

ENGLISH



MEETUP ONLINE '23

Optimizing your data collection with
Low-level discovery and dependent
items



Dimitri Bellini

CEO, Quadrata Service Group, Italy
and United Arab Emirates (UAE)



Low-level discovery

- *a way to automatically create items, triggers, graphs and hosts*
- *used in most of Out of The Box Zabbix Templates*
- *highly customizable base on your needs*

All templates / Linux filesystems by Zabbix agent ... Discovery list / **Mounted filesystem discovery**

Item prototypes 4 Trigger prototypes 4 Graph prototypes 1 Host prototypes

Discovery rule Preprocessing LLD macros Filters 4 Overrides

* Name

Type

* Key

* Update interval

Custom intervals

Type	Interval	Period	Action
☒ Flexible ☐ Scheduling	50s	1-7,00:00-24:00	Remove
Add			

* Keep lost resources period

Out Of The Box Discovery Rules

- mounted filesystems
- network interfaces
- CPUs and CPU cores
- SNMP OIDs
- JMX objects
- ODBC SQL queries
- Windows services
- Windows performance counter instances
- Systemd services
- host interfaces in Zabbix
- IPMI sensors
- WMI queries
- Prometheus data
- VMWare Entities

LLD Main components

- **Discovery rule** - specifies, most importantly, the built-in item or custom script to retrieve discovery data
- **Preprocessing** - *applies some preprocessing to the discovered data*
- **LLD macros** - allows to extract some macro values to use in discovered items, triggers, etc
- **Filters** - allows to filter the discovered values
- **Overrides** - *allows to modify items, triggers, graphs or host prototypes when applying to specific discovered objects*

LLD how it works

Discovery Rule → Collect the required data using ALL the Zabbix Type available

Discovery rule Preprocessing LLD macros Filters 4 Overri

* Name

Mounted filesystem discovery

Type

Zabbix agent (active) ▼

Zabbix agent

Zabbix agent (active)

* Key

Simple check

* Update interval

SNMP agent

Custom intervals

Interval

50s

Zabbix internal

Zabbix trapper

External check

* Keep lost resources period

Database monitor

HTTP agent

Description

IPMI agent

SSH agent

TELNET agent

JMX agent

Dependent item

Enabled

Script

Example of output of LLD

```
[
  { "#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" },
]
```


Most used Item Type

HTTP Agent → Many use cases like Kubernetes monitoring, Application Monitoring, RestAPI

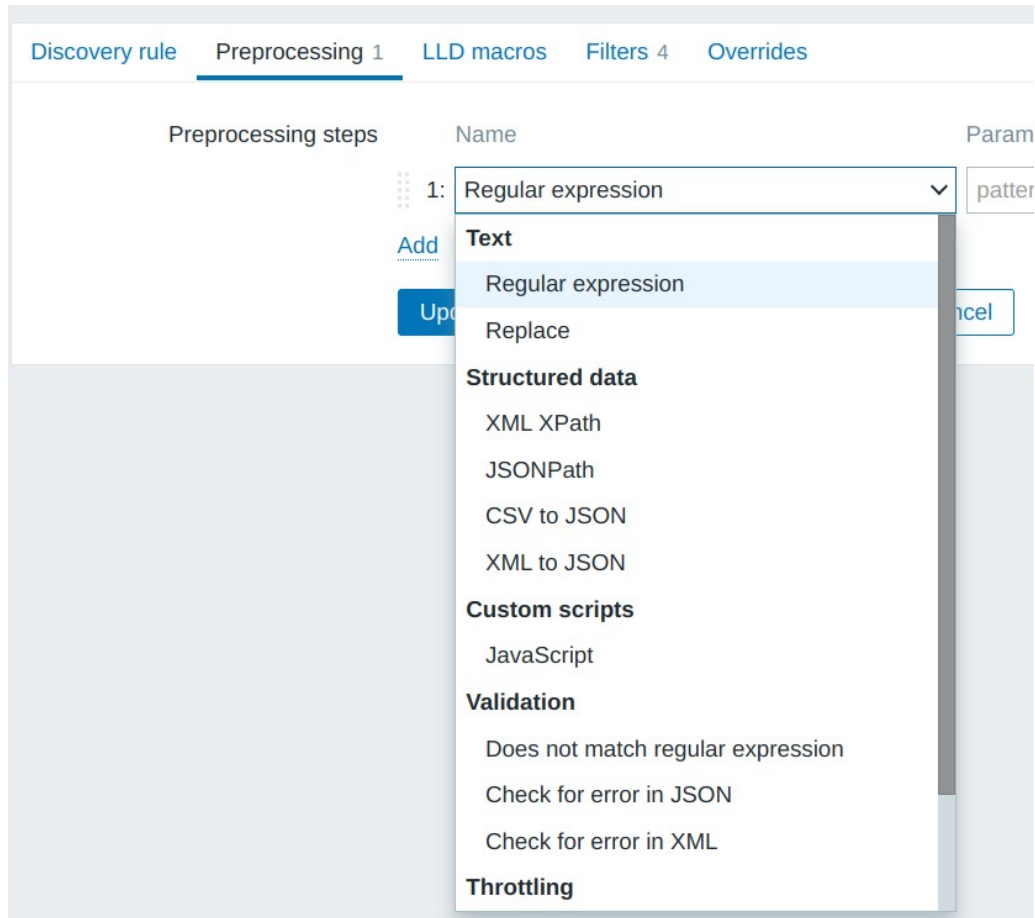
SNMP Agent → For all Router/SAN Switches etc

Dependent Item → Similar use case of HTTP Agent

Script → More complex scenarios with multiple RestAPI calls and data conversion. Ex. RestAPI execute login and receive a Token with the Token you can request the final data...

Where magic comes true

Preprocessing → Multiple way to extract the information you require coming from the LLD rule



The screenshot shows the Zabbix configuration interface for an LLD rule. The 'Preprocessing 1' tab is active. A dropdown menu is open, showing the following options:

- Text**
 - Regular expression
 - Replace
- Structured data**
 - XML XPath
 - JSONPath
 - CSV to JSON
 - XML to JSON
- Custom scripts**
 - JavaScript
- Validation**
 - Does not match regular expression
 - Check for error in JSON
 - Check for error in XML
- Throttling**

Most used Preprocessing Steps

- Regular expression
- JSONPath
- JavaScript

Next-Gen LLD discovery

- Reduce number of polling to your device using “Master/Dependant Items”
- Enhance Automation with LLD Overrides
- Enrich your LLD data using Javascript Preprocessor step
- Bring User Macros every where :-)
- Solve Tricky situations

Master/Dependant on LLD

Item Tags 1 Preprocessing 1

Parent items Proxmox VE by HTTP - Test

* Name Proxmox: Get cluster resources

Type HTTP agent

* Key proxmox.cluster.resources

Type of information Text

* URL https://{PVE.URL}:{PVE.URL.PORT}/api2/json/cluster/resources



Discovery Rule → Dependant from HTTP Agent Item

Discovery rule Preprocessing 1 LLD macros 8 Filters 2 Overrides

Parent discovery rules Proxmox VE by HTTP - Test

* Name VMs discovery

Type Dependent item

* Key proxmox.vms.discovery

* Master item Proxmox_Server: Proxmox: Get cluster resources

* Keep lost resources period 30d



HTTP Agent → Master Item Collect the main Data

- Less polling to monitored device
- Single Fetch for all the chain

Item prototype Tags 3 Preprocessing 2

Parent items Proxmox VE by HTTP - Test

* Name Proxmox: Storage [{#NODE.NAME}]/[{#STORAGE.NAME}]: Content

Type Dependent item

* Key proxmox.node.content[{#NODE.NAME},{#STORAGE.NAME}]

Type of information Character

* Master item Proxmox_Server: Proxmox: Get cluster resources

Item Prototype → Dependant from HTTP Agent Item

Master/Dependant on LLD

HTTP Agent → Raw data

```
[
  {
    "data": [
      {
        "cpu": 0.723949571936942,
        "status": "running",
        "name": "zabbix",
        "mem": 2793138336
      },
      {
        "cpu": 1.423949571936942,
        "status": "running",
        "name": "fedora",
        "mem": 2793138336
      },
      {
        "cpu": 0.2793138336,
        "status": "running",
        "name": "ubuntu",
        "mem": 2793138336
      }
    ]
  }
]
```



Discovery Rule → Hosts informations

Discovery rule	Preprocessing 1	LLD macros 2	Filters 2	Overrides
LLD macros		LLD macro	JSONPath	
		{#NODE.NAME}	\$.node	
		{#NODE.STATUS}	\$.token_id	

```
[
  {
    { "#{NODE.NAME}":"zabbix", "#{NODE.STATUS}":"running" },
    { "#{NODE.NAME}":"fedora", "#{NODE.STATUS}":"running" },
    { "#{NODE.NAME}":"ubuntu", "#{NODE.STATUS}":"running" }
  ]
]
```

Master/Dependant on LLD

Item Prototype → For all VMs create relative Items

Item prototype Tags 3 Preprocessing 1

* Name

Type

* Key

Type of information

* Master item



Item Preprocessing → Extract the value CPU coming from Master Item

Item prototype Tags 3 Preprocessing 1

Preprocessing steps	Name	Parameters
1:	JSONPath	<input type="text" value="\$data[?(@.name == \" {#node.name}\")].cpu.first()"=""/>

[Add](#)

Enhance Automation with LLD Overrides

VMWare Tag/Custom Attribute used to automate the Template association

All templates / BCOM - VMware FQDN

Discovery list / Discover VMware VMs FQDN

Item prototypes

Trigger prototypes

Graph prototypes

Host prototypes 1

Discovery rule

Preprocessing 3

LLD macros 3

Filters 2

Overrides 2

LLD macros

LLD macro	JSONPath	
{#VM.CA.MONITORING.DEPARTMENT}	\$.vmcustomattribute[?(@.name == "monitoring-department")].value.first()	Remove
{#VM.CA.MONITORING.ZBXAGENT}	\$.vmcustomattribute[?(@.name == "monitoring-zbxagent")].value.first()	Remove
{#VM.TAG.CATEGORY.MONITORING}	\$.tags[?(@.category == "Monitoring")].name.first()	Remove

Override

* Name

OS WINDOWS

If filter matches

Continue overrides

Stop processing

Filters

Type of calculation

And/Or

A and B

Label	Macro		Regular expression
A	{#VM.CA.MONITORING.ZBXAGENT}	matches	yes
B	{#VM.GUESTFAMILY}	matches	windowsGuest

Add

Operations

Condition

Host prototype matches . *

Add

Edit operation

Object

Host prototype

Condition

matches

.*

Create enabled

☐

Original

Discover

☐

Original

Link templates

☒

BCOM - Windows by Zabbix agent active

type here to search

Tags

☐

Original

Host inventory

☐

Original

Enrich your LLD data using Javascript Preprocessor step

All hosts / IRE

Discovery rule

JavaScript

```
1 var j = JSON.parse(value);
2 var name;
3 for (elem = 0; elem < Object.keys(j).length; elem++) {
4     if (typeof j[elem]["#SYSNAME"] != 'undefined') {
5         name = j[elem]["#SYSNAME"]
6         delete j[elem]["#SYSNAME"];
7     }
8 }
9 jclean = j.filter(function (e) {return e != null;})
10 if (typeof name != 'undefined') {
11     var endpoint = '{$ENDPOINT}'+'/' + name;
12     try {
13         var fields = {};
14         var req = new CurlHttpRequest();
15         resp = req.Get(endpoint, JSON.stringify({"fields": fields}));
16     } catch (error) {
17         Zabbix.Log(3, "Brocade_Virt_Generic_v3: cannot reach external API")
18     }
19     return JSON.stringify(jclean);
20 }
```



Push new Array elements to your Discovery Rule,
maybe based on result of a custom RestAPI

Bring User Macros every where

All hosts / Proxmox

Discovery rule

JavaScript

```
function (value) {  
  1 // Add User Macro to Discovery Output as an LLD Macro  
  2 var vms = JSON.parse(value);  
  3 var i= 0;  
  4 for (i = 0; i < vms['data'].length; i++) {  
  5   vms['data'][i]['token_id'] = '{$PVE.TOKEN.ID}';  
  6   vms['data'][i]['token_secret'] = '{$PVE.TOKEN.SECRET}';  
  7   vms['data'][i]['pve_url'] = '{$PVE.URL}';  
  8   vms['data'][i]['pve_url_port'] = '{$PVE.URL.PORT}';  
  9 }  
 10 //Zabbix.log(3, "Proxmox Discovery VMs: "+JSON.stringify(vms));  
 11 return JSON.stringify(vms);  
}
```

Add new Array elements to your Discovery Rule output to bring your User MACROS everywhere

Solve Tricky situations

Parent discovery rules [Template_Brocade_Mainframe_Virt_FabricOS_9.x](#)

* Name	Interfaces
Type	SNMP agent
* Key	discovery_brocade_with_portid
* Host interface	192.168.64.130 : 161
* SNMP OID	discovery[{{#SNMPVALUE}},1.3.6.1.4.1.1588.2.1.1.6.2.1.37,{{#SWIDXOR}},1.3.6.1.2



Add new Array elements to your Discovery Rule output to bring some Specific usefull OIDs or transform the value contents

```
function (value) {
  1 var obj = JSON.parse(value);
  2 var temp = obj.pop();
  3 var idxlong = temp["#{SNMPINDEX}"];
  4 for (var v in obj){
  5   if ( obj[v]['#{SWIDXOR}'] != undefined) {
  6     var temp1 = '' + obj[v]['#{SWIDXOR}'] + '';
  7     var temp = temp1.substr(4,2);
  8     obj[v]['#{SWIDXOR}'] = temp;
  9     obj[v]["#{SNMPINDEXLONG}"] = idxlong
 10   } else {
 11     obj[v]['#{SWIDXOR}'] = "NA";
 12     obj[v]["#{SNMPINDEXLONG}"] = idxlong
 13   }
 14 }
 15 return JSON.stringify(obj);
}
```



Solve Tricky situations

```
▼<Impianto Hostaddress=[REDACTED] Hostname=[REDACTED] NumeroCartelli="3" PosizioneImpianto="P" RevisioneSwCr01="1.29 ( build 5 )" SottoTipo="*" Tipologia="A__"
  UltimoAggiornamento="20230213_151754">
    <Sntp EnterpriseOid="17377" SogliaOccupazioneFS1="80" SogliaOccupazioneFS2="88"/>
    ▶<StatoFunzionamento AvvioControlloDisplay="20230212_162019" OutputLed="00010101" Protezione="1">
      ...
    </StatoFunzionamento>
    ▼<Cartelli NumeroCartelliInclusi="3" NumeroTotaleCartelli="3">
      ▼<Cartello Descrizione="" IdCartello="0" IdPanel="0" NomeCartello="" SottoTipo="*" Tipologia="A__">
        ▶<StatoFunzionamento Protezione="1">
          ...
        </StatoFunzionamento>
        ▼<Displays NumeroDisplay="2">
          <Display IdRow="0" IndirizzoDisplay="A" Installato="1" LenCodicePitto="8" NomeDisplay="PITT01" ProtocolloCCLDisplay="3_10" RevisioneFw="DSPMV72144UBRE\D8C11"
            Semaforo_acceso="0">
              ...
            </Display>
            <Display IdRow="1" IndirizzoDisplay="B" Installato="1" LenCodicePitto="8" NomeDisplay="FC1" ProtocolloCCLDisplay="3_10" RevisioneFw="DSPMV48048UBRE\D9E28C[2"
              Semaforo_acceso="0">
                ...
              </Display>
            </Displays>
          </Cartello>
        </Cartelli>
      </Impianto>
```

First Level LLD MACRO

Second Level LLD MACRO

Single Discover with Multiple LLD MACRO Levels

```
7 for (j = 0; j < displayCount; j++) {
8   Zabbix.log(3, "PMV Display Inizio Count: "+displayCount);
9   query="/Impianto/Cartelli/Cartello[@IdCartello="+i+"]/Displays/Display[@IdRow="+j+"]";
10  Zabbix.log(3, "PMV display Cartelli Query String: "+query);
11  out = XML.query(value,query);
12  out = XML.query(out, '/Display/@IdRow');
```

Solved with JavaScript Preproc:

→ XML.query

→ Array.push

Still *waiting* interesting enhancements

- **Ability to show LLD MACRO output** → ZBXNEXT-6812 (see if LLD macros are correctly populated)

Thank you!



 www.quadrata-group.com



#zabbixitalia



QUADRATA
SERVICE GROUP s.r.l.