

## 4 Discovery of SNMP OIDs

### Overview

In this section we will perform an SNMP [discovery](#) on a switch.

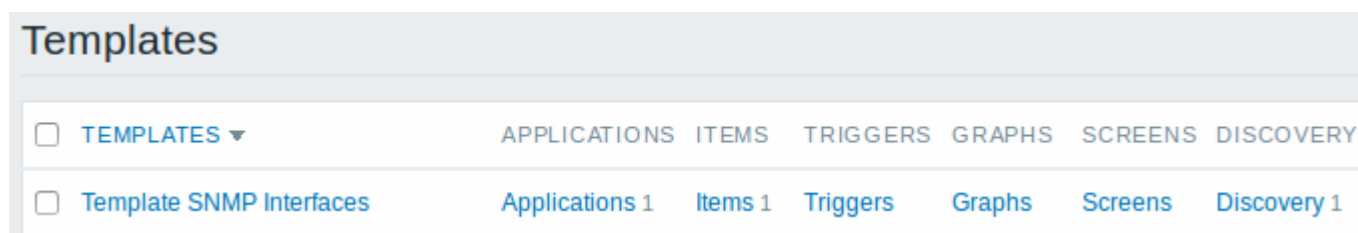
### Item key

Unlike with file system and network interface discovery, the item does not necessarily has to have an "snmp.discovery" key - item type of SNMP agent is sufficient.

Discovery of SNMP OIDs is supported since Zabbix server/proxy 2.0.

To configure the discovery rule, do the following:

- Go to: *Configuration* → *Templates*
- Click on *Discovery* in the row of an appropriate template



- Click on *Create discovery rule* in the upper right corner of the screen
- Fill in the discovery rule form with the required details as in the screenshot below

The screenshot shows the configuration for a discovery rule named "Network interfaces". The "Type" is set to "SNMPv2 agent" and the "Key" is "ifDescr". The "SNMP OID" field contains the expression "discovery[#{#IFDESCR},IF-MIB::ifDescr]". The "SNMP community" is set to "\${SNMP\_COMMUNITY}". The "Update interval" is "1h". Under "Custom intervals", there is one entry with "Type" set to "Flexible Scheduling", "Interval" set to "50s", and "Period" set to "1-7,00:00-24:00". The "Keep lost resources period" is "30d". The "Description" field contains the text: "You may also consider using IF-MIB::ifType or IF-MIB::ifAlias for discovery depending on your filtering needs. \${SNMP\_COMMUNITY} is a global macro." The "Enabled" checkbox is checked. At the bottom, there are "Add" and "Cancel" buttons.

The OIDs to discover are defined in SNMP OID field in the following format: `discovery[#{#MACRO1}, oid1, {#MACRO2}, oid2, ...,]`

where `{#MACRO1}`, `{#MACRO2}` ... are valid lld macro names and `oid1`, `oid2`... are OIDs capable of generating meaningful values for these macros. A built-in macro `{#SNMPINDEX}` containing index of the discovered OID is applied to discovered entities. The discovered entities are grouped by `{#SNMPINDEX}` macro value.

To understand what we mean, let us perform few snmpwalks on our switch:

```
$ 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

$ snmpwalk -v 2c -c public 192.168.1.1 IF-MIB::ifPhysAddress
IF-MIB::ifPhysAddress.1 = STRING: 8:0:27:90:7a:75
IF-MIB::ifPhysAddress.2 = STRING: 8:0:27:90:7a:76
```

```
IF-MIB::ifPhysAddress.3 = STRING: 8:0:27:2b:af:9e
```

And set SNMP OID to: `discovery[{-#IFDESCR}, ifDescr, {#IFPHYSADDRESS}, ifPhysAddress]`

Now this rule will discover entities with `{#IFDESCR}` macros set to **WAN**, **LAN1** and **LAN2**, `{#IFPHYSADDRESS}` macros set to **8:0:27:90:7a:75**, **8:0:27:90:7a:76**, and **8:0:27:2b:af:9e**, `{#SNMPINDEX}` macros set to the discovered OIDs indexes **1**, **2** and **3**:

```
{
  "data": [
    {
      "#{SNMPINDEX}": "1",
      "#{IFDESCR}": "WAN",
      "#{IFPHYSADDRESS}": "8:0:27:90:7a:75"
    },
    {
      "#{SNMPINDEX}": "2",
      "#{IFDESCR}": "LAN1",
      "#{IFPHYSADDRESS}": "8:0:27:90:7a:76"
    },
    {
      "#{SNMPINDEX}": "3",
      "#{IFDESCR}": "LAN2",
      "#{IFPHYSADDRESS}": "8:0:27:2b:af:9e"
    }
  ]
}
```

If an entity does not have the specified OID, then the corresponding macro will be omitted for this entity. For example if we have the following data:

```
ifDescr.1 "Interface #1"
ifDescr.2 "Interface #2"
ifDescr.4 "Interface #4"

ifAlias.1 "eth0"
ifAlias.2 "eth1"
ifAlias.3 "eth2"
ifAlias.5 "eth4"
```

Then in this case SNMP discovery `discovery[{-#IFDESCR}, ifDescr, {#IFALIAS}, ifAlias]` will return the following structure:

```
{
  "data": [
    {
      "#{SNMPINDEX}": 1,
      "#{IFDESCR}": "Interface #1",
      "#{IFALIAS}": "eth0"
    },
  ],
}
```

```
{
  "{#SNMPINDEX}": 2,
  "{#IFDESCR}": "Interface #2",
  "{#IFALIAS}": "eth1"
},
{
  "{#SNMPINDEX}": 3,
  "{#IFALIAS}": "eth2"
},
{
  "{#SNMPINDEX}": 4,
  "{#IFDESCR}": "Interface #4"
},
{
  "{#SNMPINDEX}": 5,
  "{#IFALIAS}": "eth4"
}
]
```

## Item prototypes

The following screenshot illustrates how we can use these macros in item prototypes:

**Item prototype** Preprocessing

Name Incoming traffic on interface \$1

Type SNMPv2 agent

Key ifInOctets[#{IFDESCR}]

SNMP OID IF-MIB::ifInOctets.#{SNMPINDEX}

SNMP community {\$SNMP\_COMMUNITY}

Port

Type of information Numeric (unsigned)

Units bps

Update interval 1m

Custom intervals		Type	Interval	Period
<input checked="" type="checkbox"/>	Flexible	Scheduling	50s	1-7,00:00-24:00

[Add](#)

History storage period 1w

Trend storage period 365d

Show value As is [show value mappings](#)

New application

Again, creating as many item prototypes as needed:

## Item prototypes

All templates / Template SNMP Interfaces    Discovery list / Network interfaces    **Item prototypes 8**

<input type="checkbox"/> NAME ▲	KEY	INTERVAL	HI
<input type="checkbox"/> Admin status of interface {#IFDESCR}	ifAdminStatus[{#IFDESCR}]	1m	7d
<input type="checkbox"/> Alias of interface {#IFDESCR}	ifAlias[{#IFDESCR}]	1h	7d
<input type="checkbox"/> Description of interface {#IFDESCR}	ifDescr[{#IFDESCR}]	1h	7d
<input type="checkbox"/> Inbound errors on interface {#IFDESCR}	ifInErrors[{#IFDESCR}]	1m	7d
<input type="checkbox"/> Incoming traffic on interface {#IFDESCR}	ifInOctets[{#IFDESCR}]	1m	7d
<input type="checkbox"/> Operational status of interface {#IFDESCR}	ifOperStatus[{#IFDESCR}]	1m	7d
<input type="checkbox"/> Outbound errors on interface {#IFDESCR}	ifOutErrors[{#IFDESCR}]	1m	7d
<input type="checkbox"/> Outgoing traffic on interface {#IFDESCR}	ifOutOctets[{#IFDESCR}]	1m	7d

## Trigger prototypes

The following screenshot illustrates how we can use these macros in trigger prototypes:

**Trigger prototype** Dependencies

Name:

Severity:  Not classified  Information  Warning  Average  High  Critical

Expression:

[Expression constructor](#)

OK event generation:  Expression  Recovery expression  None

PROBLEM event generation mode:  Single  Multiple

OK event closes:  All problems  All problems if tag values match

Tags: 

<input type="text" value="tag"/>	<input type="text" value="value"/>	<input type="button" value="Remove"/>
----------------------------------	------------------------------------	---------------------------------------

  
[Add](#)

Allow manual close:

URL:

Description:

Create enabled:

### Trigger prototypes

All templates / Template SNMP Interfaces Discovery list / Network interfaces Item prototypes 8

<input type="checkbox"/>	SEVERITY	NAME ▲	EXPR
<input type="checkbox"/>	Information	Operational status was changed on {HOST.NAME} interface {#IFDESCR}	{Temp

### Graph prototypes

The following screenshot illustrates how we can use these macros in graph prototypes:

**Graph prototype**
Preview

Name

Width

Height

Graph type

Show legend

Show working time

Show triggers

Percentile line (left)

Percentile line (right)

Y axis MIN value

Y axis MAX value

	Items	Name	Function	Draw st
⋮	1: Template SNMP Interfaces: Incoming traffic on interface {#IFDESCR}		<input style="width: 40px;" type="text" value="avg"/>	<input style="width: 40px;" type="text" value="Gradie"/>
⋮	2: Template SNMP Interfaces: Outgoing traffic on interface {#IFDESCR}		<input style="width: 40px;" type="text" value="avg"/>	<input style="width: 40px;" type="text" value="Gradie"/>

[Add](#) [Add prototype](#)

### Graph prototypes

[All templates / Template SNMP Interfaces](#)
[Discovery list / Network interfaces](#)
[Item prototypes 8](#)

	NAME ▲	WIDTH
<input type="checkbox"/>	Traffic on interface {#SNMPVALUE}	900

A summary of our discovery rule:

### Discovery rules

[All templates / Template SNMP Interfaces](#)
[Applications 1](#)
[Items 1](#)
[Triggers](#)
[Graphs](#)
[Screens](#)

	NAME ▲	ITEMS	TRIGGERS	GRAPHS	HO
<input type="checkbox"/>	Network interfaces	Item prototypes 8	Trigger prototypes 1	Graph prototypes 1	Ho



## Discovered entities

When server runs, it will create real items, triggers and graphs based on the values the SNMP discovery rule returns. In the host configuration they are prefixed with an orange link to a discovery rule they come from.

### Items

All hosts / Switch1 Enabled ZBX SNMP JMX IPMI Applications 1 Items 241 Triggers 30 Gr

Filter

<input type="checkbox"/>	Wizard	Name	Triggers	Key
<input type="checkbox"/>		Network interfaces: Admin status of interface 1		ifAdminStatus[1]
<input type="checkbox"/>		Network interfaces: Admin status of interface 2		ifAdminStatus[2]
<input type="checkbox"/>		Network interfaces: Admin status of interface 3		ifAdminStatus[3]
<input type="checkbox"/>		Network interfaces: Admin status of interface 4		ifAdminStatus[4]

### Triggers

All hosts / Switch1 Enabled ZBX SNMP JMX IPMI Applications 1 Items 241 Triggers 30 Gr

Filter

<input type="checkbox"/>	Severity	Name	Exp
<input type="checkbox"/>	Information	Network interfaces: Operational status was changed on {HOST.NAME} interface 1	{pr
<input type="checkbox"/>	Information	Network interfaces: Operational status was changed on {HOST.NAME} interface 2	{pr
<input type="checkbox"/>	Information	Network interfaces: Operational status was changed on {HOST.NAME} interface 3	{pr
<input type="checkbox"/>	Information	Network interfaces: Operational status was changed on {HOST.NAME} interface 4	{pr

### Graphs

Group all

All hosts / Switch1 Enabled ZBX SNMP JMX IPMI Applications 1 Items 241 Triggers 30 Gr

<input type="checkbox"/>	Name
<input type="checkbox"/>	Network interfaces: Traffic on interface 1
<input type="checkbox"/>	Network interfaces: Traffic on interface 2
<input type="checkbox"/>	Network interfaces: Traffic on interface 3
<input type="checkbox"/>	Network interfaces: Traffic on interface 4

From: <https://www.zabbix.com/documentation/3.4/> - **Zabbix Documentation 3.4**

Permanent link: [https://www.zabbix.com/documentation/3.4/manual/discovery/low\\_level\\_discovery/snmp\\_oids?rev=1498124732](https://www.zabbix.com/documentation/3.4/manual/discovery/low_level_discovery/snmp_oids?rev=1498124732)

Last update: **2017/06/22 09:45**

