所创建的图标映射。

Return values 返回值

(object) Returns an object containing the IDs of the created icon maps under the `iconmapids` property. The order of the returned IDs matches the order of the passed icon maps.返回一个对象其中包含在iconmapids属性下已创建图标拓扑图的ID。返回ID的命令与传递图标拓扑图的命令匹配。

Examples范例

Create an icon map 创建一个图标拓扑图

Create an icon map to display hosts of different types.创建一个图标拓扑图来显示不同类型的主机。

Request:

```

1. {
2.   "jsonrpc": "2.0",
3.   "method": "iconmap.create",
4.   "params": {
5.     "name": "Type icons",
6.     "default_iconid": "2",
7.     "mappings": [
8.       {
9.         "inventory_link": 1,
10.        "expression": "server",
11.        "iconid": "3"
12.      },

```

```

13.      {
14.          "inventory_link": 1,
15.          "expression": "switch",
16.          "iconid": "4"
17.      }
18.  ],
19. },
20. "auth": "038e1d7b1735c6a5436ee9eae095879e",
21. "id": 1
22. }
```

Response:

```

1. {
2.     "jsonrpc": "2.0",
3.     "result": {
4.         "iconmapids": [
5.             "2"
6.         ]
7.     },
8.     "id": 1
9. }
```

See also 参见

- [Icon mapping](#)

Source来源

`CIconMap::create()` in *frontends/php/include/classes/api/services/CIconMap.php*.

iconmap.delete

Description 说明

```
object iconmap.delete(array iconMapIds)
```

This method allows to delete icon maps.此方法允许删除图标拓扑图。

Parameters 参数

(array) IDs of the icon maps to delete.需要删除的图标拓扑图ID。

Return values 返回值

(object) Returns an object containing the IDs of the deleted icon maps under the iconmapids property.返回一个对象其中包含在iconmapids属性下的已删除图标拓扑图ID。

Examples 范例

Delete multiple icon maps 删除多个图标拓扑图

Delete two icon maps.删除两个图标拓扑图

Request:

```
1. {
2.   "jsonrpc": "2.0",
3.   "method": "iconmap.delete",
4.   "params": [
5.     "2",
6.     "5"
7.   ],
8.   "auth": "3a57200802b24cda67c4e4010b50c065",
9.   "id": 1
10. }
```

Response:

```
1. {
2.   "jsonrpc": "2.0",
3.   "result": {
4.     "iconmapids": [
5.       "2",
6.       "5"
7.     ]
8.   },
9.   "id": 1
}
```

iconmap.delete

```
10. }
```

Source来源

CIconMap::delete() in *frontends/php/include/classes/api/services/CIconMap.php*.

iconmap.get

Description 说明

```
integer/array iconmap.get(object parameters)
```

The method allows to retrieve icon maps according to the given parameters. 此方法允许根据指定的参数获取图标拓扑图。

Parameters 参数

(object) Parameters defining the desired output. 定义期望输出值的参数。

The method supports the following parameters. 该方法支持以下参数。

参数	类型	描述
iconmapids	string/array	Return only icon maps with the given IDs. 仅返回指定ID的图标拓扑图。
sysmapids	string/array	Return only icon maps that are used in the given maps. 仅返回在指定拓扑图中使用的图标拓扑图。
selectMappings	query	Return used icon mappings in the <code>mappings</code> property. 返回在mappings属性中使用的图标映射。
sortfield	string/array	Sort the result by the given properties. 以指定的属性将结果排序。Possible values are: <code>iconmapid</code> and <code>name</code> . 可能的值为: iconmapid和name。
countOutput	flag	These parameters being common for all <code>get</code> methods are described in detail in the reference commentary . 在reference commentary中详细描述了所有“get”方法的共同参数.
editable	boolean	
excludeSearch	flag	
filter	object	
limit	integer	
output	query	
preservekeys	flag	
search	object	
searchByAny	boolean	
searchWildcardsEnabled	boolean	
sortorder	string/array	
startSearch	flag	

Return values 返回值

(integer/array) Returns either:返回其中任一:

- an array of objects;一组对象;
- the count of retrieved objects, if the `countOutput` parameter has been used.如果已经使用了“countOutput”参数，则检索对象的计数.

Examples 范例

Retrieve an icon map 获取一个图标拓扑图

Retrieve all data about icon map “3”.获取所有关于ID为3的图标拓扑图数据。

Request:

```

1. {
2.   "jsonrpc": "2.0",
3.   "method": "iconmap.get",
4.   "params": {
5.     "iconmapids": "3",
6.     "output": "extend",
7.     "selectMappings": "extend"
8.   },
9.   "auth": "038e1d7b1735c6a5436ee9eae095879e",
10.  "id": 1
11. }
```

Response:

```

1. {
2.   "jsonrpc": "2.0",
3.   "result": [
4.     {
5.       "mappings": [
6.         {
7.           "iconmappingid": "3",
8.           "iconapid": "3",
9.           "iconid": "6",
10.          "inventory_link": "1",
11.          "expression": "server",
12.          "sortorder": "0"
13.        },
14.        {
15.           "iconmappingid": "4",
16.           "iconapid": "3",
17.           "iconid": "10",
18.           "inventory_link": "1",
19.           "expression": "switch",
```

```
20.          "sortorder": "1"
21.      }
22.  ],
23.  "iconmapid": "3",
24.  "name": "Host type icons",
25.  "default_iconid": "2"
26. }
27. ],
28. "id": 1
29. }
```

See also 参见

- [Icon mapping](#)

Source 来源

`CIconMap::get()` in *frontends/php/include/classes/api/services/CIconMap.php*.

iconmap.update

Description 说明

```
object iconmap.update(object/array iconMaps)
```

This method allows to update existing icon maps.此方法允许更新已有的图标拓扑图。

Parameters 参数

(object/array) Icon map properties to be updated.需要更新的图标拓扑图属性。

The `iconmapid` property must be defined for each icon map, all other properties are optional. Only the passed properties will be updated, all others will remain unchanged.每一个图标拓扑图的iconmapid属性必须已定义过，其他属性为可选项。仅被传递的属性会被更新，其他属性保持不变。

Additionally to the `standard icon map properties`, the method accepts the following parameters.另外，对于标准图标拓扑图属性，此方法接受以下参数。

参数	类型	描述
<code>mappings</code>	array	Icon mappings to replace the existing icon mappings.替换当前图标映射。

Return values 返回值

(object) Returns an object containing the IDs of the updated icon maps under the `iconmapids` property.返回一个对象其中包含在iconmapids属性下已更新图标拓扑图的ID。

Examples 范例

Rename icon map重命名图标拓扑图

Rename an icon map to “OS icons”.将图标拓扑图重命名为“OS icons”。

Request:

```

1. {
2.   "jsonrpc": "2.0",
3.   "method": "iconmap.update",
4.   "params": {
5.     "iconmapid": "1",
6.     "name": "OS icons"
7.   },
8.   "auth": "038e1d7b1735c6a5436ee9eae095879e",
9.   "id": 1
10. }
```

Response:

```
1. {
2.     "jsonrpc": "2.0",
3.     "result": {
4.         "iconmapids": [
5.             "1"
6.         ],
7.     },
8.     "id": 1
9. }
```

See also 参见

- [Icon mapping](#)

Source 来源

`CIconMap::update()` in *frontends/php/include/classes/api/services/CIconMap.php*.

Item prototype Item原型

This class is designed to work with item prototypes. 此类旨辅助Item原型的使用

Object references: 对象引用:

- [Item prototype](#)

可用的方法:

- [itemprototype.create](#) - creating new item prototypes 创建新的Item原型
- [itemprototype.delete](#) - deleting item prototypes 删除Item原型
- [itemprototype.get](#) - retrieving item prototypes 获取Item原型
- [itemprototype.update](#) - updating item prototypes 更新Item原型

> Item prototype object

The following objects are directly related to the [itemprototype API](#). 以下对象与“itemprototype”API直接相关。

Item prototype

The item prototype object has the following properties. Item prototype对象具有以下属性。

属性	类型	说明
<code>itemid</code>	<code>string</code>	(readonly) item prototype的ID.
<code>delay(required)</code>	<code>string</code>	Update interval of the item prototype. Accepts seconds or time unit with suffix and with or without one or more custom intervals that consist of either flexible intervals and scheduling intervals as serialized strings. Also accepts user macros and LLD macros. Flexible intervals could be written as two macros separated by a forward slash. Intervals are separated by a semicolon. 更新item prototype的间隔。 接受具有后缀的秒或时间单位，并且具有或不具有由灵活间隔和调度间隔组成的一个或多个自定义间隔作为串行化字符串。还接受用户宏和LLD宏。 灵活的间隔可以写成两个由正斜杠分隔的宏。 间隔用分号分隔。
<code>hostid(required)</code>	<code>string</code>	ID of the host that the item prototype belongs to.item prototype所属的主机的ID
<code>interfaceid(required)</code>	<code>string</code>	ID of the item prototype's host interface. Used only for host item prototypes.item prototypes的主机接口的ID。 仅用于主机item prototypes Optional for Zabbix agent (active), Zabbix internal, Zabbix trapper, Zabbix aggregate, database monitor and calculated item prototypes. 可选的Zabbix代理（活动），Zabbix内部，Zabbix陷阱，Zabbix聚合，数据库监视和计算item prototypes。
<code>key_(required)</code>	<code>string</code>	Item prototype key.
<code>name(required)</code>	<code>string</code>	Name of the item prototype.item prototypes的名称
<code>type(required)</code>	<code>integer</code>	Type of the item prototype. item prototypes的类型Possible values: 可能的值0 - Zabbix agent; 1 - SNMPv1 agent; 2 - Zabbix trapper; 3 - simple check; 4 - SNMPv2 agent; 5 - Zabbix internal; 6 - SNMPv3 agent; 7 - Zabbix agent (active); 8 - Zabbix aggregate; 10 - external check; 11 - database monitor; 12 - IPMI agent; 13 - SSH agent; 14 - TELNET agent; 15 - calculated; 16 - JMX agent; 17 - SNMP trap.
<code>value_type(required)</code>	<code>integer</code>	Type of information of the item prototype. item prototypes信息类别Possible values: 可能的值: 0 - numeric float; 1 - character; 2 - log; 3 - numeric unsigned; 4 - text.

authtype	integer	SSH authentication method. Used only by SSH agent item prototypes. Possible values: 0 - (default) password; 1 - public key.
datatype	integer	Data type of the item prototype. item prototypes的数据类别Possible values: 可能的值 0 - (default) decimal; 1 - octal; 2 - hexadecimal; 3 - boolean.
delta	integer	Value that will be stored. 将被存储的值。 Possible values: 可能的值 0 - (default) as is; 1 - Delta, speed per second; 2 - Delta, simple change.
description	string	Description of the item prototype.item prototype说明
formula	integer/float	Custom multiplier. Default: 1.
history	string	A time unit of how long the history data should be stored. Also accepts user macro and LLD macro. 历史数据应该存储多长时间的单位。还接受用户宏和LLD宏。 Default: 90d.
ipmi_sensor	string	IPMI sensor. Used only by IPMI item prototypes. IPMI传感器 仅由IPMI item prototypes使用。
logtimefmt	string	Format of the time in log entries. Used only by log item prototypes. 日志条目中的时间格式。 仅用于日志item prototypes
multiplier	integer	Whether to use a custom multiplier. 是否使用自定义乘数。
params	string	Additional parameters depending on the type of the item prototype: 附加参数取决于 item prototype: 的类型: - executed script for SSH and Telnet item prototypes; 执行SSH 和Telnet item prototypes的脚本; - SQL query for database monitor item prototypes; SQL 查询数据库监视item prototypes;- formula for calculated item prototypes. 计算item prototypes.的公式。
password	string	Password for authentication. Used by simple check, SSH, Telnet, database monitor and JMX item prototypes. 认证密码。 用于简单检查, SSH, Telnet, 数据库监控和JMX item prototypes。
port	string	Port monitored by the item prototype. Used only by SNMP items prototype.items prototype监控的端, 仅由SNMP items prototype 使用。
privatekey	string	Name of the private key file. 私钥文件的名称。
publickey	string	Name of the public key file. 公钥文件的名称
snmp_community	string	SNMP community. Used only by SNMPv1 and SNMPv2 item prototypes. 仅由SNMPv1和SNMPv2 item prototypes使用。
snmp_oid	string	SNMP OID.
snmpv3_authpassphrase	string	SNMPv3 auth passphrase. Used only by SNMPv3 item prototypes. NMPv3 auth

		passphrase。仅用于SNMPv3 item prototypes。
snmpv3_authprotocol	integer	SNMPv3 authentication protocol. Used only by SNMPv3 items. SNMPv3认证协议 仅用于SNMPv3 items. Possible values: 可能的值: 0 - (default) MD5; 1 - SHA.
snmpv3_contextname	string	SNMPv3 context name. Used only by SNMPv3 item prototypes. SNMPv3上下文名称。仅用于SNMPv3 item prototypes.。
snmpv3_privpassphrase	string	SNMPv3 priv passphrase. Used only by SNMPv3 item prototypes. SNMPv3 priv密码。仅用于SNMPv3 item prototypes.
snmpv3_privprotocol	integer	
snmpv3_securitylevel	integer	SNMPv3 security level. Used only by SNMPv3 item prototypes. Possible values: 0 - noAuthNoPriv; 1 - authNoPriv; 2 - authPriv.
snmpv3_securityname	string	SNMPv3 security name. Used only by SNMPv3 item prototypes.
status	integer	Status of the item prototype. Possible values: 0 - (default) enabled item prototype; 1 - disabled item prototype; 3 - unsupported item prototype.
templateid	string	(readonly) ID of the parent template item prototype.
trapper_hosts	string	Allowed hosts. Used only by trapper item prototypes.
trends	string	A time unit of how long the trends data should be stored. Also accepts user macro and LLD macro. Default: 365d.
units	string	Value units.
username	string	Username for authentication. Used by simple check, SSH, Telnet, database monitor and JMX item prototypes. Required by SSH and Telnet item prototypes.
valuemapid	string	ID of the associated value map.

itemprototype.create

Description 说明

```
object itemprototype.create(object/array itemPrototypes)
```

This method allows to create new item prototypes.这种方法允许创建新的item prototypes。

Parameters 参数

(object/array) Item prototype to create.要创建的item prototypes

Additionally to the standard item prototype properties, the method accepts the following parameters.除标准项原型属性外，该方法还接受以下参数。

属性	类型	说明
ruleid(required)	string	ID of the LLD rule that the item belongs to.该项所属的LLD规则的ID。
applications	array	IDs of applications to be assigned to the discovered items. 要分配给发现项目的应用程序的ID。
applicationPrototypes	array	Names of application prototypes to be assigned to the item prototype.要分配给项目原型的应用程序原型的名称。

Return values 返回值

(object) Returns an object containing the IDs of the created item prototypes under the itemids property. The order of the returned IDs matches the order of the passed item prototypes.返回一个包含“itemid”属性下创建的item prototypes的ID的对象。返回的ID的顺序与传递的item prototypes的顺序相匹配。

Examples 示例

Creating an item prototype 创建一个item prototypes

Create an item prototype to monitor free disc space on a discovered file system. Discovered items should be numeric Zabbix agent items updated every 30 seconds.

Request:

```

1. {
2.   "jsonrpc": "2.0",
3.   "method": "itemprototype.create",
4.   "params": {
5.     "name": "Free disk space on $1",
6.     "key_": "vfs.fs.size[#{FSNAME},free]",

```

```
itemprototype.create
```

```
7.     "hostid": "10197",
8.     "ruleid": "27665",
9.     "type": 0,
10.    "value_type": 3,
11.    "interfaceid": "112",
12.    "delay": "30s"
13. },
14. "auth": "038e1d7b1735c6a5436ee9eae095879e",
15. "id": 1
16. }
```

Response:

```
1. {
2.   "jsonrpc": "2.0",
3.   "result": {
4.     "itemids": [
5.       "27666"
6.     ]
7.   },
8.   "id": 1
9. }
```

Source 来源

CItemPrototype::create() in
frontends/php/include/classes/api/services/CItemPrototype.php.

itemprototype.delete

Description 说明

```
object itemprototype.delete(array itemPrototypeIds)
```

This method allows to delete item prototypes.此方法允许删除item prototypes。

Parameters 参数

(array) IDs of the item prototypes to delete.要删除的item prototypes的ID

Return values 返回值

(object) Returns an object containing the IDs of the deleted item prototypes under the `prototypeids` property.返回一个对象，该对象包含“prototypeids”属性下的已删除item prototypes的ID。

Examples 示例

Deleting multiple item prototypes

Delete two item prototypes.

Request:

```
1. {
2.   "jsonrpc": "2.0",
3.   "method": "itemprototype.delete",
4.   "params": [
5.     "27352",
6.     "27356"
7.   ],
8.   "auth": "3a57200802b24cda67c4e4010b50c065",
9.   "id": 1
10. }
```

Response:

```
1. {
2.   "jsonrpc": "2.0",
3.   "result": {
4.     "prototypeids": [
5.       "27352",
6.       "27356"
7.     ]
8.   },
9. }
```

itemprototype.delete

```
9.      "id": 1  
10. }
```

Source 来源

CItemPrototype::delete() in
frontends/php/include/classes/api/services/CItemPrototype.php.

itemprototype.get

Description 说明

```
integer/array itemprototype.get(object parameters)
```

The method allows to retrieve item prototypes according to the given parameters.该方法允许根据给定的参数检索item prototypes。

Parameters 参数

(object) Parameters defining the desired output.定义所需输出的参数。

The method supports the following parameters.该方法支持以下参数。

属性	类型	说明
discoveryids	string/array	Return only item prototypes that belong to the given LLD rules.
graphids	string/array	Return only item prototypes that are used in the given graph prototypes.
hostids	string/array	Return only item prototypes that belong to the given hosts.
inherited	boolean	If set to <code>true</code> return only item prototypes inherited from a template.
itemidss	string/array	Return only item prototypes with the given IDs.
monitored	boolean	If set to <code>true</code> return only enabled item prototypes that belong to monitored hosts.
templated	boolean	If set to <code>true</code> return only item prototypes that belong to templates.
templateids	string/array	Return only item prototypes that belong to the given templates.
triggerids	string/array	Return only item prototypes that are used in the given trigger prototypes.
selectApplications	query	Return applications that the item prototype belongs to in the <code>applications</code> property.
selectApplicationPrototypes	query	Return application prototypes linked to item prototype in <code>applicationPrototypes</code> property.
selectDiscoveryRule	query	Return the low-level discovery rule that the graph prototype belongs to in the <code>discoveryRule</code> property.
selectGraphs	query	Return graph prototypes that the item prototype is used in in the <code>graphs</code> property. Supports <code>count</code> .

selectHosts	query	Returns the host that the item prototype belongs to as an array in the <code>hosts</code> property.
selectTriggers	query	Return trigger prototypes that the item prototype is used in in the <code>triggers</code> property. Supports <code>count</code> .
filter	object	Return only those results that exactly match the given filter. Accepts an array, where the keys are property names, and the values are either a single value or an array of values to match against. Supports additional filters: <code>host</code> - technical name of the host that the item prototype belongs to.
limitSelects	integer	Limits the number of records returned by subselects. Applies to the following subselects: <code>selectGraphs</code> - results will be sorted by <code>name</code> ; <code>selectTriggers</code> - results will be sorted by <code>description</code> .
sortfield	string/array	Sort the result by the given properties. Possible values are: <code>itemid</code> , <code>name</code> , <code>key_</code> , <code>delay</code> , <code>type</code> and <code>status</code> .
countOutput	flag	These parameters being common for all <code>get</code> methods are described in detail in the reference commentary .
editable	boolean	
excludeSearch	flag	
limit	integer	
output	query	
preservekeys	flag	
search	object	
searchByAny	boolean	
searchWildcardsEnabled	boolean	
sortorder	string/array	
startSearch	flag	

Return values

(integer/array) Returns either:

- an array of objects;
- the count of retrieved objects, if the `countOutput` parameter has been used.

Examples示例

Retrieving item prototypes from an LLD rule

Retrieve all item prototypes from an LLD rule.

Request:

```

1. {
2.   "jsonrpc": "2.0",
3.   "method": "itemprototype.get",
4.   "params": {
5.     "output": "extend",
6.     "discoveryids": "27426"
7.   },
8.   "auth": "038e1d7b1735c6a5436ee9eae095879e",
9.   "id": 1
10. }
```

Response:

```

1. {
2.   "jsonrpc": "2.0",
3.   "result": [
4.     {
5.       "itemid": "27427",
6.       "type": "0",
7.       "snmp_community": "",
8.       "snmp_oid": "",
9.       "hostid": "10202",
10.      "name": "Incoming network traffic on $1 23",
11.      "key_": "2net.if.in[{#IFNAME}]",
12.      "delay": "1m",
13.      "history": "7d",
14.      "trends": "365d",
15.      "status": "0",
16.      "value_type": "3",
17.      "trapper_hosts": "",
18.      "units": "bps",
19.      "multiplier": "1",
20.      "delta": "1",
21.      "snmpv3_securityname": "",
22.      "snmpv3_securitylevel": "0",
23.      "snmpv3_authpassphrase": "",
24.      "snmpv3_privpassphrase": "",
25.      "formula": "8",
26.      "logtimefmt": "",
27.      "templateid": "23881",
28.      "valuemapid": "0",
29.      "params": "",
30.      "ipmi_sensor": "",
31.      "data_type": "0",
32.      "authtype": "0",
33.      "username": ""}
```

```
34.         "password": "",  
35.         "publickey": "",  
36.         "privatekey": "",  
37.         "mtime": "0",  
38.         "filter": "",  
39.         "interfaceid": "119",  
40.         "port": "",  
41.         "description": "",  
42.         "snmpv3_authprotocol": "0",  
43.         "snmpv3_privprotocol": "0"  
44.     },  
45.     {  
46.         "itemid": "27428",  
47.         "type": "0",  
48.         "snmp_community": "",  
49.         "snmp_oid": "",  
50.         "hostid": "10202",  
51.         "name": "Incoming network traffic on $1",  
52.         "key_": "net.if.in[{#IFNAME}]",  
53.         "delay": "1m",  
54.         "history": "7d",  
55.         "trends": "365d",  
56.         "status": "0",  
57.         "value_type": "3",  
58.         "trapper_hosts": "",  
59.         "units": "bps",  
60.         "multiplier": "1",  
61.         "delta": "1",  
62.         "snmpv3_securityname": "",  
63.         "snmpv3_securitylevel": "0",  
64.         "snmpv3_authpassphrase": "",  
65.         "snmpv3_privpassphrase": "",  
66.         "formula": "8",  
67.         "logtimefmt": "",  
68.         "templateid": "22446",  
69.         "valuemapid": "0",  
70.         "params": "",  
71.         "ipmi_sensor": "",  
72.         "data_type": "0",  
73.         "authtype": "0",  
74.         "username": "",  
75.         "password": "",  
76.         "publickey": "",  
77.         "privatekey": "",  
78.         "mtime": "0",  
79.         "filter": "",  
80.         "interfaceid": "119",  
81.         "port": "",  
82.         "description": "",  
83.         "snmpv3_authprotocol": "0",  
84.         "snmpv3_privprotocol": "0"  
85.     }  
86. ],
```

```
87.      "id": 1  
88. }
```

See also 参见

- [Application](#)
- [Host](#)
- [Graph prototype](#)
- [Trigger prototype](#)

Source 来源

`CItemPrototype::get()` in
frontends/php/include/classes/api/services/CItemPrototype.php.

itemprototype.update

Description 说明

```
object itemprototype.update(object/array itemPrototypes)
```

This method allows to update existing item prototypes.此方法允许更新现有item prototypes。

Parameters 参数

(object/array) Item prototype properties to be updated.

The `itemid` property must be defined for each item prototype, all other properties are optional. Only the passed properties will be updated, all others will remain unchanged.必须为每个项目原型定义“itemid”属性，所有其他属性都是可选的。只有通过的属性将被更新，所有其他属性将保持不变。

Additionally to the [standard item prototype properties](#), the method accepts the following parameters.除标准项原型属性外，该方法还接受以下参数。

属性	类型	说明
applications	array	IDs of the applications to replace the current applications.
applicationPrototypes	array	Names of the application prototypes to replace the current application prototypes.

Return values 返回值

(object) Returns an object containing the IDs of the updated item prototypes under the `itemid` property.返回包含“itemid”属性下更新的项目原型的ID的对象。

Examples 示例

Changing the interface of an item prototype

Change the host interface that will be used by discovered items.

Request:

```

1. {
2.   "jsonrpc": "2.0",
3.   "method": "itemprototype.update",
4.   "params": {
5.     "itemid": "27428",
6.     "interfaceid": "132"
7.   },

```

```
8.     "auth": "038e1d7b1735c6a5436ee9eae095879e",
9.     "id": 1
10. }
```

Response:

```
1. {
2.     "jsonrpc": "2.0",
3.     "result": {
4.         "itemids": [
5.             "27428"
6.         ],
7.     },
8.     "id": 1
9. }
```

Source 来源

CItemPrototype::update() in
frontends/php/include/classes/api/services/CItemPrototype.php.

LLD rule LLD规则

This class is designed to work with low level discovery rules. 此类旨在与低级别的发现规则配合使用。

Object references:对象引用:

- [LLD rule](#)

Available methods:可用的方法:

- [discoveryrule.copy](#) - copying LLD rules 复制LLD规则
- [discoveryrule.create](#) - creating new LLD rules 创建新的LLD规则
- [discoveryrule.delete](#) - deleting LLD rules 删除LLD规则
- [discoveryrule.get](#) - retrieving LLD rules 检索LLD规则
- [discoveryrule.update](#) - updating LLD rules 更新LLD规则

> LLD rule object LLD规则对象

The following objects are directly related to the [discoveryrule API](#). 以下对象与“discoveryrule”API直接相关。

LLD rule LLD规则

The low-level discovery rule object has the following properties. 低级别发现规则对象具有以下属性。

属性	类型	说明
<code>itemid</code>	string	(readonly) ID of the LLD rule.
<code>delay(required)</code>	string	Update interval of the LLD rule. Accepts seconds or time unit with suffix and with or without one or more custom intervals that consist of either flexible intervals and scheduling intervals as serialized strings. Also accepts user macros. Flexible intervals could be written as two macros separated by a forward slash. Intervals are separated by a semicolon.
<code>hostid(required)</code>	string	ID of the host that the LLD rule belongs to.
<code>interfaceid(required)</code>	string	ID of the LLD rule's host interface. Used only for host LLD rules. Optional for Zabbix agent (active), Zabbix internal, Zabbix trapper and database monitor LLD rules.
<code>key_(required)</code>	string	LLD rule key.
<code>name(required)</code>	string	Name of the LLD rule.
<code>type(required)</code>	integer	Type of the LLD rule. Possible values: 0 - Zabbix agent; 1 - SNMPv1 agent; 2 - Zabbix trapper; 3 - simple check; 4 - SNMPv2 agent; 5 - Zabbix internal; 6 - SNMPv3 agent; 7 - Zabbix agent (active); 10 - external check; 11 - database monitor; 12 - IPMI agent; 13 - SSH agent; 14 - TELNET agent; 16 - JMX agent.
<code>authtype</code>	integer	SSH authentication method. Used only by SSH agent LLD rules. Possible values: 0 - (default) password; 1 - public key.
<code>description</code>	string	Description of the LLD rule.
<code>error</code>	string	(readonly) Error text if there are problems updating the LLD rule.
<code>ipmisensor</code>	string	IPMI sensor. Used only by IPMI LLD rules.
<code>lifetime</code>	string	Time period after which items that are no longer discovered will be deleted. Accepts seconds, time unit with suffix and user macro. Default: <code>30d</code> .
<code>params</code>	string	Additional parameters depending on the type of the LLD rule: - executed script for SSH and Telnet LLD rules; - SQL query for database monitor LLD rules; - formula for calculated LLD

		rules.
password	string	Password for authentication. Used by simple check, SSH, Telnet, database monitor and JMX LLD rules.
port	string	Port used by the LLD rule. Used only by SNMP LLD rules.
privatekey	string	Name of the private key file.
publickey	string	Name of the public key file.
snmp_community	string	SNMP community. Required for SNMPv1 and SNMPv2 LLD rules.
snmp_oid	string	SNMP OID.
snmpv3_authpassphrase	string	SNMPv3 auth passphrase. Used only by SNMPv3 LLD rules.
snmpv3_authprotocol	integer	SNMPv3 authentication protocol. Used only by SNMPv3 LLD rules. Possible values: 0 - (default) MD5; 1 - SHA.
snmpv3_contextname	string	SNMPv3 context name. Used only by SNMPv3 checks.
snmpv3_privpassphrase	string	SNMPv3 priv passphrase. Used only by SNMPv3 LLD rules.
snmpv3_privprotocol	integer	SNMPv3 privacy protocol. Used only by SNMPv3 LLD rules. Possible values: 0 - (default) DES; 1 - AES.
snmpv3_securitylevel	integer	SNMPv3 security level. Used only by SNMPv3 LLD rules. Possible values: 0 - noAuthNoPriv; 1 - authNoPriv; 2 - authPriv.
snmpv3_securityname	string	SNMPv3 security name. Used only by SNMPv3 LLD rules.
state	integer	(readonly) State of the LLD rule. Possible values: 0 - (default) normal; 1 - not supported.
status	integer	Status of the LLD rule. Possible values: 0 - (default) enabled LLD rule; 1 - disabled LLD rule.
templateid	string	(readonly) ID of the parent template LLD rule.
trapper_hosts	string	Allowed hosts. Used only by trapper LLD rules.
username	string	Username for authentication. Used by simple check, SSH, Telnet, database monitor and JMX LLD rules. Required by SSH and Telnet LLD rules.

LLD rule filter LLD规则过滤器

The LLD rule filter object defines a set of conditions that can be used to filter discovered objects. It has the following properties: LLD规则过滤器对象定义了一组可用于过滤发现对象的条件。 它具有以下属性：

属性	类型	说明
conditions (required)	array	Set of filter conditions to use for filtering results.

evaltype (required)	integer	Filter condition evaluation method. Possible values: 0 - and/or; 1 - and; 2 - or; 3 - custom expression.
evalformula	string	(readonly) Generated expression that will be used for evaluating filter conditions. The expression contains IDs that reference specific filter conditions by its <code>formulaid</code> . The value of <code>eval_formula</code> is equal to the value of <code>formula</code> for filters with a custom expression.
formula	string	User-defined expression to be used for evaluating conditions of filters with a custom expression. The expression must contain IDs that reference specific filter conditions by its <code>formulaid</code> . The IDs used in the expression must exactly match the ones defined in the filter conditions: no condition can remain unused or omitted. Required for custom expression filters.

LLD rule filter condition LLD规则过滤条件

The LLD rule filter condition object defines a separate check to perform on the value of an LLD macro. It has the following properties: LLD规则过滤条件对象定义了对LLD宏的值执行的单独检查。它具有以下属性：

属性	类型	说明
macro (required)	string	LLD macro to perform the check on.
value (required)	string	Value to compare with.
formulaid	string	Arbitrary unique ID that is used to reference the condition from a custom expression. Can only contain capital-case letters. The ID must be defined by the user when modifying filter conditions, but will be generated anew when requesting them afterward.
operator	integer	Condition operator. Possible values: 8 - (default) matches regular expression.

To better understand how to use filters with various types of expressions, see examples on the [discoveryrule.get](#) and [discoveryrule.create](#) method pages.

discoverrule.copy

Description 说明

```
object discoverrule.copy(object parameters)
```

This method allows to copy LLD rules with all of the prototypes to the given hosts.该方法允许将LLD规则与所有原型复制到给定的主机

Parameters 参数

(object) Parameters defining the LLD rules to copy and the target hosts.定义要复制的LLD规则和目标主机的参数。

属性	类型	说明
discoveryids	array	IDs of the LLD rules to be copied. 要复制的LLD规则的ID。
hostids	array	IDs of the hosts to copy the LLD rules to.要将LLD规则复制到的主机的ID。

Return values 返回值

(boolean) Returns true if the copying was successful. (boolean) 如果复制成功，则返回 true。

Examples 示例

Copy an LLD rule to multiple hosts

Copy an LLD rule to two hosts.

Request:

```

1. {
2.     "jsonrpc": "2.0",
3.     "method": "discoverrule.copy",
4.     "params": {
5.         "discoveryids": [
6.             "27426"
7.         ],
8.         "hostids": [
9.             "10196",
10.            "10197"
11.        ]
12.    },
13.    "auth": "038e1d7b1735c6a5436ee9eae095879e",
14.    "id": 1

```

```
15. }
```

Response:

```
1. {
2.     "jsonrpc": "2.0",
3.     "result": true,
4.     "id": 1
5. }
```

Source 来源

`CDiscoveryrule::copy()` in
frontends/php/include/classes/api/services/CDiscoveryRule.php.

discoveryrule.create

Description 说明

```
object discoveryrule.create(object/array lldRules)
```

This method allows to create new LLD rules. 此方法允许创建新的LLD规则。

Parameters 参数

```
(object/array) LLD rules to create.
```

Additionally to the [standard LLD rule properties](#), the method accepts the following parameters. 除了标准LLD规则属性之外，该方法接受以下参数。

属性	类型	说明
filter	object	LLD rule filter object for the LLD rule. LLD规则的LLD规则过滤器对象。

Return values 返回值

```
(object) Returns an object containing the IDs of the created LLD rules under the itemids property. The order of the returned IDs matches the order of the passed LLD rules. 返回一个包含“itemid”属性下创建的LLD规则的ID的对象。 返回的ID的顺序与传递的LLD规则的顺序相匹配。
```

Examples 示例

Creating an LLD rule

Create a Zabbix agent LLD rule to discover mounted file systems. Discovered items will be updated every 30 seconds.

Request:

```

1. {
2.   "jsonrpc": "2.0",
3.   "method": "discoveryrule.create",
4.   "params": {
5.     "name": "Mounted filesystem discovery",
6.     "key_": "vfs.fs.discovery",
7.     "hostid": "10197",
8.     "type": "0",
9.     "interfaceid": "112",
10.    "delay": "30s"
11.  },
12.  "auth": "038e1d7b1735c6a5436ee9eae095879e",

```

```

13.      "id": 1
14.  }

```

Response:

```

1.  {
2.      "jsonrpc": "2.0",
3.      "result": {
4.          "itemids": [
5.              "27665"
6.          ]
7.      },
8.      "id": 1
9.  }

```

Using a filter

Create an LLD rule with a set of conditions to filter the results by. The conditions will be grouped together using the logical “and” operator.

Request:

```

1.  {
2.      "jsonrpc": "2.0",
3.      "method": "discoverrule.create",
4.      "params": {
5.          "name": "Filtered LLD rule",
6.          "key_": "lld",
7.          "hostid": "10116",
8.          "type": "0",
9.          "interfaceid": "13",
10.         "delay": "30s",
11.         "filter": {
12.             "evaltype": 1,
13.             "conditions": [
14.                 {
15.                     "macro": "{#MACRO1}",
16.                     "value": "@regex1"
17.                 },
18.                 {
19.                     "macro": "{#MACRO2}",
20.                     "value": "@regex2"
21.                 },
22.                 {
23.                     "macro": "{#MACRO3}",
24.                     "value": "@regex3"
25.                 }
26.             ]
27.         },
28.         "auth": "038e1d7b1735c6a5436ee9eae095879e",
29.     }

```

```

30.      "id": 1
31. }

```

Response:

```

1. {
2.   "jsonrpc": "2.0",
3.   "result": {
4.     "itemids": [
5.       "27665"
6.     ]
7.   },
8.   "id": 1
9. }

```

Using a custom expression filter

Create an LLD rule with a filter that will use a custom expression to evaluate the conditions. The LLD rule must only discover objects the “{#MACRO1}” macro value of which matches both regular expression “regex1” and “regex2”, and the value of “{#MACRO2}” matches either “regex3” or “regex4”. The formula IDs “A”, “B”, “C” and “D” have been chosen arbitrarily.

Request:

```

1. {
2.   "jsonrpc": "2.0",
3.   "method": "discoveryrule.create",
4.   "params": {
5.     "name": "Filtered LLD rule",
6.     "key_": "lld",
7.     "hostid": "10116",
8.     "type": "0",
9.     "interfaceid": "13",
10.    "delay": "30s",
11.    "filter": {
12.      "evaltype": 3,
13.      "formula": "(A and B) and (C or D)",
14.      "conditions": [
15.        {
16.          "macro": "{#MACRO1}",
17.          "value": "@regex1",
18.          "formulaid": "A"
19.        },
20.        {
21.          "macro": "{#MACRO1}",
22.          "value": "@regex2",
23.          "formulaid": "B"
24.        },
25.        {
26.          "macro": "{#MACRO2}",

```

```

27.          "value": "@regex3",
28.          "formulaid": "C"
29.        },
30.      {
31.        "macro": "{#MACRO2}",
32.        "value": "@regex4",
33.        "formulaid": "D"
34.      }
35.    ]
36.  }
37. },
38. "auth": "038e1d7b1735c6a5436ee9eae095879e",
39. "id": 1
40. }
```

Response:

```

1. {
2.   "jsonrpc": "2.0",
3.   "result": {
4.     "itemids": [
5.       "27665"
6.     ]
7.   },
8.   "id": 1
9. }
```

See also 参见

- [LLD rule filter](#)

Source 来源

`CDiscoveryRule::create()` in
frontends/php/include/classes/api/services/CDiscoveryRule.php.

discoverrule.delete

Description 说明

```
object discoverrule.delete(array lldRuleIds)
```

This method allows to delete LLD rules.此方法允许删除LLD规则。

Parameters 参数

(array) IDs of the LLD rules to delete.要删除的LLD规则的ID。

Return values 返回值

(object) Returns an object containing the IDs of the deleted LLD rules under the itemids property.返回一个包含“itemid”属性下删除的LLD规则的ID的对象。

Examples 示例

Deleting multiple LLD rules

Delete two LLD rules.

Request:

```
1. {
2.   "jsonrpc": "2.0",
3.   "method": "discoverrule.delete",
4.   "params": [
5.     "27665",
6.     "27668"
7.   ],
8.   "auth": "3a57200802b24cda67c4e4010b50c065",
9.   "id": 1
10. }
```

Response:

```
1. {
2.   "jsonrpc": "2.0",
3.   "result": {
4.     "ruleids": [
5.       "27665",
6.       "27668"
7.     ]
8.   },
9.   "id": 1
}
```

discoveryrule.delete

```
10. }
```

Source 来源

`CDiscoveryRule::delete() in
frontends/php/include/classes/api/services/CDiscoveryRule.php.`

discoverrule.get

Description

```
integer/array discoverrule.get(object parameters)
```

The method allows to retrieve LLD rules according to the given parameters.该方法允许根据给定的参数检索LLD规则。

Parameters 参数

(object) Parameters defining the desired output.定义所需输出的参数。

The method supports the following parameters.该方法支持以下参数。

属性	类型	说明
itemids	string/array	Return only LLD rules with the given IDs.
hostids	string/array	Return only LLD rules that belong to the given hosts.
inherited	boolean	If set to <code>true</code> return only LLD rules inherited from a template.
interfaceids	string/array	Return only LLD rules use the given host interfaces.
monitored	boolean	If set to <code>true</code> return only enabled LLD rules that belong to monitored hosts.
templated	boolean	If set to <code>true</code> return only LLD rules that belong to templates.
templateids	string/array	Return only LLD rules that belong to the given templates.
selectFilter	query	Returns the filter used by the LLD rule in the <code>filter</code> property.
selectGraphs	query	Returns graph prototypes that belong to the LLD rule in the <code>graphs</code> property. Supports <code>count</code> .
selectHostPrototypes	query	Returns host prototypes that belong to the LLD rule in the <code>hostPrototypes</code> property. Supports <code>count</code> .
selectHosts	query	Returns the host that the LLD rule belongs to as an array in the <code>hosts</code> property.
selectItems	query	Returns item prototypes that belong to the LLD rule in the <code>items</code> property. Supports <code>count</code> .
selectTriggers	query	Returns trigger prototypes that belong to the LLD rule in the <code>triggers</code> property. Supports <code>count</code> .

filter	object	Return only those results that exactly match the given filter. Accepts an array, where the keys are property names, and the values are either a single value or an array of values to match against. Supports additional filters: <code>host</code> - technical name of the host that the LLD rule belongs to.
limitSelects	integer	Limits the number of records returned by subselects. Applies to the following subselects: <code>selectItems</code> ; <code>selectGraphs</code> ; <code>selectTriggers</code> .
sortfield	string/array	Sort the result by the given properties. Possible values are: <code>itemid</code> , <code>name</code> , <code>key_</code> , <code>delay</code> , <code>type</code> and <code>status</code> .
countOutput	flag	These parameters being common for all <code>get</code> methods are described in detail in the reference commentary .
editable	boolean	
excludeSearch	flag	
limit	integer	
output	query	
preservekeys	flag	
search	object	
searchByAny	boolean	
searchWildcardsEnabled	boolean	
sortorder	string/array	
startSearch	flag	

Return values 返回值

(integer/array) Returns either:

- an array of objects;
- the count of retrieved objects, if the `countOutput` parameter has been used.

Examples 示例

Retrieving discovery rules from a host

Retrieve all discovery rules from host "10202".

Request:

```

1. {
2.   "jsonrpc": "2.0",
3.   "method": "discoveryrule.get",
4.   "params": {

```

```

5.      "output": "extend",
6.      "hostids": "10202"
7.    },
8.    "auth": "038e1d7b1735c6a5436ee9eae095879e",
9.    "id": 1
10. }

```

Response:

```

1. {
2.   "jsonrpc": "2.0",
3.   "result": [
4.     {
5.       "itemid": "27425",
6.       "type": "0",
7.       "snmp_community": "",
8.       "snmp_oid": "",
9.       "hostid": "10202",
10.      "name": "Network interface discovery",
11.      "key_": "net.if.discovery",
12.      "delay": "1h",
13.      "state": "0",
14.      "status": "0",
15.      "trapper_hosts": "",
16.      "snmpv3_securityname": "",
17.      "snmpv3_securitylevel": "0",
18.      "snmpv3_authpassphrase": "",
19.      "snmpv3_privpassphrase": "",
20.      "error": "",
21.      "templateid": "22444",
22.      "params": "",
23.      "ipmi_sensor": "",
24.      "authtype": "0",
25.      "username": "",
26.      "password": "",
27.      "publickey": "",
28.      "privatekey": "",
29.      "interfaceid": "119",
30.      "port": "",
31.      "description": "Discovery of network interfaces as defined in global regular expression \\\"Network
32.      interfaces for discovery\\\".",
33.      "lifetime": "30d",
34.      "snmpv3_authprotocol": "0",
35.      "snmpv3_privprotocol": "0"
35.    },
36.    {
37.      "itemid": "27426",
38.      "type": "0",
39.      "snmp_community": "",
40.      "snmp_oid": "",
41.      "hostid": "10202",
42.      "name": "Mounted filesystem discovery",
43.      "key_": "vfs.fs.discovery",

```

```

44.         "delay": "1h",
45.         "state": "0",
46.         "status": "0",
47.         "trapper_hosts": "",
48.         "snmpv3_securityname": "",
49.         "snmpv3_securitylevel": "0",
50.         "snmpv3_authpassphrase": "",
51.         "snmpv3_privpassphrase": "",
52.         "error": "",
53.         "templateid": "22450",
54.         "params": "",
55.         "ipmi_sensor": "",
56.         "authtype": "0",
57.         "username": "",
58.         "password": "",
59.         "publickey": "",
60.         "privatekey": "",
61.         "interfaceid": "119",
62.         "port": "",
63.         "description": "Discovery of file systems of different types as defined in global regular expression \\"File systems for discovery\\",
64.         "lifetime": "30d",
65.         "snmpv3_authprotocol": "0",
66.         "snmpv3_privprotocol": "0"
67.     },
68. ],
69. "id": 2
70. }

```

Retrieving filter conditions

Retrieve the name of the LLD rule “24681” and its filter conditions. The filter uses the “and” evaluation type, so the `formula` property is empty and `eval_formula` is generated automatically.

Request:

```

1. {
2.     "jsonrpc": "2.0",
3.     "method": "discoverrule.get",
4.     "params": {
5.         "output": [
6.             "name"
7.         ],
8.         "selectFilter": "extend",
9.         "itemids": ["24681"]
10.    },
11.    "auth": "038e1d7b1735c6a5436ee9eae095879e",
12.    "id": 1
13. }

```

Response:

```

1. {
2.     "jsonrpc": "2.0",
3.     "result": [
4.         {
5.             "itemid": "24681",
6.             "name": "Filtered LLD rule",
7.             "filter": {
8.                 "evaltype": "1",
9.                 "formula": "",
10.                "conditions": [
11.                    {
12.                        "macro": "{#MACRO1}",
13.                        "value": "@regex1",
14.                        "operator": "8",
15.                        "formulaid": "A"
16.                    },
17.                    {
18.                        "macro": "{#MACRO2}",
19.                        "value": "@regex2",
20.                        "operator": "8",
21.                        "formulaid": "B"
22.                    },
23.                    {
24.                        "macro": "{#MACRO3}",
25.                        "value": "@regex3",
26.                        "operator": "8",
27.                        "formulaid": "C"
28.                    }
29.                ],
30.                "eval_formula": "A and B and C"
31.            }
32.        },
33.    ],
34.    "id": 1
35. }
```

See also 参见

- [Graph prototype](#)
- [Host](#)
- [Item prototype](#)
- [LLD rule filter](#)
- [Trigger prototype](#)

Source 来源

discoveryrule.get

CDiscoveryRule::get() in
frontends/php/include/classes/api/services/CDiscoveryRule.php.

discoverrule.update

Description 说明

`object discoverrule.update(object/array lldRules)`

This method allows to update existing LLD rules. 此方法允许更新现有的LLD规则。

Parameters 参数

`(object/array)` LLD rule properties to be updated. 要更新的LLD规则属性。

The `itemid` property must be defined for each LLD rule, all other properties are optional. Only the passed properties will be updated, all others will remain unchanged. 必须为每个LLD规则定义“itemid”属性，所有其他属性都是可选的。 只有通过的属性将被更新，所有其他属性将保持不变。

Additionally to the `standard LLD rule properties`, the method accepts the following parameters. 除了标准LLD规则属性之外，该方法接受以下参数。

属性	类型	说明
filter	object	LLD rule filter object to replace the current filter.

Return values 返回值

`(object)` Returns an object containing the IDs of the updated LLD rules under the `itemids` property.

Examples 示例

Adding a filter to an LLD rule

Add a filter so that the contents of the `{#FSTYPE}` macro would match the `@File systems for discovery` regexp.

Request:

```

1. {
2.   "jsonrpc": "2.0",
3.   "method": "discoverrule.update",
4.   "params": {
5.     "itemid": "24682",
6.     "filter": {
7.       "evaltype": 1,
8.       "conditions": [
9.         {

```

```
10.          "macro": "{#FSTYPE}",
11.          "value": "@File systems for discovery"
12.      }
13.  ]
14. }
15. },
16. "auth": "038e1d7b1735c6a5436ee9eae095879e",
17. "id": 1
18. }
```

Response:

```
1. {
2.   "jsonrpc": "2.0",
3.   "result": {
4.     "itemids": [
5.       "22450"
6.     ]
7.   },
8.   "id": 1
9. }
```

Source 来源

`CDiscoveryRule::update()` in
frontends/php/include/classes/api/services/CDiscoveryRule.php.

Maintenance 维护

This class is designed to work with maintenances. 此类讲解了maintenances的使用

Object references: 对象引用:

- [Maintenance](#)
- [Time period](#)

Available methods: 可用的方法:

- [maintenance.create](#) - creating new maintenances
- [maintenance.delete](#) - deleting maintenances
- [maintenance.get](#) - retrieving maintenances
- [maintenance.update](#) - updating maintenances

> Maintenance object

The following objects are directly related to the [maintenance API](#). 以下对象与“维护”API直接相关

Maintenance 维护

The maintenance object has the following properties. 维护对象具有以下属性。

属性	类型	说明
maintenanceid	string	(readonly) ID of the maintenance.
name(required)	string	Name of the maintenance.
activesince	timestamp	Time when the maintenance becomes active. Default: current time.
active_till	timestamp	Time when the maintenance stops being active. Default: the next day.
description	string	Description of the maintenance.
maintenance_type	integer	Type of maintenance. Possible values: 0 - (default) with data collection; 1 - without data collection.

Time period 时间段

The time period object is used to define periods when the maintenance must come into effect. It has the following properties. 时间段对象用于定义维护生效期间。 它具有以下属性。

属性	类型	说明
timeperiodid	string	(readonly) ID of the maintenance.
day	integer	Day of the month when the maintenance must come into effect. Required only for monthly time periods.
dayofweek	integer	Days of the week when the maintenance must come into effect. Days are stored in binary form with each bit representing the corresponding day. For example, 4 equals 100 in binary and means, that maintenance will be enabled on Wednesday. Used for weekly and monthly time periods. Required only for weekly time periods.
every	integer	For daily and weekly periods <code>every</code> defines day or week intervals at which the maintenance must come into effect. For monthly periods <code>every</code> defines the week of the month when the maintenance must come into effect. Possible values: 1 - first week; 2 - second week; 3 - third week; 4 - fourth week; 5 - last week.
month	integer	Months when the maintenance must come into effect. Months are stored in binary form with each bit representing the corresponding month. For example, 5 equals 101 in binary and means, that maintenance

		will be enabled in January and March. Required only for monthly time periods.
period	integer	Duration of the maintenance period in seconds. Default: 3600.
startdate	timestamp	Date when the maintenance period must come into effect. Required only for one time periods. Default: current date.
start_time	integer	Time of day when the maintenance starts in seconds. Required for daily, weekly and monthly periods.
timeperiod_type	integer	Type of time period. Possible values: 0 - (default) - one time only; 2 - daily; 3 - weekly; 4 - monthly.

maintenance.create

Description 说明

```
object maintenance.create(object/array maintenances)
```

This method allows to create new maintenances.这种方法允许创建新的维护。

Parameters 参数

(object/array) Maintenances to create.要创建的Maintenances。

Additionally to the standard maintenance properties, the method accepts the following parameters.除标准维护属性外，该方法还接受以下参数。

属性	类型	说明
groupids(required)	array	IDs of the host groups that will undergo maintenance.
hostids(required)	array	IDs of the hosts that will undergo maintenance.
timeperiods(required)	array	Maintenance time periods.

At least one host or host group must be defined for each maintenance.

Return values 返回值

(object) Returns an object containing the IDs of the created maintenances under the maintenancesids property. The order of the returned IDs matches the order of the passed maintenances.返回一个包含“维护”属性下的创建维护ID的对象。返回的ID的顺序与通过的维护的顺序相匹配。

Examples 示例

Creating a maintenance

Create a maintenance with data collection for host group "2". It must be active from 22.01.2013 till 22.01.2014, come in effect each Sunday at 18:00 and last for one hour.

Request:

```

1. {
2.   "jsonrpc": "2.0",
3.   "method": "maintenance.create",
4.   "params": {
5.     "name": "Sunday maintenance",
6.     "active_since": 1358844540,
7.     "active_till": 1390466940,
8.     "groupids": [

```

```

9.          "2"
10.         ],
11.         "timeperiods": [
12.           {
13.             "timeperiod_type": 3,
14.             "every": 1,
15.             "dayofweek": 64,
16.             "start_time": 64800,
17.             "period": 3600
18.           }
19.         ]
20.       },
21.       "auth": "038e1d7b1735c6a5436ee9eae095879e",
22.       "id": 1
23.     }

```

Response:

```

1.  {
2.    "jsonrpc": "2.0",
3.    "result": {
4.      "maintenanceids": [
5.        "3"
6.      ]
7.    },
8.    "id": 1
9.  }

```

See also 参见

- [Time period](#)

Source 来源

`CMaintenance::create()` in *frontends/php/include/classes/api/services/CMaintenance.php*.

maintenance.delete

Description 说明

```
object maintenance.delete(array maintenanceIds)
```

This method allows to delete maintenances.该方法允许删除维护。

Parameters 参数

(array) IDs of the maintenances to delete.要删除的维护ID。

Return values 返回值

(object) Returns an object containing the IDs of the deleted maintenances under the maintenanceids property.返回一个包含“维护”属性下的已删除维护ID的对象。

Examples 示例

Deleting multiple maintenances

Delete two maintenances.

Request:

```
1. {
2.   "jsonrpc": "2.0",
3.   "method": "maintenance.delete",
4.   "params": [
5.     "3",
6.     "1"
7.   ],
8.   "auth": "3a57200802b24cda67c4e4010b50c065",
9.   "id": 1
10. }
```

Response:

```
1. {
2.   "jsonrpc": "2.0",
3.   "result": {
4.     "maintenanceids": [
5.       "3",
6.       "1"
7.     ]
8.   },
9.   "id": 1
}
```

```
10. }
```

Source 来源

CMaintenance::delete() in *frontends/php/include/classes/api/services/CMaintenance.php*.

maintenance.get

Description 说明

```
integer/array maintenance.get(object parameters)
```

The method allows to retrieve maintenances according to the given parameters.该方法允许根据给定的参数检索维护。

Parameters 参数

(object) Parameters defining the desired output.定义所需输出的参数。

The method supports the following parameters.该方法支持以下参数。

属性	类型	说明
groupids	string/array	Return only maintenances that are assigned to the given host groups.
hostids	string/array	Return only maintenances that are assigned to the given hosts.
maintenanceids	string/array	Return only maintenances with the given IDs.
selectGroups	query	Return host groups assigned to the maintenance in the <code>groups</code> property.
selectHosts	query	Return hosts assigned to the maintenance in the <code>hosts</code> property.
selectTimeperiods	query	Return the maintenance's time periods in the <code>timeperiods</code> property.
sortfield	string/array	Sort the result by the given properties. Possible values are: <code>maintenanceid</code> , <code>name</code> and <code>maintenance_type</code> .
countOutput	flag	These parameters being common for all <code>get</code> methods are described in detail in the reference commentary .
editable	boolean	
excludeSearch	flag	
filter	object	
limit	integer	
output	query	
preservekeys	flag	
search	object	
searchByAny	boolean	
searchWildcardsEnabled	boolean	

sortorder	string/array
startSearch	flag

Return values 返回值

(integer/array) Returns either:

- an array of objects;
- the count of retrieved objects, if the `countOutput` parameter has been used.

Examples 示例

Retrieving maintenances

Retrieve all configured maintenances, and the data about the assigned host groups, hosts and defined time periods.

Request:

```

1. {
2.   "jsonrpc": "2.0",
3.   "method": "maintenance.get",
4.   "params": {
5.     "output": "extend",
6.     "selectGroups": "extend",
7.     "selectTimeperiods": "extend"
8.   },
9.   "auth": "038e1d7b1735c6a5436ee9eae095879e",
10.  "id": 1
11. }
```

Response:

```

1. {
2.   "jsonrpc": "2.0",
3.   "result": [
4.     {
5.       "maintenanceid": "3",
6.       "name": "Sunday maintenance",
7.       "maintenance_type": "0",
8.       "description": "",
9.       "active_since": "1358844540",
10.      "active_till": "1390466940",
11.      "groups": [
12.        {
13.          "groupid": "4",
14.          "name": "Zabbix servers",
15.          "internal": "0"
```

```
16.          }
17.      ],
18.      "timeperiods": [
19.          {
20.              "timeperiodid": "4",
21.              "timeperiod_type": "3",
22.              "every": "1",
23.              "month": "0",
24.              "dayofweek": "1",
25.              "day": "0",
26.              "start_time": "64800",
27.              "period": "3600",
28.              "start_date": "2147483647"
29.          }
30.      ]
31.  },
32.  ],
33.  "id": 1
34. }
```

See also 参见

- [Host](#)
- [Host group](#)
- [Time period](#)

Source 来源

CMaintenance::get() in *frontends/php/include/classes/api/services/CMaintenance.php*.

maintenance.update

Description 说明

```
object maintenance.update(object/array maintenances)
```

This method allows to update existing maintenances. 该方法允许更新现有维护。

Parameters 参数

(object/array) Maintenance properties to be updated. 要更新的维护属性。

The `maintenanceid` property must be defined for each maintenance, all other properties are optional. Only the passed properties will be updated, all others will remain unchanged. 必须为每个维护定义“维护ID”属性，所有其他属性都是可选的。 只有通过的属性将被更新，所有其他属性将保持不变。

At this time, partial maintenance update is not supported, all parameters are mandatory. See [ZBX-6167](#) for current status. 此时，不支持部分维护更新，所有参数都是必需的。 有关当前状态，请参阅[ZBX-6167](#)。

Additionally to the [standard maintenance properties](#), the method accepts the following parameters. 除了标准维护属性外，该方法接受以下参数。

属性	类型	说明
groupids	array	IDs of the host groups to replace the current groups.
hostids	array	IDs of the hosts to replace the current hosts.
timeperiods	array	Maintenance time periods to replace the current periods.

At least one host or host group must be defined for each maintenance.

Return values 返回值

(object) Returns an object containing the IDs of the updated maintenances under the `maintenanceids` property.

Examples 示例

Assigning different hosts

Replace the hosts currently assigned to maintenance “3” with two different ones.

Request:

```
1. {
```

```
2.     "jsonrpc": "2.0",
3.     "method": "maintenance.update",
4.     "params": {
5.         "maintenanceid": "3",
6.         "hostids": [
7.             "10085",
8.             "10084"
9.         ]
10.    },
11.    "auth": "038e1d7b1735c6a5436ee9eae095879e",
12.    "id": 1
13. }
```

Response:

```
1. {
2.     "jsonrpc": "2.0",
3.     "result": {
4.         "maintenanceids": [
5.             "3"
6.         ]
7.     },
8.     "id": 1
9. }
```

See also 参见

- [Time period](#)

Source 来源

CMaintenance::update() in *frontends/php/include/classes/api/services/CMaintenance.php*.

Map 地图

This class is designed to work with maps. 此类旨在协同使用maps

Object references: 对象引用:

- [Map](#)
- [Map element](#)
- [Map link](#)
- [Map URL](#)
- [Map user](#)
- [Map user group](#)

Available methods: 可用的方法:

- `map.create` - create new maps
- `map.delete` - delete maps
- `map.get` - retrieve maps
- `map.update` - update maps

> Map object

The following objects are directly related to the [map API](#).

Map

The map object has the following properties.

Property	Type	Description
sysmapid	string	(readonly) ID of the map.
height(required)	integer	Height of the map in pixels.
name(required)	string	Name of the map.
width(required)	integer	Width of the map in pixels.
backgroundid	string	ID of the image used as the background for the map.
expandmacros	integer	Whether to expand macros in labels when configuring the map. Possible values: 0 - (default) <i>do not expand macros</i> ; 1 - <i>expand macros</i> .
expandproblem	integer	Whether the problem trigger will be displayed for elements with a single problem. Possible values: 0 - always display the number of problems; 1 - (default) <i>display the problem trigger if there's only one problem</i> .
grid_align	integer	Whether to enable grid aligning. Possible values: 0 - disable grid aligning; 1 - (default) <i>enable grid aligning</i> .
grid_show	integer	Whether to show the grid on the map. Possible values: 0 - do not show the grid; 1 - (default) <i>show the grid</i> .
grid_size	integer	Size of the map grid in pixels. Supported values: 20, 40, 50, 75 and 100. Default: 50.
highlight	integer	Whether icon highlighting is enabled. Possible values: 0 - highlighting disabled; 1 - (default) <i>highlighting enabled</i> .
iconmapid	string	ID of the icon map used on the map.
label_format	integer	Whether to enable advanced labels. Possible values: 0 - (default) <i>disable advanced labels</i> ; 1 - <i>enable advanced labels</i> .
label_location	integer	Location of the map element label. Possible values: 0 - (default) <i>bottom</i> ; 1 - <i>left</i> ; 2 - <i>right</i> ; 3 - <i>top</i> .
label_string_host	string	Custom label for host elements. Required for maps with custom host label type.
label_string_hostgroup	string	Custom label for host group elements. Required for maps with custom host group label type.
label_string_image	string	Custom label for image elements. Required for

label_string_image	string	maps with custom image label type.
label_string_map	string	Custom label for map elements. Required for maps with custom map label type.
label_string_trigger	string	Custom label for trigger elements. Required for maps with custom trigger label type.
label_type	integer	Map element label type. Possible values: 0 - label; 1 - IP address; 2 - (default) element name; 3 - status only; 4 - nothing.
label_type_host	integer	Label type for host elements. Possible values: 0 - label; 1 - IP address; 2 - (default) element name; 3 - status only; 4 - nothing; 5 - custom.
label_type_hostgroup	integer	Label type for host group elements. Possible values: 0 - label; 2 - (default) element name; 3 - status only; 4 - nothing; 5 - custom.
label_type_image	integer	Label type for host group elements. Possible values: 0 - label; 2 - (default) element name; 4 - nothing; 5 - custom.
label_type_map	integer	Label type for map elements. Possible values: 0 - label; 2 - (default) element name; 3 - status only; 4 - nothing; 5 - custom.
label_type_trigger	integer	Label type for trigger elements. Possible values: 0 - label; 2 - (default) element name; 3 - status only; 4 - nothing; 5 - custom.
markelements	integer	Whether to highlight map elements that have recently changed their status. Possible values: 0 - (default) do not highlight elements; 1 - highlight elements.
severity_min	integer	Minimum severity of the triggers that will be displayed on the map. Refer to the trigger "severity" property for a list of supported trigger severities.
show_unack	integer	How problems should be displayed. Possible values: 0 - (default) display the count of all problems; 1 - display only the count of unacknowledged problems; 2 - display the count of acknowledged and unacknowledged problems separately.
userid	string	Map owner user ID.
private	integer	Type of map sharing. Possible values: 0 - public map; 1 - (default) private map.

Map element

The map element object defines an object displayed on a map. It has the following properties.

Property	Type	Description
selementid	string	(readonly) ID of the map element.
		ID of the object that the map element

		trigger and map type elements.
elementtype (required)	integer	Type of map element. Possible values: 0 - host; 1 - map; 2 - trigger; 3 - host group; 4 - image.
iconid_off (required)	string	ID of the image used to display the element in default state.
areatype	integer	How separate host group hosts should be displayed. Possible values: 0 - (default) the host group element will take up the whole map; 1 - the host group element will have a fixed size.
application	string	Name of the application to display problems from. Used only for host and host group map elements.
elements subtype	integer	How a host group element should be displayed on a map. Possible values: 0 - (default) display the host group as a single element; 1 - display each host in the group separately.
height	integer	Height of the fixed size host group element in pixels. Default: 200.
iconid disabled	string	ID of the image used to display disabled map elements. Unused for image elements.
iconid maintenance	string	ID of the image used to display map elements in maintenance. Unused for image elements.
iconid on	string	ID of the image used to display map elements with problems. Unused for image elements.
label	string	Label of the element.
label location	integer	Location of the map element label. Possible values: -1 - (default) <i>default location</i> ; 0 - <i>bottom</i> ; 1 - <i>left</i> ; 2 - <i>right</i> ; 3 - <i>top</i> .
sysmap id	string	(readonly) <i>ID of the map that the element belongs to</i> .
urls	array	Map element URLs. The map element URL object is described in detail below .
use iconmap	integer	Whether icon mapping must be used for host elements. Possible values: 0 - do not use icon mapping; 1 - (default) <i>use icon mapping</i> .
viewtype	integer	Host group element placing algorithm. Possible values: 0 - (default) <i>grid</i> .
width	integer	Width of the fixed size host group element in pixels. Default: 200.
x	integer	X-coordinates of the element in pixels. Default: 0.
y	integer	Y-coordinates of the element in pixels. Default: 0.

Map element URL

The map element [URL](#) object defines a clickable link that will be available for a specific map element. It has the following properties:

--	--	--

Property	Type	Description
sysmapelementurlid	string	(readonly) ID of the map element URL.
name (required)	string	Link caption.
url (required)	string	Link URL.
selementid	string	ID of the map element that the URL belongs to.

Map link

The map link object defines a link between two map elements. It has the following properties.

Property	Type	Description
linkid	string	(readonly) ID of the map link.
selementid1 (required)	string	ID of the first map element linked on one end.
selementid2 (required)	string	ID of the first map element linked on the other end.
color	string	Line color as a hexadecimal color code. Default: <code>000000</code> .
drawtype	integer	Link line draw style. Possible values: 0 - (default) line; 2 - bold line; 3 - dotted line; 4 - dashed line.
label	string	Link label.
linktriggers	array	Map link triggers to use as link status indicators. The map link trigger object is described in detail below .
sysmapid	string	ID of the map the link belongs to.

Map link trigger

The map link trigger object defines a map link status indicator based on the state of a trigger. It has the following properties:

Property	Type	Description
linktriggerid	string	(readonly) ID of the map link trigger.
triggerid (required)	string	ID of the trigger used as a link indicator.
color	string	Indicator color as a hexadecimal color code. Default: <code>DD0000</code> .
drawtype	integer	Indicator draw style. Possible values: 0 - (default) line; 2 - bold line; 3 - dotted line; 4 - dashed line.
linkid	string	ID of the map link that the link trigger belongs to.

Map URL

The `map URL` object defines a clickable link that will be available for all elements of a specific type on the map. It has the following properties:

Property	Type	Description
<code>sysmapurlid</code>	<code>string</code>	(<i>readonly</i>) ID of the map <code>URL</code> .
<code>name(required)</code>	<code>string</code>	Link caption.
<code>url(required)</code>	<code>string</code>	Link <code>URL</code> .
<code>elementtype</code>	<code>integer</code>	Type of map element for which the <code>URL</code> will be available. Refer to the <code>map element "type"</code> property for a list of supported types. Default: 0.
<code>sysmapid</code>	<code>string</code>	ID of the map that the <code>URL</code> belongs to.

Map user

List of map permissions based on users. It has the following properties:

Property	Type	Description
<code>sysmapuserid</code>	<code>string</code>	(<i>readonly</i>) ID of the map user.
<code>userid(required)</code>	<code>string</code>	User ID.
<code>permission(required)</code>	<code>integer</code>	Type of permission level. Possible values: 2 - read only; 3 - read-write;

Map user group

List of map permissions based on user groups. It has the following properties:

Property	Type	Description
<code>sysmapusrgpid</code>	<code>string</code>	(<i>readonly</i>) ID of the map user group.
<code>usrgrpid(required)</code>	<code>string</code>	User group ID.
<code>permission(required)</code>	<code>integer</code>	Type of permission level. Possible values: 2 - read only; 3 - read-write;

map.create

Description

```
object map.create(object/array maps)
```

This method allows to create new maps.

Parameters

(object/array) Maps to create.

Additionally to the [standard map properties](#), the method accepts the following parameters.

Parameter	Type	Description
links	array	Map links to be created on the map.
selements	array	Map elements to be created on the map.
urls	array	Map URLs to be created on the map.
users	array	Map user shares to be created on the map.
userGroups	array	Map user group shares to be created on the map.

To create map links you'll need to set a map elements `selementid` to an arbitrary value and then use this value to reference this element in the links `selementid1` or `selementid2` properties. When the element is created, this value will be replaced with the correct ID generated by Zabbix. See [example](#).

Return values

(object) Returns an object containing the IDs of the created maps under the `sysmapids` property. The order of the returned IDs matches the order of the passed maps.

Examples

Create an empty map

Create a map with no elements.

Request:

```
1. {
2.     "jsonrpc": "2.0",
3.     "method": "map.create",
4.     "params": {
5.         "name": "Map",
```

```
map.create
```

```
6.      "width": 600,
7.      "height": 600
8.    },
9.    "auth": "038e1d7b1735c6a5436ee9eae095879e",
10.   "id": 1
11. }
```

Response:

```
1. {
2.   "jsonrpc": "2.0",
3.   "result": {
4.     "sysmapids": [
5.       "8"
6.     ],
7.   },
8.   "id": 1
9. }
```

Create a host map

Create a map with two host elements and a link between them. Note the use of temporary “selementid1” and “selementid2” values in the map link object to refer to map elements.

Request:

```
1. {
2.   "jsonrpc": "2.0",
3.   "method": "map.create",
4.   "params": {
5.     "name": "Host map",
6.     "width": 600,
7.     "height": 600,
8.     "selements": [
9.       {
10.         "elementid": "1033",
11.         "selementid": "1",
12.         "elementtype": 0,
13.         "iconid_off": "2"
14.       },
15.       {
16.         "elementid": "1037",
17.         "selementid": "2",
18.         "elementtype": 0,
19.         "iconid_off": "2"
20.       }
21.     ],
22.     "links": [
23.       {
24.         "selementid1": "1",
```

```
map.create
```

```
25.         "selementid2": "2"
26.     }
27.   ]
28. },
29. "auth": "038e1d7b1735c6a5436ee9eae095879e",
30. "id": 1
31. }
```

Response:

```
1. {
2.   "jsonrpc": "2.0",
3.   "result": {
4.     "sysmapids": [
5.       "9"
6.     ],
7.   },
8.   "id": 1
9. }
```

Map sharing

Create a map with two types of sharing (user and user group).

Request:

```
1. {
2.   "jsonrpc": "2.0",
3.   "method": "map.create",
4.   "params": {
5.     "name": "Map sharing",
6.     "width": 600,
7.     "height": 600,
8.     "users": [
9.       {
10.         "userid": "4",
11.         "permission": "3"
12.       }
13.     ],
14.     "userGroups": [
15.       {
16.         "usrgrpid": "7",
17.         "permission": "2"
18.       }
19.     ],
20.   },
21.   "auth": "038e1d7b1735c6a5436ee9eae095879e",
22.   "id": 1
23. }
```

Response:

```
1.  {
2.      "jsonrpc": "2.0",
3.      "result": {
4.          "sysmapids": [
5.              "9"
6.          ],
7.      },
8.      "id": 1
9. }
```

See also

- [Map element](#)
- [Map link](#)
- [Map URL](#)
- [Map user](#)
- [Map user group](#)

Source

CMap::create() in *frontends/php/include/classes/api/services/CMap.php*.

map.delete

Description

```
object map.delete(array mapIds)
```

This method allows to delete maps.

Parameters

(array) IDs of the maps to delete.

Return values

(object) Returns an object containing the IDs of the deleted maps under the `sysmapids` property.

Examples

Delete multiple maps

Delete two maps.

Request:

```
1. {
2.   "jsonrpc": "2.0",
3.   "method": "map.delete",
4.   "params": [
5.     "12",
6.     "34"
7.   ],
8.   "auth": "3a57200802b24cda67c4e4010b50c065",
9.   "id": 1
10. }
```

Response:

```
1. {
2.   "jsonrpc": "2.0",
3.   "result": {
4.     "sysmapids": [
5.       "12",
6.       "34"
7.     ]
8.   },
9.   "id": 1
}
```

```
map.delete
```

```
10. }
```

Source

CMap::delete() in *frontends/php/include/classes/api/services/CMap.php*.

map.get

Description

```
integer/array map.get(object parameters)
```

The method allows to retrieve maps according to the given parameters.

Parameters

(object) Parameters defining the desired output.

The method supports the following parameters.

Parameter	Type	Description
sysmapids	string/array	Return only maps with the given IDs.
userids	string/array	Return only maps that belong to the given user IDs.
expandUrls	flag	Adds global map URLs to the corresponding map elements and expands macros in all map element URLs.
selectIconMap	query	Returns the icon map used on the map in the <code>iconmap</code> property.
selectLinks	query	Returns map links between elements in the <code>links</code> property.
selectSelements	query	Returns the map elements from the map in the <code>selements</code> property.
selectUrls	query	Returns the map URLs in the <code>urls</code> property.
selectUsers	query	Returns users that the map is shared with in <code>users</code> property.
selectUserGroups	query	Returns user groups that the map is shared with in <code>userGroups</code> property.
sortfield	string/array	Sort the result by the given properties. Possible values are: <code>name</code> , <code>width</code> and <code>height</code> .
countOutput	flag	These parameters being common for all <code>get</code> methods are described in detail in the reference commentary .
editable	boolean	
excludeSearch	flag	
filter	object	
limit	integer	
output	query	

preservekeys	flag
search	object
searchByAny	boolean
searchWildcardsEnabled	boolean
sortorder	string/array
startSearch	flag

Return values

(integer/array) Returns either:

- an array of objects;
- the count of retrieved objects, if the `countOutput` parameter has been used.

Examples

Retrieve a map

Retrieve all data about map "3".

Request:

```

1. {
2.   "jsonrpc": "2.0",
3.   "method": "map.get",
4.   "params": {
5.     "output": "extend",
6.     "selectSelements": "extend",
7.     "selectLinks": "extend",
8.     "selectUsers": "extend",
9.     "selectUserGroups": "extend",
10.    "sysmapids": "3"
11.  },
12.  "auth": "038e1d7b1735c6a5436ee9eae095879e",
13.  "id": 1
14. }
```

Response:

```

1. {
2.   "jsonrpc": "2.0",
3.   "result": [
4.     {
5.       "selements": [
6.         {
7.           "selementid": "10",
8.           "sysmapid": "3",
9.           "name": "Element 10"
10.          }
11.        ]
12.      }
13.    ]
14. }
```

```
9.          "elementid": "0",
10.         "elementtype": "4",
11.         "iconid_off": "1",
12.         "iconid_on": "0",
13.         "label": "Zabbix server",
14.         "label_location": "3",
15.         "x": "11",
16.         "y": "141",
17.         "iconid_disabled": "0",
18.         "iconid_maintenance": "0",
19.         "elements subtype": "0",
20.         "areatype": "0",
21.         "width": "200",
22.         "height": "200",
23.         "viewtype": "0",
24.         "use_iconmap": "1",
25.         "application": "",
26.         "urls": []
27.     },
28.     {
29.         "selementid": "11",
30.         "sysmapid": "3",
31.         "elementid": "0",
32.         "elementtype": "4",
33.         "iconid_off": "1",
34.         "iconid_on": "0",
35.         "label": "Web server",
36.         "label_location": "3",
37.         "x": "211",
38.         "y": "191",
39.         "iconid_disabled": "0",
40.         "iconid_maintenance": "0",
41.         "elements subtype": "0",
42.         "areatype": "0",
43.         "width": "200",
44.         "height": "200",
45.         "viewtype": "0",
46.         "use_iconmap": "1",
47.         "application": "",
48.         "urls": []
49.     }
50. ],
51. "links": [
52.     {
53.         "linkid": "23",
54.         "sysmapid": "3",
55.         "selementid1": "10",
56.         "selementid2": "11",
57.         "drawtype": "0",
58.         "color": "00CC00",
59.         "label": "",
60.         "linktriggers": []
61.     }
]
```

map.get

```
62.         ],
63.         "users": [
64.             {
65.                 "sysmapuserid": "1",
66.                 "userid": "2",
67.                 "permission": "2"
68.             }
69.         ],
70.         "userGroups": [
71.             {
72.                 "sysmapusrggrpid": "1",
73.                 "usrggrpid": "7",
74.                 "permission": "2"
75.             }
76.         ],
77.         "sysmapid": "3",
78.         "name": "Local network",
79.         "width": "400",
80.         "height": "400",
81.         "backgroundid": "0",
82.         "label_type": "2",
83.         "label_location": "3",
84.         "highlight": "1",
85.         "expandproblem": "1",
86.         "markelements": "0",
87.         "show_unack": "0",
88.         "grid_size": "50",
89.         "grid_show": "1",
90.         "grid_align": "1",
91.         "label_format": "0",
92.         "label_type_host": "2",
93.         "label_type_hostgroup": "2",
94.         "label_type_trigger": "2",
95.         "label_type_map": "2",
96.         "label_type_image": "2",
97.         "label_string_host": "",
98.         "label_string_hostgroup": "",
99.         "label_string_trigger": "",
100.        "label_string_map": "",
101.        "label_string_image": "",
102.        "iconmapid": "0",
103.        "expand_macros": "0",
104.        "severity_min": "0",
105.        "userid": "1",
106.        "private": "1"
107.    }
108. ],
109. "id": 1
110. }
```

See also

map.get

- [Icon map](#)
- [Map element](#)
- [Map link](#)
- [Map URL](#)
- [Map user](#)
- [Map user group](#)

Source

CMap::get() in *frontends/php/include/classes/api/services/CMap.php*.

map.update

Description

```
object map.update(object/array maps)
```

This method allows to update existing maps.

Parameters

(object/array) Map properties to be updated.

The `mapid` property must be defined for each map, all other properties are optional. Only the passed properties will be updated, all others will remain unchanged.

Additionally to the [standard map properties](#), the method accepts the following parameters.

Parameter	Type	Description
links	array	Map links to replace the existing links.
selements	array	Map elements to replace the existing elements.
urls	array	Map URLs to replace the existing URLs.
users	array	Map user shares to replace the existing elements.
userGroups	array	Map user group shares to replace the existing elements.

To create map links between new map elements you'll need to set an elements `selementid` to an arbitrary value and then use this value to reference this element in the links `selementid1` or `selementid2` properties. When the element is created, this value will be replaced with the correct ID generated by Zabbix. See [example for map.create](#).

Return values

(object) Returns an object containing the IDs of the updated maps under the `sysmapids` property.

Examples

Resize a map

Change the size of the map to 1200x1200 pixels.

Request:

```
1. {
```

map.update

```
2.     "jsonrpc": "2.0",
3.     "method": "map.update",
4.     "params": {
5.         "sysmapid": "8",
6.         "width": 1200,
7.         "height": 1200
8.     },
9.     "auth": "038e1d7b1735c6a5436ee9eae095879e",
10.    "id": 1
11. }
```

Response:

```
1. {
2.     "jsonrpc": "2.0",
3.     "result": {
4.         "sysmapids": [
5.             "8"
6.         ],
7.     },
8.     "id": 1
9. }
```

Change map owner

Available only for admins and super admins.

Request:

```
1. {
2.     "jsonrpc": "2.0",
3.     "method": "map.update",
4.     "params": {
5.         "sysmapid": "9",
6.         "userid": "1"
7.     },
8.     "auth": "038e1d7b1735c6a5436ee9eae095879e",
9.     "id": 2
10. }
```

Response:

```
1. {
2.     "jsonrpc": "2.0",
3.     "result": {
4.         "sysmapids": [
5.             "9"
6.         ],
7.     },
8.     "id": 2
```

map.update

```
9. }
```

See also

- [Map element](#)
- [Map link](#)
- [Map URL](#)
- [Map user](#)
- [Map user group](#)

Source

`CMap::update()` in *frontends/php/include/classes/api/services/CMap.php*.

Media type媒体类型

This class is designed to work with media types. 此项用作与媒体类型协作

Object references:对象引用:

- [Media type](#)

Available methods:可用的方法:

- [mediatype.create](#) - 创建新的媒体类型
- [mediatype.delete](#) - deleting media types删除媒体类型
- [mediatype.get](#) - retrieving media types检索媒体类型
- [mediatype.update](#) - updating media types更新媒体类型

> Media type object 媒体类型对象

The following objects are directly related to the [mediatype API](#). 以下对象与“mediatype”API直接相关。

Media type 媒体类型

The media type object has the following properties. 媒体类型对象具有以下属性。

属性	类型	描述
mediatypeid	string	(readonly) ID of the media type. 媒体类型的ID。
description(required)	string	Name of the media type. 媒体类型的名称
type(required)	integer	Transport used by the media type. 媒体类型使用的传输。Possible values: 可能的值: 0 - email; 邮件1 - script; 脚本 2 - SMS; 3 - Jabber; 100 - Ez Texting.
execpath	string	For script media types <code>exec_path</code> contains the name of the executed script. 对于脚本媒体类型“exec_path”包含已执行脚本的名称。For Ez Texting <code>exec_path</code> contains the message text limit. 对于Ez Texting, “exec_path”包含消息文本限制。Possible text limit values: 可能的文本限制值: 0 - USA (160 characters160个字符); 1 - Canada (136 characters136个字符). Required for script and Ez Texting media types. 需要脚本和Ez Texting媒体类型
gsm_modem	string	Serial device name of the GSM modem. GSM调制解调器的串行设备名称。Required for SMS media types. 需要SMS媒体类型。
passwd	string	Authentication password. 认证密码Required for Jabber and Ez Texting media types. 需要Jabber和Ez Texting媒体类型
smtp_email	string	Email address from which notifications will be sent. 发送通知的电子邮件地址。Required for email media types. 需要电子邮件媒体类型。
smtp_helo	string	SMTP HELO. Required for email media types. 需要电子邮件媒体类型。
smtp_server	string	SMTP server. Required for email media types. 需要电子邮件媒体类型。
status	integer	Whether the media type is enabled. 媒体类型是否启用 Possible values: 可能的值: 0 - (default) _enabled; (默认)启用1 - disabled. 禁用
username	string	Username or Jabber identifier. 用户名或Jabber标识符。Required for Jabber and Ez Texting media types. 需要Jabber和Ez Texting媒体类型。
exec_params	string	Script parameters. 脚本参数。Each parameter ends with a new line feed. 每个参数以新换行结束。

mediatype.create

Description描述

```
object mediatype.create(object/array mediaTypes)
```

This method allows to create new media types.此方法允许创建新的媒体类型。

Parameters 参数

(object/array) Media types to create要创建的媒体类型。.

The method accepts media types with the standard media type properties.该方法接受具有标准媒介类型属性的媒介类型

Return values返回值

(object) Returns an object containing the IDs of the created media types under the mediatypeids property. The order of the returned IDs matches the order of the passed media types.返回包含“mediatypeids”属性下创建的媒体类型的ID的对象。返回的ID的顺序与传递的媒体类型的顺序相匹配。

Examples示例

Creating a media type创建媒体类型

Create a new e-mail media type.创建一个新的电子邮件媒体类型

Request:

```

1. {
2.   "jsonrpc": "2.0",
3.   "method": "mediatype.create",
4.   "params": {
5.     "description": "E-mail",
6.     "type": 0,
7.     "smtp_server": "[email protected]",
8.     "smtp_hello": "company.com",
9.     "smtp_email": "[email protected]"
10.    },
11.    "auth": "038e1d7b1735c6a5436ee9eae095879e",
12.    "id": 1
13. }
```

Response:

```
1.  {
2.      "jsonrpc": "2.0",
3.      "result": {
4.          "mediatypeids": [
5.              "7"
6.          ],
7.      },
8.      "id": 1
9. }
```

Source来源

CMediaType::create() in *frontends/php/include/classes/api/services/CMediaType.php*.

mediatype.delete

Description描述

```
object mediatype.delete(array mediaTypeIds)
```

This method allows to delete media types.此方法允许删除媒体类型。

Parameters参数

(array) IDs of the media types to delete. (array) 要删除的媒体类型的ID。

Return values返回值

(object) Returns an object containing the IDs of the deleted media types under the mediatypeids property. (object) 返回一个包含“mediatypeids”属性下的已删除媒体类型的ID的对象。

Examples示例

Deleting multiple media types 删除多种媒体类型

Delete two media types.删除两种媒体类型。

Request:

```
1. {
2.   "jsonrpc": "2.0",
3.   "method": "mediatype.delete",
4.   "params": [
5.     "3",
6.     "5"
7.   ],
8.   "auth": "3a57200802b24cda67c4e4010b50c065",
9.   "id": 1
10. }
```

Response:

```
1. {
2.   "jsonrpc": "2.0",
3.   "result": {
4.     "mediatypeids": [
5.       "3",
6.       "5"
7.     ],
8.     "id": 1
9.   }
}
```

mediatype.delete

```
10. }
```

Source 来源

CMediaType::delete() in *frontends/php/include/classes/api/services/CMediaType.php*.

mediatype.get

Description描述

```
integer/array mediatype.get(object parameters)
```

The method allows to retrieve media types according to the given parameters. 该方法允许根据给定的参数来检索媒体类型。

Parameters参数

(object) Parameters defining the desired output. (object) 定义所需输出的参数。

The method supports the following parameters. 该方法支持以下参数。

参数	类型	描述
mediatypeids	string/array	Return only media types with the given IDs. 只返回具有给定ID的媒体类型。
mediaids	string/array	Return only media types used by the given media. 只返回给定媒体使用的媒体类型。
userids	string/array	Return only media types used by the given users. 仅返回给定用户使用的媒体类型。
selectUsers	query	Return the users that use the media type in the <code>users</code> property. 在“users”属性中返回使用媒体类型的用户。
sortfield	string/array	Sort the result by the given properties. 按照给定的属性对结果进行排序。Possible values are: 可能的值: <code>mediatypeid</code> .
countOutput	flag	These parameters being common for all <code>get</code> methods are described in detail in the reference commentary . 这些参数对于所有的“get”方法是常见的，在参考评论中有详细描述
editable	boolean	
excludeSearch	flag	
filter	object	
limit	integer	
output	query	
preservekeys	flag	
search	object	
searchByAny	boolean	
searchWildcardsEnabled	boolean	
sortorder	string/array	
startSearch	flag	

Return values返回值

(integer/array) Returns either:返回:

- an array of objects;一组对象；
- the count of retrieved objects, if the `countOutput` parameter has been used.如果已经使用“countOutput”参数，则检索到的对象的计数。

Examples 示例

Retrieving media types检索媒体类型

Retrieve all configured media types.检索所有配置的媒体类型。

Request:

```

1. {
2.   "jsonrpc": "2.0",
3.   "method": "mediatype.get",
4.   "params": {
5.     "output": "extend"
6.   },
7.   "auth": "038e1d7b1735c6a5436ee9eae095879e",
8.   "id": 1
9. }
```

Response:

```

1. {
2.   "jsonrpc": "2.0",
3.   "result": [
4.     {
5.       "mediatypeid": "1",
6.       "type": "0",
7.       "description": "Email",
8.       "smtp_server": "mail.company.com",
9.       "smtp_helo": "company.com",
10.      "smtp_email": "[email protected]",
11.      "exec_path": "",
12.      "gsm_modem": "",
13.      "username": "",
14.      "passwd": "",
15.      "status": "0"
16.    },
17.    {
18.      "mediatypeid": "2",
19.      "type": "3",
20.      "description": "Jabber",
```

```
21.         "smtp_server": "",  
22.         "smtp_hello": "",  
23.         "smtp_email": "",  
24.         "exec_path": "",  
25.         "gsm_modem": "",  
26.         "username": "[email protected]",  
27.         "passwd": "zabbix",  
28.         "status": "0"  
29.     },  
30.     {  
31.         "mediatypeid": "3",  
32.         "type": "2",  
33.         "description": "SMS",  
34.         "smtp_server": "",  
35.         "smtp_hello": "",  
36.         "smtp_email": "",  
37.         "exec_path": "",  
38.         "gsm_modem": "/dev/ttyS0",  
39.         "username": "",  
40.         "passwd": "",  
41.         "status": "0"  
42.     }  
43. ],
44. "id": 1
45. }
```

See also 参见

- [User](#)

Source 来源

CMediaType::get() in *frontends/php/include/classes/api/services/CMediaType.php*.

mediatype.update

Description描述

```
object mediatype.update(object/array mediaTypes)
```

This method allows to update existing media types.此方法允许更新现有的媒体类型。

Parameters参数

(object/array) Media type properties to be updated.要更新的媒体类型属性。

The `mediatypeid` property must be defined for each media type, all other properties are optional. Only the passed properties will be updated, all others will remain unchanged.必须为每个媒体类型定义“mediatypeid”属性，所有其他属性都是可选的。只有通过的属性将被更新，所有其他属性将保持不变。

Return values 返回值

(object) Returns an object containing the IDs of the updated media types under the `mediatypeids` property. (object) 返回一个包含“mediatypeids”属性下更新的媒体类型的ID的对象。

Examples示例

Enabling a media type 启用媒体类型。

Enable a media type, that is, set its status to 0.启用媒体类型，即将其状态设置为0。

Request:

```

1. {
2.   "jsonrpc": "2.0",
3.   "method": "mediatype.update",
4.   "params": {
5.     "mediatypeid": "6",
6.     "status": 0
7.   },
8.   "auth": "038e1d7b1735c6a5436ee9eae095879e",
9.   "id": 1
10. }
```

Response:

```

1. {
2.   "jsonrpc": "2.0",
3.   "result": {
```

```
4.      "mediatypeids": [
5.        "6"
6.      ],
7.    },
8.    "id": 1
9. }
```

Source来源

CMediaType::update() in *frontends/php/include/classes/api/services/CMediaType.php*.

Media媒介

This class is designed to work with media. 此部分用来协同媒体使用

Object references: 对象引用:

- `Media`

Available methods: 可用的方法:

- `usermedia.get` - retrieving media 检索媒体

Methods to configure media via the `user` API:

- `user.addmedia` - creating media 创建媒体
- `user.updatemedia` - updating media 更新媒体
- `user.deletemedia` - deleting media 删除媒体

> Media object媒体对象

以下对象与 `usermedia` API直接相关.

Media媒体

媒体可以通过[user API](#)被创建，更新和删除.

The media object defines how a media type should be used for a user. It has the following properties. 媒体对象定义了如何为用户使用媒体类型。 它具有以下属性。

属性	类型	描述
<code>mediaid</code>	<code>string</code>	(readonly) ID of the media. 媒体ID
<code>active(required)</code>	<code>integer</code>	Whether the media is enabled. 媒体是否启用 Possible values: 可能的值: 0 - enabled启用; 1 - disabled禁用.
<code>mediatypeid(required)</code>	<code>string</code>	ID of the media type used by the media. 媒体使用的媒体类型的ID。
<code>period(required)</code>	<code>string</code>	Time when the notifications can be sent as a time period or user macros separated by a semicolon. 通知可以作为时间段发送的时间或用分号分隔的用户宏。
<code>sendto(required)</code>	<code>string</code>	Address, user name or other identifier of the recipient. 地址, 用户名或收件人的其他标识符。
<code>severity(required)</code>	<code>integer</code>	Trigger severities to send notifications about. 触发严重性发送通知。 Severities are stored in binary form with each bit representing the corresponding severity. For example, 12 equals 1100 in binary and means, that notifications will be sent from triggers with severities warning and average. 严重性以二进制形式存储，每个位表示相应的严重性。 例如，12以二进制等于1100，意味着通知将从具有严重性警告和平均值的触发器发送。 Refer to the trigger object page for a list of supported trigger severities. 请参阅触发器对象页面以获取支持的触发严重程度列表。
<code>userid(required)</code>	<code>string</code>	ID of the user that uses the media. 使用媒体的用户的ID。

usermedia.get

Description描述

```
integer/array usermedia.get(object parameters)
```

The method allows to retrieve media according to the given parameters. 该方法允许根据给定的参数来检索媒体。

This method is deprecated and will be removed in the future. 此方法已被弃用，将来会被删除。Please use [user.get](#) instead. 请改用[user.get](#)。

Parameters参数

(object) Parameters defining the desired output. 定义所需输出的参数。

The method supports the following parameters. 该方法支持以下参数。

参数	类型	描述
mediaids	string/array	Return only media with the given IDs. 只返回具有给定ID的媒体。
usrgrpids	string/array	Return only media that are used by users in the given user groups. 只返回给定用户组中用户使用的媒体。
userids	string/array	Return only media that are used by the given users. 只返回给定用户使用的媒体。
mediatypeids	string/array	Return only media that use the given media types. 只返回使用给定媒体类型的媒体。
sortfield	string/array	Sort the result by the given properties. 按照给定的属性对结果进行排序。Possible values are: 可能的值是: <code>mediaid</code> , <code>userid</code> and <code>mediatypeid</code> .
countOutput	flag	These parameters being common for all get methods are described in detail in the reference commentary . 这些参数对于所有的“get”方法是常见的，在参考评论中有详细描述
editable	boolean	
excludeSearch	flag	
filter	object	
limit	integer	
output	query	
preservekeys	flag	
search	object	
searchByAny	boolean	

searchWildcardsEnabled	boolean
sortorder	string/array
startSearch	flag

Return values 返回值

(integer/array) Returns either: 返回:

- an array of objects; 一组对象;
- the count of retrieved objects, if the `countOutput` parameter has been used. 如果已经使用“countOutput”参数，则检索到的对象的计数。

Examples示例

Retrieving media by user 用户检索媒体

Retrieve all media for the given user. 检索给定用户的所有媒体。

Request:

```

1. {
2.   "jsonrpc": "2.0",
3.   "method": "usermedia.get",
4.   "params": {
5.     "output": "extend",
6.     "userids": "1"
7.   },
8.   "auth": "038e1d7b1735c6a5436ee9eae095879e",
9.   "id": 1
10. }
```

Response:

```

1. {
2.   "jsonrpc": "2.0",
3.   "result": [
4.     {
5.       "mediaid": "8",
6.       "userid": "1",
7.       "mediatypeid": "3",
8.       "sendto": "+3711231233",
9.       "active": "0",
10.      "severity": "48",
11.      "period": "1-5,09:00-18:00"
12.    },
13.    {
14.      "mediaid": "9",
15.      "userid": "1",
```

```
16.         "mediatypeid": "1",
17.         "sendto": "[email protected]",
18.         "active": "0",
19.         "severity": "63",
20.         "period": "1-7,00:00-24:00"
21.     },
22. ],
23. "id": 1
24. }
```

Source来源

CUserMedia::get() in *frontends/php/include/classes/api/services/CUserMedia.php*.

Problem 异常

This class is designed to work with problems.这个类用于配合使用 problems

Object references:对象引用:

- [Problem](#)

Available methods:可用的方法:

- [problem.get](#) - retrieving problems

> Problem object

problems are created by the Zabbix server and cannot be modified via the [API](#).

The problem object has the following properties.

Property	Type	Description
eventid	string	ID of the problem event.
source	integer	Type of the problem event. Possible values: 0 - event created by a trigger; 3 - internal event.
object	integer	Type of object that is related to the problem event. Possible values for trigger events: 0 - trigger. Possible values for internal events: 0 - trigger; 4 - item; 5 - LLD rule.
objectid	string	ID of the related object.
clock	timestamp	Time when the problem event was created.
ns	integer	Nanoseconds when the problem event was created.
r_eventid	string	Recovery event ID.
r_clock	timestamp	Time when the recovery event was created.
r_ns	integer	Nanoseconds when the recovery event was created.
correlationid	string	Correlation rule ID if this event was recovered by global correlation rule.
userid	string	User ID if the problem was manually closed.

problem.get

Description

```
integer/array problem.get(object parameters)
```

The method allows to retrieve problems according to the given parameters.

Parameters

(object) Parameters defining the desired output.

The method supports the following parameters.

Parameter	Type	Description
eventids	string/array	Return only problems with the given IDs.
groupids	string/array	Return only problems created by objects that belong to the given host groups.
hostids	string/array	Return only problems created by objects that belong to the given hosts.
objectids	string/array	Return only problems created by the given objects.
applicationids	string/array	Return only problems created by objects that belong to the given applications. Applies only if object is trigger or item.
source	integer	Return only problems with the given type. Refer to the problem event object page for a list of supported event types. Default: 0 - problem created by a trigger.
object	integer	Return only problems created by objects of the given type. Refer to the problem event object page for a list of supported object types. Default: 0 - trigger.
acknowledged	boolean	<code>true</code> - return acknowledged problems only; <code>false</code> - unacknowledged only.
severities	integer/array	Return only problems with given trigger severities. Applies only if object is trigger.
tags	array of objects	Return only problems with given tags. Exact match by tag and case-insensitive search by value. Format: <code>[{"tag": "<tag>", "value": "<value>"}, ...]</code> . An empty array returns all problems.
recent	string	<code>true</code> - return PROBLEM and recently RESOLVED problems (depends on Display OK triggers for N seconds) Default: <code>false</code> - UNRESOLVED problems only

eventid_from	string	Return only problems with IDs greater or equal to the given ID.
eventid_till	string	Return only problems with IDs less or equal to the given ID.
time_from	timestamp	Return only problems that have been created after or at the given time.
time_till	timestamp	Return only problems that have been created before or at the given time.
selectAcknowledges	query	Return problem's acknowledges in the <code>acknowledges</code> property. Acknowledges are sorted in reverse chronological order. The problem acknowledgement object has the following properties: <code>acknowledgeid</code> - (string) acknowledgement's ID; <code>userid</code> - (string) ID of the user that acknowledged the event; <code>eventid</code> - (string) ID of the acknowledged event; <code>clock</code> - (timestamp) time when the event was acknowledged; <code>message</code> - (string) text of the acknowledgement message; Supports <code>count</code> .
selectTags	query	Return problem's tags. Output format: <code>[{"tag": "<tag>", "value": "<value>"}, ...]</code> .
sortfield	string/array	Sort the result by the given properties. Possible values are: <code>eventid</code> .
countOutput	flag	These parameters being common for all <code>get</code> methods are described in detail in the reference commentary page.
editable	boolean	
excludeSearch	flag	
filter	object	
limit	integer	
output	query	
preservekeys	flag	
search	object	
searchByAny	boolean	
searchWildcardsEnabled	boolean	
sortorder	string/array	
startSearch	flag	

Return values

(integer/array) Returns either:

- an array of objects;
- the count of retrieved objects, if the `countOutput` parameter has been used.

Examples

Retrieving trigger problem events

Retrieve recent events from trigger "15112."

Request:

```

1. {
2.     "jsonrpc": "2.0",
3.     "method": "problem.get",
4.     "params": {
5.         "output": "extend",
6.         "selectAcknowledges": "extend",
7.         "selectTags": "extend",
8.         "objectids": "15112",
9.         "recent": "true",
10.        "sortfield": ["eventid"],
11.        "sortorder": "DESC"
12.    },
13.    "auth": "67f45d3eb1173338e1b1647c4bdc1916",
14.    "id": 1
15. }
```

Response:

```

1. {
2.     "jsonrpc": "2.0",
3.     "result": [
4.         {
5.             "eventid": "1245463",
6.             "source": "0",
7.             "object": "0",
8.             "objectid": "15112",
9.             "clock": "1472457242",
10.            "ns": "209442442",
11.            "r_eventid": "1245468",
12.            "r_clock": "1472457285",
13.            "r_ns": "125644870",
14.            "correlationid": "0",
15.            "userid": "1",
16.            "acknowledges": [
17.                {
18.                    "acknowledgeid": "14443",
19.                    "userid": "1",
20.                    "eventid": "1245463",
21.                    "clock": "1472457281",
22.                    "message": "problem solved",
23.                    "action": "1"
24.                }
25.            ],
26.        }
27.    ]
28. }
```

```
26.         "tags": [
27.             {
28.                 "tag": "test tag",
29.                 "value": "test value"
30.             }
31.         ],
32.     },
33.     "id": 1
34. }
35. }
```

See also

- [Alert](#)
- [Item](#)
- [Host](#)
- [LLD rule](#)
- [Trigger](#)

Source

CEvent::get() in *frontends/php/include/classes/api/services/CProblem.php*.

Proxy 代理

This class is designed to work with proxies. 这个类用于配合使用. 代理。

Object references:对象引用:

- [Proxy](#)
- [Proxy interface](#)

Available methods:可用的方法:

- [proxy.create](#) - create new proxies
- [proxy.delete](#) - delete proxies
- [proxy.get](#) - retrieve proxies
- [proxy.update](#) - update proxies

> Proxy object

The following objects are directly related to the `proxy` API.

Proxy

The proxy object has the following properties.

Property	Type	Description
<code>proxyid</code>	<code>string</code>	(readonly) ID of the proxy.
<code>host(required)</code>	<code>string</code>	Name of the proxy.
<code>status(required)</code>	<code>integer</code>	Type of proxy. Possible values:5 - active proxy;6 - passive proxy.
<code>description</code>	<code>text</code>	Description of the proxy.
<code>lastaccess</code>	<code>timestamp</code>	(readonly) Time when the proxy last connected to the server.
<code>tlsconnect</code>	<code>integer</code>	Connections to host. Possible values are: 1 - (default) No encryption; 2 - PSK; 4 - certificate.
<code>tls_accept</code>	<code>integer</code>	Connections from host. Possible bitmap values are: 1 - (default) No encryption; 2 - PSK; 4 - certificate.
<code>tls_issuer</code>	<code>string</code>	Certificate issuer.
<code>tls_subject</code>	<code>string</code>	Certificate subject.
<code>tls_psk_identity</code>	<code>string</code>	PSK identity. Required if either <code>tls_connect</code> or <code>tls_accept</code> has PSK enabled.
<code>tls_psk</code>	<code>string</code>	The preshared key, at least 32 hex digits. Required if either <code>tls_connect</code> or <code>tls_accept</code> has PSK enabled.

Proxy interface

The proxy interface object defines the interface used to connect to a passive proxy. It has the following properties.

Property	Type	Description
<code>interfaceid</code>	<code>string</code>	(readonly) ID of the interface.
<code>dns(required)</code>	<code>string</code>	DNS name to connect to. Can be empty if connections are made via IP address.
<code>ip(required)</code>	<code>string</code>	IP address to connect to. Can be empty if connections are made via DNS names.
<code>port(required)</code>	<code>string</code>	Port number to connect to.
<code>useip(required)</code>	<code>integer</code>	Whether the connection should be made via IP address. Possible values are: 0 - connect using DNS name; 1 - connect using IP address.
<code>hostid</code>	<code>string</code>	(readonly) ID of the proxy the interface belongs to.

proxy.create

Description

```
object proxy.create(object/array proxies)
```

This method allows to create new proxies.

Parameters

(object/array) Proxies to create.

Additionally to the [standard proxy properties](#), the method accepts the following parameters.

Parameter	Type	Description
hosts	array	Hosts to be monitored by the proxy. If a host is already monitored by a different proxy, it will be reassigned to the current proxy. The hosts must have the <code>hostid</code> property defined.
interface	object	Host interface to be created for the passive proxy. Required for passive proxies.

Return values

(object) Returns an object containing the IDs of the created proxies under the `proxyids` property. The order of the returned IDs matches the order of the passed proxies.

Examples

Create an active proxy

Create an action proxy "Active proxy" and assign a host to be monitored by it.

Request:

```

1. {
2.   "jsonrpc": "2.0",
3.   "method": "proxy.create",
4.   "params": {
5.     "host": "Active proxy",
6.     "status": "5",
7.     "hosts": [
8.       {
9.         "hostid": "10279"
10.      }

```

```
proxy.create
```

```
11.     ],
12.   },
13.   "auth": "ab9638041ec6922cb14b07982b268f47",
14.   "id": 1
15. }
```

Response:

```
1. {
2.   "jsonrpc": "2.0",
3.   "result": {
4.     "proxyids": [
5.       "10280"
6.     ]
7.   },
8.   "id": 1
9. }
```

Create a passive proxy

Create a passive proxy “Passive proxy” and assign two hosts to be monitored by it.

Request:

```
1. {
2.   "jsonrpc": "2.0",
3.   "method": "proxy.create",
4.   "params": {
5.     "host": "Passive proxy",
6.     "status": "6",
7.     "interface": {
8.       "ip": "127.0.0.1",
9.       "dns": "",
10.      "useip": "1",
11.      "port": "10051"
12.    },
13.    "hosts": [
14.      {
15.        "hostid": "10192"
16.      },
17.      {
18.        "hostid": "10139"
19.      }
20.    ]
21.  },
22.  "auth": "ab9638041ec6922cb14b07982b268f47",
23.  "id": 1
24. }
```

Response:

```
1.  {
2.      "jsonrpc": "2.0",
3.      "result": {
4.          "proxyids": [
5.              "10284"
6.          ],
7.      },
8.      "id": 1
9. }
```

See also

- [Host](#)
- [Proxy interface](#)

Source

`CProxy::create()` in *frontends/php/include/classes/api/services/CProxy.php*.

proxy.delete

Description

```
object proxy.delete(array proxies)
```

This method allows to delete proxies.

Parameters

(array) IDs of proxies to delete.

Return values

(object) Returns an object containing the IDs of the deleted proxies under the proxyids property.

Examples

Delete multiple proxies

Delete two proxies.

Request:

```
1. {
2.   "jsonrpc": "2.0",
3.   "method": "proxy.delete",
4.   "params": [
5.     "10286",
6.     "10285"
7.   ],
8.   "auth": "3a57200802b24cda67c4e4010b50c065",
9.   "id": 1
10. }
```

Response:

```
1. {
2.   "jsonrpc": "2.0",
3.   "result": {
4.     "proxyids": [
5.       "10286",
6.       "10285"
7.     ]
8.   },
9.   "id": 1
}
```

proxy.delete

```
10. }
```

Source

CProxy::delete() in *frontends/php/include/classes/api/services/CProxy.php*.

proxy.get

Description

```
integer/array proxy.get(object parameters)
```

The method allows to retrieve proxies according to the given parameters.

Parameters

(object) Parameters defining the desired output.

The method supports the following parameters.

Parameter	Type	Description
proxyids	string/array	Return only proxies with the given IDs.
selectHosts	query	Return hosts monitored by the proxy in the <code>hosts</code> property.
selectInterface	query	Return the proxy interface used by a passive proxy in the <code>interface</code> property.
sortfield	string/array	Sort the result by the given properties. Possible values are: <code>hostid</code> , <code>host</code> and <code>status</code> .
countOutput	flag	These parameters being common for all <code>get</code> methods are described in detail in the reference commentary .
editable	boolean	
excludeSearch	flag	
filter	object	
limit	integer	
output	query	
preservekeys	flag	
search	object	
searchByAny	boolean	
searchWildcardsEnabled	boolean	
sortorder	string/array	
startSearch	flag	

Return values

(integer/array) Returns either:

- an array of objects;
- the count of retrieved objects, if the `countOutput` parameter has been used.

Examples

Retrieve all proxies

Retrieve all configured proxies and their interfaces.

Request:

```

1. {
2.   "jsonrpc": "2.0",
3.   "method": "proxy.get",
4.   "params": {
5.     "output": "extend",
6.     "selectInterface": "extend"
7.   },
8.   "auth": "038e1d7b1735c6a5436ee9eae095879e",
9.   "id": 1
10. }
```

Response:

```

1. {
2.   "jsonrpc": "2.0",
3.   "result": [
4.     {
5.       "interface": [],
6.       "host": "Active proxy",
7.       "status": "5",
8.       "lastaccess": "0",
9.       "proxyid": "30091",
10.      "description": "",
11.      "tls_connect": "1",
12.      "tls_accept": "1",
13.      "tls_issuer": "",
14.      "tls_subject": "",
15.      "tls_psk_identity": "",
16.      "tls_psk": ""
17.    },
18.    {
19.      "interface": {
20.        "interfaceid": "30109",
21.        "hostid": "30092",
22.        "useip": "1",
23.        "ip": "127.0.0.1",
24.        "dns": "",
25.        "port": "10051"
26.      },
27.      "host": "127.0.0.1"
28.    }
29.  ],
30.  "count": 2
31. }
```

```
27.         "host": "Passive proxy",
28.         "status": "6",
29.         "lastaccess": "0",
30.         "proxyid": "30092",
31.         "description": ""
32.     },
33. ],
34. "id": 1
35. }
```

See also

- [Host](#)
- [Proxy interface](#)

Source

CProxy::get() in *frontends/php/include/classes/api/services/CProxy.php*.

proxy.update

Description

```
object proxy.update(object/array proxies)
```

This method allows to update existing proxies.

Parameters

(object/array) Proxy properties to be updated.

The `proxyid` property must be defined for each proxy, all other properties are optional. Only the passed properties will be updated, all others will remain unchanged.

Additionally to the [standard proxy properties](#), the method accepts the following parameters.

Parameter	Type	Description
hosts	array	Hosts to be monitored by the proxy. If a host is already monitored by a different proxy, it will be reassigned to the current proxy. The hosts must have the <code>hostid</code> property defined.
interface	object	Host interface to replace the existing interface for the passive proxy.

Return values

(object) Returns an object containing the IDs of the updated proxies under the `proxyids` property.

Examples

Change hosts monitored by a proxy

Update the proxy to monitor the two given hosts.

Request:

```

1. {
2.   "jsonrpc": "2.0",
3.   "method": "proxy.update",
4.   "params": {
5.     "proxyid": "10293",
6.     "hosts": [
7.       "10294",

```

proxy.update

```
1.         "10295"
2.     ],
3. },
4.     "auth": "038e1d7b1735c6a5436ee9eae095879e",
5.     "id": 1
6. }
```

Response:

```
1. {
2.     "jsonrpc": "2.0",
3.     "result": {
4.         "proxyids": [
5.             "10293"
6.         ]
7.     },
8.     "id": 1
9. }
```

Change proxy status

Change the proxy to an active proxy and rename it to "Active proxy".

Request:

```
1. {
2.     "jsonrpc": "2.0",
3.     "method": "proxy.update",
4.     "params": {
5.         "proxyid": "10293",
6.         "host": "Active proxy",
7.         "status": "5"
8.     },
9.     "auth": "038e1d7b1735c6a5436ee9eae095879e",
10.    "id": 1
11. }
```

Response:

```
1. {
2.     "jsonrpc": "2.0",
3.     "result": {
4.         "proxyids": [
5.             "10293"
6.         ]
7.     },
8.     "id": 1
9. }
```

See also

- [Host](#)
- [Proxy interface](#)

Source

`CProxy::update()` in *frontends/php/include/classes/api/services/CProxy.php*.

Service服务

This class is designed to work with services 该类用于配合服务使用。

Object references 对象引用：

- [Service服务](#)
- [Service time服务时间](#)
- [Service dependency服务依赖](#)
- [Service alarm服务告警](#)

Available methods 可用的方法：

- `service.adddependencies` - adding dependencies between IT services 增加IT服务之间的依赖关系
- `service.addtimes` - adding service times 增加服务时间
- `service.create` - creating new IT services 创建新的IT服务
- `service.delete` - deleting IT services 删除IT服务
- `service.deletedependencies` - deleting dependencies between IT services 删除IT服务之间的依赖关系
- `service.deletetimes` - deleting service times 删除服务时间
- `service.get` - retrieving IT services 检索IT服务
- `service.getsla` - retrieving availability information about IT services 检索有关IT服务的可用性信息
- `service.update` - updating IT services 更新IT服务

> Service object服务对象

The following objects are directly related to the `service` API以下对象与 `service` API直接相关.

Service服务

The service object has the following properties服务对象具有以下属性.

Property参数	Type类型	Description说明
<code>serviceid</code>	<code>string</code>	(<code>readonly只读</code>) ID of the service服务的ID.
<code>algorithm(required必要)</code>	<code>integer</code>	Algorithm used to calculate the state of the service用于计算服务状态的算法. Possible values可能的值为: 0 - do not calculate不计算; 1 - problem问题, if at least one child has a problem至少有一个问题; 2 - problem问题, if all children have problems都有问题.
<code>name(required必要)</code>	<code>string</code>	Name of the service服务的名称.
<code>showsla(required必要)</code>	<code>integer</code>	Whether SLA should be calculated是否应计算SLA. Possible values可能的值为: 0 - do not calculate不计算; 1 - calculate计算.
<code>sortorder(required必要)</code>	<code>integer</code>	Position of the service used for sorting用于排序服务的位置.
<code>goodsla</code>	<code>float</code>	Minimum acceptable SLA value. If the SLA drops lower, the service is considered to be in problem state最低可接受的SLA值, 如果SLA降低, 则该服务被认为处于问题状态. Default: 99.9.
<code>status</code>	<code>integer</code>	(<code>readonly只读</code>) Whether the service is in OK or problem state服务是否处于OK或问题状态. If the service is in problem state, <code>status</code> is equal either to如果服务处于问题状态, 则 <code>status</code> 等于: - the priority of the linked trigger if it is set to 2, "Warning" or higher (priorities 0, "Not classified" and 1, "Information" are ignored);链接触发器的优先级设置为2, "Warning"或更高(优先级0, "Not classified"和1, "Information"可忽略) - the highest status of a child service in problem state处于问题状态的子服务的最高状态. If the service is in OK state, <code>status</code> is equal to 0如果服务处于OK状态, 则 <code>status</code> 等于0.
<code>triggerid</code>	<code>string</code>	Trigger associated with the service. Can only be set for services that don't have children. 与服务相关的触发器, 只能为无子服务设置Default: 0

Service time服务时间

The service time object defines periods, when an service is scheduled to be up or down. It has the following properties.服务时间对象定义周期, 当服务被调度为向上或向下时。它具有以下属性:

	Type类型	

	型	
timeid	string	(readonly只读) ID of the service time服务时间的ID.
serviceid(required必要)	string	ID of the service服务的ID. Cannot be updated不能被更新.
ts_from(required必要)	integer	Time when the service time comes into effect服务时间生效的时间. For onetime downtimes <code>ts_from</code> must be set as a Unix timestamp一次宕机时间 <code>ts_from</code> 必须设置为Unix时间戳, for other types对于其他类型 - as a specific time in a week, in seconds, for example, 90000 for Tue, 2:00 AM.
ts_to(required必要)	integer	Time when the service time ends. For onetime uptimes <code>ts_to</code> must be set as a Unix timestamp, for other types - as a specific time in a week, in seconds, for example, 90000 for Tue, 2:00 AM.作为一周中的特定时间, 以秒为单位, 例如, 星期二, 上午2:00的90000
type(required必要)	integer	Service time type服务时间类型. Possible values可能的值是: 0 - planned uptime, repeated every week计划运行时间, 每周重复; 1 - planned downtime, repeated every week; 计划停机, 每周重复2 - one-time downtime一次性停机.
note	string	Additional information about the service time有关服务时间的附加信息.

Service dependency服务依赖

The service dependency object represents a dependency between services. It has the following properties服务依赖对象表示服务之间的依赖关系, 它具有以下属性.

Property参数	Type类型	Description说明
linkid	string	(readonly只读) ID of the service dependency服务依赖关系的ID.
servicedownid(required必要)	string	ID of the service, that a service depends on, that is, the child service. An service can have multiple children. 服务依赖的服务ID, 即子服务。一个服务可以有多个子服务。
serviceupid(required必要)	string	ID of the service, that is dependent on a service, that is, the parent service. An service can have multiple parents forming a directed graph.服务的ID依赖服务, 即父服务, 服务可以有多个父级形成有向图
soft(required必要)	integer	Type of dependency between services服务之间的依赖关系类型. Possible values可能的值为: 0 - hard dependency硬依赖; 1 - soft dependency软依赖. An service can have only one hard-dependent parent. This attribute has no effect on status or SLA calculation and is only used to create a core service tree. Additional parents can be added as soft dependencies forming a graph.一个服务只能有一个强依赖的父服务。该属性对状态或SLA计算没有影响, 仅用于创建核心服务树。新增的父服务可以作为形成图形的软依赖添加 An service can not be deleted if it has hard-dependent children.如果服务有硬依赖子服务, 则无法删除该服务。

Service alarm服务告警

Service alrams cannot be directly created, updated or deleted via the Zabbix API不能通过Zabbix API直接创建，更新或删除服务告警。

The service alarm objects represents an service's state change. It has the following properties服务告警对象代表服务的状态变化，它具有以下属性。

Property参数	Type类型	Description说明
servicealarmid	string	ID of the service alarm服务告警的ID.
serviceid	string	ID of the service服务的ID.
clock	timestamp	Time when the service state change has happened服务状态发生变化的时间.
value	integer	Status of the service服务状态. Refer the the service status property for a list of possible values. 请参阅 service status property 以获取可能的值列表.

service.adddependencies

Description说明

```
object service.adddependencies(object/array serviceDependencies)
```

This method allows to create dependencies between services此方法允许创建服务之间的依赖关系.

Parameters参数

(object/array) Service dependencies to create创建服务依赖关系.

Each service dependency has the following parameters每个服务依赖项具有以下参数.

Parameter参数	Type类型	Description说明
serviceid	string	ID of the service that depends on a service, that is, the parent service.取决于服务的服务ID, 即父服务
dependsOnServiceid	string	ID of the service that a service depends on, that is, the child service.服务所依赖的服务ID, 即子服务
soft	string	Type of dependency依赖类型. Refer to the service dependency object page for more information on dependency types.有关依赖关系类型的更多信息, 请参阅 service dependency object page

Return values返回值

(object) Returns an object containing the IDs of the affected parent services under the `serviceids` property. (object) 返回一个包含 `serviceids` 属性下的受影响的父服务的ID的对象。

Examples范例

Creating a hard dependency创造强依赖

Make service "2" a hard-dependent child of service "3"使服务"2"成为服务"3"的强依赖的子服务.

Request请求:

```

1. {
2.   "jsonrpc": "2.0",
3.   "method": "service.adddependencies",
4.   "params": {
5.     "serviceid": "3",
6.     "dependsOnServiceid": "2",
7.     "soft": 0
8.   },
9.   "auth": "038e1d7b1735c6a5436ee9eae095879e",

```

service.adddependencies

```
10.     "id": 1
11. }
```

Response响应：

```
1. {
2.     "jsonrpc": "2.0",
3.     "result": {
4.         "serviceids": [
5.             "3"
6.         ]
7.     },
8.     "id": 1
9. }
```

See also参考

- [service.update](#)

Source源码

CService::addDependencies() in
frontends/php/include/classes/api/services/CService.php.

service.addtimes

Description说明

```
object service.addtimes(object/array serviceTimes)
```

This method allows to create new service times此方法允许创建新的服务时间.

Parameters参数

(object/array) Service times to create创建服务时间.

The method accepts service times with the standard service time properties.该方法使用standard service time properties接受服务时间.

Return values返回值

(object) Returns an object containing the IDs of the affected services under the serviceids property. (object) 返回一个包含 serviceids 属性下受影响的服务的ID的对象。

Examples范例

Adding a scheduled downtime添加预定的停机时间

Add a downtime for service "2" scheduled weekly from Monday 22:00 till Tuesday 10:00.
从星期一22:00至星期二10:00每周定期维修"2"的停机时间

Request请求:

```

1. {
2.   "jsonrpc": "2.0",
3.   "method": "service.addtimes",
4.   "params": {
5.     "serviceid": "4",
6.     "type": 1,
7.     "ts_from": 165600,
8.     "ts_to": 201600
9.   },
10.  "auth": "038e1d7b1735c6a5436ee9eae095879e",
11.  "id": 1
12. }
```

Response响应:

```

1. {
2.   "jsonrpc": "2.0",
3.   "result": {
```

```
4.     "serviceids": [
5.         "4"
6.     ],
7. },
8. "id": 1
9. }
```

See also参考

- [service.update](#)

Source源码

`CService::addTimes()` in `frontends/php/include/classes/api/services/CService.php`.

service.create

Description说明

```
object service.create(object/array services)
```

This method allows to create new services这种方法允许创建新的服务.

Parameters参数

(object/array) services to create服务创建.

Additionally to the [standard service properties](#), the method accepts the following parameters.除[standard service properties](#)之外，该方法接受以下参数。

Parameter参数	Type类型	Description说明
dependencies	array	Service dependencies服务依赖. Each service dependency has the following parameters每个服务依赖项具有以下参数: - <code>dependsOnServiceid</code> - (string) ID of a service the service depends on, that is, the child service服务所依赖的服务的ID, 即子服务. - <code>soft</code> - (integer) type of service dependency服务依赖类型; refer to the service dependency object page for more information on dependency types. 有关依赖关系类型的更多信息, 请参阅 service dependency object page
parentid	string	ID of a hard-linked parent service硬链接的父服务的ID.
times	array	Service times to be created for the service要为服务创建的服务时间.

Return values返回值

(object) Returns an object containing the IDs of the created services under the `serviceids` property. The order of the returned IDs matches the order of the passed services. (object) 返回一个包含 `serviceids` 属性下创建的服务ID的对象, 返回的ID的顺序与传递的服务的顺序相匹配。

Examples范例

Creating an service创建服务

Create an service that will be switched to problem state, if at least one child has a problem. SLA calculation will be on and the minimum acceptable SLA is 99.99%. 创建一个将被切换到问题状态的服务, 如果至少子级有问题。SLA计算将打开, 最小可接受SLA为99.99%。

Request请求:

```
1. {
```

service.create

```
2.     "jsonrpc": "2.0",
3.     "method": "service.create",
4.     "params": {
5.         "name": "Server 1",
6.         "algorithm": 1,
7.         "showsla": 1,
8.         "goodsla": 99.99,
9.         "sortorder": 1
10.    },
11.    "auth": "038e1d7b1735c6a5436ee9eae095879e",
12.    "id": 1
13. }
```

Response响应：

```
1. {
2.     "jsonrpc": "2.0",
3.     "result": {
4.         "serviceids": [
5.             "5"
6.         ],
7.     },
8.     "id": 1
9. }
```

Source源码

CService::create() in *frontends/php/include/classes/api/services/CService.php*.

service.delete

Description说明

```
object service.delete(array serviceIds)
```

This method allows to delete services此方法允许删除服务。

Services with hard-dependent child services cannot be deleted服务与强依赖的子级服务无法删除。

Parameters参数

(array) IDs of the services to delete要删除的服务的ID。

Return values返回值

(object) Returns an object containing the IDs of the deleted services under the serviceids property. (object) 返回一个包含 serviceids 属性下删除服务ID的对象

Examples范例

Deleting multiple services删除多个服务

Delete two services删除两个服务。

Request请求：

```
1. {
2.   "jsonrpc": "2.0",
3.   "method": "service.delete",
4.   "params": [
5.     "4",
6.     "5"
7.   ],
8.   "auth": "3a57200802b24cda67c4e4010b50c065",
9.   "id": 1
10. }
```

Response响应：

```
1. {
2.   "jsonrpc": "2.0",
3.   "result": {
4.     "serviceids": [
5.       "4",
6.       "5"
7.     ]
8.   }
9. }
```

service.delete

```
7.      ],
8.    },
9.    "id": 1
10. }
```

Source源码

CService::delete() in *frontends/php/include/classes/api/services/CService.php*.

service.deletedependencies

Description说明

```
object service.deletedependencies(string/array serviceIds)
```

This method allows to delete all dependencies from services此方法允许从服务中删除所有依赖关系.

Parameters参数

(string/array) IDs of the services to delete all dependencies from删除所有依赖关系的服务的ID.

Return values返回值

(object) Returns an object containing the IDs of the affected services under the serviceids property. (object) 返回一个包含 serviceids 属性下受影响的服务的ID的对象。

Examples范例

Deleting dependencies from an service从服务中删除依赖关系

Delete all dependencies from service "2"从服务"2"删除所有依赖项.

Request请求:

```
1. {
2.   "jsonrpc": "2.0",
3.   "method": "service.deletedependencies",
4.   "params": [
5.     "2"
6.   ],
7.   "auth": "3a57200802b24cda67c4e4010b50c065",
8.   "id": 1
9. }
```

Response响应:

```
1. {
2.   "jsonrpc": "2.0",
3.   "result": {
4.     "serviceids": [
5.       "2"
6.     ]
7.   },
8. }
```

```
8.      "id": 1  
9. }
```

See also参考

- [service.update](#)

Source源码

`CService::delete()` in *frontends/php/include/classes/api/services/CService.php*.

service.deletetimes

Description说明

```
object service.deletetimes(string/array serviceIds)
```

This method allows to delete all service times from services此方法允许从服务中删除所有服务时间.

Parameters参数

(string/array) IDs of the services to delete all service times from删除所有服务时间的服务的ID.

Return values返回值

(object) Returns an object containing the IDs of the affected services under the serviceids property. (object) 返回一个包含 serviceids 属性下受影响的服务ID的对象.

Examples范例

Deleting service times from an service从服务中删除服务时间

Delete all service times from service "2"从服务"2"删除所有服务时间.

Request请求:

```
1. {
2.   "jsonrpc": "2.0",
3.   "method": "service.deletetimes",
4.   "params": [
5.     "2"
6.   ],
7.   "auth": "3a57200802b24cda67c4e4010b50c065",
8.   "id": 1
9. }
```

Response响应:

```
1. {
2.   "jsonrpc": "2.0",
3.   "result": {
4.     "serviceids": [
5.       "2"
6.     ]
7.   },
8. }
```

```
8.      "id": 1  
9. }
```

See also参考

- [service.update](#)

Source源码

`CService::delete()` in *frontends/php/include/classes/api/services/CService.php*.

service.get

Description说明

```
integer/array service.get(object parameters)
```

The method allows to retrieve services according to the given parameters该方法允许根据给定的参数检索服务.

Parameters参数

(object) Parameters defining the desired output定义所需输出的参数.

The method supports the following parameters该方法支持以下参数.

Parameter参数	Type类型	Description说明
serviceids	string/array	Return only services with the given IDs只返回具有指定ID的服务.
parentids	string/array	Return only services with the given hard-dependent parent services只返回服务与给定硬依赖的父级服务.
childids	string/array	Return only services that are hard-dependent on the given child services仅返回对给定的子服务强依赖的服务.
selectParent	query	

Examples范例

Retrieving all services检索所有服务

Retrieve all data about all services and their dependencies检索有关所有服务及其依赖关系的所有数据.

Request请求:

```

1. {
2.   "jsonrpc": "2.0",
3.   "method": "service.get",
4.   "params": {
5.     "output": "extend",
6.     "selectDependencies": "extend"
7.   },
8.   "auth": "038e1d7b1735c6a5436ee9eae095879e",
9.   "id": 1
10. }
```

Response响应:

```

1.  {
2.      "jsonrpc": "2.0",
3.      "result": [
4.          {
5.              "serviceid": "2",
6.              "name": "Server 1",
7.              "status": "0",
8.              "algorithm": "1",
9.              "triggerid": "0",
10.             "showsla": "1",
11.             "goodsla": "99.9000",
12.             "sortorder": "0",
13.             "dependencies": []
14.         },
15.         {
16.             "serviceid": "3",
17.             "name": "Data center 1",
18.             "status": "0",
19.             "algorithm": "1",
20.             "triggerid": "0",
21.             "showsla": "1",
22.             "goodsla": "99.9000",
23.             "sortorder": "0",
24.             "dependencies": [
25.                 {
26.                     "linkid": "11",
27.                     "serviceupid": "3",
28.                     "servicedownid": "2",
29.                     "soft": "0",
30.                     "sortorder": "0",
31.                     "serviceid": "2"
32.                 },
33.                 {
34.                     "linkid": "10",
35.                     "serviceupid": "3",
36.                     "servicedownid": "5",
37.                     "soft": "0",
38.                     "sortorder": "1",
39.                     "serviceid": "5"
40.                 }
41.             ]
42.         },
43.         {
44.             "serviceid": "5",
45.             "name": "Server 2",
46.             "status": "0",
47.             "algorithm": "1",
48.             "triggerid": "0",
49.             "showsla": "1",
50.             "goodsla": "99.9900",
51.             "sortorder": "1",
52.             "dependencies": []
53.         }

```

service.get

```
54.     ],
55.     "id": 1
56. }
```

Source源码

CService::get() in *frontends/php/include/classes/api/services/CService.php*.

service.getsla

Description说明

```
object service.getsla(object parameters)
```

This method allows to calculate availability information about services该方法允许计算有关服务的可用性信息.

Parameters 参数

(object) Parameters containing the IDs of the services and time intervals to calculate SLA包含服务ID和计算SLA的时间间隔的参数.

Parameter 参数	Type类型	Description说明
serviceids	string/array	IDs of services to return availability information for 提供可用性信息的服务ID.
intervals	array	Time intervals to return service layer availability information about 返回服务层可用性信息的时间间隔. Each time interval must have the following parameters 每个时间间隔必须具有以下参数: - <code>from</code> - (timestamp) interval start time 间隔开始时间; - <code>to</code> - (timestamp) interval end time 间隔结束时间.

Return values返回值

(object) Returns the following availability information about each service under the corresponding service ID返回相应服务ID下的每个服务下可用性信息.

Property 参数	Type类型	Description说明
status	integer	Current status of the service 服务的当前状态. Refer to the service object page for more information on service statuses. 有关服务状态的更多信息, 请参阅 service object page
problems	array	Triggers that are currently in problem state and are linked either to the service or one of its descendants 目前处于问题状态并与服务或其后相关联的触发器.
sla	array	SLA data about each time period 每个时间段的SLA数据. Each SLA object has the following properties 每个SLA对象具有以下属性: - <code>from</code> - (timestamp) interval start time 间隔开始时间; - <code>to</code> - (timestamp) interval end time 间隔结束时间; - <code>sla</code> - (float) SLA for the given time interval SLA在给定的时间间隔; - <code>okTime</code> - (integer) time the service was in OK state, in seconds; 时间服务处于OK状态, 单位秒; - <code>problemTime</code> - (integer) time the service was in problem state, in seconds; 时间服务处于问题状态, 单位秒; - <code>downtimeTime</code> - (integer) time the service was in scheduled downtime, in seconds. 在预定宕机时间的服务时间, 单位秒

Examples范例

Retrieving availability information for a service检索服务的可用性信息

Retrieve availability information about a service during a week在一周内检索有关服务的可用性信息。.

Request请求：

```

1. {
2.     "jsonrpc": "2.0",
3.     "method": "service.getsla",
4.     "params": {
5.         "serviceids": "2",
6.         "intervals": [
7.             {
8.                 "from": 1352452201,
9.                 "to": 1353057001
10.            }
11.        ]
12.    },
13.    "auth": "038e1d7b1735c6a5436ee9eae095879e",
14.    "id": 1
15. }
```

Response响应：

```

1. {
2.     "jsonrpc": "2.0",
3.     "result": {
4.         "2": {
5.             "status": "3",
6.             "problems": {
7.                 "13904": {
8.                     "triggerid": "13904",
9.                     "expression": "{13359}=0",
10.                    "description": "Service unavailable",
11.                    "url": "",
12.                    "status": "0",
13.                    "value": "1",
14.                    "priority": "3",
15.                    "lastchange": "1352967420",
16.                    "comments": "",
17.                    "error": "",
18.                    "templateid": "0",
19.                    "type": "0",
20.                    "value_flags": "0",
21.                    "flags": "0"
22.                }
23.            },
24.        }
25.    }
26. }
```

```
24.         "sla": [
25.             {
26.                 "from": 1352452201,
27.                 "to": 1353057001,
28.                 "sla": 97.046296296296,
29.                 "okTime": 586936,
30.                 "problemTime": 17864,
31.                 "downtimeTime": 0
32.             }
33.         ]
34.     },
35.     "id": 1
36. }
```

See also参考

- [Trigger](#)

Source源码

`CService::getSla()` in *frontends/php/include/classes/api/services/CService.php*.

service.update

Description说明

```
object service.update(object/array services)
```

This method allows to update existing services此方法允许更新现有服务.

Parameters参数

(object/array) service properties to be updated服务属性要更新.

The `serviceid` property must be defined for each service, all other properties are optional. Only the passed properties will be updated, all others will remain unchanged. 必须为每个服务定义 `serviceid` 属性，所有其他属性都是可选的。只有通过的属性将被更新，所有其他属性将保持不变。

Additionally to the `standard service properties`, the method accepts the following parameters. 除`standard service properties`之外，该方法接受以下参数。

Parameter参数	Type类型	Description说明
dependencies	array	Service dependencies to replace the current service dependencies 替换当前服务依赖关系的服务依赖关系. Each service dependency has the following parameters 每个服务依赖项具有以下参数: - <code>dependsOnServiceid</code> - (string) ID of a service the service depends on, that is, the child service 服务所依赖的服务的ID, 即子服务. - <code>soft</code> - (integer) type of service dependency 服务依赖类型; refer to the <code>service dependency object page</code> for more information on dependency types. 有关依赖关系类型的更多信息，请参阅 <code>service dependency object page</code>
parentid	string	ID of a hard-linked parent service 硬链接的父服务的ID.
times	array	Service times to replace the current service times 服务时间来替换当前的服务时间.

Return values返回值

(object) Returns an object containing the IDs of the updated services under the `serviceids` property. (object) 返回一个包含 `serviceids` 属性下更新服务的ID的对象。

Examples范例

Setting the parent of an service 设置父级服务

Make service "3" the hard-linked parent of service "5". 使服务"3"链接于父级服务"5".

Request请求:

```
1. {
2.     "jsonrpc": "2.0",
3.     "method": "service.update",
4.     "params": {
5.         "serviceid": "5",
6.         "parentid": "3"
7.     },
8.     "auth": "038e1d7b1735c6a5436ee9eae095879e",
9.     "id": 1
10. }
```

Response响应：

```
1. {
2.     "jsonrpc": "2.0",
3.     "result": {
4.         "serviceids": [
5.             "5"
6.         ]
7.     },
8.     "id": 1
9. }
```

See also参考

- [service.adddependencies](#)
- [service.addtimes](#)
- [service.deletedependencies](#)
- [service.deletetimes](#)

Source源码

`CService::update()` in *frontends/php/include/classes/api/services/CService.php*.

Template screen item模板聚合图形项

This class is designed to work with template screen items 此类用于配合模板聚合图形项的使用。

Object references: 对象引用:

- [Template screen item](#)
- [模板聚合图形项](#)

Available methods: 可用的方法:

- `templatescreenitem.get` - retrieve template screen items
- `templatescreenitem.get` - 检索模板聚合图形项

> Template screen item object模板聚合图形项对象

The following objects are directly related to the `templatescreenitem` API. 以下对象是与 `templatescreenitem` API有直接关系 .

Template screen item模板聚合图形项

The template screen item object defines an element displayed on a template screen. It has the following properties. 模板聚合图形项该对象为模板上聚合图形显示的元素. 它具有以下属性.

Property参数	Type类型	Description说明
<code>screenitemid</code>	<code>string</code>	(<code>readonly</code> 只读) ID of the template screen item模板聚合图形项的ID.
<code>resourceid</code> (required必须)	<code>string</code>	ID of the object from the parent template displayed on the template screen item来自父模板的对象的ID显示在模板聚合图形项上. Depending on the type of screen item, the <code>resourceid</code> property can reference different objects根据聚合图形项的类型, <code>resourceid</code> 属性可引用不同的对象. Unused by clock and URL template screen items聚合图形项不用于clock 和URL. Note: the <code>resourceid</code> property always references an object used in the parent template object, even if the screen item itself is inherited on a host or template. 注意: 即使聚合图形项自身继承在某一主机或模板上, 其 <code>resourceid</code> 属性始终引用父模板对象中使用的对象。
<code>resourcetype</code> (required必须)	<code>integer</code>	Type of template screen item模板聚合图形项. Possible values可能的值: 0 - graph图形; 1 - simple graph简单图形; 3 - plain text纯文本; 7 - clock时钟; 11 - URL; 19 - simple graph prototype简单图形原型; 20 - graph prototype图形原型.
<code>screenid</code> (required必须)	<code>string</code>	ID of the template screen that the item belongs to模板聚合图形项所属ID.
<code>colspan</code>	<code>integer</code>	Number of columns the template screen item will span across模板聚合图形项所跨的列数. Default: 1.
<code>elements</code>	<code>integer</code>	Number of lines to display on the template screen item在模板聚合图形项上显示的行数. Default: 25.
<code>halign</code>	<code>integer</code>	Specifies how the template screen item must be aligned horizontally in the cell指定模板监控项如何在单元格中水平对齐. Possible values可能的值: 0 - (default) center中心; 1 - left左边; 2 - right右边.
<code>height</code>	<code>integer</code>	Height of the template screen item in pixels模板聚合图形项的高度(以像素为单位). Default: 200.
<code>maxcolumns</code>	<code>integer</code>	Specifies the maximum amount of columns a graph prototype or simple graph prototype screen element can have指定的图形原型或简单图形原型聚合图形元素可具有的最大列数. Default: 3.
<code>rowspan</code>	<code>integer</code>	Number of rows the template screen item will span across模板聚合图形项所跨的行数. Default: 1.
		Template screen item display option模板聚合图形项显

style	integer	示的选项. Possible values for clock screen items聚合图形项clock的可能值: 0 - (default) local time当地时间; 1 - server time server时间 ; 2 - host time主机时间. Possible values for plain text screen items聚合图形项纯文本的可能值: 0 - (default) display values as plain text显示值为纯文本; 1 - display values as HTML显示值为HTML.
url	string	URL of the webpage to be displayed in the template screen item. Used by URL template screen items.在模板聚合图形项中显示的网页的URL，由URL模板聚合图形项使用
valign	integer	Specifies how the template screen item must be aligned vertically in the cell指定模板聚合图形项如何在单元格中垂直对齐. Possible values可能的值: 0 - (default) middle中间; 1 - top顶部; 2 - bottom底部.
width	integer	Width of the template screen item in pixels模板聚合图形项的宽度 (以像素为单位) . Default: 320.
x	integer	X-coordinates of the template screen item on the screen, from left to right模板上聚合图形项在screen的X轴，从左到右. Default: 0.
y	integer	Y-coordinates of the template screen item on the screen, from top to bottom模板上聚合图形项在screen的Y轴，从上到下. Default: 0.

templatescreenitem.get

Description说明

```
integer/array templatescreenitem.get(object parameters)
```

The method allows to retrieve template screen items according to the given parameters
该方法用于根据规定的参数获取模板聚合图形项.

Parameters参数

(object) Parameters defining the desired output 定义所需输出的参数.

The method supports the following parameters 该方法提供以下参数.

Parameter参数	Type类型	Description说明
screenids	string/array	Return only template screen items that belong to the given template screens 只返回指定模板聚合图形所属ID的模板聚合图形项.
screenitemid	string/array	Return only template screen items with the given IDs 只返回指定ID的模板聚合监控项.
hostids	string/array	Returns an additional <code>real_resourceid</code> property for each template screen item, that belongs to a screen from the given hosts or templates. The <code>real_resourceid</code> property contains the ID of object displayed on the screen. 为每个模板聚合图形项返回一个额外的“ <code>real_resourceid</code> ”属性，该属性属于指定主机或模板的聚合图形。 <code>real_resourceid</code> 属性包含屏幕上显示的对象ID
sortfield	string/array	Sort the result by the given properties 按照给定的属性对结果进行排序. Possible values are 可能的值是: <code>screenitemid</code> and <code>screenid</code> .
countOutput	flag	These parameters being common for all <code>get</code> methods are described in detail in the <code>reference commentary</code> . 这些参数对于所有的 <code>get</code> 方法是常见的，在 <code>reference commentsary</code> 中有详细描述。
editable	boolean	
excludeSearch	flag	
filter	object	
limit	integer	
output	query	
preservekeys	flag	
search	object	
searchByAny	boolean	

searchWildcardsEnabled	boolean
sortorder	string/array
startSearch	flag

Return values 返回值

(integer/array) Returns either 返回两者其中任一：

- an array of objects 一组对象；
- the count of retrieved objects, if the `countOutput` parameter has been used. 如果已经使用了“countOutput”参数，则检索对象的计数。

Examples 范例

Retrieve template screen items for screen 从聚合图形上检索模板聚合图形项

Return all template screen items from template screen “15”. 从模板聚合图形“15”返回所有模板聚合图形项。

Request 请求：

```

1. {
2.   "jsonrpc": "2.0",
3.   "method": "templateScreenItem.get",
4.   "params": {
5.     "output": "extend",
6.     "screenids": "15"
7.   },
8.   "auth": "038e1d7b1735c6a5436ee9eae095879e",
9.   "id": 1
10. }
```

Response 响应：

```

1. {
2.   "jsonrpc": "2.0",
3.   "result": [
4.     {
5.       "screenitemid": "42",
6.       "screenid": "15",
7.       "resourcetype": "0",
8.       "resourceid": "454",
9.       "width": "500",
10.      "height": "200",
11.      "x": "0",
12.      "y": "0",
```

```
13.         "colspan": "1",
14.         "rowspan": "1",
15.         "elements": "0",
16.         "valign": "1",
17.         "halign": "0",
18.         "style": "0",
19.         "url": "",
20.         "max_columns": "3"
21.     },
22.     {
23.         "screenitemid": "43",
24.         "screenid": "15",
25.         "resourcetype": "0",
26.         "resourceid": "455",
27.         "width": "500",
28.         "height": "270",
29.         "x": "1",
30.         "y": "0",
31.         "colspan": "1",
32.         "rowspan": "1",
33.         "elements": "0",
34.         "valign": "1",
35.         "halign": "0",
36.         "style": "0",
37.         "url": "",
38.         "max_columns": "3"
39.     }
40. ],
41. "id": 1
42. }
```

Source来源

CTemplateScreenItem::get() in
frontends/php/include/classes/api/services/CTemplateScreenItem.php.

Template screen模板聚合图形

This class is designed to work with template screens此类用于配合模板聚合图形的使用.

Object references对象引用:

- [Template screen模板聚合图形](#)

Available methods可用的方法:

- `templatescreen.copy` - copy template screens复制模板聚合图形
- `templatescreen.create` - create new template screens创建模板聚合图形
- `templatescreen.delete` - delete template screens删除模板聚合图形
- `templatescreen.get` - retrieve template screens检索模板聚合图形
- `templatescreen.update` - update template screens更新模板聚合图形

> Template screen object模板聚合图形对象

The following objects are directly related to the [templatesscreen API](#) 以下对象是与 [templatesscreen API](#) 有直接关系.

Template screen模板聚合图形

The template screen object has the following properties模板聚合图形对象具有以下属性.

Property参数	Type类型	Description说明
screenid	string	(readonly) ID of the template screen模板聚合图形的ID.
name (required)	string	Name of the template screen模板聚合图形的名称.
templateid (required)	string	ID of the template that the screen belongs to聚合图形所属模板的ID.
hsize	integer	Width of the template screen模板聚合图形的宽度. Default: 1
vsize	integer	Height of the template screen模板聚合图形的高度. Default: 1

templatescreen.copy

Description说明

```
object templatescreen.copy(object parameters)
```

This method allows to copy template screens to the given templates该方法允许将模板聚合图形复制到指定的模板 .

Parameters参数

(object) Parameters defining the template screens to copy and the target templates定义要复制的模板聚合图形的参数和目标模板 .

Parameter参数	Type类型	Description说明
screenIds (required)	string/array	IDs of template screens to copy要复制的模板聚合图形的ID.
templateIds (required)	string/array	IDs of templates to copy the screens to将聚合图形复制到模板的ID.

Return values返回值

(boolean) Returns true if the copying was successful. (boolean) 如果复制成功，则返回 true .

Examples范例

Copy a template screen 复制模板聚合图形

Copy template screen “25” to template “30085” 将模板聚合图形“25”复制到模板“30085” .

Request请求：

```

1. {
2.   "jsonrpc": "2.0",
3.   "method": "templatescreen.copy",
4.   "params": {
5.     "screenIds": "25",
6.     "templateIds": "30085"
7.   },
8.   "auth": "038e1d7b1735c6a5436ee9eae095879e",
9.   "id": 1
10. }
```

Response响应：

```
1. {
2.     "jsonrpc": "2.0",
3.     "result": true,
4.     "id": 1
5. }
```

Source来源

CTemplateScreen::copy() in
frontends/php/include/classes/api/services/CTemplateScreen.php.

templatescreen.create

Description说明

```
object templatescreen.create(object/array templateScreens)
```

This method allows to create new template screens该方法允许创建新的模板聚合图形.

Parameters参数

(object/array) Template screens to create要创建的模板聚合图形.

Additionally to the standard template screen properties, the method accepts the following parameters.除standard template screen properties之外，该方法接受以下参数.

Parameter参数	Type类型	Description说明
screenitems	array	Template screen items to create on the screen聚合图形上要创建的模板聚合图形项.

Return values返回值

(object) Returns an object containing the IDs of the created template screens under the screenids property. The order of the returned IDs matches the order of the passed template screens. (object) 返回一个包含 screenids 属性下创建的模板聚合图形的ID的对象，返回的ID的顺序与传递的模板聚合图形的顺序相匹配。

Examples范例

Create a template screen 创建模板聚合图形

Create a template screen named "Graphs" with 2 rows and 3 columns and add a graph to the upper-left cell创建一个名为"Graphs"的模板聚合图形，带有2行和3列，并将图形添加到左上角的单元格。.

Request请求:

```

1. {
2.   "jsonrpc": "2.0",
3.   "method": "templatescreen.create",
4.   "params": {
5.     "name": "Graphs",
6.     "templateid": "10047",
7.     "hsize": 3,
8.     "vsize": 2,
9.     "screenitems": [
10.       {

```

```
11.         "resourcetype": 0,
12.         "resourceid": "410",
13.         "x": 0,
14.         "y": 0
15.     }
16. ]
17. },
18. "auth": "038e1d7b1735c6a5436ee9eae095879e",
19. "id": 1
20. }
```

Response响应：

```
1. {
2.     "jsonrpc": "2.0",
3.     "result": {
4.         "screenids": [
5.             "45"
6.         ]
7.     },
8.     "id": 1
9. }
```

See also参考

- [Template screen item](#)

Source来源

CTemplateScreen::create() in
frontends/php/include/classes/api/services/CTemplateScreen.php.

templatescreen.delete

Description说明

```
object templatescreen.delete(array templateScreenIds)
```

This method allows to delete template screens此方法允许删除模板聚合图形.

Parameters参数

(array) IDs of the template screens to delete要删除的模板聚合图形的ID.

Return values返回值

(object) Returns an object containing the IDs of the deleted template screens under the screenids property. (object) 返回一个包含 screenids 属性下删除的模板聚合图形ID的对象.

Examples范例

Delete multiple template screens删除多个模板聚合图形

Delete two template screens删除两个模板聚合图形.

Request请求:

```
1. {
2.   "jsonrpc": "2.0",
3.   "method": "templatescreen.delete",
4.   "params": [
5.     "45",
6.     "46"
7.   ],
8.   "auth": "3a57200802b24cda67c4e4010b50c065",
9.   "id": 1
10. }
```

Response响应:

```
1. {
2.   "jsonrpc": "2.0",
3.   "result": {
4.     "screenids": [
5.       "45",
6.       "46"
7.     ]
8.   },
9.   "id": 1
}
```

```
10. }
```

Source来源

CTemplateScreen::delete() in
frontends/php/include/classes/api/services/CTemplateScreen.php.

templatescreen.get

Description说明

```
integer/array templatescreen.get(object parameters)
```

The method allows to retrieve template screens according to the given parameters该方法允许根据指定的参数来检索模板聚合图形.

Parameters参数

(object) Parameters defining the desired output定义所需输出的参数.

The method supports the following parameters该方法支持以下参数.

Parameter参数	Type类型	Description说明
hostids	string/array	Return only template screens that belong to the given hosts只返回所属指定主机的模板聚合图形.
screenids	string/array	Return only template screens with the given IDs只返回具有指定ID的模板聚合图形.
screenitemids	string/array	Return only template screens that contain the given screen items只返回包含指定聚合图形项的模板聚合图形.
templateids	string/arary	Return only template screens that belong to the given templates只返回所属指定模板的模板聚合图形.
noInheritance	flag	Do not return inherited template screens不返回继承的模板聚合图形.
selectScreenItems	query	Return the screen items that are used in the template screen in the screenitems property返回 screenitems 属性中模板聚合图形中使用的聚合图形项.
sortfield	string/array	Sort the result by the given properties按指定的属性对结果分类. Possible values are 可能的值为: screenid and name .
countOutput	flag	These parameters being common for all get methods are described in detail in the reference commentary. 在reference commentary中详细描述了所有 get 方法的这些参数.
editable	boolean	
excludeSearch	flag	
filter	object	
limit	integer	
output	query	

preservekeys	flag
search	object
searchByAny	boolean
searchWildcardsEnabled	boolean
sortorder	string/array
startSearch	flag

Return values返回值

(integer/array) Returns either 返回两者其中任一：

- an array of objects 一组对象；
- the count of retrieved objects, if the `countOutput` parameter has been used 如果已经使用了 `countOutput` 参数，则检索对象的计数.

Examples范例

Retrieve screens from template 从模板检索聚合图形

Retrieve all screens from template “10001” and all of the screen items 从模板“10001”和所有聚合图形项中检索所有聚合图形.

Request 请求：

```

1. {
2.   "jsonrpc": "2.0",
3.   "method": "templatescreen.get",
4.   "params": {
5.     "output": "extend",
6.     "selectScreenItems": "extend",
7.     "templateids": "10001"
8.   },
9.   "auth": "038e1d7b1735c6a5436ee9eae095879e",
10.  "id": 1
11. }
```

Response 响应：

```

1. {
2.   "jsonrpc": "2.0",
3.   "result": [
4.     {
5.       "screenid": "3",
6.       "name": "System performance",
7.       "hsize": "2",
8.       "vsize": "2",
9.       "templateid": "10001",
```

```
10.         "screenitems": [
11.             {
12.                 "screenitemid": "20",
13.                 "screenid": "3",
14.                 "resourcetype": "0",
15.                 "resourceid": "433",
16.                 "width": "500",
17.                 "height": "120",
18.                 "x": "0",
19.                 "y": "0",
20.                 "colspan": "1",
21.                 "rowspan": "1",
22.                 "elements": "0",
23.                 "valign": "1",
24.                 "halign": "0",
25.                 "style": "0",
26.                 "url": ""
27.             },
28.             {
29.                 "screenitemid": "21",
30.                 "screenid": "3",
31.                 "resourcetype": "0",
32.                 "resourceid": "387",
33.                 "width": "500",
34.                 "height": "100",
35.                 "x": "0",
36.                 "y": "1",
37.                 "colspan": "1",
38.                 "rowspan": "1",
39.                 "elements": "0",
40.                 "valign": "1",
41.                 "halign": "0",
42.                 "style": "0",
43.                 "url": ""
44.             },
45.             {
46.                 "screenitemid": "22",
47.                 "screenid": "3",
48.                 "resourcetype": "1",
49.                 "resourceid": "10013",
50.                 "width": "500",
51.                 "height": "148",
52.                 "x": "1",
53.                 "y": "0",
54.                 "colspan": "1",
55.                 "rowspan": "1",
56.                 "elements": "0",
57.                 "valign": "1",
58.                 "halign": "0",
59.                 "style": "0",
60.                 "url": ""
61.             },
62.             {
```

```
63.         "screenitemid": "23",
64.         "screenid": "3",
65.         "resourcetype": "1",
66.         "resourceid": "22181",
67.         "width": "500",
68.         "height": "184",
69.         "x": "1",
70.         "y": "1",
71.         "colspan": "1",
72.         "rowspan": "1",
73.         "elements": "0",
74.         "valign": "1",
75.         "halign": "0",
76.         "style": "0",
77.         "url": ""
78.     },
79. ],
80. },
81. ],
82. "id": 1
83. }
```

See also参考

- [Template screen item](#)

Source来源

`CTemplateScreen::get() in
frontends/php/include/classes/api/services/CTemplateScreen.php.`

templatescreen.update

Description说明

```
object templatescreen.update(object/array templateScreens)
```

This method allows to update existing template screens此方法允许更新现有的模板聚合图形.

Parameters参数

(object/array) Template screen properties to be updated要更新的模板聚合图形属性.

The `screenid` property must be defined for each template screen, all other properties are optional. Only the passed properties will be updated, all others will remain unchanged必须为每个模板屏幕定义 `screenid` 属性，所有其他属性都是可选的。 只有通过的属性将被更新，所有其他属性将保持不变.

Additionally to the `standard template screen properties`, the method accepts the following parameters.除`standard template screen properties`之外，该方法接受以下参数.

Parameter参数	Type类型	Description说明
screenitems	array	Screen items to replace existing screen items聚合图形项替换现有的聚合图形项. Screen items are updated by coordinates, so each screen item must have the <code>x</code> and <code>y</code> properties defined聚合图形项通过坐标轴更新，因此每个聚合图形项必须定义 <code>x</code> 和 <code>y</code> 属性.

Return values返回值

(object) Returns an object containing the IDs of the updated template screens under the `screenids` property. (object) 返回一个包含 `screenids` 属性下更新的模板聚合图形的ID的对象

Examples范例

Rename a template screen重命名模板聚合图形

Rename the template screen to “Performance graphs”将模板聚合图形重命名为“Performance graphs”.

Request请求:

```

1. {
2.   "jsonrpc": "2.0",
3.   "method": "templatescreen.update",
4.   "params": {
5.     "screenid": "3",
6.     "name": "Performance graphs"

```

templateScreen.update

```
7.     },
8.     "auth": "038e1d7b1735c6a5436ee9eae095879e",
9.     "id": 1
10. }
```

Response响应：

```
1. {
2.   "jsonrpc": "2.0",
3.   "result": {
4.     "screenids": [
5.       "3"
6.     ],
7.   },
8.   "id": 1
9. }
```

Source来源

CTemplateScreen::update() in
frontends/php/include/classes/api/services/CTemplateScreen.php.

Trigger prototype (触发器原型)

该类用于管理触发器原型。

对象引用：

- [Trigger prototype](#)

相关方法：

- [triggerprototype.create](#) - 创建新的触发器原型
- [triggerprototype.delete](#) - 删除触发器原型
- [triggerprototype.get](#) - 获取触发器原型
- [triggerprototype.update](#) - 更新触发器原型

> Trigger prototype object (触发器原型对象)

以下是 `triggerprototype` API 的使用方法。

Trigger (触发器)

触发器原型对象包含以下属性。

属性	类型	说明
<code>triggerid</code>	<code>string</code>	(只读) 触发器原型的ID。
<code>description</code> (必须)	<code>string</code>	触发器原型的名字。
<code>expression</code> (必须)	<code>string</code>	减少触发器表达式。
<code>comments</code>	<code>string</code>	触发器原型的附加注释。
<code>priority</code>	<code>integer</code>	触发器原型的危险等级。 可能的值为: 0 - (默认) not classified; 1 - information; 2 - warning; 3 - average; 4 - high; 5 - disaster.
<code>status</code>	<code>integer</code>	触发器原型开启还是关闭。 可能的值为: 0 - (默认) enabled; 1 - disabled.
<code>templateid</code>	<code>string</code>	(只读) 触发器原型父模板的ID。
<code>type</code>	<code>integer</code>	触发器原型能否生成多异常事件。 可能的值为: 0 - (默认) 不生成多事件; 1 - 生成多事件。
<code>url</code>	<code>string</code>	关联到触发器原型的 URL。
<code>recovery_mode</code>	<code>integer</code>	正常事件生成模式。 可能的值为: 0 - (默认) Expression; 1 - Recovery expression; 2 - None.
<code>recovery_expression</code>	<code>string</code>	减少触发器的恢复表达式。
<code>correlation_mode</code>	<code>integer</code>	关闭正常事件。 可能的值为: 0 - (默认) 所有异常; 1 - 匹配标签值的所有异常。
<code>correlation_tag</code>	<code>string</code>	匹配的标签。
<code>manual_close</code>	<code>integer</code>	允许手动关闭。 可能的值为: 0 - (默认) No; 1 - Yes.

triggerprototype.create

说明

```
object triggerprototype.create(object/array triggerPrototypes)
```

该方法允许创建新的触发器原型。

参数

(object/array) 需要去创建的触发器原型。

除 standard trigger prototype properties 之外，此方法还接受以下参数。

参数	类型	说明
dependencies	array	触发器原型依赖的触发器和触发器原型。 触发器必须定义 triggerid 属性。
tags	array	触发器原型标签。

触发器表达式必须以扩展形式给定，并且必须包含至少一个项目原型。

返回值

(object) 返回一个包含 triggerids 属性的触发器原型 ID 的对象。返回 ID 的顺序与传递的触发器原型的顺序想匹配。

范例

创建一个触发器原型

创建一个触发器原型来检测一个文件系统剩余空间是否小于20%。

请求：

```

1. {
2.   "jsonrpc": "2.0",
3.   "method": "triggerprototype.create",
4.   "params": {
5.     "description": "Free disk space is less than 20% on volume #{FSNAME}",
6.     "expression": "{Zabbix server:vfs.fs.size[#{FSNAME},pfree].last()}<20",
7.     "tags": [
8.       {
9.         "tag": "volume",
10.        "value": " #{FSNAME}"
11.      },
12.      {
13.        "tag": "type",
14.        "value": " #{FSTYPE}"

```

```
15.         }
16.     ]
17.   },
18.   "auth": "038e1d7b1735c6a5436ee9eae095879e",
19.   "id": 1
20. }
```

响应：

```
1. {
2.   "jsonrpc": "2.0",
3.   "result": {
4.     "triggerids": [
5.       "17372"
6.     ]
7.   },
8.   "id": 1
9. }
```

来源

`CTriggerPrototype::create()` in
`frontends/php/include/classes/api/services/CTriggerPrototype.php`.

triggerprototype.delete

说明

```
object triggerprototype.delete(array triggerPrototypeIds)
```

该方法用于删除触发器原型。

参数

(array) 需要去删除的触发器原型 ID。

返回值

(object) 返回一个 triggerids 属性下的要删除的触发器原型 ID 的对象。

范例

删除多个触发器原型

删除两个触发器原型。

请求：

```
1. {
2.   "jsonrpc": "2.0",
3.   "method": "triggerprototype.delete",
4.   "params": [
5.     "12002",
6.     "12003"
7.   ],
8.   "auth": "3a57200802b24cda67c4e4010b50c065",
9.   "id": 1
10. }
```

响应：

```
1. {
2.   "jsonrpc": "2.0",
3.   "result": {
4.     "triggerids": [
5.       "12002",
6.       "12003"
7.     ]
8.   },
9.   "id": 1
10. }
```

来源

```
CTriggerPrototype::delete() in  
frontends/php/include/classes/api/services/CTriggerPrototype.php.
```

triggerprototype.get

说明

```
integer/array triggerprototype.get(object parameters)
```

该方法用于根据规定的参数获取触发器原型。

参数

(object) 定义所需输出的参数。

该方法提供以下参数。

参数	类型	说明
active	flag	Return only enabled trigger prototypes that belong to monitored hosts. 只返回所属被监控主机的已启用触发器原型。
applicationids	string/array	Return only trigger prototypes that contain items from the given applications. 只返回指定应用集包含监控项所属的触发器原型。
discoveryids	string/array	Return only trigger prototypes that belong to the given LLD rules. 只返回所属被给定 LLD 规则的触发器原型。
functions	string/array	Return only triggers that use the given functions. 只返回指定函数所属的触发器。Refer to the supported trigger functions page for a list of supported functions. 有关支持的功能列表，请参阅 supported trigger functions 页面。
group	string	Return only trigger prototypes that belong to hosts from the host groups with the given name. 只返回指定名称的主机组中属于主机的触发器原型。
groupids	string/array	Return only trigger prototypes that belong to hosts from the given host groups. 只返回指定主机组所属主机的触发器原型。
host	string	Return only trigger prototypes that belong to hosts with the given name. 只返回指定名称的主机的触发器原型。
hostids	string/array	Return only trigger prototypes that belong to the given hosts. 只返回指定主机所属的触发器原型。
inherited	boolean	If set to <code>true</code> return only trigger prototypes inherited from a template. 如果设置为 <code>true</code> ，则只返回从模板继承的触发器原型。
maintenance	boolean	If set to <code>true</code> return only enabled trigger prototypes that belong to hosts in maintenance. 如果设置为 <code>true</code> ，则只返回所属维护中主机的已启用触发器原型。
		Return only trigger prototypes with

<code>minseverity</code>	integer	severity greater or equal than the given severity.只返回严重级别大于或等于给定严重性的触发器原型。
<code>monitored</code>	flag	Return only enabled trigger prototypes that belong to monitored hosts and contain only enabled items.只返回所属被监控主机启用触发器原型，且包含已启用的监控项。
<code>templated</code>	boolean	If set to <code>true</code> return only trigger prototypes that belong to templates.如果设置为 <code>true</code> ，则只返回所属模板的触发器原型。
<code>templateids</code>	string/array	Return only trigger prototypes that belong to the given templates.只返回指定模板的触发器原型。
<code>triggerids</code>	string/array	Return only trigger prototypes with the given IDs.只返回指定 ID 的触发器原型。
<code>expandExpression</code>	flag	Expand functions and macros in the trigger expression.扩展触发器表达式中的函数和宏。
<code>selectDiscoveryRule</code>	query	Return the LLD rule that the trigger prototype belongs to.返回所属触发器原型的 LLD 规格。
<code>selectFunctions</code>	query	Return functions used in the trigger prototype in the <code>functions</code> property.返回作用于包含 <code>functions</code> 属性的触发器原型的函数。The function objects represents the functions used in the trigger expression and has the following properties: 函数对象表示触发器表达式中使用的函数，并具有以下属性： <code>functionid</code> - (string) ID of the function; 函数的ID; <code>itemid</code> - (string) ID of the item used in the function; 函数中使用的监控项的ID; <code>function</code> - (string) name of the function; 函数名称; <code>parameter</code> - (string) parameter passed to the function. 参数传递给函数。
<code>selectGroups</code>	query	Return the host groups that the trigger prototype belongs to in the <code>groups</code> property.在 <code>groups</code> 属性中返回触发器原型所属的主机组。
<code>selectHosts</code>	query	Return the hosts that the trigger prototype belongs to in the <code>hosts</code> property.在 <code>hosts</code> 属性中返回触发器原型所属的主机。
<code>selectItems</code>	query	Return items and item prototypes used the trigger prototype in the <code>items</code> property.在 <code>items</code> 属性中返回触发器原型所包含的监控项和项目原型。
<code>selectDependencies</code>	query	Return trigger prototypes and triggers that the trigger prototype depends on in the <code>dependencies</code> property.返回触发器原型依赖于 <code>dependencies</code> 属性的触发器和触发器原型。
<code>selectTags</code>	query	Return the trigger prototype tags in <code>tags</code> property.在 <code>tags</code> 属性中返回触发器原型标签。
		Return only those results that exactly match the given filter.只返回与给定过滤器完全匹配的结果。Accepts an array, where the keys

filter	object	are property names, and the values are either a single value or an array of values to match against. 接受一个数组，其中 keys 是属性名称，并且值是单个值或要匹配值的数组。 Supports additional filters: 支持新增的过滤器： <code>host</code> - 技术性名称 of the host that the trigger prototype belongs to; <code>host</code> - 触发器原型所属主机的技术性名称； <code>hostid</code> - ID of the host that the trigger prototype belongs to. <code>hostid</code> - 触发器原型所属主机的 ID。
limitSelects	integer	Limits the number of records returned by subselects. 限制子选择返回的记录数。 Applies to the following subselects: 适用于以下子选择： <code>selectHosts</code> - results will be sorted by <code>host</code> . <code>selectHosts</code> - 结果将按 <code>host</code> 分类。
sortfield	string/array	使用规定的属性将结果分类。 可能的值： <code>triggerid</code> , <code>description</code> , <code>status</code> 和 <code>priority</code> 。
countOutput	flag	在 reference commentary 中详细描述了所有 <code>get</code> 方法的相关参数。
editable	boolean	
excludeSearch	flag	
limit	integer	
output	query	
preservekeys	flag	
search	object	
searchByAny	boolean	
searchWildcardsEnabled	boolean	
sortorder	string/array	
startSearch	flag	

Return values

(integer/array) 返回两者其中任一：

- an array of objects; 一组对象
- 如果已经使用了 `countOutput` 参数，则检索对象的计数。

例如

从一条 LLD 规则检索触发器原型

从一条 LLD 规则检索所有的触发器原型和函数。

请求：

```
1. {
```

```
triggerprototype.get
```

```
2.     "jsonrpc": "2.0",
3.     "method": "triggerprototype.get",
4.     "params": {
5.         "output": "extend",
6.         "selectFunctions": "extend",
7.         "discoveryids": "22450"
8.     },
9.     "auth": "038e1d7b1735c6a5436ee9eae095879e",
10.    "id": 1
11. }
```

响应：

```
1. {
2.     "jsonrpc": "2.0",
3.     "result": [
4.         {
5.             "functions": [
6.                 {
7.                     "functionid": "12598",
8.                     "itemid": "22454",
9.                     "function": "last",
10.                    "parameter": "0"
11.                }
12.            ],
13.            "triggerid": "13272",
14.            "expression": "{12598}<20",
15.            "description": "Free inodes is less than 20% on volume {#FSNAME}",
16.            "url": "",
17.            "status": "0",
18.            "priority": "2",
19.            "comments": "",
20.            "templateid": "0",
21.            "type": "0",
22.            "flags": "2",
23.            "recovery_mode": "0",
24.            "recovery_expression": "",
25.            "correlation_mode": "0",
26.            "correlation_tag": "",
27.            "manual_close": "0"
28.        },
29.        {
30.            "functions": [
31.                {
32.                    "functionid": "13500",
33.                    "itemid": "22686",
34.                    "function": "last",
35.                    "parameter": "0"
36.                }
37.            ],
38.            "triggerid": "13266",
39.            "expression": "{13500}<201",
```

```

40.         "description": "Free disk space is less than 20% on volume {#FSNAME}",
41.         "url": "",
42.         "status": "0",
43.         "priority": "2",
44.         "comments": "",
45.         "templateid": "0",
46.         "type": "0",
47.         "flags": "2",
48.         "recovery_mode": "0",
49.         "recovery_expression": "",
50.         "correlation_mode": "0",
51.         "correlation_tag": "",
52.         "manual_close": "0"
53.     }
54. ],
55. "id": 1
56. }
```

根据标签检索一个特定的触发器原型

请求：

```

1. {
2.   "jsonrpc": "2.0",
3.   "method": "triggerprototype.get",
4.   "params": {
5.     "output": [
6.       "triggerid",
7.       "description"
8.     ]
9.     "selectTags": "extend",
10.    "triggerids": [
11.      "17373"
12.    ]
13.  },
14.  "auth": "038e1d7b1735c6a5436ee9eae095879e",
15.  "id": 1
16. }
```

响应：

```

1. {
2.   "jsonrpc": "2.0",
3.   "result": [
4.     {
5.       "triggerid": "17373",
6.       "description": "Free disk space is less than 20% on volume {#FSNAME}",
7.       "tags": [
8.         {
9.           "tag": "volume",
10.          "value": "{#FSNAME}"}
```

```
11.          },
12.          {
13.              "tag": "type",
14.              "value": "{#FSTYPE}"
15.          }
16.      ]
17.  }
18. ],
19. "id": 1
20. }
```

参见

- [Discovery rule](#)
- [Item](#)
- [Host](#)
- [Host group](#)

来源

`CTriggerPrototype::get()` in
frontends/php/include/classes/api/services/CTriggerPrototype.php.

triggerprototype.update

说明

```
object triggerprototype.update(object/array triggerPrototypes)
```

此方法用于更新目前的触发器原型。

参数

(object/array) 需要被更新的触发器原型属性Trigger prototype properties。

triggerid 属性必须在每个触发器原型中定义，其他所有属性为可选项。只有传递过去的属性会被更新，其他所有属性仍然保持不变。

除 standard trigger prototype properties 之外，该方法接受以下参数。

参数	类型	说明
dependencies	array	触发器原型依赖的触发器和触发器原型。 触发器必须定义 triggerid 属性。
tags	array	触发器原型标签。

触发器表达式必须以扩展形式给定，并且必须包含至少一个项目原型。

返回值

(object) 返回一个 triggerids 属性下已更新触发器原型的ID的对象。

范例

启用触发器原型

启用触发器，即将其状态设置为0。

请求：

```

1. {
2.   "jsonrpc": "2.0",
3.   "method": "triggerprototype.update",
4.   "params": {
5.     "triggerid": "13938",
6.     "status": 0
7.   },
8.   "auth": "038e1d7b1735c6a5436ee9eae095879e",
9.   "id": 1
10. }
```

```
triggerprototype.update
```

Response:

```
1. {
2.     "jsonrpc": "2.0",
3.     "result": {
4.         "triggerids": [
5.             "13938"
6.         ]
7.     },
8.     "id": 1
9. }
```

替换触发器原型标签

为一个触发器原型替换标签。

请求:

```
1. {
2.     "jsonrpc": "2.0",
3.     "method": "triggerprototype.update",
4.     "params": {
5.         "triggerid": "17373",
6.         "tags": [
7.             {
8.                 "tag": "volume",
9.                 "value": "{#FSNAME}"
10.            },
11.            {
12.                "tag": "type",
13.                "value": "{#FSTYPE}"
14.            }
15.        ],
16.        "auth": "038e1d7b1735c6a5436ee9eae095879e",
17.        "id": 1
18.    }
19. }
```

响应:

```
1. {
2.     "jsonrpc": "2.0",
3.     "result": {
4.         "triggerids": [
5.             "17373"
6.         ]
7.     },
8.     "id": 1
9. }
```

来源

```
CTriggerPrototype::update() in  
frontends/php/include/classes/api/services/CTriggerPrototype.php.
```

Web场景

此类用于Web场景的使用。

对象引用：

- [Web scenario](#)
- [Scenario step](#)

可用的方法：

- [http://test.create](#) - 创建新的Web场景
- [http://test.delete](#) - 删除Web场景
- [http://test.get](#) - 获取Web场景
- [http://test.update](#) - 更新Web场景

http://bookstack.cn

说明

`object http://bookstack.cn.create(object/array webScenarios)`

此方法允许创建新的Web场景。

创建Web场景将自动创建一组web监控项。

参数

`(object/array)` 要创建的Web场景。

除了 [标准Web场景属性](#)之外，该方法接受以下参数

参数	类型	说明
<code>steps(required)</code>	array	Web方案步骤。

返回值

`(object)` 返回一个包含“http://bookstack.cn.ids”属性下创建的Web场景的ID的对象。返回的ID的顺序与传递的Web方案的顺序相匹配。

示例

创建Web场景

创建一个Web场景来监视公司主页。该方案将有两个步骤，以检查主页和“关于”页面，并确保它们返回HTTP状态代码200。Request：

```

1. {
2.   "jsonrpc": "2.0",
3.   "method": "http://bookstack.cn.create",
4.   "params": {
5.     "name": "Homepage check",
6.     "hostid": "10085",
7.     "steps": [
8.       {
9.         "name": "Homepage",
10.        "url": "http://mycompany.com",
11.        "status_codes": 200,
12.        "no": 1
13.      },
14.      {
15.        "name": "Homepage / About",
16.        "url": "http://mycompany.com/about",

```

```
17.         "status_codes": 200,
18.         "no": 2
19.     }
20.   ]
21. },
22. "auth": "038e1d7b1735c6a5436ee9eae095879e",
23. "id": 1
24. }
```

Response:

```
1. {
2.   "jsonrpc": "2.0",
3.   "result": {
4.     "http://test.ids": [
5.       "5"
6.     ]
7.   },
8.   "id": 1
9. }
```

参见

- [Scenario step](#)

来源

`CHttpTest::create()` in *frontends/php/include/classes/api/services/CHttpTest.php*.

http://test.delete

说明

```
object http://test.delete(array webScenarioIds)
```

此方法允许删除Web场景。

参数

(array) 要删除的网络场景的ID。

返回值

(object) 返回包含“http://test.ids”属性下删除的Web方案的ID的对象。

示例

删除多个Web场景

删除2个Web场景

Request:

```
1. {
2.   "jsonrpc": "2.0",
3.   "method": "http://test.delete",
4.   "params": [
5.     "2",
6.     "3"
7.   ],
8.   "auth": "3a57200802b24cda67c4e4010b50c065",
9.   "id": 1
10. }
```

Response:

```
1. {
2.   "jsonrpc": "2.0",
3.   "result": {
4.     "http://test.ids": [
5.       "2",
6.       "3"
7.     ]
8.   },
9.   "id": 1
10. }
```

来源

`CHttpTest::delete() in frontends/php/include/classes/api/services/CHttpTest.php.`

httpstest.get

说明

```
integer/array httpstest.get(object parameters)
```

该方法允许根据给定的参数检索Web场景。

参数

(object) 定义所需输出的参数。

该方法支持以下参数。

参数	类型	描述
applicationids	string/array	仅返回属于给定应用程序的Web场景。
groupids	string/array	仅返回属于给定主机组的Web方案。
hostids	string/array	仅返回属于给定主机的Web场景。
httpstestids	string/array	只返回具有给定ID的Web场景。
inherited	boolean	如果设置为“true”，只返回从模板继承的Web场景。
monitored	boolean	如果设置为“true”，则只返回属于受监视主机的启用的Web场景。
templated	boolean	如果设置为“true”，则只返回属于模板的Web场景。
templateids	string/array	仅返回属于给定模板的Web场景
expandName	flag	以Web方案的名称展开宏。
expandStepName	flag	在方案步骤的名称中展开宏。
selectHosts	query	将网站场景所属的主机作为“hosts”属性中的数组返回。
selectSteps	query	在 <code>steps</code> 属性中返回Web方案步骤。
sortfield	string/array	按照给定的属性对结果进行排序。 可能的值为： <code>httpstestid</code> 和 <code>name</code> 。
countOutput	flag	这些参数对于所有的“get”方法是常见的，在 考评论 中有详细描述
editable	boolean	
excludeSearch	flag	
filter	object	
limit	integer	
output	query	
preservekeys	flag	
search	object	

searchByAny	boolean
searchWildcardsEnabled	boolean
sortorder	string/array
startSearch	flag

返回值

(integer/array) 返回:

- 一组对象；
- 如果已经使用“countOutput”参数，则检索到的对象的计数。

示例

检索网络场景

Retrieve all data about web scenario “4”.

Request:

```

1. {
2.     "jsonrpc": "2.0",
3.     "method": "http://127.0.0.1:19999/http/test.get",
4.     "params": {
5.         "output": "extend",
6.         "selectSteps": "extend",
7.         "httptestids": "9"
8.     },
9.     "auth": "038e1d7b1735c6a5436ee9eae095879e",
10.    "id": 1
11. }
```

Response:

```

1. {
2.     "jsonrpc": "2.0",
3.     "result": [
4.         {
5.             "httptestid": "9",
6.             "name": "Homepage check",
7.             "applicationid": "0",
8.             "nextcheck": "0",
9.             "delay": "1m",
10.            "status": "0",
11.            "variables": "",
12.            "agent": "Zabbix",
13.            "authentication": "0",
14.            "http_user": ""}
```

```

15.         "http_password": "",
16.         "hostid": "10084",
17.         "templateid": "0",
18.         "http_proxy": "",
19.         "retries": "1",
20.         "ssl_cert_file": "",
21.         "ssl_key_file": "",
22.         "ssl_key_password": "",
23.         "verify_peer": "0",
24.         "verify_host": "0",
25.         "headers": "",
26.         "steps": [
27.             {
28.                 "httpstepid": "36",
29.                 "httptestid": "9",
30.                 "name": "Homepage",
31.                 "no": "1",
32.                 "url": "http://mycompany.com",
33.                 "timeout": "15",
34.                 "posts": "",
35.                 "required": "",
36.                 "status_codes": "200",
37.                 "variables": "",
38.                 "follow_redirects": "1",
39.                 "retrieve_mode": "0",
40.                 "headers": ""
41.             },
42.             {
43.                 "httpstepid": "37",
44.                 "httptestid": "9",
45.                 "name": "Homepage / About",
46.                 "no": "2",
47.                 "url": "http://mycompany.com/about",
48.                 "timeout": "15",
49.                 "posts": "",
50.                 "required": "",
51.                 "status_codes": "200",
52.                 "variables": "",
53.                 "follow_redirects": "1",
54.                 "retrieve_mode": "0",
55.                 "headers": ""
56.             }
57.         ]
58.     },
59. ],
60. "id": 1
61. }

```

参见

- [Host](#)

- Scenario step

来源

`CHttpTest::get() in frontends/php/include/classes/api/services/CHttpTest.php.`

http://127.0.0.1:8089

http://127.0.0.1:8089

Description

`object http://127.0.0.1:8089.update(object/array webScenarios)` 此方法允许更新现有的Web场景。

参数

`(object/array)` 要更新的Web场景属性。

必须为每个Web场景定义“`http://127.0.0.1:8089.id`”属性，所有其他属性都是可选的。 只有通过的属性将被更新，所有其他属性将保持不变除了[标准Web场景属性](#)外，该方法接受以下参数。

参数	类型	说明
<code>steps</code>	<code>array</code>	用来替代现有的步骤的方案步骤。

返回值

`(object)` Returns an object containing the IDs of the updated web scenarios under the `http://127.0.0.1:8089.id` property.

示例

Enabling a web scenario

Enable a web scenario, that is, set its status to “0”.

Request:

```

1. {
2.   "jsonrpc": "2.0",
3.   "method": "http://127.0.0.1:8089.update",
4.   "params": {
5.     "http://127.0.0.1:8089.id": "5",
6.     "status": 0
7.   },
8.   "auth": "700ca65537074ec963db7efabda78259",
9.   "id": 1
10. }
```

Response:

```

1. {
2.   "jsonrpc": "2.0",
3.   "result": {
4.     "http://127.0.0.1:8089.ids": [
```

```
5.          "5"
6.        ]
7.      },
8.      "id": 1
9. }
```

参见

- [Scenario step](#)

来源

`CHttpTest::update()` in *frontends/php/include/classes/api/services/CHttpTest.php*.

Web场景对象

以下对象与“webcheck”API直接相关。

Web场景

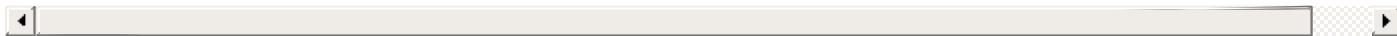
Web场景对象具有以下属性。

属性	类型	说明
httpitestid	string	(readonly) Web场景的ID
hostid(required)	string	Web场景所属主机的ID。
name(required)	string	Web场景的名称
agent	string	将由Web场景使用的用户代理字符串。默认：Zabbix
applicationid	string	Web场景所属应用程序的ID。
authentication	integer	将由Web场景使用的身份验证方法。 可能的值： 0 - (默认) 无； 1 - 基本的HTTP认证； 2 - NTLM身份验证
delay	string	Web场景的执行间隔。 接受秒，时间单位后缀和用户宏。 默认： 1m.
headers	string	执行请求时将发送的HTTP标题。
httppassword	string	用于认证的密码。 对于具有基本HTTP或NTLM身份验证的Web场景是必需的。
http_proxy	string	将由Web场景使用的代理 http://[username[:password]@]proxy.example.com[:port]
httpuser	string	用于认证的用户名 对于具有基本HTTP或NTLM身份验证的Web场景，必需。
nextcheck	timestamp	(readonly) 下一个Web场景执行的时间。
retries	integer	Web场景在失败之前尝试执行每个步骤的次数。默认：1.
ssl_cert_file	string	用于客户端身份验证的SSL证书文件的名称（必须为PEM格式）。
ssl_key_file	string	用于客户端认证的SSL私钥文件的名称（必须为PEM格式）。
ssl_key_password	string	SSL私钥密码。
status	integer	是否启用了Web方案。 可能的值： 0 - (默认) 启用； 1 - 禁用。
templateid	string	(readonly) 父模板Web方案的ID。
variables	string	Web场景变量。
verify_host	integer	验证SSL证书中指定的主机名是否与场景中使用的主机名相匹配。 可能的值： 0 - (默认) 跳过主机验证； 1 - 验证主机。
verify_peer	integer	是否验证Web服务器的SSL证书。 \可能的值： 0 - (默认) 跳过对等验证； 1 - 验证对等

场景步骤

场景步骤对象定义特定的Web场景检查。 它具有以下属性。

属性	类型	说明
httpstepid	string	(readonly) 情景步骤的ID
name(required)	string	场景步骤的名称。
no(required)	integer	Web场景中步骤的序列号。
url(required)	string	要检查的URL。
followredirects	integer	是否遵循HTTP重定向 可能的值： 0 - 不要重新导向； 1 - (default) 遵循重定向
headers	string	执行请求时将发送的HTTP标题。方案步骤标题将覆盖为Web方案指定的标题。
httptestid	string	(readonly) 该步骤所属的Web方案的ID。
posts	string	HTTP POST变量作为字符串。
required	string	必须在响应中存在的文本。
retrieve_mode	integer	方案步骤必须检索的HTTP响应的一部分。 \可能的值： 0 - (default)_ 仅有文体； 1 - 仅有标题.
status_codes	string	所需HTTP状态代码的范围用逗号分隔。
timeout	integer	在几秒钟内请求超时。 默认： 15.
variables	string	情景步骤变量。



主机

这个类用于管理主机。

对象引用：

- [Host](#)
- [Host inventory](#)

相关方法：

- [host.create](#) - 创建新的主机
- [host.delete](#) - 删除主机
- [host.get](#) - 获取主机信息
- [host.massadd](#) - 添加相关对象到主机
- [host.massremove](#) - 从主机中移除相关对象
- [host.massupdate](#) - 从主机中替换或移除相关对象
- [host.update](#) - 更新主机

> Host object 主机对象

The following objects are directly related to the host API.以下是主机API的使用方法。

Host 主机

The host object has the following properties.主机对象拥有以下属性。

属性	类型	说明
hostid	string	(readonly) ID of the host. (只读) 主机的ID。
host(required)	string	Technical name of the host. 主机的正式名称。
available	integer	(readonly) Availability of Zabbix agent. (只读) Zabbix agent的可用性。Possible values are: 可能的值为: 0 - (default) unknown; (默认)未知; 1 - available; 可用; 2 - unavailable. 失效
description	text	Description of the host. 主机说明。
disableuntil	timestamp	(readonly) The next polling time of an unavailable Zabbix agent. (只读) 对失效Zabbix agent下一次的轮询时间。
error	string	(readonly) Error text if Zabbix agent is unavailable. (只读) Zabbix agent失效的错误文本。
errors_from	timestamp	(readonly) Time when Zabbix agent became unavailable. (只读) 当Zabbix agent失效时的时间。
flags	integer	(readonly) Origin of the host. (只读) 主机的来源。Possible values: 可能的值为: 0 - a plain host; 普通主机; 4 - a discovered host. 自动发现的主机。
inventory_mode	integer	Host inventory population mode. 主机资产清单群体模式。Possible values are: 可能的值: -1 - disabled; 已禁用; 0 - (default) manual; (默认)手动; 1 - automatic. 自动。
ipmi_authtype	integer	IPMI authentication algorithm. IPMI认证算法。Possible values are: 可能的值为: -1 - (default) default; (默认)默认; 0 - none; 无; 1 - MD2; 2 - MD5 4 - straight; 直连; 5 - OEM; 原厂; 6 - RMCP+. 远程管理控制协议。
ipmi_available	integer	(readonly) Availability of IPMI agent. (只读) IPMI agent的可用性。Possible values are: 可能的值为: 0 - (default) unknown; (默认)未知; 1 - available; 可用; 2 - unavailable. 失效。
ipmi_disable_until	timestamp	(readonly) The next polling time of an unavailable IPMI agent. (只读) 对失效IPMI agent下一次的轮询时间。
ipmi_error	string	(readonly) Error text if IPMI agent is unavailable. (只读) IPMI agent失效的错误文本。
ipmi_errors_from	timestamp	(readonly) Time when IPMI agent became unavailable. (只读) 当IPMI agent失效时的时间。
ipmi_password	string	IPMI password. IPMI密码。
		IPMI privilege level. IPMI特权级别。 Possible

ipmi_privilege	integer	values are: 可能的值为: 1 - callback; 回调; 2 - (default) user; (默认) 用户; 3 - operator; 操作员; 4 - admin; 管理员; 5 - OEM. 原厂。
ipmi_username	string	IPMI username. IPMI用户名。
jmx_available	integer	(readonly) Availability of JMX agent. (只读) JMX agent的可用性。Possible values are: 可能的值为: 0 - (default) unknown; (默认) 未知; 1 - available; 可用; 2 - unavailable. 失效。
jmx_disable_until	timestamp	(readonly) The next polling time of an unavailable JMX agent. (只读) 对失效JMX agent下一次的轮询时间。
jmx_error	string	(readonly) Error text if JMX agent is unavailable. (只读) JMX失效的错误文本。
jmx_errors_from	timestamp	(readonly) Time when JMX agent became unavailable. (只读) 当JMX agent失效时的时间。
maintenance_from	timestamp	(readonly) Starting time of the effective maintenance. (只读) 开始有效管理的时间。
maintenance_status	integer	(readonly) Effective maintenance status. (只读) 有效管理的状态。Possible values are: 可能的值为: 0 - (default) no maintenance; (默认) 管理断开; 1 - maintenance in effect. 管理有效。
maintenance_type	integer	(readonly) Effective maintenance type. (只读) 有效管理的类型。Possible values are: 可能的值为: 0 - (default) maintenance with data collection; (默认) 有数据收集的管理; 1 - maintenance without data collection. 没有数据收集的管理。
maintenanceid	string	(readonly) ID of the maintenance that is currently in effect on the host. (只读) 当前对主机起效的管理ID。
name	string	Visible name of the host. 主机显示名称。Default: host property value. 默认: host属性值。
proxy_hostid	string	ID of the proxy that is used to monitor the host. 用于监控主机的代理服务器 ID。
snmp_available	integer	(readonly) Availability of SNMP agent. (只读) SNMP agent的可用性。Possible values are: 可能的值为: 0 - (default) unknown; (默认) 未知; 1 - available; 可用; 2 - unavailable. 失效。
snmp_disable_until	timestamp	(readonly) The next polling time of an unavailable SNMP agent. (只读) 对失效SNMP agent下一次的轮询时间。
snmp_error	string	(readonly) Error text if SNMP agent is unavailable. (只读) SNMP agent失效的错误文本。
snmp_errors_from	timestamp	(readonly) Time when SNMP agent became unavailable. (只读) 当SNMP agent失效时的时间。
status	integer	Status and function of the host. 主机的状态和功能。Possible values are: 可能的值为: 0 - (default) monitored host; (默认) 已监控主机; 1 - unmonitored host. 未监控主机
tls_issuer	string	Certificate issuer. 证书颁布。
tls_subject	string	Certificate subject. 证书标题。
		PSK identity. Required if either <code>tls_connect</code> or

<code>tls_psk_identity</code>	string	<code>tls_accept</code> has PSK enabled. PSK身份, 若要启用 <code>tls_connect</code> 或 <code>tls_accept</code> , 就必须启用PSK。
<code>tls_psk</code>	string	The preshared key, at least 32 hex digits. Required if either <code>tls_connect</code> or <code>tls_accept</code> has PSK enabled. PSK至少需要32位16进制数字构成。若要启用 <code>tls_connect</code> 或 <code>tls_accept</code> , 就必须启用PSK。

Host inventory 主机资产清单

The host inventory object has the following properties. 主机资产清单对象有以下属性。

Each property has it's own unique ID number, which is used to associate host inventory fields with items. 每一个属性拥有自己唯一的ID编号, 用于将主机资产清单字段和事项关联在一起。

ID	属性	类型	说明
4	<code>alias</code>	string	Alias. 别名
11	<code>asset_tag</code>	string	Asset tag. 资产编号。
28	<code>chassis</code>	string	Chassis. 机架。
23	<code>contact</code>	string	Contact person. 联系人。
32	<code>contract_number</code>	string	Contract number. 联系号码。
47	<code>date_hw_decomm</code>	string	HW decommissioning date. 硬件淘汰时间。
46	<code>date_hw_expiry</code>	string	HW maintenance expiry date. 硬件维保过期时间。
45	<code>date_hw_install</code>	string	HW installation date. 硬件安装时间。
44	<code>date_hw_purchase</code>	string	HW purchase date. 硬件购买时间。
34	<code>deployment_status</code>	string	Deployment status. 部署状态。
14	<code>hardware</code>	string	Hardware. 硬件设备。
15	<code>hardware_full</code>	string	Detailed hardware. 硬件设备详情。
39	<code>host_netmask</code>	string	Host subnet mask. 主机子网掩码。
38	<code>host_networks</code>	string	Host networks. 主机网络。
40	<code>host_router</code>	string	Host router. 主机路由。
30	<code>hw_arch</code>	string	HW architecture. 硬件架构。
33	<code>installer_name</code>	string	Installer name. 安装人员姓名。
24	<code>location</code>	string	Location. 地点。
25	<code>location_lat</code>	string	Location latitude. 纬度位置。
26	<code>location_lon</code>	string	Location longitude. 经度位置。
12	<code>macaddress_a</code>	string	MAC address A. MAC地址一。
13	<code>macaddress_b</code>	string	MAC address B. MAC地址二。
29	<code>model</code>	string	Model. 型号。
3	<code>name</code>	string	Name. 名称。
27	<code>notes</code>	string	Notes. 注释。
41	<code>oob_ip</code>	string	OOB IP address. 带外IP地址。

42	oob_netmask	string	0OB host subnet mask. 带外主机子网掩码。
43	oob_router	string	0OB router. 带外路由。
5	os	string	OS name. 操作系统名称。
6	os_full	string	Detailed OS name. 详尽操作系统名称。
7	os_short	string	Short OS name. 简短操作系统名称。
61	poc_1_cell	string	Primary POC mobile number. 主POC移动号码。
58	poc_1_email	string	Primary email. 主POC邮箱。
57	poc_1_name	string	Primary POC name. 主POC名称。
63	poc_1_notes	string	Primary POC notes. 主POC注释。
59	poc_1_phone_a	string	Primary POC phone A. 主POC电话一。
60	poc_1_phone_b	string	Primary POC phone B. 主POC电话二。
62	poc_1_screen	string	Primary POC screen name. 主POC显示名。
68	poc_2_cell	string	Secondary POC mobile number. 次POC移动号码。
65	poc_2_email	string	Secondary POC email. 次POC邮箱。
64	poc_2_name	string	Secondary POC name. 次POC名称。
70	poc_2_notes	string	Secondary POC notes. 次POC注释。
66	poc_2_phone_a	string	Secondary POC phone A. 次POC电话一。
67	poc_2_phone_b	string	Secondary POC phone B. 次POC电话二。
69	poc_2_screen	string	Secondary POC screen name. 次POC显示名。
8	serialno_a	string	Serial number A. 序列号一。
9	serialno_b	string	Serial number B. 序列号二。
48	site_address_a	string	Site address A. 所在地址一。
49	site_address_b	string	Site address B. 所在地址二。
50	site_address_c	string	Site address C. 所在地址三。
51	site_city	string	Site city. 所在城市。
53	site_country	string	Site country. 所在国家。
56	site_notes	string	Site notes. 所在地注解。
55	site_rack	string	Site rack location. 所在地机柜位置
52	site_state	string	Site state. 所在地说明。
54	site_zip	string	Site ZIP/postal code. 所在地邮政编码。
16	software	string	Software. 软件。
18	software_app_a	string	Software application A. 应用软件一。
19	software_app_b	string	Software application B. 应用软件二。
20	software_app_c	string	Software application C. 应用软件三。
21	software_app_d	string	Software application D. 应用软件四。
22	software_app_e	string	Software application E. 应用软件五。
17	software_full	string	Software details. 软件详情。

10	tag	string	Tag. 标签。
1	type	string	Type. 类型。
2	type_full	string	Type details. 具体类型。
35	url_a	string	URL A. 网址一。
36	url_b	string	URL B. 网址二。
37	url_c	string	URL C. 网址三。
31	vendor	string	Vendor. 供应商。

host.create

Description说明

```
object host.create(object/array hosts)
```

This method allows to create new hosts.此方法允许创建新的主机。

Parameters 参数

(object/array) Hosts to create.要创建的主机。

Additionally to the [standard host properties](#), the method accepts the following parameters.另外，对于标准主机属性，此方法接受以下参数。

参数	类型	说明
groups(required)	object/array	Host groups to add the host to.将主机添加到主机组中。The host groups must have the <code>groupid</code> property defined.主机组必须已定义groupid属性。
interfaces(required)	object/array	Interfaces to be created for the host.为主机创建的接口。
templates	object/array	Templates to be linked to the host.链接到主机的模板。The templates must have the <code>templateid</code> property defined. 模板必须已定义过templateid属性。
macros	object/array	User macros to be created for the host.为主机创建的用户宏。
inventory	object	Host inventory properties. 主机资产清单属性。

Return values 返回值

(object) Returns an object containing the IDs of the created hosts under the `hostids` property. The order of the returned IDs matches the order of the passed hosts.返回一个对象其中包含在hostids属性下已创建主机的ID。返回ID的命令与传递主机的命令相匹配。

Examples范例

Creating a host 创建一个主机

Create a host called “Linux server” with an IP interface, add it to a group, link a template to it and set the MAC addresses in the host inventory.创建一个具有IP接口的“Linux Server”主机，将其添加到主机组中，链接一个模板并且把MAC地址设置到主机资产清单里。

Request:

```
1. {
```

```
host.create
```

```
2.     "jsonrpc": "2.0",
3.     "method": "host.create",
4.     "params": {
5.         "host": "Linux server",
6.         "interfaces": [
7.             {
8.                 "type": 1,
9.                 "main": 1,
10.                "useip": 1,
11.                "ip": "192.168.3.1",
12.                "dns": "",
13.                "port": "10050"
14.            }
15.        ],
16.        "groups": [
17.            {
18.                "groupid": "50"
19.            }
20.        ],
21.        "templates": [
22.            {
23.                "templateid": "20045"
24.            }
25.        ],
26.        "inventory_mode": 0,
27.        "inventory": {
28.            "macaddress_a": "01234",
29.            "macaddress_b": "56768"
30.        }
31.    },
32.    "auth": "038e1d7b1735c6a5436ee9eae095879e",
33.    "id": 1
34. }
```

Response:

```
1. {
2.     "jsonrpc": "2.0",
3.     "result": {
4.         "hostids": [
5.             "107819"
6.         ]
7.     },
8.     "id": 1
9. }
```

See also 参见

- [Host group](#)
- [Template](#)

- [User macro](#)
- [Host interface](#)
- [Host inventory](#)

Source 来源

`CHost::create()` in *frontends/php/include/classes/api/services/CHost.php*.

host.delete

Description 说明

```
object host.delete(array hosts)
```

This method allows to delete hosts.此方法允许删除主机。

Parameters 参数

(array) IDs of hosts to delete. 删除主机的ID

Return values 返回值

(object) Returns an object containing the IDs of the deleted hosts under the hostids property.返回一个对象其中包含在hostids属性下已删除主机的ID。

Examples 范例

Deleting multiple hosts 删除多个主机

Delete two hosts.删除两个主机。

Request:

```

1. {
2.   "jsonrpc": "2.0",
3.   "method": "host.delete",
4.   "params": [
5.     "13",
6.     "32"
7.   ],
8.   "auth": "038e1d7b1735c6a5436ee9eae095879e",
9.   "id": 1
10. }
```

Response:

```

1. {
2.   "jsonrpc": "2.0",
3.   "result": {
4.     "hostids": [
5.       "13",
6.       "32"
7.     ]
8.   },
9.   "id": 1

```

```
10. }
```

Source 来源

`CHost::delete()` in *frontends/php/include/classes/api/services/CHost.php*.

host.get

Description 说明

`integer/array host.get(object parameters)`

此方法允许根据指定的参数获取主机。

Parameters参数

(object) 定义期望输出的参数

该方法支持以下参数。

参数	类型	说明
groupids	string/array	仅返回指定主机组所属的主机。
applicationids	string/array	仅返回含有指定应用集的主机。
dserviceids	string/array	仅返回与指定自动发现服务相关的主机。
graphids	string/array	仅返回含有指定图表的主机。
hostids	string/array	仅返回指定主机ID的主机。
httptestids	string/array	仅返回含有指定网页监测的主机。
interfaceids	string/array	仅返回使用指定接口的主机。
itemid	string/array	仅返回含有指定监控项的主机。
maintenanceids	string/array	仅返回被指定管理影响到的主机。
monitoredhosts	flag	仅返回被监控的主机。
proxy_hosts	flag	仅返回代理服务器。
proxyids	string/array	仅返回被代理服务器监控的主机。
templated_hosts	flag	同时返回主机和模板。
templateids	string/array	仅返回与指定模板链接的主机。
triggerids	string/array	仅返回含有指定触发器的主机。
with_items	flag	仅返回含有监控项的主机。Overrides <code>t</code> and <code>with_simple_graph_items</code> parameters. 覆盖“with_monitored_items”和“with_items”参数。
with_applications	flag	仅返回含有应用集的主机。
with_graphs	flag	仅返回含有图表的主机。
with_httptests	flag	仅返回含有网页监测的主机。 Overrides <code>with_monitored_httptests</code> parameter. 覆盖“with_monitored_httptests”参数。
with_monitored_httptests	flag	仅返回启用网页监测的主机。
	.仅返回启用监控项的主机。	

with_monitored_items	Overrides the <code>with_simple_graph_items</code> parameter. 覆盖 <code>with_simple_graph_items</code> 参数。	
with_monitored_triggers	flag	Return only hosts that have enabled the items used in the trigger monitor. 仅返回启用触发器的主机。所有在触发器中用。
with_simple_graph_items	flag	仅返回含有数字类信息监控项的主机。
with_triggers	flag	仅返回含有触发器的主机。 Overrides the <code>with_monitored_triggers</code> parameter. 覆盖“with_monitored_triggers”参数。
withInventory	flag	仅返回含有资产清单数据的主机。
selectGroups	query	返回在groups属性中主机所属的主机组。
selectApplications	query	返回在applications属性中来自主机的应用。 Supports <code>count</code> . 支持count。
selectDiscoveries	query	返回在discoveries属性中来自主机的底层自动发现规则。 Supports <code>count</code> . 支持count。
selectDiscoveryRule	query	Return the discoveryRule property. 返回在discoveryRule属性中创建主机的底层自动发现规则。
selectGraphs	query	返回在graphs属性中来自主机的图表。 Supports <code>count</code> .
selectHostDiscovery	query	返回在hostDiscovery属性中的主机自动发现规则。自动发现对象将一个自动发现的主机和一个原型主机连接起来，通过自动发现规则将一个底层自动发现规则连接起来，从而将一个自动发现的主机和一个原型主机连接起来。 <code>host</code> - (string) host of the host prototype; (字符串) 自动发现的主机。 <code>hostid</code> - (string) ID of the host prototype; (字符串) 原型主机ID。 <code>parent_hostid</code> - (string) ID of the parent host from which the host has been created. (字符串) 原型主机的底层自动发现规则ID。 <code>last_discovery_time</code> - (string) time when the host was last discovered. (字符串) 最近一次自动发现主机时的时间。 <code>ts_deleted</code> - (string) time when a host that is no longer discovered was deleted. (时间戳) 当不再自动发现时的时间。
selectHttpTests	query	Return the web scenarios from the httpTests property. 返回在httpTests属性中来自主机的Web场景。 Supports <code>count</code> . 支持count。
selectInterfaces	query	Return the interfaces property. 返回在interfaces属性中来自主机的接口。 Supports <code>count</code> . 支持count。
selectInventory	query	Return the inventory property. 返回在inventory属性中来自主机的资产。
selectItems	query	Return the items property. 返回在items属性中来自主机的监控项。 Supports <code>count</code> . 支持count。
selectMacros	query	Return the macros property. 返回在macros属性中来自主机的宏。
selectParentTemplates	query	Return the parentTemplates property. 返回在parentTemplates属性中与主机相关的父模板。 Supports <code>count</code> . 支持count。
selectScreens	query	Return the screens property. 返回在screens属性中来自主机的屏幕。 Supports <code>count</code> .

selectTriggers	query	返回在triggers属性中主机里的触发器。count。
filter	object	仅返回完全匹配指定筛选后的结果。接受-其值要么是一个单一的值，要么是一个匹配filtering by interface properties选。
limitSelects	integer	限定由子查询返回的记录数量。适用于以查询: <code>selectParentTemplates - result host</code> ; 结果将由host排序; <code>selectInt - sorted by name</code> ; 由name排序; <code>sorted by name</code> ; 由name排序; <code>sel by description</code> ; 由description排序; <code>sorted by name</code> ; 由name排序; <code>sel sorted by name</code> ; 由name排序; <code>sel by name</code> 。由name排序;
search	object	Return results that match the g 返回与指定通配符搜索匹配的结果。Acce the keys are property names, an strings to search for. If no ad given, this will perform a <code>LIKE</code> 数组, 其中键值是属性名, 其值为搜索到的 外选项, 将会以LIKE "%..."方式执行搜索 by interface properties. Works fields. 允许通过接口属性搜索, 仅对文
searchInventory	object	仅返回与指定通配符搜索资产清单数据匹配 parameter is affected by the sa parameters as <code>search</code> 。这个参数被 search所影响。
sortfield	string/array	用指定的属性排序结果。Possible val <code>host</code> , <code>name</code> , <code>status</code> 。可能的值为: status。
countOutput	flag	These parameters being common f are described in detail in the 在reference commentary中详细描述了 数。
editable	boolean	
excludeSearch	flag	
limit	integer	
output	query	
preservekeys	flag	
searchByAny	boolean	
searchWildcardsEnabled	boolean	
sortorder	string/array	
startSearch	flag	

返回值

(integer/array) Returns either: 返回其中任一:

- an array of objects; 一组对象;

- the count of retrieved objects, if the `countOutput` parameter has been used.如果已经使用了“countOutput”参数，则检索对象的计数。

范例

根据名称检索数据

Retrieve all data about two hosts named “Zabbix server” and “Linux server”.检索所有关于主机名为“Zabbix server”和“Linux server”的数据。

Request:

```

1. {
2.   "jsonrpc": "2.0",
3.   "method": "host.get",
4.   "params": {
5.     "output": "extend",
6.     "filter": {
7.       "host": [
8.         "Zabbix server",
9.         "Linux server"
10.      ]
11.    }
12.  },
13.  "auth": "038e1d7b1735c6a5436ee9eae095879e",
14.  "id": 1
15. }
```

Response:

```

1. {
2.   "jsonrpc": "2.0",
3.   "result": [
4.     {
5.       "maintenances": [],
6.       "hostid": "10160",
7.       "proxy_hostid": "0",
8.       "host": "Zabbix server",
9.       "status": "0",
10.      "disable_until": "0",
11.      "error": "",
12.      "available": "0",
13.      "errors_from": "0",
14.      "lastaccess": "0",
15.      "ipmi_authtype": "-1",
16.      "ipmi_privilege": "2",
17.      "ipmi_username": "",
18.      "ipmi_password": "",
19.      "ipmi_disable_until": "0",
20.      "ipmi_available": "0",
21.      "snmp_disable_until": "0",
```

```
22.     "snmp_available": "0",
23.     "maintenanceid": "0",
24.     "maintenance_status": "0",
25.     "maintenance_type": "0",
26.     "maintenance_from": "0",
27.     "ipmi_errors_from": "0",
28.     "snmp_errors_from": "0",
29.     "ipmi_error": "",
30.     "snmp_error": "",
31.     "jmx_disable_until": "0",
32.     "jmx_available": "0",
33.     "jmx_errors_from": "0",
34.     "jmx_error": "",
35.     "name": "Zabbix server",
36.     "description": "The Zabbix monitoring server.",
37.     "tls_connect": "1",
38.     "tls_accept": "1",
39.     "tls_issuer": "",
40.     "tls_subject": "",
41.     "tls_psk_identity": "",
42.     "tls_psk": ""
43.   },
44.   {
45.     "maintenances": [],
46.     "hostid": "10167",
47.     "proxy_hostid": "0",
48.     "host": "Linux server",
49.     "status": "0",
50.     "disable_until": "0",
51.     "error": "",
52.     "available": "0",
53.     "errors_from": "0",
54.     "lastaccess": "0",
55.     "ipmi_authtype": "-1",
56.     "ipmi_privilege": "2",
57.     "ipmi_username": "",
58.     "ipmi_password": "",
59.     "ipmi_disable_until": "0",
60.     "ipmi_available": "0",
61.     "snmp_disable_until": "0",
62.     "snmp_available": "0",
63.     "maintenanceid": "0",
64.     "maintenance_status": "0",
65.     "maintenance_type": "0",
66.     "maintenance_from": "0",
67.     "ipmi_errors_from": "0",
68.     "snmp_errors_from": "0",
69.     "ipmi_error": "",
70.     "snmp_error": "",
71.     "jmx_disable_until": "0",
72.     "jmx_available": "0",
73.     "jmx_errors_from": "0",
74.     "jmx_error": ""
```

```
host.get
```

```
75.         "name": "Linux server",
76.         "description": "",
77.         "tls_connect": "1",
78.         "tls_accept": "1",
79.         "tls_issuer": "",
80.         "tls_subject": "",
81.         "tls_psk_identity": "",
82.         "tls_psk": ""
83.     }
84. ],
85. "id": 1
86. }
```

Retrieving host groups 检索主机组

Retrieve names of the groups host “Zabbix server” is member of, but no host details themselves. 检索主机“Zabbix server”隶属于的组名，但是不检索主机本身的详情。

Request:

```
1. {
2.     "jsonrpc": "2.0",
3.     "method": "host.get",
4.     "params": {
5.         "output": ["hostid"],
6.         "selectGroups": "extend",
7.         "filter": {
8.             "host": [
9.                 "Zabbix server"
10.            ]
11.        }
12.    },
13.    "auth": "038e1d7b1735c6a5436ee9eae095879e",
14.    "id": 2
15. }
```

Response:

```
1. {
2.     "jsonrpc": "2.0",
3.     "result": [
4.         {
5.             "hostid": "10085",
6.             "groups": [
7.                 {
8.                     "groupid": "2",
9.                     "name": "Linux servers",
10.                    "internal": "0",
11.                    "flags": "0"
12.                },
13.                {
14.                    "groupid": "3",
15.                    "name": "Production",
16.                    "internal": "0",
17.                    "flags": "0"
18.                }
19.            ]
20.        }
21.    ]
22. }
```

```
host.get
```

```
14.         "groupid": "4",
15.         "name": "Zabbix servers",
16.         "internal": "0",
17.         "flags": "0"
18.     }
19.   ]
20. }
21. ],
22. "id": 2
23. }
```

Retrieving linked templates 检索相关模板

Retrieve the IDs and names of templates linked to host "10084". 检索主机"10084"相关模板的ID和名称。

Request:

```
1. {
2.   "jsonrpc": "2.0",
3.   "method": "host.get",
4.   "params": {
5.     "output": ["hostid"],
6.     "selectParentTemplates": [
7.       "templateid",
8.       "name"
9.     ],
10.    "hostids": "10084"
11.  },
12.  "id": 1,
13.  "auth": "70785d2b494a7302309b48afcdb3a401"
14. }
```

Response:

```
1. {
2.   "jsonrpc": "2.0",
3.   "result": [
4.     {
5.       "hostid": "10084",
6.       "parentTemplates": [
7.         {
8.           "name": "Template OS Linux",
9.           "templateid": "10001"
10.        },
11.        {
12.          "name": "Template App Zabbix Server",
13.          "templateid": "10047"
14.        }
15.      ]
16.    }
```

```
host.get
```

```
17.     ],
18.     "id": 1
19. }
```

Searching by host inventory data 根据主机资产清单数据搜索

Retrieve hosts that contain “Linux” in the host inventory “os” field. 检索在资产清单“操作系统”字段中包含“Linux”的主机。

Request:

```
1. {
2.     "jsonrpc": "2.0",
3.     "method": "host.get",
4.     "params": {
5.         "output": [
6.             "host"
7.         ],
8.         "selectInventory": [
9.             "os"
10.        ],
11.        "searchInventory": {
12.            "os": "Linux"
13.        }
14.    },
15.    "id": 2,
16.    "auth": "7f9e00124c75e8f25facd5c093f3e9a0"
17. }
```

Response:

```
1. {
2.     "jsonrpc": "2.0",
3.     "result": [
4.         {
5.             "hostid": "10084",
6.             "host": "Zabbix server",
7.             "inventory": {
8.                 "os": "Linux Ubuntu"
9.             }
10.        },
11.        {
12.            "hostid": "10107",
13.            "host": "Linux server",
14.            "inventory": {
15.                "os": "Linux Mint"
16.            }
17.        }
18.    ],
19.    "id": 1
20. }
```

See also 参见

- [Host group](#)
- [Template](#)
- [User macro](#)
- [Host interface](#)

Source 来源

`CHost::get()` in *frontends/php/include/classes/api/services/CHost.php*.



host.massadd

Description说明

```
object host.massadd(object parameters)
```

This method allows to simultaneously add multiple related objects to all the given hosts. 此方法允许同时添加多个相关对象到指定的主机。

Parameters 参数

(object) Parameters containing the IDs of the hosts to update and the objects to add to all the hosts. 参数包含升级主机的ID和添加到所有主机中的对象。

The method accepts the following parameters. 该方法接受以下参数。

参数	类型	说明
hosts (required)	object/array	Hosts to be updated. 需要更新的主机。The hosts must have the <code>hostid</code> property defined. 主机必须已定义过hostid属性。
groups	object/array	Host groups to add to the given hosts. 主机组添加到指定的主机中。The host groups must have the <code>groupid</code> property defined. 主机组必须已定义过groupid属性。
interfaces	object/array	Host interfaces to be created for the given hosts. 为指定主机创建的主机接口。
macros	object/array	User macros to be created for the given hosts. 为指定主机创建的宏。
templates	object/array	Templates to link to the given hosts. 链接到指定主机的模板。The templates must have the <code>templateid</code> property defined. 模板必须已定义过templateid属性。

Return values返回值

(object) Returns an object containing the IDs of the updated hosts under the `hostids` property. 返回一个对象其中包含在hostids属性下已更新主机的ID。

Examples 范例

Adding macros 添加宏

Add two new macros to two hosts. 添加两个新的宏到两个主机中。

Request:

```
1. {
2.     "jsonrpc": "2.0",
```

```

3.     "method": "host.massadd",
4.     "params": {
5.       "hosts": [
6.         {
7.           "hostid": "10160"
8.         },
9.         {
10.           "hostid": "10167"
11.         }
12.       ],
13.       "macros": [
14.         {
15.           "macro": "{$TEST1}",
16.           "value": "MACROTEST1"
17.         },
18.         {
19.           "macro": "{$TEST2}",
20.           "value": "MACROTEST2"
21.         }
22.       ]
23.     },
24.     "auth": "038e1d7b1735c6a5436ee9eae095879e",
25.     "id": 1
26.   }

```

Response:

```

1. {
2.   "jsonrpc": "2.0",
3.   "result": {
4.     "hostids": [
5.       "10160",
6.       "10167"
7.     ]
8.   },
9.   "id": 1
10. }

```

See also参见

- [host.update](#)
- [Host group](#)
- [Template](#)
- [User macro](#)
- [Host interface](#)

Source 来源

`CHost::massAdd() in frontends/php/include/classes/api/services/CHost.php.`

host.massremove

Description说明

```
object host.massremove(object parameters)
```

This method allows to remove related objects from multiple hosts.此方法允许从多个主机中移除相关对象。

Parameters参数

(object) Parameters containing the IDs of the hosts to update and the objects that should be removed.参数包含更新主机的ID和需要被移除的对象。

参数	类型	说明
hostids(required)	string/array	IDs of the hosts to be updated.需要更新的主机ID。
groupids	string/array	Host groups to remove the given hosts from. 从指定的主机中移除主机组。
interfaces	object/array	Host interfaces to remove from the given hosts. 从指定的主机中移除主机接口。 The host interface object must have the ip, dns and port properties defined. 主机接口对象必须已定义过ip, dns 和port属性。
macros	string/array	User macros to delete from the given hosts. 从指定主机中删除用户宏。
templateids	string/array	Templates to unlink from the given hosts. 从指定主机中删除模板链接。
templateids_clear	string/array	Templates to unlink and clear from the given hosts. 从指定主机中删除并清除模板链接。

Return values返回值

(object) Returns an object containing the IDs of the updated hosts under the hostids property.返回一个对象其中包含在hostids属性下已更新主机的ID。

Examples范例

Unlinking templates 删除模板链接

Unlink a template from two hosts and delete all of the templated entities.从两个主机中删除一个模板链接并且删除所有模板实体。

Request:

```
1. {
2.     "jsonrpc": "2.0",
```

```
3.     "method": "host.massremove",
4.     "params": {
5.       "hostids": ["69665", "69666"],
6.       "templateids_clear": "325"
7.     },
8.     "auth": "038e1d7b1735c6a5436ee9eae095879e",
9.     "id": 1
10. }
```

Response:

```
1. {
2.   "jsonrpc": "2.0",
3.   "result": {
4.     "hostids": [
5.       "69665",
6.       "69666"
7.     ]
8.   },
9.   "id": 1
10. }
```

See also 参见

- [host.update](#)
- [User macro](#)
- [Host interface](#)

Source 来源

`CHost::massRemove()` in *frontends/php/include/classes/api/services/CHost.php*.

host.massupdate

Description 说明

```
object host.massupdate(object parameters)
```

This method allows to simultaneously replace or remove related objects and update properties on multiple hosts. 此方法允许同时替换或移除相关对象和在多个主机中更新属性。

Parameters 参数

(object) Parameters containing the IDs of the hosts to update and the properties that should be updated. 参数包含更新主机的ID和需要更新的属性。

Additionally to the [standard host properties](#), the method accepts the following parameters. 另外，对于标准主机属性，此方法接受以下参数。

参数	类型	说明
hosts(required)	object/array	Hosts to be updated. 需要更新的主机。The hosts must have the <code>hostid</code> property defined. 主机必须已定义过hostid属性。
groups	object/array	Host groups to replace the current host groups the hosts belong to. 替换当前主机所属主机组。The host groups must have the <code>groupid</code> property defined. 主机组必须已定义过groupid属性。
interfaces	object/array	Host interfaces to replace the current host interfaces on the given hosts. 在指定主机上替换当前主机接口。
inventory	object	Host inventory properties. 主机资产清单属性。 Host inventory mode cannot be updated using the <code>inventory</code> parameter, use <code>inventory_mode</code> instead. 使用参数inventory无法更新主机资产清单模式，用参数inventory_mode替换
inventory_mode	integer	Host inventory population mode. 主机资产清单群体模式。Refer to the host inventory object page for a list of supported inventory modes. 参考host inventory object page支持的资产清单模式列表
macros	object/array	User macros to replace the current user macros on the given hosts. 在指定主机中替换当前用户宏。
templates	object/array	Templates to replace the currently linked templates on the given hosts. 在指定主机中替换当前链接的模板。The templates must have the <code>templateid</code> property defined. 模板必须已定义过templateid属性。
templates_clear	object/array	Templates to unlink and clear from the given hosts. 在指定主机中删除模板链接并清除。The templates must have the <code>templateid</code> property defined. 模板必须已定义过templateid属性。

Return values 返回值

(object) Returns an object containing the IDs of the updated hosts under the hostids property.返回一个对象其中包含在hostids属性下已更新主机的ID

Examples 范例

Enabling multiple hosts 启用多个主机

Enable monitoring of two hosts, i.e., set their status to 0.启用两个主机的监控，例如，把这两个主机的状态设置为0。

Request:

```
1. {
2.     "jsonrpc": "2.0",
3.     "method": "host.massupdate",
4.     "params": {
5.         "hosts": [
6.             {
7.                 "hostid": "69665"
8.             },
9.             {
10.                "hostid": "69666"
11.            }
12.        ],
13.        "status": 0
14.    },
15.    "auth": "038e1d7b1735c6a5436ee9eae095879e",
16.    "id": 1
17. }
```

Response:

```
1. {
2.     "jsonrpc": "2.0",
3.     "result": {
4.         "hostids": [
5.             "69665",
6.             "69666"
7.         ]
8.     },
9.     "id": 1
10. }
```

See also 参见

- [host.update](#)
- [host.massadd](#)

- [host.massremove](#)
- [Host group](#)
- [Template](#)
- [User macro](#)
- [Host interface](#)

Source 来源

`CHost::massUpdate()` in *frontends/php/include/classes/api/services/CHost.php*.

host.update

Description 说明

```
object host.update(object/array hosts)
```

This method allows to update existing hosts. 该方法允许更新目前的主机。

Parameters 参数

(object/array) Host properties to be updated. 需要更新的主机属性。

The `hostid` property must be defined for each host, all other properties are optional. Only the given properties will be updated, all others will remain unchanged. 每一个主机的hostid属性必须已定义过，其他属性为可选项。Only the given properties will be updated, all others will remain unchanged. 仅指定属性会被更新，其他属性保持不变。

Additionally to the [standard host properties](#), the method accepts the following parameters. 另外，对于标准主机属性，此方法接受以下参数。

参数	类型	说明
groups	object/array	Host groups to replace the current host groups the host belongs to. 替换当前主机所属主机组。The host groups must have the <code>groupid</code> property defined. 主机组必须已定义过groupid属性
interfaces	object/array	Host interfaces to replace the current host interfaces. 替换当前主机接口。
inventory	object	Host inventory properties. 主机资产清单属性。
macros	object/array	User macros to replace the current user macros. 替换当前用户宏。
templates	object/array	Templates to replace the currently linked templates. Templates that are not passed are only unlinked. 替换当前链接的模板。 模板没有传递仅删除链接。 The templates must have the <code>templateid</code> property defined. 模板必须已定义过templateid属性。
templates_clear	object/array	Templates to unlink and clear from the host. 从主机中删除模板链接并清除。The templates must have the <code>templateid</code> property defined. 模板必须已定义过templateid属性。

As opposed to the Zabbix frontend, when `name` is the same as `host`, updating `host` will not automatically update `name`. Both properties need to be updated explicitly. 相对于Zabbix前端，当name和host一致，更新host的时候不会自动更新name。两个属性需要明确的更新。

Return values 返回值

(object) Returns an object containing the IDs of the updated hosts under the `hostids`

host.update

property. 返回一个对象其中包含在hostids属性下已更新主机的ID。

Examples 范例

Enabling a host 启用一个主机

Enable host monitoring, i.e. set its status to 0. 启用主机监控，例如，把主机状态设置为0。

Request:

```
1. {
2.     "jsonrpc": "2.0",
3.     "method": "host.update",
4.     "params": {
5.         "hostid": "10092",
6.         "status": 0
7.     },
8.     "auth": "038e1d7b1735c6a5436ee9eae095879e",
9.     "id": 1
10. }
```

Response:

```
1. {
2.     "jsonrpc": "2.0",
3.     "result": {
4.         "hostids": [
5.             "10126"
6.         ]
7.     },
8.     "id": 1
9. }
```

Unlinking templates 删除模板链接

Unlink and clear two templates from host. 从主机中删除链接并清除两个模板。

Request:

```
1. {
2.     "jsonrpc": "2.0",
3.     "method": "host.update",
4.     "params": {
5.         "hostid": "10126",
6.         "templates_clear": [
7.             {
8.                 "templateid": "10124"
9.             },
10.            {
11.                "templateid": "10125"
12.            }
13.        ]
14.    }
15. }
```

host.update

```
12.         }
13.     ],
14.   },
15.   "auth": "038e1d7b1735c6a5436ee9eae095879e",
16.   "id": 1
17. }
```

Response:

```
1. {
2.   "jsonrpc": "2.0",
3.   "result": {
4.     "hostids": [
5.       "10126"
6.     ]
7.   },
8.   "id": 1
9. }
```

Updating host macros更新主机的宏

Replace all host macros with two new ones.用两个新的宏替换主机所有的宏。

Request:

```
1. {
2.   "jsonrpc": "2.0",
3.   "method": "host.update",
4.   "params": {
5.     "hostid": "10126",
6.     "macros": [
7.       {
8.         "macro": "{$PASS}",
9.         "value": "password"
10.      },
11.      {
12.        "macro": "{$DISC}",
13.        "value": "sda"
14.      }
15.    ],
16.  },
17.  "auth": "038e1d7b1735c6a5436ee9eae095879e",
18.  "id": 1
19. }
```

Response:

```
1. {
2.   "jsonrpc": "2.0",
3.   "result": {
```

```
host.update
```

```
4.     "hostids": [
5.         "10126"
6.     ],
7. },
8. "id": 1
9. }
```

Updating host inventory 更新主机资产清单

Change inventory mode and add location更改资产清单模式并添加地点。

Request:

```
1. {
2.     "jsonrpc": "2.0",
3.     "method": "host.update",
4.     "params": {
5.         "hostid": "10387",
6.         "inventory_mode": 0,
7.         "inventory": {
8.             "location": "Latvia, Riga"
9.         }
10.    },
11.    "auth": "038e1d7b1735c6a5436ee9eae095879e",
12.    "id": 1
13. }
```

Response:

```
1. {
2.     "jsonrpc": "2.0",
3.     "result": {
4.         "hostids": [
5.             "10387"
6.         ]
7.     },
8.     "id": 2
9. }
```

See also 参见

- [host.massadd](#)
- [host.massupdate](#)
- [host.massremove](#)
- [Host group](#)
- [Template](#)

- [User macro](#)
- [Host interface](#)
- [Host inventory](#)

Source 来源

`CHost::update()` in *frontends/php/include/classes/api/services/CHost.php*.

主机组

该类用于管理主机组。

对象引用：

- [主机组](#)

可用的方法：

- [hostgroup.create](#) - creating new host groups
- [hostgroup.delete](#) - deleting host groups
- [hostgroup.get](#) - retrieving host groups
- [hostgroup.massadd](#) - adding related objects to host groups
- [hostgroup.massremove](#) - removing related objects from host groups
- [hostgroup.massupdate](#) - replacing or removing related objects from host groups
- [hostgroup.update](#) - updating host groups

> Host group object主机组对象

The following objects are directly related to the [hostgroup API](#). 以下对象与“主机组”API直接相关。

Host group主机组

The host group object has the following properties. 主机组对象具有以下属性。

属性	类型	说明
groupid	string	(readonly) ID of the host group. 主机组的ID
name(required)	string	Name of the host group. 主机组的名称
flags	integer	(readonly) Origin of the host group. 主机组的来源。 Possible values: 可能的值0 - a plain host group;一个普通的主机组 4 - a discovered host group. 被发现的主机组。
internal	integer	(readonly) Whether the group is used internally by the system. An internal group cannot be deleted. 该组是否由系统内部使用。无法删除内部组。Possible values: 可能的值: 0 - (default默认) not internal;不是内部的 1 - internal. 内部的

hostgroup.create

Description说明

```
object hostgroup.create(object/array hostGroups)
```

This method allows to create new host groups.此方法允许创建新的主机组。

Parameters 参数

(object/array) Host groups to create. The method accepts host groups with the standard host group properties.主机组创建。 该方法接受具有标准主机组属性的主机组

Return values 返回值

(object) Returns an object containing the IDs of the created host groups under the groupids property. The order of the returned IDs matches the order of the passed host groups.返回包含“groupids”属性下创建的主机组的ID的对象。返回的ID的顺序与传递的主机组的顺序相匹配。

Examples示例

Creating a host group创建主机组

Create a host group called “Linux servers”.建名为“Linux服务器”的主机组。

Request:

```
1. {
2.   "jsonrpc": "2.0",
3.   "method": "hostgroup.create",
4.   "params": {
5.     "name": "Linux servers"
6.   },
7.   "auth": "038e1d7b1735c6a5436ee9eae095879e",
8.   "id": 1
9. }
```

Response:

```
1. {
2.   "jsonrpc": "2.0",
3.   "result": {
4.     "groupids": [
5.       "107819"
6.     ]
7.   },
8. }
```

```
8.      "id": 1  
9. }
```

Source来源

CHostGroup::create() in *frontends/php/include/classes/api/services/CHostGroup.php*.

hostgroup.delete

Description 说明

```
object hostgroup.delete(array hostGroupIds)
```

This method allows to delete host groups.此方法允许删除主机组。

A host group can not be deleted if:如果主机组有以下情况，则不能被删除：

- it contains hosts that belong to this group only;它只包含属于此组的主机；
- it is marked as internal;被标记为内部；
- it is used by a host prototype;被主机原型使用；
- it is used in a global script;在全局脚本中使用；
- it is used in a correlation condition.在相关条件下使用。

Parameters参数

(array) IDs of the host groups to delete.要删除的主机组的ID。

Return values 返回值

(object) Returns an object containing the IDs of the deleted host groups under the groupids property.返回包含“groupids”属性下删除的主机组的ID的对象。

Examples 示例

Deleting multiple host groups删除多个主机组

Delete two host groups.删除两个主机组

Request:

```

1. {
2.   "jsonrpc": "2.0",
3.   "method": "hostgroup.delete",
4.   "params": [
5.     "107824",
6.     "107825"
7.   ],
8.   "auth": "3a57200802b24cda67c4e4010b50c065",
9.   "id": 1
10. }
```

Response:

```
1. {
2.     "jsonrpc": "2.0",
3.     "result": {
4.         "groupids": [
5.             "107824",
6.             "107825"
7.         ]
8.     },
9.     "id": 1
10. }
```

Source来源

CHostGroup::delete() in *frontends/php/include/classes/api/services/CHostGroup.php*.

hostgroup.get

Description 说明

```
integer/array hostgroup.get(object parameters)
```

The method allows to retrieve host groups according to the given parameters. 该方法允许根据给定的参数检索主机组。

Parameters 参数

(object) Parameters defining the desired output. 定义所需输出的参数。

The method supports the following parameters. 该方法支持以下参数。

参数	类型	描述
graphids	string/array	Return only host groups that contain hosts templates with the given graphs. 只返回包含具有主机或模板的主机组。
groupids	string/array	Return only host groups with the given host IDs. 只返回具有给定主机组ID的主机组。
hostids	string/array	Return only host groups that contain the given hosts. 只返回包含给定主机的主机组。
maintenanceids	string/array	Return only host groups that are affected by given maintenances. 仅返回受指定维护影响的主机组。
monitoredhosts	flag	Return only host groups that contain monitored hosts. 仅返回包含受监视主机的主机组。
real_hosts	flag	Return only host groups that contain hosts. 只返回包含真实主机的主机组。
templated_hosts	flag	Return only host groups that contain templates. 只返回包含模板的主机组。
templateids	string/array	Return only host groups that contain the given templates. 只返回包含给定模板的主机组。
triggerids	string/array	Return only host groups that contain hosts templates with the given triggers. 仅返回包含触发器的主机或模板的主机组。
with_applications	flag	Return only host groups that contain hosts applications. 仅返回包含具有应用程序主机的主机组。
with_graphs	flag	Return only host groups that contain hosts graphs. 仅返回包含具有图表的主机的主机组。
with_hosts_and_templates	flag	Return only host groups that contain hosts templates. 只返回包含主机或模板的主机组。
with_httptests	flag	Return only host groups that contain hosts checks. 仅返回包含具有Web检查的主机的主机组。 Overrides the <code>with_monitored_httptests</code> parameter. 覆盖“with_monitored_httptests”参数。

with_items	flag	Return only host groups that contain hosts templates with items. 仅返回包含主题或包含项目的组。 Overrides the <code>with_monitored_items</code> and <code>with_simple_graph_items</code> parameters. 覆盖“with_monitored_items”和“with_simple_graph_items”参数。
with_monitored_httptests	flag	Return only host groups that contain hosts enabled web checks. 仅返回包含启用Web检查的主机组。
with_monitored_items	flag	Return only host groups that contain hosts templates with enabled items. 仅返回包含启用项模板的主机组。 Overrides the <code>with_simple_graph_items</code> parameter. 覆盖 <code>with_simple_graph_items</code> 参数。
with_monitored_triggers	flag	Return only host groups that contain hosts enabled triggers. All of the items used in trigger must also be enabled. 仅返回包含启用触发器的主机组。 触发器中使用的所有项目也必须启用。
with_simple_graph_items	flag	Return only host groups that contain hosts numeric items. 仅返回包含具有数字项目的主机的主机组。
with_triggers	flag	Return only host groups that contain hosts triggers. 仅返回包含具有触发器的主机的主机组。 Overrides the <code>with_monitored_triggers</code> parameter. 覆盖“with_monitored_triggers”参数。
selectDiscoveryRule	query	Return the LLD rule that created the host group in the <code>discoveryRule</code> property. 返回在“discoveryRule”属性中返回创建主机组的LLD规则。
selectGroupDiscovery	query	Return the host group discovery object in the <code>groupDiscovery</code> property. 在“groupDiscovery”属性中返回发现对象。The host group discovery object discovered host group to a host group prototype has the following properties: 主机组发现对象将主机组链接到主机组原型，并具有以下属性： <code>groupid</code> - (string) ID of the discovered host group; 发现的主机组的ID; <code>lastcheck</code> - (timestamp) time when the host group was last discovered; 主机组最后发现的时间; <code>name</code> - (string) name of the host group prototype; 主机组原型; <code>parent_group_prototypeid</code> - (string) ID of group prototype from which the host group prototype was created; 创建主机组的主机组原型的ID; <code>ts_delete</code> - (timestamp) time when a host group that is discovered will be deleted. 不再发现主机组将被删除。
selectHosts	query	Return the hosts that belong to the host group in the <code>hosts</code> property. 在“hosts”属性中返回属于主机组的主机。 Supports <code>count</code> .
selectTemplates	query	Return the templates that belong to the host group in the <code>templates</code> property. 在“templates”属性中返回属于主机组的模板。 Supports <code>count</code> .
limitSelects	integer	Limits the number of records returned by subselects: 适用于以下子选项： <code>selectHosts</code> - results will be sorted by <code>host</code> ; 结果将按“主机”排序; <code>selectTemplates</code> - results will be sorted by <code>name</code> ; 结果将按“模板”排序。
sortfield	string/array	Sort the result by the given properties. 按指定属性对结果进行排序。 Possible values are: <code>groupid</code> , <code>name</code> . 可能的值为： <code>groupid</code> , <code>name</code> 。

countOutput	flag	These parameters being common for all <code>get</code> are described in detail in the reference page. 这些参数对于所有的“获取”方法是常见的，在参考详细描述。
editable	boolean	
excludeSearch	flag	
filter	object	
limit	integer	
output	query	
preservekeys	flag	
search	object	
searchByAny	boolean	
searchWildcardsEnabled	boolean	
sortorder	string/array	
startSearch	flag	

Return values 返回值

(integer/array) Returns either:返回:

- an array of objects;一组对象;
- the count of retrieved objects, if the `countOutput` parameter has been used. 如果已经使用“countOutput”参数，则检索到的对象的计数。

Examples 示例

Retrieving data by name按名称检索数据

Retrieve all data about two host groups named “Zabbix servers” and “Linux servers”. 检索有关名为“Zabbix服务器”和“Linux服务器”的两个主机组的所有数据。

Request:

```

1. {
2.   "jsonrpc": "2.0",
3.   "method": "hostgroup.get",
4.   "params": {
5.     "output": "extend",
6.     "filter": {
7.       "name": [
8.         "Zabbix servers",
9.         "Linux servers"
10.      ]
11.    }
12.  },

```

hostgroup.get

```
13.     "auth": "6f38cddc44cfbb6c1bd186f9a220b5a0",
14.     "id": 1
15. }
```

Response:

```
1. {
2.     "jsonrpc": "2.0",
3.     "result": [
4.         {
5.             "groupid": "2",
6.             "name": "Linux servers",
7.             "internal": "0"
8.         },
9.         {
10.            "groupid": "4",
11.            "name": "Zabbix servers",
12.            "internal": "0"
13.        }
14.    ],
15.    "id": 1
16. }
```

See also 参见

- [Host](#)
- [Template](#)

Source来源

`CHostGroup::get()` in *frontends/php/include/classes/api/services/CHostGroup.php*.



hostgroup.massadd

Description 说明

```
object hostgroup.massadd(object parameters)
```

This method allows to simultaneously add multiple related objects to all the given host groups. 该方法允许同时向所有给定的主机组添加多个相关对象。

Parameters 参数

(object) Parameters containing the IDs of the host groups to update and the objects to add to all the host groups. 包含要更新的主机组的ID和要添加到所有主机组的对象的参数。

The method accepts the following parameters. 该方法接受以下参数。

参数	类型	描述
groups(required)	object/array	Host groups to be updated. 要更新的主机组。 The host groups must have the <code>groupid</code> property defined. 主机组必须定义“groupid”属性。
hosts	object/array	Hosts to add to all host groups. 添加到主机组的所有主机。 The hosts must have the <code>hostid</code> property defined. 主机必须定义“hostid”属性。
templates	object/array	Templates to add to all host groups. 添加到所有主机组的模板。 The templates must have the <code>templateid</code> property defined. 模板必须定义“templateid”属性。

Return values 返回值

(object) Returns an object containing the IDs of the updated host groups under the `groupids` property. 返回包含“groupids”属性下更新的主机组的ID的对象。

Examples示例

Adding hosts to host groups 将主机添加到主机组

Add two hosts to host groups with IDs 5 and 6. 将两个主机添加到ID为5和6的主机组。

Request:

```

1. {
2.   "jsonrpc": "2.0",
3.   "method": "hostgroup.massadd",
4.   "params": {
5.     "groups": [
6.       {
7.         "groupid": "5"

```

```

8.          },
9.          {
10.             "groupid": "6"
11.         }
12.       ],
13.       "hosts": [
14.         {
15.           "hostid": "30050"
16.         },
17.         {
18.           "hostid": "30001"
19.         }
20.       ]
21.     },
22.     "auth": "f223adf833b2bf2ff38574a67bba6372",
23.     "id": 1
24.   }

```

Response:

```

1.  {
2.    "jsonrpc": "2.0",
3.    "result": {
4.      "groupids": [
5.        "5",
6.        "6"
7.      ]
8.    },
9.    "id": 1
10. }

```

See also 参见

- [Host](#)
- [Template](#)

Source 来源

`CHostGroup::massAdd()` in `frontends/php/include/classes/api/services/CHostGroup.php`.

hostgroup.massremove

Description 说明

```
object hostgroup.massremove(object parameters)
```

This method allows to remove related objects from multiple host groups.此方法允许从多个主机组中删除相关对象。

Parameters 参数

(object) Parameters containing the IDs of the host groups to update and the objects that should be removed.含要更新的主机组的ID和应该删除的对象的参数。

参数	类型	描述
groupids(required)	string/array	IDs of the host groups to be updated.要更新的主机组的ID。
hostids	string/array	Hosts to remove from all host groups.要从所有主机组中删除的主机。
templateids	string/array	Templates to remove from all host groups.要从所有主机组中删除的模板。

Return values 返回值

(object) Returns an object containing the IDs of the updated host groups under the groupids property.返回包含“groupids”属性下更新的主机组的ID的对象。

Examples示例

Removing hosts from host groups从主机组中删除主机

Remove two hosts from the given host groups.从给定的主机组中删除两个主机。

Request:

```

1. {
2.   "jsonrpc": "2.0",
3.   "method": "hostgroup.massremove",
4.   "params": {
5.     "groupids": [
6.       "5",
7.       "6"
8.     ],
9.     "hostids": [
10.      "30050",
11.      "30001"

```

hostgroup.massremove

```
12.     ],
13.   },
14.   "auth": "038e1d7b1735c6a5436ee9eae095879e",
15.   "id": 1
16. }
```

Response:

```
1. {
2.   "jsonrpc": "2.0",
3.   "result": {
4.     "groupids": [
5.       "5",
6.       "6"
7.     ]
8.   },
9.   "id": 1
10. }
```

Source 来源

CHostGroup::massRemove() in *frontends/php/include/classes/api/services/CHostGroup.php*.

hostgroup.massupdate

Description 说明

```
object hostgroup.massupdate(object parameters)
```

This method allows to simultaneously replace or remove related objects for multiple host groups. 该方法允许同时替换或删除多个主机组的相关对象。

Parameters 参数

(object) Parameters containing the IDs of the host groups to update and the objects that should be updated. 包含要更新的主机组的ID和应更新的对象的参数。

参数	类型	描述
groups(required)	object/array	Host groups to be updated. 要更新的主机组。The host groups must have the <code>groupid</code> property defined. 主机组必须定义“groupid”属性。
hosts	object/array	Hosts to replace the current hosts on the given host groups. 主机替换给定主机组上的当前主机。The hosts must have the <code>hostid</code> property defined. 主机必须定义“hostid”属性。
templates	object/array	Templates to replace the current templates on the given host groups. 用于替换给定主机组上当前模板的模板。The templates must have the <code>templateid</code> property defined. 模板必须定义“templateid”属性。

Return values 返回值

(object) Returns an object containing the IDs of the updated host groups under the `groupids` property. 返回包含“groupids”属性下更新的主机组的ID的对象。

Examples 示例

Replacing hosts in a host group 更换主机组中的主机

Replace all hosts in the host group with ID. 用ID替换主机组中的所有主机。

Request:

```

1. {
2.   "jsonrpc": "2.0",
3.   "method": "hostgroup.massupdate",
4.   "params": {
5.     "groups": [
6.       {
7.         "groupid": "6"

```

```
8.         }
9.     ],
10.    "hosts": [
11.      {
12.        "hostid": "30050"
13.      }
14.    ]
15.  },
16.  "auth": "f223adf833b2bf2ff38574a67bba6372",
17.  "id": 1
18. }
```

Response:

```
1. {
2.   "jsonrpc": "2.0",
3.   "result": {
4.     "groupids": [
5.       "6",
6.     ]
7.   },
8.   "id": 1
9. }
```

See also 参见

- [hostgroup.update](#)
- [hostgroup.massadd](#)
- [Host](#)
- [Template](#)

Source 来源

`CHostGroup::massUpdate()` in *frontends/php/include/classes/api/services/CHostGroup.php*.

hostgroup.update

Description 说明

```
object hostgroup.update(object/array hostGroups)
```

This method allows to update existing hosts groups.此方法允许更新现有的主机组。

Parameters 参数

(object/array) Host group properties to be updated. 要更新的主机组属性。

The `groupid` property must be defined for each host group, all other properties are optional. Only the given properties will be updated, all others will remain unchanged. 必须为每个主机组定义“groupid”属性，所有其他属性都是可选的。只有给定的属性将被更新，所有其他属性将保持不变。

Return values 返回值

(object) Returns an object containing the IDs of the updated host groups under the `groupids` property.返回包含“groupids”属性下更新的主机组的ID的对象。

Examples 示例

Renaming a host group 重命名主机组

Rename a host group to “Linux hosts.”将主机组命名为“Linux主机”。

Request:

```

1. {
2.   "jsonrpc": "2.0",
3.   "method": "hostgroup.update",
4.   "params": {
5.     "groupid": "7",
6.     "name": "Linux hosts"
7.   },
8.   "auth": "700ca65537074ec963db7efabda78259",
9.   "id": 1
10. }
```

Response:

```

1. {
2.   "jsonrpc": "2.0",
3.   "result": {
```

```
4.      "groupids": [
5.        "7"
6.      ],
7.    },
8.    "id": 1
9. }
```

Source来源

CHostGroup::update() in *frontends/php/include/classes/api/services/CHostGroup.php*.

事件

这个类适用于配合使用事件。

对象引用：

- [Event](#)

可用的方法：

- `event.get` - 获取事件
- `event.acknowledge` - 确认事件

> Event object

The following objects are directly related to the [event API](#). 以下对象与“事件”API直接相关。

Event 事件

Events are created by the Zabbix server and cannot be modified via the [API](#). 事件由Zabbix服务器创建，不能通过API进行修改。

The event object has the following properties. 事件对象具有以下属性。

参数	类型	描述
eventid	string	ID of the event. 事件的ID
source	integer	Type of the event. 事件的类型 Possible values: 可能的值: 0 - event created by a trigger; 由触发器创建的事件 1 - event created by a discovery rule; 由发现规则创建的事件; 2 - event created by active agent auto-registration; 活动代理自动注册创建的事件; 3 - internal event. 内部事件
object	integer	Type of object that is related to the event. 与事件相关的对象类型。 Possible values for trigger events: 触发事件的可能值: 0 - trigger. 触发器 Possible values for discovery events: 发现事件的可能值: 1 - discovered host; 发现主机 2 - discovered service. 发现服务 Possible values for auto-registration events: 自动注册事件的可能值: 3 - auto-registered host. 自动注册的主机 Possible values for internal events: 内部事件的可能值: 0 - trigger; 触发器 4 - item; 项 5 - LLD rule. LLD规则。
objectid	string	ID of the related object. 相关对象的ID。
acknowledged	integer	Whether the event has been acknowledged. 事件是否被确认
clock	timestamp	Time when the event was created. 事件创建时间
ns	integer	Nanoseconds when the event was created. 事件创建时的纳秒。
value	integer	State of the related object. 相关对象的状态 Possible values for trigger events: 触发事件的可能值: 0 - OK; 正常 1 - problem. 异常 Possible values for discovery events: 发现事件的可能值: 0 - host or service up; 主机或服务正常 1 - host or service down; 机或服务故障 2 - host or service discovered; 主机或服务发现; 3 - host or service lost. 主机或服务丢失。 Possible values for internal events: 内部事件的可能值: 0 - "normal" state; "正常"状态 1 - "unknown" or "not supported" state. "未知"或"不支持"状态。 This parameter is not used for active agent auto-registration events. 此参数不用于活动代理自动注册事件。
r_eventid	string	Recovery event ID 恢复事件ID
c_eventid	string	Problem event ID who generated OK event 生成OK事件的问题事件ID
correlationid	string	Correlation ID 关联ID
userid	string	User ID if the event was manually closed. 事件被手动关闭时的用户ID。

event.acknowledge

Description 说明

```
object event.acknowledge(object/array parameters)
```

This method allows to acknowledge events and add an acknowledgement message. If an event is already acknowledged, a new message will still be added. 该方法允许确认事件并添加确认消息。如果事件已被确认，仍可添加一条新消息。

Only trigger events can be acknowledged. 只有触发事件可以被确认。

Parameters 参数

(object/array) Parameters containing the IDs of the events acknowledge and a message. 包含事件确认的ID的参数和消息。

参数	类别	说明
eventids(required)	string/object	IDs of the events to acknowledge. 要确认的事件的ID。
message	string	Text of the acknowledgement message. 确认消息的文本。
action	integer	

event.get

Description 说明

```
integer/array event.get(object parameters)
```

The method allows to retrieve events according to the given parameters. 该方法允许根据给定的参数检索事件。

Parameters 参数

(object) Parameters defining the desired output. 定义所需输出的参数。

The method supports the following parameters. 该方法支持以下参数。

参数	类型	说明
eventids	string/array	Return only events with the given IDs. 仅返回具有给定ID的事件
groupids	string/array	Return only events created by objects that belong to the given host groups. 仅返回由属于给定主机组的对象创建的事件。
hostids	string/array	Return only events created by objects that belong to the given hosts. 仅返回由属于给定主机的对象创建的事件。
objectids	string/array	Return only events created by the given objects. 仅返回由给定对象创建的事件。
applicationids	string/array	Return only events created by objects that belong to the given applications. Applies only if object is trigger or item. 仅返回属于给定应用程序的对象创建的事件。仅当对象为触发器或项目时才适用。
source	integer	Return only events with the given type. 只返回给定类型的事件。 Refer to the event object page for a list of supported event types. 有关支持的事件类型的列表，请参阅事件对象页面。 Default: 0 - trigger events. 默认值：0 - 触发事件。
object	integer	Return only events created by objects of the given type. 只返回由给定类型的对象创建的事件。 Refer to the event object page for a list of supported object types. 有关支持的对象类型的列表，请参阅事件对象页面。 Default: 0 - trigger. 默认值：0 - 触发。
acknowledged	boolean	If set to <code>true</code> return only acknowledged events. 如果设置为“true”，则只返回确认的事件。
severities	integer/array	Return only events with given trigger severities. Applies only if object is trigger. 仅返回给定触发严重程度的事件。仅当对象被触发时才适用。
		Return only events with given tags. Exact

tags	object	match by tag and case-insensitive search by value. 只返回给定标签的事件。按照标签的精确匹配和按值搜索不区分大小写的搜索。Format: [{"tag": "<tag>", "value": "<value>"}, ...] . 格式: [{"tag": "<tag>", "value": "<value>"}, ...] . An empty array returns all events. 一个空数组返回所有事件。
eventid_from	string	Return only events with IDs greater or equal to the given ID. 只返回ID大于或等于给定ID的事件。
eventid_till	string	Return only events with IDs less or equal to the given ID. 只返回ID小于或等于给定ID的事件。
time_from	timestamp	Return only events that have been created after or at the given time. 仅返回在给定时间时或之后创建的事件
time_till	timestamp	Return only events that have been created before or at the given time. 仅返回在特定时间时或之前创建的事件。
value	integer/array	Return only events with the given values. 仅返回具有给定值的事件。
selectHosts	query	Return hosts containing the object that created the event in the <code>hosts</code> property. Supported only for events generated by triggers, items or LLD rules. 在“hosts”属性中返回包含创建事件的对象的主机。仅对触发器，项目或LLD规则生成的事件支持。
selectRelatedObject	query	

Examples 示例

Retrieving trigger events 检索触发事件

Retrieve the latest events from trigger “13926.” 从触发器“13926”检索最新的事件。

Request:

```

1. {
2.   "jsonrpc": "2.0",
3.   "method": "event.get",
4.   "params": {
5.     "output": "extend",
6.     "select_acknowledges": "extend",
7.     "objectids": "13926",
8.     "sortfield": ["clock", "eventid"],
9.     "sortorder": "DESC"
10.    },
11.    "auth": "038e1d7b1735c6a5436ee9eae095879e",
12.    "id": 1
13.  }

```

Response:

```
1.  {
2.      "jsonrpc": "2.0",
3.      "result": [
4.          {
5.              "acknowledges": [
6.                  {
7.                      "acknowledgeid": "1",
8.                      "userid": "1",
9.                      "eventid": "9695",
10.                     "clock": "1350640590",
11.                     "message": "Problem resolved.\n\n-----[BULK ACKNOWLEDGE]-----",
12.                     "alias": "Admin"
13.                 }
14.             ],
15.             "eventid": "9695",
16.             "source": "0",
17.             "object": "0",
18.             "objectid": "13926",
19.             "clock": "1347970410",
20.             "value": "1",
21.             "acknowledged": "1",
22.             "ns": "413316245",
23.             "r_eventid": "0",
24.             "c_eventid": "0",
25.             "correlationid": "0",
26.             "userid": "0",
27.             "tags": [
28.                 {
29.                     "tag": "service",
30.                     "value": "mysqld"
31.                 },
32.                 {
33.                     "tag": "error",
34.                     "value": ""
35.                 }
36.             ]
37.         },
38.         {
39.             "acknowledges": [],
40.             "eventid": "9671",
41.             "source": "0",
42.             "object": "0",
43.             "objectid": "13926",
44.             "clock": "1347970347",
45.             "value": "0",
46.             "acknowledged": "0",
47.             "ns": "0",
48.             "r_eventid": "0",
49.             "c_eventid": "0",
50.             "correlationid": "0",
51.             "userid": "0",
52.             "tags": []
53.         }
54.     ]
55. }
```

```
event.get
```

```
54.     ],
55.     "id": 1
56. }
```

Retrieving events by time period 按时间段检索事件

Retrieve all events that have been created between October 9 and 10, 2012, in reverse chronological order. 检索2012年10月9日至10日之间以反时间顺序创建的所有活动。

Request:

```
1. {
2.     "jsonrpc": "2.0",
3.     "method": "event.get",
4.     "params": {
5.         "output": "extend",
6.         "time_from": "1349797228",
7.         "time_till": "1350661228",
8.         "sortfield": ["clock", "eventid"],
9.         "sortorder": "desc"
10.    },
11.    "auth": "038e1d7b1735c6a5436ee9eae095879e",
12.    "id": 1
13. }
```

Response:

```
1. {
2.     "jsonrpc": "2.0",
3.     "result": [
4.         {
5.             "eventid": "20616",
6.             "source": "0",
7.             "object": "0",
8.             "objectid": "14282",
9.             "clock": "1350477814",
10.            "value": "1",
11.            "acknowledged": "0",
12.            "ns": "0",
13.            "r_eventid": "0",
14.            "c_eventid": "0",
15.            "correlationid": "0",
16.            "userid": "0"
17.        },
18.        {
19.            "eventid": "20617",
20.            "source": "0",
21.            "object": "0",
22.            "objectid": "14283",
23.            "clock": "1350477814",
24.            "value": "0",
```

```
event.get
```

```
25.         "acknowledged": "0",
26.         "ns": "0",
27.         "r_eventid": "0",
28.         "c_eventid": "0",
29.         "correlationid": "0",
30.         "userid": "0"
31.     },
32.     {
33.         "eventid": "20618",
34.         "source": "0",
35.         "object": "0",
36.         "objectid": "14284",
37.         "clock": "1350477815",
38.         "value": "1",
39.         "acknowledged": "0",
40.         "ns": "0",
41.         "r_eventid": "0",
42.         "c_eventid": "0",
43.         "correlationid": "0",
44.         "userid": "0"
45.     }
46. ],
47. "id": 1
48. }
```

See also 参见

- [Alert](#)
- [Item](#)
- [Host](#)
- [LLD rule](#)
- [Trigger](#)

Source 来源

`CEvent::get()` in *frontends/php/include/classes/api/services/CEvent.php*.

值映射

此类用于值映射的使用

对象引用：

- `Value map`

可用的方法：

- `valuemap.create` - 创建新的value maps
- `valuemap.delete` - 删除value maps
- `valuemap.get` - 检索value maps
- `valuemap.update` - 更新 value maps

> 值映射对象

以下对象与“VALUE”API直接相关。

值映射

值映射对象具有以下属性。

属性	类型	说明
valuemapid	string	(<i>readonly</i>) 值映射的ID
name (required)	string	值映射的名称。
mappings (required)	array	值映射当前映射值。值映射对象 value_mappings 细节描述如下。

价值映射

值映射对象定义值映射的映射值。 它具有以下属性。

属性	类型	说明
value (required)	string	原值。
newvalue (required)	string	原始值映射到的值。

valuemap.create

说明

```
object valuemap.create(object/array valuemaps)
```

此方法允许创建新的值映射。

参数

(object/array) 要创建的值映射。

该方法接受有 [标准值映射属性](#) 的值映射 .

返回值

(object) 返回一个包含创建值的ID的对象映射“'valemapids'"属性。 返回的ID的顺序与传递的值映射的顺序相匹配。

示例

创建一个值映射

Create one value map with two mappings.

Request:

```
1. {
2.     "jsonrpc": "2.0",
3.     "method": "valuemap.create",
4.     "params": {
5.         "name": "Service state",
6.         "mappings": [
7.             {
8.                 "value": "0",
9.                 "newvalue": "Down"
10.            },
11.            {
12.                "value": "1",
13.                "newvalue": "Up"
14.            }
15.        ],
16.    },
17.    "auth": "57562fd409b3b3b9a4d916d45207bbcb",
18.    "id": 1
19. }
```

Response:

```
1. {
2.     "jsonrpc": "2.0",
3.     "result": {
4.         "valuemapids": [
5.             "1"
6.         ],
7.     },
8.     "id": 1
9. }
```

来源

CValueMap::create() in *frontends/php/include/classes/api/services/CValueMap.php*.

valuemap.delete

说明

```
object valuemap.delete(array valuemapids)
```

此方法允许删除值映射。

参数

(array) 要被删除的映射的ID。

返回值

(object) 返回一个对象，该对象包含“VALUE”属性下的已删除值映射的ID。

示例

Deleting multiple value maps

Delete two value maps.

Request:

```
1. {
2.   "jsonrpc": "2.0",
3.   "method": "valuemap.delete",
4.   "params": [
5.     "1",
6.     "2"
7.   ],
8.   "auth": "57562fd409b3b3b9a4d916d45207bbcb",
9.   "id": 1
10. }
```

Response:

```
1. {
2.   "jsonrpc": "2.0",
3.   "result": {
4.     "valuemapids": [
5.       "1",
6.       "2"
7.     ]
8.   },
9.   "id": 1
10. }
```

来源

CValueMap::delete() in *frontends/php/include/classes/api/services/CValueMap.php*.

valuemap.get

说明

`integer/array valuemap.get(object parameters)`

该方法允许根据给定的参数来检索值映射。

参数

(object) 定义所需输出的参数。

该方法支持以下参数。

属性	类型	说明
valuemapids	string/array	只返回具有给定ID的值映射。
selectMappings	query	在“映射”属性中返回当前值映射的值映射。
sortfield	string/array	按照给定的属性对结果进行排序。可能的值为: <code>valuemapid</code> , <code>name</code> 。
countOutput	flag	这些参数对于所有的“get”方法是常见的，在 参考评论 中有详细描述。
editable	boolean	
excludeSearch	flag	
filter	object	
limit	integer	
output	query	
preservekeys	flag	
search	object	
searchByAny	boolean	
searchWildcardsEnabled	boolean	
sortorder	string/array	
startSearch	flag	

Return values

(integer/array) Returns either:

- an array of objects;
- the count of retrieved objects, if the `countOutput` parameter has been used.

示例

Retrieving value maps

Retrieve all configured value maps.

Request:

```

1. {
2.   "jsonrpc": "2.0",
3.   "method": "valuemap.get",
4.   "params": {
5.     "output": "extend"
6.   },
7.   "auth": "57562fd409b3b3b9a4d916d45207bbcb",
8.   "id": 1
9. }
```

Response:

```

1. {
2.   "jsonrpc": "2.0",
3.   "result": [
4.     {
5.       "valuemapid": "4",
6.       "name": "APC Battery Replacement Status"
7.     },
8.     {
9.       "valuemapid": "5",
10.      "name": "APC Battery Status"
11.    },
12.    {
13.      "valuemapid": "7",
14.      "name": "Dell Open Manage System Status"
15.    }
16.  ],
17.  "id": 1
18. }
```

Retrieve one value map with its mappings.

Request:

```

1. {
2.   "jsonrpc": "2.0",
3.   "method": "valuemap.get",
4.   "params": {
5.     "output": "extend",
6.     "selectMappings": "extend",
7.     "valuemapid": ["4"]}
```

```
valuemap.get
```

```
8.     },
9.     "auth": "57562fd409b3b3b9a4d916d45207bbcb",
10.    "id": 1
11. }
```

Response:

```
1. {
2.   "jsonrpc": "2.0",
3.   "result": [
4.     {
5.       "valuemapid": "4",
6.       "name": "APC Battery Replacement Status",
7.       "mappings": [
8.         {
9.           "value": "1",
10.          "newvalue": "unknown"
11.        },
12.        {
13.          "value": "2",
14.          "newvalue": "notInstalled"
15.        },
16.        {
17.          "value": "3",
18.          "newvalue": "ok"
19.        },
20.        {
21.          "value": "4",
22.          "newvalue": "failed"
23.        },
24.        {
25.          "value": "5",
26.          "newvalue": "highTemperature"
27.        },
28.        {
29.          "value": "6",
30.          "newvalue": "replaceImmediately"
31.        },
32.        {
33.          "value": "7",
34.          "newvalue": "lowCapacity"
35.        }
36.      ]
37.    }
38.  ],
39.  "id": 1
40. }
```

来源

CValueMap::get() in *frontends/php/include/classes/api/services/CValueMap.php*.

valuemap.update

说明

```
object valuemap.update(object/array valuemaps)
```

该方法允许更新现有的值映射。

P参数

(object/array) 要更新的值映射特性。

必须为每个值映射定义 `valuemapid`'属性，所有其他属性都是可选的。只有通过的属性将被更新，所有其他属性将保持不变。

===== 返回值 =====

(object) 返回一个对象，它包含 `valuemapids`'属性下更新的值映射的ID。

示例

Changing value map name

Change value map name to "Device status".

Request:

```
1. {
2.   "jsonrpc": "2.0",
3.   "method": "valuemap.update",
4.   "params": {
5.     "valuemapid": "2",
6.     "name": "Device status"
7.   },
8.   "auth": "57562fd409b3b3b9a4d916d45207bbcb",
9.   "id": 1
10. }
```

Response:

```
1. {
2.   "jsonrpc": "2.0",
3.   "result": {
4.     "valuemapids": [
5.       "2"
6.     ]
7.   },
8. }
```

```
valuemap.update
```

```
8.     "id": 1
9. }
```

Changing mappings for one value map.

Request:

```
1. {
2.     "jsonrpc": "2.0",
3.     "method": "valuemap.update",
4.     "params": {
5.         "valuemapid": "2",
6.         "mappings": [
7.             {
8.                 "value": "0",
9.                 "newvalue": "Online"
10.            },
11.            {
12.                "value": "1",
13.                "newvalue": "Offline"
14.            }
15.        ],
16.    },
17.    "auth": "57562fd409b3b3b9a4d916d45207bbcb",
18.    "id": 1
19. }
```

Response:

```
1. {
2.     "jsonrpc": "2.0",
3.     "result": {
4.         "valuemaps": [
5.             "2"
6.         ],
7.     },
8.     "id": 1
9. }
```

来源

CValueMap::update() in *frontends/php/include/classes/api/services/CValueMap.php*.

历史数据

该类用于使用历史数据。

对象引用：

- [历史数据](#)

可用的方法：

- `history.get` - 获取历史数据

> History object历史对象

The following objects are directly related to the [history API](#). 以下对象与“历史”API直接相关。

History objects differ depending on the item's type of information. They are created by the Zabbix server and cannot be modified via the [API](#). 历史对象根据项目的信息类型而有所不同。它们由Zabbix服务器创建，不能通过API进行修改。

Float history 浮动历史

The float history object has the following properties. 浮动历史对象具有以下属性。

属性	类型	说明
clock	timestamp	Time when that value was received. 收到这个值的时间
itemid	string	ID of the related item. 相关项的ID。
ns	integer	Nanoseconds when the value was received. 收到值时纳秒。
value	float	Received value. 获取值。

Integer history 整数历史

The integer history object has the following properties. 整数历史对象具有以下属性。

属性	类型	说明
clock	timestamp	Time when that value was received. 收到这个值的时间
itemid	string	ID of the related item. 相关项的ID。
ns	integer	Nanoseconds when the value was received. 收到值时的纳秒。
value	integer	Received value. 获取值。

String history 字符串历史

The string history object has the following properties. 字符串历史对象具有以下属性。

属性	类型	说明
clock	timestamp	Time when that value was received. 收到这个值的时间
itemid	string	ID of the related item. 相关项的ID。
ns	integer	Nanoseconds when the value was received. 收到值时的纳秒。
value	string	Received value. 获取值。

Text history 文字记录

The text history object has the following properties. 文本历史对象具有以下属性。

属性	类型	说明
id	string	ID of the history entry. 历史记录条目的ID。
clock	timestamp	Time when that value was received. 收到这个值的时间
itemid	string	ID of the related item. 相关项的ID。
ns	integer	Nanoseconds when the value was received. 收到值时的纳秒。
value	text	Received value. 获取值。

Log history日志历史

The log history object has the following properties. 日志历史对象具有以下属性。

属性	类型	说明
id	string	ID of the history entry. 历史记录条目的ID。
clock	timestamp	Time when that value was received. 收到这个值的时间
itemid	string	ID of the related item. 相关项的ID。
logeventid	integer	Windows event log entry ID. Windows事件日志条目ID。
ns	integer	Nanoseconds when the value was received. 收到值时的纳秒。
severity	integer	Windows event log entry level. Windows事件日志条目级别。
source	string	Windows event log entry source. Windows事件日志条目源。
timestamp	timestamp	Windows event log entry time. Windows事件日志输入时间。
value	text	Received value. 获取值。

history.get

Description 说明

```
integer/array history.get(object parameters)
```

The method allows to retrieve history data according to the given parameters.该方法允许根据给定的参数检索历史数据。

Parameters参数

(object) Parameters defining the desired output. 定义所需输出的参数。

The method supports the following parameters.该方法支持以下参数。

参数	类型	描述
history	integer	History object types to return. 要返回的历史对象类型。可能的值 0 - numeric float; 数字浮点数 1 - character; 字符 2 - log; 日志 3 - numeric unsigned; 数字符号 4 - text. 文本 Default: 3. 默认: 3;
hostids	string/array	Return only history from the given hosts. 只返回给定主机的历史记录。
itemids	string/array	Return only history from the given items. 只返回给定项的历史记录。
time_from	timestamp	Return only values that have been received after or at the given time. 仅返回在给定时间时或之后收到的值
time_till	timestamp	Return only values that have been received before or at the given time. 仅返回在给定时间时或之前收到的值。
sortfield	string/array	Sort the result by the given properties. 按照给定的属性对结果进行排序。可能的值为: itemid 和 clock 。
countOutput	flag	These parameters being common for all get methods are described in detail in the reference commentary page. 这些参数对于所有的“获取”方法是常见的，在参考评论页中有详细描述。
editable	boolean	
excludeSearch	flag	
filter	object	
limit	integer	
output	query	
search	object	
searchByAny	boolean	

searchWildcardsEnabled	boolean
sortorder	string/array
startSearch	flag

Return values 返回值

(integer/array) Returns either: 返回:

- an array of objects; 一组对象;
- the count of retrieved objects, if the `countOutput` parameter has been used. 如果已经使用“countOutput”参数，则检索到的对象的计数。

Examples 示例

Retrieving item history data 检索项目历史数据

Return 10 latest values received from a numeric(float) item. 从数字(浮动)项返回10个最新值。

Request:

```

1. {
2.   "jsonrpc": "2.0",
3.   "method": "history.get",
4.   "params": {
5.     "output": "extend",
6.     "history": 0,
7.     "itemids": "23296",
8.     "sortfield": "clock",
9.     "sortorder": "DESC",
10.    "limit": 10
11.  },
12.  "auth": "038e1d7b1735c6a5436ee9eae095879e",
13.  "id": 1
14. }
```

Response:

```

1. {
2.   "jsonrpc": "2.0",
3.   "result": [
4.     {
5.       "itemid": "23296",
6.       "clock": "1351090996",
7.       "value": "0.0850",
8.       "ns": "563157632"
9.     },
10.    {
```

```
11.         "itemid": "23296",
12.         "clock": "1351090936",
13.         "value": "0.1600",
14.         "ns": "549216402"
15.     },
16.     {
17.         "itemid": "23296",
18.         "clock": "1351090876",
19.         "value": "0.1800",
20.         "ns": "537418114"
21.     },
22.     {
23.         "itemid": "23296",
24.         "clock": "1351090816",
25.         "value": "0.2100",
26.         "ns": "522659528"
27.     },
28.     {
29.         "itemid": "23296",
30.         "clock": "1351090756",
31.         "value": "0.2150",
32.         "ns": "507809457"
33.     },
34.     {
35.         "itemid": "23296",
36.         "clock": "1351090696",
37.         "value": "0.2550",
38.         "ns": "495509699"
39.     },
40.     {
41.         "itemid": "23296",
42.         "clock": "1351090636",
43.         "value": "0.3600",
44.         "ns": "477708209"
45.     },
46.     {
47.         "itemid": "23296",
48.         "clock": "1351090576",
49.         "value": "0.3750",
50.         "ns": "463251343"
51.     },
52.     {
53.         "itemid": "23296",
54.         "clock": "1351090516",
55.         "value": "0.3150",
56.         "ns": "447947017"
57.     },
58.     {
59.         "itemid": "23296",
60.         "clock": "1351090456",
61.         "value": "0.2750",
62.         "ns": "435307141"
63.     }
```

history.get

```
64.     ],
65.     "id": 1
66. }
```

Source来源

CHistory::get() in *frontends/php/include/classes/api/services/CHistory.php*.

发现主机

This class is designed to work with discovered hosts.

该类被用来处理主机发现

Object references:

对象引用:

- [Discovered host](#)

Available methods:

可用方法:

- [dhost.get](#) - 获取已发现主机

> 主机发现对象

以下是主机发现API相关的对象

已发现主机

已发现主机是被Zabbix Server创建,且不能通过API进行修改.

主机发现对象包含通过网络发现规则发现的主机信息，有如下属性：

属性	类型	描述
dhostid	string	已发现主机的ID.
druleid	string	探测到主机的发现规则的ID.
lastdown	timestamp	已发现主机最近一次宕机的时间.
lastup	timestamp	已发现主机最近一次恢复的时间.
status	integer	已发现主机是否宕机的状态. 如果至少有一个存活的服务发现, 则主机是存活的. 可能的值: 0 - up; 1 - down.

dhost.get

描述

整数/数组 dhost.get(对象 参数)

该方法允许根据给定的参数检索已发现的主机

参数

(对象) 定义所需输出的参数

该方法支持以下参数：

参数	类型	描述
dhostids	string/array	只返回给定ID的已发现主机.
druleids	string/array	只返回给定发现规则(ID)创建的已发现主机.
dserviceids	string/array	只返回正在运行给定服务(ID)的已发现主机.
selectDRules	query	将探测到主机的发现规则作为 drules 属性中的数组返回.
selectDServices	query	将主机上已发现的服务作为 dservices 属性返回. 支持 count .
limitSelects	integer	限制子选项返回的记录数. 适用于以下子选项： selectDServices - 结果按照 dserviceid .
sortfield	string/array	按照给定的属性对结果进行排序. 可能的值： dhostid 和 druleid .
countOutput	flag	下面这些参数的 get 方法在 reference commentary 里面有详细说明.
editable	boolean	
excludeSearch	flag	
filter	object	
limit	integer	
output	query	
preservekeys	flag	
search	object	
searchByAny	boolean	
searchWildcardsEnabled	boolean	
sortorder	string/array	
startSearch	flag	

返回值

(integer/array) 返回:

- 对象数组;
- 如果使用了 `countOutput` 参数, 将返回对象的计数.

示例

获取通过发现规则发现的主机

获取所有被ID为4的发现规则发现的所有主机及其上正在运行的服务

请求:

```

1. {
2.   "jsonrpc": "2.0",
3.   "method": "dhost.get",
4.   "params": {
5.     "output": "extend",
6.     "selectDServices": "extend",
7.     "druleids": "4"
8.   },
9.   "auth": "038e1d7b1735c6a5436ee9eae095879e",
10.  "id": 1
11. }
```

响应:

```

1. {
2.   "jsonrpc": "2.0",
3.   "result": [
4.     {
5.       "dservices": [
6.         {
7.           "dserviceid": "1",
8.           "dhostid": "1",
9.           "type": "4",
10.          "key_": "",
11.          "value": "",
12.          "port": "80",
13.          "status": "0",
14.          "lastup": "1337697227",
15.          "lastdown": "0",
16.          "dcheckid": "5",
17.          "ip": "192.168.1.1",
18.          "dns": "station.company.lan"
19.        }
20.      ],
21.      "dhostid": "1",
22.      "druleid": "4",
23.      "status": "0",
24.    }
25.  ]
26. }
```

```
24.         "lastup": "1337697227",
25.         "lastdown": "0"
26.     },
27.     {
28.         "dservices": [
29.             {
30.                 "dserviceid": "2",
31.                 "dhostid": "2",
32.                 "type": "4",
33.                 "key_": "",
34.                 "value": "",
35.                 "port": "80",
36.                 "status": "0",
37.                 "lastup": "1337697234",
38.                 "lastdown": "0",
39.                 "dcheckid": "5",
40.                 "ip": "192.168.1.4",
41.                 "dns": "john.company.lan"
42.             }
43.         ],
44.         "dhostid": "2",
45.         "druleid": "4",
46.         "status": "0",
47.         "lastup": "1337697234",
48.         "lastdown": "0"
49.     },
50.     {
51.         "dservices": [
52.             {
53.                 "dserviceid": "3",
54.                 "dhostid": "3",
55.                 "type": "4",
56.                 "key_": "",
57.                 "value": "",
58.                 "port": "80",
59.                 "status": "0",
60.                 "lastup": "1337697234",
61.                 "lastdown": "0",
62.                 "dcheckid": "5",
63.                 "ip": "192.168.1.26",
64.                 "dns": "printer.company.lan"
65.             }
66.         ],
67.         "dhostid": "3",
68.         "druleid": "4",
69.         "status": "0",
70.         "lastup": "1337697234",
71.         "lastdown": "0"
72.     },
73.     {
74.         "dservices": [
75.             {
76.                 "dserviceid": "4",
```

```
77.         "dhostid": "4",
78.         "type": "4",
79.         "key_": "",
80.         "value": "",
81.         "port": "80",
82.         "status": "0",
83.         "lastup": "1337697234",
84.         "lastdown": "0",
85.         "dcheckid": "5",
86.         "ip": "192.168.1.7",
87.         "dns": "mail.company.lan"
88.     },
89. ],
90.     "dhostid": "4",
91.     "druleid": "4",
92.     "status": "0",
93.     "lastup": "1337697234",
94.     "lastdown": "0"
95.   },
96. ],
97.   "id": 1
98. }
```

参考

- [Discovered service](#)
- [Discovery rule](#)

来源

`CDHost::get()` in *frontends/php/include/classes/api/services/CDHost.php*.

发现服务

该类被用来处理服务发现.

对象引用:

- [Discovered service](#)

可用方法:

- [dservice.get](#) - 获取已发现的服务

> 服务发现对象

以下是服务发现API相关的对象 .

已发现服务

已发现服务是被Zabbix Server创建，且不能通过API进行修改 .

服务发现对象包含通过主机上的网络发现规则发现的服务信息，有如下属性：

属性	类型	描述
dserviceid	string	已发现服务的ID .
dcheckid	string	用于探测服务的发现检查项ID .
dhostid	string	运行服务的被发现主机的ID .
dns	string	运行服务主机的DNS .
ip	string	运行服务主机的IP .
lastdown	timestamp	已发现服务最近一次down掉的时间 .
lastup	timestamp	已发现服务最近一次存活的时间 .
port	integer	服务的端口号 .
status	integer	服务状态. 可能的值：0 - up; 1 - down .
value	string	当使用Zabbix Agent, SNMPv1, SNMPv2, SNMPv3做发现检查时, 服务的返回值 .

dservice.get

描述

`整数/数组 dservice.get(对象 参数)`

The method allows to retrieve discovered services according to the given parameters.

该方法允许根据给定的参数检索已发现的服务

参数

`(object) Parameters defining the desired output.`

(对象) 定义所需输出的参数

The method supports the following parameters.

该方法支持以下参数

参数	类型	描述
dserviceids	string/array	Return only discovered services with the given IDs. 只返回指定ID的已发现服务
dhostids	string/array	Return only discovered services that belong to the given discovered hosts. 只返回指定已发现主机的发现服务
dcheckids	string/array	Return only discovered services that have been detected by the given discovery checks. 只返回指定检查ID探测到的服务
druleids	string/array	Return only discovered services that have been detected by the given discovery rules. 只返回给指定现规则ID探测到的服务
selectDRules	query	Return the discovery rule that detected the service as an array in the <code>drules</code> property. 将探测到服务的发现规则作为 <code>drules</code> 属性中的数组返回
selectDHosts	query	Return the discovered host that service belongs to as an array in the <code>dhosts</code> property. 将服务归属的已发现主机作为 <code>dhosts</code> 属性中的数组返回
selectHosts	query	Return the hosts with the same IP address as the service in the <code>hosts</code> property. Supports <code>count</code> .
limitSelects	integer	Limits the number of records returned by subselects. 限制子选项返回的记录数 Applies to the following subselects: 子选项的应用 <code>selectHosts</code> - result will be sorted by <code>hostid</code> . 结果按照 <code>hostid</code> 排序

sortfield	string/array	Sort the result by the given properties. 结果按照指定属性排序 可能的值: <code>dserviceid</code> , <code>dhostid</code> 和 <code>ip</code> .
countOutput	flag	下面这些参数对于所有get方法都是通用的, 详情可参考 reference commentary .
editable	boolean	
excludeSearch	flag	
filter	object	
limit	integer	
output	query	
preservekeys	flag	
search	object	
searchByAny	boolean	
searchWildcardsEnabled	boolean	
sortorder	string/array	
startSearch	flag	

返回值

返回 `整数/数组` :

- 返回一个对象数组;
- 如果 `countOutput` 被使用, 返回检索对象的计数.

示例

Retrieve services discovered on a host

检索一台主机上已发现的服务

Retrieve all discovered services detected on discovered host "11". 获取在已发现主机 `11` 上探测到发现的服务

请求:

```

1. {
2.   "jsonrpc": "2.0",
3.   "method": "dservice.get",
4.   "params": {
5.     "output": "extend",
6.     "dhostids": "11"
7.   },
8.   "auth": "038e1d7b1735c6a5436ee9eae095879e",
9.   "id": 1
10. }
```

响应：

```

1.  {
2.      "jsonrpc": "2.0",
3.      "result": [
4.          {
5.              "dserviceid": "12",
6.              "dhostid": "11",
7.              "value": "",
8.              "port": "80",
9.              "status": "1",
10.             "lastup": "0",
11.             "lastdown": "1348650607",
12.             "dcheckid": "5",
13.             "ip": "192.168.1.134",
14.             "dns": "john.local"
15.         },
16.         {
17.             "dserviceid": "13",
18.             "dhostid": "11",
19.             "value": "",
20.             "port": "21",
21.             "status": "1",
22.             "lastup": "0",
23.             "lastdown": "1348650610",
24.             "dcheckid": "6",
25.             "ip": "192.168.1.134",
26.             "dns": "john.local"
27.         }
28.     ],
29.     "id": 1
30. }
```

参考

- [Discovered host](#)
- [Discovery check](#)
- [Host](#)

来源

`CDSERVICE::get()` in *frontends/php/include/classes/api/services/CDSERVICE.php*.

发现检查

该类被用来处理发现检查 .

对象引用:

- [Discovery check](#)

可用方法:

- `dcheck.get` - retrieve discovery checks

> Discovery check object

> 发现检查对象

The following objects are directly related to the `dcheck` API.

以下是发现检查API相关的对象

Discovery check

发现检查

The discovery check object defines a specific check performed by a network discovery rule. It has the following properties.

发现检查对象定义了通过网络发现规则执行特定的检查。它有如下属性：

属性	类型	描述
<code>dcheckid</code>	<code>string</code>	(只读) ID of the discovery check. 发现检查的ID。
<code>druleid</code>	<code>string</code>	ID of the discovery rule that the check belongs to. 发现检查所属的发现规则的ID。
<code>key</code>	<code>string</code>	The value of this property differs depending on the type type of the check: 这个属性值会根据不同的检查类型而不同 - key to query for Zabbix agent checks, required; - SNMP OID for SNMPv1, SNMPv2 and SNMPv3 checks, required.
<code>ports</code>	<code>string</code>	One or several port ranges to check separated by commas. Used for all checks except for ICMP. Default: 0.
<code>snmp_community</code>	<code>string</code>	SNMP community. Required for SNMPv1 and SNMPv2 agent checks.
<code>snmpv3_authpassphrase</code>	<code>string</code>	Auth passphrase used for SNMPv3 agent checks with security level set to <code>_authNoPriv</code> or <code>authPriv</code> .
<code>snmpv3authprotocol</code>	<code>integer</code>	Authentication protocol used for SNMPv3 agent checks with security level set to <code>_authNoPriv</code> or <code>authPriv</code> . Possible values: 0 - (default) MD5; 1 - SHA.
<code>snmpv3contextname</code>	<code>string</code>	SNMPv3 context name. Used only by SNMPv3 checks.
<code>snmpv3_privpassphrase</code>	<code>string</code>	Priv passphrase used for SNMPv3 agent checks with security level set to <code>_authPriv</code> .
<code>snmpv3privprotocol</code>	<code>integer</code>	Privacy protocol used for SNMPv3 agent checks with security level set to <code>_authPriv</code> . Possible values: 0 - (default) DES; 1 - AES.
<code>snmpv3securitylevel</code>	<code>string</code>	Security level used for SNMPv3 agent checks. Possible values: 0 - <code>noAuthNoPriv</code> ; 1 - <code>authNoPriv</code> ; 2 - <code>authPriv</code> .

snmpv3_securityname	string	Security name used for SNMPv3 agent checks.
type	integer	Type of check. Possible values: 0 - (default) <i>SSH</i> ; 1 - <i>LDAP</i> ; 2 - <i>SMTP</i> ; 3 - <i>FTP</i> ; 4 - <i>HTTP</i> ; 5 - <i>POP</i> ; 6 - <i>NNTP</i> ; 7 - <i>IMAP</i> ; 8 - <i>TCP</i> ; 9 - Zabbix agent; 10 - SNMPv1 agent; 11 - SNMPv2 agent; 12 - ICMP ping; 13 - SNMPv3 agent; 14 - HTTPS; 15 - Telnet.
uniq	integer	Whether to use this check as a device uniqueness criteria. Only a single unique check can be configured for a discovery rule. Used for Zabbix agent, SNMPv1, SNMPv2 and SNMPv3 agent checks. Possible values: 0 - (default) do not use this check as a uniqueness criteria; 1 - use this check as a uniqueness criteria.

dcheck.get

描述

```
integer/array dcheck.get(object parameters)
```

The method allows to retrieve discovery checks according to the given parameters.

该方法可以根据给定的参数检索发现检查

参数

(object) 定义希望输出的参数 .

该方法支持如下参数 .

参数	类型	描述
dcheckids	string/array	Return only discovery checks with the given IDs. 返回给定ID的发现检查
druleids	string/array	Return only discovery checks that belong to the given discovery rules. 返回属于给定发现规则的发现检查
dserviceids	string/array	Return only discovery checks that have detected the given discovered services. 返回探测到发现服务的发现检查
sortfield	string/array	Sort the result by the given properties. 根据给定的属性排序. 可能的值: <code>dcheckid</code> 和 <code>druleid</code> .
countOutput	flag	These parameters being common for all <code>get</code> methods are described in detail in the reference commentary . 下面这些参数的get方法在 reference commentary 里面有详细说明
editable	boolean	
excludeSearch	flag	
filter	object	
limit	integer	
output	query	
preservekeys	flag	
search	object	
searchByAny	boolean	
searchWildcardsEnabled	boolean	
sortorder	string/array	
startSearch	flag	

返回值

(integer/array) 返回:

- 对象数组;
- 如果使用了countOutput参数, 将返回对象的计数.

例子

Retrieve discovery checks for a discovery rule 获取发现规则的发现检查

Retrieve all discovery checks used by discovery rule "6".

获取被发现规则“6”使用的发现检查 .

请求:

```

1. {
2.   "jsonrpc": "2.0",
3.   "method": "dcheck.get",
4.   "params": {
5.     "output": "extend",
6.     "dcheckids": "6"
7.   },
8.   "auth": "038e1d7b1735c6a5436ee9eae095879e",
9.   "id": 1
10. }
```

响应:

```

1. {
2.   "jsonrpc": "2.0",
3.   "result": [
4.     {
5.       "dcheckid": "6",
6.       "druleid": "4",
7.       "type": "3",
8.       "key_": "",
9.       "snmp_community": "",
10.      "ports": "21",
11.      "snmpv3_securityname": "",
12.      "snmpv3_securitylevel": "0",
13.      "snmpv3_authpassphrase": "",
14.      "snmpv3_privpassphrase": "",
15.      "uniq": "0",
16.      "snmpv3_authprotocol": "0",
17.      "snmpv3_privprotocol": "0"
18.    }
]
```

```
19.     ],
20.     "id": 1
21. }
```

来源

CDCheck::get() in *frontends/php/include/classes/api/services/CDCheck.php*.

发现规则

该类被用来处理发现规则

这个API是用来处理网络发现规则.对于低级发现规则可以参考[LLD rule API](#).

对象引用:

- [Discovery rule](#)

可用方法:

- [drule.create](#) - 创建发现规则
- [drule.delete](#) - 删除发现规则
- [drule.get](#) - 获取发现规则
- [drule.update](#) - 更新发现规则

> Discovery rule object 发现规则对象

以下是和 `drule` API相关的对象

发现规则

发现规则对象定义了一个网络发现规则. 它有如下属性:

属性	类型	描述
<code>druleid</code>	<code>string</code>	(只读) 发现规则的ID
<code>iprange(required)</code>	<code>string</code>	一个或多个要检查的IP范围, 逗号分隔对于更多关于IP范围格式可以参考 network discovery configuration
<code>name(required)</code>	<code>string</code>	发现规则名称
<code>delay</code>	<code>string</code>	执行发现规则的时间间隔. 使用带有时间后缀或者宏作为时间单位, 可以使用秒, 默认1h
<code>nextcheck</code>	<code>timestamp</code>	(只读) 发现规则下一次执行的时间
<code>proxyhostid</code>	<code>string</code>	用于发现的proxy的ID
<code>status</code>	<code>integer</code>	发现规则是否启用 可能的值: 0 - (默认)启用; 1 - 停用.

drule.create

描述

```
object drule.create(object/array discoveryRules)
```

该方法允许创建新的发现规则.

参数

(对象/数组) 要创建的发现规则.

除了**standard discovery rule properties**外, 这个方法还接受如下参数.

参数	类型	描述
dchecks (必选)	array	用来创建发现规则的发现检查

返回值

(object) 在 **druleids** 属性下返回已创建的发现规则的ID对象. 返回的ID的顺序与意见通过的发现规则顺序保持一致

例子

创建发现规则

创建一个发现规则, 用于发现在本地网络中运行Zabbix Agent的主机. 规则必须用一个运行在10050端口上的Zabbix代理检查

请求:

```

1. {
2.     "jsonrpc": "2.0",
3.     "method": "drule.create",
4.     "params": {
5.         "name": "Zabbix agent discovery",
6.         "iprange": "192.168.1.1-255",
7.         "dchecks": [
8.             {
9.                 "type": "9",
10.                "key_": "system.uname",
11.                "ports": "10050",
12.                "uniq": "0"
13.            }
14.        ],
15.    },
16.    "auth": "038e1d7b1735c6a5436ee9eae095879e",

```

```
17.      "id": 1
18. }
```

响应：

```
1. {
2.   "jsonrpc": "2.0",
3.   "result": {
4.     "druleids": [
5.       "6"
6.     ]
7.   },
8.   "id": 1
9. }
```

参考

- [Discovery check](#)

来源

`CDRule::create()` in *frontends/php/include/classes/api/services/CDRule.php*.

drule.delete

描述

```
object drule.delete(array discoveryRuleIds)
```

该方法允许删除发现规则.

参数

(array) 要删除发现规则的ID.

返回值

(object) 在 druleids 属性下返回包含删除的发现规则的ID对象.

例子

删除多个发现规则

删除两个发现规则.

请求:

```
1. {
2.     "jsonrpc": "2.0",
3.     "method": "drule.delete",
4.     "params": [
5.         "4",
6.         "6"
7.     ],
8.     "auth": "3a57200802b24cda67c4e4010b50c065",
9.     "id": 1
10. }
```

响应:

```
1. {
2.     "jsonrpc": "2.0",
3.     "result": {
4.         "druleids": [
5.             "4",
6.             "6"
7.         ]
8.     },
9.     "id": 1
10. }
```

来源

`CDRule::delete() in frontends/php/include/classes/api/services/CDRule.php.`

drule.get

描述

```
integer/array drule.get(object parameters)
```

该方法允许根据给定的参数检索发现规则.

参数

(object) 定义希望输出的参数 .

该方法支持以下参数.

参数	类型	描述
dhostids	string/array	返回给定发现主机创建的发现规则
druleids	string/array	返回给定ID的发现规则
dserviceids	string/array	返回给定发现服务创建的发现规则
selectDChecks	query	属性下返回被发现规则使用的发现检查 支持 <code>count</code> .
selectDHosts	query	在 <code>dhosts</code> 属性下返回发现规则创建的发现主机支持 <code>count</code> .
limitSelects	integer	子选项返回的记录数限制. \适用于如下子选项： <code>selectDChecks</code> - 结果按 <code>dcheckid</code> 排序; <code>selectDHosts</code> - 结果按 <code>dhostsid</code> 排序.
sortfield	string/array	根据指定的属性字段进行排序. 可能的值: <code>druleid</code> and <code>name</code> .
countOutput	flag	下面这些参数的get方法在 reference commentary 里面有详细说明.
editable	boolean	
excludeSearch	flag	
filter	object	
limit	integer	
output	query	
preservekeys	flag	
search	object	
searchByAny	boolean	
searchWildcardsEnabled	boolean	
sortorder	string/array	
startSearch	flag	

返回值

(integer/array) 返回：

- 对象数组；
- 如果使用了countOutput参数, 将返回对象的计数.

例子

检索所有的发现规则

获取已经配置的发现规则及它们所使用的发现检查 .

请求：

```

1. {
2.     "jsonrpc": "2.0",
3.     "method": "drule.get",
4.     "params": {
5.         "output": "extend",
6.         "selectDChecks": "extend"
7.     },
8.     "auth": "038e1d7b1735c6a5436ee9eae095879e",
9.     "id": 1
10. }
```

响应：

```

1. {
2.     "jsonrpc": "2.0",
3.     "result": [
4.         {
5.             "druleid": "2",
6.             "proxy_hostid": "0",
7.             "name": "Local network",
8.             "iprange": "192.168.3.1-255",
9.             "delay": "5s",
10.            "nextcheck": "1348754327",
11.            "status": "0",
12.            "dchecks": [
13.                {
14.                    "dcheckid": "7",
15.                    "druleid": "2",
16.                    "type": "3",
17.                    "key_": "",
18.                    "snmp_community": "",
19.                    "ports": "21",
20.                    "snmpv3_securityname": "",
21.                    "snmpv3_securitylevel": "0",
22.                    "snmpv3_authpassphrase": ""}
```

```
23.         "snmpv3_privpassphrase": "",  
24.         "uniq": "0",  
25.         "snmpv3_authprotocol": "0",  
26.         "snmpv3_privprotocol": "0"  
27.     },  
28.     {  
29.         "dcheckid": "8",  
30.         "druleid": "2",  
31.         "type": "4",  
32.         "key_": "",  
33.         "snmp_community": "",  
34.         "ports": "80",  
35.         "snmpv3_securityname": "",  
36.         "snmpv3_securitylevel": "0",  
37.         "snmpv3_authpassphrase": "",  
38.         "snmpv3_privpassphrase": "",  
39.         "uniq": "0",  
40.         "snmpv3_authprotocol": "0",  
41.         "snmpv3_privprotocol": "0"  
42.     }  
43.   ]  
44. },  
45. {  
46.     "druleid": "6",  
47.     "proxy_hostid": "0",  
48.     "name": "Zabbix agent discovery",  
49.     "iprange": "192.168.1.1-255",  
50.     "delay": "1h",  
51.     "nextcheck": "0",  
52.     "status": "0",  
53.     "dchecks": [  
54.       {  
55.           "dcheckid": "10",  
56.           "druleid": "6",  
57.           "type": "9",  
58.           "key_": "system.uname",  
59.           "snmp_community": "",  
60.           "ports": "10050",  
61.           "snmpv3_securityname": "",  
62.           "snmpv3_securitylevel": "0",  
63.           "snmpv3_authpassphrase": "",  
64.           "snmpv3_privpassphrase": "",  
65.           "uniq": "0",  
66.           "snmpv3_authprotocol": "0",  
67.           "snmpv3_privprotocol": "0"  
68.       }  
69.     ]  
70.   },  
71.   ],  
72.   "id": 1  
73. }
```

参考

- [Discovered host](#)
- [Discovery check](#)

来源

`CDRule::get()` in *frontends/php/include/classes/api/services/CDRule.php*.

drule.update

描述

```
object drule.update(object/array discoveryRules)
```

该方法允许更新已存在的发现规则.

参数

(object/array) 要更新的发现规则的属性.

必须为每条发现规则定义 `druleid` 属性, 其它属性是可选的. 只有传参进去的属性才会被更新, 其它属性不变.

除了`standard discovery rule properties`外, 该方法还接受如下参数:

属性	类型	描述
dchecks	array	替代已存在的发现检查.

返回值

(object) 在 `druleids` 属性下返回包含更新的发现规则的ID对象.

例子

改变发现规则的IP范围

将发现规则的IP范围更改为192.168.2.1-255.

请求:

```

1. {
2.   "jsonrpc": "2.0",
3.   "method": "drule.update",
4.   "params": {
5.     "druleid": "6",
6.     "iprange": "192.168.2.1-255"
7.   },
8.   "auth": "038e1d7b1735c6a5436ee9eae095879e",
9.   "id": 1
10. }
```

响应:

```

1. {
2.   "jsonrpc": "2.0",
```

```
3.     "result": {
4.         "druleids": [
5.             "6"
6.         ]
7.     },
8.     "id": 1
9. }
```

参考

- [Discovery check](#)

来源

`CDRule::update()` in *frontends/php/include/classes/api/services/CDRule.php*.

告警

这个对象用于告警模块。

对象引用：

- [Alert](#)

相关方法：

- [alert.get](#) - 获取告警

> Alert object

The following objects are directly related to the [alert API](#). 以下是alert API的使用方法

Alert

Alerts are created by the Zabbix server and cannot be modified via the [API](#).

Alerts是由Zabbix server创建，无法通过API修改。

The alert object contains information about whether certain action operations have been executed successfully. It has the following properties. alert对象包含有关某些action操作是否已成功执行的信息，它具有以下特性。

Property	Type	Description
特性	类型	描述
alertid	string	ID of the alert.
alertid	string	Alert ID值.
actionid	string	ID of the action that generated the alert.
actionid	string	Alert生成的Action ID.
alerttype	integer	Alert type. Possible values: 0 - message; 1 - remote command.
alerttype	integer	Alert类型. 可用值: 0 - message; 1 - remote command.
clock	timestamp	Time when the alert was generated.
clock	timestamp	Alert生成的时间.
error	string	Error text if there are problems sending a message or running a command.
error	string	Alert发送Message或者执行一个命令产生的报错信息.
esc_step	integer	Action escalation step during which the alert was generated.
esc_step	integer	生成Alert后Action的处理步骤.
eventid	string	ID of the event that triggered the action.
eventid	string	触发Action的事件ID.
mediatypeid	string	ID of the media type that was used to send the message.
报警媒介类型的id	string	用于发送消息的报警媒介类型的ID.
message	text	Message text. Used for message alerts.
message	text	消息文本. 用于消息告警.
retries	integer	Number of times Zabbix tried to send the message.
retries	integer	Zabbix尝试发送消息的次数.
sendto	string	Address, user name or other identifier of the recipient.

sendto	string	Used for message alerts.
sendto	string	地址, 用户名或接收者的其他标识符. 用于消息告警.
status	integer	Status indicating whether the action operation has been executed successfully. Possible values for message alerts: 0 - message not sent; 1 - message sent; 2 - failed after a number of retries. Possible values for command alerts: 1 - command run; 2 - tried to run the command on the Zabbix agent but it was unavailable.
status	integer	指示action操作是否已被成功执行的状态. 消息告警的可能值: 0 - 消息未发送; 1 - 消息已发送; 2 - 经多次重试后失败. 命令告警的可能值: 1 - 命令运行; 2 - 尝试在Zabbix agent上运行命令但不可用.
subject	string	Message subject. Used for message alerts.
subject	string	消息主题. 用于消息告警.
userid	string	ID of the user that the message was sent to.
userid	string	发送消息的用户ID.

alert.get

Description

描述

`integer/array alert.get(object parameters)` 整数/数组 `alert.get(对象 参数)`

The method allows to retrieve alerts according to the given parameters. 该方式允许根据给定的参数检索警报。

Parameters

参数

`(object) Parameters defining the desired output.` `(object)` 定义所需输出的参数。

The method supports the following parameters. 该方法支持以下参数。

Parameter	Type	Description
参数	类型	描述
alertids	string/array	Return only alerts with the given IDs.
alertids	string/array	只返回给定ID的alerts。
actionids	string/array	Return only alerts generated by the given actions.
actionids	string/array	只返回给定actions生成的alerts。
eventids	string/array	Return only alerts generated by the given events.
eventids	string/array	只返回给定事件生成的alerts。
groupids	string/array	Return only alerts generated by objects from the given host groups.
groupids	string/array	只返回来自给定主机组的对象生成的alerts。
hostids	string/array	Return only alerts generated by objects from the given hosts.
hostids	string/array	只返回来自给定主机的对象生成的alerts。
mediatypeids	string/array	Return only message alerts that used the given media types.
mediatypeids	string/array	只返回用于给定报警媒介类型的消息警报。
objectids	string/array	Return only alerts generated by the given objects
objectids	string/array	只返回给定对象生成的alerts
		Return only message alerts that were sent

		to the given users.
userids	string/array	只返回发送给给定用户的消息警报.
eventobject	integer	Return only alerts generated by events related to objects of the given type. Refer to the event "object" property for a list of supported object types. Default: 0 - trigger.
eventsource	integer	Return only alerts generated by events of the given type. Refer to the event "source" property for a list of supported event types. Default: 0 - trigger events.
time_from	timestamp	Return only alerts that have been generated after the given time.
time_till	timestamp	Return only alerts that have been generated before the given time.
事件对象	整数	只返回与给定类型的对象相关的事件生成的警报. 有关支持对象类型的列表, 请参阅 event "object" property . 默认值: 0 - 触发器.
事件来源	整数	只返回给定类型的事件生成的警报. 有关支持事件类型的列表, 请参阅 event "source" property . 默认值: 0 - 触发事件.
time_from	时间戳	只返回在给定时间之后生成的警报.
time_till	时间戳	只返回在给定时间之前生成的警报.
selectHosts	query	Return the hosts that triggered the action operation in the <code>hosts</code> property.
selectHosts	query	在“hosts”属性中返回触发action操作的主机.
selectMediatypes	query	Return the media type that was used for the message alert as an array in the <code>mediatypes</code> property.
selectMediatypes	query	将用于消息警报的报警媒介类型作为“mediatypes”属性中的数组返回.
selectUsers	query	Return the user that the message was addressed to as an array in the <code>users</code> property.
selectUsers	query	将消息被寻址的用户作为“users”属性中的数组返回.
sortfield	string/array	Sort the result by the given properties. Possible values are: <code>alertid</code> , <code>clock</code> , <code>eventid</code> and <code>status</code> .
sortfield	string/array	按给定属性的结果进行排序. \可能的值为: <code>alertid</code> , <code>clock</code> , <code>eventid</code> 还有 <code>status</code> .
countOutput	flag	These parameters being common for all <code>get</code> methods are described in the reference commentary .
countOutput	flag	在 reference commentary 中描述了所有“get”方法的共同参数.
editable	boolean	

editable	boolean
excludeSearch	flag
filter	object
limit	integer
output	query
preservekeys	flag
search	object
searchByAny	boolean
searchWildcardsEnabled	boolean
sortorder	string/array
startSearch	flag

Return values

返回值

(integer/array) Returns either:

- an array of objects;
- 一组对象；
- the count of retrieved objects, if the `countOutput` parameter has been used.
- 如果已经使用了“countOutput”参数，则检索对象的计数.

范例

Retrieve alerts by action ID

通过动作ID检索警报

Retrieve all alerts generated by action “3”.

Request:

```

1. {
2.   "jsonrpc": "2.0",
3.   "method": "alert.get",
4.   "params": {
5.     "output": "extend",
6.     "actionids": "3"
7.   },
8.   "auth": "038e1d7b1735c6a5436ee9eae095879e",
9.   "id": 1
10. }
```

Response:

```

1.  {
2.      "jsonrpc": "2.0",
3.      "result": [
4.          {
5.              "alertid": "1",
6.              "actionid": "3",
7.              "eventid": "21243",
8.              "userid": "1",
9.              "clock": "1362128008",
10.             "mediatypeid": "1",
11.             "sendto": "[email protected]",
12.             "subject": "PROBLEM: Zabbix agent on Linux server is unreachable for 5 minutes: ",
13.             "message": "Trigger: Zabbix agent on Linux server is unreachable for 5 minutes: \nTrigger status: PROBLEM\nTrigger severity: Not classified",
14.             "status": "0",
15.             "retries": "3",
16.             "error": "",
17.             "esc_step": "1",
18.             "alerttype": "0"
19.         }
20.     ],
21.     "id": 1
22. }
```

See also

参见

- [Host](#)
- [Media type](#)
- [User](#)

Source

来源

`CAlert::get()` in *frontends/php/include/classes/api/services/CAlert.php*.

图像

此类用于管理图像。

对象引用：

- [Image](#)

可用的方法：

- [image.create](#) - 创建新图像
- [image.delete](#) - 删除图像
- [image.get](#) - 获取图像
- [image.update](#) - 更新图像

> Image object 图像对象

The following objects are directly related to the [image API](#). 以下对象与“image”API直接相关。

Image 图像

The image object has the following properties. 图像对象具有以下属性。

属性	类型	说明
imageid	string	(readonly) ID of the image. 图像ID。
name (required)	string	Name of the image. 图像名称
imagetype	integer	Type of image. Possible values: 可能的值: 1 - (default默认) icon;图标 2 - background image. 背景图。

image.create

Description 说明

`object image.create(object/array images)`

This method allows to create new images. 该方法允许创建新的图像。

Parameters 参数

`(object/array) Images to create.` 要创建的图像

Additionally to the [standard image properties](#), the method accepts the following parameters. 除了标准图像属性外，该方法接受以下参数。

属性	类型	说明
<code>image(required)</code>	<code>string</code>	Base64 encoded image. The maximum size of the encoded image is 1 MB. Base64编码图像。 编码图像的最大大小为1 MB。

Return values 返回值

`(object)` Returns an object containing the IDs of the created images under the `imageids` property. The order of the returned IDs matches the order of the passed images. `(object)` 返回一个包含“imageid”属性下创建的图像ID的对象。返回的ID的顺序与传递的图像的顺序相匹配。

Examples 示例

Create an image 创建图像

Create a cloud icon. 创建一个云图标。

Request:

```

1. {
2.   "jsonrpc": "2.0",
3.   "method": "image.create",
4.   "params": {
5.     "imagetype": 1,
6.     "name": "Cloud_(24)",
7.     "image":
"iVBORw0KGgoAAAANSUhEUgAAABgAAAANCAYAACzBk7QAAAABHNCVQICAgIfAhkiAAAAAlwSF1zAAACmAAAAPgBNtNH3wAAAB10RVh0U29mdHdcmUAAd3d3Lmlua3NjYXB1Lm9yZ5vuPBoAAAIcSURVDjLrZLbSxRRHMDPKiEiRQ89CD0s+N5j9BIMEf4Hg/jWexD2ZEXQbC9tWUFZimtLhsuZiVujk1UjmYXw9PaCUdtb83enL3P7s6ss5f5dc7EUsmqkPuFH3M4/0b7+V00AgC0UyDENFEU03rh1uN0s/1FG75o2i2/rkd9Y3Tgyj3Hiaezbukdh9A/rP4E9vWi0u+Y4fuGnMf3DRgYc3Z/84YrQSkD3mgKhFAC+KAEK74Y2Lj3MjPo0okQ3Xyx/1GHeXCifbf061RPH/wi+AvZQhGSsgKxdB5CCRKGpDgMXBMBukTc4vK5/wRHizsq7fZ12LFuvE4T0BZDTXHtgv4TNuqlUolsqQL2qQwbDEXzBBTIJ7I4y/cfAENmHZF4XrY9Mc+X9HAFmoyXS2ddy1I0g6/KNyBcM0DFP/wFZFCCoY4N9Mw0YKCT0fhdl5AfZQXQBFn2t/0DXHC8FYVcoWjNEQ03qqwTJ5FdI44jg/ms0B2Zd5ZKq3q6evA1FUS60bYyyj3AJf3V72HiLZJQxTtRLk1C2IYEg4mTNg63hPd1m0Jd7Ict9110MN1WEf0nFxpCt16zcshTuLpGSwDDuPiFv0xzNyQYVGicC0cgUUd

```

```
LM6Xp021vvW/V2EBssnx1SGmWsxljw0znV9XfPLjTCW84r+cn7Jc8c2eWrbM6Wbe6/aTJbhJ/TNkWc9/xXW592Xb9iPkKnUfH8BKdLgFy0lDyQ
AAAAASUVORK5CYII="
8.      },
9.      "auth": "038e1d7b1735c6a5436ee9eae095879e",
10.     "id": 1
11. }
```

Response:

```
1.  {
2.      "jsonrpc": "2.0",
3.      "result": {
4.          "imageids": [
5.              "188"
6.          ],
7.      },
8.      "id": 1
9. }
```

Source 来源

`CImage::create()` in *frontends/php/include/classes/api/services/CImage.php*.

image.delete

Description 说明

```
object image.delete(array imageIds)
```

This method allows to delete images.此方法允许删除图像。

Parameters 参数

(array) IDs of the images to delete.要删除的图像的ID。

Return values 返回值

(object) Returns an object containing the IDs of the deleted images under the imageids property.返回一个包含“imageid”属性下删除图像ID的对象。

Examples 示例

Delete multiple images 删除多个图像

Delete two images.删除两个图像

Request:

```
1. {
2.     "jsonrpc": "2.0",
3.     "method": "image.delete",
4.     "params": [
5.         "188",
6.         "192"
7.     ],
8.     "auth": "3a57200802b24cda67c4e4010b50c065",
9.     "id": 1
10. }
```

Response:

```
1. {
2.     "jsonrpc": "2.0",
3.     "result": {
4.         "imageids": [
5.             "188",
6.             "192"
7.         ]
8.     },
9.     "id": 1
}
```

image.delete

```
10. }
```

Source来源

`CImage::delete() in frontends/php/include/classes/api/services/CImage.php.`

image.get

Description 说明

```
integer/array image.get(object parameters)
```

The method allows to retrieve images according to the given parameters. 该方法允许根据给定的参数检索图像

Parameters 参数

(object) Parameters defining the desired output. 定义所需输出的参数。

The method supports the following parameters. 该方法支持以下参数。

属性	类型	说明
imageids	string/array	Return only images with the given IDs. 只返回具有给定ID的图像
sysmapids	string/array	Return images that are used on the given maps. 返回给定地图上使用的图像。
select_image	flag	Return the Base64 encoded image in the <code>image</code> property. 返回“image”属性中的Base64编码图像。
sortfield	string/array	Sort the result by the given properties. 按照给定的属性对结果进行排序 Possible values are: <code>imageid</code> and <code>name</code> . 可能的值为: <code>imageid</code> 和 <code>name</code> 。
countOutput	flag	These parameters being common for all <code>get</code> methods are described in detail in the reference commentary . 这些参数对于所有的“get”方法是常见的，在参考评论中有详细描述
editable	boolean	
excludeSearch	flag	
filter	object	
limit	integer	
output	query	
preservekeys	flag	
search	object	
searchByAny	boolean	
searchWildcardsEnabled	boolean	
sortorder	string/array	
startSearch	flag	

Return values 返回值

(integer/array) Returns either:返回:

- an array of objects;一组对象;
- the count of retrieved objects, if the `countOutput` parameter has been used.如果已经使用“countOutput”参数，则检索到的对象的计数。

Examples 示例

Retrieve an image 检索图像

Retrieve all data for image with ID “2”.检索ID为“2”的图像的所有数据。

Request:

```

1. {
2.   "jsonrpc": "2.0",
3.   "method": "image.get",
4.   "params": {
5.     "output": "extend",
6.     "select_image": true,
7.     "imageids": "2"
8.   },
9.   "auth": "038e1d7b1735c6a5436ee9eae095879e",
10.  "id": 1
11. }
```

Response:

```

1. {
2.   "jsonrpc": "2.0",
3.   "result": [
4.     {
5.       "imageid": "2",
6.       "imagetype": "1",
7.       "name": "Cloud_(24)",
8.       "image":
9.         "iVBORw0KGgoAAAANSUhEUgAAABgAAAANCAYAACzbK7QAAAABHNCVQICAgIfAhkiAAAAAlwSF1zAAACmAAAAPgBNtNH3wAAAB10RVh0U29md
10.      HdhcwUAAd3d3Lmlua3NjYXB1Lm9yZ5vuPB0AAAIcSURBVDjLrZLbSxRRHMdPKiEiRQ89CD0s+N5j9BIMEf4Hg/jWexDZEXQbC9tWUFZimtLhs
11.      uZiVujK1UJmYXW9PaCUdtb83enL3P7s6ss5f5dc7EUsmqkPuFH3M4/Ob7+V0OAgC0UyDENFEU03rh1uN0s/1FG75o2i2/rkd9Y3Tgyj3Hiae
12.      zukd9A/rP4E9vVi0u+Y4fuGnMf3DRgYc3Z/84YrQSkD3mgKhFAC+KAEK74Y2Lj3MjPo0okQ3xyx/1GHeXCifbf061RPH/wi+AvZQhGSsgKxdB5
13.      CCRkCGPbDgMXBMbukTc4vK5/WRHizsq7fZ12LFuvE4T0BZDTXHtgv4TNUqlUolsqQL2qQwbDEXzBBTIJ7I4y/cfAENmHZF4XrY9Mc+X9HAFmoy
14.      XS2ddy1I0g6/KNyBcM0DFP/wFZFCC0y4N9Mw0YkCT0fhL5AfZQXQBFn2t/0DXHC8FYVcoWjNEQ03qqwTJ5FdI44jg/msoB2Zd5ZKq3q6evA1F
15.      US60bYyyj3AJf3V72HiLZJQxTtRLK1C2IYEg4mTNg63hPd1m0Jd7Ict9110MN1wEf0nFxpCt16zcshTuLpGSwDDuPIfv0xzNyQYYVGicC0cgUUD
16.      LM6Xp021vvW/V2EBssnx1SGmWsxljw0znV9XfPLjTCW84r+cn7Jc8c2eWrbM6Wbe6/aTJbhJ/TNkWc9/xxW592Xb9iPkKnUfH8BKdLgFy01dyQ
17.      AAAASASUVORK5CYII="
18.    }
19.  ],
20.  "id": 1
21. }
```

image.get

Source 来源

`CImage::get() in frontends/php/include/classes/api/services/CImage.php.`

image.update

Description 说明

`(object) image.update(object/array images)`

This method allows to update existing images. 该方法允许更新现有的图像。

Parameters 参数

`(object/array)` Image properties to be updated. 要更新的图像属性。

The `imageid` property must be defined for each image, all other properties are optional. Only the passed properties will be updated, all others will remain unchanged. 必须为每个图像定义“image id”属性，所有其他属性都是可选的。只有通过的属性将被更新，所有其他属性将保持不变。

Additionally to the `standard image properties`, the method accepts the following parameters. 除了标准图像属性外，该方法接受以下参数。

属性	类型	说明
<code>image</code>	<code>string</code>	Base64 encoded image. The maximum size of the encoded image is 1 MB. Base64编码图像。 编码图像的最大大小为1 MB。

Return values 返回值

`(object)` Returns an object containing the IDs of the updated images under the `imageids` property. 返回包含“imageid”属性下更新的图像的ID的对象。

Examples 示例

Rename image 重命名图像

Rename image to “Cloud icon”. 将图像重命名为“Cloud icon”。

Request:

```

1. {
2.   "jsonrpc": "2.0",
3.   "method": "image.update",
4.   "params": {
5.     "imageid": "2",
6.     "name": "Cloud icon"
7.   },
8.   "auth": "038e1d7b1735c6a5436ee9eae095879e",
9.   "id": 1
10. }
```

Response:

```
1. {
2.     "jsonrpc": "2.0",
3.     "result": {
4.         "imageids": [
5.             "2"
6.         ],
7.     },
8.     "id": 1
9. }
```

Source 来源

`CImage::update()` in *frontends/php/include/classes/api/services/CImage.php*.

应用集

这个类用于管理应用集。

对象引用:

- [Application\(应用集\)](#)

相关方法:

- `application.create` - 创建新的应用集
- `application.delete` - 删除应用集
- `application.get` - 获取应用集
- `application.massadd` - 更新应用集
- `application.update` - 添加监控项到应用集

> 应用集对象

以下是 `application` API 的使用方法。

应用集

应用集对象包含以下属性。

属性	类型	说明
<code>applicationid</code>	<code>string</code>	(只读)应用集的 ID。
<code>hostid</code> (必须)	<code>string</code>	应用集所属主机的 ID。 不能进行更新。
<code>name</code> (必须)	<code>string</code>	应用集名称。
<code>flags</code>	<code>integer</code>	(只读) 应用集的来源。 可能的值为： 0 - 普通应用集； 4 - 自动发现的应用集。
<code>templateids</code>	<code>array</code>	(只读) 上级模板应用集的 ID。

application.create

说明

```
object application.create(object/array applications)
```

此方法允许创建新的应用集。

参数

(object/array) 需要去创建的应用集。

此方法接受创建的应用集带有**标准应用集属性**。

返回值

(object) 返回一个包含“applicationID”属性的应用程序 ID 的对象。返回的 ID 的顺序与传递的应用程序的顺序相匹配

范例

创建一个应用集

创建一个应用集来存储 SNMP 监控项。

请求：

```
1. {
2.     "jsonrpc": "2.0",
3.     "method": "application.create",
4.     "params": {
5.         "name": "SNMP Items",
6.         "hostid": "10050"
7.     },
8.     "auth": "038e1d7b1735c6a5436ee9eae095879e",
9.     "id": 1
10. }
```

响应：

```
1. {
2.     "jsonrpc": "2.0",
3.     "result": {
4.         "applicationids": [
5.             "356"
6.         ]
7.     },
8. }
```

application.create

```
8.      "id": 1  
9. }
```

来源

CApplication::create() in *frontends/php/include/classes/api/services/CApplication.php*.

application.delete

说明

```
object application.delete(array applicationIds)
```

此方法用于删除应用集。

参数

(array) 需要去删除的应用集ID

返回值

(object) 返回一个 “applicationid” 属性下的要删除的应用程序 ID 的对象。

范例

删除多个应用集

删除两个应用集

请求：

```
1. {
2.     "jsonrpc": "2.0",
3.     "method": "application.delete",
4.     "params": [
5.         "356",
6.         "358"
7.     ],
8.     "auth": "3a57200802b24cda67c4e4010b50c065",
9.     "id": 1
10. }
```

响应：

```
1. {
2.     "jsonrpc": "2.0",
3.     "result": {
4.         "applicationids": [
5.             "356",
6.             "358"
7.         ]
8.     },
9.     "id": 1
10. }
```

来源

CApplication::delete() in *frontends/php/include/classes/api/services/CApplication.php*.

application.get

说明

`integer/array application.get(object parameters)` 该方法用于根据规定的参数获取应用集。

参数

(object) 定义所需输出的参数。

该方法提供以下参数。

参数	类型	说明
applicationids	string/array	只返回指定 ID 的应用集。
groupids	string/array	只返回指定主机组所属主机的应用集。
hostids	string/array	只返回指定主机所属的应用集。
inherited	boolean	如果设定为 <code>true</code> , 只返回继承该模板的应用集。
itemids	string/array	只返回包含特定监控项的应用集。
templated	boolean	如果设定为 <code>true</code> , 只返回属于该模板的应用集。
templateids	string/array	只返回指定模板的应用集。
selectHost	query	返回值中会包括应用集所属的主机名属性。
selectItems	query	返回值中会应用集包含的监控项属性。
selectDiscoveryRule	query	返回值中会包括应用集的底层自动发现规则属性。
selectApplicationDiscovery	query	返回值中会包括应用集发现的对象属性。
sortfield	string/array	使用规定的属性将结果分类。 可能的值: <code>applicationid</code> 和 <code>name</code> 。
countOutput	flag	在 reference commentary 中详细描述了所有“get”方法的相关参数.
editable	boolean	
excludeSearch	flag	
filter	object	
limit	integer	
output	query	
preservekeys	flag	
search	object	
searchByAny	boolean	
searchWildcardsEnabled	boolean	
sortorder	string/array	

startSearch	flag
-------------	------

返回值

(integer/array) 返回两者其中任一：

- an array of objects;一组对象
- 如果已经使用了“countOutput”参数，则检索对象的计数.

范例

从主机中检索应用集

从主机中根据名称排序检索所有的应用集。

请求：

```

1. {
2.     "jsonrpc": "2.0",
3.     "method": "application.get",
4.     "params": {
5.         "output": "extend",
6.         "hostids": "10001",
7.         "sortfield": "name"
8.     },
9.     "auth": "038e1d7b1735c6a5436ee9eae095879e",
10.    "id": 1
11. }
```

响应：

```

1. {
2.     "jsonrpc": "2.0",
3.     "result": [
4.         {
5.             "applicationid": "13",
6.             "hostid": "10001",
7.             "name": "CPU",
8.             "templateids": []
9.         },
10.        {
11.            "applicationid": "5",
12.            "hostid": "10001",
13.            "name": "Filesystems",
14.            "templateids": []
15.        },
16.        {
17.            "applicationid": "21",
18.            "hostid": "10001",
```

```
19.         "name": "General",
20.         "templateids": []
21.     },
22.     {
23.         "applicationid": "15",
24.         "hostid": "10001",
25.         "name": "Memory",
26.         "templateids": []
27.     },
28. ],
29. "id": 1
30. }
```

参见

- [Host](#)
- [Item](#)

来源

CApplication::get() in *frontends/php/include/classes/api/services/CApplication.php*.

application.massadd

说明

```
object application.massadd(object parameters)
```

此方法用于同时添加多个监控项到指定的应用集。

参数

(object) 参数包含更新应用集和加入应用集监控项的 ID。

该方法接受以下参数。

参数	类型	说明
applications (必须)	array/object	需要更新的应用集。 应用集必须已定义好 <code>applicationid</code> 属性。
items	array/object	监控项加入到指定的应用集。 监控项必须已定义好 <code>itemid</code> 属性。

返回值

(object) 返回一个其中在 `applicationid` 属性下已更新应用集的 ID 的对象。

范例

添加监控项到多个应用集。

添加指定的监控项到两个应用集。

请求：

```

1. {
2.   "jsonrpc": "2.0",
3.   "method": "application.massadd",
4.   "params": {
5.     "applications": [
6.       {
7.         "applicationid": "247"
8.       },
9.       {
10.        "applicationid": "246"
11.      }
12.    ],
13.    "items": [
14.      {
15.        "itemid": "22800"

```

```
16.         },
17.         {
18.             "itemid": "22801"
19.         }
20.     ],
21. },
22. "auth": "038e1d7b1735c6a5436ee9eae095879e",
23. "id": 1
24. }
```

响应：

```
1. {
2.     "jsonrpc": "2.0",
3.     "result": {
4.         "applicationids": [
5.             "247",
6.             "246"
7.         ],
8.     },
9.     "id": 1
10. }
```

参见

- [Item](#)

来源

CAppliapplication::massAdd() in
frontends/php/include/classes/api/services/CAppliapplication.php.

application.update

说明

```
object application.update(object/array applications)
```

此方法用于更新目前的应用集。

参数

(object/array) 需要被更新的 应用集属性。

Applicationid属性必须在每个应用集中已定义，其他所有属性为可选项。只有传递过去的属性会被更新，其他所有属性仍然保持不变。

返回值

(object) 返回一个 applicationids 属性下已更新应用集的ID的对象。

范例

更新应用集的名称。

更新应用集的名称为“Processes and performance”。

请求：

```
1. {
2.     "jsonrpc": "2.0",
3.     "method": "application.update",
4.     "params": {
5.         "applicationid": "13",
6.         "name": "Processes and performance"
7.     },
8.     "auth": "038e1d7b1735c6a5436ee9eae095879e",
9.     "id": 1
10. }
```

响应：

```
1. {
2.     "jsonrpc": "2.0",
3.     "result": {
4.         "applicationids": [
5.             "13"
6.         ]
7.     },
8. }
```

application.update

```
8.      "id": 1  
9. }
```

来源

CApplication::update() in *frontends/php/include/classes/api/services/CApplication.php*.

模板

此类用于管理模板 .

对象引用:

- [模板](#)

可用的方法:

- [template.create](#) - 创建新模板
- [template.delete](#) - 删除模板
- [template.get](#) - 检索模板
- [template.massadd](#) - 添加相关对象到模板中
- [template.massremove](#) - 从模板中删除相关对象
- [template.massupdate](#) - 从模板中替换或删除相关对象
- [template.update](#) - 更新模板

> Template object模板对象

The following objects are directly related to the `template` API以下对象与 `template` API直接相关。.

Template模板

The template object has the following properties模板对象具有以下属性。.

Property参数	Type类型	Description说明
<code>templateid</code>	<code>string</code>	(readonly) ID of the template模板的ID.
<code>host(required)</code>	<code>string</code>	Technical name of the template模板的技术性名称.
<code>description</code>	<code>text</code>	Description of the template模板说明.
<code>name</code>	<code>string</code>	Visible name of the host主机的可见名称. Default: <code>host</code> property value属性值.

template.create

Description说明

```
object template.create(object/array templates)
```

This method allows to create new templates此方法允许创建新模板.

Parameters参数

(object/array) Templates to create创建模板.

Additionally to the standard template properties, the method accepts the following parameters.除standard template properties之外，该方法接受以下参数.

Parameter参数	Type类型	Description说明
groups(required)	object/array	Host groups to add the template to将模板添加到主机组. The host groups must have the <code>groupid</code> property defined主机组必须定义 <code>groupid</code> 属性. .
templates	object/array	Templates to be linked to the template要链接到模板的模板. The templates must have the <code>templateid</code> property defined模板必须定义 <code>templateid</code> 属性. .
macros	object/array	User macros to be created for the template为模板创建的用户宏.
hosts	object/array	Hosts to link the template to链接模板的主机.

Return values返回值

(object) Returns an object containing the IDs of the created templates under the `templateids` property. The order of the returned IDs matches the order of the passed templates. (object) 返回一个对象，它包含 `templateids` 属性下创建的模板的ID，返回的ID的顺序与传递的模板的顺序相匹配。

Examples范例

Creating a template创建模板

Create a template and link it to two hosts创建一个模板并将其链接到两个主机.

Request请求:

```

1. {
2.   "jsonrpc": "2.0",
3.   "method": "template.create",
4.   "params": {
5.     "host": "Linux template",

```

```
6.     "groups": {
7.         "groupid": 1
8.     },
9.     "hosts": [
10.        {
11.            "hostid": "10084"
12.        },
13.        {
14.            "hostid": "10090"
15.        }
16.    ]
17. },
18. "auth": "038e1d7b1735c6a5436ee9eae095879e",
19. "id": 1
20. }
```

Response响应：

```
1. {
2.     "jsonrpc": "2.0",
3.     "result": {
4.         "templateids": [
5.             "10086"
6.         ]
7.     },
8.     "id": 1
9. }
```

Source来源

CTemplate::create() in *frontends/php/include/classes/api/services/CTemplate.php*.

template.delete

Description说明

```
object template.delete(array templateIds)
```

This method allows to delete templates此方法允许删除模板.

Parameters参数

(array) IDs of the templates to delete要删除的模板的ID.

Return values返回值

(object) Returns an object containing the IDs of the deleted templates under the templateids property. (object) 返回一个对象，该对象包含 templateids 属性下删除的模板的ID.

Examples范例

Deleting multiple templates删除多个模板

Delete two templates删除两个模板.

Request请求:

```
1. {
2.   "jsonrpc": "2.0",
3.   "method": "template.delete",
4.   "params": [
5.     "13",
6.     "32"
7.   ],
8.   "auth": "038e1d7b1735c6a5436ee9eae095879e",
9.   "id": 1
10. }
```

Response响应:

```
1. {
2.   "jsonrpc": "2.0",
3.   "result": {
4.     "templateids": [
5.       "13",
6.       "32"
7.     ]
8.   },
9.   "id": 1
```

```
template.delete
```

```
10. }
```

Source来源

`CTemplate::delete()` in `frontends/php/include/classes/api/services/CTemplate.php`.

template.get

Description说明

```
integer/array template.get(object parameters)
```

The method allows to retrieve templates according to the given parameters该方法允许根据指定的参数检索模板.

Parameters参数

(object) Parameters defining the desired output定义所需输出的参数.

The method supports the following parameters该方法支持以下参数.

Parameter参数	Type类型	Description说明
templateids	string/array	Return only templates with the given template IDs只返回具有给定模板ID的模板.
groupids	string/array	Return only templates that belong to the given host groups只返回所属指定的主机组的模板.
parentTemplateids	string/array	Return only templates that are children of the given templates只返回作为指定模板的子项的模板.
hostids	string/array	Return only templates that are linked to the given hosts只返回链接到指定主机的模板.
graphids	string/array	Return only templates that contain the given graphs只返回包含指定图形的模板.
itemids	string/array	Return only templates that contain the given items只返回包含指定监控项的模板.
triggerids	string/array	Return only templates that contain the given triggers只返回包含指定触发器的模板.
with_items	flag	Return only templates that have items只返回具有监控项的模板.
with_triggers	flag	Return only templates that have triggers只返回具有触发器的模板.
with_graphs	flag	Return only templates that have graphs只返回具有图形的模板.
with_httptests	flag	Return only templates that have web scenarios只返回具有Web场景的模板.
selectGroups	query	Return the host groups that the template belongs to in the <code>groups</code> property 在 <code>groups</code> 属性中返回模板所属的主机组.
selectHosts	query	Return the hosts that are linked to the template in the <code>hosts</code> property在 <code>hosts</code> 属性中返回链接到模板的主机. Supports <code>count</code> .

selectTemplates	query	Return the child templates in the <code>templates</code> property返回 <code>templates</code> 属性中的子模板. Supports <code>count</code> .
selectParentTemplates	query	Return the parent templates in the <code>parentTemplates</code> property返回 <code>parentTemplates</code> 属性中的父模板. Supports <code>count</code> .
selectHttpTests	query	Return the web scenarios from the template in the <code>httpSteps</code> property从 <code>httpSteps</code> 属性中的模板返回Web场景. Supports <code>count</code> .
selectItems	query	Return items from the template in the <code>items</code> property在 <code>items</code> 属性中返回模板中的监控项. Supports <code>count</code> .
selectDiscoveries	query	Return low-level discoveries from the template in the <code>discoveries</code> property从 <code>discoveries</code> 属性中的模板返回低水平自动发现. Supports <code>count</code> .
selectTriggers	query	

Examples范例

Retrieving templates by name按名称检索模板

Retrieve all data about two templates named “Template OS Linux” and “Template OS Windows”. 检索有关名为“Template OS Linux”和“Template OS Windows”的两个模板的所有数据.

Request请求：

```

1. {
2.   "jsonrpc": "2.0",
3.   "method": "template.get",
4.   "params": {
5.     "output": "extend",
6.     "filter": {
7.       "host": [
8.         "Template OS Linux",
9.         "Template OS Windows"
10.      ]
11.    }
12.  },
13.  "auth": "038e1d7b1735c6a5436ee9eae095879e",
14.  "id": 1
15. }
```

Response响应：

```

1. {
2.   "jsonrpc": "2.0",
3.   "result": [
4.     {
5.       "proxy_hostid": "0",
6.       "host": "Template OS Linux",
```

```
7.         "status": "3",
8.         "disable_until": "0",
9.         "error": "",
10.        "available": "0",
11.        "errors_from": "0",
12.        "lastaccess": "0",
13.        "ipmi_authtype": "0",
14.        "ipmi_privilege": "2",
15.        "ipmi_username": "",
16.        "ipmi_password": "",
17.        "ipmi_disable_until": "0",
18.        "ipmi_available": "0",
19.        "snmp_disable_until": "0",
20.        "snmp_available": "0",
21.        "maintenanceid": "0",
22.        "maintenance_status": "0",
23.        "maintenance_type": "0",
24.        "maintenance_from": "0",
25.        "ipmi_errors_from": "0",
26.        "snmp_errors_from": "0",
27.        "ipmi_error": "",
28.        "snmp_error": "",
29.        "jmx_disable_until": "0",
30.        "jmx_available": "0",
31.        "jmx_errors_from": "0",
32.        "jmx_error": "",
33.        "name": "Template OS Linux",
34.        "flags": "0",
35.        "templateid": "10001",
36.        "description": "",
37.        "tls_connect": "1",
38.        "tls_accept": "1",
39.        "tls_issuer": "",
40.        "tls_subject": "",
41.        "tls_psk_identity": "",
42.        "tls_psk": ""
43.    },
44.    {
45.        "proxy_hostid": "0",
46.        "host": "Template OS Windows",
47.        "status": "3",
48.        "disable_until": "0",
49.        "error": "",
50.        "available": "0",
51.        "errors_from": "0",
52.        "lastaccess": "0",
53.        "ipmi_authtype": "0",
54.        "ipmi_privilege": "2",
55.        "ipmi_username": "",
56.        "ipmi_password": "",
57.        "ipmi_disable_until": "0",
58.        "ipmi_available": "0",
59.        "snmp_disable_until": "0",
```

```

60.         "snmp_available": "0",
61.         "maintenanceid": "0",
62.         "maintenance_status": "0",
63.         "maintenance_type": "0",
64.         "maintenance_from": "0",
65.         "ipmi_errors_from": "0",
66.         "snmp_errors_from": "0",
67.         "ipmi_error": "",
68.         "snmp_error": "",
69.         "jmx_disable_until": "0",
70.         "jmx_available": "0",
71.         "jmx_errors_from": "0",
72.         "jmx_error": "",
73.         "name": "Template OS Windows",
74.         "flags": "0",
75.         "templateid": "10081",
76.         "description": "",
77.         "tls_connect": "1",
78.         "tls_accept": "1",
79.         "tls_issuer": "",
80.         "tls_subject": "",
81.         "tls_psk_identity": "",
82.         "tls_psk": ""
83.     }
84. ],
85. "id": 1
86. }
```

See also参考

- [Host group](#)
- [Template](#)
- [User macro](#)
- [Host interface](#)

Source来源

`CTemplate::get()` in *frontends/php/include/classes/api/services/CTemplate.php*.

template.massadd

Description说明

```
object template.massadd(object parameters)
```

This method allows to simultaneously add multiple related objects to the given templates该方法允许同时向给定的模板添加多个相关对象。.

Parameters参数

(object) Parameters containing the IDs of the templates to update and the objects to add to the templates包含要更新的模板的ID和要添加到模板的对象的参数。

The method accepts the following parameters该方法接受以下参数。

Parameter参数	Type类型	Description说明
templates(required)	object/array	Templates to be updated要更新的模板。The templates must have the templateid property defined模板必须定义 templateid 属性。
groups	object/array	Host groups to add the given templates to主机组添加指定的模板。The host groups must have the groupid property defined主机组必须定义 groupid 属性。
hosts	object/array	Hosts and templates to link the given templates to将给定模板链接到的主机和模板。The hosts must have the hostid property defined主机必须定义 hostid 属性。
macros	object/array	User macros to be created for the given templates为给定的模板创建用户宏。
templates_link	object/array	Templates to link to the given templates链接到给定模板的模板。The templates must have the templateid property defined模板必须定义 templateid 属性。

Return values返回值

(object) Returns an object containing the IDs of the updated templates under the templateids property. (object) 返回一个对象，它包含 templateids 属性下更新的模板的ID

Examples范例

Adding templates to a group将模板添加到组

Add two templates to the host group "2"将两个模板添加到主机组"2"。

Request请求：

```

1. {
2.     "jsonrpc": "2.0",
3.     "method": "template.massadd",
4.     "params": {
5.         "templates": [
6.             {
7.                 "templateid": "10085"
8.             },
9.             {
10.                "templateid": "10086"
11.            }
12.        ],
13.        "groups": [
14.            {
15.                "groupid": "2"
16.            }
17.        ]
18.    },
19.    "auth": "038e1d7b1735c6a5436ee9eae095879e",
20.    "id": 1
21. }

```

Response响应：

```

1. {
2.     "jsonrpc": "2.0",
3.     "result": {
4.         "templateids": [
5.             "10085",
6.             "10086"
7.         ]
8.     },
9.     "id": 1
10. }

```

Linking a template to hosts将模板链接到主机

Link template “10073” to two hosts链接模板“10073”到两个主机。.

Request请求：

```

1. {
2.     "jsonrpc": "2.0",
3.     "method": "template.massadd",
4.     "params": {
5.         "templates": [
6.             {
7.                 "templateid": "10073"
8.             }
9.         ],
10.        "hosts": [

```

```

11.      {
12.          "hostid": "10106"
13.      },
14.      {
15.          "hostid": "10104"
16.      }
17.  ]
18. },
19. "auth": "038e1d7b1735c6a5436ee9eae095879e",
20. "id": 1
21. }
```

Response响应：

```

1. {
2.     "jsonrpc": "2.0",
3.     "result": {
4.         "templateids": [
5.             "10073"
6.         ]
7.     },
8.     "id": 1
9. }
```

See also参考

- [template.update](#)
- [Host](#)
- [Host group](#)
- [User macro](#)

Source来源

`CTemplate::massAdd()` in *frontends/php/include/classes/api/services/CTemplate.php*.

template.massremove

Description说明

```
object template.massremove(object parameters)
```

This method allows to remove related objects from multiple templates该方法允许从多个模板中删除相关对象.

Parameters参数

(object) Parameters containing the IDs of the templates to update and the objects that should be removed包含要更新的模板的ID和应删除的对象的参数.

Parameter参数	Type类型	Description说明
templateids(required)	string/array	IDs of the templates to be updated要更新的模板的ID.
groupids	string/array	Host groups to remove the given templates from从给定的模板中删除主机组.
hostids	string/array	Hosts or templates to unlink the given templates from取消链接给定模板的主机或模板.
macros	string/array	User macros to delete from the given templates删除指定模板的用户宏.
templateids_clear	string/array	Templates to unlink and clear from the given templates从给定的模板中取消链接和清除的模板.
templateids_link	string/array	Templates to unlink from the given templates取消与给定模板链接的模板.

Return values返回值

(object) Returns an object containing the IDs of the updated templates under the templateids property. (object) 返回一个对象，它包含 templateids 属性下更新的模板的ID.

Examples范例

Removing templates from a group从组中删除模板

Remove two templates from group "2"从组"2"中删除两个模板.

Request请求:

```

1. {
2.   "jsonrpc": "2.0",
3.   "method": "template.massremove",
4.   "params": {
```

```

5.     "templateids": [
6.         "10085",
7.         "10086"
8.     ],
9.     "groupids": "2"
10.    },
11.    "auth": "038e1d7b1735c6a5436ee9eae095879e",
12.    "id": 1
13. }
```

Response响应：

```

1. {
2.     "jsonrpc": "2.0",
3.     "result": {
4.         "templateids": [
5.             "10085",
6.             "10086"
7.         ]
8.     },
9.     "id": 1
10. }
```

Unlinking templates from a host从主机取消链接模板

Unlink template “10085” from two hosts从两个主机取消链接模板“10085”.**Request请求：**

```

1. {
2.     "jsonrpc": "2.0",
3.     "method": "template.massremove",
4.     "params": {
5.         "templateids": "10085",
6.         "hostids": [
7.             "10106",
8.             "10104"
9.         ]
10.    },
11.    "auth": "038e1d7b1735c6a5436ee9eae095879e",
12.    "id": 1
13. }
```

Response响应：

```

1. {
2.     "jsonrpc": "2.0",
3.     "result": {
4.         "templateids": [
5.             "10085"
```

```
6.      ]
7.    },
8.    "id": 1
9. }
```

See also参考

- [template.update](#)
- [User macro](#)

Source来源

`CTemplate::massRemove()` in *frontends/php/include/classes/api/services/CTemplate.php*.

template.massupdate

Description说明

```
object template.massupdate(object parameters)
```

This method allows to simultaneously replace or remove related objects and update properties on multiple templates此方法允许同时替换或删除相关对象并更新多个模板上的属性.

Parameters参数

(object) Parameters containing the IDs of the templates to update and the properties that should be updated包含要更新的模板的ID和应更新的属性的参数.

Additionally to the [standard template properties](#), the method accepts the following parameters.除[standard template properties](#)之外，该方法接受以下参数。

Parameter参数	Type类型	Description说明
templates(required)	object/array	Templates to be updated要更新的模板. The templates must have the <code>templateid</code> property defined模板必须定义 <code>templateid</code> 属性.
groups	object/array	Host groups to replace the current host groups the templates belong to主机组替换模板所属的当前主机组. The host groups must have the <code>groupid</code> property defined主机组必须定义 <code>groupid</code> 属性.
hosts	object/array	Hosts and templates to replace the ones the templates are currently linked to主机和模板来替换当前链接到的模板. Both hosts and templates must use the <code>hostid</code> property to pass an ID主机和模板都必须使用 <code>hostid</code> 属性传递一个ID.
macros	object/array	User macros to replace the current user macros on the given templates用户宏替换给定模板上的当前用户宏.
templates_clear	object/array	Templates to unlink and clear from the given templates从指定模板中取消链接和清除的模板. The templates must have the <code>templateid</code> property defined模板必须定义 <code>templateid</code> 属性.
templates_link	object/array	Templates to replace the currently linked templates用于替换当前链接的模板的模板. The templates must have the <code>templateid</code> property defined模板必须定义 <code>templateid</code> 属性.

Return values返回值

(object) Returns an object containing the IDs of the updated templates under the `templateids` property. (object) 返回一个对象，它包含 `templateids` 属性下更新的模板的ID.

Examples范例

Replacing host groups更换主机组

Unlink and clear template "10091" from the given templates从给定的模板中取消链接并清除模板"10091".

Request请求:

```

1. {
2.     "jsonrpc": "2.0",
3.     "method": "template.massupdate",
4.     "params": {
5.         "templates": [
6.             {
7.                 "templateid": "10085"
8.             },
9.             {
10.                "templateid": "10086"
11.            }
12.        ],
13.        "templates_clear": [
14.            {
15.                "templateid": "10091"
16.            }
17.        ]
18.    },
19.    "auth": "038e1d7b1735c6a5436ee9eae095879e",
20.    "id": 1
21. }
```

Response响应:

```

1. {
2.     "jsonrpc": "2.0",
3.     "result": {
4.         "templateids": [
5.             "10085",
6.             "10086"
7.         ]
8.     },
9.     "id": 1
10. }
```

See also参考

- [template.update](#)
- [template.massadd](#)

- [Host group](#)
- [User macro](#)

Source来源

`CTemplate::massUpdate() in frontends/php/include/classes/api/services/CTemplate.php.`

template.update

Description说明

```
object template.update(object/array templates)
```

This method allows to update existing templates此方法允许更新现有模板。

Parameters参数

(object/array) Template properties to be updated要更新的模板属性。

The `templateid` property must be defined for each template, all other properties are optional. Only the given properties will be updated, all others will remain unchanged. 必须为每个模板定义 `templateid` 属性，所有其他属性都是可选的。只有给定的属性将被更新，所有其他属性将保持不变。

Additionally to the `standard template properties`, the method accepts the following parameters.除 `standard template properties`之外，该方法接受以下参数。

Parameter参数	Type类型	Description说明
groups	object/array	Host groups to replace the current host groups the templates belong to主机组替换模板所属的当前主机组。The host groups must have the <code>groupid</code> property defined主机组必须定义 <code>groupid</code> 属性。
hosts	object/array	Hosts and templates to replace the ones the templates are currently linked to主机和模板来替换当前链接到的模板。Both hosts and templates must use the <code>hostid</code> property to pass an ID主机和模板都必须使用 <code>hostid</code> 属性传递一个ID。
macros	object/array	User macros to replace the current user macros on the given templates用户宏替换给定模板上的当前用户宏。
templates	object/array	Templates to replace the currently linked templates. Templates that are not passed are only unlinked.用于替换当前链接的模板的模板，不通过的模板仅被取消链接。The templates must have the <code>templateid</code> property defined模板必须定义 <code>templateid</code> 属性。
templates_clear	object/array	Templates to unlink and clear from the given templates从指定模板中取消链接和清除的模板。The templates must have the <code>templateid</code> property defined模板必须定义 <code>templateid</code> 属性。

Return values返回值

(object) Returns an object containing the IDs of the updated templates under the `templateids` property. (object) 返回一个对象，它包含 `templateids` 属性下更新的模板的ID。

Examples范例

Renaming a template重命名模板

Rename the template to "Template OS Linux" 将模板重命名为"Template OS Linux".

Request请求：

```
1. {
2.     "jsonrpc": "2.0",
3.     "method": "template.update",
4.     "params": {
5.         "templateid": "10086",
6.         "name": "Template OS Linux"
7.     },
8.     "auth": "038e1d7b1735c6a5436ee9eae095879e",
9.     "id": 1
10. }
```

Response响应：

```
1. {
2.     "jsonrpc": "2.0",
3.     "result": {
4.         "templateids": [
5.             "10086"
6.         ]
7.     },
8.     "id": 1
9. }
```

Source来源

CTemplate::update() in *frontends/php/include/classes/api/services/CTemplate.php*.

用户

该类用于用户的使用。

对象引用：

- [User](#)

可用的方法：

- [user.addmedia](#) - 向用户添加媒介
- [user.create](#) - 创建新用户
- [user.delete](#) - 删除用户
- [user.deletemedia](#) - 从用户中删除媒介
- [user.get](#) - 检索用户
- [user.login](#) - 登录到API
- [user.logout](#) - 退出API
- [user.update](#) - 更新用户
- [user.updatemedia](#) - 更新用户媒介
- [user.updateprofile](#) - 更新当前登录的用户

> User object

The following objects are directly related to the [user API](#).

User

The user object has the following properties.

Property	Type	Description
userid	string	(readonly) ID of the user.
alias(required)	string	User alias.
attemptclock	timestamp	(readonly) Time of the last unsuccessful login attempt.
attempt_failed	integer	(readonly) Recent failed login attempt count.
attempt_ip	string	(readonly) IP address from where the last unsuccessful login attempt came from.
autologin	integer	Whether to enable auto-login. Possible values: 0 - (default) auto-login disabled; 1 - auto-login enabled.
autologout	string	User session life time. Accepts seconds and time unit with suffix. If set to 0s, the session will never expire. Default: 15m.
lang	string	Language code of the user's language. Default: enGB .
name	string	Name of the user.
refresh	string	Automatic refresh period. Accepts seconds and time unit with suffix. Default: 30s.
rowsper_page	integer	Amount of object rows to show per page. Default: 50.
surname	string	Surname of the user.
theme	string	User's theme. Possible values: default - (default) system default; blue-theme - Blue; dark-theme - Dark.
type	integer	Type of the user. Possible values: 1 - (default) Zabbix user; 2 - Zabbix admin; 3 - Zabbix super admin.
url	string	URL of the page to redirect the user to after logging in.

Media

The media object has the following properties.

Property	Type	Description
mediatypeid(required)	string	ID of the media type used by the media.
		Address, user name or other identifier of the

		recipient.
active	integer	Whether the media is enabled. Possible values: 0 - (<i>default</i>) enabled; 1 - disabled.
severity	integer	Trigger severities to send notifications about. Severities are stored in binary form with each bit representing the corresponding severity. For example, 12 equals 1100 in binary and means, that notifications will be sent from triggers with severities warning and average. Refer to the trigger object page for a list of supported trigger severities. Default: 63
period	string	Time when the notifications can be sent as a time period or user macros separated by a semicolon. Default: 1-7,00:00-24:00

user.addmedia

Description

```
object user.addmedia(object parameters)
```

This method allows to add new media to multiple users.

This method is deprecated and will be removed in the future. Please use [user.update](#) instead.

Parameters

(object) Parameters defining the media to create and the users to add them to.

Parameter	Type	Description
medias (required)	object/array	Media to create for the given users. The media <code>userid</code> property must not be defined.
users (required)	object/array	Users to add the media to. The users must have the <code>userid</code> property defined.

Return values

(object) Returns an object containing the IDs of the created media under the **mediaids** property.

Examples

Adding a media to multiple users

Create a common e-mail media for two users. The media must send notifications about all alerts at any time.

Request:

```

1. {
2.   "jsonrpc": "2.0",
3.   "method": "user.addmedia",
4.   "params": {
5.     "users": [
6.       {
7.         "userid": "1"
8.       },
9.       {
10.        "userid": "2"
11.      }
12.    ],

```

```

13.     "medias": {
14.         "mediatypeid": "1",
15.         "sendto": "[email protected]",
16.         "active": 0,
17.         "severity": 63,
18.         "period": "1-7,00:00-24:00"
19.     }
20. },
21. "auth": "038e1d7b1735c6a5436ee9eae095879e",
22. "id": 1
23. }
```

Response:

```

1. {
2.     "jsonrpc": "2.0",
3.     "result": {
4.         "mediaids": [
5.             "12",
6.             "13"
7.         ],
8.     },
9.     "id": 1
10. }
```

See also

- [user.update](#)
- [user.updatemedia](#)
- [Media](#)
- [User](#)

Source

`CUser::addMedia()` in *frontends/php/include/classes/api/services/CUser.php*.

user.create

Description

```
object user.create(object/array users)
```

This method allows to create new users.

Parameters

(object/array) Users to create.

Additionally to the [standard user properties](#), the method accepts the following parameters.

Parameter	Type	Description
passwd(required)	string	User's password.
usrgrps(required)	array	User groups to add the user to. The user groups must have the <code>usrgrpid</code> property defined.
user_medias	array	Medias to create for the user.

Return values

(object) Returns an object containing the IDs of the created users under the `userids` property. The order of the returned IDs matches the order of the passed users.

Examples

Creating a user

Create a new user, add him to a user group and create a new media for him.

Request:

```

1. {
2.   "jsonrpc": "2.0",
3.   "method": "user.create",
4.   "params": {
5.     "alias": "John",
6.     "passwd": "Doe123",
7.     "usrgrps": [
8.       {
9.         "usrgrpid": "7"
10.      }
11.    ],
12.    "user_medias": [

```

```
13.      {
14.          "mediatypeid": "1",
15.          "sendto": "[email protected]",
16.          "active": 0,
17.          "severity": 63,
18.          "period": "1-7,00:00-24:00"
19.      }
20.  ]
21. },
22. "auth": "038e1d7b1735c6a5436ee9eae095879e",
23. "id": 1
24. }
```

Response:

```
1. {
2.     "jsonrpc": "2.0",
3.     "result": {
4.         "userids": [
5.             "12"
6.         ]
7.     },
8.     "id": 1
9. }
```

See also

- [Media](#)
- [User group](#)

Source

`CUser::create()` in *frontends/php/include/classes/api/services/CUser.php*.

user.delete

Description

```
object user.delete(array users)
```

This method allows to delete users.

Parameters

(array) IDs of users to delete.

Return values

(object) Returns an object containing the IDs of the deleted users under the `userids` property.

Examples

Deleting multiple users

Delete two users.

Request:

```
1. {
2.   "jsonrpc": "2.0",
3.   "method": "user.delete",
4.   "params": [
5.     "1",
6.     "5"
7.   ],
8.   "auth": "3a57200802b24cda67c4e4010b50c065",
9.   "id": 1
10. }
```

Response:

```
1. {
2.   "jsonrpc": "2.0",
3.   "result": {
4.     "userids": [
5.       "1",
6.       "5"
7.     ]
8.   },
9.   "id": 1
}
```

user.delete

```
10. }
```

Source

CUser::delete() in *frontends/php/include/classes/api/services/CUser.php*.

user.deletemedia

Description

```
object user.deletemedia(string/array mediaIds)
```

This method allows to delete media.

This method is deprecated and will be removed in the future. Please use [user.update](#) instead.

Parameters

(string/array) IDs of the media to delete.

Return values

(object) Returns an object containing the IDs of the deleted media under the `mediaids` property.

Examples

Deleting multiple media

Delete two media.

Request:

```
1. {
2.   "jsonrpc": "2.0",
3.   "method": "user.deletemedia",
4.   "params": [
5.     "11",
6.     "13"
7.   ],
8.   "auth": "3a57200802b24cda67c4e4010b50c065",
9.   "id": 1
10. }
```

Response:

```
1. {
2.   "jsonrpc": "2.0",
3.   "result": {
4.     "mediaids": [
5.       "11",
6.       "13"
7.     ]
8.   }
9. }
```

```
7.      ],
8.    },
9.    "id": 1
10. }
```

See also

- [user.update](#)
- [user.updatemedia](#)

Source

CUser::deleteMedia() in *frontends/php/include/classes/api/services/CUser.php*.

user.get

Description

```
integer/array user.get(object parameters)
```

The method allows to retrieve users according to the given parameters.

Parameters

(object) Parameters defining the desired output.

The method supports the following parameters.

Parameter	Type	Description
mediaids	string/array	Return only users that use the given media.
mediatypeids	string/array	Return only users that use the given media types.
userids	string/array	Return only users with the given IDs.
usrgrpids	string/array	Return only users that belong to the given user groups.
getAccess	flag	Adds additional information about user permissions. Adds the following properties for each user: <code>guiaccess</code> - (integer) <i>user's frontend authentication method</i> . Refer to the <code>gui_access</code> property of the <code>user group object</code> for a list of possible values. <code>debug_mode</code> - (integer) indicates whether debug is enabled for the user. Possible values: 0 - debug disabled, 1 - debug enabled. <code>users_status</code> - (integer) indicates whether the user is disabled. Possible values: 0 - user enabled, 1 - user disabled.
selectMedias	query	Return media used by the user in the <code>medias</code> property.
selectMediatypes	query	Return media types used by the user in the <code>mediatypes</code> property.
selectUsrgrps	query	Return user groups that the user belongs to in the <code>usrgrps</code> property.
sortfield	string/array	Sort the result by the given properties. Possible values are: <code>userid</code> and <code>alias</code> .
countOutput	flag	These parameters being common for all <code>get</code> methods are described in detail in the reference commentary .
editable	boolean	

excludeSearch	flag
filter	object
limit	integer
output	query
preservekeys	flag
search	object
searchByAny	boolean
searchWildcardsEnabled	boolean
sortorder	string/array
startSearch	flag

Return values

(integer/array) Returns either:

- an array of objects;
- the count of retrieved objects, if the `countOutput` parameter has been used.

Examples

Retrieving users

Retrieve all of the configured users.

Request:

```

1. {
2.   "jsonrpc": "2.0",
3.   "method": "user.get",
4.   "params": {
5.     "output": "extend"
6.   },
7.   "auth": "038e1d7b1735c6a5436ee9eae095879e",
8.   "id": 1
9. }
```

Response:

```

1. {
2.   "jsonrpc": "2.0",
3.   "result": [
4.     {
5.       "userid": "1",
6.       "alias": "Admin",
7.       "name": "Zabbix",
```

user.get

```
8.         "surname": "Administrator",
9.         "url": "",
10.        "autologin": "1",
11.        "autologout": "0s",
12.        "lang": "ru_RU",
13.        "refresh": "0s",
14.        "type": "3",
15.        "theme": "default",
16.        "attempt_failed": "0",
17.        "attempt_ip": "",
18.        "attempt_clock": "0",
19.        "rows_per_page": "50"
20.    },
21.    {
22.        "userid": "2",
23.        "alias": "guest",
24.        "name": "Default2",
25.        "surname": "User",
26.        "url": "",
27.        "autologin": "0",
28.        "autologout": "15m",
29.        "lang": "en_GB",
30.        "refresh": "30s",
31.        "type": "1",
32.        "theme": "default",
33.        "attempt_failed": "0",
34.        "attempt_ip": "",
35.        "attempt_clock": "0",
36.        "rows_per_page": "50"
37.    }
38. ],
39. "id": 1
40. }
```

See also

- [Media](#)
- [Media type](#)
- [User group](#)

Source

CUser::get() in *frontends/php/include/classes/api/services/CUser.php*.

user.login

Description

```
string/object user.login(object parameters)
```

This method allows to log in to the [API](#) and generate an authentication token.

Parameters

This method is available to unauthenticated users and must be called without the `auth` parameter in the JSON-RPC request.

`(object) Parameters containing the user name and password.`

The method accepts the following parameters.

Parameter	Type	Description
<code>password</code> (required)	string	User password. Unused for HTTP authentication.
<code>user</code> (required)	string	User name.
<code>userData</code>	flag	Return information about the authenticated user.

When using HTTP authentication, the user name in the [API](#) request must match the one used in the `Authorization` header. The password will not be validated and can be omitted.

Return values

`(string/object) If the userData parameter is used, returns an object containing information about the authenticated user.`

Additionally to the [standard user properties](#), the following information is returned:

Property	Type	Description
<code>debug_mode</code>	boolean	Whether debug mode is enabled for the user.
<code>gui_access</code>	integer	User's authentication method to the frontend. Refer to the <code>gui_access</code> property of the user group object for a list of possible values.
<code>sessionid</code>	string	Authentication token, which must be used in the following API requests.
<code>userip</code>	string	IP address of the user.

If a user has been successfully authenticated after one or more failed attempts, the method will return the current values for the `attempt_clock`, `attempt_failed` and `attempt_ip` properties and then reset them.

If the `userData` parameter is not used, the method returns an authentication token.

The generated authentication token should be remembered and used in the `auth` parameter of the following JSON-RPC requests. It is also required when using HTTP authentication.

Examples

Authenticating a user

Authenticate a user.

Request:

```
1. {
2.   "jsonrpc": "2.0",
3.   "method": "user.login",
4.   "params": {
5.     "user": "Admin",
6.     "password": "zabbix"
7.   },
8.   "id": 1
9. }
```

Response:

```
1. {
2.   "jsonrpc": "2.0",
3.   "result": "0424bd59b807674191e7d77572075f33",
4.   "id": 1
5. }
```

Requesting authenticated user's information

Authenticate and return additional information about the user.

Request:

```
1. {
2.   "jsonrpc": "2.0",
3.   "method": "user.login",
4.   "params": {
5.     "user": "Admin",
6.     "password": "zabbix",
7.     "userData": true
8.   },
9.   "id": 1
10. }
```

Response:

```
1. {
2.     "jsonrpc": "2.0",
3.     "result": {
4.         "userid": "1",
5.         "alias": "Admin",
6.         "name": "Zabbix",
7.         "surname": "Administrator",
8.         "url": "",
9.         "autologin": "1",
10.        "autologout": "0",
11.        "lang": "ru_RU",
12.        "refresh": "0",
13.        "type": "3",
14.        "theme": "default",
15.        "attempt_failed": "0",
16.        "attempt_ip": "127.0.0.1",
17.        "attempt_clock": "1355919038",
18.        "rows_per_page": "50",
19.        "debug_mode": true,
20.        "userip": "127.0.0.1",
21.        "sessionid": "5b56eee8be445e98f0bd42b435736e42",
22.        "gui_access": "0"
23.    },
24.    "id": 1
25. }
```

See also

- [user.logout](#)

Source

CUser::login() in *frontends/php/include/classes/api/services/CUser.php*.

user.logout

Description

```
string/object user.logout(array)
```

This method allows to log out of the [API](#) and invalidates the current authentication token.

Parameters

(array) The method accepts an empty array.

Return values

(boolean) Returns `true` if the user has been logged out successfully.

Examples

Logging out

Log out from the [API](#).

Request:

```
1. {
2.   "jsonrpc": "2.0",
3.   "method": "user.logout",
4.   "params": [],
5.   "id": 1,
6.   "auth": "16a46baf181ef9602e1687f3110abf8a"
7. }
```

Response:

```
1. {
2.   "jsonrpc": "2.0",
3.   "result": true,
4.   "id": 1
5. }
```

See also

- [user.login](#)

Source

CUser::login() in *frontends/php/include/classes/api/services/CUser.php*.

user.update

Description

```
object user.update(object/array users)
```

This method allows to update existing users.

Parameters

(object/array) User properties to be updated.

The `userid` property must be defined for each user, all other properties are optional. Only the passed properties will be updated, all others will remain unchanged.

Additionally to the [standard user properties](#), the method accepts the following parameters.

Parameter	Type	Description
passwd	string	User's password.
usrgrps	array	User groups to replace existing user groups. The user groups must have the <code>usrgrpid</code> property defined.
user_medias	array	Medias to replace existing medias.

Return values

(object) Returns an object containing the IDs of the updated users under the `userids` property.

Examples

Renaming a user

Rename a user to John Doe.

Request:

```

1. {
2.   "jsonrpc": "2.0",
3.   "method": "user.update",
4.   "params": {
5.     "userid": "1",
6.     "name": "John",
7.     "surname": "Doe"
8.   },
9.   "auth": "038e1d7b1735c6a5436ee9eae095879e",

```

user.update

```
10.      "id": 1
11. }
```

Response:

```
1. {
2.   "jsonrpc": "2.0",
3.   "result": {
4.     "userids": [
5.       "1"
6.     ]
7.   },
8.   "id": 1
9. }
```

See also

- [user.updateprofile](#)

Source

CUser::update() in *frontends/php/include/classes/api/services/CUser.php*.

user.updatemedia

Description

```
object user.updatemedia(object parameters)
```

This method allows to update media for multiple users.

This method is deprecated and will be removed in the future. Please use [user.update](#) instead.

Parameters

(object) Parameters defining the media and users to be updated.

Parameter	Type	Description
medias (required)	object/array	Media to replace existing media. If a media has the <code>mediaid</code> property defined it will be updated, otherwise a new media will be created.
users (required)	object/array	Users to update. The users must have the <code>userid</code> property defined.

Return values

(object) Returns an object containing the IDs of the updated users under the `userids` property.

Examples

Replacing media for multiple users

Replace all media used by the two users with a common e-mail media. The media must send notifications about all alerts at any time.

Request:

```
1. {
2.     "jsonrpc": "2.0",
3.     "method": "user.updatemedia",
4.     "params": {
5.         "users": [
6.             {
7.                 "userid": "1"
8.             },
9.             {
10.                "userid": "2"
11.            }
12.        ]
13.    }
14. }
```

```
12.     ],
13.     "medias": {
14.         "mediatypeid": "1",
15.         "sendto": "[email protected]",
16.         "active": 0,
17.         "severity": 63,
18.         "period": "1-7,00:00-24:00"
19.     }
20. },
21. "auth": "038e1d7b1735c6a5436ee9eae095879e",
22. "id": 1
23. }
```

Response:

```
1. {
2.     "jsonrpc": "2.0",
3.     "result": {
4.         "userids": [
5.             "1",
6.             "2"
7.         ]
8.     },
9.     "id": 1
10. }
```

See also

- [user.addmedia](#)
- [user.deletemedia](#)
- [user.updatemedia](#)
- [Media](#)
- [User](#)

Source

CUser::updateMedia() in *frontends/php/include/classes/api/services/CUser.php*.

user.updateprofile

Description

```
object user.updateprofile(object parameters)
```

This method allows to update the currently logged in user.

This method is deprecated and will be removed in the future. Please use [user.update](#) instead.

Parameters

([object/array](#)) User properties to be updated.

The `userid` property must not be defined. Only the passed properties will be updated, all others will remain unchanged.

Additionally to the [standard user properties](#), the method accepts the following parameters.

Parameter	Type	Description
passwd	string	User's password.
usrgrps	array	User groups to replace existing user groups. The user groups must have the <code>usrgrpid</code> property defined.

Return values

([object](#)) Returns an object containing the ID of the updated user under the `userids` property.

Examples

Renaming the current user

Rename the current user to John Doe.

Request:

```
1. {
2.     "jsonrpc": "2.0",
3.     "method": "user.updateprofile",
4.     "params": {
5.         "name": "John",
6.         "lastname": "Doe"
7.     },
```

user.updateprofile

```
8.     "auth": "038e1d7b1735c6a5436ee9eae095879e",
9.     "id": 1
10. }
```

Response:

```
1. {
2.     "jsonrpc": "2.0",
3.     "result": {
4.         "userids": [
5.             "1"
6.         ],
7.     },
8.     "id": 1
9. }
```

See also

- [user.update](#)

Source

CUser::update() in *frontends/php/include/classes/api/services/CUser.php*.

用户宏

此类用于全局宏的使用。

对象引用：

- [Global macro](#)
- [Host macro](#)

可用的方法：

- [usermacro.create](#) - 创建新的主机宏
- [usermacro.createglobal](#) - 创建新的全局宏
- [usermacro.delete](#) - 删除主机宏
- [usermacro.deleteglobal](#) - 删除全局宏
- [usermacro.get](#) - 检索主机和全局宏
- [usermacro.update](#) - 更新主机宏
- [usermacro.updateglobal](#) - 更新全局宏

> 用户宏对象

以下对象与“usermacro”API直接相关。

全局宏

全局宏对象具有以下属性。

属性	类型	说明
globalmacroid	string	(readonly) 全局宏的ID。
macro (required)	string	宏字符串。
value (required)	string	宏的价值。

主机宏

主机宏对象定义主机或模板上可用的宏。 它具有以下属性。

属性	类型	说明
hostmacroid	string	(readonly) 主机宏的ID。
hostid (required)	string	宏所属主机的ID。
macro (required)	string	宏字符串。
value (required)	string	宏的值。

usermacro.create

说明

```
object usermacro.create(object/array hostMacros)
```

此方法允许创建新的主机宏。

参数

(object/array) 要创建的主机宏。

该方法接受有[标准主机宏属性](#)的主机宏.

返回值

(object) 返回包含“hostMacroids”属性下创建的主机宏的ID的对象。 返回的ID的顺序与传递的主机宏的顺序相匹配。

示例

Creating a host macro

Create a host macro “{\$SNMP_COMMUNITY}” with the value “public” on host “10198”.

Request:

```
1. {
2.     "jsonrpc": "2.0",
3.     "method": "usermacro.create",
4.     "params": {
5.         "hostid": "10198",
6.         "macro": "{$SNMP_COMMUNITY}",
7.         "value": "public"
8.     },
9.     "auth": "038e1d7b1735c6a5436ee9eae095879e",
10.    "id": 1
11. }
```

Response:

```
1. {
2.     "jsonrpc": "2.0",
3.     "result": {
4.         "hostmacroids": [
5.             "11"
6.         ]
7.     }
8. }
```

```
7.     },
8.     "id": 1
9. }
```

来源

CUserMacro::create() in *frontends/php/include/classes/api/services/CUserMacro.php*.

usermacro.createglobal

说明

```
object usermacro.createglobal(object/array globalMacros)
```

此方法允许创建新的全局宏。

参数

(object/array) 要创建的全局宏。

该方法接受具有[标准全局宏属性](#)的全局宏。

返回值

(object) 返回包含“globalmacroids”属性下创建的全局宏的ID的对象。返回的ID的顺序与传递的全局宏的顺序相匹配。

示例

Creating a global macro

Create a global macro “{\$SNMP_COMMUNITY}” with value “public”.

Request:

```
1. {
2.     "jsonrpc": "2.0",
3.     "method": "usermacro.createglobal",
4.     "params": {
5.         "macro": "{$SNMP_COMMUNITY}",
6.         "value": "public"
7.     },
8.     "auth": "038e1d7b1735c6a5436ee9eae095879e",
9.     "id": 1
10. }
```

Response:

```
1. {
2.     "jsonrpc": "2.0",
3.     "result": {
4.         "globalmacroids": [
5.             "6"
6.         ]
7.     },
8. }
```

```
8.      "id": 1  
9. }
```

来源

CUserMacro::createGlobal() in
frontends/php/include/classes/api/services/CUserMacro.php.

usermacro.delete

说明

```
object usermacro.delete(array hostMacroIds)
```

此方法允许删除主机宏。

参数

(array) 要删除的主机宏的ID。

返回值

(object) 返回一个包含“hostMacs”属性下删除的主机宏ID的对象。

示例

Deleting multiple host macros

Delete two host macros.

Request:

```
1. {
2.     "jsonrpc": "2.0",
3.     "method": "usermacro.delete",
4.     "params": [
5.         "32",
6.         "11"
7.     ],
8.     "auth": "3a57200802b24cda67c4e4010b50c065",
9.     "id": 1
10. }
```

Response:

```
1. {
2.     "jsonrpc": "2.0",
3.     "result": {
4.         "hostmacroids": [
5.             "32",
6.             "11"
7.         ]
8.     },
9.     "id": 1
10. }
```

来源

CUserMacro::delete() in *frontends/php/include/classes/api/services/CUserMacro.php*.

usermacro.deleteglobal

说明

```
object usermacro.deleteglobal(array globalMacroIds)
```

此方法允许删除全局宏。

参数

(array) 要删除的全局宏的ID。

返回值

(object) 返回包含“globalmacroids”属性下删除的全局宏ID的对象。

示例

Deleting multiple global macros

Delete two global macros.

Request:

```
1. {
2.   "jsonrpc": "2.0",
3.   "method": "usermacro.deleteglobal",
4.   "params": [
5.     "32",
6.     "11"
7.   ],
8.   "auth": "3a57200802b24cda67c4e4010b50c065",
9.   "id": 1
10. }
```

Response:

```
1. {
2.   "jsonrpc": "2.0",
3.   "result": {
4.     "globalmacroids": [
5.       "32",
6.       "11"
7.     ]
8.   },
9.   "id": 1
10. }
```

来源

```
CUserMacro::deleteGlobal() in  
frontends/php/include/classes/api/services/CUserMacro.php.
```

usermacro.get

说明

`integer/array usermacro.get(object parameters)` 该方法允许根据给定的参数检索主机和全局宏。

参数

(object) 定义所需输出的参数。该方法支持以下参数。

属性	类型	说明
globalmacro	flag	返回全局宏而不是主机宏。
globalmacroids	string/array	仅返回具有给定ID的全局宏。
groupids	string/array	只返回属于主机的主机宏或来自给定主机组的模板。
hostids	string/array	仅返回属于给定主机的主机宏。
hostmacroids	string/array	只返回具有给定ID的主机宏。
templateids	string/array	只返回属于给定模板的主机宏。
selectGroups	query	在“groups”属性中返回主机宏所属的主机组。 仅在检索主机宏时使用。
selectHosts	query	在“hosts”属性中返回主机宏所属的主机。 仅在检索主机宏时使用。
selectTemplates	query	

- 一组对象；
- 如果已经使用“countOutput”参数，则检索到的对象的计数。

示例

Retrieving host macros for a host

Retrieve all host macros defined for host “10198”.

Request:

```

1. {
2.   "jsonrpc": "2.0",
3.   "method": "usermacro.get",
4.   "params": {
5.     "output": "extend",
6.     "hostids": "10198"
7.   },
8.   "auth": "038e1d7b1735c6a5436ee9eae095879e",
9.   "id": 1

```

usermacro.get

```
10. }
```

Response:

```
1. {
2.     "jsonrpc": "2.0",
3.     "result": [
4.         {
5.             "hostmacroid": "9",
6.             "hostid": "10198",
7.             "macro": "{$INTERFACE}",
8.             "value": "eth0"
9.         },
10.        {
11.            "hostmacroid": "11",
12.            "hostid": "10198",
13.            "macro": "{$SNMP_COMMUNITY}",
14.            "value": "public"
15.        }
16.    ],
17.    "id": 1
18. }
```

Retrieving global macros

Retrieve all global macros.

Request:

```
1. {
2.     "jsonrpc": "2.0",
3.     "method": "usermacro.get",
4.     "params": {
5.         "output": "extend",
6.         "globalmacro": true
7.     },
8.     "auth": "038e1d7b1735c6a5436ee9eae095879e",
9.     "id": 1
10. }
```

Response:

```
1. {
2.     "jsonrpc": "2.0",
3.     "result": [
4.         {
5.             "globalmacroid": "6",
6.             "macro": "{$SNMP_COMMUNITY}",
7.             "value": "public"
8.         }
9.     ]
10. }
```

```
9.      ],
10.     "id": 1
11. }
```

来源

CUserMacro::get() in *frontends/php/include/classes/api/services/CUserMacro.php*.

usermacro.update

说明

```
object usermacro.update(object/array hostMacros)
```

此方法允许更新现有的主机宏。

参数

(object/array) 要更新的主机宏属性。

必须为每个主机宏定义“hostmacroid”属性，所有其他属性都是可选的。 只有通过的属性将被更新，所有其他属性将保持不变。

返回值

(object) 返回包含“hostMacroids”属性下更新的主机宏的ID的对象。

示例

Changing the value of a host macro

Change the value of a host macro to “public”.

Request:

```
1. {
2.     "jsonrpc": "2.0",
3.     "method": "usermacro.update",
4.     "params": {
5.         "hostmacroid": "1",
6.         "value": "public"
7.     },
8.     "auth": "038e1d7b1735c6a5436ee9eae095879e",
9.     "id": 1
10. }
```

Response:

```
1. {
2.     "jsonrpc": "2.0",
3.     "result": {
4.         "hostmacroids": [
5.             "1"
6.         ]
7.     },
8. }
```

usermacro.update

```
8.      "id": 1  
9. }
```

来源

CUserMacro::update() in *frontends/php/include/classes/api/services/CUserMacro.php*.

usermacro.updateglobal

说明

```
object usermacro.updateglobal(object/array globalMacros)
```

此方法允许更新现有的全局宏。

参数

(object/array) 要更新的全局宏属性。

必须为每个全局宏定义“globalmacroid”属性，所有其他属性都是可选的。 只有通过的属性将被更新，所有其他属性将保持不变。

返回值

(object) 返回包含“globalmacroids”属性下更新的全局宏的ID的对象。

示例

Changing the value of a global macro

Change the value of a global macro to “public”.

Request:

```
1. {
2.     "jsonrpc": "2.0",
3.     "method": "usermacro.updateglobal",
4.     "params": {
5.         "globalmacroid": "1",
6.         "value": "public"
7.     },
8.     "auth": "038e1d7b1735c6a5436ee9eae095879e",
9.     "id": 1
10. }
```

Response:

```
1. {
2.     "jsonrpc": "2.0",
3.     "result": {
4.         "globalmacroids": [
5.             "1"
6.         ]
7.     },
8. }
```

```
8.      "id": 1  
9. }
```

来源

CUserMacro::updateGlobal() in
frontends/php/include/classes/api/services/CUserMacro.php.

用户组

该类用于使用用户组

对象引用：

- [用户组](#)

可用的方法：

- [usergroup.create](#) - 创建新的用户组
- [usergroup.delete](#) - 删除用户组
- [usergroup.get](#) - 检索用户组
- [usergroup.massadd](#) - 向用户组添加权限和用户
- [usergroup.massupdate](#) - 同时更新多个用户组
- [usergroup.update](#) - 更新用户组

> 用户组对象

以下对象与“用户组”API直接相关。

用户组

用户组对象具有以下属性。

属性	类型	说明
usrgrpid	string	(readonly) 用户组的ID。
name(required)	string	用户组的名称
debugmode	integer	是否启用或禁用调试模式。 可能的值： 0 - (默认) 禁用； 1 - 启用。
gui_access	integer	组中用户的前端身份验证方法。 可能的值： 0 - (默认) 使用系统默认身份验证方法； 1 - 使用内部认证； 2 - 禁止访问前端。
users_status	integer	用户组是启用还是禁用。 可能的值： 0 - (默认) 启用； 1 - 禁用。

权限

权限对象具有以下属性。

属性	类型	说明
id(required)	string	要添加权限的主机组的ID。
permission(required)	integer	访问到主机组的级别。可能的值： 0 - 拒绝访问； 2 - 只读访问； 3 - 读写访问。

usergroup.create

说明

`object usergroup.create(object/array userGroups)` 此方法允许创建新的用户组。

参数

`(object/array)` 要创建的用户组。

除了[标准用户组属性](#)之外，该方法接受以下参数。

属性	类型	说明
<code>rights</code>	<code>object/array</code>	分配给组的权限
<code>userids</code>	<code>string/array</code>	要添加到用户组的用户的ID。

返回值

`(object)` 返回包含“usrgrpid”属性下创建的用户组的ID的对象。返回的ID的顺序与传递的用户组的顺序相匹配。

示例

Creating a user group

Create a user group, which denies access to host group “2”， and add a user to it.

Request:

```

1. {
2.   "jsonrpc": "2.0",
3.   "method": "usergroup.create",
4.   "params": {
5.     "name": "Operation managers",
6.     "rights": {
7.       "permission": 0,
8.       "id": "2"
9.     },
10.    "userids": "12"
11.  },
12.  "auth": "038e1d7b1735c6a5436ee9eae095879e",
13.  "id": 1
14. }
```

Response:

```
1.  {
2.      "jsonrpc": "2.0",
3.      "result": {
4.          "usrgrpids": [
5.              "20"
6.          ],
7.      },
8.      "id": 1
9. }
```

参见

- [Permission](#)

来源

`CUserGroup::create()` in *frontends/php/include/classes/api/services/CUserGroup.php*.

usergroup.delete

说明

```
object usergroup.delete(array userGroupIds)
```

此方法允许删除用户组。

参数

(array) 要删除的用户组的ID。

返回值

(object) 返回包含“usrgrpid”属性下删除的用户组的ID的对象。

示例

Deleting multiple user groups

Delete two user groups.

Request:

```
1. {
2.     "jsonrpc": "2.0",
3.     "method": "usergroup.delete",
4.     "params": [
5.         "20",
6.         "21"
7.     ],
8.     "auth": "3a57200802b24cda67c4e4010b50c065",
9.     "id": 1
10. }
```

Response:

```
1. {
2.     "jsonrpc": "2.0",
3.     "result": {
4.         "usrgrpid": [
5.             "20",
6.             "21"
7.         ]
8.     },
9.     "id": 1
10. }
```

来源

CUserGroup::delete() in *frontends/php/include/classes/api/services/CUserGroup.php*.

usergroup.get

说明

```
integer/array usergroup.get(object parameters)
```

该方法允许根据给定的参数检索用户组。

参数

(object) 定义所需输出的参数。

该方法支持以下参数。

属性	类型	说明
status	integer	只返回具有给定状态的用户组。 请参阅 用户组页面 以获取支持的状态列表。
userids	string/array	只返回包含给定用户的用户组。
usrgrpids	string/array	只返回具有给定ID的用户组。
with_gui_access	integer	只返回具有给定前端身份验证方法的用户组。 有关支持的方法的列表，请参阅 用户组页面 。
selectUsers	query	在“users”属性中返回用户组中的用户。
selectRights	query	在“权限”属性中返回用户组权限。 它具有以下属性： 权限 - (整数) 访问级别到主机组； id - (string) 主机组的ID。 有关主机组的访问级别列表，请参阅 用户组页面 。
limitSelects	integer	限制子选择返回的记录数。
sortfield	string/array	按照给定的属性对结果进行排序。 可能的值为： usrgrpid , name 。
countOutput	flag	参考文献 中详细描述了所有“获得”方法的常用参数。 .
editable	boolean	
excludeSearch	flag	
filter	object	
limit	integer	
output	query	
preservekeys	flag	
search	object	
searchByAny	boolean	
searchWildcardsEnabled	boolean	
sortorder	string/array	
startSearch	flag	

返回值

(integer/array) 返回:

- 一组对象；
- 如果已经使用“countOutput”参数，则检索到的对象的计数。

示例

Retrieving enabled user groups

Retrieve all enabled user groups.

Request:

```

1. {
2.     "jsonrpc": "2.0",
3.     "method": "usergroup.get",
4.     "params": {
5.         "output": "extend",
6.         "status": 0
7.     },
8.     "auth": "038e1d7b1735c6a5436ee9eae095879e",
9.     "id": 1
10. }
```

Response:

```

1. {
2.     "jsonrpc": "2.0",
3.     "result": [
4.         {
5.             "usrgrpid": "7",
6.             "name": "Zabbix administrators",
7.             "gui_access": "0",
8.             "users_status": "0",
9.             "debug_mode": "1"
10.        },
11.        {
12.            "usrgrpid": "8",
13.            "name": "Guests",
14.            "gui_access": "0",
15.            "users_status": "0",
16.            "debug_mode": "0"
17.        },
18.        {
19.            "usrgrpid": "11",
20.            "name": "Enabled debug mode",
```

```
21.         "gui_access": "0",
22.         "users_status": "0",
23.         "debug_mode": "1"
24.     },
25.     {
26.         "usrgrpid": "12",
27.         "name": "No access to the frontend",
28.         "gui_access": "2",
29.         "users_status": "0",
30.         "debug_mode": "0"
31.     },
32.     {
33.         "usrgrpid": "14",
34.         "name": "Read only",
35.         "gui_access": "0",
36.         "users_status": "0",
37.         "debug_mode": "0"
38.     },
39.     {
40.         "usrgrpid": "18",
41.         "name": "Deny",
42.         "gui_access": "0",
43.         "users_status": "0",
44.         "debug_mode": "0"
45.     }
46. ],
47. "id": 1
48. }
```

参见

- [User](#)

来源

CUserGroup::get() in *frontends/php/include/classes/api/services/CUserGroup.php*.

usergroup.massadd

说明

`object usergroup.massadd(object parameters)`

此方法允许同时向多个用户组添加权限和用户。

此方法已被弃用，将来会被删除。请改用 [usergroup.update](#)。

参数

(object) 包含要更新的用户组的ID以及要添加的权限和用户的参数。

该方法接受以下参数。

属性	类型	说明
<code>usrgrpids</code> (required)	string/array	要更新的用户组的ID。
<code>rights</code>	object/array	分配给用户组的权限。
<code>userids</code>	string/array	要添加到用户组的用户的ID。

返回值

(object) 返回包含“usrgrpids”属性下更新的用户组的ID的对象。

示例

Denying access to host group

Deny two user groups access to host group “2”.

Request:

```

1. {
2.   "jsonrpc": "2.0",
3.   "method": "usergroup.massadd",
4.   "params": {
5.     "usrgrpids": [
6.       "17",
7.       "19"
8.     ],
9.     "rights": {
10.       "permission": 0,
11.       "id": "2"
12.     }
13.   },

```

```
14.     "auth": "038e1d7b1735c6a5436ee9eae095879e",
15.     "id": 1
16. }
```

Response:

```
1. {
2.     "jsonrpc": "2.0",
3.     "result": {
4.         "usrgrpids": [
5.             "17",
6.             "19"
7.         ],
8.     },
9.     "id": 1
10. }
```

参见

- [Permission](#)
- [usergroup.massupdate](#)
- [usergroup.update](#)

来源

CUserGroup::massAdd() in *frontends/php/include/classes/api/services/CUserGroup.php*.

usergroup.massupdate

说明

```
object usergroup.massupdate(object parameters)
```

此方法允许同时更新多个用户组的属性，用户或权限。

此方法已被弃用，将来会被删除。请改用[usergroup.update](#)

参数

(object) 包含要更新的用户组的ID和应更新的属性的参数。

除了 [标准用户组属性](#)，该方法接受以下参数。

属性	类型	说明
usrgrpids(required)	string/array	要更新的用户组的ID。
rights	string/array	更改分配给用户组的当前权限的权限。
userids	object/array	用户的ID替换组中的用户。

返回值

(object) 返回包含“usrgrpids”属性下更新的用户组的ID的对象。

示例

Changing permissions for a user group

Update the permissions for two user groups to only allow read-write access to two host groups.

Request:

```

1. {
2.   "jsonrpc": "2.0",
3.   "method": "usergroup.massupdate",
4.   "params": {
5.     "usrgrpids": [
6.       "17",
7.       "19"
8.     ],
9.     "rights": [
10.      {
11.        "permission": 3,
12.        "id": "2"
13.      }
14.    ]
15.  }
16. }
```

```
13.         },
14.         {
15.             "permission": 3,
16.             "id": "3"
17.         }
18.     ],
19. },
20. "auth": "038e1d7b1735c6a5436ee9eae095879e",
21. "id": 1
22. }
```

Response:

```
1. {
2.     "jsonrpc": "2.0",
3.     "result": {
4.         "usrgrpids": [
5.             "17",
6.             "19"
7.         ],
8.     },
9.     "id": 1
10. }
```

参见

- [Permission](#)
- [usergroup.massadd](#)
- [usergroup.update](#)

来源

CUserGroup::massUpdate() in *frontends/php/include/classes/api/services/CUserGroup.php*.

usergroup.update

说明

`object usergroup.update(object/array userGroups)`

此方法允许更新现有的用户组。

参数

`(object/array)` 要更新的用户组属性。

必须为每个用户组定义“usrgrpid”属性，所有其他属性都是可选的。 只有通过的属性将被更新，所有其他属性将保持不变。

除了[标准用户组属性](#)之外，该方法接受以下参数。

属性	类型	说明
rights	object/array	更改分配给用户组的当前权限的权限。
userids	string/array	用户的ID替换组中的用户。

返回值

`(object)` 返回包含“usrgrpid”属性下更新的用户组的ID的对象。

示例

Disabling a user group

Disable a user group.

Request:

```

1. {
2.   "jsonrpc": "2.0",
3.   "method": "usergroup.update",
4.   "params": {
5.     "usrgrpid": "17",
6.     "users_status": "1"
7.   },
8.   "auth": "038e1d7b1735c6a5436ee9eae095879e",
9.   "id": 1
10. }
```

Response:

```
1. {
2.     "jsonrpc": "2.0",
3.     "result": {
4.         "usrgrpids": [
5.             "17"
6.         ],
7.     },
8.     "id": 1
9. }
```

参见

- [Permission](#)
- [usergroup.massadd](#)
- [usergroup.massupdate](#)

来源

CUserGroup::update() in *frontends/php/include/classes/api/services/CUserGroup.php*.

监控项

此类用于管理监控项。

对象引用：

- [Item](#)

可用的方法：

- [item.create](#) - 创建新的监控项
- [item.delete](#) - 删除监控项
- [item.get](#) - 检索监控项
- [item.update](#) - 更新监控项

> Item object

以下对象与“item” API直接相关。

监控项

Web 监控项无法通过 Zabbix API 直接创建，更新或删除。

监控项对象具有以下属性。

属性	类型	说明
itemid	string	(只读) 监控项ID
delay(required)	string	更新监控项的间隔。 接受具有后缀的秒或时间单位，并且具有或不具有由灵活间隔和调度间隔组成的一个或多个自定义间隔作为串行化字符串。也接受用户宏。灵活的间隔可以写成两个由正斜杠分隔的宏。间隔用分号分隔。
hostid(required)	string	该项所属的主机 ID。
interfaceid(required)	string	ID of the item's host interface. Used only for host items. 项主机接口的ID。仅用于主机项。Optional for Zabbix agent (active), Zabbix internal, Zabbix trapper, Zabbix aggregate, database monitor and calculated items. 适用于Zabbix代理(活动)，Zabbix内部，Zabbix陷阱，Zabbix聚合，数据库监控和计算项。
key_(required)	string	Item key.
name(required)	string	Name of the item. item的名称
type(required)	integer	Type of the item. item的类别Possible values: 可能的值: 0 - Zabbix agent; 1 - SNMPv1 agent; 2 - Zabbix trapper; 3 - simple check; 4 - SNMPv2 agent; 5 - Zabbix internal; 6 - SNMPv3 agent; 7 - Zabbix agent (active); 8 - Zabbix aggregate; 9 - web item; 10 - external check; 11 - database monitor; 12 - IPMI agent; 13 - SSH agent; 14 - TELNET agent; 15 - calculated; 16 - JMX agent; 17 - SNMP trap.
value_type(required)	integer	Type of information of the item. item的信息类型。Possible values: 可能的值: 0 - numeric float; 1 - character; 2 - log; 3 - numeric unsigned; 4 - text.
authtype	integer	SSH authentication method. Used only by SSH agent items. SSH认证方式。仅用于SSH代理项Possible values: 可能的值: 0 - (default) password; 1 - public key.
datatype	integer	Data type of the item. item的数据类型 Possible values: 可能的值: 0 - (default) decimal; 1 - octal; 2 - hexadecimal; 3 - boolean.
		Value that will be stored. 将被存储的值

delta	integer	Possible values: 可能的值: 0 - (default) as is; 1 - Delta, speed per second; 2 - Delta, simple change.
description	string	Description of the item.item说明
error	string	(readonly) Error text if there are problems updating the item. 如果更新项目时出现问题，则显示错误文本。
flags	integer	(readonly) Origin of the item. item的来源 Possible values: 可能的值0 - a plain item; 4 - a discovered item.
formula	integer/float	Custom multiplier. Default: 1.
history	string	A time unit of how long the history data should be stored. Also accepts user macro. 历史数据应该存储多长时间的单位。也接受用户宏。 Default: 90d.
inventory_link	integer	ID of the host inventory field that is populated by the item. 由item填充的主机库存字段的ID。Refer to the host inventory page for a list of supported host inventory fields and their IDs. 请参阅主机清单页面以获取支持的主机清单字段及其ID的列表。 Default: 0.
ipmi_sensor	string	IPMI sensor. Used only by IPMI items. IPMI 传感器 仅由IPMI项使用。
lastclock	timestamp	(readonly) Time when the item was last updated. item上次更新时间。This property will only return a value for the period configured in ZBX_HISTORY_PERIOD . 此属性将仅返回ZBX_HISTORY_PERIOD中配置的时间段的值
lastns	integer	(readonly) Nanoseconds when the item was last updated. item上次更新时的纳秒This property will only return a value for the period configured in ZBX_HISTORY_PERIOD . 此属性将仅返回ZBX_HISTORY_PERIOD中配置的时间段的值。
lastvalue	string	(readonly) Last value of the item. item的最后一个值。This property will only return a value for the period configured in ZBX_HISTORY_PERIOD . 此属性将仅返回ZBX_HISTORY_PERIOD中配置的时间段的值
logtimefmt	string	Format of the time in log entries. Used only by log items. 日志条目中的时间格式。仅用于日志项。
mtime	timestamp	Time when the monitored log file was last updated. Used only by log items. 受监控日志文件上次更新的时间。仅用于日志项。
multiplier	integer	Whether to use a custom multiplier. 是否使用自定义乘数。
params	string	Additional parameters depending on the type of the item: 附加参数取决于item的类型: - executed script for SSH and Telnet items; SSH和Telnet items执行脚本; - SQL query for database monitor items; SQL查询数据库监视项- formula for calculated items. 计算项公式

password	string	Password for authentication. Used by simple check, SSH, Telnet, database monitor and JMX items. 认证密码。用于简单检查, SSH, Telnet, 数据库监控和JMX项
port	string	Port monitored by the item. Used only by SNMP items. 由项监视的端口。仅由SNMP项使用。
prevvalue	string	(readonly) Previous value of the item. item的上一个值。This property will only return a value for the period configured in ZBX_HISTORY_PERIOD . 此属性将仅返回 ZBX_HISTORY_PERIOD 中配置的时间段的值
privatekey	string	Name of the private key file. 私钥文件的名称
publickey	string	Name of the public key file. 公钥文件的名称
snmp_community	string	SNMP community. Used only by SNMPv1 and SNMPv2 items. SNMP社区。仅由SNMPv1和SNMPv2项使用。
snmp_oid	string	SNMP OID.
snmpv3_authpassphrase	string	SNMPv3 auth passphrase. Used only by SNMPv3 items. 仅用于SNMPv3项。
snmpv3_authprotocol	integer	SNMPv3 authentication protocol. Used only by SNMPv3 items. SNMPv3认证协议 仅用于SNMPv3项。Possible values: 可能的值 0 - (default) MD5; 1 - SHA.
snmpv3_contextname	string	SNMPv3 context name. Used only by SNMPv3 items. SNMPv3上下文名称。仅用于SNMPv3项
snmpv3_privpassphrase	string	SNMPv3 priv passphrase. Used only by SNMPv3 items. SNMPv3 priv密码。仅用于SNMPv3项
snmpv3_privprotocol	integer	SNMPv3 privacy protocol. Used only by SNMPv3 items. SNMPv3隐私协议。仅用于SNMPv3项 Possible values: 可能的值: 0 - (default) DES; 1 - AES.
snmpv3_securitylevel	integer	SNMPv3 security level. Used only by SNMPv3 items. SNMPv3安全级别。仅用于SNMPv3项 Possible values: 可能的值: 0 - noAuthNoPriv; 1 - authNoPriv; 2 - authPriv.
snmpv3_securityname	string	SNMPv3 security name. Used only by SNMPv3 items. SNMPv3安全名称。仅用于SNMPv3项
state	integer	(readonly) State of the item. item状态。 Possible values: 可能的值: 0 - (default) normal; 1 - not supported.
status	integer	Status of the item. item状态 Possible values: 可能的值: 0 - (default) _ enabled item; 1 - disabled item.
templateid	string	(readonly) ID of the parent template item. 父模板项的ID。
trapper_hosts	string	Allowed hosts. Used only by trapper items. 允许的主机。仅由trapper items使用。
trends	string	A time unit of how long the trends data should be stored. Also accepts user macro. 趋势数据应存储多长时间的单位。也接受用户宏。 Default: 365d.

units	string	Value units.
username	string	Username for authentication. Used by simple check, SSH, Telnet, database monitor and JMX items. 验证用户名。 用于简单检查, SSH, Telnet, 数据库监控和JMX项Required by SSH and Telnet items. 需要SSH和Telnet项目。
valuemapid	string	ID of the associated value map. 相关值图的 ID。

item.create

Description 说明

```
object item.create(object/array items)
```

This method allows to create new items.此方法允许创建新的项。

Web items cannot be created via the Zabbix API.无法通过Zabbix API创建Web items。

Parameters参数

(object/array) Items to create.要创建的items

Additionally to the standard item properties, the method accepts the following parameters.除了项属性之外，该方法接受以下参数。

属性	类型	说明
applications	array	IDs of the applications to add the item to. 要添加项的应用程序的ID。

Return values 返回值

(object) Returns an object containing the IDs of the created items under the itemids property. The order of the returned IDs matches the order of the passed items.返回包含“itemids”属性下创建项的ID的对象。返回的ID的顺序与传递的项的顺序相匹配。

Examples 示例

Creating an item 创建一个item

Create a numeric Zabbix agent item to monitor free disk space on host with ID “30074” and add it to two applications.创建一个数字Zabbix代理项，以监控ID为“30074”的主机上的可用磁盘空间，并将其添加到两个应用程序中。

Request:

```

1. {
2.   "jsonrpc": "2.0",
3.   "method": "item.create",
4.   "params": {
5.     "name": "Free disk space on $1",
6.     "key_": "vfs.fs.size[/home/joe/,free]",
7.     "hostid": "30074",
8.     "type": 0,
9.     "value_type": 3,
10.    "interfaceid": "30084",

```

```
item.create
```

```
11.     "applications": [
12.         "609",
13.         "610"
14.     ],
15.     "delay": "30s"
16. },
17. "auth": "038e1d7b1735c6a5436ee9eae095879e",
18. "id": 1
19. }
```

Response:

```
1. {
2.     "jsonrpc": "2.0",
3.     "result": {
4.         "itemids": [
5.             "24758"
6.         ]
7.     },
8.     "id": 1
9. }
```

Creating a host inventory item

Create a Zabbix agent item to populate the host's “OS” inventory field.

Request:

```
1. {
2.     "jsonrpc": "2.0",
3.     "method": "item.create",
4.     "params": {
5.         "name": "uname",
6.         "key_": "system.uname",
7.         "hostid": "30021",
8.         "type": 0,
9.         "interfaceid": "30007",
10.        "value_type": 1,
11.        "delay": "10s",
12.        "inventory_link": 5
13.    },
14.    "auth": "038e1d7b1735c6a5436ee9eae095879e",
15.    "id": 1
16. }
```

Response:

```
1. {
2.     "jsonrpc": "2.0",
3.     "result": {
```

item.create

```
4.      "itemids": [
5.          "24759"
6.      ],
7.      },
8.      "id": 1
9. }
```

Source 来源

CItem::create() in *frontends/php/include/classes/api/services/CItem.php*.

item.delete

Description 说明

```
object item.delete(array itemIds)
```

This method allows to delete items.此方法允许删除项目。

Web items cannot be deleted via the Zabbix API. Web项无法通过Zabbix API进行删除。

Parameters 参数

(array) IDs of the items to delete.要删除的项的ID。

Return values 返回值

(object) Returns an object containing the IDs of the deleted items under the itemids property.返回包含“itemid”属性下的已删除项的ID的对象

Examples 示例

Deleting multiple items 删除多个items

Delete two items.删除两个items

Request:

```
1. {
2.     "jsonrpc": "2.0",
3.     "method": "item.delete",
4.     "params": [
5.         "22982",
6.         "22986"
7.     ],
8.     "auth": "3a57200802b24cda67c4e4010b50c065",
9.     "id": 1
10. }
```

Response:

```
1. {
2.     "jsonrpc": "2.0",
3.     "result": {
4.         "itemids": [
5.             "22982",
6.             "22986"
7.         ]
8.     }
9. }
```

```
item.delete
```

```
8.      },
9.      "id": 1
10. }
```

Source 来源

`CItem::delete()` in `frontends/php/include/classes/api/services/CItem.php`.

item.get

说明

```
integer/array item.get(object parameters)
```

该方法允许根据给定的参数检索监控项。

参数

(object) 定义所需输出的参数。

该方法支持以下参数。

属性	类型	说明
itemids	string/array	只返回具有给定 ID 的监控项
groupids	string/array	只返回属于给定组的主机的监控项
templateids	string/array	仅返回属于给定模板的监控项
hostids	string/array	仅返回属于给定主机的监控项
proxyids	string/array	仅返回由给定代理监视的监控项
interfaceids	string/array	仅返回使用给定主机接口的监控项
graphids	string/array	仅返回在给定图表中使用的监控项
triggerids	string/array	仅返回在给定触发器中使用的监控项
applicationids	string/array	仅返回属于给定应用程序的监控项
webitems	flag	在结果中包含 web 监控项。
inherited	boolean	如果设置为“True”，只返回从模板中承接的项。
templated	boolean	如果设置为“True”，则只返回属于模板的项
monitored	boolean	如果设置为“True”，则仅返回属于受监控主机的已启用项
group	string	仅返回属于具有给定名称的组的项
host	string	仅返回属于具有给定名称的主机的项
application	string	仅返回属于具有给定名称的应用程序的项
withtriggers	boolean	如果设置为“true”，则只返回在触发器中使用的监控项
selectHosts	query	将该项所属的主机作为“hosts”属性中的数组返回
selectInterfaces	query	将项使用的主机接口作为“interfaces”属性中的数组返回。
selectTriggers	query	在“'触发器”属性中，返回该项使用的触发。 Supports <code>count</code> .
selectGraphs	query	在“图形”属性中返回包含该项的图形。 Supports <code>count</code> .
selectApplications	query	在“应用程序”属性中返回该项所属的应用程序
selectDiscoveryRule	query	返回在“discoveryRule”属性中创建该项的LLD规则

selectItemDiscovery	query	<p>在“itemDiscovery”属性中返回项目发现对象。项发现对象将项链接到从其创建的项原型。它具有以下属性：</p> <ul style="list-style-type: none"> <code>itemid</code> - (string) item discovery的ID <code>itemid</code> - (string) 已发现Item的ID <code>parentitemid</code> - (string) 已经创建项的项原型的ID; <code>key</code> - (string) key of the item prototype; - (timestamp) 最后一次发现item的时间 <code>lastcheck</code> - (timestamp) 不再发现的项将被删除的时间。<code>ts_delete</code> - (timestamp)
filter	object	仅返回与给定过滤器完全匹配的结果。接受一个数组，其中 <code>keys</code> 是属性名称，并且值是单个值或要匹配的值的数组。支持附加的过滤器： <code>host</code> - 该项所属主机的技术名称。
limitSelects	integer	限制子选择返回的记录数。适用于以下子选项： <code>selectGraphs</code> - 结果将按“'name"排序； <code>selectTriggers</code> - results will be sorted by <code>description</code> . 结果将按“'description"排序。
sortfield	string/array	

示例

通过 Key 查询 Items

从key中具有“system”一词的ID为“10084”的主机检索所有项，并按名称进行排序。

Request:

```

1. {
2.   "jsonrpc": "2.0",
3.   "method": "item.get",
4.   "params": {
5.     "output": "extend",
6.     "hostids": "10084",
7.     "search": {
8.       "key_": "system"
9.     },
10.    "sortfield": "name"
11.  },
12.  "auth": "038e1d7b1735c6a5436ee9eae095879e",
13.  "id": 1
14. }
```

Response:

```

1. {
2.   "jsonrpc": "2.0",
3.   "result": [
4.     {
5.       "itemid": "23298",
6.       "type": "0",
7.       "snmp_community": "",
8.       "snmp_oid": "",
9.       "hostid": "10084",
```

item.get

```
10.         "name": "Context switches per second",
11.         "key_": "system.cpu.switches",
12.         "delay": "1m",
13.         "history": "7d",
14.         "trends": "365d",
15.         "lastvalue": "2552",
16.         "lastclock": "1351090998",
17.         "prevvalue": "2641",
18.         "state": "0",
19.         "status": "0",
20.         "value_type": "3",
21.         "trapper_hosts": "",
22.         "units": "sps",
23.         "multiplier": "0",
24.         "delta": "1",
25.         "snmpv3_securityname": "",
26.         "snmpv3_securitylevel": "0",
27.         "snmpv3_authpassphrase": "",
28.         "snmpv3_privpassphrase": "",
29.         "snmpv3_authprotocol": "0",
30.         "snmpv3_privprotocol": "0",
31.         "snmpv3_contextname": "",
32.         "formula": "1",
33.         "error": "",
34.         "lastlogsize": "0",
35.         "logtimefmt": "",
36.         "templateid": "22680",
37.         "valuemapid": "0",
38.         "params": "",
39.         "ipmi_sensor": "",
40.         "data_type": "0",
41.         "authtype": "0",
42.         "username": "",
43.         "password": "",
44.         "publickey": "",
45.         "privatekey": "",
46.         "mtime": "0",
47.         "lastns": "564054253",
48.         "flags": "0",
49.         "interfaceid": "1",
50.         "port": "",
51.         "description": "",
52.         "inventory_link": "0",
53.         "lifetime": "0s",
54.         "evaltype": "0"
55.     },
56.     {
57.         "itemid": "23299",
58.         "type": "0",
59.         "snmp_community": "",
60.         "snmp_oid": "",
61.         "hostid": "10084",
62.         "name": "CPU $2 time",
```

```

63.         "key_": "system.cpu.util[,idle]",
64.         "delay": "1m",
65.         "history": "7d",
66.         "trends": "365d",
67.         "lastvalue": "86.031879",
68.         "lastclock": "1351090999",
69.         "prevvalue": "85.306944",
70.         "state": "0",
71.         "status": "0",
72.         "value_type": "0",
73.         "trapper_hosts": "",
74.         "units": "%",
75.         "multiplier": "0",
76.         "delta": "0",
77.         "snmpv3_securityname": "",
78.         "snmpv3_securitylevel": "0",
79.         "snmpv3_authpassphrase": "",
80.         "snmpv3_privpassphrase": "",
81.         "snmpv3_authprotocol": "0",
82.         "snmpv3_privprotocol": "0",
83.         "snmpv3_contextname": "",
84.         "formula": "1",
85.         "error": "",
86.         "lastlogsize": "0",
87.         "logtimefmt": "",
88.         "templateid": "17354",
89.         "valuemapid": "0",
90.         "params": "",
91.         "ipmi_sensor": "",
92.         "data_type": "0",
93.         "authtype": "0",
94.         "username": "",
95.         "password": "",
96.         "publickey": "",
97.         "privatekey": "",
98.         "mtime": "0",
99.         "lastns": "564256864",
100.        "flags": "0",
101.        "interfaceid": "1",
102.        "port": "",
103.        "description": "The time the CPU has spent doing nothing.",
104.        "inventory_link": "0",
105.        "lifetime": "0s",
106.        "evaltype": "0"
107.    },
108.    {
109.        "itemid": "23300",
110.        "type": "0",
111.        "snmp_community": "",
112.        "snmp_oid": "",
113.        "hostid": "10084",
114.        "name": "CPU $2 time",
115.        "key_": "system.cpu.util[,interrupt]"

```

```

116.         "history": "7d",
117.         "trends": "365d",
118.         "lastvalue": "0.008389",
119.         "lastclock": "1351091000",
120.         "prevvalue": "0.000000",
121.         "state": "0",
122.         "status": "0",
123.         "value_type": "0",
124.         "trapper_hosts": "",
125.         "units": "%",
126.         "multiplier": "0",
127.         "delta": "0",
128.         "snmpv3_securityname": "",
129.         "snmpv3_securitylevel": "0",
130.         "snmpv3_authpassphrase": "",
131.         "snmpv3_privpassphrase": "",
132.         "snmpv3_authprotocol": "0",
133.         "snmpv3_privprotocol": "0",
134.         "snmpv3_contextname": "",
135.         "formula": "1",
136.         "error": "",
137.         "lastlogsize": "0",
138.         "logtimefmt": "",
139.         "templateid": "22671",
140.         "valuemapid": "0",
141.         "params": "",
142.         "ipmi_sensor": "",
143.         "data_type": "0",
144.         "authtype": "0",
145.         "username": "",
146.         "password": "",
147.         "publickey": "",
148.         "privatekey": "",
149.         "mtime": "0",
150.         "lastns": "564661387",
151.         "flags": "0",
152.         "interfaceid": "1",
153.         "port": "",
154.         "description": "The amount of time the CPU has been servicing hardware interrupts.",
155.         "inventory_link": "0",
156.         "lifetime": "0s",
157.         "evaltype": "0"
158.     }
159. ],
160. "id": 1
161. }
```

参见

- [Application](#)
- [Discovery rule](#)

item.get

- [Graph](#)
- [Host](#)
- [Host interface](#)
- [Trigger](#)

来源

`CIItem::get()` in *frontends/php/include/classes/api/services/CIItem.php*.

item.update

Description 说明

```
object item.update(object/array items)
```

This method allows to update existing items.此方法允许更新现有项

Web items cannot be updated via the Zabbix API. Web项无法通过Zabbix API进行更新。

Parameters 参数

(object/array) Item properties to be updated.要更新的 Item属性。

The `itemid` property must be defined for each item, all other properties are optional. Only the passed properties will be updated, all others will remain unchanged.必须为每个项定义“itemid”属性，所有其他属性都是可选的。 只有通过的属性将被更新，所有其他属性将保持不变。

Additionally to the `standard item properties`, the method accepts the following parameters.除了标准项属性之外，该方法接受以下参数。

属性	类型	说明
applications	array	IDs of the applications to replace the current applications.用于替换当前应用程序的应用程序的ID。

Return values 返回值

(object) Returns an object containing the IDs of the updated items under the `itemids` property.返回一个包含“itemids”属性下更新项的ID的对象。

Examples 示例

Enabling an item 启用项

Enable an item, that is, set its status to “0”.启用项，即将其状态设置为“0”。

Request:

```

1. {
2.   "jsonrpc": "2.0",
3.   "method": "item.update",
4.   "params": {
5.     "itemid": "10092",
6.     "status": 0
7.   },
8.   "auth": "700ca65537074ec963db7efabda78259",
9.   "id": 1

```

item.update

```
10. }
```

Response:

```
1. {
2.     "jsonrpc": "2.0",
3.     "result": {
4.         "itemids": [
5.             "10092"
6.         ],
7.     },
8.     "id": 1
9. }
```

Source 来源

CIItem::update() in *frontends/php/include/classes/api/services/CIItem.php*.

聚合图形

此类用于聚合图形的使用

对象引用：

- [Screen](#)
- [Screen user](#)
- [Screen user group](#)

可用的方法：

- [screen.create](#) - 创建新的聚合图形
- [screen.delete](#) - 删除聚合图形
- [screen.get](#) - 检索聚合图形
- [screen.update](#) - 更新聚合图形

> Screen对象

以下对象与“Screen”API直接相关。

Screen

Screen对象具有以下属性。

属性	类型	说明
screenid	string	(readonly) screen的ID.
name(required)	string	screen的名称.
hsize	integer	screen的宽度. Default: 1
vsize	integer	screen的高度. Default: 1
userid	string	Screen所有者用户ID.
private	integer	screen sharing的类型. 可能的值: 0 - 公共screen; 1 - (default) 私有 screen.

Screen 用户

基于用户的Screen权限列表。 它具有以下属性：

属性	类型	说明
screenuserid	string	(readonly) Screen 用户的ID
userid(required)	string	用户 ID.
permission(required)	integer	权限级别类型。 \可能的值: 2 - 只读; 3 - 读写;

Screen 用户组

基于用户组的Screen权限列表。 它具有以下属性：

属性	类型	说明
screenusrgrpId	string	(readonly) screen用户组的ID
usrgrpId(required)	string	用户组ID。
permission(required)	integer	权限级别类型。 可能的值: 2 - 只读; 3 - 读写;

screen.create

说明

```
object screen.create(object/array screens)
```

该方法允许创建新screen。

参数

(object/array) Screens to create.

除了 [标准screen属性](#)之外，该方法接受以下参数。

属性	类型	说明
screenitems	array	要为screen创建的Screen items
users	array	在screen上创建screen用户共享。
userGroups	array	在screen上创建screen用户组共享。

返回值

(object) 返回一个包含“screenid”属性下创建的screen ID的对象。返回的ID的顺序与传递的screen的顺序相匹配。

示例

Creating a screen

Create a screen named “Graphs” with 2 rows and 3 columns and add a graph to the upper-left cell.

Request:

```

1. {
2.   "jsonrpc": "2.0",
3.   "method": "screen.create",
4.   "params": {
5.     "name": "Graphs",
6.     "hsize": 3,
7.     "vsize": 2,
8.     "screenitems": [
9.       {
10.         "resourcetype": 0,
11.         "resourceid": "612",
12.         "rowspan": 0,
13.         "colspan": 0,

```

```
screen.create
```

```
14.          "x": 0,
15.          "y": 0
16.        }
17.      ]
18.    },
19.    "auth": "038e1d7b1735c6a5436ee9eae095879e",
20.    "id": 1
21. }
```

Response:

```
1. {
2.   "jsonrpc": "2.0",
3.   "result": {
4.     "screenids": [
5.       "26"
6.     ]
7.   },
8.   "id": 1
9. }
```

Screen共享

Create a screen with two types of sharing (user and user group).

Request:

```
1. {
2.   "jsonrpc": "2.0",
3.   "method": "screen.create",
4.   "params": {
5.     "name": "Screen sharing",
6.     "hsize": 3,
7.     "vsize": 2,
8.     "users": [
9.       {
10.         "userid": "4",
11.         "permission": "3"
12.       }
13.     ],
14.     "userGroups": [
15.       {
16.         "usrgrpid": "7",
17.         "permission": "2"
18.       }
19.     ]
20.   },
21.   "auth": "038e1d7b1735c6a5436ee9eae095879e",
22.   "id": 1
23. }
```

Response:

```
1. {
2.     "jsonrpc": "2.0",
3.     "result": {
4.         "screenids": [
5.             "83"
6.         ]
7.     },
8.     "id": 1
9. }
```

参见

- [Screen item](#)
- [Screen user](#)
- [Screen user group](#)

来源

`CScreen::create()` in *frontends/php/include/classes/api/services/CScreen.php*.

screen.delete

说明

```
object screen.delete(array screenIds)
```

此方法允许删除screen。

参数

(array) 要删除的screens的ID。

返回值

(object) 返回一个包含“screenid”属性下删除的screen ID的对象。

示例7

Deleting multiple screens

Delete two screens.

Request:

```
1. {
2.     "jsonrpc": "2.0",
3.     "method": "screen.delete",
4.     "params": [
5.         "25",
6.         "26"
7.     ],
8.     "auth": "3a57200802b24cda67c4e4010b50c065",
9.     "id": 1
10. }
```

Response:

```
1. {
2.     "jsonrpc": "2.0",
3.     "result": {
4.         "screenids": [
5.             "25",
6.             "26"
7.         ]
8.     },
9.     "id": 1
10. }
```

来源

`CScreen::delete() in frontends/php/include/classes/api/services/CScreen.php.`

screen.get

说明

```
integer/array screen.get(object parameters)
```

该方法允许根据给定的参数检索screen。

参数

(object) 定义所需输出的参数。该方法支持以下参数。

属性	类型	说明
screenids	string/array	只返回具有给定ID的screen。
userids	string/array	仅返回属于给定用户ID的screen
screenitemids	string/array	只返回包含给定screen item的屏幕。
selectUsers	query	返回用户在 users 属性中共享screen。
selectUserGroups	query	

- 一组对象；
- 如果已经使用“countOutput”参数，则检索到的对象的计数。

示例

按ID检索Screen

Retrieve all data about screen “26” and its screen items.

Request:

```

1. {
2.   "jsonrpc": "2.0",
3.   "method": "screen.get",
4.   "params": {
5.     "output": "extend",
6.     "selectScreenItems": "extend",
7.     "selectUsers": "extend",
8.     "selectUserGroups": "extend",
9.     "screenids": "26"
10.    },
11.    "auth": "038e1d7b1735c6a5436ee9eae095879e",
12.    "id": 1
13.  }
```

Response:

```

1.  {
2.      "jsonrpc": "2.0",
3.      "result": [
4.          {
5.              "screenitems": [
6.                  {
7.                      "screenitemid": "67",
8.                      "screenid": "26",
9.                      "resourcetype": "0",
10.                     "resourceid": "612",
11.                     "width": "320",
12.                     "height": "200",
13.                     "x": "0",
14.                     "y": "0",
15.                     "colspan": "0",
16.                     "rowspan": "0",
17.                     "elements": "25",
18.                     "valign": "0",
19.                     "halign": "0",
20.                     "style": "0",
21.                     "url": "",
22.                     "dynamic": "0",
23.                     "sort_triggers": "0"
24.                 }
25.             ],
26.             "users": [
27.                 {
28.                     "sysmapuserid": "1",
29.                     "userid": "2",
30.                     "permission": "2"
31.                 }
32.             ],
33.             "userGroups": [
34.                 {
35.                     "screenusrgrpid": "1",
36.                     "usrgrpid": "7",
37.                     "permission": "3"
38.                 }
39.             ],
40.             "screenid": "26",
41.             "name": "CPU Graphs",
42.             "hsize": "3",
43.             "vsize": "2",
44.             "templateid": "0",
45.             "userid": "1",
46.             "private": "1"
47.         }
48.     ],
49.     "id": 1
50. }
```

参见

- [Screen item](#)
- [Screen user](#)
- [Screen user group](#)

来源

`CScreen::get()` in *frontends/php/include/classes/api/services/CScreen.php*.

screen.update

说明

```
object screen.update(object/array screens)
```

此方法允许更新现有screen。

参数

(object/array) 要更新的screen属性。

必须为每个屏幕定义“屏幕ID”属性，所有其他属性都是可选的。 只有通过的属性将被更新，所有其他属性将保持不变。

除了[标准screen属性](#)之外，该方法接受以下参数。

属性	类型	说明
screenitems	array	替代现有Screen items的screen items. screen items通过坐标更新，因此每个screen items必须定义“x”和“y”属性。
users	array	Screen用户共享替换现有元素。
userGroups	array	Screen用户组共享以替换现有元素。

返回值

(object) 返回一个包含“screenid”属性下更新屏幕的ID的对象。

示例

Renaming a screen

Rename a screen to “CPU Graphs”.

Request:

```

1. {
2.   "jsonrpc": "2.0",
3.   "method": "screen.update",
4.   "params": {
5.     "screenid": "26",
6.     "name": "CPU Graphs"
7.   },
8.   "auth": "038e1d7b1735c6a5436ee9eae095879e",
9.   "id": 1
10. }
```

screen.update

Response:

```
1. {
2.     "jsonrpc": "2.0",
3.     "result": {
4.         "screenids": [
5.             "26"
6.         ]
7.     },
8.     "id": 1
9. }
```

Change screen owner

Available only for admins and super admins.

Request:

```
1. {
2.     "jsonrpc": "2.0",
3.     "method": "screen.update",
4.     "params": {
5.         "screenid": "83",
6.         "userid": "1"
7.     },
8.     "auth": "038e1d7b1735c6a5436ee9eae095879e",
9.     "id": 2
10. }
```

Response:

```
1. {
2.     "jsonrpc": "2.0",
3.     "result": {
4.         "screenids": [
5.             "83"
6.         ]
7.     },
8.     "id": 2
9. }
```

参见

- [Screen item](#)
- [screenitem.create](#)
- [screenitem.update](#)

screen.update

- `screenitem.updatebyposition`
- `Screen user`
- `Screen user group`

来源

`CScreen::update()` in *frontends/php/include/classes/api/services/CScreen.php*.

聚合图形项目

这个类与聚合图形项配合使用

Object references:

- `Screen item`

对象引用:

- `Screen item`

Available methods: 可用方法:

- `screenitem.create` - 创建新的聚合图形项
- `screenitem.delete` - 删除聚合图形项
- `screenitem.get` - 检索聚合图形项
- `screenitem.update` - 更新聚合图形项
- `screenitem.updatebyposition` - 更新一个特定的聚合图形屏幕

> 聚合图形项目对象

以下对象与 [聚合图形项目](#) API相关.

Screen item 聚合图形项目

聚合图形项目对象定义了聚合图形上显示的元素，它具有以下特性.

属性	类型	描述
screenitemid	字符串	(只读) 聚合图形项目ID
resourcetype(必选项)	整数型	聚合图形项目类型. 可用值: 0 - 图形; 1 - 简单图形; 2 - 映射; 3 - 纯文本; 4 - 主机信息; 5 - 触发器信息; 6 - Zabbix状态; 7 - 时钟; 8 - 屏幕; 9 - 触发器概述 10 - 数据概述; 11 - URL; 12 - 历史动作; 13 - 历史事件; 14 - 主机组最新问题; 15 - 系统状态; 16 - 主机最新问题; 19 - 简单图形原型; 20 - 图形原型.
screenid(必选项)	字符串	聚合图形ID.
application	字符串	应用或部分应用用作聚合图形项目的过滤条件, 适用于资源类型: "数据概述"和"触发器概述".
colspan	整数型	聚合图形项目的列数量. 默认值: 1.
dynamic	整数型	聚合图形项目是否动态. 可用值: 0 - (默认值) 不动态; 1 - 动态.
elements	整数型	聚合图形项目显示的行数量. 默认值: 25.
halign	整数型	指定聚合图形项目的对齐方式. 可用值: 0 - (默认值) 居中; 1 - 向左对齐; 2 - 向右对齐.
height	整数型	聚合图形项目的高度(单位:像素). 默认值: 200.
maxcolumns	整数型	指定图形原型或简单图形原型的最大显示列的数量. 默认值: 3.
resourceid	字符串	聚合图形项目显示对象的ID. 根据聚合图形项目类型, "resourceid"属性可以引用不同的对象. 聚合图形项目所需要的数据概述, 图形, 映射, 纯文本, 聚合图形, 简单图形和触发器概述的数据. 聚合图形项目不使用本地和服务器的时钟, 历史动作, 历史事件, 主机信息, zabbix状态, 系统状态和.
rowspan	整数型	聚合图形项目显示的行数量. 默认值: 1.
		对动作或触发器进行必要的排序. 聚合图形元素中历史动作可用值: 3 - 时间,

sort_triggers	整数型	升序；4 - 时间，降序；5 - 类型，升序；6 - 类型，降序；7 - 状态，升序；8 - 状态，降序；9 - 重试左，升序；10 - 重试左，降序；11 - 容器，升序；12 - 容器，降序。聚合图形项目主机组和主机最新问题可以值：0 - (默认值) 最新变化，降序；1 - 级别，降序；2 - 主机，升序。
style	整数型	聚合图形项目显示选项。聚合图形项目中数据概述与触发器概述的可用值：0 - (默认值) 在左侧显示主机；1 - 在顶部显示主机。聚合图形属性中主机信息与触发器信息的可用之：0 - (默认值) 水平布局；1 - 垂直布局。聚合图形项目属性中时钟可用值：0 - (默认值) 本地时间；1 - 服务器时间；2 - 主机时间。聚合图形项目中纯文本可用值：0 - (默认值) 纯文本显示值；1 - 显示HTML。
url	字符串	聚合图形项目中显示Web页面的URL信息，用于Web监控聚合图形项目。
valign	整数型	指定聚合图形项目中的垂直排列的方式。可用值：0 - (默认值) 居中；1 - 顶端；2 - 底部。
width	整数型	聚合图形项目宽度(单位:像素)。默认值：320.
x	整数型	聚合图形项目在聚合图形的X轴，从左到右。默认值：0.
y	整数型	聚合图形项目在聚合图形的Y轴，从上到下。默认值：0.

screenitem.create

描述

```
object screenitem.create(object/array screenItems)
```

创建聚合图形项目的方法 .

参数

(object/array) Screen items to create.

聚合图形项目允许的方法 [standard screen item properties](#).

返回值

(对象) 在创建聚合图形项目 [screenitemids](#) 下面返回一个对象的ID. 返回对象ID的顺序与聚合图形项目的顺序相匹配 .

示例

创建一个聚合图形项目

在聚合图形的左上角创建一个显示图形的聚合图形项目 .

Request:

```
1. {
2.     "jsonrpc": "2.0",
3.     "method": "screenitem.create",
4.     "params": {
5.         "screenid": 16,
6.         "resourcetype": 0,
7.         "resourceid": 612,
8.         "x": 0,
9.         "y": 0
10.    },
11.    "auth": "038e1d7b1735c6a5436ee9eae095879e",
12.    "id": 1
13. }
```

Response:

```
1. {
2.     "jsonrpc": "2.0",
3.     "result": {
4.         "screenitemids": [
```

```
5.           "65"
6.         ],
7.       },
8.     "id": 1
9. }
```

参考

- [screen.update](#)

来源

`CScreenItem::create()` in `frontends/php/include/classes/api/services/CScreenItem.php`.

screenitem.delete

描述

```
object screenitem.delete(array screenItemIds)
```

删除聚合图形项目的方法.

参数

(array) 删除聚合图形项目的ID.

返回值

(对象) 在删除聚合图形项目 `screenitemids` 下面返回一个对象的ID. 返回对象ID的顺序与聚合图形项目的顺序相匹配.

示例

删除多个聚合图形项目

删除多个聚合图形项目

Request:

```
1. {
2.     "jsonrpc": "2.0",
3.     "method": "screenitem.delete",
4.     "params": [
5.         "65",
6.         "63"
7.     ],
8.     "auth": "3a57200802b24cda67c4e4010b50c065",
9.     "id": 1
10. }
```

Response:

```
1. {
2.     "jsonrpc": "2.0",
3.     "result": {
4.         "screenitemids": [
5.             "65",
6.             "63"
7.         ]
8.     },
9.     "id": 1
}
```

screenitem.delete

```
10. }
```

参考

- [screen.update](#)

来源

`CScreenItem::delete()` in `frontends/php/include/classes/api/services/CScreenItem.php`.

screenitem.get

描述

`integer/array screenitem.get(对象 参数)`

根据给定的参数检索聚合图形项目的方法.

参数

(对象) 定义参数的输出 .

该方法支持以下参数.

Parameter	Type	Description
screenitemids	string/array	返回给定的聚合图形项目的ID.
screenids	string/array	返回聚合图形给定的聚合图形项目 .
sortfield	string/array	返回给定属性的排序. 可用值: <code>screenitemid</code> 和 <code>screenid</code> .
countOutput	flag	返回给定参数的所有内容的数量, 参数参考 reference commentary page 页面.
editable	boolean	
excludeSearch	flag	
filter	object	
limit	integer	
output	query	
preservekeys	flag	
search	object	
searchByAny	boolean	
searchWildcardsEnabled	boolean	
sortorder	string/array	
startSearch	flag	

返回值

(integer/array) 返回以下内容:

- 对象数组;
- 检索对象的数量, 使用参数 `countOutput` 可用 .

示例

获取聚合图形中聚合图形项目

从聚合图形中获取所有的聚合图形项目 .

Request:

```

1. {
2.     "jsonrpc": "2.0",
3.     "method": "screenitem.get",
4.     "params": {
5.         "output": "extend",
6.         "screenids": "3"
7.     },
8.     "auth": "038e1d7b1735c6a5436ee9eae095879e",
9.     "id": 1
10. }
```

Response:

```

1. {
2.     "jsonrpc": "2.0",
3.     "result": [
4.         {
5.             "screenitemid": "20",
6.             "screenid": "3",
7.             "resourctype": "0",
8.             "resourceid": "433",
9.             "width": "500",
10.            "height": "120",
11.            "x": "0",
12.            "y": "0",
13.            "colspan": "1",
14.            "rowspan": "1",
15.            "elements": "0",
16.            "valign": "1",
17.            "halign": "0",
18.            "style": "0",
19.            "url": "",
20.            "dynamic": "0",
21.            "sort_triggers": "0",
22.            "application": "",
23.            "max_columns": "3"
24.        },
25.        {
26.            "screenitemid": "21",
27.            "screenid": "3",
28.            "resourctype": "0",
29.            "resourceid": "387",
30.            "width": "500",
```

```
31.          "height": "100",
32.          "x": "0",
33.          "y": "1",
34.          "colspan": "1",
35.          "rowspan": "1",
36.          "elements": "0",
37.          "valign": "1",
38.          "halign": "0",
39.          "style": "0",
40.          "url": "",
41.          "dynamic": "0",
42.          "sort_triggers": "0",
43.          "application": "",
44.          "max_columns": "3"
45.      },
46.      {
47.          "screenitemid": "22",
48.          "screenid": "3",
49.          "resourcetype": "1",
50.          "resourceid": "10013",
51.          "width": "500",
52.          "height": "148",
53.          "x": "1",
54.          "y": "0",
55.          "colspan": "1",
56.          "rowspan": "1",
57.          "elements": "0",
58.          "valign": "1",
59.          "halign": "0",
60.          "style": "0",
61.          "url": "",
62.          "dynamic": "0",
63.          "sort_triggers": "0",
64.          "application": "",
65.          "max_columns": "3"
66.      },
67.      {
68.          "screenitemid": "23",
69.          "screenid": "3",
70.          "resourcetype": "1",
71.          "resourceid": "22181",
72.          "width": "500",
73.          "height": "184",
74.          "x": "1",
75.          "y": "1",
76.          "colspan": "1",
77.          "rowspan": "1",
78.          "elements": "0",
79.          "valign": "1",
80.          "halign": "0",
81.          "style": "0",
82.          "url": "",
83.          "dynamic": "0",
```

```
84.         "sort_triggers": "0",
85.         "application": "",
86.         "max_columns": "3"
87.     }
88. ],
89. "id": 1
90. }
```

来源

CScreenItem::get() in *frontends/php/include/classes/api/services/CScreenItem.php*.

screenitem.update

描述

```
object screenitem.update(object/array screenItems)
```

更新现有聚合图形项目的方法.

参数

(object/array) Screen item properties 更新.

必须定义每个聚合图形项目的 `screenitemid` , 其他属性可选, 聚合图形项目的更新只能通过输入的参数, 没有输入参数的内容将不改变.

返回值

(对象) 返回更新聚合图形项目下 `screenitemids` 属性的对象ID.

示例

设置聚合图形项目的大小

设置聚合图形项目的宽度为500px和高度为300px.

Request:

```
1. {
2.     "jsonrpc": "2.0",
3.     "method": "screenitem.update",
4.     "params": {
5.         "screenitemid": "20",
6.         "width": 500,
7.         "height": 300
8.     },
9.     "auth": "038e1d7b1735c6a5436ee9eae095879e",
10.    "id": 1
11. }
```

Response:

```
1. {
2.     "jsonrpc": "2.0",
3.     "result": {
4.         "screenitemids": [
5.             "20"
6.         ]
7.     }
8. }
```

screenitem.update

```
7.      },
8.      "id": 1
9. }
```

参考

- [screenitem.updatebyposition](#)

来源

`CScreenItem::update()` in *frontends/php/include/classes/api/services/CScreenItem.php*.

screenitem.updatebyposition

描述

对象 screenitem.updatebyposition(array `screenItems`)

更新聚合图形给定单元格内的聚合图形项目方法, 如果该单元格内容为空, 则创建一个新的聚合图形项目.

参数

(array) `Screen item properties` 更新.

必须定义每个聚合图形项目的 `x`, `y` 和 `screenid` 的属性, 其他属性可选, 只更新聚合图形定义的属性, 没有定义的属性将不改变.

Return values

(对象) 返回更新和创建的聚合图形属性 `screenitemids` 的对象ID.

Examples

改变聚合图形项目的资源ID

改变位于聚合图形左上角聚合图形元素的资源ID.

Request:

```

1. {
2.     "jsonrpc": "2.0",
3.     "method": "screenitem.updatebyposition",
4.     "params": [
5.         {
6.             "screenid": "16",
7.             "x": 0,
8.             "y": 0,
9.             "resourceid": "644"
10.        }
11.    ],
12.    "auth": "038e1d7b1735c6a5436ee9eae095879e",
13.    "id": 1
14. }
```

Response:

```

1. {
2.     "jsonrpc": "2.0",
3.     "result": {
```

```
4.      "screenitemids": [
5.          "66"
6.      ],
7.      },
8.      "id": 1
9. }
```

参考

- [screenitem.update](#)

来源

`CScreenItem::update()` in *frontends/php/include/classes/api/services/CScreenItem.php*.

脚本

这个类被设计来配合使用脚本。

对象引用：

- [脚本](#)

可用的方法：

- `script.create` - 创建新的脚本
- `script.delete` - 删除脚本
- `script.execute` - 运行脚本
- `script.get` - 检索脚本
- `script.getscriptsbyhosts` - 检索主机脚本
- `script.update` - 更新脚本

> Script object(脚本对象)

以下对象跟“脚本”API直接相关。

脚本

脚本对象具备以下属性：

属性	类型	说明
scriptid	string	(只读) 脚本ID
command(required)	string	运行的命令
name(required)	string	脚本的名称
confirmation	string	弹出确认文本，当尝试在Zabbix前端运行脚本时，会出现弹出窗口
description	string	脚本说明
executeon	integer	在哪里运行脚本。 可能出现的值：0 - 在Zabbix Agent代理上运行；1 - (默认) 在Zabbix Server上运行
groupid	string	可以运行脚本的主机组ID，如果设置为0，则脚本在所有主机组上可用 默认:0
host_access	integer	主机运行脚本所需要的权限可能出现的值：2 - (默认) 读；3 - 写
type	integer	脚本类型 可能出现的值：0 - (默认) 脚本；1 - IPMI
usrgrpid	string	将被允许运行脚本的用户组ID，如果设置为0，则脚本在所有用户组上可用 默认: 0

script.create

说明

```
object script.create(object/array scripts)
```

这个方法运行创建一个新的脚本。

参数

(object/array) 脚本创建。

该方法使用[标准脚本属性](#)接受脚本。

返回值

(object) 返回一个包含在 `scriptids` 属性下创建的脚本ID的对象，返回ID的顺序与脚本的传递顺序相匹配。

例子

创建一个脚本

创建一个用于重启服务器的脚本，脚本需要有对主机的写入权限并在前端运行脚本之前显示配置消息。

请求：

```
1. {
2.   "jsonrpc": "2.0",
3.   "method": "script.create",
4.   "params": {
5.     "name": "Reboot server",
6.     "command": "reboot server 1",
7.     "host_access": 3,
8.     "confirmation": "Are you sure you would like to reboot the server?"
9.   },
10.  "auth": "038e1d7b1735c6a5436ee9eae095879e",
11.  "id": 1
12. }
```

响应：

```
1. {
2.   "jsonrpc": "2.0",
3.   "result": {
4.     "scriptids": [
5.       "3"
6.     ]
7.   }
8. }
```

script.create

```
7.      },
8.      "id": 1
9. }
```

源码

CScript::create() 在*frontends/php/include/classes/api/services/CScript.php*查看。

script.delete

说明

```
object script.delete(array scriptIds)
```

这个方法运行删除脚本。

参数

(array) 要删除的脚本ID。

返回值

(object) Returns an object containing the IDs of the deleted scripts under the scriptids property. (object) 返回一个包含被删除脚本ID的对象。

Examples范例

Delete multiple scripts删除多个脚本

Delete two scripts删除两个脚本。

Request请求：

```
1. {
2.     "jsonrpc": "2.0",
3.     "method": "script.delete",
4.     "params": [
5.         "3",
6.         "4"
7.     ],
8.     "auth": "3a57200802b24cda67c4e4010b50c065",
9.     "id": 1
10. }
```

Response响应：

```
1. {
2.     "jsonrpc": "2.0",
3.     "result": {
4.         "scriptids": [
5.             "3",
6.             "4"
7.         ]
8.     },
9.     "id": 1
}
```

script.delete

```
10. }
```

Source源码

CScript::delete() in *frontends/php/include/classes/api/services/CScript.php*.

script.execute

Description说明

```
object script.execute(object parameters)
```

This method allows to run a script on a host此方法允许在主机上运行脚本.

Parameters参数

(object) Parameters containing the ID of the script to run and the ID of the host包含要运行的脚本的ID和主机的ID的参数.

Parameter参数	Type类型	Description说明
hostid (required必须)	string	ID of the host to run the script on要运行脚本的主机的ID.
scriptid (required必须)	string	ID of the script to run要运行的脚本的ID.

Return values返回值

(object) Returns the result of script execution返回执行脚本的结果.

Property参数	Type类型	Description说明
response	string	Whether the script was run successfully脚本是否成功运行. Possible values可能的值为: <code>success</code> or 或 <code>failed</code> .
value	string	Script output脚本输出.

Examples范例

Run a script运行脚本

Run a "ping" script on a host在主机上运行"ping"脚本.

Request请求:

```

1. {
2.     "jsonrpc": "2.0",
3.     "method": "script.execute",
4.     "params": {
5.         "scriptid": "1",
6.         "hostid": "30079"
7.     },
8.     "auth": "038e1d7b1735c6a5436ee9eae095879e",

```

```
script.execute
```

```
9.      "id": 1
10. }
```

Response响应：

```
1. {
2.     "jsonrpc": "2.0",
3.     "result": {
4.         "response": "success",
5.         "value": "PING 127.0.0.1 (127.0.0.1) 56(84) bytes of data.\n64 bytes from 127.0.0.1: icmp_req=1 ttl=64 time=0.074 ms\n64 bytes from 127.0.0.1: icmp_req=2 ttl=64 time=0.030 ms\n64 bytes from 127.0.0.1: icmp_req=3 ttl=64 time=0.030 ms\n--- 127.0.0.1 ping statistics ---\n3 packets transmitted, 3 received, 0% packet loss, time 1998ms\nrtt min/avg/max/mdev = 0.030/0.044/0.074/0.022 ms\n"
6.     },
7.     "id": 1
8. }
```

Source源码

CScript::execute() in *frontends/php/include/classes/api/services/CScript.php*.

script.get

Description说明

```
integer/array script.get(object parameters)
```

The method allows to retrieve scripts according to the given parameters该方法允许根据给定的参数检索脚本.

Parameters参数

(object) Parameters defining the desired output定义所需输出的参数.

The method supports the following parameters该方法支持以下参数.

Parameter参数	Type类型	Description说明
groupids	string/array	Return only scripts that can be run on the given host groups只返回可以在指定主机组上运行的脚本.
hostids	string/array	Return only scripts that can be run on the given hosts只返回可以在指定主机上运行的脚本.
scriptids	string/array	Return only scripts with the given IDs只返回具有指定ID的脚本.
usrgrpids	string/array	Return only scripts that can be run by users in the given user groups只返回指定用户组中的用户可以运行的脚本.
selectGroups	query	Return host groups that the script can be run on in the <code>groups</code> property返回在 <code>groups</code> 属性中可运行脚本的主机组.
selectHosts	query	Return hosts that the script can be run on in the <code>hosts</code> property返回在 <code>hosts</code> 属性中可运行脚本的主机.
sortfield	string/array	Sort the result by the given properties按指定的属性对结果分类. Possible values are可能的值为: <code>scriptid</code> and 和 <code>name</code> .
countOutput	flag	These parameters being common for all <code>get</code> methods are described in detail in the reference commentary . 在 reference commentary 中详细描述了所有“get”方法的这些参数.
editable	boolean	
excludeSearch	flag	
filter	object	
limit	integer	
output	query	
preservekeys	flag	

search	object
searchByAny	boolean
searchWildcardsEnabled	boolean
sortorder	string/array
startSearch	flag

Return values返回值

(integer/array) Returns either 返回两者其中任一：

- an array of objects 一组对象；
- the count of retrieved objects, if the `countOutput` parameter has been used 如果已经使用了 `countOutput` 参数，则检索对象的计数.

Examples范例

Retrieve all scripts 检索所有脚本

Retrieve all configured scripts 检索所有已配置的脚本.

Request 请求：

```

1. {
2.   "jsonrpc": "2.0",
3.   "method": "script.get",
4.   "params": {
5.     "output": "extend"
6.   },
7.   "auth": "038e1d7b1735c6a5436ee9eae095879e",
8.   "id": 1
9. }
```

Response 响应：

```

1. {
2.   "jsonrpc": "2.0",
3.   "result": [
4.     {
5.       "scriptid": "1",
6.       "name": "Ping",
7.       "command": "/bin/ping -c 3 {HOST.CONN} 2>&1",
8.       "host_access": "2",
9.       "usrgrpid": "0",
10.      "groupid": "0",
11.      "description": "",
12.      "confirmation": "",
13.      "type": "0",
```

script.get

```
14.          "execute_on": "1"
15.      },
16.      {
17.          "scriptid": "2",
18.          "name": "Traceroute",
19.          "command": "/usr/bin/traceroute {HOST.CONN} 2>&1",
20.          "host_access": "2",
21.          "usrgrpid": "0",
22.          "groupid": "0",
23.          "description": "",
24.          "confirmation": "",
25.          "type": "0",
26.          "execute_on": "1"
27.      },
28.      {
29.          "scriptid": "3",
30.          "name": "Detect operating system",
31.          "command": "sudo /usr/bin/nmap -O {HOST.CONN} 2>&1",
32.          "host_access": "2",
33.          "usrgrpid": "7",
34.          "groupid": "0",
35.          "description": "",
36.          "confirmation": "",
37.          "type": "0",
38.          "execute_on": "1"
39.      }
40.  ],
41.  "id": 1
42. }
```

See also参考

- [Host](#)
- [Host group](#)

Source源码

CScript::get() in *frontends/php/include/classes/api/services/CScript.php*.

script.getscriptsbyhosts

Description说明

```
object script.getscriptsbyhosts(array hostIds)
```

This method allows to retrieve scripts available on the given hosts此方法允许检索给定主机上可用的脚本.

Parameters参数

(string/array) IDs of hosts to return scripts for要返回脚本的主机的ID.

Return values返回值

(object) Returns an object with host IDs as properties and arrays of available scripts as values返回一个具有主机ID的对象和可用脚本的数组作为值.

The method will automatically expand macros in the confirmation text该方法将自动在 confirmation 文本中展开宏.

Examples范例

Retrieve scripts by host IDs通过主机ID检索脚本

Retrieve all scripts available on hosts "30079" and "30073"检索主机"30079" 和"30073"上可用的所有脚本。.

Request请求:

```

1. {
2.   "jsonrpc": "2.0",
3.   "method": "script.getscriptsbyhosts",
4.   "params": [
5.     "30079",
6.     "30073"
7.   ],
8.   "auth": "038e1d7b1735c6a5436ee9eae095879e",
9.   "id": 1
10. }
```

Response响应:

```

1. {
2.   "jsonrpc": "2.0",
3.   "result": {
```

```
4.     "30079": [
5.         {
6.             "scriptid": "3",
7.             "name": "Detect operating system",
8.             "command": "sudo /usr/bin/nmap -O {HOST.CONN} 2>&1",
9.             "host_access": "2",
10.            "usrgrpid": "7",
11.            "groupid": "0",
12.            "description": "",
13.            "confirmation": "",
14.            "type": "0",
15.            "execute_on": "1",
16.            "hostid": "10001"
17.        },
18.        {
19.            "scriptid": "1",
20.            "name": "Ping",
21.            "command": "/bin/ping -c 3 {HOST.CONN} 2>&1",
22.            "host_access": "2",
23.            "usrgrpid": "0",
24.            "groupid": "0",
25.            "description": "",
26.            "confirmation": "",
27.            "type": "0",
28.            "execute_on": "1",
29.            "hostid": "10001"
30.        },
31.        {
32.            "scriptid": "2",
33.            "name": "Traceroute",
34.            "command": "/usr/bin/traceroute {HOST.CONN} 2>&1",
35.            "host_access": "2",
36.            "usrgrpid": "0",
37.            "groupid": "0",
38.            "description": "",
39.            "confirmation": "",
40.            "type": "0",
41.            "execute_on": "1",
42.            "hostid": "10001"
43.        }
44.    ],
45.    "30073": [
46.        {
47.            "scriptid": "3",
48.            "name": "Detect operating system",
49.            "command": "sudo /usr/bin/nmap -O {HOST.CONN} 2>&1",
50.            "host_access": "2",
51.            "usrgrpid": "7",
52.            "groupid": "0",
53.            "description": "",
54.            "confirmation": "",
55.            "type": "0",
56.            "execute_on": "1",
```

```
57.          "hostid": "10001"
58.      },
59.      {
60.          "scriptid": "1",
61.          "name": "Ping",
62.          "command": "/bin/ping -c 3 {HOST.CONN} 2>&1",
63.          "host_access": "2",
64.          "usrgrpid": "0",
65.          "groupid": "0",
66.          "description": "",
67.          "confirmation": "",
68.          "type": "0",
69.          "execute_on": "1",
70.          "hostid": "10001"
71.      },
72.      {
73.          "scriptid": "2",
74.          "name": "Traceroute",
75.          "command": "/usr/bin/traceroute {HOST.CONN} 2>&1",
76.          "host_access": "2",
77.          "usrgrpid": "0",
78.          "groupid": "0",
79.          "description": "",
80.          "confirmation": "",
81.          "type": "0",
82.          "execute_on": "1",
83.          "hostid": "10001"
84.      }
85.  ],
86. },
87. "id": 1
88. }
```

Source源码

```
CScript::getScriptsByHosts() in
frontends/php/include/classes/api/services/CScript.php.
```

script.update

Description说明

```
object script.update(object/array scripts)
```

This method allows to update existing scripts此方法允许更新现有脚本.

Parameters参数

(object/array) Script properties to be updated被更新.

The `scriptid` property must be defined for each script, all other properties are optional. Only the passed properties will be updated, all others will remain unchanged. 必须为每个脚本定义 `scriptid` 属性，所有其他属性都是可选的。只有通过的属性将被更新，所有其他属性将保持不变。

Return values返回值

(object) Returns an object containing the IDs of the updated scripts under the `scriptids` property. (object) 返回一个包含 `scriptids` 属性下更新脚本的ID的对象。

Examples范例

Change script command更改脚本命令

Change the command of the script to "/bin/ping -c 10 {HOST.CONN} 2>&1". 将脚本的命令更改
为"/bin/ping -c 10 {HOST.CONN} 2>&1".

Request请求:

```
1. {
2.     "jsonrpc": "2.0",
3.     "method": "script.update",
4.     "params": {
5.         "scriptid": "1",
6.         "command": "/bin/ping -c 10 {HOST.CONN} 2>&1"
7.     },
8.     "auth": "038e1d7b1735c6a5436ee9eae095879e",
9.     "id": 1
10. }
```

Response响应:

```
1. {
2.     "jsonrpc": "2.0",
```

```
3.     "result": {
4.         "scriptids": [
5.             "1"
6.         ],
7.     },
8.     "id": 1
9. }
```

Source源码

CScript::update() in *frontends/php/include/classes/api/services/CScript.php*.

触发器

该类作用于触发器。

引用对象：

- [Trigger](#)

有效的方法：

- `trigger.adddependencies` - 添加新的触发器依赖
- `trigger.create` - 创建新的触发器
- `trigger.delete` - 删除触发器
- `trigger.deletedependencies` - 删除触发器依赖
- `trigger.get` - 检索触发器
- `trigger.update` - 更新触发器

> Trigger object触发器对象

The following objects are directly related to the `trigger` API以下是`trigger` API 的使用方法.

Trigger触发器

The trigger object has the following properties触发器对象具有以下属性.

Property参数	Type类型	Description说明
<code>triggerid</code>	<code>string</code>	(readonly) ID of the trigger触发器的ID.
<code>description(required)</code>	<code>string</code>	Name of the trigger触发器的名称.
<code>expression(required)</code>	<code>string</code>	Reduced trigger expression生成触发表达式.
<code>comments</code>	<code>string</code>	Additional comments to the trigger触发器的附加注释.
<code>error</code>	<code>string</code>	(readonly) Error text if there have been any problems when updating the state of the trigger如果在更新触发器的状态时出现任何问题，则显示文本错误.
<code>flags</code>	<code>integer</code>	(readonly) Origin of the trigger.原始的触发器 Possible values are可能的值是: 0 - (default) a plain trigger一个常用触发器; 4 - a discovered trigger一个被发现的触发器.
<code>lastchange</code>	<code>timestamp</code>	(readonly) Time when the trigger last changed its state触发器最后更改其状态的时间.
<code>priority</code>	<code>integer</code>	Severity of the trigger触发器的严重性级别. Possible values are可能的值为: 0 - (default) not classified未分类; 1 - information信息; 2 - warning警告; 3 - average一般严重; 4 - high严重; 5 - disaster灾难.
<code>state</code>	<code>integer</code>	(readonly) State of the trigger触发状态. Possible values可能的值为: 0 - (default) trigger state is up to date触发器状态是最新的; 1 - current trigger state is unknown当前的触发器状态是未知的.
<code>status</code>	<code>integer</code>	Whether the trigger is enabled or disabled触发器是启用还是禁用. Possible values are可能的值为: 0 - (default) enabled启用; 1 - disabled禁用.
<code>templateid</code>	<code>string</code>	(readonly) ID of the parent template trigger主模板触发器的ID.
<code>type</code>	<code>integer</code>	Whether the trigger can generate multiple problem events触发器是否可以生成多个问题事件. Possible values are可能的值为: 0 - (default) do not generate multiple events不生成多个事件; 1 - generate multiple events生成多个事件.
<code>url</code>	<code>string</code>	URL associated with the trigger与触发器相关联的URL.
		(readonly) Whether the trigger is in OK or problem state触发器是否处于OK或问题状态. Possible

		values are可能的值为: 0 - (default) OK正常; 1 - problem有问题.
recoverymode	integer	OK event generation mode OK事件生成模式. Possible values are可能的值为: 0 - (default) <i>Expression</i> 表达式; 1 - <i>Recovery expression</i> 恢复表达式; 2 - <i>None</i> 无.
recovery_expression	string	Reduced trigger recovery expression生成触发器恢复表达式.
correlation_mode	integer	OK event closes OK事件关闭. Possible values are 可能的值为: 0 - (default) All problems所有问题; 1 - All problems if tag values match标签值匹配时的所有问题.
correlation_tag	string	Tag for matching用于匹配的标签.
manual_close	integer	Allow manual close允许手动关闭. Possible values are可能的值为: 0 - (default) No不可以; 1 - Yes可以.

trigger.adddependencies

Description说明

```
object trigger.adddependencies(object/array triggerDependencies)
```

This method allows to create new trigger dependencies此方法允许创建新的触发器依赖关系。

Parameters参数

(object/array) Trigger dependencies to create触发依赖关系创建。

Each trigger dependency has the following parameters每个触发器依赖关系具有以下参数：

Parameter参数	Type类型	Description说明
triggerid(required)	string	ID of the dependent trigger依赖触发器的ID。
dependsOnTriggerid(required)	string	ID of the trigger that the trigger depends on触发依赖的触发器的ID。

Return values返回值

(object) Returns an object containing the IDs of the dependent triggers under the `triggerids` property. (object) 返回一个对象，该对象包含 `triggerids` 属性下的依赖触发器的ID。

Examples范例

Add a trigger dependency添加触发器依赖关系

Make trigger "14092" dependent on trigger "13565."触发器"14092"依赖于触发器"13565。"。

Request请求：

```

1. {
2.   "jsonrpc": "2.0",
3.   "method": "trigger.adddependencies",
4.   "params": {
5.     "triggerid": "14092",
6.     "dependsOnTriggerid": "13565"
7.   },
8.   "auth": "038e1d7b1735c6a5436ee9eae095879e",
9.   "id": 1
10. }
```

Response响应：

```
1.  {
2.      "jsonrpc": "2.0",
3.      "result": {
4.          "triggerids": [
5.              "14092"
6.          ],
7.      },
8.      "id": 1
9. }
```

See also参考

- [trigger.update](#)

Source来源

`CTrigger::addDependencies()` in
frontends/php/include/classes/api/services/CTrigger.php.

trigger.create

Description说明

```
object trigger.create(object/array triggers)
```

This method allows to create new triggers此方法允许创建新的触发器.

Parameters参数

(object/array) Triggers to create需要去创建的触发器.

Additionally to the [standard trigger properties](#) the method accepts the following parameters.除[standard trigger properties](#)之外，该方法接受以下参数。

Parameter参数	Type类型	Description说明
dependencies	array	Triggers that the trigger is dependent on触发触发器依赖. The triggers must have the <code>triggerid</code> property defined触发器必须定义 <code>triggerid</code> 属性.
tags	array	Trigger tags触发器标签.

The trigger expression has to be given in its expanded form触发器表达式必须以其扩展形式给出.

Return values返回值

(object) Returns an object containing the IDs of the created triggers under the `triggerids` property. The order of the returned IDs matches the order of the passed triggers. (object) 返回一个包含 `triggerids` 属性下创建触发器的ID的对象.返回的ID的顺序与传递的触发器的顺序相匹配。

Examples范例

Creating a trigger创建一个触发器

Create a trigger with a single trigger dependency创建具有单个触发依赖关系的触发器.

Request请求:

```

1. {
2.   "jsonrpc": "2.0",
3.   "method": "trigger.create",
4.   "params": [
5.     {
6.       "description": "Processor load is too high on {HOST.NAME}",
7.       "expression": "{Linux server:system.cpu.load[percpu,avg1].last()}>5",

```

trigger.create

```
8.         "dependencies": [
9.             {
10.                 "triggerid": "17367"
11.             }
12.         ],
13.     },
14.     {
15.         "description": "Service status",
16.         "expression": "{Linux server:log[/var/log/system,Service .* has stopped].strlen()}<>0",
17.         "dependencies": [
18.             {
19.                 "triggerid": "17368"
20.             }
21.         ],
22.         "tags": [
23.             {
24.                 "tag": "service",
25.                 "value": "{$ITEM.VALUE}.regsub(\"Service (.*) has stopped\", \"$\\1\")"
26.             },
27.             {
28.                 "tag": "error",
29.                 "value": ""
30.             }
31.         ]
32.     },
33. ],
34. "auth": "038e1d7b1735c6a5436ee9eae095879e",
35. "id": 1
36. }
```

Response响应：

```
1. {
2.     "jsonrpc": "2.0",
3.     "result": {
4.         "triggerids": [
5.             "17369",
6.             "17370"
7.         ]
8.     },
9.     "id": 1
10. }
```

Source来源

CTrigger::create() in *frontends/php/include/classes/api/services/CTrigger.php*.

trigger.delete

Description说明

```
object trigger.delete(array triggerIds)
```

This method allows to delete triggers此方法允许删除触发器.

Parameters参数

(array) IDs of the triggers to delete. (array) 要删除的触发器的ID

Return values返回值

(object) Returns an object containing the IDs of the deleted triggers under the triggerids property. (object) 返回一个包含 triggerids 属性下的已删除触发器ID的对象

Examples范例

Delete multiple triggers删除多个触发器

Delete two triggers删除两个触发器.

Request请求:

```
1. {
2.   "jsonrpc": "2.0",
3.   "method": "trigger.delete",
4.   "params": [
5.     "12002",
6.     "12003"
7.   ],
8.   "auth": "3a57200802b24cda67c4e4010b50c065",
9.   "id": 1
10. }
```

Response响应:

```
1. {
2.   "jsonrpc": "2.0",
3.   "result": {
4.     "triggerids": [
5.       "12002",
6.       "12003"
7.     ]
8.   },
9.   "id": 1
}
```

trigger.delete

```
10. }
```

Source来源

CTrigger::delete() in *frontends/php/include/classes/api/services/CTrigger.php*.

trigger.deletedependencies

Description说明

```
object trigger.deletedependencies(string/array triggers)
```

This method allows to delete all trigger dependencies from the given triggers该方法允许从指定的触发器中删除所有的触发依赖关系 .

Parameters参数

(string/array) Triggers to delete the trigger dependencies from. (string / array) 从中删除触发器依赖关系的触发器 .

Return values返回值

(object) Returns an object containing the IDs of the affected triggers under the triggerids property. (object) 返回一个包含'triggerids'属性下的受影响触发器的ID的对象

Examples范例

Deleting dependencies from multiple triggers从多个触发器中删除依赖关系

Delete all dependencies from two triggers从两个触发器中删除所有依赖关系 .

Request请求:

```

1. {
2.   "jsonrpc": "2.0",
3.   "method": "trigger.deleteDependencies",
4.   "params": [
5.     {
6.       "triggerid": "14544"
7.     },
8.     {
9.       "triggerid": "14545"
10.    }
11.  ],
12.  "auth": "038e1d7b1735c6a5436ee9eae095879e",
13.  "id": 1
14. }
```

Response响应:

```
1. {
```

```
2.     "jsonrpc": "2.0",
3.     "result": {
4.       "triggerids": [
5.         "14544",
6.         "14545"
7.       ]
8.     },
9.     "id": 1
10. }
```

See also参考

- [trigger.update](#)

Source来源

`CTrigger::deleteDependencies()` in
frontends/php/include/classes/api/services/CTrigger.php.

trigger.get

Description说明

`integer/array trigger.get(object parameters)` 该方法用于指定的参数检索触发器 .

Parameters参数

(object) 定义所需输出的参数。

该方法提供以下参数 .

参数	类型	说明
triggerids	string/array	只返回指定 ID 的触发器 .
groupids	string/array	只返回指定主机组所属主机的触发器 .
templateids	string/array	只返回指定模板的触发器 .
hostids	string/array	只返回指定主机所属的触发器 .
itemids	string/array	只返回指定监控项所属的触发器 .
applicationids	string/array	只返回指定应用集包含监控项所属的触发器 .
functions	string/array	只返回指定函数所属的触发器 . 有关支持的功能列表, 请参阅 supported trigger functions 页面
group	string	只返回指定名称的主机组中属于主机的触发器 .
host	string	只返回指定名称的主机的触发器 .
inherited	boolean	如果设置为“true”, 则只返回从模板继承的触发器 .
templated	boolean	如果设置为“true”, 则只返回所属模板的触发器 .
monitored	flag	只返回所属被监控主机启用触发器, 且包含已启用的监控项 .
active	flag	只返回所属被监控主机的已启用触发器 .
maintenance	boolean	如果设置为“true”, 则只返回所属维护中主机的已启用触发器 .
withUnacknowledgedEvents	flag	只返回具有未确认事件的触发器 .
withAcknowledgedEvents	flag	只返回所有事件确认的触发器 .
withLastEventUnacknowledged	flag	只返回最后一个未确认事件的触发器 .
skipDependent	flag	在依赖于其他触发器的问题状态中跳过触发器。请注意: 如果禁用, 禁用监控项或禁用主机监控项, 其他触发器将被忽略
lastChangeSince	timestamp	只返回指定时间之后改变状态的触发器 .
lastChangeTill	timestamp	只返回指定时间之前更改其状态的触发器 .
onlytrue	flag	只返回最近处于问题状态的触发器 .

min_severity	integer	只返回严重级别大于或等于给定严重性的触发器.
expandComment	flag	在触发器注释中扩展宏.
expandDescription	flag	以触发器的名称扩展宏.
expandExpression	flag	扩展触发器表达式中的函数和宏.
selectGroups	query	在 <code>groups</code> 属性中返回触发器所属的主机组.
selectHosts	query	在 <code>hosts</code> 属性中返回触发器所属的主机.
selectItems	query	在 <code>items</code> 属性中返回触发器所包含的监控项.
selectFunctions	query	Return functions used in the trigger in the <code>functions</code> property. 函数对象表示触发器表达式中使用的函数，并具有以下属性： <code>functionid</code> - (string) 函数的ID; <code>itemid</code> - (string) 函数中使用的监控项的 ID; <code>function</code> - (string) 函数名; <code>parameter</code> - (string) 参数传递给函数.
selectDependencies	query	返回触发器依赖于 <code>dependencies</code> 属性的触发器.
selectDiscoveryRule	query	返回创建触发器的低水平发现规则.
selectLastEvent	query	返回 <code>lastEvent</code> 属性中的最后一个触发事件.
selectTags	query	在“标签”属性中返回触发器标记.
filter	object	只返回与给定过滤器完全匹配的结果. 接受一个数组，其中keys是属性名称，并且值是单个值或要匹配值的数组. 支持新增的过滤器： <code>host</code> - technical name of the host that the trigger belongs to触发器所属主机的技术性名称; <code>hostid</code> - ID of the host that the trigger belongs to触发器所属主机的 ID.
limitSelects	integer	限制子选择返回的记录数. 适用于以下子选择： <code>selectHosts</code> - results will be sorted by <code>host</code> 结果将按 <code>host</code> 分类.
sortfield	string/array	<code>Sort</code> the result by the given properties给定属性的结果.Possible values are可能的值为: <code>triggerid</code> , <code>description</code> , <code>status</code> , <code>priority</code> , <code>lastchange</code> and <code>hostname</code> .
countOutput	flag	These parameters being common for all <code>get</code> methods are described in detail in the reference commentary page这些参数对于所有的 <code>get</code> 方法是常见的，在 reference commentary 页面中有详细描述.
editable	boolean	
excludeSearch	flag	
limit	integer	
output	query	
preservekeys	flag	
search	object	
searchByAny	boolean	

searchWildcardsEnabled	boolean
sortorder	string/array
startSearch	flag

Return values

(integer/array) Returns either 返回两者其中任一：

- 一组对象；
- 如果已经使用了“countOutput”参数，则检索对象的计数。

范例

从触发器ID中检索数据

检索触发器“14062”中使用的所有数据和功能。

请求：

```

1. {
2.     "jsonrpc": "2.0",
3.     "method": "trigger.get",
4.     "params": {
5.         "triggerids": "14062",
6.         "output": "extend",
7.         "selectFunctions": "extend"
8.     },
9.     "auth": "038e1d7b1735c6a5436ee9eae095879e",
10.    "id": 1
11. }
```

响应：

```

1. {
2.     "jsonrpc": "2.0",
3.     "result": [
4.         {
5.             "functions": [
6.                 {
7.                     "functionid": "13513",
8.                     "itemid": "24350",
9.                     "function": "diff",
10.                    "parameter": "0"
11.                }
12.            ],
13.            "triggerid": "14062",
14.            "expression": "{13513}>0",
15.            "description": "/etc/passwd has been changed on {HOST.NAME}",
```

```
trigger.get
```

```
16.         "url": "",  
17.         "status": "0",  
18.         "value": "0",  
19.         "priority": "2",  
20.         "lastchange": "0",  
21.         "comments": "",  
22.         "error": "",  
23.         "templateid": "10016",  
24.         "type": "0",  
25.         "state": "0",  
26.         "flags": "0",  
27.         "recovery_mode": "0",  
28.         "recovery_expression": "",  
29.         "correlation_mode": "0",  
30.         "correlation_tag": "",  
31.         "manual_close": "0"  
32.     }  
33. ],  
34. "id": 1  
35. }
```

在问题状态下检索触发器

检索在问题状态下的所有触发器的ID，名称和严重性，并按严重性级别按降序分类。.

请求：

```
1. {  
2.     "jsonrpc": "2.0",  
3.     "method": "trigger.get",  
4.     "params": {  
5.         "output": [  
6.             "triggerid",  
7.             "description",  
8.             "priority"  
9.         ],  
10.        "filter": {  
11.            "value": 1  
12.        },  
13.        "sortfield": "priority",  
14.        "sortorder": "DESC"  
15.    },  
16.    "auth": "038e1d7b1735c6a5436ee9eae095879e",  
17.    "id": 1  
18. }
```

响应：

```
1. {  
2.     "jsonrpc": "2.0",  
3.     "result": [
```

```
trigger.get
```

```
4.      {
5.          "triggerid": "13907",
6.          "description": "Zabbix self-monitoring processes < 100% busy",
7.          "priority": "4"
8.      },
9.      {
10.         "triggerid": "13824",
11.         "description": "Zabbix discoverer processes more than 75% busy",
12.         "priority": "3"
13.     }
14. ],
15. "id": 1
16. }
```

使用标签检索特定触发器

使用标签检索特定触发器 .

请求：

```
1. {
2.     "jsonrpc": "2.0",
3.     "method": "trigger.get",
4.     "params": {
5.         "output": [
6.             "triggerid",
7.             "description"
8.         ],
9.         "selectTags": "extend",
10.        "triggerids": [
11.            "17578"
12.        ]
13.    },
14.    "auth": "038e1d7b1735c6a5436ee9eae095879e",
15.    "id": 1
16. }
```

响应：

```
1. {
2.     "jsonrpc": "2.0",
3.     "result": [
4.         {
5.             "triggerid": "17370",
6.             "description": "Service status",
7.             "tags": [
8.                 {
9.                     "tag": "service",
10.                     "value": "{$ITEM.VALUE}.regsub(\"Service (.*) has stopped\", \"$\\1\")"
11.                 },
12.                 {
13.                     "tag": "status",
14.                     "value": "{$ITEM.VALUE}.regsub(\"Service (.*) has stopped\", \"$\\1\")"
15.                 }
16.             ]
17.         }
18.     ]
19. }
```

```
trigger.get
```

```
13.          "tag": "error",
14.          "value": ""
15.      }
16.  ]
17. }
18. ],
19. "id": 1
20. }
```

参考

- [Discovery rule](#)
- [Item](#)
- [Host](#)
- [Host group](#)

来源

`CTrigger::get()` in *frontends/php/include/classes/api/services/CTrigger.php*.

trigger.update

Description说明

```
object trigger.update(object/array triggers)
```

This method allows to update existing triggers此方法用于更新目前的触发器.

Parameters参数

(object/array) Trigger properties to be updated需要被更新的触发器属性.

The `triggerid` property must be defined for each trigger, all other properties are optional. Only the passed properties will be updated, all others will remain unchanged. `triggerid` 属性必须在每个应用集中已定义，其他所有属性为可选项。只有传递过去的属性会被更新，其他所有属性仍然保持不变。

Additionally to the `standard trigger properties` the method accepts the following parameters除 `standard trigger properties`之外，该方法接受以下参数。.

Parameter参数	Type类型	Description说明
dependencies	array	Triggers that the trigger is dependent on触发触发器依赖. The triggers must have the <code>triggerid</code> property defined触发器必须定义 <code>triggerid</code> 属性.
tags	array	Trigger tags触发器标签.

The trigger expression has to be given in its expanded form触发器表达式必须以其扩展形式给出.

Return values返回值

(object) Returns an object containing the IDs of the updated triggers under the `triggerids` property. (object) 返回一个 `triggerids` 属性下已更新应用集的ID的对象

Examples范例

Enabling a trigger启用触发器

Enable a trigger, that is, set its status to 0.启用触发器，即将其状态设置为0。

Request请求：

```
1. {
2.   "jsonrpc": "2.0",
3.   "method": "trigger.update",
```

```
trigger.update
```

```
4.     "params": {
5.         "triggerid": "13938",
6.         "status": 0
7.     },
8.     "auth": "038e1d7b1735c6a5436ee9eae095879e",
9.     "id": 1
10. }
```

Response响应：

```
1. {
2.     "jsonrpc": "2.0",
3.     "result": {
4.         "triggerids": [
5.             "13938"
6.         ],
7.         "id": 1
8.     }
9. }
```

Replacing triggers tags替换触发器标签

Replace tags for trigger替换触发器的标签。

Request请求：

```
1. {
2.     "jsonrpc": "2.0",
3.     "method": "trigger.update",
4.     "params": {
5.         "triggerid": "13938",
6.         "tags": [
7.             {
8.                 "tag": "service",
9.                 "value": "{$ITEM.VALUE}.regsub(\"Service (.*) has stopped\", \"$\\1\")"
10.            },
11.            {
12.                "tag": "error",
13.                "value": ""
14.            }
15.        ]
16.    },
17.    "auth": "038e1d7b1735c6a5436ee9eae095879e",
18.    "id": 1
19. }
```

Response响应：

```
1. {
2.     "jsonrpc": "2.0",
```

```
3.     "result": {
4.         "triggerids": [
5.             "13938"
6.         ]
7.     },
8.     "id": 1
9. }
```

See also 参见

- [trigger.adddependencies](#)
- [trigger.deletedependencies](#)

Source 来源

`CTrigger::update()` in `frontends/php/include/classes/api/services/CTrigger.php`.

趋势

该类旨在查询趋势数据 .

对象引用:

- [Trend](#)

可用方法:

- [trend.get](#) - 检索趋势数据

> Trend object趋势对象

The following objects are directly related to the trend API. 以下对象是与 trend API 有直接关系 .

Trend objects differ depending on the item's type of information. They are created by the Zabbix server and cannot be modified via the API. 趋势对象根据item的信息类型而有所不同. 它们由Zabbix server创建，不能通过API进行修改。

Float trend浮点型趋势

The float trend object has the following properties 浮点型趋势对象具有以下属性.

Property参数	Type类型	Description说明
clock	timestamp	Time when that value was received 获取此值的时间.
itemid	string	ID of the related item 相关项的ID.
num	integer	Number of values within this hour 在该小时内的数值.
value_min	float	Hourly minimum value 每小时最小值.
value_avg	float	Hourly average value 每小时平均值.
value_max	float	Hourly maximum value 每小时最大值.

Integer trend整型趋势

The integer trend object has the following properties 整型趋势对象具有以下属性.

Property参数	Type类型	Description说明
clock	timestamp	Time when that value was received 获取此值的时间.
itemid	string	ID of the related item 相关项的ID.
num	integer	Number of values within this hour 在该小时内的数值.
value_min	integer	Hourly minimum value 每小时最小值.
value_avg	integer	Hourly average value 每小时平均值.
value_max	integer	Hourly maximum value 每小时最大值.

trend.get

Description说明

```
integer/array trend.get(object parameters)
```

The method allows to retrieve trend data according to the given parameters该方法用于根据规定的参数获取趋势数据.

Parameters参数

(object) Parameters defining the desired output定义所需输出的参数.

The method supports the following parameters该方法提供以下参数.

Parameter参数	Type类型	Description说明
itemids	string/array	Return only trends with the given item IDs只返回包含特定监控项的趋势.
time_from	timestamp	Return only values that have been collected after or at the given time只返回包含特定时间后或之后获取的值.
time_till	timestamp	Return only values that have been collected before or at the given time只返回包含特定时间前或之前获取的值.
countOutput	flag	Count the number of retrieved objects计算检索对象的数量.
limit	integer	Limit the amount of retrieved objects限制检索对象的数量.
output	query	Set fields to output将字段设置为输出.

Return values返回值

(integer/array) Returns either 返回两者其中任一:

- an array of objects一组对象;
- the count of retrieved objects, if the `countOutput` parameter has been used.如果已经使用了“countOutput”参数，则检索对象的计数.

Examples范例

Retrieving item trend data从监控项中检索趋势数据

Request请求:

```
1. {
2.     "jsonrpc": "2.0",
3.     "method": "trend.get",
```

```
trend.get
```

```
4.     "params": {
5.         "output": [
6.             "itemid",
7.             "clock",
8.             "num",
9.             "value_min",
10.            "value_avg",
11.            "value_max",
12.        ],
13.        "itemids": [
14.            "23715"
15.        ],
16.        "limit": "1"
17.    },
18.    "auth": "038e1d7b1735c6a5436ee9eae095879e",
19.    "id": 1
20. }
```

Response响应：

```
1. {
2.     "jsonrpc": "2.0",
3.     "result": [
4.         {
5.             "itemid": "23715",
6.             "clock": "1446199200",
7.             "num": "60",
8.             "value_min": "0.1650",
9.             "value_avg": "0.2168",
10.            "value_max": "0.3500",
11.        }
12.    ],
13.    "id": 1
14. }
```

Source来源

CTrend::get() in *frontends/php/include/classes/api/services/CTrend.php*.

配置

这个类用于导入和导出 Zabbix 的配置数据。

相关方法：

- `configuration.export` - 导出配置
- `configuration.import` - 导入配置

configuration.export

Description

```
string configuration.export(object parameters)
```

This method allows to export configuration data as a serialized string.此方法允许作为序列化字符串导出配置数据。

Parameters参数

(object) Parameters defining the objects to be exported and the format to use.参数定义了导出的对象以及使用的格式。

参数	类型	说明
format(required) (必须)	string	Format in which the data must be exported.导出数据的格式。可能的值为: <code>json</code> - JSON; <code>xml</code> - XML.
options(required) (必须)	object	Objects to be exported. 导出的对象。The <code>options</code> object has the following parameters:对象有以下参数: <code>groups</code> - (array) IDs of host groups to export; 主机组ID的导出; <code>hosts</code> - (array) IDs of hosts to export; 主机ID的导出; <code>images</code> - (array) IDs of images to export; 图表ID的导出; <code>maps</code> - (array) IDs of maps to export. 拓扑图ID的导出; <code>screens</code> - (array) IDs of screens to export; 屏幕ID的导出; <code>templates</code> - (array) IDs of templates to export; 模板ID的导出; <code>valueMaps</code> - (array) IDs of value maps to export; 值映射ID的导出;

Return values返回值

(string) Returns a serialized string containing the requested configuration data.返回一个序列化字符串包含请求的配置数据。

Examples 范例

Exporting a host导出一个主机。

Export the configuration of a host as an XML string.导出一个XML字符串的主机配置。

Request:

```

1. {
2.   "jsonrpc": "2.0",
3.   "method": "configuration.export",
4.   "params": {
5.     "options": {
6.       "hosts": [

```

```

7.          "10161"
8.      ],
9.    },
10.   "format": "xml"
11. },
12.   "auth": "038e1d7b1735c6a5436ee9eae095879e",
13.   "id": 1
14. }

```

Response:

```

1. {
2.   "jsonrpc": "2.0",
3.   "result": "<?xml version=\"1.0\" encoding=\"UTF-8\"?>\n<zabbix_export><version>3.4</version><date>2017-02-13T14:27:34Z</date><groups><group><name>Zabbix servers</name></group></groups><hosts><host><host>Export host</host><name>Export host</name><description/><proxy/><status>0</status><ipmi_authtype>-1</ipmi_authtype><ipmi_privilege>2</ipmi_privilege><ipmi_username/><ipmi_password/><tls_connect>1</tls_connect><tls_accept>1</tls_accept><tls_issuer/><tls_subject/><tls_psk_identity/><tls_psk/><templates/><groups><group><name>Zabbix servers</name></group></groups><interfaces><interface><default>1</default><type>1</type><useip>1</useip><ip>127.0.0.1</ip><dns></dns><port>10050</port><bulk>1</bulk><interface_ref>if1</interface_ref></interface></interfaces><applications><application><name>Application</name></application></applications><items><item><name>Item</name><type>0</type><snmp_community/><snmp_oid/><key>item.key</key><delay>30s</delay><history>90d</history><trends>365d</trends><status>0</status><value_type>3</value_type><allowed_hosts/><units/><snmpv3_contextname/><snmpv3_securityname/><snmpv3_securitylevel>0</snmpv3_securitylevel><snmpv3_authprotocol>0</snmpv3_authprotocol><snmpv3_authpassphrase/><snmpv3_privprotocol>0</snmpv3_privprotocol><snmpv3_privpassphrase/><params/><ipmi_sensor/><authtype>0</authtype><username/><password/><publickey/><privatekey/><port/><description/><inventory_link>0</inventory_link><applications><application><name>Application</name></application></applications><valuemap><name>Host status</name></valuemap><logtimefmt/><preprocessing/><interface_ref>if1</interface_ref></item></items><discovery_rules/><macros/><inventory/></host></hosts><triggers><trigger><expression>{Export host:item.key.last()}=0</expression><name>Trigger</name><url/><status>0</status><priority>2</priority><description>Host trigger</description><type>0</type><recovery_mode>1</recovery_mode><recovery_expression>{Export host:item.key.last()}=2</recovery_expression><dependencies/><tags/><correlation_mode>1</correlation_mode><correlation_tag>Tag 01</correlation_tag><manual_close>0</manual_close></trigger></triggers><graphs><graph><name>Graph</name><width>900</width><height>200</height><yaxismin>0.0000</yaxismin><yaxismax>100.0000</yaxismax><show_work_period>1</show_work_period><show_triggers>1</show_triggers><type>0</type><show_legend>1</show_legend><show_3d>0</show_3d><percent_left>0.0000</percent_left><percent_right>0.0000</percent_right><ymin_type_1>0</ymin_type_1><ymax_type_1>0</ymax_type_1><ymin_item_1>0</ymin_item_1><ymax_item_1>0</ymax_item_1><graph_items><graph_item><sortorder>0</sortorder><drawtype>0</drawtype><color>C80000</color><yaxisside>0</yaxisside><calc_fnc>7</calc_fnc><type>0</type><item><host>Export host</host><key>item.key</key></item></graph_item></graph_items></graph></graphs><value_maps><value_map><name>Host status</name><mappings><mapping><value>0</value><newvalue>Up</newvalue></mapping><mapping><value>2</value><newvalue>Unreachable</newvalue></mapping></mappings></value_map></value_maps></zabbix_export>\n",
4.   "id": 1
5. }

```

Source来源

```
CConfiguration::export() in
frontends/php/include/classes/api/services/CConfiguration.php.
```

configuration.import

Description 说明

```
boolean configuration.import(object parameters)
```

This method allows to import configuration data from a serialized string.此方法允许作为序列化字符串导入配置数据。

Parameters参数

(object) Parameters containing the data to import and rules how the data should be handled.参数包含导入的数据以及如何处理数据的规则。

参数	类型	说明
format(required) (必须)	string	Format of the serialized string. 序列化字符串的格式。 Possible values: 可能的值为: <code>json</code> - JSON; <code>xml</code> - XML.
source(required) (必须)	string	Serialized string containing the configuration data. 序列化字符串包含的配置数据。
rules(required) (必须)	object	Rules on how new and existing objects should be imported. 关于将新的和已有的对象 应该如何导入的规则。The <code>rules</code> parameter is described in detail in the table below. 参数在下表详细描述。

If no rules are given, the configuration will not be updated.如果没有给定规则，配置将不会被更新。

The `rules` object supports the following parameters.对象提供以下参数。

参数	类型	说明
applications	object	Rules on how to import applications.关于如何导入应用集的规则。 Supported parameters: 支持的参数: <code>createMissing</code> - <code>(boolean)</code> if set to <code>true</code> , new applications will be created; default: <code>false</code> ; 如果设置为true, 新的应用集将会被创建; 默认: false; <code>deleteMissing</code> - <code>(boolean)</code> if set to <code>true</code> , applications not present in the imported data will be deleted from the database; default: <code>false</code> . 如果设置为true, 不在导入数据中的应用集将会从数据库中被删除; 默认: false.
discoveryRules	object	Rules on how to import LLD rules.关于如何导入底层自动发现规则的规则。 Supported parameters:支持的参数: <code>createMissing</code> - <code>(boolean)</code> if set to <code>true</code> , new LLD rules will be created; default: <code>false</code> ; 如果设置为true, 新的底层自动发现规则将会被创建; 默认: false; <code>updateExisting</code> - <code>(boolean)</code> if set to <code>true</code> , existing LLD rules will be updated; default: <code>false</code> ; 如果设置为true, 已有的底层自动发现规则将会被更新; 默认: false; <code>deleteMissing</code> - <code>(boolean)</code> if set to <code>true</code> , LLD rules not present in the imported data will be deleted from the database; default: <code>false</code> .如果设置为true, 不在导入数据中的底层自动发现规则将会从数据库中被删除; 默认false.
		Rules on how to import graphs.关于如何导入图表的规则。

graphs	object	Supported parameters: 支持的参数: <code>createMissing</code> - <code>(boolean)</code> if set to <code>true</code> , new graphs will be created; default: <code>false</code> ; 新的图表将会被创建; 默认: <code>false</code> ; <code>updateExisting</code> - <code>(boolean)</code> if set to <code>true</code> , existing graphs will be updated; default: <code>false</code> ; 如何设置为true, 已有的图表将会被更新; 默认: <code>false</code> ; <code>deleteMissing</code> - <code>(boolean)</code> if set to <code>true</code> , graphs not present in the imported data will be deleted from the database; default: <code>false</code> . 如果设置为true, 不在导入数据中的图表将会从数据库中被删除; 默认: <code>false</code> 。
groups	object	Rules on how to import host groups. 关于如何导入主机组的规则。 Supported parameters: 支持的参数: <code>createMissing</code> - <code>(boolean)</code> if set to <code>true</code> , new host groups will be created; default: <code>false</code> . 如果设置为true, 新的主机组将会被创建; 默认: <code>false</code> 。
hosts	object	Rules on how to import hosts. 关于如何导入主机的规则。 Supported parameters: 支持的参数: <code>createMissing</code> - <code>(boolean)</code> if set to <code>true</code> , new hosts will be created; default: <code>false</code> ; 如果设置为true, 新的主机将会被创建; 默认: <code>false</code> ; <code>updateExisting</code> - <code>(boolean)</code> if set to <code>true</code> , existing hosts will be updated; default: <code>false</code> . 如果设置为true, 已有的主机将会被更新; 默认: <code>false</code> 。
images	object	Rules on how to import images. 关于如何导入图片的规则。 Supported parameters: 支持的参数: <code>createMissing</code> - <code>(boolean)</code> if set to <code>true</code> , new images will be created; default: <code>false</code> ; 如果设置为true, 新的图片将会被创建; 默认: <code>false</code> ; <code>updateExisting</code> - <code>(boolean)</code> if set to <code>true</code> , existing images will be updated; default: <code>false</code> . 如果设置为true, 已有的图片将会被创建; 默认: <code>false</code> 。
items	object	Rules on how to import items. 关于如何导入监控项的规则。 Supported parameters: 支持的参数: <code>createMissing</code> - <code>(boolean)</code> if set to <code>true</code> , new items will be created; default: <code>false</code> ; 如果设置为true, 新的监控项将会被创建; 默认: <code>false</code> ; <code>updateExisting</code> - <code>(boolean)</code> if set to <code>true</code> , existing items will be updated; default: <code>false</code> ; 如果设置为true, 已有的监控项将会被更新; 默认: <code>false</code> ; <code>deleteMissing</code> - <code>(boolean)</code> if set to <code>true</code> , items not present in the imported data will be deleted from the database; default: <code>false</code> . 如果设置为true, 不在导入数据中的监控项将会从数据库中被删除; 默认: <code>false</code> 。
maps	object	Rules on how to import maps. 关于如何导入拓扑图的规则。 Supported parameters: 支持的参数: <code>createMissing</code> - <code>(boolean)</code> if set to <code>true</code> , new maps will be created; default: <code>false</code> ; 如果设置为true, 新的拓扑图将会被创建; 默认: <code>false</code> ; <code>updateExisting</code> - <code>(boolean)</code> if set to <code>true</code> , existing maps will be updated; default: <code>false</code> . 如果设置为true, 已有的拓扑图将会被更新; 默认: <code>false</code> 。
screens	object	Rules on how to import screens. 关于如何导入屏幕的规则。 Supported parameters: 支持的参数: <code>createMissing</code> - <code>(boolean)</code> if set to <code>true</code> , new screens will be created; default: <code>false</code> ; 如果设置为true, 新的屏幕将会被创建; 默认: <code>false</code> ; <code>updateExisting</code> - <code>(boolean)</code> if set to <code>true</code> , existing screens will be updated; default: <code>false</code> . 如果设置为true, 已有的屏幕将会被更新; 默认: <code>false</code> 。
templateLinkage	object	Rules on how to import template links. 关于如何导入模板链接的规则。 Supported parameters: 支持的参数: <code>createMissing</code> - <code>(boolean)</code> if set to <code>true</code> , new links between templates and host will be created; default: <code>false</code> . .如果设置为true, 新的连接模板和主机的链接将会被创建; 默认: <code>false</code> 。
		Rules on how to import templates. 关于如何导入模板的规则。

templates	object	Supported parameters: 支持的参数： <code>createMissing</code> - <code>(boolean)</code> if set to <code>true</code> , new templates will be created; default: <code>false</code> ; <code>updateExisting</code> - <code>(boolean)</code> if set to <code>true</code> , existing templates will be updated; default: <code>false</code> . 如果设置为true, 新的模板将会被创建; 默认: false; <code>default</code> : <code>false</code> .
templateScreens	object	Rules on how to import template screens. 关于如何导入屏幕模板的规则。 Supported parameters: 支持的参数： <code>createMissing</code> - <code>(boolean)</code> if set to <code>true</code> , new template screens will be created; default: <code>false</code> ; <code>updateExisting</code> - <code>(boolean)</code> if set to <code>true</code> , existing template screens will be updated; default: <code>false</code> ; <code>deleteMissing</code> - <code>(boolean)</code> if set to <code>true</code> , template screens not present in the imported data will be deleted from the database; default: <code>false</code> . 如果设置为true, 不在导入数据中的屏幕模板将会在数据库中被删除; 默认: false.
triggers	object	Rules on how to import triggers. 关于如何导入触发器的规则。 Supported parameters: 支持的参数： <code>createMissing</code> - <code>(boolean)</code> if set to <code>true</code> , new triggers will be created; default: <code>false</code> ; <code>updateExisting</code> - <code>(boolean)</code> if set to <code>true</code> , existing triggers will be updated; default: <code>false</code> ; <code>deleteMissing</code> - <code>(boolean)</code> if set to <code>true</code> , triggers not present in the imported data will be deleted from the database; default: <code>false</code> . 如果设置为true, 不在导入数据中的触发器将会在数据库中被删除; 默认: false;
valueMaps	object	Rules on how to import value maps. 关于如何导入植映射的规则。 Supported parameters: 支持的参数： <code>createMissing</code> - <code>(boolean)</code> if set to <code>true</code> , new value maps will be created; default: <code>false</code> ; <code>updateExisting</code> - <code>(boolean)</code> if set to <code>true</code> , existing value maps will be updated; default: <code>false</code> . 如果设置为true, 已有的植映射将会被更新; 默认: false.

Return values 返回值

`(boolean)` Returns `true` if importing has been successful. 如果导入成功则返回true。

Examples范例

Importing hosts and items 导入主机和监控项

Import the host and items contained in the XML string. 导入的主机和监控项包含在XML字符串中。 If any items in XML are missing, they will be deleted from the database, and everything else will be left unchanged. 如果在XML中遗漏了任何监控项，这些监控项将会在数据库中被删除，其他的则不改变。

Request:

```

1. {
2.   "jsonrpc": "2.0",
3.   "method": "configuration.import",
4.   "params": {

```

```

5.      "format": "xml",
6.      "rules": {
7.          "hosts": {
8.              "createMissing": true,
9.              "updateExisting": true
10.         },
11.         "items": {
12.             "createMissing": true,
13.             "updateExisting": true,
14.             "deleteMissing": true
15.         }
16.     },
17.     "source": "<?xml version=\"1.0\" encoding=\"UTF-8\"?><zabbix_export><version>3.4</version><date>2012-04-18T11:20:14Z</date><groups><group><name>Zabbix servers</name></group></groups><hosts><host><host>Export host</host><name>Export host</name><description/><proxy/><status>0</status><ipmi_authtype>-1</ipmi_authtype><ipmi_privilege>2</ipmi_privilege><ipmi_username/><ipmi_password/><tls_connect>1</tls_connect><tls_accept>1</tls_accept><tls_issuer/><tls_subject/><tls_psk_identity/><tls_psk/><templates/><groups><group><name>Zabbix servers</name></group></groups><interfaces><interface><default>1</default><type>1</type><useip>1</useip><ip>127.0.0.1</ip><dns/><port>10050</port><bulk>1</bulk><interface_ref>if1</interface_ref></interface></interfaces><applications><application><name>Application</name></application></applications><items><item><name>Item</name><type>0</type><snmp_community/><snmp_oid/><key>item.key</key><delay>30s</delay><history>90d</history><trends>365d</trends><status>0</status><value_type>3</value_type><allowed_hosts/><units/><snmpv3_contextname/><snmpv3_securityname/><snmpv3_securitylevel>0</snmpv3_securitylevel><snmpv3_authprotocol>0</snmpv3_authprotocol><snmpv3_authpassphrase/><snmpv3_privprotocol>0</snmpv3_privprotocol><snmpv3_privpassphrase/><params/><ipmi_sensor/><authtype>0</authtype><username/><password/><publickey/><privatekey/><port/><description/><inventory_link>0</inventory_link><applications><application><name>Application</name></application></applications><valuemap><name>Host status</name><valuemap><logtimefmt/><preprocessing/><interface_ref>if1</interface_ref></item></items><discovery_rules/><macros/><inventory/></host></hosts><triggers><trigger><expression>{Export host:item.key.last()}=0</expression><name>Trigger</name><url/><status>0</status><priorty>2</priority><description>Host trigger</description><type>0</type><recovery_mode>1</recovery_mode><recovery_expression>{Export host:item.key.last()}=2</recovery_expression><dependencies/><tags/><correlation_mode>1</correlation_mode><correlation_tag>Tag 01</correlation_tag><manual_close>0</manual_close></trigger></triggers><graphs><graph><name>Graph</name><width>900</width><height>200</height><yaxismin>0.0000</yaxismin><yaxismax>100.0000</yaxismax><show_work_period>1</show_work_period><show_triggers>1</show_triggers><type>0</type><show_legend>1</show_legend><show_3d>0</show_3d><percent_left>0.0000</percent_left><percent_right>0.0000</percent_right><ymin_type_1>0</ymin_type_1><ymax_type_1>0</ymax_type_1><ymin_item_1>0</ymin_item_1><ymax_item_1>0</ymax_item_1><graph_items><graph_item><sortorder>0</sortorder><drawtype>0</drawtype><color>C80000</color><yaxisside>0</yaxisside><calc_fnc>7</calc_fnc><type>0</type><item><host>Export host</host><key>item.key</key><item><graph_item></graph_item></graph_items></graph></graphs><value_maps><value_map><name>Host status</name><mappings><mapping><value>0</value><newvalue>Up</newvalue></mapping></mappings></value_map></value_maps></zabbix_export>"
18.   },
19.   "auth": "038e1d7b1735c6a5436ee9eae095879e",
20.   "id": 1
21. }
```

Response:

```

1. {
2.   "jsonrpc": "2.0",
3.   "result": true,
4.   "id": 1
5. }
```

Source 来源

```
CConfiguration::import() in  
frontends/php/include/classes/api/services/CConfiguration.php.
```

附录1.参考

注释

数据类型

Zabbix API支持以下数据类型：

Type	Description
bool	布尔值，接受 <code>true</code> 或 <code>false</code> 。
flag	如果它被传递而不等于 <code>null</code> 和 <code>false</code> ，该值被认为是 <code>true</code> 。
integer	一个整数。
float	浮点数。
string	文本字符串。
text	较长的文本字符串。
timestamp	Unix时间戳。
array	有序的值序列，即数组。
object	关联数组。
query	用于定义应返回哪些数据的值。可以定义为一个属性名称数组，仅返回特定属性，或者作为预定值之一： <code>extend</code> - 返回所有对象属性； <code>count</code> - 返回检索的记录数，仅由某些子选择器支持。

属性标签

一些对象属性用短标签标记以描述它们的行为。使用以下标签：

- `readonly` - 一些对象属性用短标签标记以描述它们的行为。使用以下标签：
- `constant` - 属性的值可以在创建对象时设置，但不能在之后更改。

常用的“get”方法参数

所有get方法都支持以下参数：

Parameter	Type	Description
<code>countOutput</code>	flag	返回结果中的记录数，而不是实际数据。
<code>editable</code>	boolean	如果设置为 <code>true</code> 仅返回用户有写权限的对象。默认值： <code>false</code> 。
<code>excludeSearch</code>	flag	返回与参数中给定的条件不匹配的结果 <code>search</code> 。
<code>filter</code>	object	仅返回与给定过滤器完全匹配的结果。接受数组，其中键是属性名，值可以是单个值或要匹配的值的数组。不适用于 <code>text</code> 。
<code>limit</code>	integer	限制返回的记录数。

output	query	要返回的对象属性。 默认值： <code>extend</code> 。
preservekeys	flag	在结果数组中使用ID作为键。
search	object	返回与给定通配符搜索匹配的结果。接受数组，其中键是属性名，值是要搜索的字符串。如果没有给出额外的选项，这将执行 <code>LIKE "%...%"</code> 搜索。仅适用于 <code>string</code> 和 <code>text</code> 。
searchByAny	boolean	如果设置为 <code>true</code> 返回与 <code>filter</code> 或 <code>search</code> 参数中给定的任何条件匹配的结果，“search”而不是所有条件 默认值： <code>false</code> 。
searchWildcardsEnabled	boolean	如果设置为 <code>true</code> 启用在参数中使用 “*” 作为通配符 <code>search</code> 。 默认值： <code>false</code> 。
sortfield	string/array	按给定属性对结果排序。请参阅特定的API <code>get</code> 方法说明，以获取可用于排序的属性列表。宏在排序之前不会展开。
sortorder	string/array	排序顺序。如果传递数组，每个值都将匹配参数中给定的相应 <code>sortfield</code> 属性。可能的值为： <code>ASC</code> - 升序； <code>DESC</code> - 降序。
startSearch	flag	<code>search</code> 参数将比较字段的开始，即执行 <code>LIKE "...%"</code> 搜索。

附录2.从3.2到3.4的变化

向后不兼容更改

通用

更改: ZBXNEXT-3570 删除对所有已弃用的 `isreadable` 和 `iswritable` 方法的支持
ZBXNEXT-3570 删除支持已弃用的 `proxy.interfaces` 参数

发现

更改: ZBXNEXT-1675 删除对 `delay flex` 的支持

监控项

更改: ZBXNEXT-1675 删除对 `delay flex` 的支持

监控项原型

更改: ZBXNEXT-1675 删除对 `delay flex` 的支持

用户

更改: ZBX-3783 用户在操作中使用时无法删除

用户组

更改: ZBX-3783 当用户组属于某个动作时，不能被删除

其他更改和bug修复

通用

更改: ZBX-5116 阻止正斜杠转义，同时将API响应编码为JSON文本

动作

更改: ZBXNEXT-1675 在 `esc_period` 字段中实现了带有后缀和用户宏的时间单位的支持

应用

更改: ZBX-3783 增加的输入参数严格验证 `create()`，`update()` 和 `delete()` 方法；提高性能

配置

更改: ZBXNEXT-1675 实现支持带后缀和用户宏的时间单位 ZBX-3783 添加了对输入参数的严格验

证 `import()` 和 `export()` 方法

发现规则

更改: ZBXNEXT-1675 实现支持带后缀, 用户宏和自定义间隔的时间单位在 `delay` 字段 ZBXNEXT-1675 实现支持时间单位与后缀和用户宏在 `lifetime` 字段

规则

更改: ZBXNEXT-1675 在 `delay` 字段中实现了带有后缀和用户宏的时间单位的支持

主机组

更改: ZBX-3783 增加的输入参数严格验证 `create()`, `update()` 和 `delete()` 方法; 提高性能

http测试

更改: ZBXNEXT-1675 在 `delay` 字段中实现了带有后缀和用户宏的时间单位的支持

图像地图

更改: ZBX-3783 增加的输入参数严格验证 `create()`, `update()` 和 `delete()` 方法; 改进的性能 ZBX-3783 `mappings.sortorder` 参数已被弃用 `create()` 和 `update()` 方法

监控项

更改: ZBXNEXT-1675 实现支持带后缀, 用户宏和自定义间隔的时间单位在 `delay` 字段 ZBXNEXT-1675 实现支持带有后缀和用户宏的时间单位 `history` 和 `trends` 字段

监控项原型

更改: ZBXNEXT-1675 实现支持带后缀, 用户宏, LLD宏和自定义间隔的时间单位实现对带有后缀, 用户宏, LLD宏和自定义间隔 `delay` 字段 ZBXNEXT-1675 实现支持带后缀, 用户宏, LLD宏和自定义间隔的时间单位实现对带有后缀, 和LLD宏的时间单位 `history` 和 `trends` 字段的支持

脚本

更改: ZBX-3783 增加的输入参数严格验证的 `create()`, `update()`, `delete()` 和 `execute()` 方法; 提高性能

用户

更改: ZBXNEXT-1675 实现支持时间单位与后缀 `autologout` 和 `refresh` 字段 ZBXNEXT-1675 媒体支持实施时间单位后缀和用户宏 `period` 场

ZBX-3783 增加的输入参数严格验证

的 `create()`, `update()`, `delete()`, `login()`, `logout()` 和 `checkauthentication()` 方法; 改进的性能 ZBX-3783 `updateprofile()`, `addmedia()`, `updatemedia()` 和 `deletemedia()` 方法已过时 ZBX-3783 增加了支

持的user_medias通过参数update()的方法

用户组

更改： ZBX-3783 增加的输入参数严格验证 `create()` , `update()` 和 `delete()` 方法；改进的性能ZBX-3783
`massadd()` 和 `massupdate()` 方法已被弃用

用户宏

更改： ZBX-3783 增加的输入参数严格验证 `createglobal()` , `updateglobal()` 和 `deleteglobal()` 方法；提高性能
能

用户媒介

更改： ZBX-3783 `get()` 方法已弃用

地图

更改： ZBX-3783 增加的输入参数严格验证 `create()` , `update()` 和 `delete()` 方法；提高性能

附录

请使用侧栏访问附录部分中的内容。

1 常见问题/疑难解答

常见问题

- Q: 可以更新或清空队列（如，管理→队列中展示的队列）？A: 不可以。
- Q: 如何从一个数据库迁移到另一个数据库？A: 只需要转存数据（对于 MySQL，用 `flag -t or --no-create-info`），用Zabbix的 `schema` 文件创建新的数据库，导入数据。
- Q: 我想用下划线替换我的监控项中所有的keys，因为空格只是在老版本中起作用，在3.0版本中空格不是一个有效的标示符（或者因为其他需要大量修改监控项key的）A: 可以使用数据库查询来用下划线替换所有出现的空格:`update items set key=replace(key, ' ', '_');`触发器可以使用这些监控项而不需要额外的改动，但是需要修改以下位置的监控项引用：*Notifications (actions) Map element and link labels* Calculated item formulas*
- Q: 我的图形中有一些点而不是线或者有一些空白，为什么会这样？A: 数据丢失，这种情况的发生有多种原因——Zabbix数据库的性能问题、Zabbix服务器问题、网络问题、监控设备问题...
- Q: Zabbix守护进程无法启动一个消息 _监听器，错误消息为：`socket() for [[-]:10050] failed with error 22: Invalid argument.`A: 这个错误发生在试图运行编译的Zabbix agent版本为2.6.27而运行在一个内核2.6.26和更低的系统上。注意，在这种情况下，静态链接不会起作用，因为它是不支持早期内核的SOCK_CLOEXEC标志的socket()系统调用。[ZBX-3395](#)
- Q: 我尝试使用一个命令设置一个灵活的用户参数(可传入参数的参数)，使用类似的位置参数 \$1，但它不起作用(使用监控项参数代替). 怎么解决这个问题？A: 使用两个 符合，像这样 `1`
- Q: 在Opera11中，所有的下拉菜单都有一个滚动条，看起来不太美观，为什么会这样呢？A: 对于Opera11.00和11.01来说，这是个bug；更多信息请访问 [Zabbix 问题跟踪](#)。
- Q: 如何更改自定义主题的背景图片？A: 参照数据库中的graph_theme表和[主题帮助](#)。
- Q: 调试等级4时，在服务器/代理日志中出现“Trapper got [] len 0”信息，- 这是什么？A: 很有可能是前端，连接检查服务器是否仍在运行。
- Q: 我的系统时间设置为将来的某一时间，导致没有数据出现。这个问题怎么解决？A: 清除数据库中的字段 `hosts.disable_until*`, `drules.nextcheck`, `httptest.nextcheck` 值，并重启服务器/代理。
- Q: 在前端，当使用 `{ITEM.VALUE}` 宏 或者其他情况下时，item值会被切割为20个符号，这种情况正常吗？A: 是的，在include/items.inc.php 中有一个硬编码限制。

另见

- [zabbix官方主页的问题解决版块](#)

2 安装

1 数据库创建脚本

概述

Zabbix数据库必须在Zabbix服务器和代理安装的时候创建。这部分提供了创建数据库的脚本。对于每一个支持的数据库都提供了不同的架构脚本。

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

对于Zabbix proxy数据库，只需要导入 `schema.sql` (不需要 `images.sql` 和 `data.sql`)

脚本

MySQL

```

1. shell> mysql -uroot -p<password>
2. mysql> create database zabbix character set utf8 collate utf8_bin;
3. mysql> grant all privileges on zabbix.* to [email protected] identified by '<password>';
4. mysql> quit;
5. shell> cd database/mysql
6. shell> mysql -uzabbix -p<password> zabbix < schema.sql
7. # stop here if you are creating database for Zabbix proxy
8. shell> mysql -uzabbix -p<password> zabbix < images.sql
9. shell> mysql -uzabbix -p<password> zabbix < data.sql

```

PostgreSQL

我们假定存在一个用户，并且有权限创建数据库。

```

1. shell> psql -U <username>
2. psql> create database zabbix;
3. psql> \q
4. shell> cd database/postgresql
5. shell> psql -U <username> zabbix < schema.sql
6. # stop here if you are creating database for Zabbix proxy
7. shell> psql -U <username> zabbix < images.sql
8. shell> psql -U <username> zabbix < data.sql

```

Oracle

我们假定存在一个 `zabbix` 数据库用户和 `password` 密码 并且有权限创建数据库 in ORCL service located on the host Oracle database server with a user shell user having write access to /tmp directory. Zabbix 要求 Unicode 数据库字符集 和 `UTF8` 通用字符集。检查当前设置：

```

1. sqlplus> select parameter,value from v$nlsparameters where parameter='NLS_CHARACTERSET' or
parameter='NLS_NCHAR_CHARACTERSET';

```

1 数据库创建脚本

如果要创建一个Zabbix server的数据库，需要在Oracle主机上指定目录有images。在Oracle主机上，复制images from misc/images/png_modern目录下的文件到 /tmp/zabbix_images 目录：

```
1. shell> cd /path/to/zabbix-sources  
2. shell> scp -r misc/images/png_modern [email protected]:/tmp/zabbix_images
```

Now prepare the database:

```
1. shell> cd database/oracle  
2. shell> sqlplus zabbix/[email protected]/ORCL  
3. sqlplus> @schema.sql  
4. # stop here if you are creating database for Zabbix proxy  
5. sqlplus> @images.sql  
6. sqlplus> @data.sql
```

执行完 images.sql 脚本后，可以删除临时目录 /tmp/zabbix_images。

IBM DB2

```
1. shell> db2 "create database zabbix using codeset utf-8 territory us pagesize 32768"  
2. shell> cd database/ibm_db2  
3. shell> db2batch -d zabbix -f schema.sql  
4. # stop here if you are creating database for Zabbix proxy  
5. shell> db2batch -d zabbix -f images.sql  
6. shell> db2batch -d zabbix -f data.sql
```

SQLite

```
1. shell> cd database/sqlite3  
2. shell> sqlite3 /var/lib/sqlite/zabbix.db < schema.sql  
3. # stop here if you are creating database for Zabbix proxy  
4. shell> sqlite3 /var/lib/sqlite/zabbix.db < images.sql  
5. shell> sqlite3 /var/lib/sqlite/zabbix.db < data.sql
```

如果Zabbix proxy使用SQLite，若没有数据库，会自动创建。

返回 [安装部分](#)。

2 Windows 下的Zabbix agent

配置 agent

Zabbix agent 作为Windows服务运行。

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

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

关于Zabbix Windows agent 更多详细信息，参考 [配置文件](#)。

主机参数

主机执行 `active checks` 时，Zabbix agent 需要定义主机名字。而且，agent端的主机名字必须和前端配置的主机名字“`Host name`”完全匹配。

agent端主机名字可以通过配置文件[configuration file](#)中的`Hostname` 或 `HostnameItem`参数定义 - 如果不指定参数值将使用默认的主机名字。

参数`HostnameItem` 的默认值即agent端key值为“`system.hostname`”的返回值，对于Windows平台返回的是NetBIOS的主机名。

`Hostname`默认值为`HostnameItem` 参数返回的值。所以，实际上，如果这两个参数都是未指定的，实际的主机名将是主机NetBIOS名称；Zabbix代理将使用NetBIOS主机名从Zabbix服务器检索活动检查列表，并将结果发送给它。

`system.hostname`键始终返回限制为15个符号的NetBIOS主机名，且仅在UPPERCASE中 - 而不管实际主机名中的长度和小写/大写字符。

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

`system.hostname[host]`键总是返回完整的，实际的（区分大小写的）Windows主机名。

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

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

主机名也用作Windows服务名称的一部分，用于安装，启动，停止和卸载Windows服务。例如，如果Zabbix代理配置文件指定 `Hostname=Windows_db_server`，那么代理将作为Windows服务安装，“`Zabbix Agent [Windows_db_server]`”。因此，要为每个Zabbix代理实例拥有不同的Windows服务名称，每个实例都必须使用不同的主机名。

将代理安装为Windows服务

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

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

在64位系统上，与运行64位进程相关的所有检查都需要64位Zabbix代理版本才能正常工作。

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

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

应指定配置文件的全路径。

Zabbix agent多实例安装命令如下：

```
1. zabbix_agentd.exe --config <configuration_file_for_instance_1> --install --multiple-agents  
2. zabbix_agentd.exe --config <configuration_file_for_instance_2> --install --multiple-agents  
3. ...  
4. zabbix_agentd.exe --config <configuration_file_for_instance_N> --install --multiple-agents
```

现在在控制面板中可以看到安装的服务。

启动 agent

启动 agent 服务，可以通过控制面板或通过命令去完成。

启动单实例的Zabbix agent，且配置文件路径为默认路径：

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

启动单实例的Zabbix agent，且配置文件路径不是默认路径：

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

启动多实例的Zabbix agent之一：

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

停止 agent

停止 agent 服务，可以通过控制面板或通过命令去完成

停止单实例的Zabbix agent，且配置文件路径为默认路径：

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

停止单实例的Zabbix agent，且配置文件路径不是默认路径：

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

停止多实例的Zabbix agent之一：

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

卸载 agent 服务

卸载单实例的Zabbix agent，且配置文件路径为默认路径：

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

卸载单实例的Zabbix agent，且配置文件路径不是默认路径：

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

停止多实例的Zabbix agent：

```
1. zabbix_agentd.exe --config <configuration_file_for_instance_1> --uninstall --multiple-agents  
2. zabbix_agentd.exe --config <configuration_file_for_instance_2> --uninstall --multiple-agents  
3. ...  
4. zabbix_agentd.exe --config <configuration_file_for_instance_N> --uninstall --multiple-agents
```

3 Elasticsearch setup

Elasticsearch support is experimental! (supported since Zabbix 3.4.5) Setup procedure considered in this section is applicable to the following Elasticsearch versions: **5.0.x -> 6.1.x**. In case an earlier or later version of Elasticsearch is used, some functionality may not work as intended.

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

Configuration

To ensure proper communication between all elements involved make sure server configuration file and frontend configuration file parameters are properly configured.

Zabbix server and frontend

Zabbix server configuration file draft with parameters to be updated:

```

1. ### Option: HistoryStorageURL
2. #     History storage HTTP[S] URL.
3. #
4. # Mandatory: no
5. # Default:
6. # HistoryStorageURL=
7. ### Option: HistoryStorageTypes
8. #     Comma separated list of value types to be sent to the history storage.
9. #
10. # Mandatory: no
11. # Default:
12. # HistoryStorageTypes=uint dbl str log text

```

Example parameter values to fill the Zabbix server configuration file with:

```

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

```

This configuration forces Zabbix server to store history values of numeric types in the corresponding database and textual history data in Elasticsearch.

Elasticsearch supports the following item types:

```
1. uint,dbl,str,log,text
```

Supported item type explanation:

Item value type	Database table	Elasticsearch type
-----------------	----------------	--------------------

Numeric (unsigned)	history_uint	uint
Numeric (float)	history	dbl
Character	history_str	str
Log	history_log	log
Text	history_text	text

Zabbix frontend configuration file (`conf/zabbix.conf.php`) draft with parameters to be updated:

```

1. // Elasticsearch url (can be string if same url is used for all types).
2. $HISTORY['url'] = [
3.     'uint' => 'http://localhost:9200',
4.     'text' => 'http://localhost:9200'
5. ];
6. // Value types stored in Elasticsearch.
7. $HISTORY['types'] = ['uint', 'text'];

```

Example parameter values to fill the Zabbix frontend configuration file with:

```

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

```

This configuration forces to store `Text`, `Character` and `Log` history values in Elasticsearch.

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

```

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

```

Installing Elasticsearch and creating mapping

Final two steps of making things work are installing Elasticsearch itself and creating a mapping process.

To install Elasticsearch please refer to [Elasticsearch installation guide](#).

Mapping is a data structure in Elasticsearch (similar to a table in a database).

Mapping for all history data types is available here: [database/elasticsearch/elasticsearch.map](#).

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

To create mapping for `text` type send the following request to Elasticsearch:

```

1. curl -X PUT \
2. http://your-elasticsearch.here:9200/text \

```

```

3. -H 'content-type:application/json' \
4. -d '{
5.   "settings" : {
6.     "index" : {
7.       "number_of_replicas" : 1,
8.       "number_of_shards" : 5
9.     }
10. },
11. "mappings" : {
12.   "values" : {
13.     "properties" : {
14.       "itemid" : {
15.         "type" : "long"
16.       },
17.       "clock" : {
18.         "format" : "epoch_second",
19.         "type" : "date"
20.       },
21.       "value" : {
22.         "fields" : {
23.           "analyzed" : {
24.             "index" : true,
25.             "type" : "text",
26.             "analyzer" : "standard"
27.           }
28.         },
29.         "index" : false,
30.         "type" : "text"
31.       }
32.     }
33.   }
34. }
35. }'

```

Similar request is required to be executed for `character` and `Log` history values mapping creation with corresponding type correction.

To work with Elasticsearch please refer to [Requirement page](#) for additional information.

`Housekeeper` is not deleting any data from Elasticsearch.

Troubleshooting

The following steps may help you troubleshoot problems with Elasticsearch setup:

- Check if the mapping is correct (GET request to required index URL like `http://localhost:9200/uint`).
- Check if shards are not in failed state (restart of Elasticsearch should help).
- Check the configuration of Elasticsearch. Configuration should allow access from

the Zabbix frontend host and the Zabbix server host.

- Check Elasticsearch logs.

If you are still experiencing problems with your installation then please create a bug report with all the information from this list (mapping, error logs, configuration, version, etc.)

3 后端配置

1 Zabbix server

默认值仅代表守护进程的默认值，而不是已加载的配置文件中的值。

Zabbix服务端支持的参数如下：

参数	必填	范围	默认值
AlertScriptsPath	否		/usr/local/share/zabbix/alertscripts
AllowRoot	否		0
CacheSize	no	128K-8G	8M
CacheUpdateFrequency	no	1-3600	60
DBHost	no		localhost
DBName	yes		
DBPassword	no		
DBPort	no	1024-65535	3306
DBSchema	no		
DBSocket	no		/tmp/mysql.sock
DBUser	no		
DebugLevel	no	0-5	3
ExternalScripts	no		/usr/local/share/zabbix/externalscripts
Fping6Location	no		/usr/sbin/fping6
FpingLocation	no		/usr/sbin/fping

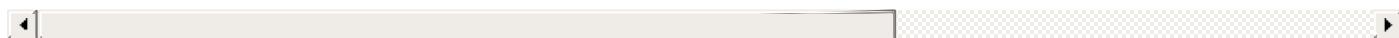
HistoryCacheSize	no	128K-2G	16M
HistoryIndexCacheSize	no	128K-2G	4M
HousekeepingFrequency	no	0-24	1
Include	no		
JavaGateway	no		
JavaGatewayPort	no	1024-32767	10052
ListenIP	no		0.0.0.0
ListenPort	no	1024-32767	10051
LoadModule	no		
LoadModulePath	no		
LogFile	yes, if LogType is set to file,		

	otherwiseno		
LogFileSize	no	0-1024	1
LogType	no		file
LogSlowQueries	no	0- 3600000	0
MaxHousekeeperDelete	no	0- 1000000	5000
PidFile	no		/tmp/zabbix_server.pid
ProxyConfigFrequency	no	1- 604800	3600
ProxyDataFrequency	no	1-3600	1
SenderFrequency	no	5-3600	30
SNMPTrapperFile	no		/tmp/zabbix_traps.tmp
SocketDir	no		/tmp
SourceIP	no		
SSHKeyLocation	no		
SSLCertLocation	no		
SSLKeyLocation	no		
SSLCALocation	no		

StartDBSyncers	no	1-100	4
StartDiscoverers	no	0-250	1
StartEscalators	no	1-100	1
StartHTTPPollers	no	0-1000	1
StartIPMIPollers	no	0-1000	0
StartJavaPollers	no	0-1000	0
StartPingers	no	0-1000	1
StartPollersUnreachable	no	0-1000	1
StartPollers	no	0-1000	5
StartProxyPollers	no	0-250	1
StartSNMPTrapper	no	0-1	0
StartTimers	no	1-1000	1
StartTrappers	no	0-1000	5
StartVMwareCollectors	no	0-250	0
Timeout	no	1-30	3
TLSCAFfile	no		
TLSCertFile	no		

TLSCRLFile	no		
TLSKeyFile	no		
TmpDir	no		/tmp
TrapperTimeout	no	1-300	300
TrendCacheSize	no	128K-2G	4M
UnavailableDelay	no	1-3600	60
UnreachableDelay	no	1-3600	15
UnreachablePeriod	no	1-3600	45
User	no		zabbix
ValueCacheSize	no	0, 128K-64G	8M
VMwareCacheSize	no	256K-2G	8M
VMwareFrequency	no	10-86400	60
VMwarePerfFrequency	no	10-86400	60
VMwareTimeout	no	1-300	10

Zabbix 支持的配置文件是无BOM的utf-8编码. 注释只能在行的开头以 “#” 开始.



2 Zabbix proxy

默认值仅代表守护进程的默认值，而不是已加载的配置文件中的值。

Zabbix proxy支持的参数如下::

Parameter	Mandatory	Range	Default
AllowRoot	no		0
CacheSize	no	128K-8G	8M
ConfigFrequency	no	1-604800	3600
DBHost	no		localhost
DBName	yes		
DBPassword	no		
DBSchema	no		
DBSocket	no		3306
DBUser			
DataSenderFrequency	no	1-3600	1
DebugLevel	no	0-5	3
ExternalScripts	no		/usr/local/share/zabbix/externalscripts
Fping6Location	no		/usr/sbin/fping6
FpingLocation	no		/usr/sbin/fping

HeartbeatFrequency	no	0-3600	60
HistoryCacheSize	no	128K-2G	16M
HistoryIndexCacheSize	no	128K-2G	4M
Hostname	no		Set by HostnameItem
HostnameItem	no		system.hostname
HousekeepingFrequency	no	0-24	1
Include	no		
JavaGateway	no		
JavaGatewayPort	no	1024-32767	10052
ListenIP	no		0.0.0.0

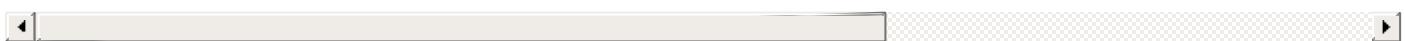
ListenPort	no	1024-32767	10051
LoadModule	no		
LoadModulePath	no		
LogFile	yes, if LogType is set to <i>file</i> , otherwise no		
LogFileSize	no	0-1024	1
LogType	no		file
LogSlowQueries	no	0-3600000	0
PidFile	no		/tmp/zabbixproxy.pid
ProxyLocalBuffer	no	0-720	0
ProxyMode	no	0-1	0
ProxyOfflineBuffer	no	1-720	1
ServerPort	no	1024-32767	10051
Server	yes		
SNMPTrapperFile	no		/tmp/zabbix_traps.tmp
SocketDir	no		/tmp

SourceIP	no		
SSHKeyLocation	no		
SSLCertLocation	no		
SSLKeyLocation	no		
SSLCALocation	no		
StartDBSyncers	no	1-100	4
StartDiscoverers	no	0-250	1
StartHTTPPollers	no	0-1000	1
StartIPMIPollers	no	0-1000	0
StartJavaPollers	no	0-1000	0
StartPingers	no	0-1000	1
StartPollersUnreachable	no	0-1000	1
StartPollers	no	0-1000	5
StartSNMPTrapper	no	0-1	0
StartTrappers	no	0-1000	5
StartVMwareCollectors	no	0-250	0
Timeout	no	1-30	3
TLSAccept	yes, if TLS certificate or PSK parameters are defined (even for unencrypted)		

	connection), otherwise no		
TLSCAFile	no		
TLSCertFile	no		
TLSConnect	yes, if TLS certificate or PSK parameters are defined (even for <i>unencrypted</i> connection), otherwise no		
TLSCRLFile	no		
TLSKeyFile	no		
TLSPSKFile	no		
TLSPSKIdentity	no		
TLSServerCertIssuer	no		
TLSServerCertSubject	no		
TmpDir	no		/tmp
TrapperTimeout	no	1-300	300
User	no		zabbix
UnavailableDelay	no	1-3600	60
UnreachableDelay	no	1-3600	15
UnreachablePeriod	no	1-3600	45
VMwareCacheSize	no	256K-2G	8M
VMwareFrequency	no	10 -	60

VMwarePerfFrequency	no	10 - 86400	60
VMwareTimeout	no	1 - 300	10

Zabbix 支持的配置文件是无 [BOM](#)的utf-8编码。 注释只能以 “#” 开始..



3 Zabbix agent (UNIX)

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

Zabbix agent 配置文件(`zabbix_agentd.conf`)支持的参数：

参数	必填	范围	默认	
Alias	no			为监控项的多个别名：item key。如： <code>\1. 检ID.Alias=9]+),,,,`</code> 义参数得到载。 <code>Alias=</code> 用 <code>cpu.ut:cpu.util[</code> 一个发现项。 <code>vfs.fs.di vfs.fs.di</code>
AllowRoot	no		0	允许 agent 会切换为'Z
BufferSend	no	1-3600	5	缓存区保存数
BufferSize	no	2-65535	100	数据缓存区 proxy。
DebugLevel	no	0-5	3	指定调试等级调试 (产生
EnableRemoteCommands	no		0	是否允许se
HostMetadata	no	0-255 characters		可选参数用通过 HostM agent 会给我
HostMetadataItem	no			可选参数定定时，使用该参数 <code>_EnableR</code> 据。在一个目息。监控项过数。
Hostname	no		Set by HostnameItem	唯一的，区的符号：字
HostnameItem	no		system.hostname	可选参数，置时，该参数 <code>EnableRem</code>
Include	no			可以在配置文件中包含相关特例。 <code>/absolute/</code>
ListenIP	no		0.0.0.0	监听Ip地址：
ListenPort	no	1024-32767	10050	监听端口。
LoadModule	no			agent端启 <code><module.s</code>

				参数。
LoadModulePath	no			agent 模块
LogFile	yes, if LogType is set to file, otherwise no			日志文件名和
LogFileSize	no	0-1024	1	日志文件大小, 文件回
LogType	no		file	日志输出类型 console
LogRemoteCommands	no		0	允许执行远
MaxLinesPerSecond	no	1-1000	20	每秒向serv 中的 'max lines that log items'
PidFile	no		/tmp/zabbixagentd.pid	PID 文件名
RefreshActiveChecks	no	60-3600	120	主动检查频
Server	no			IP地址列表 如果支持IPV6 的, ':':/6 意, "IPv4 Server=12 2.2 以后支
ServerActive	no			server 或 端, 此时用 主机的端口, 选的。如果未
SourceIP	no			对外连接的源
StartAgents	no	0-100	3	被动检查的z 任何TCP 端
Timeout	no	1-30	3	超时连接时间
TLSAccept	yes, if TLS certificate or PSK parameters are defined (even for unencrypted connection), otherwise no			接受什么类 unencrypt 接 cert -
TLSCAFile	no			包含用于对称 信。Zabbix
TLSCertFile	no			包含证书 (i 3.0.0后支持
TLSConnect	yes, if TLS certificate or PSK parameters are defined (even for unencrypted connection), otherwise no			agent 连接 - 连接不加 方式为 TLS

TLSCRLFile	no			包含证书吊销列表文件。支持该参数。
TLSKeyFile	no			包含私钥的文件。
TLSPSKFile	no			包含agent密钥文件。从3.0.0后支持。
TLSPSKIdentity	no			预共享密钥身份。
TLSServerCertIssuer	no			允许的服务颁发者。
TLSServerCertSubject	no			允许的服务主题。
UnsafeUserParameters	no	0, 1	0	允许用户自定义参数。支持" "、*、?、[、]
User	no		zabbix	降低权限为普通用户。Zabbix 2.0.0 及以上版本。
UserParameter	no			用户自定义参数。通过 command>\UserParam

在Zabbix agent 2.0.0 关于主动和被动的配置参数已经改变。更多详细信息请查看本页底部的 "[参见](#)" 部分。

Zabbix 的配置文件都是无 [BOM](#)的utf8编码的。注释只能在每行以 “#” 开头。.

参见

- [Zabbix2.0.0版本后，Zabbix agent 主动和被动配置的区别](#)



4 Zabbix agent (Windows)

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

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

Parameter	Mandatory	Range	Default	
Alias	no			设置Item key的别名参数。而且也允许别名可以用于HostMetadata中。例如：1. 检索服务File(_Total)\%g自定义参数得到CPU载.Alias=cpu.load用cpu.load key cpu.load[percpu发现项.Alias=vfs即可启动不同参数如规则。
BufferSend	no	1-3600	5	缓存区保存数据的最大值。
BufferSize	no	2-65535	100	数据缓存区大小。如proxy。
DebugLevel	no	0-5	3	指定调试等级：0 - 调试（产生大量信息）。
EnableRemoteCommands	no		0	是否允许server远程命令执行。
HostMetadata	no	0-255 characters		可选参数用来定义主通过 HostMetadata agent 会给出一个字符串。
HostMetadataItem	no			可选参数定义 _ZabbixMetadata_ 义时，使用该参数。, EnableRemoteCommands。在一个自动注册项。监控项返回值必须。
Hostname	no		Set by HostnameItem	唯一的，区分大小写。符号：字母数字，'.'
HostnameItem	no		system.hostname	可选参数， 定义一个该参数设置生效。不与EnableRemoteCommands。参考 更多详细描述 。
Include	no			可以在配置文件中指中包含相关文件，才绝对路径。/absolute/path/to/制条件特例。
ListenIP	no		0.0.0.0	监听IP地址列表，多个。
ListenPort	no	1024-32767	10050	监听端口。
LogFile	yes, if LogType is set to file, otherwise no			日志文件名称。

LogFileSize	no	0-1024	1	日志文件大小，单位小，文件回滚失败，
LogType	no		file	日志输出类型: file 件日志 , console
LogRemoteCommands	no		0	允许执行远程命令记
MaxLinesPerSecond	no	1-1000	20	每秒向server发送数据 主动检查。该参数值会 覆盖。
PerfCounter	no			语法: <parameter> <parameter_name> <period> (单位秒 个新的参数 “inter interrupts,”^Pr 引住. 创建监控项的 的样本，每秒取一次 能计数器列表。
RefreshActiveChecks	no	60-3600	120	主动检查频率， 单位
Server	no			IP地址列表，以逗号 如果支持IPv6，则 的， ':/:0' 允许意 意，“IPv4兼容 IP Server=127.0.0. 2.2 以后支持空格。
ServerActive	no	()		server 或 proxy 端，此时用逗号分割 主机的端口， IPv6选的。如果未指定该
SourceIP	no			对外连接的源IP地址
StartAgents	no	0-63 ()	3	被动检查的zabbix 何TCP 端口。1.8.1
Timeout	no	1-30	3	超时连接时间， 单位
TLSAccept	yes, if TLS certificate or PSK parameters are defined (even for _unencrypted connection), otherwise no			接受什么类型的连接 <i>unencrypted</i> - 指 接 <i>cert</i> - 接受TLS
TLSCAFile	no			包含用于对等证书验 信。Zabbix 3.0.0
TLSCertFile	no			包含证书 (证书链) 3.0.0后支持该参数
TLSConnect	yes, if TLS certificate or PSK parameters are defined (even for <i>unencrypted</i> connection), otherwise no			agent 连接server - 连接不加密 (默认 方式为 TLS 和 证书)
				包含证书吊销的文件

				支持该参数。
TLSKeyFile	no			包含私钥的文件的完
TLSPSKFile	no			包含agent预共享ke 3.0.0后支持该参数
TLPSKIdentity	no			预共享密钥身份字符
TLSServerCertIssuer	no			允许的服务器证书发
TLSServerCertSubject	no			允许的服务器证书主
UnsafeUserParameters	no	0-1	0	允许用户自定义参数 } ~ \$! & ; ()
UserParameter				用户自定义监控参数 意 shell命令一定 UserParameter=s

(*) ServerActive 中的服务器数量加上StartAgents 中指定的数量不能大于64.

在Zabbix agent 2.0.0 关于主动和被动的配置参数已经改变。更多详细信息请查看本页底部的 "[参见](#)" 部分。

Zabbix 的配置文件都是无BOM的utf8编码的。注释只能在每行以 “#” 开头。

参见

- [Zabbix2.0.0版本后，Zabbix agent 主动和被动配置的区别。](#)



5 Zabbix Java 网关

如果你使用 `startup.sh` 和 `shutdown.sh` 脚本启动和停止 Zabbix Java 网关，那么就可以在 `settings.sh` 文件中指定必要的配置参数。`startup` 和 `shutdown` 脚本以配置文件为源，并且转换 shell 变量（第一列）为 Java 参数（第二列）。

如果你允许 `java` 直接起动 Zabbix Java 网关，可以通过命令行指定 Java 通信参数。

变量	参数	必选	范围	默认值	描述
LISTEN_IP	<code>zabbix.listenIP</code>	no		0.0.0.0	监听 IP。
LISTEN_PORT	<code>zabbix.listenPort</code>	no	1024-32767	10052	监听端口。
PID_FILE	<code>zabbix.pidFile</code>	no		/tmp/zabbix_java.pid	PID 文件名称。如果省略，Zabbix Java 网关会作为控制台程序启动。
START_POLLERS	<code>zabbix.startPollers</code>	no	1-1000	5	启动的工作进程数量。
TIMEOUT	<code>zabbix.timeout</code>	no	1-30	3	等待网络操作的时长。该参数从 Zabbix 2.0.15, 2.2.10 和 2.4.5 版本以后开始支持。

端口 10052 没有 IANA 注册。

6 "Include"参数的特别说明

如果 `Include` 参数用来包含一个文件， 该文件必须可读。

如果 `Include` 参数用来包含一个目录：

- 该目录下所有文件必须可读。
- 不应该考虑包容性的特定顺序 (e.g. 文件不按字母顺序包含). 因此, 不要在几个 `Include` 文件中定义一个参数(e.g. 以特定覆盖一般设置).
- 该目录下的所有文件都包含在配置文件中。
- 注意一些文本编辑器会自动创建文件备份。如, 如果编辑 `include/my_specific.conf` 会产生一个副本 `include/my_specific.conf.BAK` , 两个文件都会包含在内。从“Include”目录中移出 `include/my_specific.conf.BAK` 文件 。 在Linux系统中, `Include` 目录的内容可以通过命令 `ls -al` 来检查是否有不必要的文件。

如果使用 `Include` 参数用于使用模式包括文件：

- 与模式匹配的所有文件必须是可读的。
- 不应该考虑包容性的特定顺序 (e.g. 文件不按字母顺序包含). 因此, 不要在几个 `Include` 文件中定义一个参数(e.g. 以特定覆盖一般设置).

4 监控项

1 不同平台支持的监控项

下表列出了不同平台支持的Zabbix agent监控项目：

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

只支持Windows Zabbix agent items 不在该表中.

NetBSD											
	OpenBSD	▼▼									
Mac OS X		▼▼									
Tru64		▼▼									
AIX		▼▼									
HP-UX		▼▼									
Solaris		▼▼									
FreeBSD		▼▼									
Linux 2.6 (and later)		▼▼									
Linux 2.4		▼▼									
Windows		▼▼									
Parameter / system		▼▼									
▼▼	1		2	3	4	5	6	7	8	9	10
agent.hostname	X		X	X	X	X	X	X	X	X	X
agent.ping	X		X	X	X	X	X	X	X	X	X
agent.version	X		X	X	X	X	X	X	X	X	X
kernel.maxfiles	-		X	X	X	-	-	-	?	X	X
kernel.maxproc	-		-	X	X	X	-	-	?	X	X
log[file,<regexp>,<encoding>,<maxlines>,<mode>,<output>]	X		X	X	X	X	X	X	X	X	X
logrt[file_format,<regexp>,<encoding>,<maxlines>,<mode>,<output>]	X		X	X	X	X	X	X	X	X	X
net.dns[<ip>,zone,<type>,<timeout>,<count>]	X		X	X	X	X	X	X	X	X	X
net.dns.record[<ip>,zone,<type>,<timeout>,<count>]	X		X	X	X	X	X	X	X	X	X
net.if.collisions[if]	-		X	X	X	X	-	X	-	X	X

net.if.discovery	X	X	X	X	X	X	X	-	-	X	
net.if.in[if,<mode>]	X	X	X	X	X	X 1	X	-	X	X	
<i>mode ▲</i>	bytes (default)	X	X	X	X	X 2	X	X	-	X	
packets	X	X	X	X	X	X	X	-	X	X	
errors	X	X	X	X	X 2	X	X	-	X	X	
dropped	X	X	X	X	-	X	-	-	X	X	
net.if.out[if,<mode>]	X	X	X	X	X	X 1	X	-	X	X	
<i>mode ▲</i>	bytes (default)	X	X	X	X	X 2	X	X	-	X	
packets	X	X	X	X	X	X	X	-	X	X	
errors	X	X	X	X	X 2	X	X	-	X	X	
dropped	X	X	X	-	-	X	-	-	-	-	
net.if.total[if,<mode>]	X	X	X	X	X	X 1	X	-	X	X	
<i>mode ▲</i>	bytes (default)	X	X	X	X	X 2	X	X	-	X	
packets	X	X	X	X	X	X	X	-	X	X	
errors	X	X	X	X	X 2	X	X	-	X	X	
dropped	X	X	X	-	-	X	-	-	-	-	
net.tcp.listen[port]	X	X	X	X	X	-	-	-	X	-	
net.tcp.port[<ip>,port]	X	X	X	X	X	X	X	X	X	X	
net.tcp.service[service,<ip>,<port>]	X	X	X	X	X	X	X	X	X	X	
net.tcp.service.perf[service,<ip>,<port>]	X	X	X	X	X	X	X	X	X	X	
net.udp.listen[port]	-	X	X	X	X	-	-	-	X	-	
net.udp.service[service,<ip>,<port>]	X	X	X	X	X	X	X	X	X	X	
net.udp.service.perf[service,<ip>,<port>]	X	X	X	X	X	X	X	X	X	X	
	1	2	3	4	5	6	7	8	9	10	
proc.cpu.util[<name>,<user>,<type>,<cmdline>,<mode>,<zone>]	-	X	X	-	X 3	-	-	-	-	-	
<i>type ▲</i>	total (default)	-	X	X	-	X	-	-	-	-	
user	-	X	X	-	X	-	-	-	-	-	
system	-	X	X	-	X	-	-	-	-	-	

<i>mode ▲</i>	<i>avg1 (default)</i>	-	X	X	-	X	-	-	-	-	-
avg5	-	X	X	-	X	-	-	-	-	-	-
avg15	-	X	X	-	X	-	-	-	-	-	-
<i>zone ▲</i>	<i>current (default)</i>	-	-	-	-	X	-	-	-	-	-
all	-	-	-	-	X	-	-	-	-	-	-
proc.mem[<name>, <user>, <mode>, <cmdline><memtype>]	-	X	X	X	X 3	-	X	X	-	X	
<i>mode ▲</i>	<i>sum (default)</i>	-	X	X	X	X	-	X	X	-	
avg	-	X	X	X	X	-	X	X	-	X	
max	-	X	X	X	X	-	X	X	-	X	
min	-	X	X	X	X	-	X	X	-	X	
<i>memtype ▲</i>		-	X	X	X	X	-	X	-	-	
proc.num[<name>, <user>, <state>, <cmdline>]	X	X	X	X	X 3	X	X	X	-	X	
<i>state ▲</i>	<i>all (default)</i>	-	X	X	X	X	X	X	X	-	
sleep	-	X	X	X	X	X	X	X	-	X	
zomb	-	X	X	X	X	X	X	X	-	X	
run	-	X	X	X	X	X	X	X	-	X	
<i>cmdline ▲</i>		-	X	X	X	X	X	X	X	-	
sensor[device, sensor, <mode>]	-	X	X	-	-	-	-	-	-	X	
system.boottime	-	X	X	X	X	-	-	-	-	X	X
system.cpu.discovery	X	X	X	X	X	X	X	X	X	X	X
system.cpu.intr	-	X	X	X	X	-	X	-	-	X	
system.cpu.load[<cpu>, <mode>]	X	X	X	X	X	X	X	X	X	X	X
<i>cpu ▲</i>	<i>all (default)</i>	X	X	X	X	X	X	X	X	X	X
percpu	X	X	X	X	X	X	X	-	X	X	
<i>mode ▲</i>	<i>avg1 (default)</i>	X	X	X	X	X	X	X	X	X	X
avg5	X	X	X	X	X	X	X	X	X	X	X
avg15	X	X	X	X	X	X	X	X	X	X	X
system.cpu.num[<type>]	X	X	X	X	X	X	X	-	X	X	
<i>type ▲</i>	<i>online (default)</i>	X	X	X	X	X	X	X	-	X	
max	-	X	X	X	X	-	-	-	X	-	
system.cpu.switches	-	X	X	X	X	-	X	-	-	X	
system.cpu.util[<cpu>, <type>]	X	X	X	X	X	X	X	X	-	X	

<mode>]	X	X	X	X	X	X	X	X	-	X
type ▲	<i>user (default)</i>	-	X	X	X	X	X	X	X	-
nice	-	X	X	X	-	X	-	X	-	X
idle	-	X	X	X	X	X	X	X	-	X
system	X	X	X	X	X	X	X	X	-	X
iowait	-	-	X	-	X	-	X	-	-	-
interrupt	-	-	X	X	-	-	-	-	-	X
softirq	-	-	X	-	-	-	-	-	-	-
steal	-	-	X	-	-	-	-	-	-	-
guest	-	-	X	-	-	-	-	-	-	-
guestnice	-	-	X	-	-	-	-	-	-	-
_mode ▲	<i>avg1 (default)</i>	X	X	X	X	X	X	X	X	-
avg5	X	X	X	X	X	X	X	-	-	X
avg15	X	X	X	X	X	X	X	-	-	X
	1	2	3	4	5	6	7	8	9	10
system.hostname[<type>]	X	X	X	X	X	X	X	X	X	X
system.hw.chassis[<info>]	-	X	X	-	-	-	-	-	-	-
system.hw.cpu[<cpu>, <info>]	-	X	X	-	-	-	-	-	-	-
system.hw.devices[<type>]	-	X	X	-	-	-	-	-	-	-
system.hw.macaddr[<interface>, <format>]	-	X	X	-	-	-	-	-	-	-
system.loclatime[<type>]	X	X	X	X	X	X	X	X	X	X
type ▲	<i>utc (default)</i>	X	X	X	X	X	X	X	X	X
local	X	X	X	X	X	X	X	X	X	X
system.run[command, <mode>]	X	X	X	X	X	X	X	X	X	X
mode ▲	<i>wait (default)</i>	X	X	X	X	X	X	X	X	X
nowait	X	X	X	X	X	X	X	X	X	X
system.stat[resource, <type>]	-	-	-	-	-	-	X	-	-	-
system.sw.arch	X	X	X	X	X	X	X	X	X	X
system.sw.os[<info>]	-	X	X	-	-	-	-	-	-	-
system.sw.packages[<package>, <manager>, <format>]	-	X	X	-	-	-	-	-	-	-
system.swap.in[<device>, <type>] (specifying a device is only supported under Linux)	-	X	X	-	X	-	-	-	-	X
	<i>count (default)</i>									

<i>ifdevice was not specified)</i>	<i>under all except Linux)</i>	-	X	X	-	X	-	-	-	-	-
sectors	-	X	X	-	-	-	-	-	-	-	-
pages (<i>default under Linux</i>)	-	X	X	-	X	-	-	-	-	-	X
system.swap.out[<device>, <type>] (<i>specifying a device is only supported under Linux</i>)	-	X	X	-	X	-	-	-	-	-	X
<i>type ▲(pages will only work ifdevice was not specified)</i>	<i>count (default under all except Linux)</i>	-	X	X	-	X	-	-	-	-	-
sectors	-	X	X	-	-	-	-	-	-	-	-
pages (<i>default under Linux</i>)	-	X	X	-	X	-	-	-	-	-	X
system.swap.size[<device>, <type>] (<i>specifying a device is only supported under FreeBSD, for other platforms must be empty or "all"</i>)	X	X	X	X	X	-	X	X	-	-	X
<i>type ▲</i>	<i>free (default)</i>	X	X	X	X	X	-	X	X	-	-
total	X	X	X	X	X	-	X	X	-	-	X
used	X	X	X	X	X	-	X	X	-	-	X
pfree	X	X	X	X	X	-	X	X	-	-	X
pused	-	X	X	X	X	-	X	X	-	-	X
system.uname	X	X	X	X	X	X	X	X	X	X	X
system.uptime	X	X	X	X	X	-	X	?	X	X	
system.users.num		-	X	X	X	X	X	X	X	X	X
	1	2	3	4	5	6	7	8	9	10	
vfs.dev.read[<device>, <type>, <mode>]	-	X	X	X	X	-	X	-	-	-	X
<i>type ▲ (defaults are different under various OSes)</i>	<i>sectors</i>	-	X	X	-	-	-	-	-	-	-
operations	-	X	X	X	X	-	X	-	-	-	X
bytes	-	-	-	X	X	-	X	-	-	-	X
sps	-	X	X	-	-	-	-	-	-	-	-
ops	-	X	X	X	-	-	-	-	-	-	-
bps	-	-	-	X	-	-	-	-	-	-	-
<i>mode ▲ (compatible only with type in: sps, ops, bps)</i>	<i>avg1 (default)</i>	-	X	X	X	-	-	-	-	-	-
avg5	-	X	X	X	-	-	-	-	-	-	-
avg15	-	X	X	X	-	-	-	-	-	-	-
vfs.dev.write[<device>, <type>, <mode>]	-	X	X	X	X	-	X	-	-	-	X

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

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

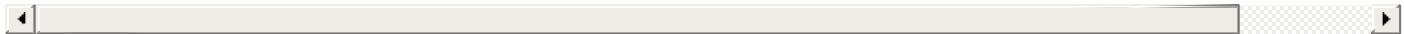
另请参见 [vm.memory.size](#) 参数说明。

脚注

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

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

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



2 参数vm.memory.size

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

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

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

特定系统的注意事项

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

另见

- [关于不同操作系统内存计算的详细信息](#)

3 被动和主动代理检查

概述

本节提供关于Zabbix代理执行的被动和主动检查的详细信息。

Zabbix使用一个基于JSON的通信协议来与Zabbix代理进行通信。

这里有一些Zabbix使用的协议细节中的使用到的定义：

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

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

被动检查

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

Server 请求

```
1. <item key>\n
```

Agent 响应

```
1. <HEADER><DATALEN><DATA>[\0<ERROR>]
```

在上面，方括号中的部分是可选的，只发送到不受支持的项目。

例如，对于支持的监控项：

- Server 打开一个TCP连接
- Server 发送 `agent.ping\n`
- Agent 读取请求并响应
1
- Server 处理数据以获取值，'1' in our case
- TCP连接关闭

对于不支持的监控项：

- Server 打开一个TCP连接

- Server 发送 `vfs.fs.size[/nono]\n`

- Agent 读取请求并响应

```
ZBX_NOTSUPPORTED\0Cannot obtain filesystem information: [2] No such file or
directory
```

- Server 处理数据，更改项目状态为不支持并显示指定的错误消息

- TCP连接关闭

主动检查

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

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

agent然后定期向服务器发送新值。

获取监控项列表

Agent 请求

```
1. <HEADER><DATALEN>{
2.     "request":"active checks",
3.     "host":"<hostname>"
4. }
```

Server 响应

```
1. <HEADER><DATALEN>{
2.     "response":"success",
3.     "data":[
4.         {
5.             "key":"/home/zabbix/logs/zabbix_agentd.log",
6.             "delay":30,
7.             "lastlogsize":0,
8.             "mtime":0
9.         },
10.        {
11.            "key":"agent.version",
12.            "delay":600,
13.            "lastlogsize":0,
14.            "mtime":0
15.        },
16.        {
17.            "key":"vfs.fs.size[/nono]",
18.            "delay":600,
19.            "lastlogsize":0,
20.            "mtime":0
21.        }
22. }
```

```

22.     ]
23. }
```

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

例如：

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

注意，在使用主动检查时，对于可以访问Zabbix服务器trapper端口的配置数据是可得到的。 这是可能的，因为任何一个都可以假装是一个主动agent，并请求项目配置数据；除非你使用 [加密](#) 选项，否则认证不会发生

发送收集的数据

Agent 发送

```

1. <HEADER><DATALEN>{
2.   "request": "agent data",
3.   "data": [
4.     {
5.       "host": "<hostname>",
6.       "key": "agent.version",
7.       "value": "2.4.0",
8.       "clock": 1400675595,
9.       "ns": 76808644
10.    },
11.    {
12.      "host": "<hostname>",
13.      "key": "log[/home/zabbix/logs/zabbix_agentd.log]",
14.      "lastlogsize": 112,
15.      "value": "19845:20140621:141708.521 Starting Zabbix Agent [<hostname>]. Zabbix 2.4.0 (revision
16.      50000).",
17.      "clock": 1400675595,
18.      "ns": 77053975
19.    },
20.    {
21.      "host": "<hostname>",
22.      "key": "vfs.fs.size[/nono]",
23.      "state": 1,
24.      "value": "Cannot obtain filesystem information: [2] No such file or directory",
25.      "clock": 1400675595,
26.      "ns": 78154128
27.  }
28.}
```

```

26.      }
27.      ],
28.      "clock": 1400675595,
29.      "ns": 78211329
30.  }

```

Server 响应

```

1.  <HEADER><DATALEN>{
2.      "response":"success",
3.      "info":"processed: 3; failed: 0; total: 3; seconds spent: 0.003534"
4.  }

```

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

例如:

- Agent 打开一个TCP连接
- Agent 发送一个值列表
- Server 处理数据并将状态返回
- TCP 连接关闭

注意, 上面例子中怎么不支持 `vfs.fs.size[/nono]` 的状态由 “state” 值为 1 和 “value” 中的错误消息表示。

在服务器端, 错误消息将被处理到2048个符号。

Older XML protocol

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

另请参阅

- [关于Zabbix agent协议的更多细节](#)

4 返回值的编码

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

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

在这种情况下，对不同数据库后台的行为描述如下。

MySQL

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

PostgreSQL

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

5 大文件支持

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

6 不可达/不可用 主机设置

概述

当agent检查(Zabbix, SNMP, IPMI, JMX)失败并且主机变得不可达时，一些配置 [参数](#) 定义了 Zabbix server 作何反应。

不可达主机

Zabbix, SNMP, IPMI 或 JMX agents检查(网络错误, 超时)失败后即视主机不可达。注意, Zabbix agent 主动检查不影响主机可用性。

From that moment **UnreachableDelay** 定义了主机再次检查的频率 is rechecked using one of the items (包括 LLD 规则) in this unreachability situation and such rechecks will be performed already by unreachable pollers. 默认情况下, 两次检查时间间隔为15秒。

在Zabbix server 日志中，不可达是通过类似下面的消息表示的：

- ```
1. Zabbix agent item "system.cpu.load[percpu,avg1]" on host "New host" failed: first network error, wait for 15 seconds
2. Zabbix agent item "system.cpu.load[percpu,avg15]" on host "New host" failed: another network error, wait for 15 seconds
```

注意，失败的监控项和监控项类型 (Zabbix agent) 列出来了。

在主机不可达期间, *Timeout* 参数也会影响主机再次被检查的时间。如果*Timeout* 是 20 秒, 但是 *UnreachableDelay* 是 30 秒, 下一次检查在 50 秒后。

**UnreachablePeriod**参数定义了不可达的总时长。 *UnreachablePeriod* 应该比 *UnreachableDelay*大几倍, 这样在主机变为不可用之前, 主机会被检查不止一次。

如果不可达主机再次出现, 监控自动恢复正常:

恢复 Zabbix agent 对主机 “New host”的检查：连接恢复

### 不可用主机

主机不可达期结束后主机没有再次出现, 视主机为不可用。

在server 日志中, 不可用是通过类似下面的消息来表示的：

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

在前端 主机可用性图标由绿色(或灰色)变为红色(注意, 在鼠标经过时会提示错误描述)：



UnavailableDelay 参数定义了在主机不可用期间，主机被检查的频率。

默认为 60 秒（所以此时从上面的日志信息来看，“temporarily disabling”意味着禁用检查一分钟）。

当主机连接恢复时，监控也会自动恢复正常：

1. 启用Zabbix agent 对 "New host" 主机的检查：主机变为可达

7 传感器

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

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

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

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

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

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

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

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

Linux 2.4:

(传感器读数从 `/proc/sys/dev/sensor` 目录获得)

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

例子：`sensor[w83781d-i2c-0-2d,temp1]`

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

Linux 2.6+:

(传感器读数从 /sys / class / hwmon 目录获得)

- **device** - 设备名称(非正则表达式)。设备名称可以是设备的实际名称(e.g 0000:00:18.3)或使用传感器程序获取得的名称(例如:k8temp-pci-00c3)，这由用户决定使用哪个名称；
- **sensor** - 传感器名称(非正则表达式)；
- **mode** - 可能的值:avg, max, min (如果忽略了这个参数，设备和传感器将逐字处理)。

例如：

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

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

获取传感器的名字

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

```
1. $ sensors
2. in0:          +2.24 V  (min = +0.00 V, max = +3.32 V)
3. Vcore:        +1.15 V  (min = +0.00 V, max = +2.99 V)
4. +3.3V:        +3.30 V  (min = +2.97 V, max = +3.63 V)
5. +12V:         +13.00 V  (min = +0.00 V, max = +15.94 V)
6. M/B Temp:    +30.0°C  (low = -127.0°C, high = +127.0°C)
```

在这些情况下，只有一个标签可以直接使用：

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

尝试使用其他标签（像 `Vcore` 或 `+12V`）是不会起作用的。

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

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

```
1. $ sensors -u
2. ...
3. Vcore:
4.   in1_input: 1.15
5.   in1_min: 0.00
6.   in1_max: 2.99
7.   in1_alarm: 0.00
8. ...
```

```
9. +12V:  
10. in4_input: 13.00  
11. in4_min: 0.00  
12. in4_max: 15.94  
13. in4_alarm: 0.00  
14. ...
```

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

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

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

¹⁾

根据 声明 这些是芯片引脚上的电压, 一般来说可能需要缩放。

8 proc.mem 监控项中memtype参数类型的注意事项

概述

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

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

AIX

请参见表中AIX上的“memtype”参数所支持的值。

支持的参数值	描述	Source in procentry64 structure	Tries to be compatible with
<code>vsize</code> 1)	虚拟内存大小	<code>pi_size</code>	
<code>pmem</code>	实际内存的百分比	<code>pi_prm</code>	<code>ps -o pmem</code>
<code>rss</code>	驻留集大小	<code>pi_trss + pi_drss</code>	<code>ps -o rssize</code>
<code>size</code>	进程大小(代码+数据)	<code>pi_dvm</code>	“ <code>ps gvw</code> ” SIZE column
<code>dsize</code>	数据大小	<code>pi_dsize</code>	
<code>tsize</code>	文本(代码)的大小	<code>pi_tsize</code>	“ <code>ps gvw</code> ” TSIZ column
<code>sdsiz</code>	来自共享库的数据大小	<code>pi_sdsiz</code>	
<code>drss</code>	数据驻留集大小	<code>pi_drss</code>	
<code>trss</code>	文本驻留集大小	<code>pi_trss</code>	

FreeBSD

请参见表中FreeBSD上的“memtype”参数支持的值。

Supported value	Description	Source in kinfo_proc structure	Tries to be compatible with
<code>vsize</code>	虚拟内存大小	<code>kp_eproc.e_vm.vm_map.size</code> or <code>ki_size</code>	<code>ps -o vsz</code>
<code>pmem</code>	实际内存的百分比	calculated from <code>rss</code>	<code>ps -o pmem</code>
<code>rss</code>	驻留集大小	<code>kp_eproc.e_vm.vm_rssize</code> or <code>ki_rssize</code>	<code>ps -o rss</code>
<code>size</code> 2)	进程(代码+数据+堆栈)大小	<code>tsize + dsize + ssize</code>	
<code>tsize</code>	文本(代码)的大小	<code>kp_eproc.e_vm.vm_tsize</code> or <code>ki_tsize</code>	<code>ps -o tsiz</code>
<code>dsize</code>	数据大小	<code>kp_eproc.e_vm.vm_dsize</code> or <code>ki_dsize</code>	<code>ps -o dsiz</code>

ssize	堆栈大小	kp_eproc.e_vm.vm_ssiz or ki_ssiz	ps -o ssiz
-------	------	-------------------------------------	------------

Linux

请参见表中Linux上的“memtype”参数支持的值。

Supported value	Description	Source in /proc/<pid>/status file
vsize 3)	虚拟内存大小	VmSize
pmem	实际内存的百分比	(VmRSS/total_memory) * 100
rss	驻留集大小	VmRSS
data	数据段的大小	VmData
exe	代码段的大小	VmExe
hwm	驻留集峰值大小	VmHWM
lck	锁定内存大小	VmLck
lib	共享库的大小	VmLib
peak	虚拟内存峰值大小	VmPeak
pin	固定的页面大小	VmPin
pte	页表条目的大小	VmPTE
size	进程码+数据+栈段大小	VmExe + VmData + VmStk
stk	堆栈段大小	VmStk
swap	使用的交换空间大小	VmSwap

Linux上注意事项：

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

Solaris

请参见表中的Solaris上的“memtype”参数所支持的值。

支持的参数值	描述	Source in psinfo structure	兼容
vsize 4)	Size of process image	pr_size	ps -o vsz
pmem	实际内存的百分比	pr_pctmem	ps -o pmem
rss	驻留集大小 可能会被低估 - 参看 “man ps”中rss描述.	pr_rssize	ps -o rss

1) , 2) , 3) , 4)

- default value

9 在proc.mem和proc.num项目中选择进程的注意事项

Processes modifying their cmdline

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

让我们来看一个Linux的例子，假设我们想要监视许多Zabbix代理进程。

`ps` 命令显示的进程如下

```
1. $ ps -fu zabbix
2.   UID      PID  PPID  C STIME TTY          TIME CMD
3. ...
4. zabbix    6318      1  0 12:01 ?        00:00:00 sbin/zabbix_agentd -c /home/zabbix/ZBXNEXT-
1078/zabbix_agentd.conf
5. zabbix    6319  6318  0 12:01 ?        00:00:01 sbin/zabbix_agentd: collector [idle 1 sec]
6. zabbix    6320  6318  0 12:01 ?        00:00:00 sbin/zabbix_agentd: listener #1 [waiting for connection]
7. zabbix    6321  6318  0 12:01 ?        00:00:00 sbin/zabbix_agentd: listener #2 [waiting for connection]
8. zabbix    6322  6318  0 12:01 ?        00:00:00 sbin/zabbix_agentd: listener #3 [waiting for connection]
9. zabbix    6323  6318  0 12:01 ?        00:00:00 sbin/zabbix_agentd: active checks #1 [idle 1 sec]
10. ...
...
```

通过名称和用户选择进程来完成任务：

```
1. $ zabbix_get -s localhost -k 'proc.num[zabbix_agentd,zabbix]'
2. 6
```

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

`ps` 现在显示为

```
1. $ ps -fu zabbix
2.   UID      PID  PPID  C STIME TTY          TIME CMD
3. ...
4. zabbix    6715      1  0 12:53 ?        00:00:00 sbin/zabbix_agentd_30 -c /home/zabbix/ZBXNEXT-
1078/zabbix_agentd.conf
5. zabbix    6716  6715  0 12:53 ?        00:00:00 sbin/zabbix_agentd_30: collector [idle 1 sec]
6. zabbix    6717  6715  0 12:53 ?        00:00:00 sbin/zabbix_agentd_30: listener #1 [waiting for connection]
7. zabbix    6718  6715  0 12:53 ?        00:00:00 sbin/zabbix_agentd_30: listener #2 [waiting for connection]
8. zabbix    6719  6715  0 12:53 ?        00:00:00 sbin/zabbix_agentd_30: listener #3 [waiting for connection]
9. zabbix    6720  6715  0 12:53 ?        00:00:00 sbin/zabbix_agentd_30: active checks #1 [idle 1 sec]
10. ...
...
```

现在根据名称和用户选择进程会产生不正确的结果：

```
1. $ zabbix_get -s localhost -k 'proc.num[zabbix_agentd_30,zabbix]'
2. 1
```

为什么将可执行文件重命名为更长的名称会导致完全不同的结果？

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

```
1. $ grep Name /proc/{6715,6716,6717,6718,6719,6720}/status
2. /proc/6715/status:Name: zabbix_agentd_3
3. /proc/6716/status:Name: zabbix_agentd_3
4. /proc/6717/status:Name: zabbix_agentd_3
5. /proc/6718/status:Name: zabbix_agentd_3
6. /proc/6719/status:Name: zabbix_agentd_3
7. /proc/6720/status:Name: zabbix_agentd_3
```

`status` 文件中的进程名会被截断为15个字符。

`ps` 命令会产生相似的结果：

```
1. $ ps -u zabbix
2.   PID TTY      TIME CMD
3. ...
4. 6715 ?        00:00:00 zabbix_agentd_3
5. 6716 ?        00:00:01 zabbix_agentd_3
6. 6717 ?        00:00:00 zabbix_agentd_3
7. 6718 ?        00:00:00 zabbix_agentd_3
8. 6719 ?        00:00:00 zabbix_agentd_3
9. 6720 ?        00:00:00 zabbix_agentd_3
10. ...
```

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

agent如何看待“cmdline”文件，可以通过运行一个命令来说明

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

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

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

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

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

`cmdline` 参数使用恰当的正则表达式会达到一个正确的结果：

```
1. $ zabbix_get -s localhost -k 'proc.num[,zabbix,,zabbix_agentd_30[ :]]'
2. 6
```

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

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

Linux 内核线程

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

让我们以内核线程为例：

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

可以用进程“名称”参数选择：

```
1. $ zabbix_get -s localhost -k 'proc.num[kthreadd,root]'
2. 1
```

但使用进程 `cmdline` 参数就不起作用：

```
1. $ zabbix_get -s localhost -k 'proc.num[,root,,kthreadd]'
2. 0
```

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

`'proc.mem[]'` 和 `'proc.num[]'` 监控项中的线程计数

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

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

```
1. $ zabbix_get -s localhost -k 'proc.num[kthreadd]'
2. 1
```

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

但是如果用户线程和内核线程名字相同会发生什么呢？可能会是这样：

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

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

```
1. $ zabbix_get -s localhost -k 'proc.mem[kthreadd]'  
2. 4157440
```

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

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

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

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

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

监控项 net.tcp.service 参数

ftp

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

http

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

https

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

imap

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

ldap

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

nntp

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

pop

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

smtp

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

ssh

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

tcp

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

telnet

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

Item net.udp.service parameters

ntp

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

5 触发器

1 支持的触发器函数

[trigger expressions](#) 可用的函数列出如下。

函数	描述	参数	
abschange	最近获取值与之前获取值差的绝对值。		支持值的类型 text, log =abschange 返回值:0 -
avg (sec #num, <timeshift>)	指定评估期内一个项目的平均值。	sec or #num - 评估期以秒值或最新值个数(跟在#号后)表示 time_shift (可选) - 时间偏移	支持值的类型 avg(#5) → avg(1h) → avg(1h,1d) 值。从Zabbix 持 time_shif 值和指定时 time_shif 了。
band (sec #num,mask, <time_shift>)	项目值和掩码的按位与值。	sec (可省略) or #num - 最新的第N个值 mask (必须有) - 64位无符号整数(0 - 18446744073709551615) time_shift (可选) - 参照 avg()	支持值的类型 参数和其它- last()).尽 是所有的参 如, 检查第 例如: → ba band(,12): 置, 但不是同 三位没有被设 Zabbix 2.0
change	最近获取值与之前获取值的差。		支持的值类型 text, log =change)1 返回值:0 -
count (sec #num, <pattern>,<operator>, <time_shift>)			支持值的类型 text, log 个 pattern 以 '/' 分隔: number_to count() 计 和 number_ 结果。如果指 number_to

	指定评估期内值的个数。	个数 (跟在#号后) 表示 pattern (可选) - 所需模式 (整型项目 - 精确匹配; 浮点型项目 - 误差值0.000001 内) operator (可选)支持的 operators : <code>_eq</code> - 相等 <code>ne</code> - 不相等 <code>gt</code> - 大于 <code>ge</code> - 大于等于 <code>lt</code> - 小于 <code>le</code> - 小于等于 <code>like</code> - 只要包含 pattern (区分大小写)就被匹配 <code>band</code> - 按位与 <code>regexp</code> - 给定 pattern 的正则表达式, 区分大小写 <code>iregexp</code> - 给定 pattern 的正则表达式, 不区分大小写 注意: <code>eq</code> (默认), <code>ne</code> , <code>gt</code> , <code>ge</code> , <code>lt</code> , <code>le</code> , <code>band</code> , <code>regexp</code> , <code>iregexp</code> 支持整数型项目 <code>eq</code> (默认), <code>ne</code> , <code>gt</code> , <code>ge</code> , <code>lt</code> , <code>le</code> , <code>regexp</code> , <code>iregexp</code> 支持浮点型项目 <code>like</code> (默认), <code>eq</code> , <code>ne</code> , <code>regexp</code> , <code>iregexp</code> 支持 <code>string</code> , <code>text</code> and <code>log</code> 型项目 time_shift (可选) - 参照 <code>avg()</code>	如果掩码值等于 <code>number_to</code> 定掩码值。当三个参数时, 是一个普通的正则表达式。从全局表达式匹配, 浮点型应该注意才库中) 和二进制表示中的四位小数。去10分钟值的过去10分钟 <code>count(10m)</code> 大于12的个数 → 最新10个 <code>count(10m)</code> 前十分钟值的 <code>count(10m)</code> 钟值最低三个的个数。→ 前的前十分钟开始支持 <code>#num</code> 始支持 <code>time</code> 从Zabbix 2.4.0起从Zabbix 2.4.0起 <code>iregexp</code> 运行
date	当前日期, 以 <code>YYYYMMDD</code> 格式表示。	支持值的类型	
dayofmonth	当前是本月的第几天, 取值范围从1到31。	支持值的类型	支持该函数。
dayofweek	当前是本周的第几天, 取值范围从1到7 (周一 - 1, 周日 - 7)。	支持值的类型	
delta (<code>sec #num, <timeshift></code>)	指定评估期内最大值和最小值的差 (<code>'max()'</code> 减去 <code>'min()'</code>)。	<code>sec</code> or <code>#num</code> - 评估期以秒值或最新值 个数 (跟在#号后) 表示 time_shift (可选) - see <code>avg()</code>	支持值的类型 1.8.2开始支持
diff	比较最近获取值与之前获取值是否相同。	支持值的类型 <code>text</code> , <code>log</code>	值相等

forecast (<code>sec #num, <time_shift>, time, <fit>, <mode></code>)		<p><code>sec</code> or <code>#num</code> - 评估期以秒值或最新值个数 (跟在#号后) 表示 <code>time_shift</code> (可选) - see <code>avg()</code> <code>time</code> - 需要进行估计的指定时间 <code>fit</code> (可选) - 用于匹配历史数据的函数支持的 <code>fits :_linear</code> - 线性函数 <code>polynomialN</code> - n次多项式 ($1 \leq N \leq 6$) <code>exponential</code> - 指数函数 <code>logarithmic</code> - 对数函数 <code>power</code> - 幂函数 注意: 默认是 <code>linear</code>, <code>polynomial1</code> 等同于 <code>linear</code> <code>mode</code> (可选) - demanded output 支持的 <code>modes :value</code> - 值 (默认) <code>max</code> - 最大值 <code>min</code> - 最小值 <code>delta</code> - 最大值-最小值 <code>avg</code> - 平均值 注意: <code>value</code> 估计项目值在此刻 <code>now + time max, min, delta</code> and <code>avg</code> 根据 <code>now</code> 和 <code>now + time</code> 时间段估计的项目值确定结果</p>	支持值的类型 于 9999999 -99999999 置为999999 -99999999 被误用时才不 效的参数), <code>forecast</code> (值估计一小 <code>forecast</code> (的值估计三 <code>forecast</code> (一个小时值估 <code>forecast</code> (根据过去一小 分钟后的项目 <code>forecast</code> (→ 根据过去一 计两小时的最 <code>forecast</code> (个值估计二十 <code>last()</code> 或pr 项目很少更新 次) 从Zabb Zabbix 3.1 的 <code>time</code> 值。 <code>trigger f</code>
fuzzytime (<code>sec</code>)		<p>检查项目时间戳和zabbix服务器时间相差多大。</p>	<p><code>sec</code> - 秒数</p>
iregexp (<code>pattern, <sec #num></code>)		<p>该函数和 <code>regexp()</code> 类似, 只是不区分大小写。</p>	<p>see <code>regexp()</code></p>
last (<code>sec #num, <timeshift></code>)		<p>最近的值。</p>	<p><code>sec</code> (可省略) or <code>#num</code> - 最新的第N个值 <code>time_shift</code> (可选) - see <code>avg()</code></p>

logeventid (pattern)	检查最近日志记录的EventID是否匹配正则表达式。	pattern - 使用正则表达式表示需要匹配的模式, POSIX extended 类型。	支持值的类型 - 匹配从Zabbix数。
logseverity	最近日志记录的日志等级。		支持值的类型 N - 对应的event log Warning, Audit, 8 Critical, Windows error 获取日志等级。
logsource (pattern)	检查最近的日志记录是否匹配参数的日志来源。	pattern - string类型	支持值的类型 - 匹配通常日志, 例如, logs Server").
max (sec #num, <time_shift>)	指定评估期内一个项目的最大值。	sec or #num - 评估期以秒值或最新值个数(跟在#号后)表示 time_shift (可选) - see avg()	支持值的类型 1.8.2开始支持
min (sec #num, <time_shift>)	指定评估期内一个项目的最小值。	sec or #num - 评估期以秒值或最新值个数(跟在#号后)表示 time_shift (可选) - see avg()	支持值的类型 1.8.2开始支持
nodata (sec)	检查是否没有接收到数据。	sec - 评估期以秒值表示。时间不应该少于30秒, 因为timer处理器每30秒计算一次该函数。nodata(0) 不被允许。	支持值的类型 评估期没有接收在第一个参数该函数会报错server被重命名没有数据以及示在触发器中
now	距离Epoch (00:00:00 UTC, January 1, 1970)时间的秒数。		支持值的类型
percentile (sec #num, <timeshift>, percentage)	P-th 一段时间的百分值, P (percentage)	sec or #num - 评估期以秒值或最新值个数(跟在#号后)表示 time_shift (可选) - see avg() percentage -	支持值的类型

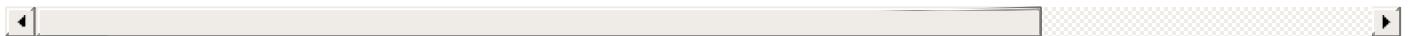
	(percentage) 做为第三个参数。	(可选) - see avg() percentage - 0 and 100 (包括)之间的一个浮点数, 小数点后最多四位	3.0.0开始支持
prev	前一个值。		支持值的类型: text, log
regexp (pattern, <sec #num>)	检查最近的值是否匹配正则表达式。	pattern - 正则表达式, POSIX extended 样式。 sec or #num (可选) - 评估期以秒值或最新值个数(跟在#号后)表示。这种情况下, 可处理一个值以上。	支持值的类型: 值:1 - 找到写。
str (pattern, <sec #num>)	从最新值中查找 一个字符串。	pattern - string型 sec or #num (可选) - 评估期以秒值或最新值个数(跟在#号后)表示。这种情况下, 可处理一个值以上。	支持值的类型: 值:1 - 找到写。
strlen (sec #num, <time_shift>)	最新值的字符长 度(而不是字节数)。	sec (可省略) or #num - 最新的第N 个值 time_shift (可选) - see avg()	支持值的类型: 处的 #num 不用。例如: strlen(#1 strlen(#3 strlen(,1 从Zabbix 1
sum (sec #num, <time_shift>)	指定评估期内项 目值的和。	sec or #num - 评估期以秒值或最新值 个数(跟在#号后)表示。 time_shift (可选) - see avg()	支持值的类型: 1.8.2开始支持
time	当前时间, 以 HHMMSS格式表 示。		支持值的类型:
timelleft (sec #num, <time_shift>,threshold, <fit>)		sec or #num - 评估期以秒值或最新值	支持值的类型: 于 9999999 999999999 将返回值设置 有在表达式初 目类型, 无交 回-1。例如: 根据最新的时 间→ time

久时间。	(可选) - see <code>avg()</code> 阈值 <code>fit</code> (可选) - see <code>forecast()</code>	间 ⇒ <code>timeleft()</code> 前的一个小时 间 ⇒ <code>timeleft()</code> 根据过去一小 项目值达到2 3.0.0开始 和 3.2.2开 <code>threshold</code> <code>trigger f</code>
------	--	--

1) 所有函数的返回值都是数值型，而不是字符串。2) 一些函数不能使用非整型参数！3) 字符串参数应该加上双引号，否则可能被错误解析。

函数和支持的项目

从Zabbix 3.2开始，`nodata()`, `date()`, `dayofmonth()`, `dayofweek()`, `now()` and `time()` 函数也归到不支持的项目里。 其它的函数需要被引用的项目处于支持状态。



6 宏

1 Macros supported by location

Overview

The table contains a complete list of macros supported by Zabbix.

Macro	Supported in	Description
{ACTION.ID}	→ Notifications and commands → Internal notifications	<i>Numeric ID of the triggered action._Supported since 2.2.0.</i>
{ACTION.NAME}	→ Notifications and commands → Internal notifications	<i>_Name of the triggered action._Supported since 2.2.0.</i>
{ALERT.MESSAGE}	→ Alert script parameters	'Default message' value from action configuration. <i>Supported since 3.0.0.</i>
{ALERT.SENDTO}	→ Alert script parameters	'Send to' value from user media configuration. <i>Supported since 3.0.0.</i>
{ALERT.SUBJECT}	→ Alert script parameters	'Default subject' value from action configuration. <i>Supported since 3.0.0.</i>
{DATE}	→ Notifications and commands → Internal notifications	<i>_Current date in yyyy.mm.dd. format.</i>
{DISCOVERY.DEVICE.IPADDRESS}	→ Discovery notifications	<i>IP address of the discovered device._Available always, does not depend on host being added.</i>
{DISCOVERY.DEVICE.DNS}	→ Discovery notifications	<i>DNS name of the discovered device._Available always, does not depend on host being added.</i>
{DISCOVERY.DEVICE.STATUS}	→ Discovery notifications	<i>Status of the discovered device: can be either UP or DOWN.</i>
{DISCOVERY.DEVICE.UPTIME}	→ Discovery notifications	<i>Time since the last change of discovery status for a particular device._For example: 1h 29m.For devices with status DOWN, this is the period of their downtime.</i>
		<i>_Name of the discovery rule</i>

{DISCOVERY.RULE.NAME}	→ Discovery notifications	that discovered the presence or absence of the device or service.
{DISCOVERY.SERVICE.NAME}	→ Discovery notifications	<i>Name of the service that was discovered._For example: HTTP.</i>
{DISCOVERY.SERVICE.PORT}	→ Discovery notifications	<i>_Port of the service that was discovered._For example: 80.</i>
{DISCOVERY.SERVICE.STATUS}	→ Discovery notifications	<i>_Status of the discovered service: can be either UP or DOWN.</i>
{DISCOVERY.SERVICE.UPTIME}	→ Discovery notifications	<i>Time since the last change of discovery status for a particular service._For example: 1h 29m.For services with status DOWN, this is the period of their downtime.</i>
{ESC.HISTORY}	→ Trigger-based notifications and commands → Internal notifications	<i>_Escalation history. Log of previously sent messages._Shows previously sent notifications, on which escalation step they were sent and their status (_sent, in progress or failed).</i>
{EVENT.ACK.HISTORY}	→ Trigger-based notifications and commands	<i>Log of acknowledgements on the problem.</i>
{EVENT.ACK.STATUS}	→ Trigger-based notifications and commands	<i>Acknowledgement status of the event (Yes/No).</i>
{EVENT.AGE}	→ Notifications and commands → Internal notifications	<i>Age of the event that triggered an action._Useful in escalated messages.</i>
{EVENT.DATE}	→ Notifications and commands → Internal notifications	<i>Date of the event that triggered an action.</i>
{EVENT.ID}	→ Notifications and commands → Internal notifications	<i>Numeric ID of the event that triggered an action.</i>
{EVENT.RECOVERY.DATE}	→ Trigger-based notifications and commands → Internal notifications	<i>Date of the recovery event._Can be used in recovery messages only.Supported since 2.2.0.</i>
	→ Trigger-	

{EVENT.RECOVERY.ID}	based notifications and commands→ Internal notifications	_Numeric ID of the recovery event._Can be used in recovery messages only. Supported since 2.2.0.
{EVENT.RECOVERY.STATUS}	→ Trigger-based notifications and commands→ Internal notifications	_Verbal value of the recovery event._Can be used in recovery messages only. Supported since 2.2.0.
{EVENT.RECOVERY.TAGS}	→ Trigger-based notifications and commands	_A comma separated list of recovery event tags. Expanded to an empty string if no tags exist. Supported since 3.2.0.
{EVENT.RECOVERY.TIME}	→ Trigger-based notifications and commands→ Internal notifications	<i>Time of the recovery event._Can be used in recovery messages only.</i> Supported since 2.2.0.
{EVENT.RECOVERY.VALUE}	→ Trigger-based notifications and commands→ Internal notifications	_Numeric value of the recovery event._Can be used in recovery messages only. Supported since 2.2.0.
{EVENT.STATUS}	→ Notifications and commands→ Internal notifications	_Verbal value of the event that triggered an action._Supported since 2.2.0.
{EVENT.TAGS}	→ Trigger-based notifications and commands	_A comma separated list of event tags. Expanded to an empty string if no tags exist. Supported since 3.2.0.
{EVENT.TIME}	→ Notifications and commands→ Internal notifications	<i>Time of the event that triggered an action.</i>
{EVENT.VALUE}	→ Notifications and commands→ Internal notifications	<i>Numeric value of the event that triggered an action._Supported since 2.2.0.</i>
	→ Trigger-based notifications and commands→ Internal notifications→ Global scripts (including confirmation text)→ Icon labels in maps ¹ → Item	

{HOST.CONN<1-9>}	<p>key parameters²→ Host interface IP/DNS→ Database monitoring additional parameters⁵→ SSH and Telnet scripts⁵→ Web monitoring⁶→ Low-level discovery rule filter regular expressions⁸→ URL field of dynamic URL screen element⁸→ Trigger names and descriptions→ Trigger URLs¹⁰</p>	<p>_Host IP address or DNS name, depending on host settings³. Supported in trigger names since 2.0.0.</p>
{HOST.DESCRIPTION<1-9>}	<p>→ Trigger-based notifications and commands→ Internal notifications→ Icon labels in maps¹</p>	<p><i>Host description.</i> _Supported since 2.4.0.</p>
{HOST.DNS<1-9>}	<p>→ Trigger-based notifications and commands→ Internal notifications→ Global scripts (including confirmation text)→ Icon labels in maps¹→ Item key parameters²→ Host interface IP/DNS→ Database monitoring additional parameters⁵→ SSH and Telnet scripts⁵→ Web monitoring⁶→ Low-level discovery rule filter regular expressions⁸→ URL field of dynamic URL</p>	<p>_Host DNS name³. Supported in trigger names since 2.0.0.</p>

	<p>screen element⁸→ Trigger names and descriptions→ Trigger URLs¹⁰</p>	
{HOST.HOST<1-9>}	<p>→ Trigger-based notifications and commands→ Auto registration notifications→ Internal notifications→ Global scripts (including confirmation text)→ Item key parameters→ Icon labels in maps¹→ Host interface IP/DNS→ Database monitoring additional parameters⁵→ SSH and Telnet scripts⁵→ Web monitoring⁶→ Low-level discovery rule filter regular expressions⁸→ URL field of dynamic URL screen element⁸→ Trigger names and descriptions→ Trigger URLs¹⁰</p>	<p><i>Host name.</i> {HOSTNAME<1-9>} is deprecated.</p>
{HOST.ID<1-9>}	<p>→ Map URLs→ URL field of dynamic URL screen element⁸→ Trigger URLs¹⁰</p>	<p><i>Host ID.</i></p>
	<p>→ Trigger-based notifications and commands→ Auto registration notifications→ Internal notifications→ Global scripts</p>	

{HOST.IP<1-9>}	<p>(including confirmation text)→ Icon labels in maps¹→ Item key parameters²→ Host interface IP/DNS→ Database monitoring additional parameters⁵→ SSH and Telnet scripts⁵→ Web monitoring⁶→ Low-level discovery rule filter regular expressions⁸→ URL field of dynamic URL screen element⁸→ Trigger names and descriptions→ Trigger URLs¹⁰</p>	<p><i>Host IP address³. Supported since 2.0.0. {IPADDRESS<1-9>} is deprecated.</i></p>
{HOST.METADATA}	<p>→ Auto registration notifications</p>	<p><i>Host metadata._Used only for active agent auto-registration. Supported since 2.2.0.</i></p>
{HOST.NAME<1-9>}	<p>→ Trigger-based notifications and commands→ Auto registration notifications→ Internal notifications→ Global scripts (including confirmation text)→ Icon labels in maps¹→ Item key parameters→ Host interface IP/DNS→ Database monitoring additional parameters⁵→ SSH and Telnet scripts⁵→ Web monitoring⁶→ Low-level discovery rule filter regular</p>	<p><i>_Visible host name._Supported since 2.0.0.</i></p>

	<p>expressions⁸→ URL field of dynamic URL screen element⁸→ Trigger names and descriptions→ Trigger URLs¹⁰</p>	
{HOST.PORT<1-9>}	<p>→ Trigger-based notifications and commands→ Auto registration notifications→ Internal notifications→ Trigger names and descriptions→ Trigger URLs¹⁰</p>	<p>_Host (agent) port³.Supported in auto-registration since 2.0.0.Supported in trigger names, trigger descriptions, internal and trigger-based notifications since 2.2.2.</p>
{HOSTGROUP.ID}	→ Map URLs	<i>Host group ID.</i>
{INVENTORY.ALIAS<1-9>}	→ Trigger-based notifications→ Internal notifications	<i>Alias field in host inventory.</i>
{INVENTORY.ASSET.TAG<1-9>}	<p>→ Trigger-based notifications→ Internal notifications</p>	<i>Asset tag field in host inventory.</i>
{INVENTORY.CHASSIS<1-9>}	<p>→ Trigger-based notifications→ Internal notifications</p>	<i>Chassis field in host inventory.</i>
{INVENTORY.CONTACT<1-9>}	<p>→ Trigger-based notifications→ Internal notifications</p>	<i>Contact field in host inventory.</i> {PROFILE.CONTACT<1-9>} is deprecated.
{INVENTORY.CONTRACT.NUMBER<1-9>}	<p>→ Trigger-based notifications→ Internal notifications</p>	<i>Contract number field in host inventory.</i>
{INVENTORY.DEPLOYMENT.STATUS<1-9>}	<p>→ Trigger-based notifications→ Internal notifications</p>	<i>Deployment status field in host inventory.</i>
{INVENTORY.HARDWARE<1-9>}	<p>→ Trigger-based notifications→ Internal notifications</p>	<i>Hardware field in host inventory.</i> {PROFILE.HARDWARE<1-9>} is deprecated.

{INVENTORY.HARDWARE.FULL<1-9>}	→ Trigger-based notifications→ Internal notifications	<i>Hardware (Full details) field in host inventory.</i>
{INVENTORY.HOST.NETMASK<1-9>}	→ Trigger-based notifications→ Internal notifications	<i>Host subnet mask field in host inventory.</i>
{INVENTORY.HOST.NETWORKS<1-9>}	→ Trigger-based notifications→ Internal notifications	<i>Host networks field in host inventory.</i>
{INVENTORY.HOST.ROUTER<1-9>}	→ Trigger-based notifications→ Internal notifications	<i>Host router field in host inventory.</i>
{INVENTORY.HW.ARCH<1-9>}	→ Trigger-based notifications→ Internal notifications	<i>Hardware architecture field in host inventory.</i>
{INVENTORY.HW.DATE.DECOMM<1-9>}	→ Trigger-based notifications→ Internal notifications	<i>Date hardware decommissioned field in host inventory.</i>
{INVENTORY.HW.DATE.EXPIRY<1-9>}	→ Trigger-based notifications→ Internal notifications	<i>Date hardware maintenance expires field in host inventory.</i>
{INVENTORY.HW.DATE.INSTALL<1-9>}	→ Trigger-based notifications→ Internal notifications	<i>Date hardware installed field in host inventory.</i>
{INVENTORY.HW.DATE.PURCHASE<1-9>}	→ Trigger-based notifications→ Internal notifications	<i>Date hardware purchased field in host inventory.</i>
{INVENTORY.INSTALLER.NAME<1-9>}	→ Trigger-based notifications→ Internal notifications	<i>Installer name field in host inventory.</i>
{INVENTORY.LOCATION<1-9>}	→ Trigger-based notifications→ Internal notifications	<i>Location field in host inventory. {PROFILE.LOCATION<1-9>} is deprecated.</i>
	→ Trigger-based	<i>Location latitude field in</i>

{INVENTORY.LOCATION.LAT<1-9>}	notifications→Internal notifications	<i>Location latitude field in host inventory.</i>
{INVENTORY.LOCATION.LON<1-9>}	→ Trigger-based notifications→Internal notifications	<i>Location longitude field in host inventory.</i>
{INVENTORY.MACADDRESS.A<1-9>}	→ Trigger-based notifications→Internal notifications	<i>MAC address A field in host inventory. {PROFILE.MACADDRESS<1-9>} is deprecated.</i>
{INVENTORY.MACADDRESS.B<1-9>}	→ Trigger-based notifications→Internal notifications	<i>MAC address B field in host inventory.</i>
{INVENTORY.MODEL<1-9>}	→ Trigger-based notifications→Internal notifications	<i>Model field in host inventory.</i>
{INVENTORY.NAME<1-9>}	→ Trigger-based notifications→Internal notifications	<i>Name field in host inventory. {PROFILE.NAME<1-9>} is deprecated.</i>
{INVENTORY.NOTES<1-9>}	→ Trigger-based notifications→Internal notifications	<i>Notes field in host inventory. {PROFILE.NOTES<1-9>} is deprecated.</i>
{INVENTORY.OOB.IP<1-9>}	→ Trigger-based notifications→Internal notifications	<i>OOB IP address field in host inventory.</i>
{INVENTORY.OOB.NETMASK<1-9>}	→ Trigger-based notifications→Internal notifications	<i>OOB subnet mask field in host inventory.</i>
{INVENTORY.OOB.ROUTER<1-9>}	→ Trigger-based notifications→Internal notifications	<i>OOB router field in host inventory.</i>
{INVENTORY.OS<1-9>}	→ Trigger-based notifications→Internal notifications	<i>OS field in host inventory. {PROFILE.OS<1-9>} is deprecated.</i>
{INVENTORY.OS.FULL<1-9>}	→ Trigger-based notifications→Internal	<i>OS (Full details) field in host inventory.</i>

	notifications	
{INVENTORY.OS.SHORT<1-9>}	→ Trigger-based notifications→ Internal notifications	<i>OS (Short) field in host inventory.</i>
{INVENTORY.POC.PRIMARY.CELL<1-9>}	→ Trigger-based notifications→ Internal notifications	<i>Primary POC cell field in host inventory.</i>
{INVENTORY.POC.PRIMARY.EMAIL<1-9>}	→ Trigger-based notifications→ Internal notifications	<i>Primary POC email field in host inventory.</i>
{INVENTORY.POC.PRIMARY.NAME<1-9>}	→ Trigger-based notifications→ Internal notifications	<i>Primary POC name field in host inventory.</i>
{INVENTORY.POC.PRIMARY.NOTES<1-9>}	→ Trigger-based notifications→ Internal notifications	<i>Primary POC notes field in host inventory.</i>
{INVENTORY.POC.PRIMARY.PHONE.A<1-9>}	→ Trigger-based notifications→ Internal notifications	<i>Primary POC phone A field in host inventory.</i>
{INVENTORY.POC.PRIMARY.PHONE.B<1-9>}	→ Trigger-based notifications→ Internal notifications	<i>Primary POC phone B field in host inventory.</i>
{INVENTORY.POC.PRIMARY.SCREEN<1-9>}	→ Trigger-based notifications→ Internal notifications	<i>Primary POC screen name field in host inventory.</i>
{INVENTORY.POC.SECONDARY.CELL<1-9>}	→ Trigger-based notifications→ Internal notifications	<i>Secondary POC cell field in host inventory.</i>
{INVENTORY.POC.SECONDARY.EMAIL<1-9>}	→ Trigger-based notifications→ Internal notifications	<i>Secondary POC email field in host inventory.</i>
{INVENTORY.POC.SECONDARY.NAME<1-9>}	→ Trigger-based notifications→ Internal notifications	<i>Secondary POC name field in host inventory.</i>
	→ Trigger-	

{INVENTORY.PO.C SECONDARY.NOTES<1-9>}	→ Trigger-based notifications→ Internal notifications	<i>Secondary POC notes field in host inventory.</i>
{INVENTORY.PO.C SECONDARY.PHONE.A<1-9>}	→ Trigger-based notifications→ Internal notifications	<i>Secondary POC phone A field in host inventory.</i>
{INVENTORY.PO.C SECONDARY.PHONE.B<1-9>}	→ Trigger-based notifications→ Internal notifications	<i>Secondary POC phone B field in host inventory.</i>
{INVENTORY.PO.C SECONDARY.SCREEN<1-9>}	→ Trigger-based notifications→ Internal notifications	<i>Secondary POC screen name field in host inventory.</i>
{INVENTORY.SERIALNO.A<1-9>}	→ Trigger-based notifications→ Internal notifications	<i>Serial number A field in host inventory.</i> {PROFILE.SERIALNO<1-9>} is deprecated.
{INVENTORY.SERIALNO.B<1-9>}	→ Trigger-based notifications→ Internal notifications	<i>Serial number B field in host inventory.</i>
{INVENTORY.SITE.ADDRESS.A<1-9>}	→ Trigger-based notifications→ Internal notifications	<i>Site address A field in host inventory.</i>
{INVENTORY.SITE.ADDRESS.B<1-9>}	→ Trigger-based notifications→ Internal notifications	<i>Site address B field in host inventory.</i>
{INVENTORY.SITE.ADDRESS.C<1-9>}	→ Trigger-based notifications→ Internal notifications	<i>Site address C field in host inventory.</i>
{INVENTORY.SITE.CITY<1-9>}	→ Trigger-based notifications→ Internal notifications	<i>Site city field in host inventory.</i>
{INVENTORY.SITE.COUNTRY<1-9>}	→ Trigger-based notifications→ Internal notifications	<i>Site country field in host inventory.</i>
	→ Trigger-based notifications→ Internal notifications	<i>Site notes field in host</i>

	Internal notifications	
{INVENTORY.SITE.RACK<1-9>}	→ Trigger-based notifications→ Internal notifications	<i>Site rack location field in host inventory.</i>
{INVENTORY.SITE.STATE<1-9>}	→ Trigger-based notifications→ Internal notifications	<i>Site state/province field in host inventory.</i>
{INVENTORY.SITE.ZIP<1-9>}	→ Trigger-based notifications→ Internal notifications	<i>Site ZIP/postal field in host inventory.</i>
{INVENTORY.SOFTWARE<1-9>}	→ Trigger-based notifications→ Internal notifications	<i>Software field in host inventory.</i> {PROFILE.SOFTWARE<1-9>} is deprecated.
{INVENTORY.SOFTWARE.APP.A<1-9>}	→ Trigger-based notifications→ Internal notifications	<i>Software application A field in host inventory.</i>
{INVENTORY.SOFTWARE.APP.B<1-9>}	→ Trigger-based notifications→ Internal notifications	<i>Software application B field in host inventory.</i>
{INVENTORY.SOFTWARE.APP.C<1-9>}	→ Trigger-based notifications→ Internal notifications	<i>Software application C field in host inventory.</i>
{INVENTORY.SOFTWARE.APP.D<1-9>}	→ Trigger-based notifications→ Internal notifications	<i>Software application D field in host inventory.</i>
{INVENTORY.SOFTWARE.APP.E<1-9>}	→ Trigger-based notifications→ Internal notifications	<i>Software application E field in host inventory.</i>
{INVENTORY.SOFTWARE.FULL<1-9>}	→ Trigger-based notifications→ Internal notifications	<i>Software (Full details) field in host inventory.</i>
{INVENTORY.TAG<1-9>}	→ Trigger-based notifications→ Internal notifications	<i>Tag field in host inventory.</i> {PROFILE.TAG<1-9>} is deprecated.

	notifications	
{INVENTORY.TYPE<1-9>}	→ Trigger-based notifications→ Internal notifications	Type field in host inventory. {PROFILE.DEVICETYPE<1-9>} is deprecated.
{INVENTORY.TYPE.FULL<1-9>}	→ Trigger-based notifications→ Internal notifications	Type (Full details) field in host inventory.
{INVENTORY.URL.A<1-9>}	→ Trigger-based notifications→ Internal notifications	URL A field in host inventory.
{INVENTORY.URL.B<1-9>}	→ Trigger-based notifications→ Internal notifications	URL B field in host inventory.
{INVENTORY.URL.C<1-9>}	→ Trigger-based notifications→ Internal notifications	URL C field in host inventory.
{INVENTORY.VENDOR<1-9>}	→ Trigger-based notifications→ Internal notifications	Vendor field in host inventory.
{ITEM.DESCRIPTION<1-9>}	→ Trigger-based notifications→ Internal notifications	Description of the Nth item in the trigger expression that caused a notification. Supported since 2.0.0.
{ITEM.ID<1-9>}	→ Trigger-based notifications→ Internal notifications	Numeric ID of the Nth item in the trigger expression that caused a notification. Supported since 1.8.12.
{ITEM.KEY<1-9>}	→ Trigger-based notifications→ Internal notifications	Key of the Nth item in the trigger expression that caused a notification. Supported since 2.0.0. {TRIGGER.KEY} is deprecated.
{ITEM.KEY.ORIG<1-9>}	→ Trigger-based notifications→ Internal notifications	Original key (with macros not expanded) of the Nth item in the trigger expression that caused a notification. Supported since 2.0.6.
{ITEM.LASTVALUE<1-9>}	→ Trigger-based notifications→ Trigger names and	

Event tags and values



7 设定时间段

1 格式

要设定一个时间段，就要运用下面的格式：

```
1. d-d, hh:mm-mm:hh:mm
```

可以运用分号(;)、分隔符来指定大于一个的时间段

```
1. d-d, hh:mm-mm:hh:mm; d-d, hh:mm-mm:hh:mm ...
```

2 描述

符号	描述
d	星期：1 - Monday, 2 - Tuesday ,... , 7 - Sunday
hh	小时：00-24
mm	分钟：00-59

3 默认值

如果时间段的参数为空，系统将默认为01-07, 00:00-24:00

时间段的上限是开区间。因此，当你指定时间段为09:00-18:00时，该时间段包含的最后一秒钟将是17:59:59。该规则从1.8.7版本之后适用于各类设定，而working time一直沿用该规则。

4 举例

工作日。星期一到星期五的9:00到18:00：

```
1. 1-5, 09:00-18:00
```

工作日加上周末。星期一到星期五的9:00到18:00，以及周六周日的10:00到16:00。

```
1. 1-5, 09:00-18:00; 6-7, 10:00-16:00
```

8 执行指令

Zabbix用常规功能执行用户参数、远程指令、系统运行[]图标，不需要“nowait”字符、脚本（警告、外部、全球）和一些内部指令。

执行步骤

在Unix和Windows系统平台上，指令/脚本的执行方式相近

- Zabbix（父进程）创建了一个交流通道。
- Zabbix将通道设置为将要被创建的子进程的输出接口
- Zabbix创建子进程（运行指令/脚本）
- 为子进程创建一个新的进程组（Unix平台）或一个作业（Windows平台）
- Zabbix读取通道信息直到超时或者没有人写到另一端（所有的句柄/文件描述符都已关闭）。注意，子进程在退出或关闭句柄/文件描述符之前可以创建更多的进程并退出。
- 如果没有达到超时，Zabbix将等待直到初始子进程退出或超时
- 如果初始子进程退出且尚未达到超时时间，Zabbix将检查初始子进程的退出代码，并将其与0值进行比较。（非零值被认为是执行失败）
- 此时将假定一切都已完成，整个进程树（即进程组或作业）已被被终止

步骤5-8不要引用远程指令，因为它们是用“nowait”字符执行的。步骤7不要引用Zabbix agent执行的自定义脚本。

Zabbix假设指令/脚本在初始子进程退出时完成了进程，并且没有其他进程让输出句柄/文件描述符处于打开状态。当进程结束时，所有创建的进程都将被终止。

所有指令中的双引号和反斜杠都以反斜杠相间隔，指令要用双引号括起来。

退出代码的检查

对执行指令/脚本的退出代码检查的操作具备以下条件：

- 脚本的退出代码、远程指令、用户参数以及Zabbix agent items system.run（不带“nowait”字符）和system.hw.devices
- 任何不同于0值的退出代码被认为是执行失败
- 对执行失败的标准错误和标准输出的内容进行了收集，在前端（执行结果显示端）可获取该内容。
- 为Zabbix服务器上的远程指令创建了额外的日志条目以保存脚本的执行输出

可能出现的失败指令/脚本的前端信息和日志条目：

- 执行失败的标准错误和标准输出的内容（如果有的话）

- “进程退出代码:N”（输出为空，退出代码不为0）
 - “信号终止信号：N”（进程被信号终止，只适用于Linux系统）
 - “进程意外终止”（进程因不明原因终止）
-

了解更多关于[用户参数](#), [远程指令](#), [警告脚本](#).

9 监控方案

综述

监控服务器的有效性

至少可以使用三种方法（或各种方法的组合）以监控服务器的有效性。

- ICMP ping (“icmpping”键）
- “zabbix[host, agent, available]”图标
- 触发函数nodata()以监控只进行主动性检查的主机的有效性

通过WinPopUps发送警告

如果你正在运行Windows操作系统，想要从Zabbix获取快讯，WinPopUps可能大有用处。它是基于电子邮件的警告消息的很好补充。关于启用WinPopUps的细节信息，详见 <http://www.zabbix.com/forum/showthread.php?t=2147>。

监控特定的应用程序

AS/400

使用SNMP可以监视IBM AS/400平台，更多信息详见 <http://publib.boulder.ibm.com/Redbooks.nsf/RedbookAbstracts/sg244504.html?Open>。

MySQL

在agent配置文件夹/usr/local/etc/zabbix_agentd.conf中，可以用若干用户参数来监控MySQL

```

1. ### Set of parameters for monitoring MySQL server (v3.23.42 and later)
2. ### Change -u and add -p if required
3. #UserParameter=mysql.ping,mysqladmin -uroot ping|grep alive|wc -l
4. #UserParameter=mysql.uptime,mysqladmin -uroot status|cut -f2 -d":"|cut -f2 -d" "
5. #UserParameter=mysql.threads,mysqladmin -uroot status|cut -f3 -d":"|cut -f2 -d" "
6. #UserParameter=mysql.questions,mysqladmin -uroot status|cut -f4 -d":"|cut -f2 -d" "
7. #UserParameter=mysql.slowqueries,mysqladmin -uroot status|cut -f5 -d":"|cut -f2 -d" "
8. #UserParameter=mysql.qps,mysqladmin -uroot status|cut -f9 -d":"|cut -f2 -d" "
9. #UserParameter=mysql.version,mysql -V

```

- *mysql.ping*

检查MySQL是否运行正常

```
1.      Result: 0 - not started 1 - alive
```

- *mysql.uptime*

MySQL运行的秒数

- *mysql.threads*

MySQL的线程数量

- *mysql.questions*

处理查询数量

- *mysql.slowqueries*

慢查询数量

- *mysql.qps*

每秒查询量

- *mysql.version*

MySQL的版本。例如：mysql 14.14版本 Distrib 5.1.53, for pc-linux-gnu (i686)

获取更过信息，请访问conf/zabbix_agentd目录下的userparameter_mysql.conf文件

Mikrotik 路由器

使用由Mikrotik提供的SNMP agent。更多信息，详见<http://www.mikrotik.com> for more information.

Windows

使用包含(预编译)到Zabbix发行中的Zabbix Windows agent

Tuxedo

在定义一个用户参数时，可以使用Tuxedo Command Line实用程序tmadmin和qmadmin，以返回每个服务器/服务/队列性能计数器、Tuxedo资源的可用性。

Informix

用标准英孚美的utility onstat，几乎可以监控Informix数据库的各个方面。而且，Zabbix可以检索由Informix SNMP agent提供的信息。

HP OpenView

通过配置 Zabbix来向OpenView服务器发送消息，必须遵循以下几个步骤：

步骤 1

定义新媒体

媒体将执行一个向OpenView发送所需信息的脚本。

步骤 2

定义新用户

用户必须与媒体相连接

步骤 3

配置操作配置操作会给用户发送所有(或已选择的)触发器状态修改。

步骤 4

编写媒体脚本

脚本将有如下逻辑操作。如果触发器为ON，那么执行 OpenView 指令 `opcmsg -id application=<application> msg_grp=<msg_grp> object=<object> msg_text=<text>`。该指令将返回唯一的信息ID，该ID必须存储在某处，并在ZABBIX数据库的新列表里处于优先位置。如果触发器为OFF，那么 `opcmsgack <message_id>` 要和从数据库中检索的信息ID一同执行。

更多关于opcmsg和opcmsgack的详情，请见OpenView官方文件

10 性能调优

这是一个正在进行的工作 .

概述

使Zabbix系统正确调整以获得最佳性能是非常重要的。

硬件

关于硬件的一般建议：

- 使用最快的处理器
- SCSI或者SAS都是比IDE和SATA更好的选择（使用实用程序hdparm可以显着提高IDE磁盘的性能）
- 15K RPM优于10K RPM， 优于7200RPM
- 使用快速RAID存储
- 使用快速以太网适配器
- 拥有更多的内存总是更好

操作系统

- 使用最新（stable！）版本的操作系统
- 从内核中排除不必要的功能
- 调整内核参数

Zabbix配置参数

可以调整许多参数以获得最佳性能。

zabbix_server

StartPollers

一般规则 - 保持此参数的值尽可能低。 zabbix_server的每个附加实例都会添加已知的开销，同时，并行性增加。当队列平均包含最小参数数量（理想情况下，在任何给定时刻为0）时，实现最佳实例数。可以通过使用内部检查zabbix [queue]来监视此值。

参见 "[See also](#)" 以了解如何配置zabbix进程的最佳数量 .

DebugLevel

最佳值为3 .

DBSocket

仅限MySQL。建议使用DBSocket连接数据库。那是最快和最安全的方式。

数据库引擎

这可能是Zabbix调优中最重要的部分。Zabbix在很大程度上取决于数据库引擎的可用性和性能。

- 使用最快的数据库引擎，即MySQL或PostgreSQL
- 从源重建MySQL或PostgreSQL以获得最大的性能
- 遵循从MySQL或PostgreSQL文档获取的性能调优说明
- 对于MySQL，使用InnoDB表结构
- 如果使用InnoDB，ZABBIX的运行速度至少要快1.5倍（与MyISAM相比）。这是因为并行性增加。但是，InnoDB需要更多的CPU电源。
- 强烈建议调整数据库服务器以获得最佳性能。
- 将数据库表保留在不同的硬盘上
- “历史”，“历史记录”，“项目”，“触发器”和“趋势”是使用最多的表格。
- 对于大型安装，建议在tmpfs中保留MySQL临时文件

一般建议

- 仅监控所需参数
- 调整所有项目的“更新间隔”。保持较小的更新间隔对于漂亮的图形可能是好的，但是这可能会超载Zabbix
- 调整默认模板的参数
- 调整管理参数
- 不监视返回相同信息的参数。
- 避免使用长期给出的触发器作为函数参数。例如，max(3600)的计算速度明显比max(60)慢。

使用“ps”和“top”查看Zabbix进程性能

由于Zabbix 2.2进程更改其命令行以显示当前活动和有意义的统计信息，如：

```

1. UID          PID  PPID  C STIME TTY      TIME CMD
2. zabbix22  4584      1  0 14:55 ?    00:00:00 zabbix_server -c /home/zabbix22/zabbix_server.conf
3. zabbix22  4587  4584  0 14:55 ?    00:00:00 zabbix_server: configuration syncer [synced configuration in
  0.041169 sec, idle 60 sec]
4. zabbix22  4588  4584  0 14:55 ?    00:00:00 zabbix_server: db watchdog [synced alerts config in 0.018748 sec,
  idle 60 sec]
5. zabbix22  4608  4584  0 14:55 ?    00:00:00 zabbix_server: timer #1 [processed 3 triggers, 0 events in
  0.007867 sec, 0 maint.periods in 0.005677 sec, idle 30 sec]
6. zabbix22  4609  4584  0 14:55 ?    00:00:00 zabbix_server: timer #2 [processed 2 triggers, 0 events in
  0.004209 sec, idle 30 sec]
7. zabbix22  4637  4584  0 14:55 ?    00:00:01 zabbix_server: history syncer #4 [synced 35 items in 0.166198 sec,

```

```

idle 5 sec]
8. zabbix22 4657 4584 0 14:55 ? 00:00:00 zabbix_server: vmware collector #1 [updated 0, removed 0 VMware
services in 0.000004 sec, idle 5 sec]
9. zabbix22 4670 1 0 14:55 ? 00:00:00 zabbix_proxy -c /home/zabbix22/zabbix_proxy.conf
10. zabbix22 4673 4670 0 14:55 ? 00:00:00 zabbix_proxy: configuration syncer [synced config 15251 bytes in
0.111861 sec, idle 60 sec]
11. zabbix22 4674 4670 0 14:55 ? 00:00:00 zabbix_proxy: heartbeat sender [sending heartbeat message success
in 0.013643 sec, idle 30 sec]
12. zabbix22 4688 4670 0 14:55 ? 00:00:00 zabbix_proxy: icmp pinger #1 [got 1 values in 1.811128 sec, idle 5
sec]
13. zabbix22 4690 4670 0 14:55 ? 00:00:00 zabbix_proxy: housekeeper [deleted 9870 records in 0.2333491 sec,
idle 3599 sec]
14. zabbix22 4701 4670 0 14:55 ? 00:00:08 zabbix_proxy: http poller #2 [got 1 values in 0.024105 sec, idle 1
sec]
15. zabbix22 4707 4670 0 14:55 ? 00:00:00 zabbix_proxy: history syncer #4 [synced 22 items in 0.008565 sec,
idle 5 sec]
16. zabbix22 4738 1 0 14:55 ? 00:00:00 zabbix_agentd -c /home/zabbix22/zabbix_agentd.conf
17. zabbix22 4739 4738 0 14:55 ? 00:00:00 zabbix_agentd: collector [idle 1 sec]
18. zabbix22 4740 4738 0 14:55 ? 00:00:00 zabbix_agentd: listener #1 [waiting for connection]
19. zabbix22 4741 4738 0 14:55 ? 00:00:00 zabbix_agentd: listener #2 [processing request]

```

主要的过程是一个例外。 而不是当前的活动，显示原始的命令行。 这有助于区分具有多个Zabbix实例的系统上的进程。

Microsoft Windows不能实现此功能。

如果日志级别设置为 `DebugLevel = 4` 这些活动和统计信息也被写入日志文件。

Linux

在Linux系统上，“ps”命令可以与“watch”命令一起使用，以观察Zabbix的工作。例如，要每秒运行“ps”命令5次以查看进程活动：

```
1. watch -n 0.2 ps -fu zabbix
```

仅显示Zabbix代理和代理进程：

```
1. watch -tn 0.2 'ps -f -C zabbix_proxy -C zabbix_agentd'
```

仅显示历史记录进程：

```
1. watch -tn 0.2 'ps -fc zabbix_server | grep history'
```

“ps”命令产生一个宽输出（大约190列），因为一些活动消息很长。 如果您的终端有少于190列文本，您可以尝试

```
1. watch -tn 0.2 'ps -o cmd -C zabbix_server -C zabbix_proxy -C zabbix_agentd'
```

仅显示没有UID，PID，开始时间等的命令行

`top` 命令也可用于观察Zabbix的性能。在 `top` 中按‘c’键显示其命令行的进程。在我们对Linux“top”和“atop”的测试中，正确显示了Zabbix进程的变化活动，但是“htop”不显示不断变化的活动。

BSD systems

如果没有安装 `watch` 命令，可以实现类似的效果

```
1. while [ 1 ]; do ps x; sleep 0.2; clear; done
```

AIX, HP-UX

如果“watch”命令不可用，可以尝试

```
1. while [ 1 ]; do ps -fu zabbix; sleep 1; clear; done
```

Solaris

默认情况下，“ps”命令不显示更改的活动。一个选项是使用 `/usr/ucb/ps`。如果未安装“watch”命令，则可以显示一个周期性更新的进程列表

```
1. while [ 1 ]; do /usr/ucb/ps gxww; sleep 1; clear; done
```

On Solaris 11:

- “`/usr/ucb/ps`”默认情况下未安装。你可能需要安装 `ucb` package，例如。`pkg install compatibility/ucb`，
- 如果Zabbix守护进程已由特权用户启动，则其活动不会显示给非特权用户。
- `sleep` 命令不仅接受整秒钟，而且接受第二秒的分数（例如“睡眠0.2”）。

参见

- [如何配置zabbix进程的最佳计数](#)

11版本兼容性

支持的agents

Zabbix 1.x, 2.x和以前版本的Zabbix 3.x的旧版代理仍可与Zabbix 3.4一起使用。 它通常不需要在代理端进行任何配置更改，除了相关参数外[logging](#) 对于3.0之前的版本，您可能需要查看。

然而，要充分利用新的和改进的项目，提高性能和减少内存使用，使用最新的3.4代理。

支持的Zabbix proxies

只有Zabbix 3.4代理可以与Zabbix服务器一起使用。 Zabbix 3.4服务器不支持Zabbix 1.8, 2.0, 2.2, 2.4, 3.0和3.2代理。

Zabbix 3.4 proxies 只能与Zabbix 3.4服务器一起使用。 它们不能与3.2或更早版本的Zabbix服务器配合工作。

支持的 XML文件

在Zabbix 3.4中支持导入1.8, 2.0, 2.2, 2.4, 3.0和3.2的XML文件。

在Zabbix 1.8 XML导出格式中，触发依赖仅由名称存储。 如果有几个具有相同名称的触发器（例如，具有不同的严重性和表达式），它们之间定义了依赖关系，则无法导入它们。 必须从XML文件中手动删除这些依赖关系，并在导入后重新添加。

12 数据库错误处理

如果Zabbix检测到后端数据库不可访问，它将发送通知消息，并继续尝试连接到数据库。对于某些数据库引擎，会识别出特定的错误代码。

MySQL

- CR_CONN_HOST_ERROR
- CR_SERVER_GONE_ERROR
- CR_CONNECTION_ERROR
- CR_SERVER_LOST
- CR_UNKNOWN_HOST
- ER_SERVER_SHUTDOWN
- ER_ACCESS_DENIED_ERROR
- ER_ILLEGAL_GRANT_FOR_TABLE
- ER_TABLEACCESS_DENIED_ERROR
- ER_UNKNOWN_ERROR

13 Zabbix sender dynamic link library for Windows

Zabbix sender Windows动态链接库: In a Windows environment applications can send data to Zabbix server/proxy directly by using the Zabbix sender dynamic link library (`zabbix_sender.dll`) instead of having to launch an external process (`zabbix_sender.exe`).

在 Windows环境中，应用程序可以直接通过Zabbix sender 动态链接库向Zabbix server/proxy发送数据，而不用使用外部程序 (`zabbix_sender.exe`).The dynamic link library with the development files is located in `bin\winXX\dev` folders. To use it, include the `zabbix_sender.h` header file and link with the `zabbix_sender.lib` library. An example file with Zabbix sender API usage can be found in `build\win32\examples\zabbix_sender` folder. 动态链接库和开发源文件的位置是`bin\winXX\dev`文件夹。要使用的话，需要加入`zabbix_sender.h`页眉文件并且链接`zabbix_sender.lib`库。使用Zabbix sender API的示例文件请查看:
`build\win32\examples\zabbix_sender` 文件夹。The following functionality is provided by the Zabbix sender dynamic link library:以下列出了一些Zabbix sender动态链接库可以提供的功能点

<pre>1. int zabbixsender_send_values(const char address, unsigned short port, const char source, const zabbix_sender_value_t values, int count, char result);</pre>	
Description	Sends an array of value structures to the proxy/server. A value structure contains host name, item key and item value.
Parameters	<p><code>address*</code> - [in] the server/proxy address <code>port</code> - [in] the trapper port on the server/proxy <code>source</code> - [in] the source IP (optional) <code>values</code> - [in] an array of values to send <code>count</code> - [in] the number of items in values array <code>result</code> - [out] the server response or an error message (optional)</p>
Return value	<code>0</code> - operation completed successfully- <code>1</code> - operation failed
Comments	If the <code>result</code> variable is specified this function allocates the necessary memory to

store server response/error message. It must be always freed afterwards with `zabbix_sender_free_result()` function. If operation was successful the result contains server response which can be parsed with `zabbix_sender_parse_result()` function. In the case of error the error message will be stored into result instead.

```
1.      int
zabbix_sender_parse_result(const
    char *result, int response,
    zabbix_sender_info_t
    info);
```

Description	Parses the result returned from <code>zabbix_sender_send_values()</code> function.
Parameters	<p><code>*result</code> - [in] the result returned by <code>zabbix_sender_send_values()</code> function. <code>response</code> - [out] the operation response: 0 - success, -1 failed. <code>info</code> - [out] the detailed information about operation, optional.</p>
Return value	<p>0 - operation completed successfully-1 - operation failed</p>
Comments	If <code>info</code> parameter was specified, but the function failed to parse the result <code>info</code> field, then the <code>info</code> structure field <code>total</code> is set to -1.
1.	<pre>void zabbix_sender_free_result(void result);</pre>
Description	Frees data allocated by <code>zabbix_sender_send_values()</code> function.
Parameters	<code>*result</code> - the result returned by

zabbix_sender_send_values() function.

The following data structures are used by the Zabbix sender dynamic link library:

```
1. typedef struct
2. {
3.     /* host name, must match the name of target host in Zabbix */
4.     char *host;
5.     /* the item key */
6.     char *key;
7.     /* the item value */
8.     char *value;
9. }
10. zabbix_sender_value_t;
11.
12. typedef struct
13. {
14.     /* number of total values processed */
15.     int total;
16.     /* number of failed values */
17.     int failed;
18.     /* time in seconds the server spent processing the sent values */
19.     double time_spent;
20. }
21. zabbix_sender_info_t;
```

