ZABBIX 20 YEARS

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

Arturs Lontons

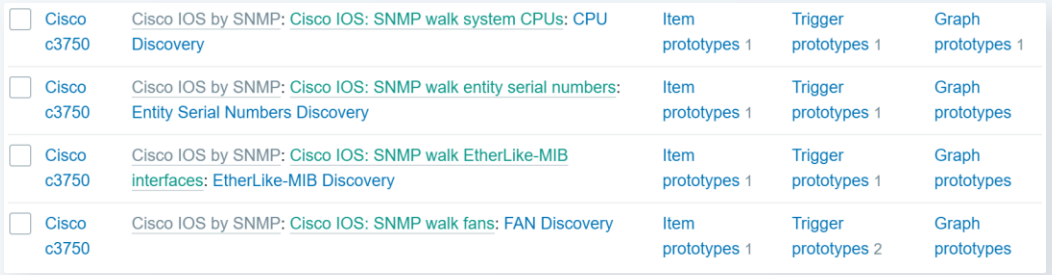
Course Content Creator

Low-level discovery

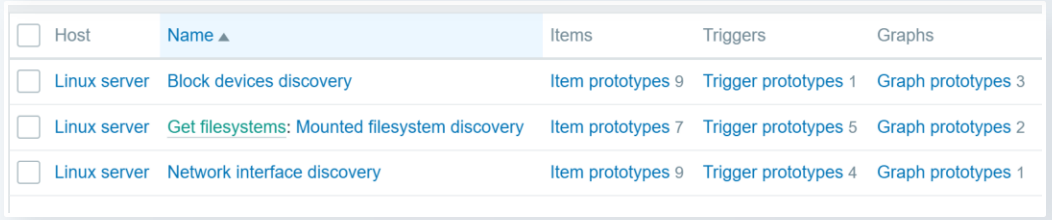
Low-level discovery

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

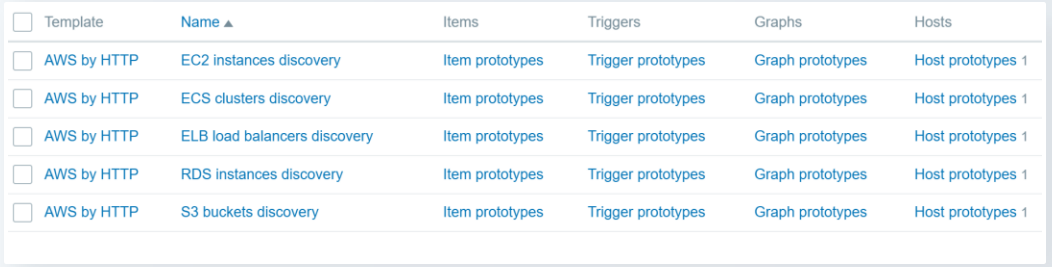
- Automatically discover resources such as file systems, services, network interfaces, containers and more
- Automatically react to resources being added or removed
- Use filters to discover only the required resources
- Instantly apply changes to all of the discovered resources



Cisco c3750	Cisco IOS by SNMP: Cisco IOS: SNMP walk system CPUs: CPU Discovery	Item prototypes 1	Trigger prototypes 1	Graph prototypes 1
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
Cisco c3750	Cisco IOS by SNMP: Cisco IOS: SNMP walk EtherLike-MIB interfaces: EtherLike-MIB Discovery	Item prototypes 1	Trigger prototypes 1	Graph prototypes
Cisco c3750	Cisco IOS by SNMP: Cisco IOS: SNMP walk fans: FAN Discovery	Item prototypes 1	Trigger prototypes 2	Graph prototypes



Host	Name ▲	Items	Triggers	Graphs
Linux server	Block devices discovery	Item prototypes 9	Trigger prototypes 1	Graph prototypes 3
Linux server	Get filesystems: Mounted filesystem discovery	Item prototypes 7	Trigger prototypes 5	Graph prototypes 2
Linux server	Network interface discovery	Item prototypes 9	Trigger prototypes 4	Graph prototypes 1



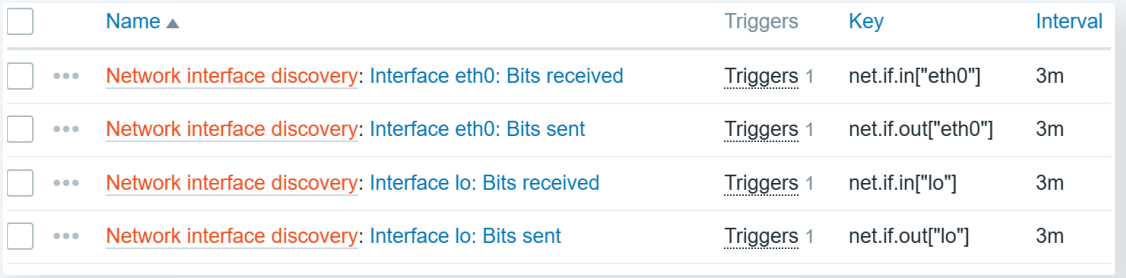
Template	Name ▲	Items	Triggers	Graphs	Hosts
AWS by HTTP	EC2 instances discovery	Item prototypes	Trigger prototypes	Graph prototypes	Host prototypes 1
AWS by HTTP	ECS clusters discovery	Item prototypes	Trigger prototypes	Graph prototypes	Host prototypes 1
AWS by HTTP	ELB load balancers discovery	Item prototypes	Trigger prototypes	Graph prototypes	Host prototypes 1
AWS by HTTP	RDS instances discovery	Item prototypes	Trigger prototypes	Graph prototypes	Host prototypes 1
AWS by HTTP	S3 buckets discovery	Item prototypes	Trigger prototypes	Graph prototypes	Host prototypes 1

Low-level discovery prototypes

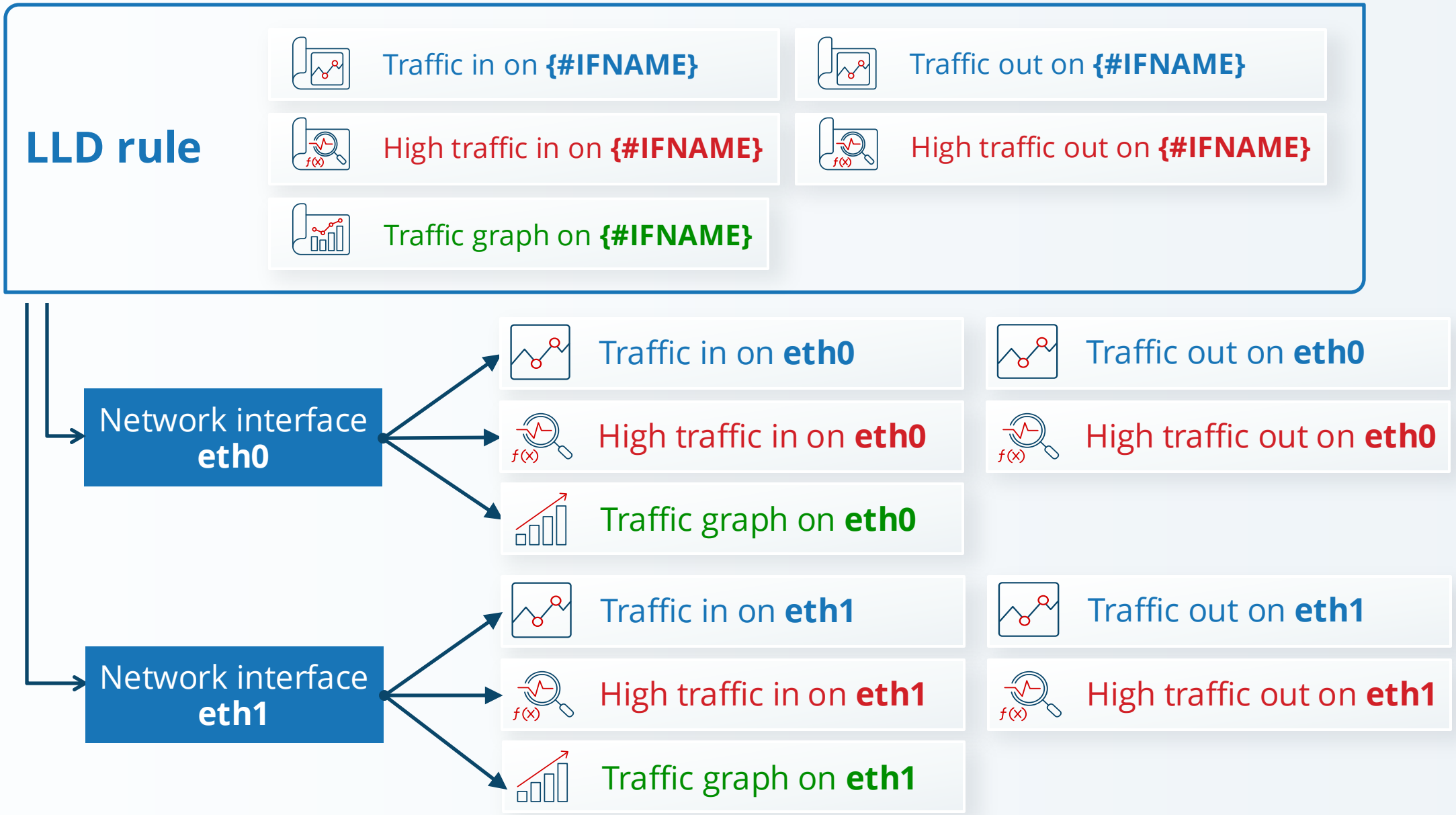
Low-level discovery rules create items, triggers, graphs and hosts from prototypes

- 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

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



Name ▲	Triggers	Key	Interval
Network interface discovery: Interface eth0: Bits received	Triggers 1	net.if.in["eth0"]	3m
Network interface discovery: Interface eth0: Bits sent	Triggers 1	net.if.out["eth0"]	3m
Network interface discovery: Interface lo: Bits received	Triggers 1	net.if.in["lo"]	3m
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}
Network interface eth0
Traffic in on eth0
Traffic out on eth0
High traffic in on eth0
High traffic out on eth0
Traffic graph on eth0
Network interface eth1
Traffic in on eth1
Traffic out on eth1
High traffic in on eth1
High traffic out on eth1
Traffic graph on eth1

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 extract from the master item via preprocessing
- An unlimited number of dependent items can be create d
- Dependent items don’t have an update interval – they are updated together with the master item

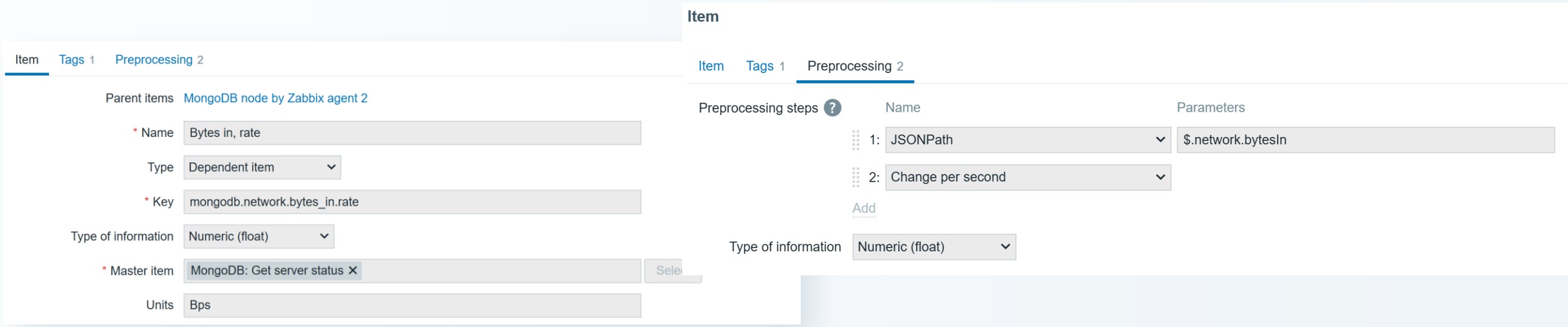
Name ▲	Triggers	Key	Interval	History	Trends	Type	Status	Tags
MySQL: Get status variables		mysql.get_status_variables["{$MYSQL.DSN}","{$MYSQL.USER}","{$MYSQL.PASSWORD}"]	1m	0		Zabbix agent	Enabled	component: raw
MySQL: Get status variables: MySQL: Threads cached		mysql.threads_cached		7d	365d	Dependent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

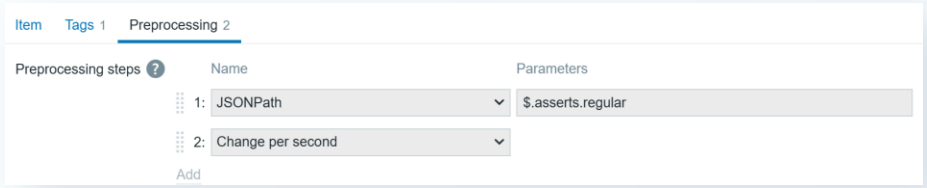
- At least one preprocessing step is required
- If no preprocessing step is defined, the dependant item will copy the values collected by the master item

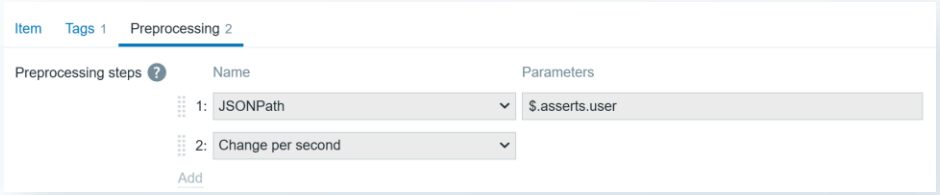


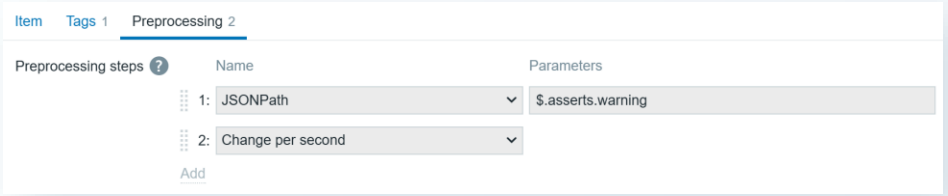
Dependent items

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:"
  }
}
```





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

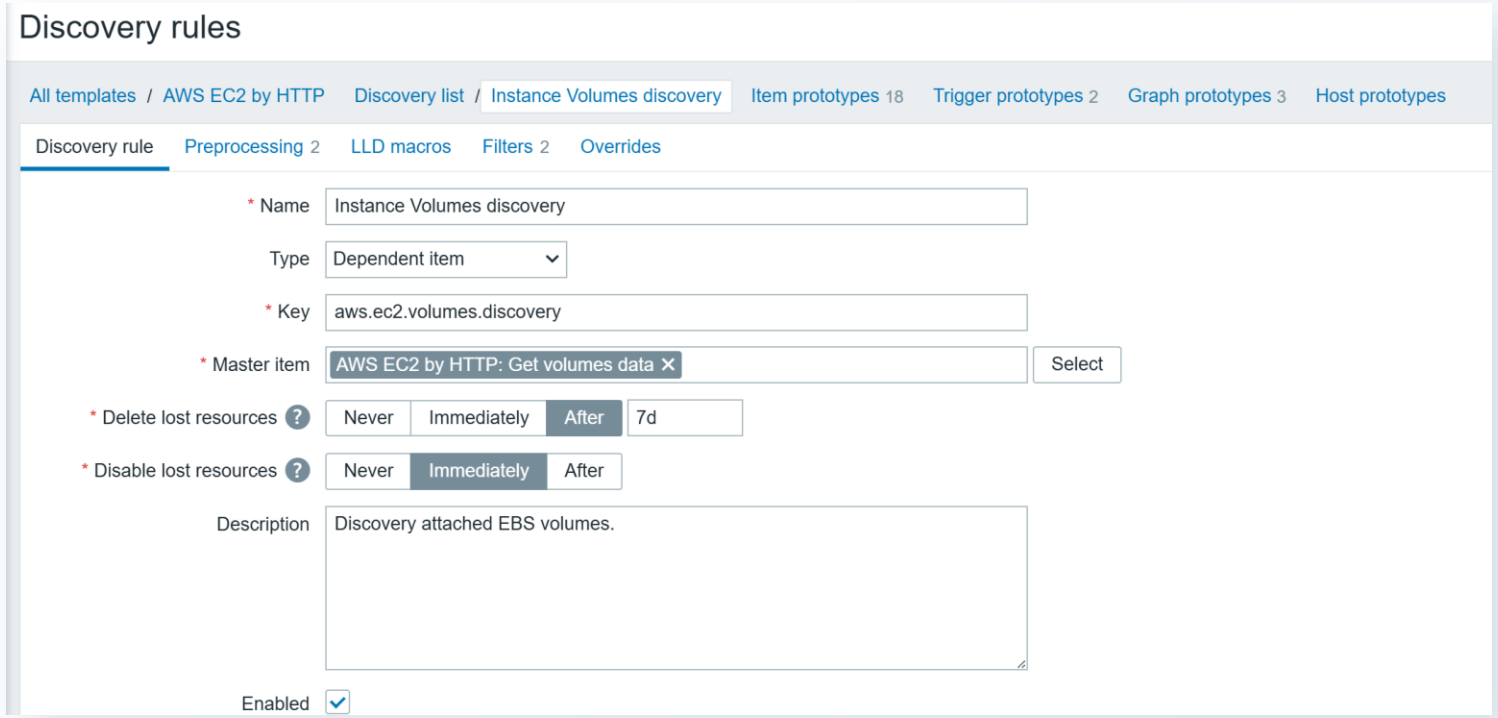
- Using dependent items can reduce data collection performance overhead
- The data is collected via a single request and processed by Zabbix
- Any type of preprocessing can be used to transform data
- Most commonly JSONPath/Xpath/Regex preprocessing is used
- Low-level discovery rules can also be of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

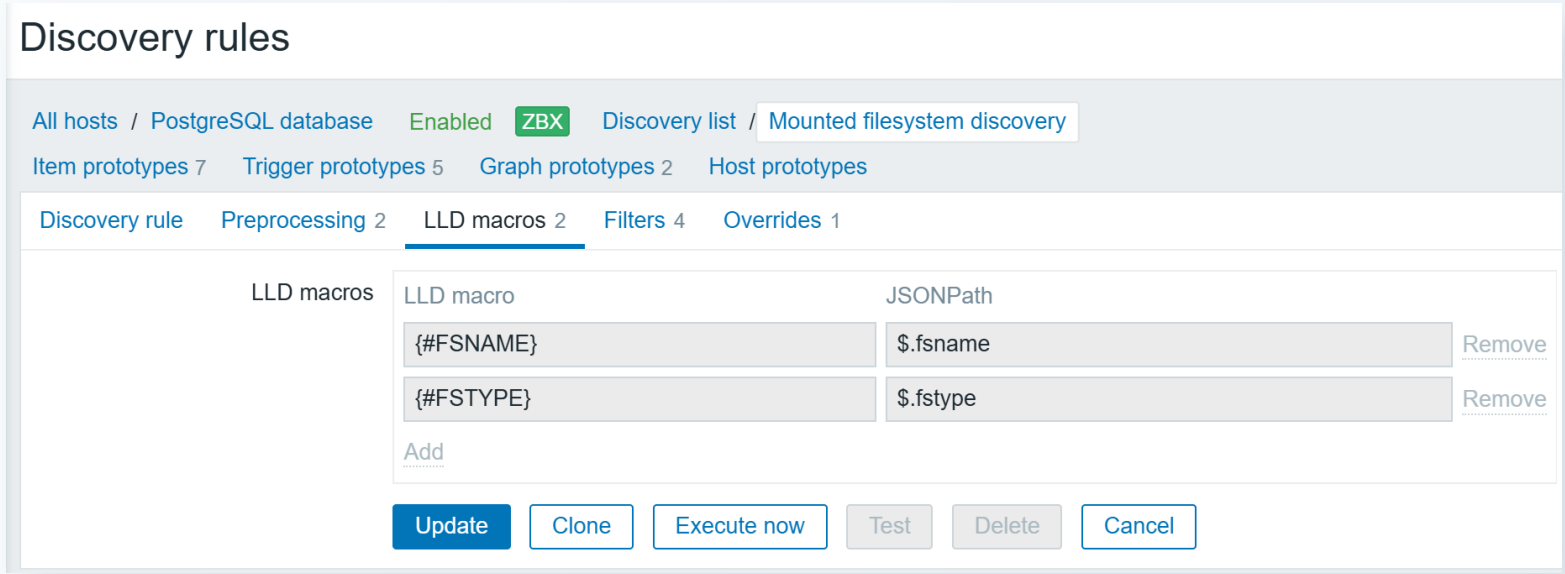


Dependent low-level discovery

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

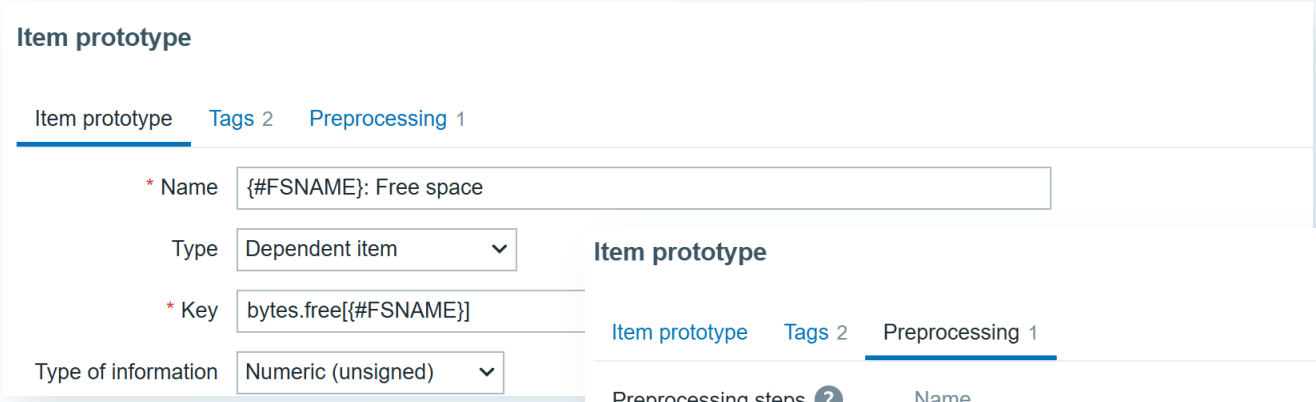

```
[
  {
    "fsname": "/",
    "fstype": "overlay",
    "bytes": {
      "total": 85829070848,
      "free": 49161748480,
      "used": 36667322368,
      "pfree": 57.27866793182881,
      "pused": 42.72133206817119
    },
    "inodes": {
      "total": 41941440,
      "free": 41664647,
      "used":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

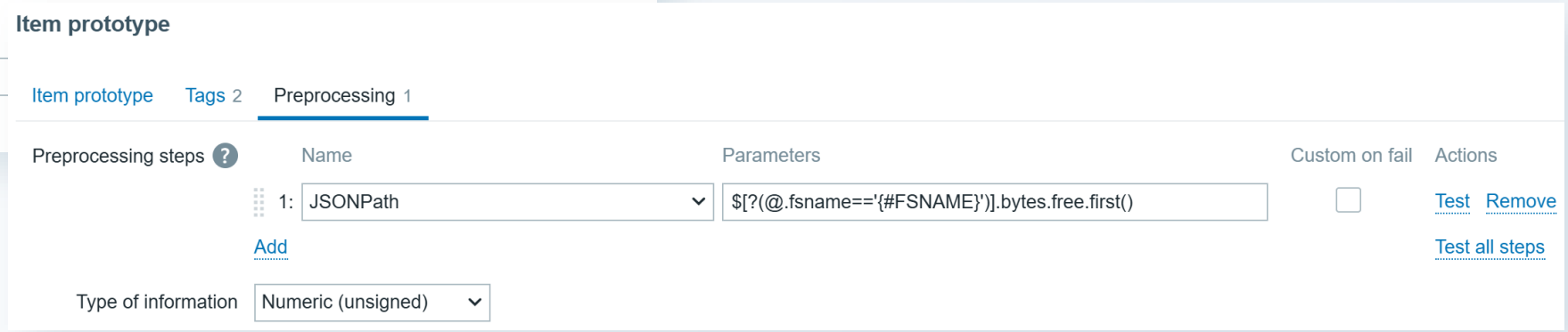



Dependent low-level discovery

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

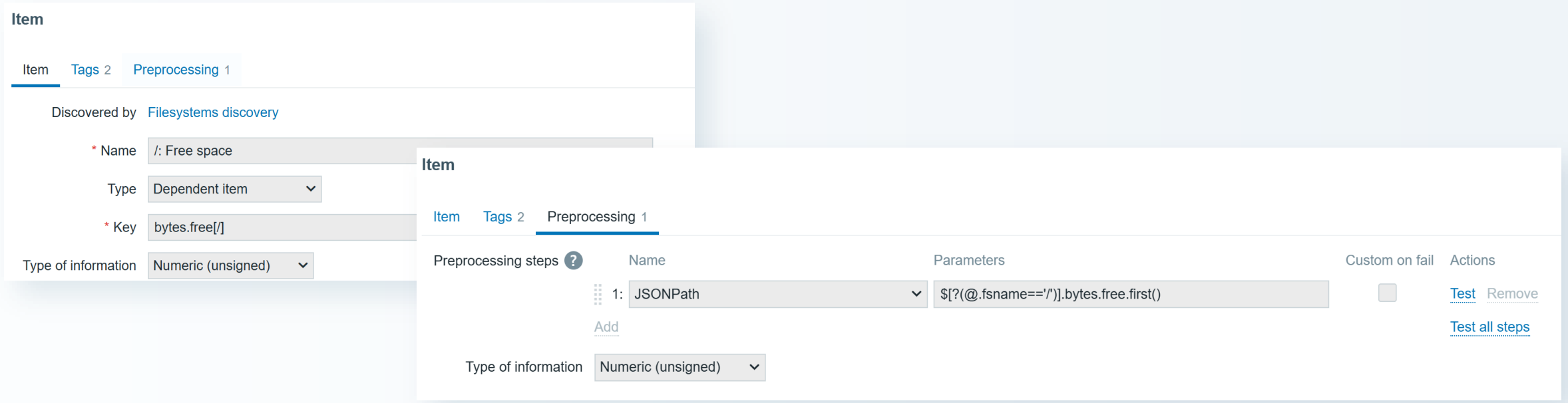




Dependent low-level discovery

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



Dependent SNMP low-level discovery

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

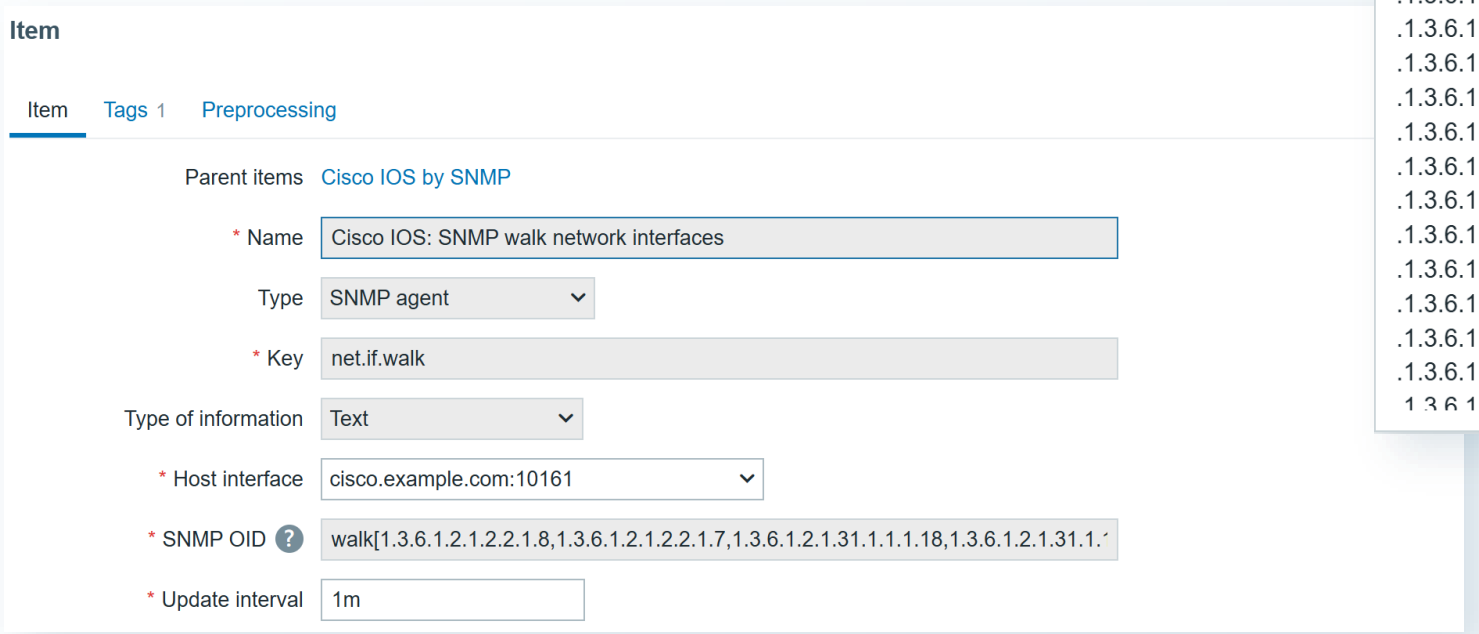
- Create a *walk[oid1,oid2,oid3, ...]* master item
- Create an LLD rule with SNMP walk to JSON preprocessing step
- Assign LLD macros in the preprocessing
- Create dependent item prototypes with SNMP walk value preprocessing step
- Apply *Discard uchnanged with heartbeat* preprocessing to reduce the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

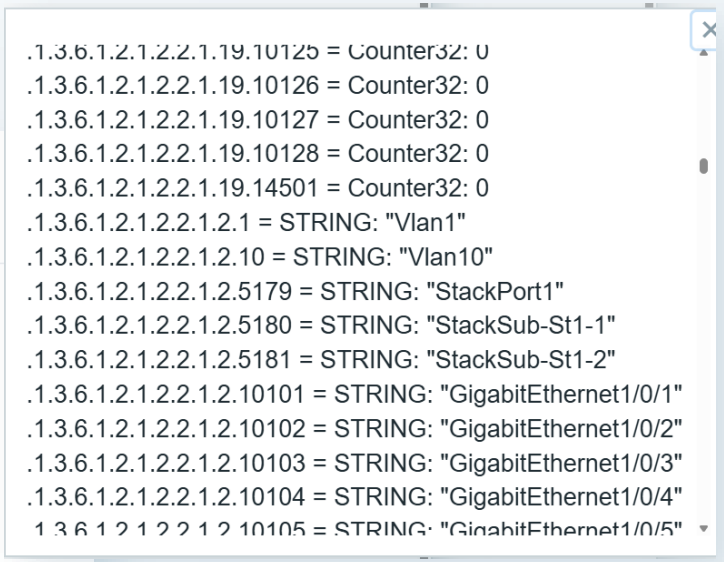
Dependent SNMP low-level discovery

ZABBIX

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

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

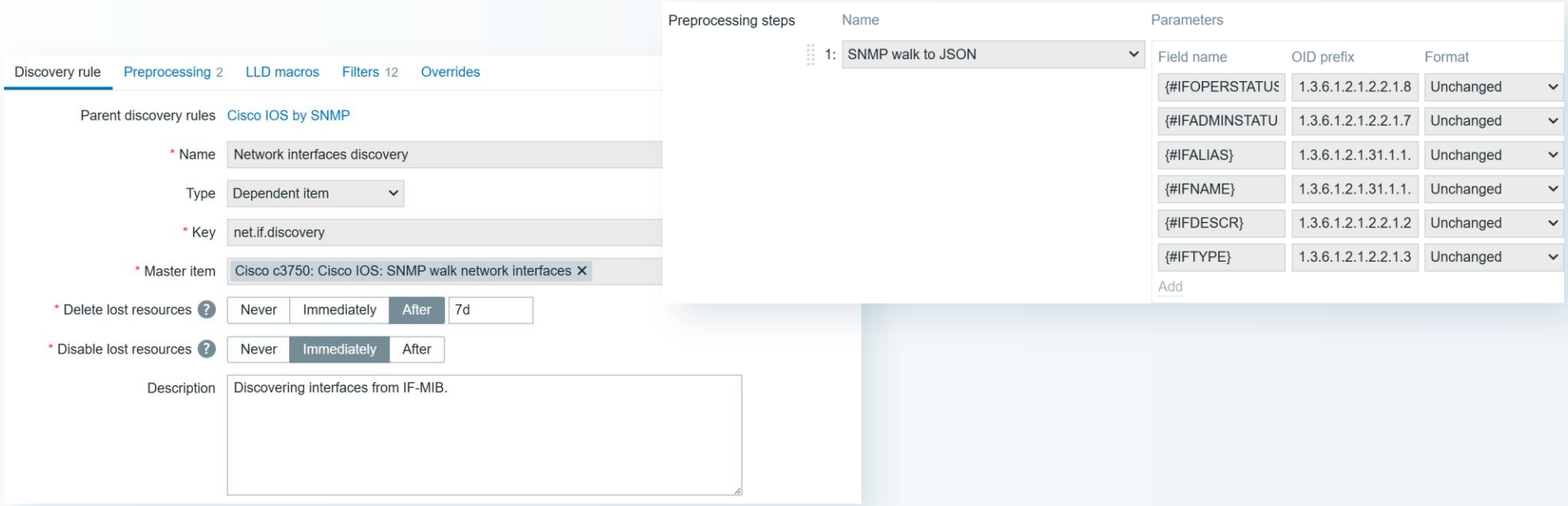




Dependent SNMP low-level discovery

Create a dependent LLD rule with SNMP walk to JSON preprocessing

- Assign LLD macro values to OIDs

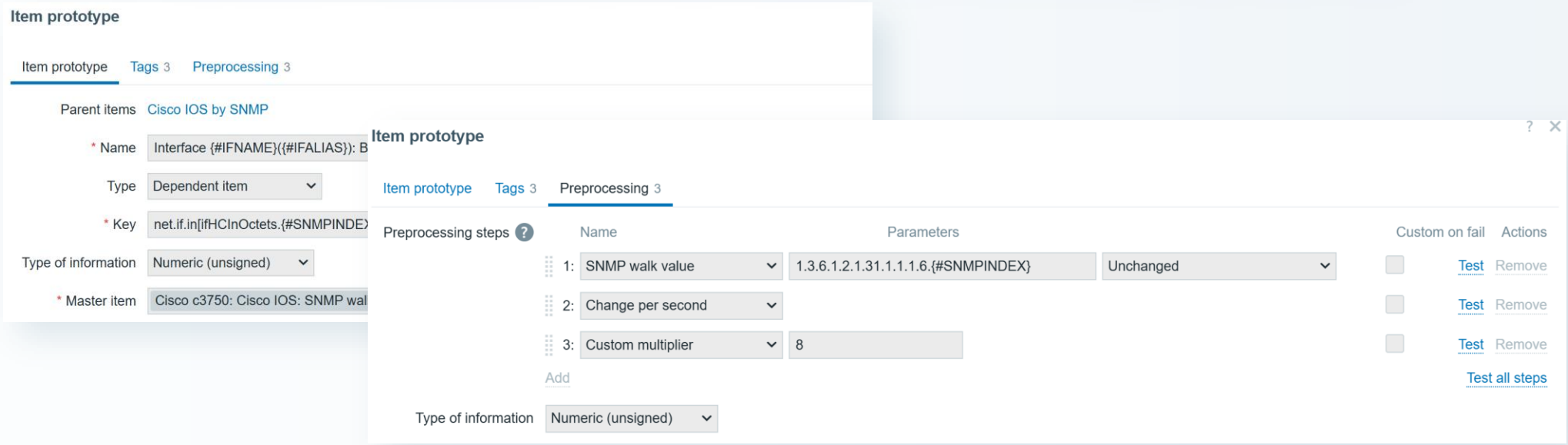


Dependent SNMP low-level discovery

ZABBIX

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

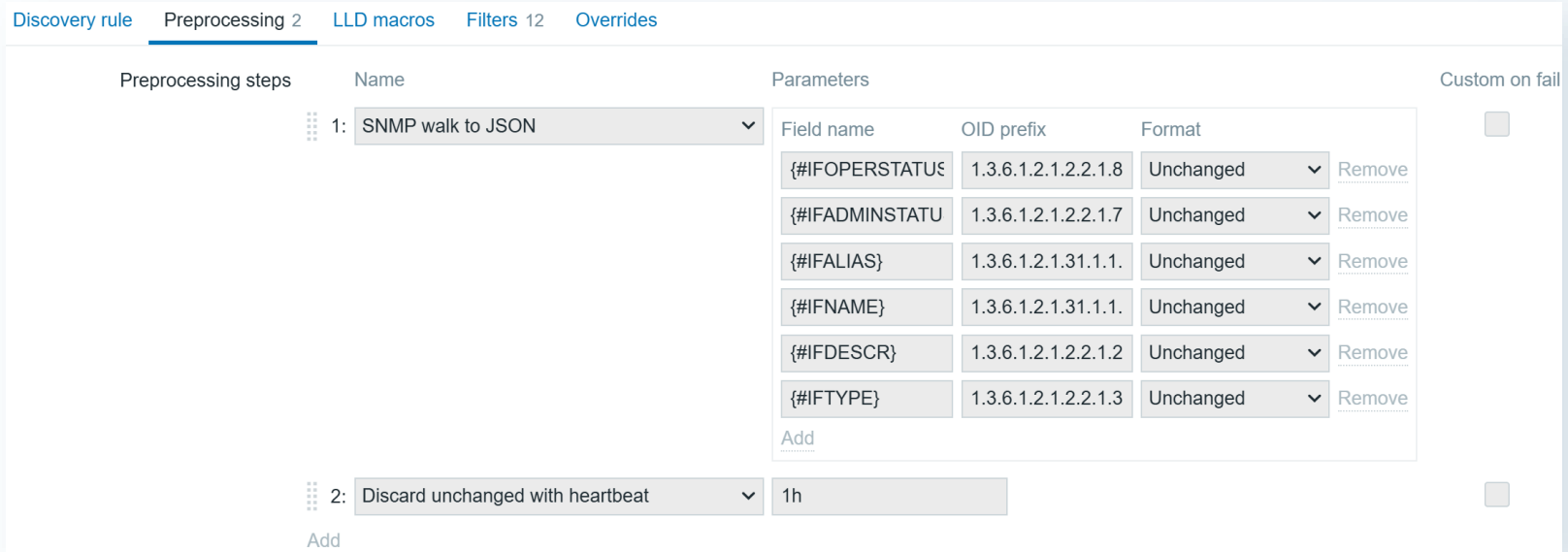


Dependent SNMP low-level discovery

ZABBIX

Apply *Discard unchaged* 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 dicarded
- This is unique throttling behavior which works only with SNMP walk to JSON preprocessing

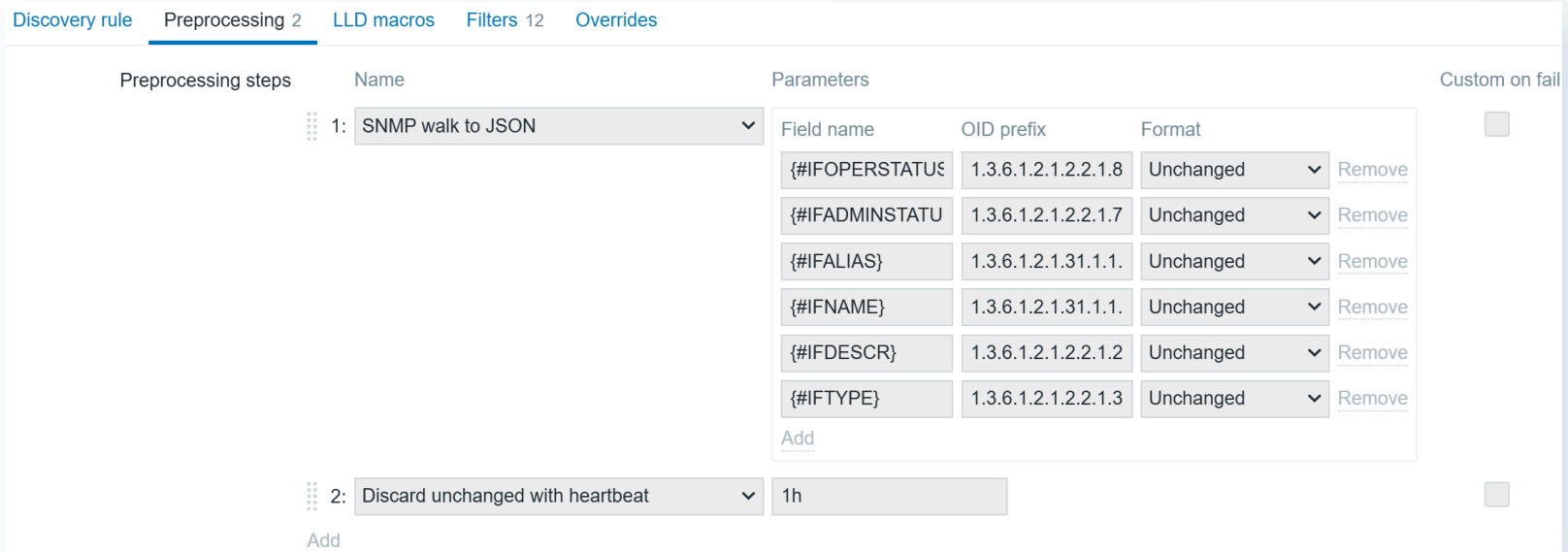


Dependent SNMP low-level discovery

ZABBIX

Apply *Discard unchaged* 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 dicarded
- This is unique throttling behavior which works only with SNMP walk to JSON preprocessing



Dependent SNMP low-level discovery

ZABBIX

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"
```

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"
```

Outbound packets discarded (Value collected by item}

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!