LLD OVERRIDES

A BRIEF INTRODUCTION TO LLD OVERRIDES
WHAT ARE OVERRIDES?

- Introduced in version 5.0
- LLD overrides allow us to set rules to modify the list of item, trigger, graph and host prototypes or their attributes for discovered objects that meet given criteria.
- LLD Filters, however, filters out the entity completely, that’s why we need overrides!
WHERE TO FIND OVERRIDES?

✔ Navigate to your discovery rule

Switch to Overrides tab
**HOW TO CONFIGURE OVERRIDES?**

- Click on Add in the Overrides block. To edit an existing override, click on the override name.
- Create a name for the override.
- Configure your override filters.
- To configure details of a new operation, click on Add within the form of an override. To edit an existing operation, click on Edit next to the operation.
- All mandatory fields are marked with a red asterisk.
USE CASES

☐ You wish to:

• Monitor Windows services with a few exceptions
• Monitor Partitions with different item update intervals
• Assign different trigger severities for your DB tables
• Create item prototypes in disabled state for later review
RUNNING THE DISCOVERY

- Create a discovery rule or assign a template: Linux filesystems by zabbix agent

- Manually execute the Mounted filesystem discovery rule

- Take note of the discovered items and triggers
RUNNING THE DISCOVERY

- You now have items created by the discovery rule
- Items created by LLD rules will have a prefix containing the parent LLD
- Item names expand LLD macro {#FSNAME}
RUNNING THE DISCOVERY

- JSON created by Zabbix

```json
[{"{#FSNAME}"":"/sys","{#FSTYPE}"":"sysfs"},{"{#FSNAME}"":"/proc","{#FSTYPE}"":"proc"},{"{#FSNAME}"":"/dev","{#FSTYPE}"":"devtmpfs"},{"{#FSNAME}"":"/sys/kernel/security","{#FSTYPE}"":"securityfs"}, ...]
```

- Zabbix_get utility or test form within frontend can be used to extract the JSON created
RUNNING THE DISCOVERY

- Two filesystems are found “/” and “/boot”
- Let's see what happens if additional filesystems appear on our host

```
# mkdir /lldfs
# mount --bind /var/log/zabbix/ /lldfs/
[root@localhost /]# mount | grep lldfs
/dev/mapper/cl-root on /lldfs type xfs (rw,relatime,attr2,inode64,noquota)
```
- Execute the discovery rule again
RUNNING THE DISCOVERY

Our new dummy filesystem is now monitored automatically
RUNNING THE DISCOVERY

✔️ If we were to use a filter, our items would simply be removed/not created

✔️ Our template filters reference user defined macros on a template level

✔️ We are interested in filtering out a filesystem name
RUNNING THE DISCOVERY

- We will add "lldfs" to macro value
- Reload configuration cache
- Run LLD rule once more
RUNNING THE DISCOVERY

- Note that the items filtered out are still present due to “Keep lost resources” setting

- Items will not be removed instantly, unless 0 value is specified

- what if we do not wish to remove ALL of the elements related to the dummy filesystem? Or what if we wish to change some of them - change update intervals, create some of the triggers in a disabled state, change trigger severities
**OVERRIDE!**

Add an override to Mounted filesystem discovery LLD

- Edit the discovery rule, create first override: “Custom update interval on /boot”
- Use a filter for the filesystem name which matches /boot
- Change the update interval to 1h for item prototypes containing the word “inodes”
- Note that you cannot add an override on a host level, if the discovery rule is linked from a template
OVERRIDE!

- Reload your configuration cache
- Take a look at your discovered items:

<table>
<thead>
<tr>
<th>Wizard</th>
<th>Name</th>
<th>Interval</th>
<th>History</th>
<th>Trends</th>
<th>Type</th>
<th>Applications</th>
<th>Status</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mounted filesystem discovery: Free inodes in %</td>
<td>vfs.fs.inode[.free]</td>
<td>1m</td>
<td>7d</td>
<td>365d</td>
<td>Zabbix agent</td>
<td>Filesystem</td>
<td>Enabled</td>
</tr>
<tr>
<td>Mounted filesystem discovery: Space utilization</td>
<td>vfs.fs.size[.used]</td>
<td>1m</td>
<td>7d</td>
<td>365d</td>
<td>Zabbix agent</td>
<td>Filesystem</td>
<td>Enabled</td>
</tr>
<tr>
<td>Mounted filesystem discovery: Total space</td>
<td>vfs.fs.size[.total]</td>
<td>1m</td>
<td>7d</td>
<td>365d</td>
<td>Zabbix agent</td>
<td>Filesystem</td>
<td>Enabled</td>
</tr>
<tr>
<td>Mounted filesystem discovery: Used space</td>
<td>vfs.fs.size[.used]</td>
<td>1m</td>
<td>7d</td>
<td>365d</td>
<td>Zabbix agent</td>
<td>Filesystem</td>
<td>Enabled</td>
</tr>
<tr>
<td>Mounted filesystem discovery: /boot Free inodes in %</td>
<td>vfs.fs.inode[.boot.free]</td>
<td>1h</td>
<td>7d</td>
<td>365d</td>
<td>Zabbix agent</td>
<td>Filesystem</td>
<td>Enabled</td>
</tr>
</tbody>
</table>

- The item “/boot: Free inodes in %” is now updated every 1h
**OVERRIDE!**

Let’s add another override – trigger prototype override

- Add the 2nd override: “/boot inode trigger override”
- For inodes triggers change the severity level to “Information”
- Specify to create these triggers in a disabled state
- And assign a tag “Environment”: “Meetup”
OVERRIDE!

Check if all of the overrides apply to already created elements (triggers):

- Has the severity been changed?
- Are the triggers properly tagged?
- Has their status changed from enabled to disabled?
OVERRIDE!

- Triggers that were created before, are still enabled
- Both triggers have changed severities

<table>
<thead>
<tr>
<th>Severity</th>
<th>Value</th>
<th>Name</th>
<th>Operational data</th>
<th>Expression</th>
<th>Status</th>
<th>Info</th>
<th>Tags</th>
</tr>
</thead>
<tbody>
<tr>
<td>Average</td>
<td>OK</td>
<td>Mounted filesystem discovery: / Running out of free inodes (free &lt; $[VFS.FS.INODE.PFREEMIN.CRIT(%)]%)}</td>
<td>Free inodes (ITEM LASTVALUE)</td>
<td>(Zabbix server/vfs fs_inode/ pfree min(5m) &lt; $[VFS.FS.INODE.PFREEMIN.CRIT(%)]%)</td>
<td>Enabled</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Warning</td>
<td>OK</td>
<td>Mounted filesystem discovery: / Running out of free inodes (free &lt; $[VFS.FS.INODE.PFREEMIN.WARN(%)]%)}</td>
<td>Free inodes (ITEM LASTVALUE)</td>
<td>(Zabbix server/vfs fs_inode/ pfree min(5m) &lt; $[VFS.FS.INODE.PFREEMIN.WARN(%)]%)</td>
<td>Enabled</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Information</td>
<td>OK</td>
<td>Mounted filesystem discovery: / Running out of free inodes (free &lt; $[VFS.FS.INODE.PFREEMIN.CRIT(%)]%)}</td>
<td>Free inodes (ITEM LASTVALUE)</td>
<td>(Zabbix server/vfs fs_inode/ pfree min(5m) &lt; $[VFS.FS.INODE.PFREEMIN.CRIT(%)]%)</td>
<td>Enabled</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Information</td>
<td>OK</td>
<td>Mounted filesystem discovery: / Running out of free inodes (free &lt; $[VFS.FS.INODE.PFREEMIN.WARN(%)]%)}</td>
<td>Free inodes (ITEM LASTVALUE)</td>
<td>(Zabbix server/vfs fs_inode/ pfree min(5m) &lt; $[VFS.FS.INODE.PFREEMIN.WARN(%)]%)</td>
<td>Enabled</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
STOP PROCESSING

- On `/boot` inode trigger discovery select “Stop processing” if filter matches
- Add a third override: “Stop discovery of filesystems if no boot partition”
- Filter the override by `{#FSNAME} = /`
- If the filter matches, “/” FS item discovery will not take place
- Reload the configuration cache and rerun the discovery
STOP PROCESSING

- Let's see what would've change if we would not use “Stop processing”

<table>
<thead>
<tr>
<th>Overrides</th>
<th>Name</th>
<th>Stop processing</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.</td>
<td>Custom update interval on /boot</td>
<td>No</td>
<td>Remove</td>
</tr>
<tr>
<td>2.</td>
<td>/boot inode trigger override</td>
<td>No</td>
<td>Remove</td>
</tr>
<tr>
<td>3.</td>
<td>Stop discovery of filesystems if no boot partition</td>
<td>No</td>
<td>Remove</td>
</tr>
</tbody>
</table>

- In this case, all of our filesystem items will not be discovered

<table>
<thead>
<tr>
<th>Wizard</th>
<th>Name</th>
<th>Triggers</th>
<th>Key</th>
<th>Internal</th>
<th>History</th>
<th>Trends</th>
<th>Type</th>
<th>Applications</th>
<th>Status</th>
<th>Info</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Mounted filesystem discovery. /boot: Free inodes</td>
<td>Triggers: 2</td>
<td>vfs.fs.inode(boot,free)</td>
<td>1h</td>
<td>7d</td>
<td>365d</td>
<td>Zabbix agent</td>
<td>Filesystem</td>
<td>Enabled</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>Mounted filesystem discovery. /boot: Space utilization</td>
<td>Triggers: 2</td>
<td>vfs.fs.size(boot.pused)</td>
<td>1m</td>
<td>7d</td>
<td>365d</td>
<td>Zabbix agent</td>
<td>Filesystem</td>
<td>Enabled</td>
<td>2</td>
</tr>
<tr>
<td></td>
<td>Mounted filesystem discovery. /boot: Total space</td>
<td>Triggers: 2</td>
<td>vfs.fs.size(boot.total)</td>
<td>1m</td>
<td>7d</td>
<td>365d</td>
<td>Zabbix agent</td>
<td>Filesystem</td>
<td>Enabled</td>
<td>3</td>
</tr>
<tr>
<td></td>
<td>Mounted filesystem discovery. /boot: Used space</td>
<td>Triggers: 2</td>
<td>vfs.fs.size(boot.used)</td>
<td>1m</td>
<td>7d</td>
<td>365d</td>
<td>Zabbix agent</td>
<td>Filesystem</td>
<td>Enabled</td>
<td>4</td>
</tr>
<tr>
<td></td>
<td>Mounted filesystem discovery. /dev: Free inodes</td>
<td>Triggers: 2</td>
<td>vfs.fs.inode(devfs,free)</td>
<td>1m</td>
<td>7d</td>
<td>365d</td>
<td>Zabbix agent</td>
<td>Filesystem</td>
<td>Enabled</td>
<td>5</td>
</tr>
<tr>
<td></td>
<td>Mounted filesystem discovery. /dev: Space utilization</td>
<td>Triggers: 2</td>
<td>vfs.fs.size(devfs.pused)</td>
<td>1m</td>
<td>7d</td>
<td>365d</td>
<td>Zabbix agent</td>
<td>Filesystem</td>
<td>Enabled</td>
<td>6</td>
</tr>
<tr>
<td></td>
<td>Mounted filesystem discovery. /dev: Total space</td>
<td>Triggers: 2</td>
<td>vfs.fs.size(devfs.total)</td>
<td>1m</td>
<td>7d</td>
<td>365d</td>
<td>Zabbix agent</td>
<td>Filesystem</td>
<td>Enabled</td>
<td>7</td>
</tr>
<tr>
<td></td>
<td>Mounted filesystem discovery. /dev: Used space</td>
<td>Triggers: 2</td>
<td>vfs.fs.size(devfs.used)</td>
<td>1m</td>
<td>7d</td>
<td>365d</td>
<td>Zabbix agent</td>
<td>Filesystem</td>
<td>Enabled</td>
<td>8</td>
</tr>
</tbody>
</table>
COMMON MISCONFIGURATION

☑️ Stop processing if filter matches usage:

<table>
<thead>
<tr>
<th>Name</th>
<th>Stop processing</th>
<th>Filter</th>
<th>(Operations) Object</th>
<th>(Operations) Condition</th>
<th>(Operations) Severity</th>
</tr>
</thead>
<tbody>
<tr>
<td>Override1</td>
<td>YES</td>
<td>{{#FSNAME}} matches /boot</td>
<td>Trigger prototype</td>
<td>contains “Disk space is low”</td>
<td>Disaster</td>
</tr>
<tr>
<td>Override2</td>
<td>YES</td>
<td>{{#FSNAME}} matches /</td>
<td>Trigger prototype</td>
<td>contains “Disk space is low”</td>
<td>Information</td>
</tr>
</tbody>
</table>

```
[  { "{{#FSNAME}}":"/","{{FSTYPE}}":"ext4"   },
  { "{{FSNAME}}":"/boot","{FSTYPE}":"ext4"   }]
```
NOTES

☐ The option “Create enabled” is affecting only the newly created entities.

☐ Override order is important

☐ Overrides cannot be created on host level if LLD rule is assigned by template

☐ Typos. Always check for typos.
GOOD LUCK DISCOVERING

THANK YOU!