YumaPro yp-snmp Manual

YANG-Based Unified Modular Automation Tools

Simple Network Management Protocol [SNMP]

Version 18.10-11

Table Of Contents

1 Preface

1.1 Legal Statements

1.2 Additional Resources

This document assumes you have successfully set up the software as described in the printed document:

YumaPro Installation Guide

Other documentation includes:

YumaPro Quickstart Guide

YumaPro User Manual

YumaPro netconfd-pro Manual

YumaPro yangcli-pro Manual

YumaPro yangdiff-pro Manual

YumaPro yangdump-pro Manual

YumaPro Developer Manual

YumaPro API Quickstart Guide

YumaPro ypgnmi Guide

YumaPro ypclient-pro Manual

YumaPro yp-system API Guide

YumaPro yp-show API Guide

YumaPro Yocto Linux Quickstart Guide

To obtain additional support you may contact YumaWorks technical support department:

support@yumaworks.com

1.2.1 WEB Sites

- **YumaWorks**
  - https://www.yumaworks.com
  - Offers support, training, and consulting for YumaPro.
- **Netconf Central**
  - http://www.netconfcentral.org/
  - Free information on NETCONF and YANG, tutorials, on-line YANG module validation and documentation database
- **Yang Central**
  - http://www.yang-central.org

- Free information and tutorials on YANG, free YANG tools for download

- **NETCONF Working Group Wiki Page**
  - http://trac.tools.ietf.org/wg/netconf/trac/wiki
  - Free information on NETCONF standardization activities and NETCONF implementations
- **NETCONF WG Status Page**
  - http://tools.ietf.org/wg/netconf/
  - IETF Internet draft status for NETCONF documents
- **libsmi Home Page**
  - http://www.ibr.cs.tu-bs.de/projects/libsmi/
  - Free tools such as smidump, to convert SMIv2 to YANG

1.2.2 Mailing Lists

- **NETCONF Working Group**
  - https://mailarchive.ietf.org/arch/browse/netconf/
  - Technical issues related to the NETCONF protocol are discussed on the NETCONF WG mailing list. Refer to the instructions on https://www.ietf.org/mailman/listinfo/netconf for joining the mailing list.
- **NETMOD Working Group**
  - https://datatracker.ietf.org/wg/netmod/documents/
  - Technical issues related to the YANG language and YANG data types are discussed on the NETMOD WG mailing list. Refer to the instructions on the WEB page for joining the mailing list.

1.3 Conventions Used in this Document

The following formatting conventions are used throughout this document:

Documentation Conventions

Convention	Description
--foo	CLI parameter foo
<foo>	XML parameter foo
foo	**yangcli-pro** command or parameter
$FOO	Environment variable FOO
$$foo	**yangcli-pro** global variable foo
`some text`	Example command or PDU
some text	Plain text
 Informational text	Useful or expanded information
 Warning text	Warning information indicating possibly unexpected side-effects

2 yp-snmp User Guide

Architectural Components

2.1 Introduction

yp-snmp enables the Simple Network Management Protocol (SNMP) to join the other **netconfd-pro** Northbound interfaces. It does this by linking to the Open Source project Net-SNMP library. This user manual describes how the SNMP function is used, how to convert MIB modules to YANG modules, instrument them, install them on the **netconfd-pro** server, and then access them with SNMP client tools (agents).

2.1.1 Features

The **yp-snmp** client has the following features:

- SNMP packet processing within the **netconfd-pro** server by integrating the **libnetsnmp** packet processing within the netconfd-pro server (agent library).
- Creating a mapping of MIB ODI to object templates by reading the YANG modules
- SNMP GET request processing
- SNMP GETNEXT request processing
- SNMP GETBULK request processing
- Asynchronous Notifications – traps & informs
- Support for SNMPv3

2.1.2 Building SNMP support

In order to link Net-SNMP to **netconfd-pro** the Net-SNMP header files have to be installed on the system you build the server. Also, to run the **netconfd-pro** server with SNMP support both the **snmpd** and **snmptrapd** must be available. To test the SNMP support having the client (agent) tools provide by Net-SNMP, such as snmpget, snmpwalk, snmpbulkget, etc. would be useful to have installed.

The following instructions will install Net-SNMP and its client tools. NOTE: there are many parameters for building Net-SNMP, this is only one of them. For other options please refer to http://www.net-snmp.org/

First download the version of Net-SNMP you wish to use. The following instructions use net-snmp-5.7.3 as an example. This will install the binaries and .h header files needed:

```
tar -zxvf net-snmp-5.7.3.tar.gz
cd net-snmp-5.7.3
./configure --with-defaults --disable-embedded-perl --without-perl-modules
make
sudo make install
```

When you have Net-SNMP installed then you can build the server. Use the WITH_SNMP=1 flag to build **netconfd-pro** with SNMP support from the source code:

```
sudo make EVERYTHING=1 WITH_SNMP=1 uninstall
make EVERYTHING=1 WITH_SNMP=1 distclean
make EVERYTHING=1 WITH_SNMP=1
sudo make EVERYTHING=1 WITH_SNMP=1 install
```

To test the SNMP client features, GET, WALK, etc., the IF-MIB has been included and built as a Server Instrumentation Library (SIL) and you will need to build and install the IF-MIB SIL. From the netconf directory:

```
cd libif-mib
make
sudo make install
```

To run **netconfd-pro** server you should launch it with the parameters below to allow you to see the debug messages as the examples are running and also avoid any issues with existing configurations. The load-module command loads the IF-MIB SIL described previously:

```
sudo netconfd-pro with-snmp=true module=IF-MIB
```

NOTE: the server needs to be run at the root level as it uses restricted ports as part of the SNMP standard.

2.1.3 snmpget example

To run **snmpget** against the loaded IF-MIB SIL:

```
snmpget -v 2c -c public localhost 1.3.6.1.2.1.2.1.0
IF-MIB::ifNumber.0 = INTEGER: 32
```

2.1.4 snmpwalk example

To run **snmpwalk** against the loaded IF-MIB SIL:

```
snmpwalk -v 2c -c public localhost 1.3.6.1.2.1.2.2
IF-MIB::ifIndex.20 = INTEGER: 20
IF-MIB::ifIndex.25 = INTEGER: 25
IF-MIB::ifIndex.30 = INTEGER: 30
IF-MIB::ifDescr.25 = STRING: Description string
IF-MIB::ifDescr.30 = STRING: Test string
IF-MIB::ifType.20 = INTEGER: softwareLoopback(24)
IF-MIB::ifType.25 = INTEGER: softwareLoopback(24)
IF-MIB::ifType.30 = INTEGER: softwareLoopback(24)
IF-MIB::ifMtu.20 = INTEGER: 1111
IF-MIB::ifMtu.25 = INTEGER: 2222
IF-MIB::ifMtu.30 = INTEGER: 1111
IF-MIB::ifSpeed.20 = Gauge32: 2030
IF-MIB::ifSpeed.25 = Gauge32: 10
IF-MIB::ifSpeed.30 = Gauge32: 2030
IF-MIB::ifPhysAddress.20 = STRING: aa:bb:cc:dd:0:ff
IF-MIB::ifPhysAddress.25 = STRING: aa:bb:cc:dd:0:ff
IF-MIB::ifPhysAddress.30 = STRING: aa:bb:cc:dd:0:ff
IF-MIB::ifAdminStatus.20 = INTEGER: testing(3)
IF-MIB::ifAdminStatus.25 = INTEGER: testing(3)
IF-MIB::ifAdminStatus.30 = INTEGER: testing(3)
IF-MIB::ifOperStatus.20 = INTEGER: testing(3)
IF-MIB::ifOperStatus.25 = INTEGER: testing(3)
IF-MIB::ifOperStatus.30 = INTEGER: testing(3)
IF-MIB::ifLastChange.20 = Timeticks: (33) 0:00:00.33
IF-MIB::ifLastChange.25 = Timeticks: (33) 0:00:00.33
IF-MIB::ifLastChange.30 = Timeticks: (33) 0:00:00.33
IF-MIB::ifInOctets.20 = Counter32: 2
IF-MIB::ifInOctets.25 = Counter32: 2
IF-MIB::ifInOctets.30 = Counter32: 2
IF-MIB::ifInUcastPkts.20 = Counter32: 3
IF-MIB::ifInUcastPkts.25 = Counter32: 3
IF-MIB::ifInUcastPkts.30 = Counter32: 3
IF-MIB::ifInDiscards.20 = Counter32: 5
IF-MIB::ifInDiscards.25 = Counter32: 5
IF-MIB::ifInDiscards.30 = Counter32: 5
IF-MIB::ifInErrors.20 = Counter32: 6
IF-MIB::ifInErrors.25 = Counter32: 6
IF-MIB::ifInErrors.30 = Counter32: 6
IF-MIB::ifInUnknownProtos.20 = Counter32: 7
IF-MIB::ifInUnknownProtos.25 = Counter32: 7
IF-MIB::ifInUnknownProtos.30 = Counter32: 7
IF-MIB::ifOutDiscards.20 = Counter32: 9
IF-MIB::ifOutDiscards.25 = Counter32: 9
IF-MIB::ifOutDiscards.30 = Counter32: 9
IF-MIB::ifOutErrors.20 = Counter32: 10
IF-MIB::ifOutErrors.25 = Counter32: 10
IF-MIB::ifOutErrors.30 = Counter32: 10
```

2.1.5 snmpbulkget example

To run **snmpbulkget** against the loaded IF-MIB SIL:

```
snmpbulkget -v 2c -Cn1 -Cr24 -c public localhost \
      IF-MIB::ifDescr.2 IF-MIB::ifAdminStatus.2
IF-MIB::ifAdminStatus.20 = INTEGER: testing(3)
IF-MIB::ifAdminStatus.25 = INTEGER: testing(3)
IF-MIB::ifAdminStatus.30 = INTEGER: testing(3)
IF-MIB::ifOperStatus.20 = INTEGER: testing(3)
IF-MIB::ifOperStatus.25 = INTEGER: testing(3)
IF-MIB::ifOperStatus.30 = INTEGER: testing(3)
IF-MIB::ifLastChange.20 = Timeticks: (33) 0:00:00.33
IF-MIB::ifLastChange.25 = Timeticks: (33) 0:00:00.33
IF-MIB::ifLastChange.30 = Timeticks: (33) 0:00:00.33
IF-MIB::ifInOctets.20 = Counter32: 2
IF-MIB::ifInOctets.25 = Counter32: 2
IF-MIB::ifInOctets.30 = Counter32: 2
IF-MIB::ifInUcastPkts.20 = Counter32: 3
IF-MIB::ifInUcastPkts.25 = Counter32: 3
IF-MIB::ifInUcastPkts.30 = Counter32: 3
IF-MIB::ifInDiscards.20 = Counter32: 5
IF-MIB::ifInDiscards.25 = Counter32: 5
IF-MIB::ifInDiscards.30 = Counter32: 5
IF-MIB::ifInErrors.20 = Counter32: 6
IF-MIB::ifInErrors.25 = Counter32: 6
IF-MIB::ifInErrors.30 = Counter32: 6
IF-MIB::ifInUnknownProtos.20 = Counter32: 7
IF-MIB::ifInUnknownProtos.25 = Counter32: 7
IF-MIB::ifInUnknownProtos.30 = Counter32: 7
```

2.1.6 Traps and Informs

NOTE: currently only SNMP Traps Version 2 are supported by the server.

To demonstrate SNMP traps make sure the following line exists in the snmpd.conf file:

```
# send v2 traps
trap2sink localhost public 162
```

A server is needed to collect and display the traps. To run the server Net-SNMP provides execute:

```
sudo snmptrapd -d -f -Lo:
```

To test that receiving notifications is working simulate sending an SNMP trap from a second terminal session using the following command, which sends a linkDown notification:

```
snmptrap -v 2c -c public localhost '' 1.3.6.1.6.3.1.1.5.3 \
      ifIndex i 2 ifAdminStatus i 1 ifOperStatus i 1
```

In the trap server terminal window you should see:

```
sudo snmptrapd -d -f -Lo:
NET-SNMP version 5.7.3

Received 119 byte packet from UDP: [127.0.0.1]:47154->[127.0.0.1]:162
0000: 30 75 02 01  01 04 06 70  75 62 6C 69  63 A7 68 02    0u.....public.h.
0016: 04 31 08 5C  A0 02 01 00  02 01 00 30  5A 30 0F 06    .1.\.......0Z0..
0032: 08 2B 06 01  02 01 01 03  00 43 03 45  C2 E2 30 17    .+.......C.E..0.
0048: 06 0A 2B 06  01 06 03 01  01 04 01 00  06 09 2B 06    ..+...........+.
0064: 01 06 03 01  01 05 03 30  0E 06 09 2B  06 01 02 01    .......0...+....
0080: 02 02 01 01  02 01 02 30  0E 06 09 2B  06 01 02 01    .......0...+....
0096: 02 02 01 07  02 01 01 30  0E 06 09 2B  06 01 02 01    .......0...+....
0112: 02 02 01 08  02 01 01                                 .......
                        .......

2017-07-22 10:40:21 localhost [UDP: [127.0.0.1]:47154->[127.0.0.1]:162]:
DISMAN-EVENT-MIB::sysUpTimeInstance = Timeticks: (4571874) 12:41:58.74     SNMPv2-
MIB::snmpTrapOID.0 = OID: IF-MIB::linkDown   IF-MIB::ifIndex = INTEGER: 2  IF-
MIB::ifAdminStatus = INTEGER: up(1)   IF-MIB::ifOperStatus = INTEGER: up(1)
```

If this works, then you can test with **netconfd-pro**.

3 SNMP security and SNMP v3

This section describes briefly the security aspects for SNMP requests specifically about authentication and authorization. The authentication mechanism is built into Net-SNMP

- Authentication in SNMP Versions 1 and 2c is provided by a password (community string) sent in clear text between a manager and agent.
- SNMP v3 defines a number of security-related capabilities. The initial specifications defined the USM and VACM, which were later followed by a transport security model that provided support for SNMPv3 over SSH and SNMPv3 over TLS and DTLS.

Netconfd-pro implements NACM (NETCONF Access Control Model) to manage and control the access to YANG objects supported by the device. Since NACM already provides the authorization, VACM has to be disabled when processing SNMP v3 requests. More information about the configuration and management of Net-SNMP authentication is available on-line as part of Net-SNMP documentation.

3.1 Security configuration files

Net-SNMP makes use of 2 configuration files to control its operation and the management information provided.

1. /var/net-snmp/snmpd.conf – This file contains the SNMP v3 specific configuration related to allowed user names and passwords.
2. /usr/local/share/snmp/snmpd.conf – This file contains generic configuration information including SNMP v1 and v2c related community strings that perform basic authentication.

3.2 Adding SNMP v3 user

Adding a new SNMP v3 user can be performed by using the scripts available as part of Net-SNMP as below. The command below adds a user “admin” with authentication and privacy. Authentication makes use of SHA and the password for authentication is “password1”. Similarly for privacy, DES is used and the associated password for privacy is “password2”.

```
sudo net-snmp-config --create-snmpv3-user -ro -a SHA -A 'password1' \
      -x DES -X 'password2' admin
```

Note: The **netconfd-pro** server must be stopped before running the above command. Once the command above is run, then the **netconfd-pro** can be run again which will make use of this updated configuration file.

3.3 Adding SNMP v1/v2c user

As mentioned earlier, SNMP v1 and v2c make use of community strings for authentication. The allowed community strings along with the access permissions is configured in the /usr/local/share/snmp/snmpd.conf file. The tokens that control these parameters are “rocommunity” for read only access and “rwcommunity” for read-write access.

4 netconfd-pro Hooks Into Net-SNMP

The **netconfd-pro** server is always listening on port 161 and 162 for SNMP agent requests when the server is started with with-snmp=true. During boot time the **netconfd-pro** creates Trap sinks and enables the **netconfd-pro** SNMP server. This includes:

- SNMP configuration file parsing
- Registering a handler for incoming SNMP packets. This is the callback registered for incoming packets
- Registering the Network Service Address Point (NSAP) with the net-snmp library and setup an agent session on the given transport. In this step **netconfd-pro** links the net-snmp library and registers all the needed callbacks and handlers that will be used for packet handling, PDU creation, and reply output

Then the server starts to check if there is any SNMP messages to process. It checks for any packets from the network. If there are any packets to process the server calls the net-snmp API to process them.

For SNMP requests, e.g. **snmpget** on a get2 node, the server will perform the following:

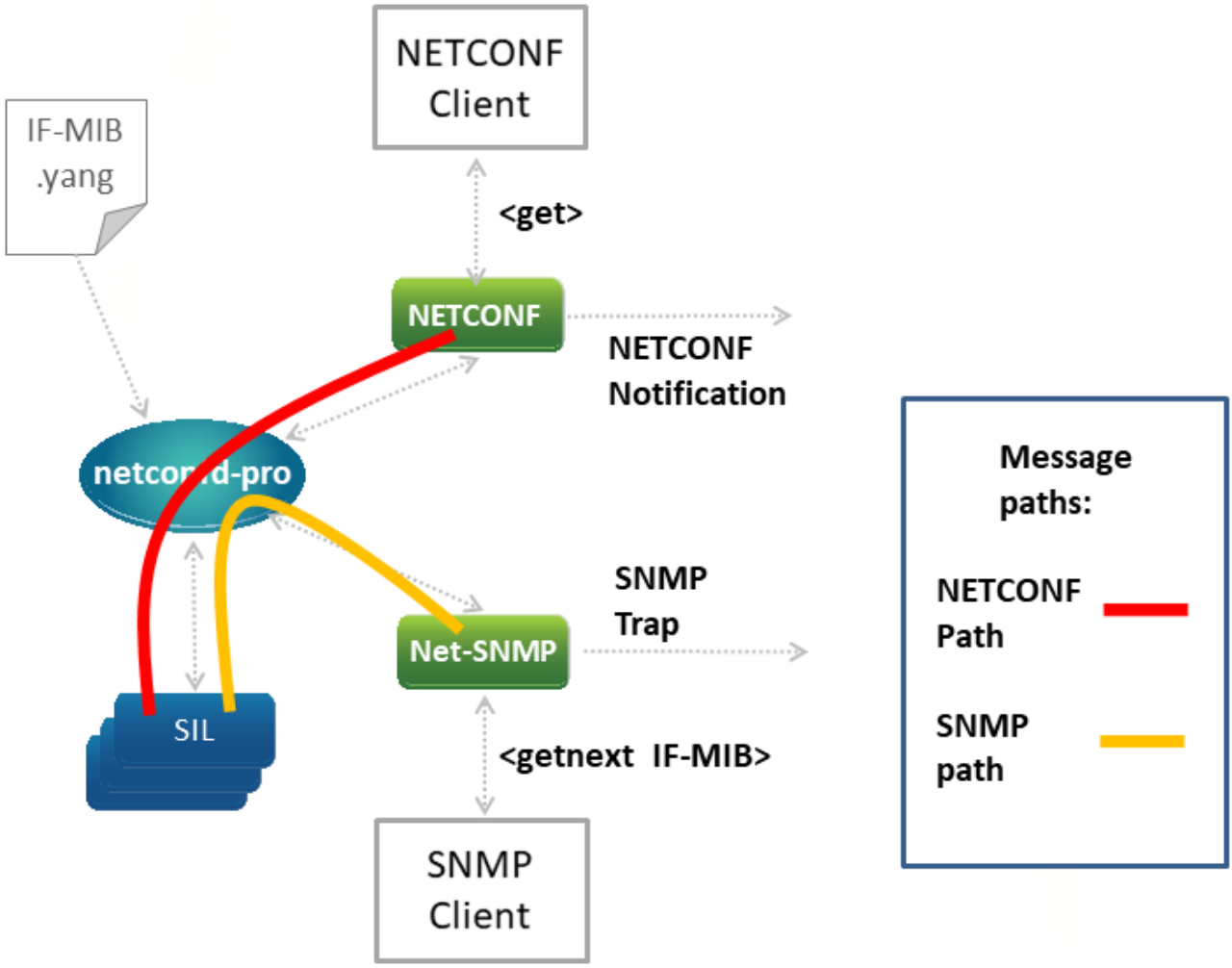
- Parse the incoming packet (OID; request type ,get, getnext, etc)
- Resolve the internal SNMP request type, request on indexed node, on scalar with no any indexes, etc., based on the request type the server will adjust the target object resolution
- Then the server will either try to find the best next OID and repeat the same steps or proceed to the actual value retrieval
- In order to get the get2 value, the server calls the get2 callbacks starting from the table of the target node - the target node will always be a leaf

Based on the callback's results, the server creates a new PDU to return, sets the return value(s) for the requested Varbind list in that PDU, and sends that packet back to the agent.

For config true and virtual nodes all the steps are the same except the server does not call get2 callbacks, it locates the requested Table in the database first, during the RESTCONF parse path processing, and after that the best value is retrieved from that Table.

4.1 yp-snmp – NETCONF and SNMP Message Paths

Message Paths Diagram



When converted MIB modules are loaded into **netconfd-pro** server the Northbound protocols, such as NETCONF, access the YANG datastores in the usual way, i.e. through the message path colored red in the above diagram. Any notifications are handled in the usual way.

SNMP messages are processed by the Net-SNMP process and the netconfd-pro server with the SIL providing the instrumentation, i.e. the message path colored yellow in the diagram above. The server generates any SNMP Traps necessary.

5 Creating MIB Instrumentation

To convert a MIB module to a YANG module and add Server Instrumentation Library (SIL) code the following steps should be followed. The example below uses the IF-MIB. An example version of the IF-MIB SIL is provided with YumaPro SDK.

1. Convert the selected MIB module to a YANG module using the smidump tool from: https://www.ibr.cs.tu-bs.de/projects/libsmi/download.html?lang=de

```
smidump -f yang --yang-smi-extensions IF-MIB > IF-MIB.yang
```

2. You should validate the conversion using **yangdump-pro**. If you want to supply additional parameters to **yangdump-pro** for your environment see the user manual yumapro-yangdump-manual.pdf or man pages.

```
yangdump-pro IF-MIB > ~/yangdump_output.log
```

3. Copy the yang files into your work folder.
4. Run **make_sil_dir_pro** to generates the instrumentation source code.

```
make_sil_dir_pro –split IF-MIB
```

5. Modify the instrumentation code as needed. You will see the tag that say “insert xxx code”. The process of converting the MIB to YANG creates smi:oid "x.y.z" tags in the YANG module for leaf instrumentation. Only the leafs with the smi:oid tag will be seen from an SNMP client. See the following section “SNMP to YANG mapping”

6. Once you’ve finished with the instrumentation code compile the code using

```
make DEBUG=1
```

7. install the code using

```
sudo make DEBUG=1 install
```

NOTE: "DEBUG=1" is an optional and used to enable debug logging.

This will install the generated library in the system path for **netconfd-pro** to load.

5.1 SNMP to YANG mapping

Only YANG leaf objects that have the smi:oid "x.y.z" tags will be visible for **netconfd-pro** SNMP engine. All other objects will be ignored, and the server will report that there is no such an object or will jump into the next object in case of **snmpgetnext**.

YANG data model cannot be utilized in full if it needs to represent MIB module. After the MIB to YANG conversion the YANG module will have several limitations and some of the regular YANG features and properties will be either ignored by the **netconfd-pro** server or even invalid. The following list illustrates the limitations:

- List or container may not have an OID number (smi:oid "x.y.z" tags), since they may not have analogy in the MIB modules
- The generic architecture for the YANG module must always be /container/list/leaf or /container/leaf in case of scalar objects. There should not be nested structures, nested architecture. Although, the **netcond-pro** server is capable to handle the complex nested architecture, it is still not recommended.
- Choice, case statements and their leaf nodes are ignored and will be invisible to **netconfd-pro** SNMP server
- If an object has a "deprecated" statement or the "status" is not current, e.g.: "obsolete", the object will be ignored
- Leafref, augment, uses, etc. are all allowed to be present in the SNMP YANG module