

### 3 自动发现LLD

#### 概述

自动发现LLD提供了一种在计算机上为不同实体自动创建监控项，触发器和图形的方法。例如Zabbix可以在你的机器上自动开始监控文件系统或网络接口，而无需为每个文件系统或网络接口手动创建监控项。此外，可以配置Zabbix根据定期执行发现后的得到实际结果，来移除不需要的监控。

用户可以自己定义发现类型，只要它们遵循特定的JSON协议。

发现过程的一般架构如下。

首先，用户在“配置”→“模板”→“发现”列中创建一个发现规则。发现规则包括（1）发现必要实体（例如，文件系统或网络接口）的项和（2）应该根据该项的值创建的监控项，触发器和图形的原型

发现所需实体的项就像其他地方所看到的常规项一样：服务器server向Zabbix agent或者对应该项的其他类型的设置）查询该项的值agent以文本值进行响应。区别在于agent响应的值应该包含特定JSON格式的已发现实体的列表。虽然这个列表的详细信息仅对自定义发现检查来说很重要，但有必要知道返回的值包含宏 ->值对的列表。例如，项目“net.if.discovery”可能会返回两对“{#IFNAME}” ->“lo”和“{#IFNAME}” ->“eth0”

这些宏用于名称，键值和其他原型字段中，然后用接收到的值为每个发现的实体创建实际的监控项，触发器，图形甚至主机。请参阅使用LLD宏选项的完整列表。

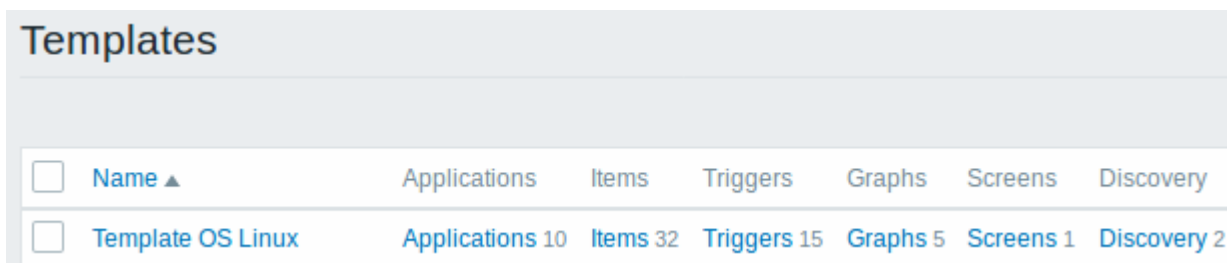
当服务器接收到已发现项的值时，它会查看宏->值对，每对都根据原型生成实际监控项，触发器和图形。在上面的“net.if.discovery”示例中，服务器将生成环路接口“lo”的一组监控项，触发器和图表，另一组用于界面“eth0”

#### 配置低级别发现LLD

我们来用文件系统发现为例子说明低级别发现LLD

请执行以下操作，配置发现：

- 进入：配置 → 模板
- 选择一个合适的模板的行点击发现



- 单击屏幕右上角的 创建发现规则
- 填写需要的详细信息

#### 发现规则

**发现规则** 选项卡包含常规发现规则属性:

The screenshot shows the 'Discovery rule' configuration page in Zabbix. The 'Filters' tab is active. The form contains the following fields and sections:

- Name:** Mounted filesystem discovery
- Type:** Zabbix agent
- Key:** vfs.fs.discovery
- Host interface:** 192.168.3.31 : 10050
- Update interval:** 1h
- Custom intervals:** A table with columns 'Type', 'Interval', and 'Period'. It contains one entry: 'Flexible Scheduling' with an interval of '50s' and a period of '1-7,00:00-2'. There is an 'Add' button below the table.
- Keep lost resources period:** 30d
- Description:** Discovery of file systems of different types as defined in global regular expression "File systems for discovery".
- Enabled:** A checked checkbox.
- Buttons:** 'Add' and 'Cancel' buttons at the bottom.

所有必填输入字段都标有红色星号。

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 <i>Zabbix agent</i> 或 <i>Zabbix agent</i> 主动) 文件系统发现。
键值	自2.0版以来, 许多平台上 <i>Zabbix agent</i> 中都内置了一个带有“vfs.fs.discovery”键的项目, (详情参见 <a href="#">支持项目键列表</a> ), 将返回一个JSON[] 包含计算机上存在的文件系统列表及其类型详情。
数据更新间隔 (秒)	此字段指定 <i>Zabbix</i> 执行发现的频率。 在开始刚刚设置文件系统发现时, 可以将其设置为一个小间隔, 但是一旦知道它可以工作之后, 可以将其设置为30分钟或更长时间, 因为文件系统通常不会经常更改。 从 <i>Zabbix 3.4.0</i> 起支持 <b>时间后缀</b> , 例如 30s[]1m[]2h[]1d[] 自 <i>Zabbix 3.4.0</i> 起支持 <b>用户宏</b> 。 注意: 如果设置为'0', 则不会轮询该项目。 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目。 注意对于现有发现规则, 可以通过点击 <b>立即检查按钮</b> 立即执行发现。
隔	您可以创建用于检查项目的自定义规则: <b>灵活</b> - 创建例外的 <b>更新间隔</b> (不同频次的间隔) <b>调度</b> - 创建自定义轮询调度。 有关详细信息, 请参阅 <a href="#">自定义时间间隔</a> 。 从 <i>Zabix 3.0.0</i> 起支持调度。
丢失资源保留期 (天)	
Description	输入一段描述
Enabled	如果选中, 规则将会处理该规则。

## 发现规则过滤器

**过滤器** 选项卡包括发现规则过滤器定义:

Parameter	Description
参数	描述
计算类型	<p>可以使用以下计算过滤器的选项：</p> <p><b>And</b> - 所有过滤器必须通过；</p> <p><b>Or</b> - 只需要一个过滤器通过就足够了；</p> <p><b>And/Or</b> - 使用具有不同的宏名称的 <i>And</i> ， 具有相同宏名称的 <i>Or</i> ；</p> <p><b>Custom expression</b> - 提供定义过滤器的自定义计算的可能性。 公式必须包含列表中的所有过滤器。 限255个符号。</p>
过滤器	<p>过滤器可以用来生成真实监控项，触发器，但是图表仅用于特定的文件系统。它期待一个 <a href="#">Perl Compatible Regular Expression (PCRE)</a> 例如，如果你只对 C:, D:, 和 E: 文件系统有想法，你可以把 <code>{#FSNAME}</code> 放进 “宏” 中 并将 <code>^C ^D ^E</code> 正则表达式放入 “正则表达式” 文本字段。使用 <code>{#FSTYPE}</code> 宏（例如 <code>^ext ^reiserfs</code> 的文件系统类型和使用 <code>{#FSDRIVETYPE}</code> 宏（例如 <code>fixed</code> 的驱动器类型（仅由 Windows agent 支持）也可以进行过滤。</p> <p>您可以在 “正则表达式” 字段中输入正则表达式或引用全局 <a href="#">正则表达式</a></p> <p>你可以用 <code>grep -E</code> 来测试一个正则表达式，例如：</p> <pre>for f in ext2 nfs reiserfs smbfs; do echo \$f   grep -E '^ext ^reiserfs'    echo "SKIP: \$f"; done</pre> <p><code>{#FSDRIVETYPE}</code> 宏从 Zabbix <b>3.0.0</b> 在 Windows 上开始支持。</p> <p>从 Zabbix <b>2.4.0</b> 开始支持定义多个过滤器。</p> <p>注意，如果一些来自过滤器在响应中丢失，那发现的实体将会被忽略。 \\ “过滤器” 下拉列表提供两个值，用于指定宏是与正则表达式匹配还是不匹配。</p>

如果要正确发现仅按大小写不同的文件系统名称，则必须将 MySQL 中的 Zabbix 数据库创建为区分大小写。在正则表达式中的错误或错字，应用在 LLD 规则中时可能会导致删除数以千计的配置元素，历史值和许多主机的活动。例如，错误的 “用于发现的文件系统” 正则表达式可能会导致删除数以千计的监控项，触发器，历史值和活动。不保留发现规则历史记录。

### 表格按钮

在表格底部的按钮会显示许多可用的操作。

	新增一个发现规则，此按钮仅可用于新的发现规则。
	更新发现规则的属性。此按钮仅适用于现有发现规则。
	在现有的发现规则的属性基础上创建另一个发现规则。
	立即根据发现规则执行发现。发现规则必须已存在。查看 <a href="#">详情</a> 。 注意 当立即执行发现时，配置缓存不会更新，因此结果并不代表发现规则配置中最新的改变。

<a href="#">Delete</a>	删除发现规则。
<a href="#">Cancel</a>	取消编辑发现规则属性。

## 监控项原型

一旦规则创建完成了，找到那条规则下的监控项，并点击“创建原型”来创建一个监控项原型。请注意在需要文件系统名称的情况下如何使用宏 `{#FSNAME}` 处理发现规则时，此宏将替换为发现的文件系统。

Item prototype **Preprocessing**

\* Name

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"/>

[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

低级别发现 **宏 and 用户 宏** 可能会被用在监控项原型配置和监控项值预处理 **参数**.

执行特定于上下文的低级别发现宏的转义，以便在正则表达式和XPath预处理参数中安全使用。

Attributes that are specific for item prototypes: 特定于监控项原型的属性:

参数	描述
全新的应用原型	你可能定义一个新的应用原型。 在应用原型中，你可以使用低级别发现宏，在发现后，将替换为实际值以创建特定于已发现实体的应用程序。 在 <a href="#">应用发现注意事项</a> 查看更详细的信息。
应用程序原型	从现有的应用程序原型选择。
创建已启用	如果选中，则监控项将以启用状态添加。 如果未选中，则该监控项将添加到已发现的实体，但处于禁用状态。

我们可以根据需求为每个文件系统指标创建许多监控项原型:

The screenshot shows the 'Item prototypes' configuration page in Zabbix. It displays a list of templates under the path 'All templates / Template OS Linux / Discovery list / Mounted filesystem discovery'. The list includes several templates for monitoring disk space on various filesystems, each with a checkbox, a name, a key, and an interval.

<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

## 触发器原型

我们创建触发器原型的方式和创建监控项原型的方式相似:

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"/>
<a href="#">Add</a>	

**Allow manual close**

**URL**

**Description**

**Create enabled**

特定于触发器原型的属性:

参数	描述
<i>创建已启用</i>	如果选中，则触发器将以启用状态添加。 如果未选中，则该触发器将添加到已发现的实体，但处于禁用状态。

当真实的触发器从原型创建，可能需要灵活地确定在表达式中用于比较的常量（在我们的示例中为'20'）。了解 [基于环境的用户宏](#) 如何有助于实现这种灵活性。

您也可以定义触发器原型之间的（从Zabbix 3.0开始支持[ ][You can define 依赖关系](#)（从Zabbix 3.0开始支持）。为此，转到 [依赖关系](#) 选项卡。触发器原型可能依赖于来自相同低级别发现[LLD]规则的另一个触发器原型或常规触发器。触发器原型可能不依赖于来自不同LLD规则的触发器原型或依赖于触发器原型创建的触发器。主机触发器原型不能依赖于来自模板的触发器。

### Trigger prototypes

[All templates / Template OS Linux](#) [Discovery list / Mounted filesystem discovery](#) [Item prototypes 5](#)

<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

## 图表原型

我们也能创建图表原型：

### Graph prototype Preview

\* Name

\* Width

\* Height

Graph type

Show legend

3D view

\* Items

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](#)

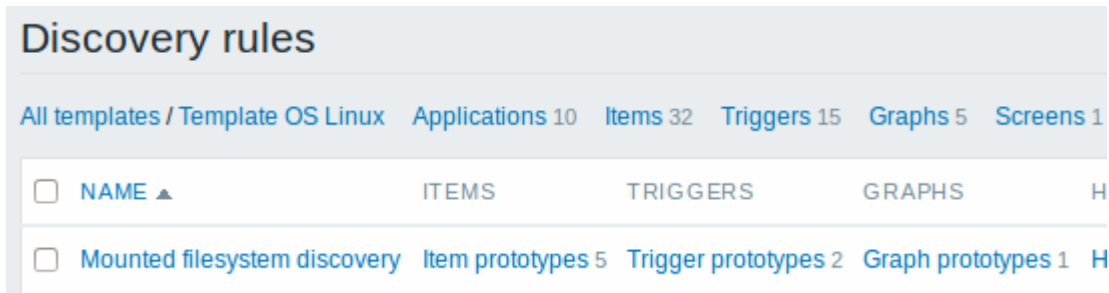
### Graph prototypes

[All templates / Template OS Linux](#) [Discovery list / Mounted filesystem discovery](#) [Item prototypes 5](#)

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

最后，我们像下面的展示的一样创建一个发现规则。有五个监控项原型，两个触发器原型，和一个图表原型。

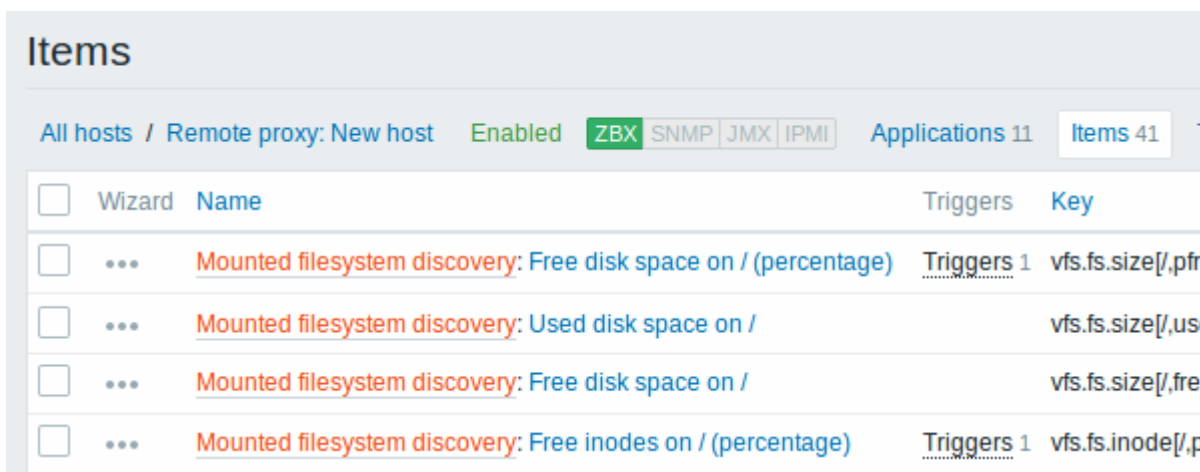




注意：有关配置[主机原型](#)的信息，请参阅虚拟机监视中有关主机原型配置的部分。

### 发现的实体

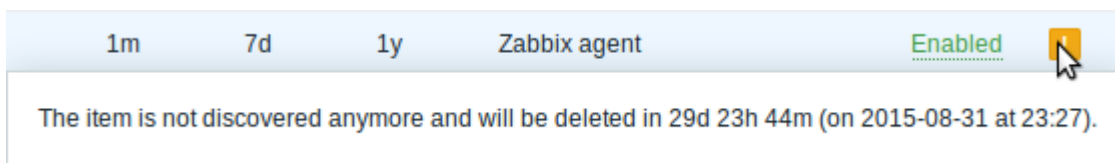
下面的屏幕截图说明了主机配置中发现的项目，触发器和图形的外观。发现的实体的前缀是橙色链接，指向它们来自的发现规则。



请注意，如果已存在具有相同唯一性条件的实体，则不会创建已发现的实体，例如，具有相同key或具有相同名称的图形的监控项。

如果一个发现的实体（文件系统，接口等）停止被发现（或是再也不通过过滤器），使用低级别发现规则创建的监控项（触发器和图表也相似）将会自动被删除。在这种情况下，在Keep lost resources period字段传递中定义日期之后，将删除监控项，触发器和图表。

当发现实体转变成“不再发现”状态，生命周期指示符显示在监控项列表中。将鼠标指针移到它上面，将显示一条消息，表示在删除项目之前剩余的天数。



如果实体已标记为删除但未在预期时间删除（已禁用的发现规则或项目主机），则下次处理发现规则时将删除这些实体。

如果在发现规则级别更改，则包含标记为删除的其他实体的实体将不会更新。例如，如果基于LLD的触发器包含标记为删除的监控项，则不会更新。

## Triggers

Group

All hosts / Remote proxy: New host Enabled **ZBX** SNMP JMX IPMI Applications 11 Items 41 T

<input type="checkbox"/>	Severity	Name ▲
<input type="checkbox"/>	Warning	<a href="#">Mounted filesystem discovery: Free disk space is less than 20% on volume /</a>
<input type="checkbox"/>	Warning	<a href="#">Mounted filesystem discovery: Free inodes is less than 20% on volume /</a>

## Graphs

Group

All hosts / Remote proxy: New host Enabled **ZBX** SNMP JMX IPMI Applications 11 Items 41 T

<input type="checkbox"/>	Name ▲
<input type="checkbox"/>	<a href="#">Template OS Linux: CPU jumps</a>
<input type="checkbox"/>	<a href="#">Template OS Linux: CPU load</a>
<input type="checkbox"/>	<a href="#">Template OS Linux: CPU utilization</a>
<input type="checkbox"/>	<a href="#">Mounted filesystem discovery: Disk space usage /</a>

### 其他类型的发现

以下部分提供了有关其他类型的开箱即用发现的更多详细信息和方法:

- discovery of [network interfaces](#);
- discovery of [CPUs and CPU cores](#);
- discovery of [SNMP OIDs](#);
- discovery of [JMX objects](#);
- discovery using [ODBC SQL queries](#);
- discovery of [Windows services](#);
- discovery of [host interfaces](#) in Zabbix.
- [网络接口](#)发现;
- [CPUs 和 CPU 核心](#)发现;
- [SNMP OIDs](#)发现;
- [JMX 对象](#)发现;
- 使用[ODBC SQL 查询](#)发现;
- [Windows services](#)发现;
- 在Zabbix中的[主机接口](#)发现

有关发现项的JSON格式的更多详细信息以及如何将自己的文件系统发现者实现为Perl脚本的示例, 请参阅[创建自定义LLD规则](#)

### 反馈值的数据限制

如果是直接来自Zabbix server那对低级别发现规则JSON数据没有限制, 因为反馈值是在没有存储在数据

库中的情况下处理的。对于定制的低级别发现规则也没有限制，但是，如果 如果要使用用户参数获取自定义LLD数据，则应用用户参数反馈值会有限制[512 KB]

如果数据必须通过Zabbix proxy则必须将此数据存储于数据库中，以便应用数据库限制，例如，在运行IBM DB2数据库的Zabbix proxy上应用2048字节。

## 同一监控项的多个LLD规则

自从Zabbix agent 3.2版本开始，就可以使用相同的发现监控项定义许多低级别发现规则了。

为此，你需要定义Alias agent 参数，在不同的发现规则中允许使用更改发现监控项密钥。例如，`vfs.fs.discovery[foo]`，`vfs.fs.discovery[bar]`等。

## 创建自定义LLD规则

也可以创建一个完整的自定义LLD规则，同时发现任何类型的实体——例如，在database server上的数据库。

为此，应创建一个反馈JSON的自定义项，指定找到的对象并可选——他们的一些属性。每个实体宏不受限制——虽然内置的发现规则反馈一个或两个宏（例如，两个用于文件系统发现），单反馈更多宏也是可能的。

这里有个例子可以最好地证明需要的JSON格式。假设我们正在运行一个接的Zabbix 1.8 agent不支持“`vfs.fs.discovery`”但我们仍旧需要发现文件系统。这有一个简单的Linux Perl脚本，可以发现挂载的文件系统，并输出JSON其中包括文件系统的名称和类型。一种使用它的方式是使用键“`vfs.fs.discovery_perl`”作为UserParameter

```
#!/usr/bin/perl

$first = 1;

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

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

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

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

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

```
print "}\n";
```

LLD宏名称的允许符号为 **0-9** , **A-Z** , **\_** , **.**

名称中不支持小写字母。

其输出示例（为清晰起见重新格式化）如下所示。 自定义发现检查的JSON必须遵循相同的格式。

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

然后，在发现规则中的“过滤器”部分，我们可以指定“`{#FSTYPE}`”作为一个宏，`“rootfs|ext3”` 作为一个正则表达式。

你不必要使用自定义LLD规则的宏名称 `FSNAME/FSTYPE` 你可以使用任何你喜欢的名称。

需要注意的是，如果使用一个用户参数，反馈值限制到512 KB 更多信息，请参考[LLD反馈值的数据限制](#)。

## 在用户宏环境中使用LLD宏

环境中的用户宏可用于在触发器表达式中实现更灵活的阈值。不同的阈值 可以在用户宏级别上定义不同的阈值，然后根据发现的环境将其用于触发器常量。当宏中使用的 [低级别发现宏](#) 被解析为实际值时，将显示已发现的环境。

为了证明我们可以使用上述的例子中的数据， 假设将发现以下文件系统： `/`, `/home`, `/tmp`, `/usr`, `/var`。

我们可以为主机定义一个自由磁盘空间触发器原型，其中阈值由具有发现环境的用户宏表示：

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

然后新增用户宏：

- `{$LOW_SPACE_LIMIT} 10`
- `{$LOW_SPACE_LIMIT:/home} 20`
- `{$LOW_SPACE_LIMIT:/tmp} 50`

现在，一旦文件系统被发现了，如果/, /usr and /var文件系统有少于**10%**的自由磁盘空间， /home文件系统——少于**20%**的自由磁盘空间或/tmp文件系统——少于**50%**的自由磁盘空间，将生成事件。

From:

<https://www.zabbix.com/documentation/5.0/> - **Zabbix Documentation 5.0**

Permanent link:

[https://www.zabbix.com/documentation/5.0/zh/manual/discovery/low\\_level\\_discovery](https://www.zabbix.com/documentation/5.0/zh/manual/discovery/low_level_discovery)

Last update: **2021/02/14 11:53**

