



Improving Your Resource Low-Level Discovery Workflows with Bulk Data Collection

Arturs Lontons

Course Content Creator



Low-level discovery

Low-level discovery

Low-level discovery is a core Zabbix feature used to automate creation and management of resources

- ▶ Automatically discover resources such as file systems, services, network interfaces, containers and more
- ▶ Automatically react to resources being added or removed
- ▶ Use filters to discover only the required resources
- ▶ Instantly apply changes to all of the discovered resources

<input type="checkbox"/>	Cisco c3750	Cisco IOS by SNMP: Cisco IOS: SNMP walk system CPUs: CPU Discovery	Item prototypes 1	Trigger prototypes 1	Graph prototypes 1
<input type="checkbox"/>	Cisco c3750	Cisco IOS by SNMP: Cisco IOS: SNMP walk entity serial numbers: Entity Serial Numbers Discovery	Item prototypes 1	Trigger prototypes 1	Graph prototypes
<input type="checkbox"/>	Cisco c3750	Cisco IOS by SNMP: Cisco IOS: SNMP walk EtherLike-MIB interfaces: EtherLike-MIB Discovery	Item prototypes 1	Trigger prototypes 1	Graph prototypes
<input type="checkbox"/>	Cisco c3750	Cisco IOS by SNMP: Cisco IOS: SNMP walk fans: FAN Discovery	Item prototypes 1	Trigger prototypes 2	Graph prototypes

<input type="checkbox"/>	Host	Name ▲	Items	Triggers	Graphs
<input type="checkbox"/>	Linux server	Block devices discovery	Item prototypes 9	Trigger prototypes 1	Graph prototypes 3
<input type="checkbox"/>	Linux server	Get filesystems: Mounted filesystem discovery	Item prototypes 7	Trigger prototypes 5	Graph prototypes 2
<input type="checkbox"/>	Linux server	Network interface discovery	Item prototypes 9	Trigger prototypes 4	Graph prototypes 1

<input type="checkbox"/>	Template	Name ▲	Items	Triggers	Graphs	Hosts
<input type="checkbox"/>	AWS by HTTP	EC2 instances discovery	Item prototypes	Trigger prototypes	Graph prototypes	Host prototypes 1
<input type="checkbox"/>	AWS by HTTP	ECS clusters discovery	Item prototypes	Trigger prototypes	Graph prototypes	Host prototypes 1
<input type="checkbox"/>	AWS by HTTP	ELB load balancers discovery	Item prototypes	Trigger prototypes	Graph prototypes	Host prototypes 1
<input type="checkbox"/>	AWS by HTTP	RDS instances discovery	Item prototypes	Trigger prototypes	Graph prototypes	Host prototypes 1
<input type="checkbox"/>	AWS by HTTP	S3 buckets discovery	Item prototypes	Trigger prototypes	Graph prototypes	Host prototypes 1

Low-level discovery prototypes





Low-level discovery rules create items, triggers, graphs and hosts from prototypes

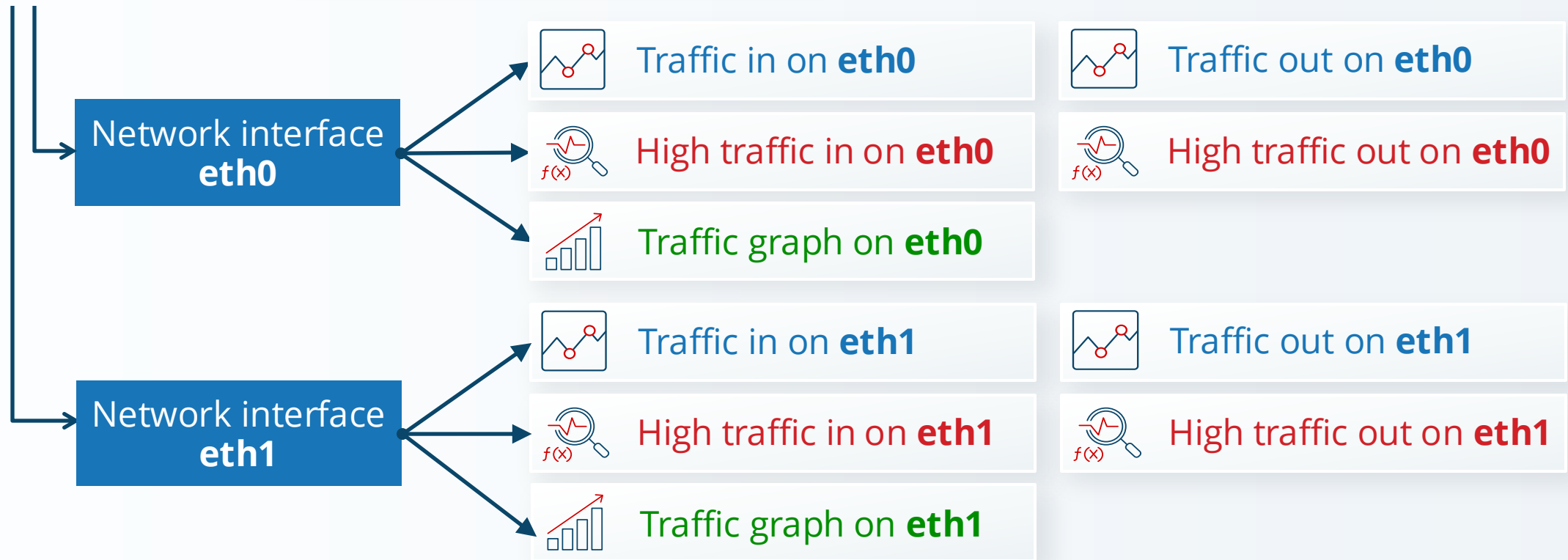
- ▶ Prototypes behave like blueprints for discovered entities
- ▶ Low-level discovery macros – {#MACRO}, are used in resource prototypes and are populated with the attributes of the discovered resources

<input type="checkbox"/>	Name ▲	Key	Interval	History	Trends	Type
<input type="checkbox"/>	... Interface {#IFNAME}: Bits received	net.if.in["{#IFNAME}"]	3m	7d	365d	Zabbix agent
<input type="checkbox"/>	... Interface {#IFNAME}: Bits sent	net.if.out["{#IFNAME}"]	3m	7d	365d	Zabbix agent

<input type="checkbox"/>	Name ▲	Triggers	Key	Interval
<input type="checkbox"/>	... Network interface discovery: Interface eth0: Bits received	Triggers 1	net.if.in["eth0"]	3m
<input type="checkbox"/>	... Network interface discovery: Interface eth0: Bits sent	Triggers 1	net.if.out["eth0"]	3m
<input type="checkbox"/>	... Network interface discovery: Interface lo: Bits received	Triggers 1	net.if.in["lo"]	3m
<input type="checkbox"/>	... Network interface discovery: Interface lo: Bits sent	Triggers 1	net.if.out["lo"]	3m

LLD rule

 Traffic in on {#IFNAME}	 Traffic out on {#IFNAME}
 High traffic in on {#IFNAME}	 High traffic out on {#IFNAME}
 Traffic graph on {#IFNAME}	



Low-level discovery data

Under the hood low-level discovery uses JSON data with "macro":"value" pairs

- ▶ Any item type can be used for low-level discovery
- ▶ For some item types, Zabbix automatically formats the collected data in LLD format
- ▶ For other item types, the data must be transformed manually – either before Zabbix receives it or once the data is received by using preprocessing and LLD macro assignments via JSONPath

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

Dependent items



Dependent items

Dependent items utilize another item – master item as the source of data

- ▶ Dependent item values are extracted from the master item via preprocessing
- ▶ An unlimited number of dependent items can be created
- ▶ Dependent items don't have an update interval – they are updated together with the master item

<input type="checkbox"/>	Name ▲	Triggers	Key	Interval	History	Trends	Type	Status	Tags
<input type="checkbox"/>	... MySQL: Get status variables		mysql.get_status_variables["\${MYSQL.DSN}","\${MYSQL.USER}","\${MYSQL.PASSWORD}"]	1m	0		Zabbix agent	Enabled	component: raw
<input type="checkbox"/>	... MySQL: Get status variables: MySQL: Threads cached		mysql.threads_cached		7d	365d	Dependent item	Enabled	component: threads
<input type="checkbox"/>	... MySQL: Get status variables: MySQL: Threads connected		mysql.threads_connected		7d	365d	Dependent item	Enabled	component: threads
<input type="checkbox"/>	... MySQL: Get status variables: MySQL: Threads created per second		mysql.threads_created.rate		7d	365d	Dependent item	Enabled	component: threads
<input type="checkbox"/>	... MySQL: Get status variables: MySQL: Threads running		mysql.threads_running		7d	365d	Dependent item	Enabled	component: threads

Displaying 5 of 5 found

Dependent items

Dependent items utilize preprocessing to extract values from the master item

- ▶ At least one preprocessing step is required
- ▶ If no preprocessing step is defined, the dependant item will copy the values collected by the master item

The image shows two screenshots of the Zabbix configuration interface. The left screenshot shows the configuration for a dependent item, and the right screenshot shows the configuration for its preprocessing steps.

Left Screenshot: Item Configuration

- Item: MongoDB node by Zabbix agent 2
- Parent items: MongoDB node by Zabbix agent 2
- Name: Bytes in, rate
- Type: Dependent item
- Key: mongodb.network.bytes_in.rate
- Type of information: Numeric (float)
- Master item: MongoDB: Get server status
- Units: Bps

Right Screenshot: Preprocessing Steps Configuration

Preprocessing steps	Name	Parameters
1:	JSONPath	\$.network.bytesIn
2:	Change per second	

Type of information: Numeric (float)

Dependent items

Master item

```

{
  "asserts": {
    "msg": 0,
    "regular": 0,
    "rollovers": 0,
    "tripwire": 0,
    "user": 773,
    "warning": 0
  },
  "batchedDeletes": {
    "batches": 1,
    "docs": 1,
    "refetchesDueToYield": 0,
    "stagedSizeBytes": 275,
    "timeInBatchMillis": 0
  },
  "changeStreamPreImages": {
    "purgingJob": {
      "bytesDeleted": 0,
      "docsDeleted": 0,
      "maxStartWallTimeMillis": 0,
      "scannedCollections": 0,
      "scannedInternalCollections": 0,
      "timeElapsedMillis": 0,
      "totalPass": 0
    }
  },
  ...
  "uri": "statistics:"
}

```

Item Tags 1 Preprocessing 2

Preprocessing steps ?	Name	Parameters
1:	JSONPath	\$.asserts.regular
2:	Change per second	

Add

Item Tags 1 Preprocessing 2

Preprocessing steps ?	Name	Parameters
1:	JSONPath	\$.asserts.user
2:	Change per second	

Add

Item Tags 1 Preprocessing 2

Preprocessing steps ?	Name	Parameters
1:	JSONPath	\$.asserts.warning
2:	Change per second	

Add

Dependent items:

Name ▲	Last check	Last value
Asserts: message, rate ?	9s	0
Asserts: regular, rate ?	9s	0
Asserts: rollovers, rate ?	9s	0
Asserts: user, rate ?	9s	0.01667
Asserts: warning, rate ?	9s	0

Dependent items - notes

- ▶ Using dependent items can reduce data collection performance overhead
- ▶ The data is collected via a single request and processed by Zabbix
- ▶ Any type of preprocessing can be used to transform data
- ▶ Most commonly JSONPath/Xpath/Regex preprocessing is used
- ▶ Low-level discovery rules can also be of dependent item type



Dependent low-level discovery

Dependent low-level discovery

A discovery rule can also be of dependent item type

- ▶ Master item collects all of the discovery related data
- ▶ Low-level discovery rule is executed every time the master item collects values

Discovery rules

All templates / AWS EC2 by HTTP / Discovery list / Instance Volumes discovery / Item prototypes 18 / Trigger prototypes 2 / Graph prototypes 3 / Host prototypes

Discovery rule / Preprocessing 2 / LLD macros / Filters 2 / Overrides

* Name

Type

* Key

* Master item

* Delete lost resources ?

* Disable lost resources ?

Description

Enabled

Dependent low-level discovery

The retrieved JSON contains the low-level discovery information and values used by the items.

- ▶ LLD macros can be assigned in the discovery rule
- ▶ JavaScript preprocessing can also be used to retrieve and populate LLD macro values

```
[
  {
    "fsname": "/",
    "fstype": "overlay",
    "bytes": {
      "total": 85829070848,
      "free": 49161748480,
      "used": 36667322368,
      "pfree": 57.27866793182881,
      "pused": 42.72133206817119
    },
    "inodes": {
      "total": 41941440,
      "free": 41664647,
      "used": 276793,
      "pfree": 99.34004888721036,
      "pused": 0.6599511127896419
    }
  },
  {
    "fsname": "/sys",
    "fstype": "sysfs",
    "bytes": {
      "total": 0,
      "free": 0,
      "used": 0,
      "pfree": 0,
      "pused": 0
    }
  }
],
...

```

Discovery rules

All hosts / PostgreSQL database Enabled ZBX Discovery list / Mounted filesystem discovery

Item prototypes 7 Trigger prototypes 5 Graph prototypes 2 Host prototypes

Discovery rule Preprocessing 2 LLD macros 2 Filters 4 Overrides 1

LLD macros	LLD macro	JSONPath	
	{#FSNAME}	\$.fsname	Remove
	{#FSTYPE}	\$.fstype	Remove
	Add		

Update
Clone
Execute now
Test
Delete
Cancel

Dependent low-level discovery

The item prototypes for dependent LLD also utilize dependent item type and use the same master item as the dependent LLD

- ▶ JSONPath preprocessing is used to extract values from the master item
- ▶ The low-level discovery macro used in JSONPath will be resolved as the element name for each discovered resource

Item prototype

Item prototype [Tags 2](#) [Preprocessing 1](#)

* Name

Type

* Key

Type of information

Item prototype

Item prototype [Tags 2](#) [Preprocessing 1](#)

Preprocessing steps ?	Name	Parameters	Custom on fail	Actions
1:	<input type="text" value="JSONPath"/>	<input type="text" value="\$[?(@.fsname=='{#FSNAME}')]bytes.free.first()"/>	<input type="checkbox"/>	Test Remove

[Add](#) [Test all steps](#)

Type of information

Dependent low-level discovery

The item prototypes for dependent LLD also utilize dependent item type and use the same master item as the dependent LLD

- ▶ JSONPath preprocessing is used to extract values from the master item
- ▶ The low-level discovery macro used in JSONPath will be resolved as the element name for each discovered resource

The screenshot displays the Zabbix Item configuration interface. The main window shows the 'Item' configuration for a dependent item. The 'Name' is set to '/: Free space', the 'Type' is 'Dependent item', and the 'Key' is 'bytes.free[]'. The 'Type of information' is 'Numeric (unsigned)'. The 'Discovered by' field is set to 'Filesystems discovery'.

An inset window shows the 'Preprocessing steps' configuration. It lists one step: 'JSONPath' with the parameters '\$[?(@.fsname==/)].bytes.free.first()'. The 'Type of information' for this step is also 'Numeric (unsigned)'. The 'Custom on fail' checkbox is unchecked. The 'Actions' column contains links for 'Test', 'Remove', and 'Test all steps'.

Preprocessing steps	Name	Parameters	Custom on fail	Actions
1:	JSONPath	\$[?(@.fsname==/)].bytes.free.first()	<input type="checkbox"/>	Test Remove

Dependent SNMP low-level discovery

Dependent LLD is also the recommended way of performing discovery of SNMP resources

- ▶ Create a *walk[oid1,oid2,oid3, ...]* master item
- ▶ Create an LLD rule with SNMP walk to JSON preprocessing step
- ▶ Assign LLD macros in the preprocessing
- ▶ Create dependent item prototypes with SNMP walk value preprocessing step
- ▶ Apply *Discard unchanged with heartbeat* preprocessing to reduce the frequency of LLD execution (Unique behavior for SNMP discovery)

Dependent SNMP low-level discovery

First, create *walk[oid1,oid2,oid3, ...]* master item

- ▶ The item will perform SNMP walk across all of the specified OIDs and return their values

Item

Item Tags 1 Preprocessing

Parent items Cisco IOS by SNMP

* Name Cisco IOS: SNMP walk network interfaces

Type SNMP agent

* Key net.if.walk

Type of information Text

* Host interface cisco.example.com:10161

* SNMP OID ? walk[1.3.6.1.2.1.2.2.1.8,1.3.6.1.2.1.2.2.1.7,1.3.6.1.2.1.31.1.1.1.18,1.3.6.1.2.1.31.1.1.1.19,1.3.6.1.2.1.2.2.1.10,1.3.6.1.2.1.2.2.1.1,1.3.6.1.2.1.2.2.1.2.5179,1.3.6.1.2.1.2.2.1.2.5180,1.3.6.1.2.1.2.2.1.2.5181,1.3.6.1.2.1.2.2.1.2.10101,1.3.6.1.2.1.2.2.1.2.10102,1.3.6.1.2.1.2.2.1.2.10103,1.3.6.1.2.1.2.2.1.2.10104,1.3.6.1.2.1.2.2.1.2.10105]

* Update interval 1m

```
.1.3.6.1.2.1.2.2.1.19.10125 = Counter32: 0  
.1.3.6.1.2.1.2.2.1.19.10126 = Counter32: 0  
.1.3.6.1.2.1.2.2.1.19.10127 = Counter32: 0  
.1.3.6.1.2.1.2.2.1.19.10128 = Counter32: 0  
.1.3.6.1.2.1.2.2.1.19.14501 = Counter32: 0  
.1.3.6.1.2.1.2.2.1.2.1 = STRING: "Vlan1"  
.1.3.6.1.2.1.2.2.1.2.10 = STRING: "Vlan10"  
.1.3.6.1.2.1.2.2.1.2.5179 = STRING: "StackPort1"  
.1.3.6.1.2.1.2.2.1.2.5180 = STRING: "StackSub-St1-1"  
.1.3.6.1.2.1.2.2.1.2.5181 = STRING: "StackSub-St1-2"  
.1.3.6.1.2.1.2.2.1.2.10101 = STRING: "GigabitEthernet1/0/1"  
.1.3.6.1.2.1.2.2.1.2.10102 = STRING: "GigabitEthernet1/0/2"  
.1.3.6.1.2.1.2.2.1.2.10103 = STRING: "GigabitEthernet1/0/3"  
.1.3.6.1.2.1.2.2.1.2.10104 = STRING: "GigabitEthernet1/0/4"  
.1.3.6.1.2.1.2.2.1.2.10105 = STRING: "GigabitEthernet1/0/5"
```

Dependent SNMP low-level discovery

Create a dependent LLD rule with SNMP walk to JSON preprocessing

- ▶ Assign LLD macro values to OIDs

Discovery rule **Preprocessing 2** LLD macros Filters 12 Overrides

Parent discovery rules **Cisco IOS by SNMP**

* Name

Type

* Key

* Master item

* Delete lost resources

* Disable lost resources

Description

Preprocessing steps	Name	Parameters																					
1:	SNMP walk to JSON	<table border="1"> <thead> <tr> <th>Field name</th> <th>OID prefix</th> <th>Format</th> </tr> </thead> <tbody> <tr> <td>{#IFOPERSTATUS}</td> <td>1.3.6.1.2.1.2.2.1.8</td> <td>Unchanged</td> </tr> <tr> <td>{#IFADMINSTATU}</td> <td>1.3.6.1.2.1.2.2.1.7</td> <td>Unchanged</td> </tr> <tr> <td>{#IFALIAS}</td> <td>1.3.6.1.2.1.31.1.1.</td> <td>Unchanged</td> </tr> <tr> <td>{#IFNAME}</td> <td>1.3.6.1.2.1.31.1.1.</td> <td>Unchanged</td> </tr> <tr> <td>{#IFDESCR}</td> <td>1.3.6.1.2.1.2.2.1.2</td> <td>Unchanged</td> </tr> <tr> <td>{#IFTYPE}</td> <td>1.3.6.1.2.1.2.2.1.3</td> <td>Unchanged</td> </tr> </tbody> </table>	Field name	OID prefix	Format	{#IFOPERSTATUS}	1.3.6.1.2.1.2.2.1.8	Unchanged	{#IFADMINSTATU}	1.3.6.1.2.1.2.2.1.7	Unchanged	{#IFALIAS}	1.3.6.1.2.1.31.1.1.	Unchanged	{#IFNAME}	1.3.6.1.2.1.31.1.1.	Unchanged	{#IFDESCR}	1.3.6.1.2.1.2.2.1.2	Unchanged	{#IFTYPE}	1.3.6.1.2.1.2.2.1.3	Unchanged
Field name	OID prefix	Format																					
{#IFOPERSTATUS}	1.3.6.1.2.1.2.2.1.8	Unchanged																					
{#IFADMINSTATU}	1.3.6.1.2.1.2.2.1.7	Unchanged																					
{#IFALIAS}	1.3.6.1.2.1.31.1.1.	Unchanged																					
{#IFNAME}	1.3.6.1.2.1.31.1.1.	Unchanged																					
{#IFDESCR}	1.3.6.1.2.1.2.2.1.2	Unchanged																					
{#IFTYPE}	1.3.6.1.2.1.2.2.1.3	Unchanged																					

Dependent SNMP low-level discovery

Create dependent item prototypes with SNMP walk value preprocessing

- ▶ Specify the OID from which to extract the item value
- ▶ Use the `{#SNMPINDEX}` LLD macro in preprocessing
- ▶ The data is collected from the master item utilized by the LLD rule

The screenshot displays the Zabbix configuration interface for an Item prototype. The main configuration window shows the following details:

- Item prototype:** Cisco IOS by SNMP
- Name:** Interface `{#IFNAME}`{`{#IFALIAS}`}: B
- Type:** Dependent item
- Key:** `net.if.in[ifHCInOctets.{#SNMPINDEX}]`
- Type of information:** Numeric (unsigned)
- Master item:** Cisco c3750: Cisco IOS: SNMP wal

An inset window titled "Item prototype" provides a detailed view of the preprocessing steps:

Preprocessing steps	Name	Parameters	Custom on fail	Actions
1:	SNMP walk value	1.3.6.1.2.1.31.1.1.1.6. <code>{#SNMPINDEX}</code>	Unchanged	<input type="checkbox"/> Test Remove
2:	Change per second			<input type="checkbox"/> Test Remove
3:	Custom multiplier	8		<input type="checkbox"/> Test Remove

Additional options in the inset window include an "Add" button and a "Type of information" dropdown set to "Numeric (unsigned)".

Dependent SNMP low-level discovery

Apply *Discard unchanged* preprocessing to prevent the LLD rule from running every time the master item receives new values

- ▶ Even though the master item receives new values (metrics), the LLD data remains unchanged and will be discarded
- ▶ This is unique throttling behavior which works only with SNMP walk to JSON preprocessing

The screenshot shows the Zabbix configuration interface for LLD rules. The breadcrumb navigation includes 'Discovery rule', 'Preprocessing 2', 'LLD macros', 'Filters 12', and 'Overrides'. The 'Preprocessing steps' section is active, showing two steps:

- Step 1: 'SNMP walk to JSON' (dropdown menu). Below it is a table of parameters:

Field name	OID prefix	Format	
{#FOPERSTATUS}	1.3.6.1.2.1.2.2.1.8	Unchanged	Remove
{#FADMINSTATU}	1.3.6.1.2.1.2.2.1.7	Unchanged	Remove
{#FALIAS}	1.3.6.1.2.1.31.1.1	Unchanged	Remove
{#FNAME}	1.3.6.1.2.1.31.1.1	Unchanged	Remove
{#FDESCR}	1.3.6.1.2.1.2.2.1.2	Unchanged	Remove
{#FATYPE}	1.3.6.1.2.1.2.2.1.3	Unchanged	Remove

Below the table is an 'Add' button. Step 2 is 'Discard unchanged with heartbeat' (dropdown menu) with a value of '1h' in a text input field. To the right of each step is a 'Custom on fail' checkbox, which is currently unchecked.

Dependent SNMP low-level discovery

Apply *Discard unchanged* preprocessing to prevent the LLD rule from running every time the master item receives new values

- ▶ Even though the master item receives new values (metrics), the LLD data remains unchanged and will be discarded
- ▶ This is unique throttling behavior which works only with SNMP walk to JSON preprocessing

The screenshot shows the Zabbix configuration interface for LLD rules. The breadcrumb navigation includes 'Discovery rule', 'Preprocessing 2', 'LLD macros', 'Filters 12', and 'Overrides'. The 'Preprocessing steps' section is active, showing two steps:

- Step 1: 'SNMP walk to JSON' (dropdown menu). Below it is a table of parameters:

Field name	OID prefix	Format	
{#FOPERSTATUS}	1.3.6.1.2.1.2.2.1.8	Unchanged	Remove
{#FADMINSTATU}	1.3.6.1.2.1.2.2.1.7	Unchanged	Remove
{#FALIAS}	1.3.6.1.2.1.31.1.1	Unchanged	Remove
{#FNAME}	1.3.6.1.2.1.31.1.1	Unchanged	Remove
{#FDESCR}	1.3.6.1.2.1.2.2.1.2	Unchanged	Remove
{#FATYPE}	1.3.6.1.2.1.2.2.1.3	Unchanged	Remove

Below the table is an 'Add' button. Step 2 is 'Discard unchanged with heartbeat' (dropdown menu) with a value of '1h' and a 'Custom on fail' checkbox.

Dependent SNMP low-level discovery

Master item data #1

```
.1.3.6.1.2.1.2.2.1.19.1 = Counter32: 0  
.1.3.6.1.2.1.2.2.1.19.10 = Counter32: 0  
.1.3.6.1.2.1.2.2.1.2.1 = STRING: "Vlan1"  
.1.3.6.1.2.1.2.2.1.2.10 = STRING: "Vlan10"
```

Outbound packets discarded (Value collected by item)

Master item data #2

```
.1.3.6.1.2.1.2.2.1.19.1 = Counter32: 0  
.1.3.6.1.2.1.2.2.1.19.10 = Counter32: 1  
.1.3.6.1.2.1.2.2.1.2.1 = STRING: "Vlan1"  
.1.3.6.1.2.1.2.2.1.2.10 = STRING: "Vlan10"
```

Interface Description ({#IFDESCR} value used in LLD)

LLD not executed with *Discard unchanged* preprocessing, because the LLD values have not changed (even though new metrics are present in the master item)

Final notes

- ▶ Using dependent low-level discovery rules allow collecting item values and discover resources from the same source – the master item
- ▶ Using a single master item to collect data in bulk reduces the performance impact on the monitored endpoint
- ▶ Zabbix proxies can be used to further move the preprocessing performance overhead from Zabbix server to Zabbix proxies
- ▶ Collected data can be transformed into LLD format by JavaScript preprocessing
- ▶ LLD macro values can be extracted from JSON data by using the LLD macros section of the LLD rule

ZABBIX 20 YEARS

Thank you!
