

3 自动发现[LLD]

概述

Low-level discovery provides a way to automatically create items, triggers, and graphs for different entities on a computer. For instance, Zabbix can automatically start monitoring file systems or network interfaces on your machine, without the need to create items for each file system or network interface manually. Additionally it is possible to configure Zabbix to remove unneeded entities automatically based on actual results of periodically performed discovery.

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。

In Zabbix, six types of discovery items are supported out of the box:

在Zabbix中，支持六种类型的发现项目：

- discovery of file systems;
- 系统文件的发现;
- discovery of network interfaces;
- 网络接口的发现;
- discovery of CPUs and CPU cores;
- CPU和CPU内核的发现
- discovery of SNMP OIDs;
- SNMP OID的发现
- discovery using ODBC SQL queries;
- 使用ODBC SQL查询的发现
- discovery of Windows services.
- Windows服务的发现

A user can define their own types of discovery, provided they follow a particular JSON protocol.

用户可以自己定义发现类型，只要它们遵循特定的JSON协议。

The general architecture of the discovery process is as follows.

发现过程的一般架构如下。

First, a user creates a discovery rule in “Configuration” → “Templates” → “Discovery” column. A discovery rule consists of (1) an item that discovers the necessary entities (for instance, file systems or network interfaces) and (2) prototypes of items, triggers, and graphs that should be created based on the value of that item.

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

An item that discovers the necessary entities is like a regular item seen elsewhere: the server asks a Zabbix agent (or whatever the type of the item is set to) for a value of that item, the agent responds with a textual value. The difference is that the value the agent responds with should contain a list of discovered entities in a specific JSON format. While the details of this format are only important for implementers of custom discovery checks, it is necessary to know that the returned value contains a

list of macro → value pairs. For instance, item “net.if.discovery” might return two pairs: “{#IFNAME}” → “lo” and “{#IFNAME}” → “eth0”.

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”

Low-level discovery items “vfs.fs.discovery” and “net.if.discovery” are supported since Zabbix agent version 2.0.

Discovery item “system.cpu.discovery” is supported since Zabbix agent version 2.4.

Discovery of SNMP OIDs is supported since Zabbix server and proxy version 2.0.

Discovery using ODBC SQL queries is supported since Zabbix server and proxy version 3.0.

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。

Return values of a low-level discovery rule are limited to 2048 bytes on a Zabbix proxy run with IBM DB2 database. This limit does not apply to Zabbix server as return values are processed without being stored in a database.

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。

These macros are used in names, keys and other prototype fields where they are then substituted with the received values for creating real items, triggers, graphs or even hosts for each discovered entity. See the full list of [options](#) for using LLD macros.

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。

When the server receives a value for a discovery item, it looks at the macro → value pairs and for each pair generates real items, triggers, and graphs, based on their prototypes. In the example with “net.if.discovery” above, the server would generate one set of items, triggers, and graphs for the loopback interface “lo”, and another set for interface “eth0”.

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”

The following sections illustrate the process described above in detail and serve as a how-to for performing all types of discovery mentioned above. The last section describes the JSON format for discovery items and gives an example of how to implement your own file system discoverer as a Perl script.

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

To configure the discovery of file systems, do the following:

要配置文件系统的发现，请执行以下操作：

- Go to: *Configuration* → *Templates*
- 转到：配置 → 模板
- Click on *Discovery* in the row of an appropriate template
- 在一个合适的模板的行点击发现

Templates						
<input type="checkbox"/> Name ▲	Applications	Items	Triggers	Graphs	Screens	Discovery
<input type="checkbox"/> Template OS Linux	Applications 10	Items 32	Triggers 15	Graphs 5	Screens 1	Discovery 2

- Click on *Create discovery rule* in the upper right corner of the screen
- 单击屏幕右上角的创建发现规则
- Fill in the form with the following details
- 填写以下详细信息。

The **Discovery rule** tab contains general discovery rule attributes:

发现规则选项卡包含常规发现规则属性：

Discovery rule [Filters](#)

Name

Type

Key

Update interval

Custom intervals

Type	Interval	Period
<input checked="" type="checkbox"/> Flexible <input type="checkbox"/> Scheduling	<input type="text" value="50s"/>	<input type="text" value="1-7,00:00-24:00"/>

[Add](#)

Keep lost resources period

Description

Enabled

参数	描述
名称	发现规则名称。
类型	The type of check to perform discovery; should be <i>Zabbix agent</i> or <i>Zabbix agent (active)</i> for file system discovery.
类型	执行发现的检查类型；可以是 <i>Zabbix agent</i> 或 <i>Zabbix agent</i> （主动）文件系统发现。
键值	An item with “vfs.fs.discovery” key is built into the Zabbix agent on many platforms (see supported item key list for details), and will return a JSON with the list of file systems present on the computer and their types.
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。
Update interval (in sec)	This field specifies how often Zabbix performs discovery. In the beginning, when you are just setting up file system discovery, you might wish to set it to a small interval, but once you know it works you can set it to 30 minutes or more, because file systems usually do not change very often. <i>Note:</i> If set to '0', the item will not be polled. However, if a flexible interval also exists with a non-zero value, the item will be polled during the flexible interval duration.
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。 <i>注意：</i> 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。
Custom intervals	You can create custom rules for checking the item: Flexible - create an exception to the <i>Update interval</i> (interval with different frequency) Scheduling - create a custom polling schedule. For detailed information see Custom intervals . Scheduling is supported since Zabbix 3.0.0.
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
Keep lost resources period (in days)	This field allows you to specify for how many days the discovered entity will be retained (won't be deleted) once its discovery status becomes “Not discovered anymore” (max 3650 days). <i>Note:</i> If set to “0”, entities will be deleted immediately. Using “0” is not recommended, since just wrongly editing the filter may end up in the entity being deleted with all the historical data.
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。 <i>注意：</i> 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。

The **Filters** tab contains discovery rule filter definitions:

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	matches	Regular expression
A	{#FSTYPE}	@File systems for discover	
B	{#MACRO}	regular expression	
Add			

Add
Cancel

参数	描述
<i>Type of calculation</i>	<p>The following options for calculating filters are available:</p> <ul style="list-style-type: none"> And - all filters must be passed; Or - enough if one filter is passed; And/Or - uses <i>And</i> with different macro names and <i>Or</i> with the same macro name; Custom expression - offers the possibility to define a custom calculation of filters. The formula must include all filters in the list. Limited to 255 symbols.
<i>计算方式</i>	<p>计算过滤器的可用选项如下：</p> <ul style="list-style-type: none"> 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
<i>Filters</i>	<p>A filter can be used to generate real items, triggers, and graphs only for certain file systems. It expects a POSIX Extended Regular Expression. For instance, if you are only interested in C:, D:, and E: file systems, you could put {#FSNAME} into "Macro" and "^C ^D ^E" regular expression into "Regular expression" text fields. Filtering is also possible by file system types using {#FSTYPE} macro (e.g. "^ext ^reiserfs") and by drive types (supported only by Windows agent) using {#FSDRIVETYPE} macro (e.g., "fixed"). You can enter a regular expression or reference a global regular expression in "Regular expression" field.</p> <p>In order to test a regular expression you can use "grep -E", for example: <pre>for f in ext2 nfs reiserfs smbfs; do echo \$f grep -E '^ext ^reiserfs' echo "SKIP: \$f"; done</pre></p> <p>{#FSDRIVETYPE} macro on Windows is supported since Zabbix 3.0.0. Defining several filters is supported since Zabbix 2.4.0. Note that if some macro from the filter is missing in the response, the found entity will be ignored.</p>

参数	描述
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{#FSDRIVETYPE}宏（例如"fixed"[]的驱动器类型（仅由Windows agent支持）进行过滤。</p> <p>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 []</p> <p>为了测试正则表达式，你可以使用“grep -E”[]例如：</p> <pre>for f in ext2 nfs reiserfs smbfs; do echo \$f grep -E '^ext ^reiserfs' echo "SKIP: \$f"; done</pre> <p>从Zabbix 3.0.0起支持Windows上的宏{#FSDRIVETYPE} []</p> <p>Zabbix 2.4.0起支持定义多个过滤器。</p> <p>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。.</p>

Zabbix database in MySQL must be created as case-sensitive if file system names that differ only by case are to be discovered correctly.

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。

Discovery rule history is not preserved.

发现规则历史记录不被保留。

Once a rule is created, go to the items for that rule and press “Create prototype” to create an item prototype. Note how macro {#FSNAME} is used where a file system name is required. When the discovery rule is processed, this macro will be substituted with the discovered file system.

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

Type

Key

Type of information

Units

Update interval

Custom intervals

Type	Interval	Period
<input checked="" type="checkbox"/> Flexible <input type="checkbox"/> Scheduling	<input type="text" value="50s"/>	<input type="text" value="1-7,00:00-24:00"/>

[Add](#)

History storage period

Trend storage period

Show value [show value mappings](#)

New application

Applications

- None-
- CPU
- Filesystems
- General
- Memory
- Network interfaces
- OS
- Performance
- Processes
- Security

New application prototype

Application prototypes

- None-

Description

Create enabled

Attributes that are specific for item prototypes: 监控项原型特有的属性:

参数	描述
<i>New application prototype</i>	You may define a new application prototype. In application prototypes you can use low-level discovery macros that, after discovery, will be substituted with real values to create applications that are specific for the discovered entity. See also application discovery notes for more specific information.
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
<i>Application prototypes</i>	Select from the existing application prototypes.
应用原型	从现有应用原型中选择。
<i>Create enabled</i>	If checked the item will be added in an enabled state. If unchecked, the item will be added to a discovered entity, but in a disabled state.
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。

We can create several item prototypes for each file system metric we are interested in:

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:

The screenshot shows the Zabbix web interface for 'Item prototypes' under the 'Mounted filesystem discovery' template. The table contains the following data:

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

Then, we create trigger prototypes in a similar way:

然后，我们以类似的方式创建触发器原型:

Trigger prototype
Dependencies

Name

Severity Not classified Information Warning Average High Critical

Expression

[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"/>
Add	

Allow manual close

URL

Description

Create enabled

Attributes that are specific for trigger prototypes:

触发原型特有的属性:

参数	描述
创建启用	If checked the trigger will be added in an enabled state. If unchecked, the trigger will be added to a discovered entity, but in a disabled state.
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。

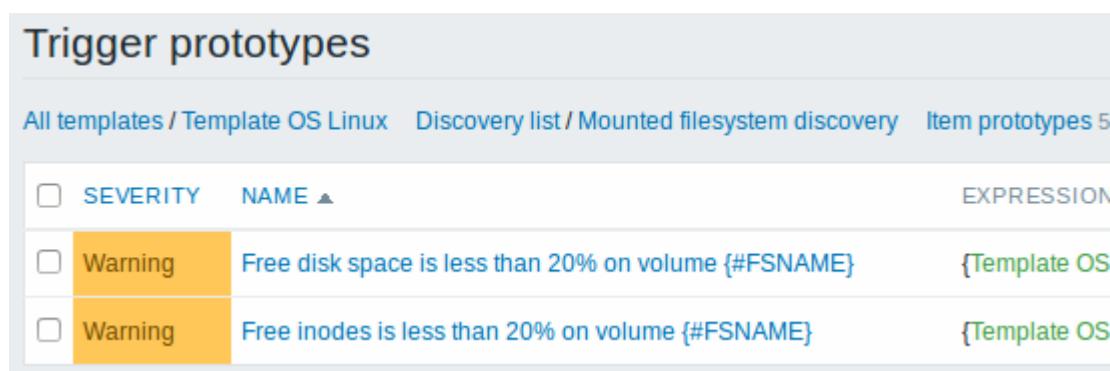
When real triggers are created from the prototypes, there may be a need to be flexible as to what constant ('20' in our example) is used for comparison in the expression. See how [user macros with context](#) can be useful to accomplish such flexibility.

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。

You can define [dependencies](#) between trigger prototypes as well (supported since Zabbix 3.0). To do that, go to the *Dependencies* tab. A trigger prototype may depend on another trigger prototype from the same low-level discovery (LLD) rule or on a regular trigger. A trigger prototype may not depend on a trigger prototype from a different LLD rule or on a trigger created from trigger prototype. Host trigger prototype cannot depend on a trigger from a template.

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。触发原型也可不依赖于不同的LLD规则触发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。

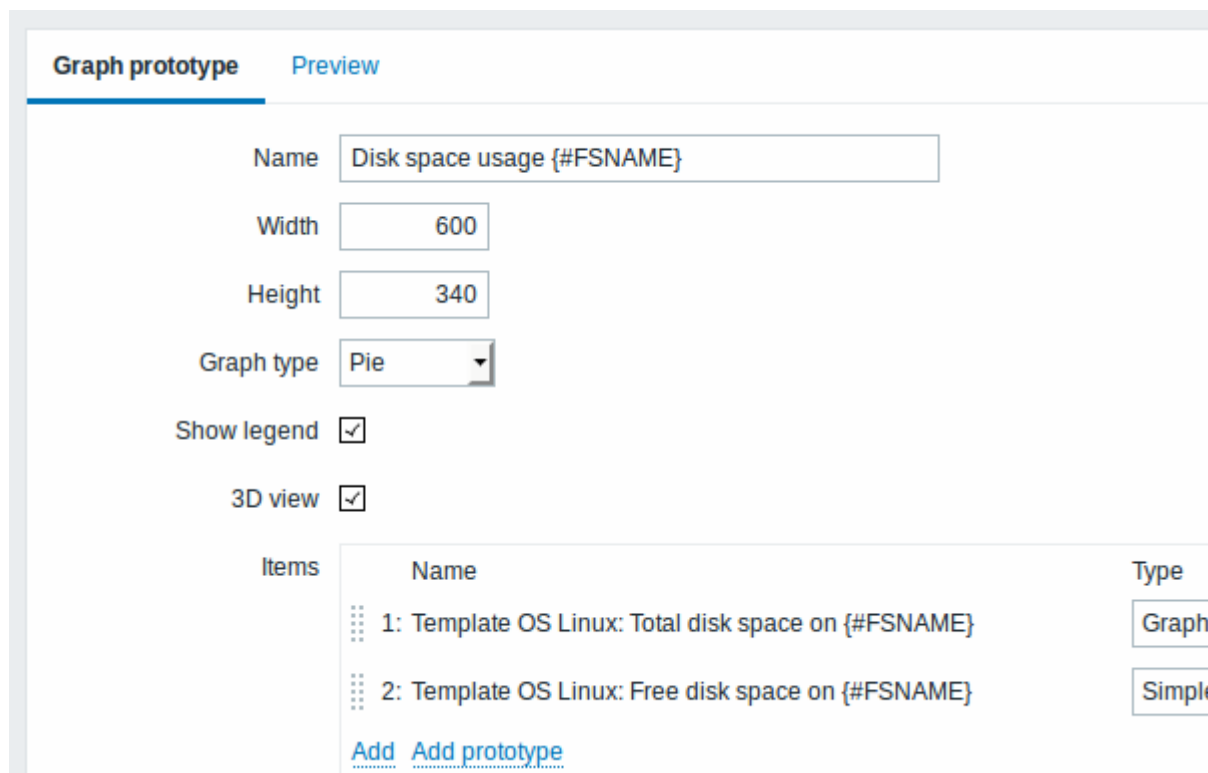


The screenshot shows a table titled "Trigger prototypes" with columns for SEVERITY, NAME, and EXPRESSION. Two rows are visible, both with a "Warning" severity and "Template OS" expression.

<input type="checkbox"/>	SEVERITY	NAME ▲	EXPRESSION
<input type="checkbox"/>	Warning	Free disk space is less than 20% on volume {#FSNAME}	{Template OS}
<input type="checkbox"/>	Warning	Free inodes is less than 20% on volume {#FSNAME}	{Template OS}

We can create graph prototypes, too:

我们也可以创建图形原型:



The screenshot shows the "Graph prototype" configuration form. Fields include Name, Width, Height, Graph type, Show legend, and 3D view. An "Items" table lists two items with their names and types.

Graph prototype [Preview](#)

Name:

Width:

Height:

Graph type:

Show legend:

3D view:

Items	Name	Type
⋮	1: Template OS Linux: Total disk space on {#FSNAME}	<input type="text" value="Graph"/>
⋮	2: Template OS Linux: Free disk space on {#FSNAME}	<input type="text" value="Simple"/>

[Add](#) [Add prototype](#)

Graph prototypes

All templates / Template OS Linux Discovery list / Mounted filesystem discovery Item prototypes 5

<input type="checkbox"/> NAME ▲	WIDTH
<input type="checkbox"/> Disk space usage {#FSNAME}	600

Finally, we have created a discovery rule that looks like shown below. It has five item prototypes, two trigger prototypes, and one graph prototype.

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

<input type="checkbox"/> NAME ▲	ITEMS	TRIGGERS	GRAPHS	H
<input type="checkbox"/> Mounted filesystem discovery	Item prototypes 5	Trigger prototypes 2	Graph prototypes 1	H

Note: For configuring host prototypes, see the section about [host prototype](#) configuration in virtual machine monitoring.

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。

The screenshots below illustrate how discovered items, triggers, and graphs look like in the host's configuration. Discovered entities are prefixed with an orange link to a discovery rule they come from.

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。

Items

All hosts / Zabbix server 1 Enabled ZBX SNMP JMX IPMI Applications 12 Items 74 Triggers 4

Filter ▼

<input type="checkbox"/> Wizard	Name	Triggers	Key ▲
<input type="checkbox"/>	Mounted filesystem discovery : Free inodes on / (percentage)	Triggers 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)	Triggers 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

Note that discovered entities will not be created in case there are already existing entities with the same uniqueness criteria, for example, an item with the same key or graph with the same name.

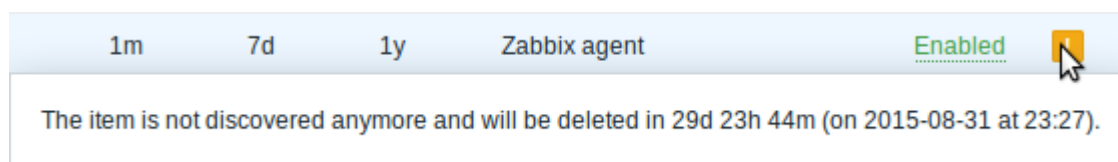
请注意，如果已经存在具有相同唯一性条件的现有实体，例如具有相同键值或具有相同名称的图形的项目，则不会创建发现的实体。

Items (similarly, triggers and graphs) created by a low-level discovery rule will be deleted automatically if a discovered entity (file system, interface, etc) stops being discovered (or does not pass the filter anymore). In this case the items, triggers and graphs will be deleted after the days defined in the *Keep lost resources period* field pass.

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时间过去后被删除。

When discovered entities become 'Not discovered anymore', a lifetime indicator is displayed in the item list. Move your mouse pointer over it and a message will be displayed indicating how many days are left until the item is deleted.

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。

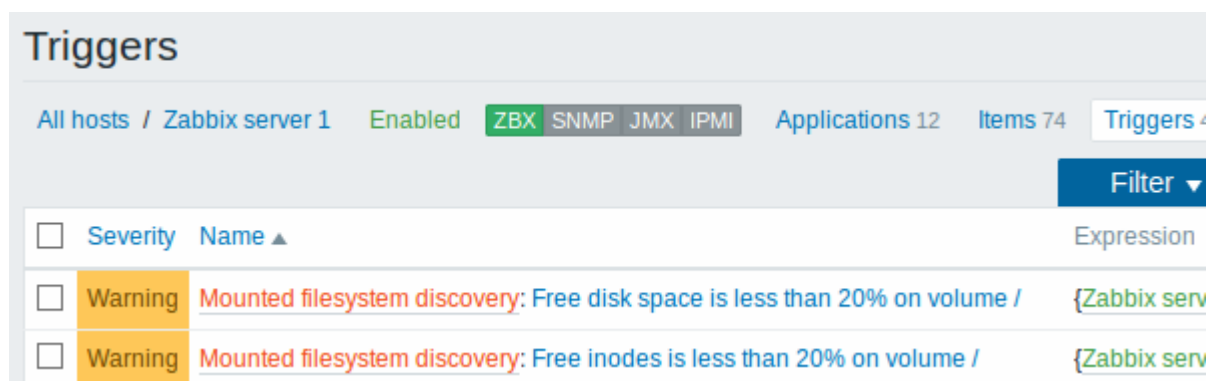


If entities were marked for deletion, but were not deleted at the expected time (disabled discovery rule or item host), they will be deleted the next time the discovery rule is processed.

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。

Entities containing other entities, which are marked for deletion, will not update if changed on the discovery rule level. For example, LLD-based triggers will not update if they contain items that are marked for deletion.

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。



The screenshot shows the Zabbix 'Graphs' page for 'Zabbix server 1'. The page is filtered to show 'Enabled' items. A list of items is displayed with checkboxes and names:

- Name ▲
- Template OS Linux_b: CPU jumps
- Template OS Linux_b: CPU load
- Template OS Linux_b: CPU utilization
- Mounted filesystem discovery: Disk space usage /

At the top right, there is a 'Group' dropdown menu set to 'all'. Below the list, there are statistics: 'Applications 12', 'Items 74', and 'Triggers 4'.

3.2 网络接口的发现

Discovery of network interfaces is done in exactly the same way as discovery of file systems, except that you use the discovery rule key “net.if.discovery” instead of “vfs.fs.discovery” and use macro {#IFNAME} instead of {#FSNAME} in filter and item/trigger/graph prototypes.

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}

Examples of item prototypes that you might wish to create based on “net.if.discovery”:
 “net.if.in[{#IFNAME},bytes]”, “net.if.out[{#IFNAME},bytes]”.

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]”

[See above](#) for more information about the filter.

有关过滤器的更多信息，请参阅[上文](#)

3.3 Discovery of CPUs and CPU cores

Discovery of CPUs and CPU cores is done in a similar fashion as network interface discovery with the exception being that the discovery rule key is “system.cpu.discovery”. This discovery key returns two macros - {#CPU.NUMBER} and {#CPU.STATUS} identifying the CPU order number and status respectively. To note, a clear distinction cannot be made between actual, physical processors, cores and hyperthreads. {#CPU.STATUS} on Linux, UNIX and BSD systems returns the status of the processor, which can be either “online” or “offline”. On Windows systems, this same macro may represent a third value - “unknown” - which indicates that a processor has been detected, but no information has been collected for it yet.

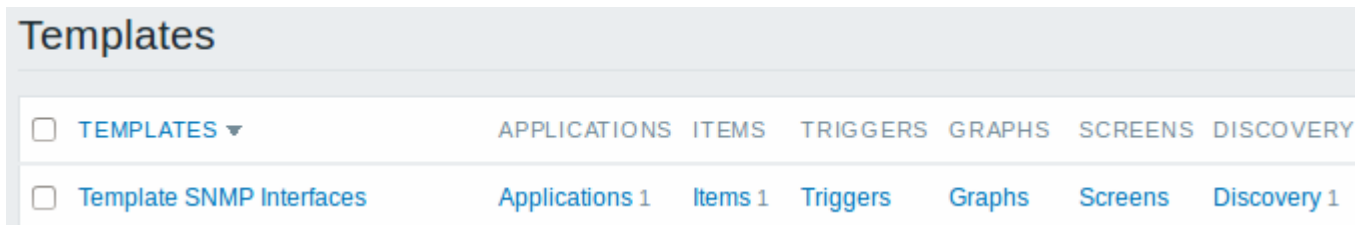
CPU discovery relies on the agent's collector process to remain consistent with the data provided by the collector and save resources on obtaining the data. This has the effect of this item key not working with the test (-t) command line flag of the agent binary, which will return a NOT_SUPPORTED status and an accompanying message indicating that the collector process has not been started.

Item prototypes that can be created based on CPU discovery include, for example,
 “system.cpu.util[{#CPU.NUMBER}, <type>, <mode>]” or “system.hw.cpu[{#CPU.NUMBER},

<info>]”.

3.4 Discovery of SNMP OIDs

In this example, we will perform SNMP discovery on a switch. First, go to “Configuration” → “Templates”.



To edit discovery rules for a template, click on the link in the “Discovery” column.

Then, press “Create rule” and fill the form with the details in the screenshot below.

Unlike file system and network interface discovery, the item does not necessarily have to have “snmp.discovery” key - item type of SNMP agent is sufficient.

The OIDs to discover are defined in SNMP OID field in the following format: `discovery[{#MACRO1}, oid1, {#MACRO2}, oid2, ...,]`

where `{#MACRO1}`, `{#MACRO2}` ... are valid lld macro names and `oid1`, `oid2`... are OIDs capable of generating meaningful values for these macros. A built-in macro `{#SNMPINDEX}` containing index of the discovered OID is applied to discovered entities. The discovered entities are grouped by `{#SNMPINDEX}` macro value.

To understand what we mean, let us perform few snmpwalks on our switch:

```
$ snmpwalk -v 2c -c public 192.168.1.1 IF-MIB::ifDescr
IF-MIB::ifDescr.1 = STRING: WAN
IF-MIB::ifDescr.2 = STRING: LAN1
IF-MIB::ifDescr.3 = STRING: LAN2

$ snmpwalk -v 2c -c public 192.168.1.1 IF-MIB::ifPhysAddress
IF-MIB::ifPhysAddress.1 = STRING: 8:0:27:90:7a:75
IF-MIB::ifPhysAddress.2 = STRING: 8:0:27:90:7a:76
IF-MIB::ifPhysAddress.3 = STRING: 8:0:27:2b:af:9e
```

And set SNMP OID to: `discovery[{#IFDESCR}, ifDescr, {#IFPHYSADDRESS}, ifPhysAddress]`

Now this rule will discover entities with `{#IFDESCR}` macros set to **WAN**, **LAN1** and **LAN2**, `{#IFPHYSADDRESS}` macros set to **8:0:27:90:7a:75**, **8:0:27:90:7a:76**, and **8:0:27:2b:af:9e**, `{#SNMPINDEX}` macros set to the discovered OIDs indexes **1**, **2** and **3**:

```
{
```

```

"data": [
  {
    "{#SNMPINDEX}": "1",
    "{#IFDESCR}": "WAN",
    "{#IFPHYSADDRESS}": "8:0:27:90:7a:75"
  },
  {
    "{#SNMPINDEX}": "2",
    "{#IFDESCR}": "LAN1",
    "{#IFPHYSADDRESS}": "8:0:27:90:7a:76"
  },
  {
    "{#SNMPINDEX}": "3",
    "{#IFDESCR}": "LAN2",
    "{#IFPHYSADDRESS}": "8:0:27:2b:af:9e"
  }
]
}

```

If an entity does not have the specified OID, then the corresponding macro will be omitted for this entity. For example if we have the following data:

```

ifDescr.1 "Interface #1"
ifDescr.2 "Interface #2"
ifDescr.4 "Interface #4"

ifAlias.1 "eth0"
ifAlias.2 "eth1"
ifAlias.3 "eth2"
ifAlias.5 "eth4"

```

Then in this case SNMP discovery `discovery[{#IFDESCR}, ifDescr, {#IFALIAS}, ifAlias]` will return the following structure:

```

{
  "data": [
    {
      "{#SNMPINDEX}": 1,
      "{#IFDESCR}": "Interface #1",
      "{#IFALIAS}": "eth0"
    },
    {
      "{#SNMPINDEX}": 2,
      "{#IFDESCR}": "Interface #2",
      "{#IFALIAS}": "eth1"
    },
    {
      "{#SNMPINDEX}": 3,
      "{#IFALIAS}": "eth2"
    },
    {

```

```
{
  "#SNMPINDEX": 4,
  "#IFDESCR": "Interface #4"
},
{
  "#SNMPINDEX": 5,
  "#IFALIAS": "eth4"
}
]
```

Discovery rule **Filters**

Name

Type

Key

SNMP OID

SNMP community

Port

Update interval

Custom intervals

Type	Interval	Period
<input checked="" type="checkbox"/> Flexible <input type="checkbox"/> Scheduling	<input type="text" value="50s"/>	<input type="text" value="1-7,00:00-24:00"/>

[Add](#)

Keep lost resources period

Description

Enabled

The following screenshot illustrates how we can use these macros in item prototypes:

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
<input checked="" type="checkbox"/>	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

Again, creating as many item prototypes as needed:

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

As well as trigger prototypes:

Trigger prototype Dependencies

Name

Severity Not classified Information Warning Average High Critical

Expression

[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"/>
----------------------------------	------------------------------------	---------------------------------------

[Add](#)

Allow manual close

URL

Description

Create enabled

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}	{Temp

And graph prototypes:

Graph prototype [Preview](#)

Name

Width

Height

Graph type

Show legend

Show working time

Show triggers

Percentile line (left)

Percentile line (right)

Y axis MIN value

Y axis MAX value

Items	Name	Function	Draw st
⋮	1: Template SNMP Interfaces: Incoming traffic on interface {#IFDESCR}	<input type="text" value="avg"/>	<input type="text" value="Gradie"/>
⋮	2: Template SNMP Interfaces: Outgoing traffic on interface {#IFDESCR}	<input type="text" value="avg"/>	<input type="text" value="Gradie"/>

[Add](#) [Add prototype](#)

Graph prototypes

[All templates / Template SNMP Interfaces](#) [Discovery list / Network interfaces](#) [Item prototypes 8](#)

<input type="checkbox"/> NAME ▲	WIDTH
<input type="checkbox"/> Traffic on interface {#SNMPVALUE}	900

A summary of our discovery rule:

Discovery rules

[All templates / Template SNMP Interfaces](#) [Applications 1](#) [Items 1](#) [Triggers](#) [Graphs](#) [Screens](#)

<input type="checkbox"/> NAME ▲	ITEMS	TRIGGERS	GRAPHS	HO
<input type="checkbox"/> Network interfaces	Item prototypes 8	Trigger prototypes 1	Graph prototypes 1	Ho

When server runs, it will create real items, triggers and graphs based on the values the SNMP discovery rule returns. In the host configuration they are prefixed with an orange link to a discovery rule they come from.

Items

All hosts / Switch1 Enabled ZBX SNMP JMX IPMI Applications 1 Items 241 Triggers 30 Gr

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 Gr

Filter

<input type="checkbox"/>	Severity	Name	Exp
<input type="checkbox"/>	Information	Network interfaces: Operational status was changed on {HOST.NAME} interface 1	{pr
<input type="checkbox"/>	Information	Network interfaces: Operational status was changed on {HOST.NAME} interface 2	{pr
<input type="checkbox"/>	Information	Network interfaces: Operational status was changed on {HOST.NAME} interface 3	{pr
<input type="checkbox"/>	Information	Network interfaces: Operational status was changed on {HOST.NAME} interface 4	{pr

Graphs

Group all

All hosts / Switch1 Enabled ZBX SNMP JMX IPMI Applications 1 Items 241 Triggers 30 Gr

<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 Discovery using ODBC SQL queries

This type of discovery is done using SQL queries, whose results get automatically transformed into a

JSON object suitable for low-level discovery. SQL queries are performed using items of type “Database monitor”. Therefore, most of the instructions on [ODBC monitoring](#) page apply in order to get a working “Database monitor” discovery rule, the only difference being that “db.odbc.discovery[<description>,<dsn>]” key should be used instead of “db.odbc.select[<description>,<dsn>]”.

As a practical example to illustrate how the SQL query is transformed into JSON, let us consider low-level discovery of Zabbix proxies by performing an ODBC query on Zabbix database. This is useful for automatic creation of “zabbix[proxy,<name>,lastaccess]” [internal items](#) to monitor which proxies are alive.

Let us start with discovery rule configuration:

The screenshot shows the configuration page for a discovery rule named "Proxy discovery". The "Type" is set to "Database monitor". The "Key" is "db.odbc.discovery[proxies,{\$DSN}]". The "SQL query" is "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;". The "Update interval" is "1h". Under "Custom intervals", there is one entry with "Type" "Flexible", "Scheduling" selected, "Interval" "50s", and "Period" "1-7,00:00-24:00". The "Keep lost resources period" is "30d". The "Enabled" checkbox is checked. "Add" and "Cancel" buttons are at the bottom.

Type	Interval	Period	
Flexible	Scheduling	50s	1-7,00:00-24:00

Here, the following direct query on Zabbix database is used to select all Zabbix proxies, together with the number of hosts they are monitoring. The number of hosts can be used, for instance, to filter out

empty proxies:

```
mysql> SELECT h1.host, COUNT(h2.host) AS count FROM hosts h1 LEFT JOIN hosts
h2 ON h1.hostid = h2.proxy_hostid WHERE h1.status IN (5, 6) GROUP BY
h1.host;
+-----+-----+
| host   | count |
+-----+-----+
| Japan 1 |     5 |
| Japan 2 |    12 |
| Latvia  |     3 |
+-----+-----+
3 rows in set (0.01 sec)
```

By the internal workings of “db.odbc.discovery[]” item, the result of this query gets automatically transformed into the following JSON:

```
{
  "data": [
    {
      "#{HOST}": "Japan 1",
      "#{COUNT}": "5"
    },
    {
      "#{HOST}": "Japan 2",
      "#{COUNT}": "12"
    },
    {
      "#{HOST}": "Latvia",
      "#{COUNT}": "3"
    }
  ]
}
```

It can be seen that column names become macro names and selected rows become the values of these macros.

If it is not obvious how a column name would be transformed into a macro name, it is suggested to use column aliases like “COUNT(h2.host) AS count” in the example above.

In case a column name cannot be converted into a valid macro name, the discovery rule becomes not supported, with the error message detailing the offending column number. If additional help is desired, the obtained column names are provided under DebugLevel=4 in Zabbix server log file:

```
$ grep db.odbc.discovery /tmp/zabbix_server.log
...
23876:20150114:153410.856 In db_odbc_discovery() query:'SELECT h1.host,
```

```
COUNT(h2.host) FROM hosts h1 LEFT JOIN hosts h2 ON h1.hostid =
h2.proxy_hostid WHERE h1.status IN (5, 6) GROUP BY h1.host;'
23876:20150114:153410.860 db_odbc_discovery() column[1]:'host'
23876:20150114:153410.860 db_odbc_discovery() column[2]:'COUNT(h2.host)'
23876:20150114:153410.860 End of db_odbc_discovery():NOTSUPPORTED
23876:20150114:153410.860 Item [Zabbix
server:db.odbc.discovery[proxies,{ $DSN}]] error: Cannot convert column #2
name to macro.
```

Now that we understand how a SQL query is transformed into a JSON object, we can use {#HOST} macro in item prototypes:

The screenshot shows the 'Preprocessing' tab of an 'Item prototype' configuration page. The fields are as follows:

- 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: A table with columns Type, Interval, and Period. It contains one entry: Flexible Scheduling, 50s, 1-7,00:00-24:00. There is an 'Add' link below the table.
- History storage period: 90d
- Trend storage period: 365d
- Show value: As is, with a 'show value mappings' link.

Once discovery is performed, an item will be created for each proxy:

<input type="checkbox"/>	Wizard	Name	Triggers	Key ▲
<input type="checkbox"/>		Proxy discovery: Last access time of proxy Japan1		zabbix[proxy,Japan1,lastacce
<input type="checkbox"/>		Proxy discovery: Last access time of proxy Japan2		zabbix[proxy,Japan2,lastacce
<input type="checkbox"/>		Proxy discovery: Last access time of proxy Latvia		zabbix[proxy,Latvia,lastaccess

3.6 Discovery of Windows services

Windows service discovery is done in the same way as discovery of file systems. The key to use in the discovery rule is “service.discovery” and the following macros are supported for use in the [filter](#) and item/trigger/graph prototypes:

```
{#SERVICE.NAME}
{#SERVICE.DISPLAYNAME}
{#SERVICE.DESCRPTION}
{#SERVICE.STATE}
{#SERVICE.STATENAME}
{#SERVICE.PATH}
{#SERVICE.USER}
{#SERVICE.STARTUP}
{#SERVICE.STARTUPNAME}
```

Based on Windows service discovery you may create an item prototype like “service.info[`{#SERVICE.NAME}`,<param>]”, where *param* accepts the following values: *state*, *displayname*, *path*, *user*, *startup* or *description*. For example, to acquire the display name of a service you should use a “service.info[`{#SERVICE.NAME}`,displayname]” item. If *param* value is not specified (“service.info[`{#SERVICE.NAME}`]”), the default parameter *state* is used.

`{#SERVICE.STATE}` and `{#SERVICE.STATENAME}` macros return the same content, however, `{#SERVICE.STATE}` returns a numerical value (0-7), while `{#SERVICE.STATENAME}` returns text (*running*, *paused*, *start pending*, *pause pending*, *continue pending*, *stop pending*, *stopped* or *unknown*). The same applies to `{#SERVICE.STARTUP}` and `{#SERVICE.STARTUPNAME}`, where one returns a numerical value (0-4) while the other - text (*automatic*, *automatic delayed*, *manual*, *disabled*, *unknown*).

3.7 Setting up multiple LLD rules for the same item

Since Zabbix agent version 3.2 it is possible to alter low-level discovery item keys using “Alias” parameter in [zabbix_agentd.conf](#) file to enable configuration of several LLD rules for the same item.

3.8 Creating custom LLD rules

It is also possible to create a completely custom LLD rule, discovering any type of entities - for example, databases on a database server.

To do so, a custom item should be created that returns JSON, specifying found objects and optionally - some properties of them. The amount of macros per entity is not limited - while the built-in discovery rules return either one or two macros (for example, two for filesystem discovery), it is possible to return more.

The required JSON format is best illustrated with an example. Suppose we are running an old Zabbix 1.8 agent (one that does not support "vfs.fs.discovery"), but we still need to discover file systems. Here is a simple Perl script for Linux that discovers mounted file systems and outputs JSON, which includes both file system name and type. One way to use it would be as a UserParameter with key "vfs.fs.discovery_perl":

```
#!/usr/bin/perl

$first = 1;

print "{\n";
print "\t\"data\": [\n\n";

for (`cat /proc/mounts`)
{
    ($fsname, $fstype) = m/\S+ (\S+) (\S+)/;

    print "\t,\n" if not $first;
    $first = 0;

    print "\t{\n";
    print "\t\t\"{#FSNAME}\" : \"$fsname\", \n";
    print "\t\t\"{#FSTYPE}\" : \"$fstype\" \n";
    print "\t}\n";
}

print "\n\t]\n";
print "}\n";
```

Allowed symbols for LLD macro names are **0-9** , **A-Z** , **_** , **.**

Lowercase letters are not supported in the names.

An example of its output (reformatted for clarity) is shown below. JSON for custom discovery checks has to follow the same format.

```
{
  "data": [
```

```

{ "#FSNAME": "/",
  "#FSTYPE": "rootfs" },
{ "#FSNAME": "/sys",
  "#FSTYPE": "sysfs" },
{ "#FSNAME": "/proc",
  "#FSTYPE": "proc" },
{ "#FSNAME": "/dev",
  "#FSTYPE": "devtmpfs" },
{ "#FSNAME": "/dev/pts",
  "#FSTYPE": "devpts" },
{ "#FSNAME": "/lib/init/rw",
  "#FSTYPE": "tmpfs" },
{ "#FSNAME": "/dev/shm",
  "#FSTYPE": "tmpfs" },
{ "#FSNAME": "/home",
  "#FSTYPE": "ext3" },
{ "#FSNAME": "/tmp",
  "#FSTYPE": "ext3" },
{ "#FSNAME": "/usr",
  "#FSTYPE": "ext3" },
{ "#FSNAME": "/var",
  "#FSTYPE": "ext3" },
{ "#FSNAME": "/sys/fs/fuse/connections",
  "#FSTYPE": "fusectl" }
]
}

```

Then, in the discovery rule's "Filter" field, we could specify "{#FSTYPE}" as a macro and "rootfs|ext3" as a regular expression.

You don't have to use macro names FSNAME/FSTYPE with custom LLD rules, you are free to use whatever names you like.

3.9 Using LLD macros in user macro contexts

User macros [with context](#) can be used to accomplish more flexible thresholds in trigger expressions. Different thresholds may be defined on user macro level and then used in trigger constants depending on the discovered context. Discovered context appears when the [low-level discovery macros](#) used in the macros are resolved to real values.

To illustrate we can use data from the example above and assume that the following file systems will be discovered: /, /home, /tmp, /usr, /var.

We may define a free-disk-space trigger prototype for a host, where the threshold is expressed by a user macro with context:

```
{host:vfs.fs.size[#{FSNAME},pfree].last()}<{LOW_SPACE_LIMIT:#{FSNAME}}
```

Then add user macros:

- {LOW_SPACE_LIMIT} **10**
- {LOW_SPACE_LIMIT:/home} **20**
- {LOW_SPACE_LIMIT:/tmp} **50**

Now, once the file systems are discovered, events will be generated if /, /usr and /var filesystems have less than **10%** of free disk space, the /home filesystem - less than **20%** of free disk space or the /tmp filesystem - less than **50%** of free disk space.

LLD macros are not supported inside of user macro contexts in [trigger function parameters](#).

Last update: 2017/04/04 09:16 zh:manual:discovery:low_level_discovery https://www.zabbix.com/documentation/3.4/zh/manual/discovery/low_level_discovery?rev=1491297411

From: <https://www.zabbix.com/documentation/3.4/> - **Zabbix Documentation 3.4**

Permanent link: https://www.zabbix.com/documentation/3.4/zh/manual/discovery/low_level_discovery?rev=1491297411

Last update: **2017/04/04 09:16**

