Going Down!

Using Low-Level Discoveries in practice

Layout by orngjce223, CC-BY

Who am I

Raymond Kuiper

- Infrastructure Specialist @ Competa IT
- Zabbix fan since 2006
- zbxtutorials.org
- NLZGG – in Dutch Zabbix user group

What is LLD?

“Low-level discovery provides a way to automatically create items, triggers, and graphs for different entities on a computer.”

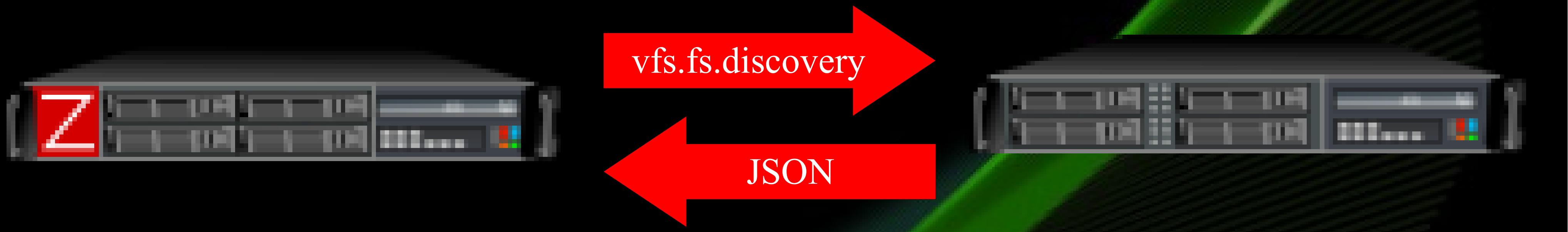
- the Zabbix manual

What is LLD?

A Low-Level Discovery rule uses:

- a discovery item that returns the discovery data in JSON formated
- that discovery data on prototypes to create new items triggers and graphs.

What is LLD?

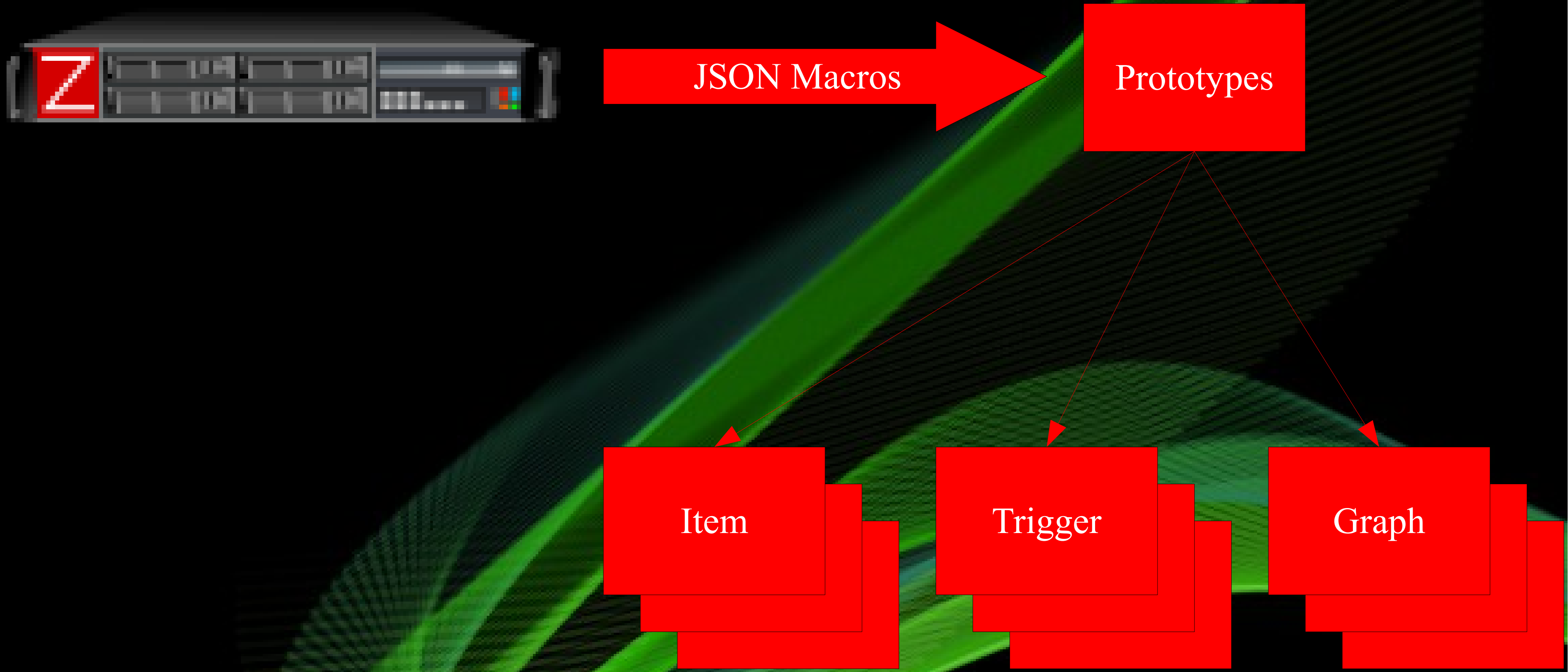


```
{
 "data":[

 { "{#FSNAME}":"\/",                          "{#FSTYPE}":"ext3"     },
 { "{#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"  }

 ]
```

What is LLD?



JSON Macros

Prototypes

Item

Trigger

Graph

What is LLD?

These macros can be used for

item prototypes in:

- names
- keys
- SNMP OIDs
- calculated item formulas
- SSH and Telnet scripts
- database monitor item parameters

trigger prototypes in:

- names
- expressions (when referencing an item key prototype)

graph prototypes in:

- names

What is LLD?

The item used for discovery can be ***any*** item as long as it outputs the JSON format Zabbix expects.

What is LLD?

Items, triggers and graphs that are no longer discovered are cleaned up after a predefined period of time.

Native Zabbix Agent LLD

The Zabbix agent has two types of LLD available by default:

- Filesystem discovery (vfs.fs.discovery)
- Network interface discovery (net.if.discovery)

Native Zabbix Agent LLD

Filesystem discovery (vfs.fs.discovery) returns these macros:

- {#FSNAME} - Mount location or drive letter
- {#FSTYPE} - Filesystem type (e.g. ext4, vfat)

Native Zabbix Agent LLD

Network interface discovery (net.if.discovery) returns this macro:

- {#IFNAME} – Interface name (e.g. eth0, lo)

Native SNMP LLD

The SNMP Discovery:

- can use any SNMP OID as a discovery item (uses an SNMP walk)

- will return these two macros:
  {#SNMPINDEX} – the last part of the OID discovered (after the “.”)
  {#SNMPVALUE} – the value of the OID discovered

Native SNMP LLD

SNMP Walk:

```
$ 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
```

Zabbix Discovery of IF-MIB::ifDescr:

```
{#SNMPINDEX} -> 1   {#SNMPVALUE} -> WAN
{#SNMPINDEX} -> 2   {#SNMPVALUE} -> LAN1
{#SNMPINDEX} -> 3   {#SNMPVALUE} -> LAN2
```

LLD Filters

You can define ***one*** regex filter per discovery rule to match macro values used for populating prototypes.

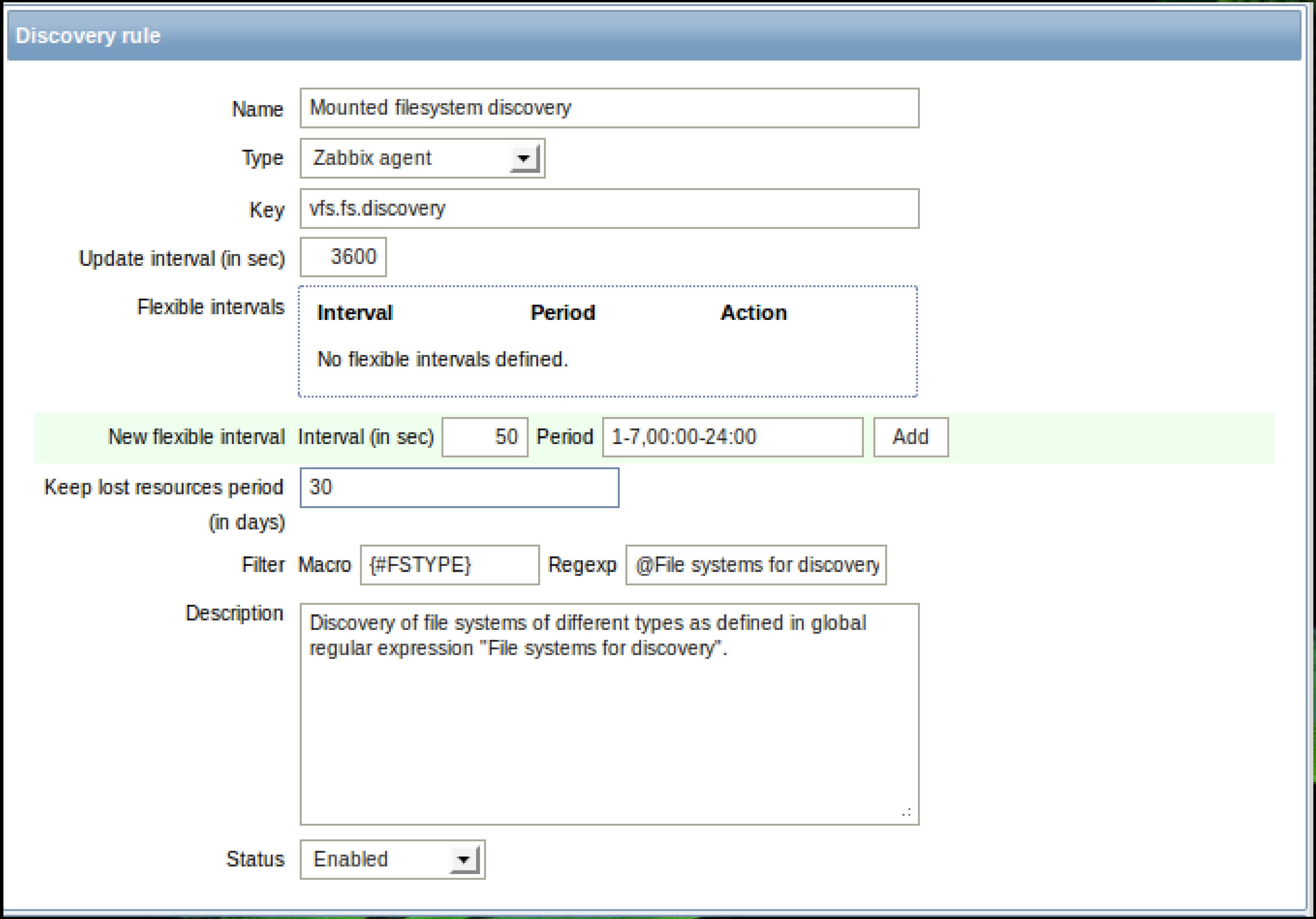
For example:

```
^(btrfs|ext2|ext3|ext4|jfs|reiser|xfs|ffs|ufs|jfs|jfs2|vxfs|hfs|ntfs|fat32)$
```

Ok, so what can I do with this?

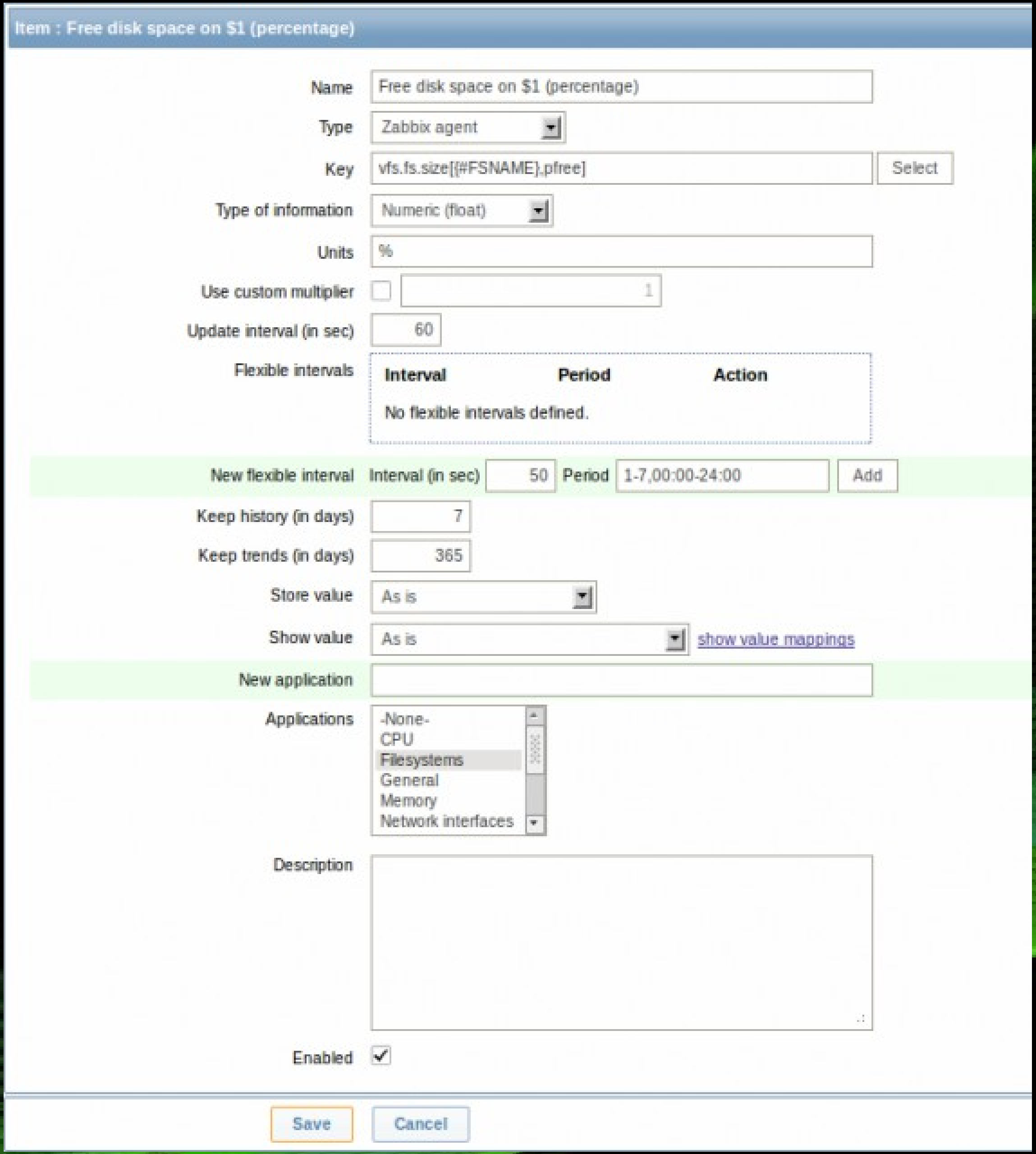
Layout by orngjce223, CC-BY

Case 1: FS discovery



Name	Mounted filesystem discovery
Type	Zabbix agent
Key	vfs.fs.discovery
Update interval (in sec)	3600
Flexible intervals	**Interval** **Period** **Action** No flexible intervals defined.
New flexible interval	Interval (in sec) 50 Period 1-7,00:00-24:00 Add
Keep lost resources period (in days)	30
Filter	Macro {#FSTYPE} Regexp @File systems for discovery
Description	Discovery of file systems of different types as defined in global regular expression "File systems for discovery".
Status	Enabled

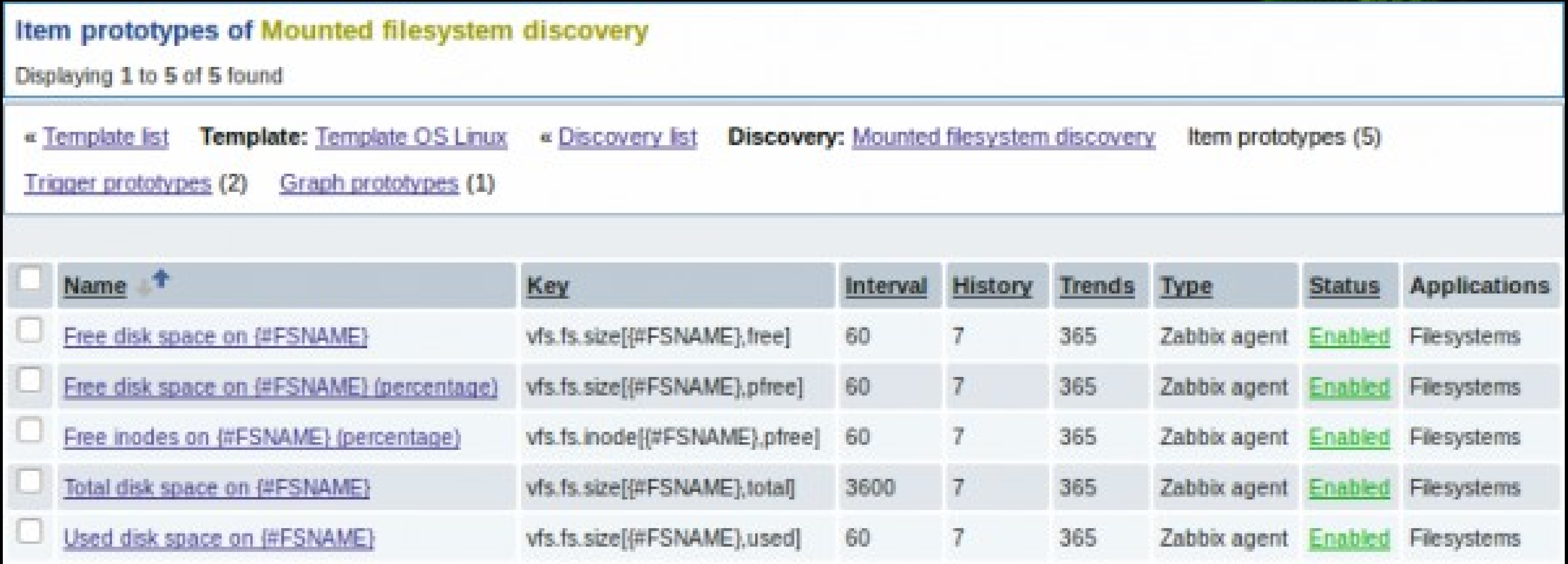
Case 1: FS discovery



Item : Free disk space on $1 (percentage)

Name	Free disk space on $1 (percentage)
Type	Zabbix agent
Key	vfs.fs.size[{#FSNAME},pfree]
Type of information	Numeric (float)
Units	%
Use custom multiplier	□ 1
Update interval (in sec)	60
Flexible intervals	Interval Period Action No flexible intervals defined.
New flexible interval	Interval (in sec) 50 Period 1-7,00:00-24:00
Keep history (in days)	7
Keep trends (in days)	365
Store value	As is
Show value	As is
New application	
Applications	-None- CPU Filesystems General Memory Network interfaces
Description	
Enabled	☑

Case 1: FS discovery



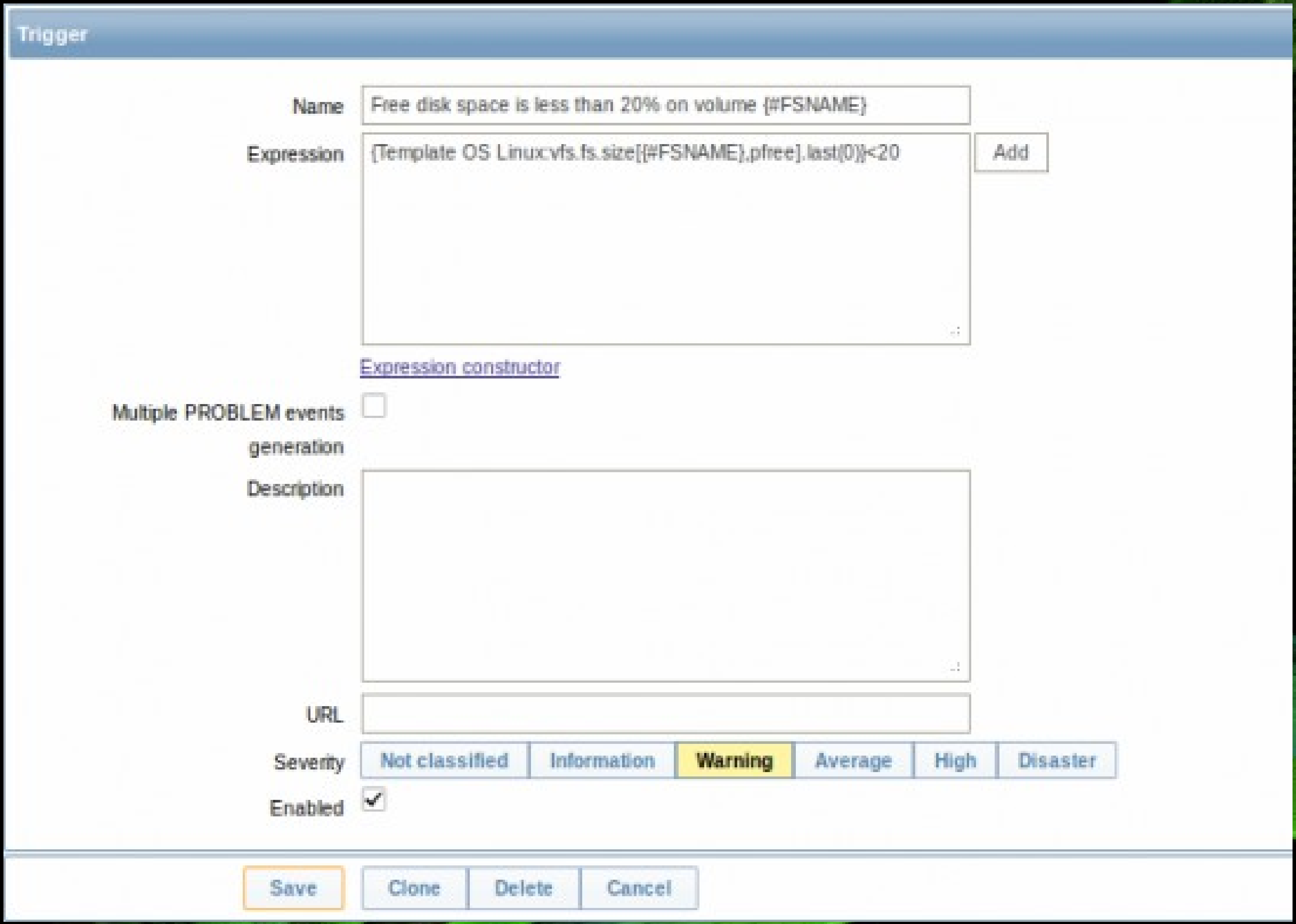
Item prototypes of Mounted filesystem discovery

Displaying **1** to **5** of **5** found

« Template list **Template:** Template OS Linux « Discovery list **Discovery:** Mounted filesystem discovery Item prototypes (5) Trigger prototypes (2) Graph prototypes (1)

□	Name	Key	Interval	History	Trends	Type	Status	Applications
□	Free disk space on {#FSNAME}	vfs.fs.size[{#FSNAME},free]	60	7	365	Zabbix agent	Enabled	Filesystems
□	Free disk space on {#FSNAME} (percentage)	vfs.fs.size[{#FSNAME},pfree]	60	7	365	Zabbix agent	Enabled	Filesystems
□	Free inodes on {#FSNAME} (percentage)	vfs.fs.inode[{#FSNAME},pfree]	60	7	365	Zabbix agent	Enabled	Filesystems
□	Total disk space on {#FSNAME}	vfs.fs.size[{#FSNAME},total]	3600	7	365	Zabbix agent	Enabled	Filesystems
□	Used disk space on {#FSNAME}	vfs.fs.size[{#FSNAME},used]	60	7	365	Zabbix agent	Enabled	Filesystems

Case 1: FS discovery



Trigger

Name: Free disk space is less than 20% on volume {#FSNAME}

Expression: {Template OS Linux:vfs.fs.size[{#FSNAME},pfree].last(0)}<20

Multiple PROBLEM events generation: □

Description:

URL:

Severity: Not classified | Information | [✓ Warning] | Average | High | Disaster

Enabled: ☑

Case 2: Windows Services

Need:

Activate triggers for services that are auto started but no longer running.

Case 2: Windows Services

Zabbix Agent has no capability of discovering Windows services

ZBXNEXT-1368
(Please vote!)

Case 2: Windows Services

My Solution:

Powershell script to grab autostart services from WMI and return their attributes as LLD macros.

(https://raw.github.com/q1x/zabbix-templates/master/service-discovery/servdisc.ps1)

Case 2: Windows Services

{#SERVICENAME}	The name of the Windows service
{#SERVICEDISPLAY}	The displayname of the Windows service
{#SERVICESTATE}	The state of the Windows service
{#SERVICEDESC}	The Windows service description

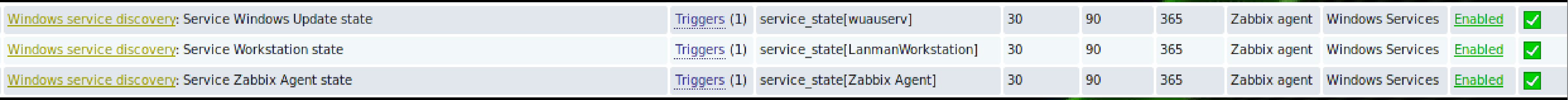
Case 2: Windows Services

Template does a discovery automatically started services currently in the running state.

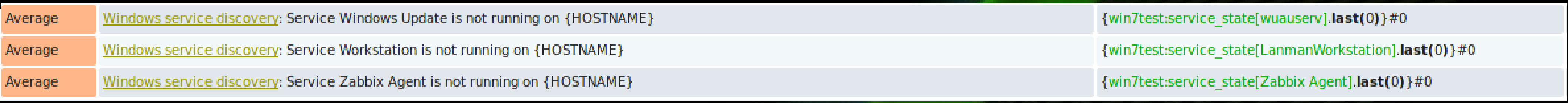
It filters the {#SERVICESTATE} macro for the string "Running".

(https://raw.github.com/q1x/zabbix-templates/master/service-discovery/Template_Windows_Service_Discovery.xml)

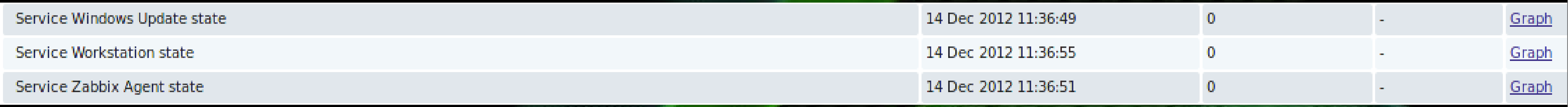
Case 2: Windows Services



Windows service discovery: Service Windows Update state	Triggers (1)	service_state[wuauserv]	30	90	365	Zabbix agent	Windows Services	Enabled	✓
Windows service discovery: Service Workstation state	Triggers (1)	service_state[LanmanWorkstation]	30	90	365	Zabbix agent	Windows Services	Enabled	✓
Windows service discovery: Service Zabbix Agent state	Triggers (1)	service_state[Zabbix Agent]	30	90	365	Zabbix agent	Windows Services	Enabled	✓



Average	Windows service discovery: Service Windows Update is not running on {HOSTNAME}	{win7test:service_state[wuauserv].**last(**0**)**}#0
Average	Windows service discovery: Service Workstation is not running on {HOSTNAME}	{win7test:service_state[LanmanWorkstation].**last(**0**)**}#0
Average	Windows service discovery: Service Zabbix Agent is not running on {HOSTNAME}	{win7test:service_state[Zabbix Agent].**last(**0**)**}#0



Service Windows Update state	14 Dec 2012 11:36:49	0	-	Graph
Service Workstation state	14 Dec 2012 11:36:55	0	-	Graph
Service Zabbix Agent state	14 Dec 2012 11:36:51	0	-	Graph

Layout by orngjce223, CC-BY

Case 3: Linux processes

Question in #zabbix:

“How to monitor CPU usage of separate processes under Linux?”

Case 3: Linux processes

Solution: Template and 2 custom user parameters

- ps.discovery - Returns a list of monitorable processes
- proc.cpu[*] - Calculates CPU usage

(https://github.com/q1x/zabbix-templates/tree/master/process-discovery)

Case 3: Linux processes

ps.discovery returns:

{#PSNAME} - The name of the found process
{#PSUSER} - The user account running the process

Case 3: Linux processes

Template includes items for the number of processes and the memory usage of each process.

(https://raw.github.com/q1x/zabbix-templates/master/service-discovery/Template_Windows_Service_Discovery.xml)

Case 3: Linux processes

<disclaimer>
!!! Highly experimental, use at your own risk !!!
</disclaimer>

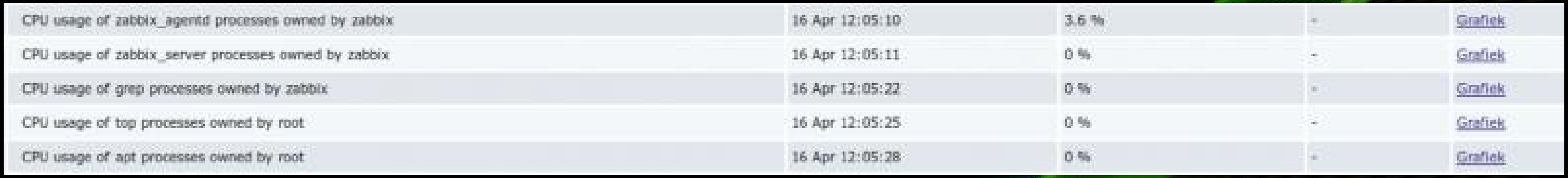
Case 3: Linux processes

Case 3: Linux processes

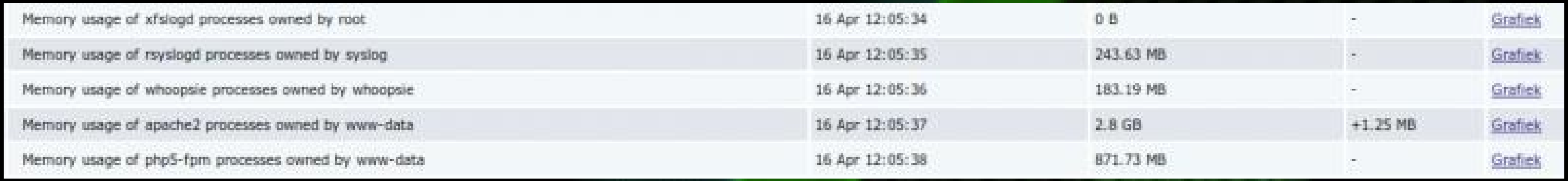
Use filters to limit the number of created items!

Item update interval will hammer the agent!

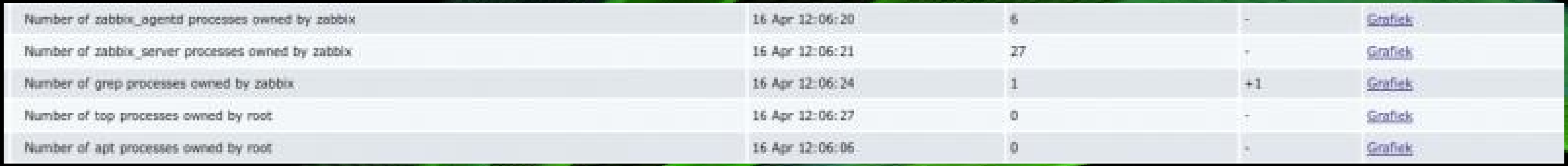
Case 3: Linux processes



CPU usage of zabbix_agentd processes owned by zabbix	16 Apr 12:05:10	3.6 %	-	Grafiek
CPU usage of zabbix_server processes owned by zabbix	16 Apr 12:05:11	0 %	-	Grafiek
CPU usage of grep processes owned by zabbix	16 Apr 12:05:22	0 %	-	Grafiek
CPU usage of top processes owned by root	16 Apr 12:05:25	0 %	-	Grafiek
CPU usage of apt processes owned by root	16 Apr 12:05:28	0 %	-	Grafiek

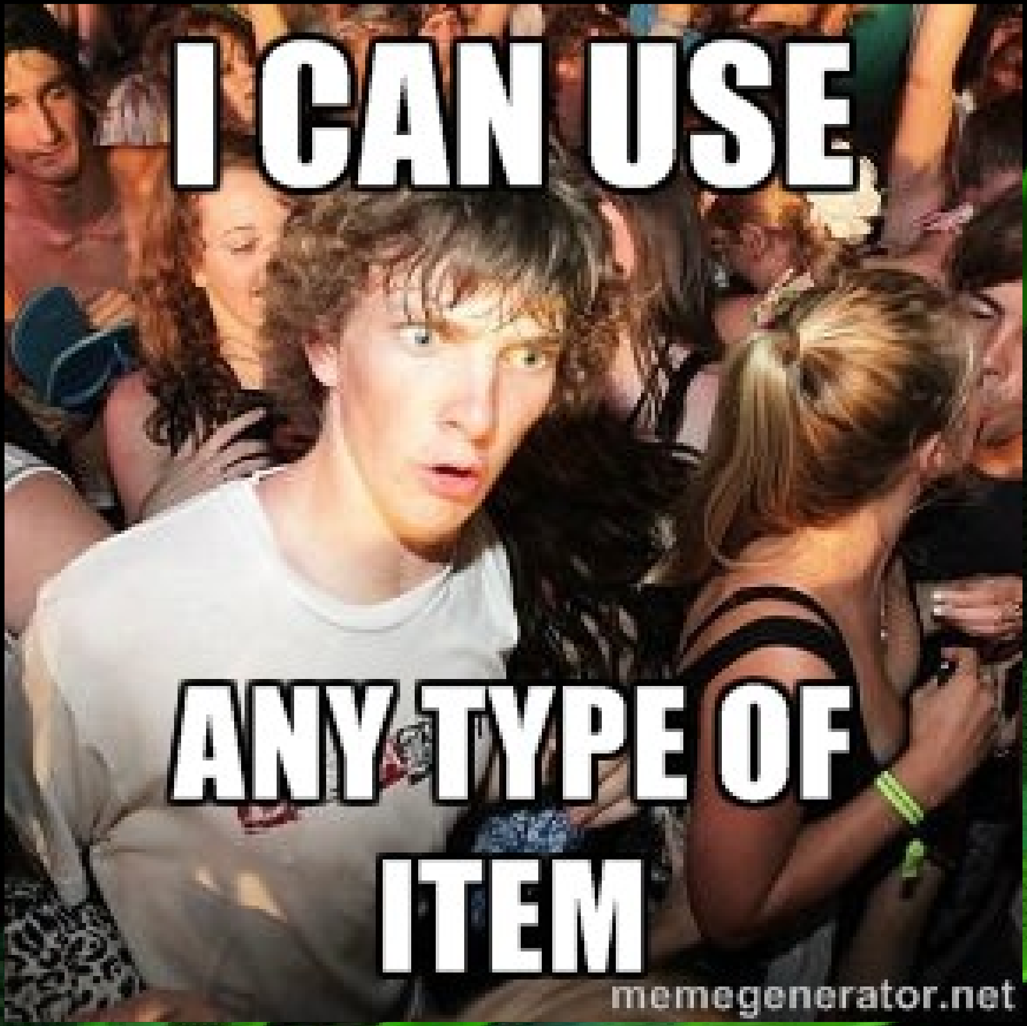


Memory usage of xfslogd processes owned by root	16 Apr 12:05:34	0 B	-	Grafiek
Memory usage of rsyslogd processes owned by syslog	16 Apr 12:05:35	243.63 MB	-	Grafiek
Memory usage of whoopsie processes owned by whoopsie	16 Apr 12:05:36	183.19 MB	-	Grafiek
Memory usage of apache2 processes owned by www-data	16 Apr 12:05:37	2.8 GB	+1.25 MB	Grafiek
Memory usage of php5-fpm processes owned by www-data	16 Apr 12:05:38	871.73 MB	-	Grafiek



Number of zabbix_agentd processes owned by zabbix	16 Apr 12:06:20	6	-	Grafiek
Number of zabbix_server processes owned by zabbix	16 Apr 12:06:21	27	-	Grafiek
Number of grep processes owned by zabbix	16 Apr 12:06:24	1	+1	Grafiek
Number of top processes owned by root	16 Apr 12:06:27	0	-	Grafiek
Number of apt processes owned by root	16 Apr 12:06:06	0	-	Grafiek

Sudden Realization



Trapper Discovery

Use zabbix_sender to send new 'discovered' items to Zabbix.

Allows for a very dynamic set of items

Case 4: NetFlow

Could be ideal for things like syslog, snmp traps or netflow!

(ZBX-6315 : LLD triggers are deleted immediately if not discovered anymore)

Case 4: NetFlow

"NetFlow is a network protocol developed by Cisco Systems for collecting IP traffic information. NetFlow has become an industry standard for traffic monitoring and is supported on various platforms."

- Wikipedia

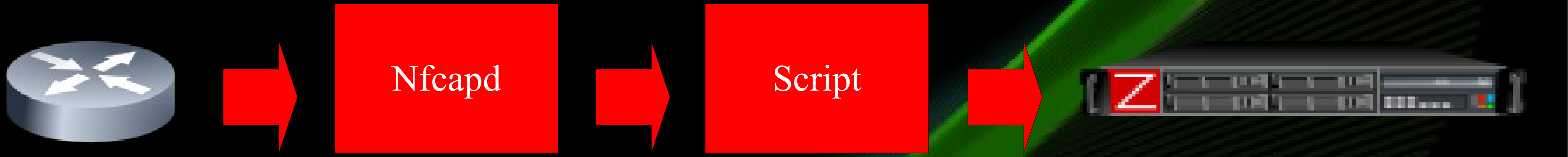
Case 4: NetFlow

Case 4: NetFlow

Using Ncapd and scripting Nfdump and Zabbix_sender, we can push Netflow data to Zabbix.

(http://http://nfdump.sourceforge.net/)

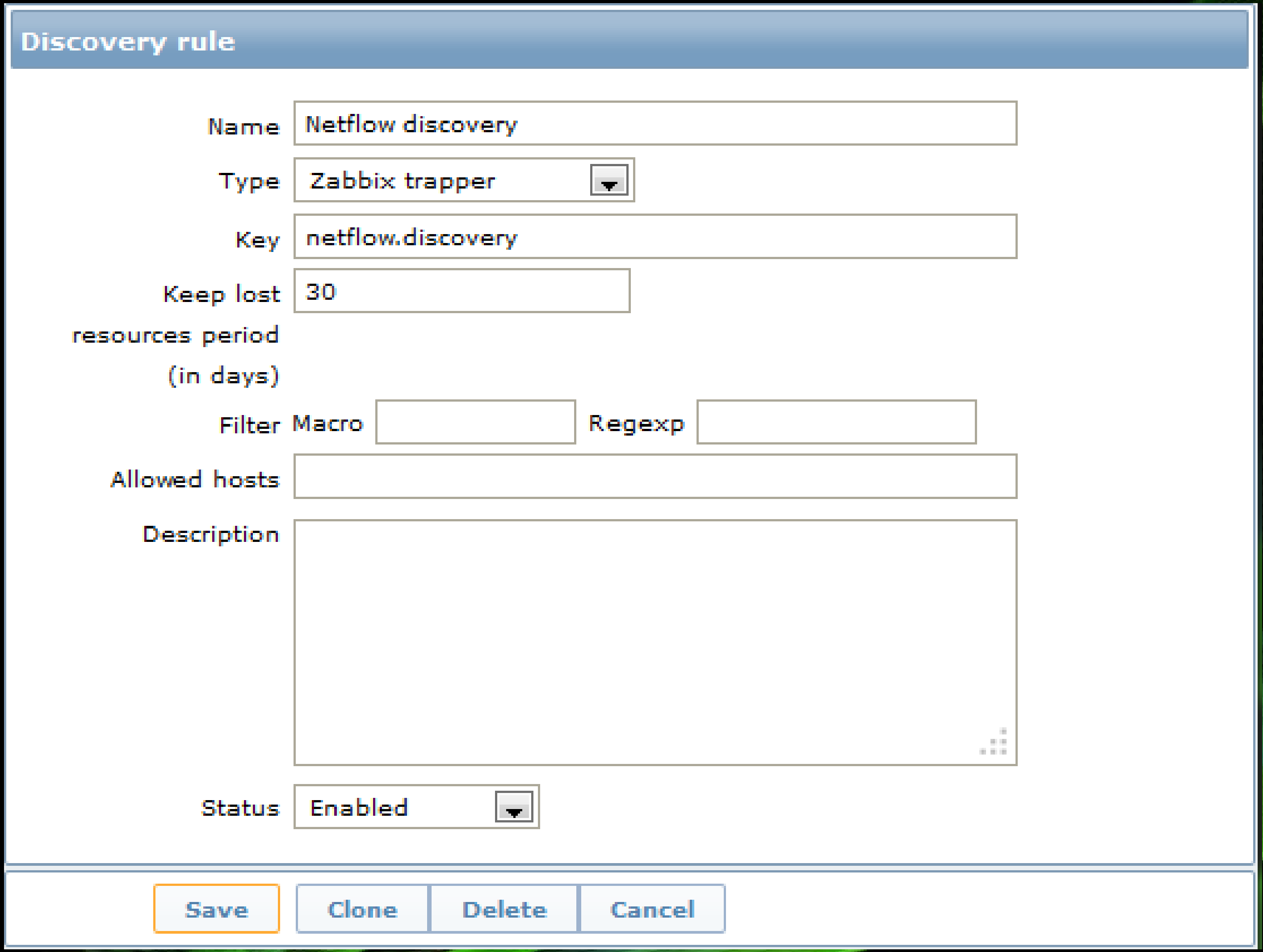
Case 4: NetFlow



Nfcapd

Script

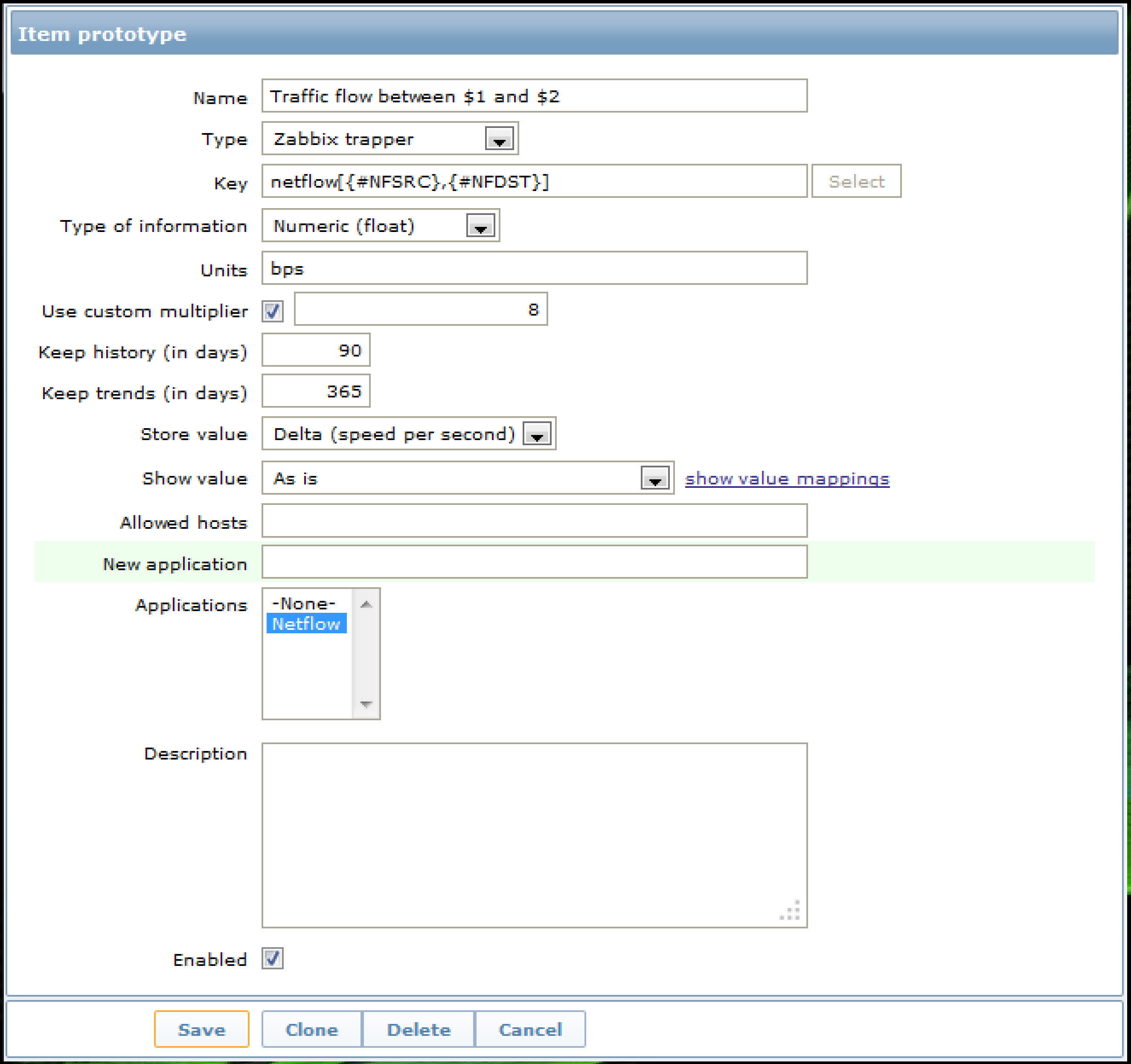
Case 4: NetFlow



Name	Netflow discovery
Type	Zabbix trapper
Key	netflow.discovery
Keep lost resources period (in days)	30
Filter	Macro Regexp
Allowed hosts	
Description	
Status	Enabled

Layout by orngjce223, CC-BY

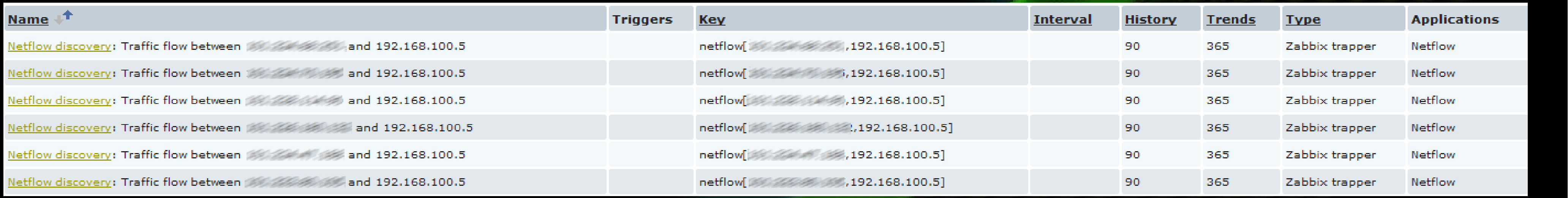
Case 4: NetFlow



Item prototype

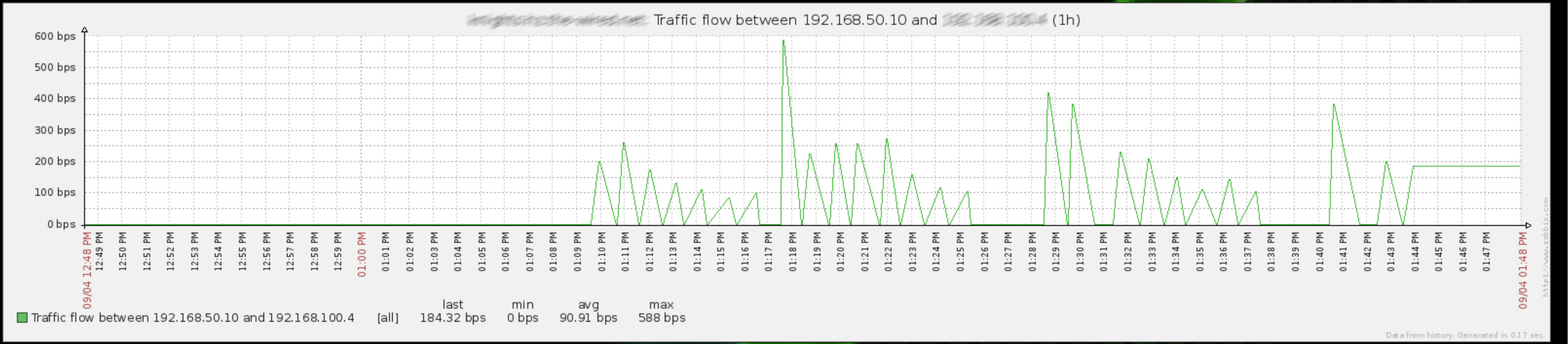
Name	Traffic flow between $1 and $2
Type	Zabbix trapper
Key	netflow[{#NFSRC},{#NFDST}]
Type of information	Numeric (float)
Units	bps
Use custom multiplier	☑ 8
Keep history (in days)	90
Keep trends (in days)	365
Store value	Delta (speed per second)
Show value	As is
Allowed hosts	
New application	
Applications	-None- Netflow
Description	
Enabled	☑

Case 4: NetFlow



Name	Triggers	Key	Interval	History	Trends	Type	Applications
Netflow discovery: Traffic flow between [redacted] and 192.168.100.5		netflow[[redacted],192.168.100.5]		90	365	Zabbix trapper	Netflow
Netflow discovery: Traffic flow between [redacted] and 192.168.100.5		netflow[[redacted],192.168.100.5]		90	365	Zabbix trapper	Netflow
Netflow discovery: Traffic flow between [redacted] and 192.168.100.5		netflow[[redacted],192.168.100.5]		90	365	Zabbix trapper	Netflow
Netflow discovery: Traffic flow between [redacted] and 192.168.100.5		netflow[[redacted],192.168.100.5]		90	365	Zabbix trapper	Netflow
Netflow discovery: Traffic flow between [redacted] and 192.168.100.5		netflow[[redacted],192.168.100.5]		90	365	Zabbix trapper	Netflow
Netflow discovery: Traffic flow between [redacted] and 192.168.100.5		netflow[[redacted],192.168.100.5]		90	365	Zabbix trapper	Netflow

Case 4: NetFlow



Conclusion

Low-Level Discovery:

- Makes SNMP life easier
- Makes FS and Network items a breeze
- Opens up a whole world of new possibilities (get creative!)

Thanks for listening!

Questions?

http://zbxtutorials.org | http://competa.com | http://nlzzg.nl

Layout by orngjce223, CC-BY