Simple network management protocol

SNMP working:

SNMP version 1, which is the SNMP standard supported by PRTG Network Monitor, was the initial development of the SNMP protocol. A description can be found in Request for Comments (RFC) 1157. It operates over User Datagram Protocol (UDP). SNMP v1 is considered the de facto network management protocol in the Internet community.

SNMP works on the basis that network management systems send out a request and the managed devices return a response. This is implemented using one of four operations: Get, GetNext, Set, and Trap. SNMP messages consist of a header and a PDU (protocol data units). The headers consist of the SNMP version number and the community name. The community name is used as a form of security in SNMP. The PDU depends on the type of message that is being sent. The Get, GetNext, and Set, as well as the response PDU, consist of PDU type, Request ID, Error status, Error index and Object/variable fields. The Trap consist of Enterprise, Agent, Agent address, Generic trap type, Specific trap code, Timestamp and Object/Value fields.

MIB:

MIBs are a collection of definitions which define the properties of the managed object within the device to be managed (such as a router, switch, etc.) Each managed device keeps a database of values for each of the definitions written in the MIB.

In order for all of this to be properly organized, all of the manageable features of all products (from each vendor) are arranged in this tree. Each “branch” of this tree has a number and a name, and the complete path from the top of the tree down to the point of interest forms the name of that point. This is the OID.

SNMP BASICS

Some general SNMP terminology that you may come across:

• Agent: A process that monitors devices for problems and sends alerts to a monitoring station.

• Community Strings: Like passwords. See earlier in this article for more information.

• Get: SNMP sends a get request to device it monitors to retrieve specific information.

• Get Next: A getnext command is used by SNMP to get the next variable in a set – for instance in a table.

• Informs: Like Traps, but sent by the agent with a request for the management station to confirm receipt. Supported in v2 and up.

• Management Station (aka Network Management Station or NMS): The software configured to receive and collect information sent from SNMP agents.

• MIB: Management Information Base. See the beginning of this article for more information on MIBs.

• MIB Browser (or MIB Walker): A tool that can pull data from SNMP enabled devices, helping to identify which objects respond to a query.

• Notification: Same as a “Trap”. V2c and up use the term Notifications to refer to a Trap.

• OID: Object Identifier. See the beginning of this article for more information on OIDs.

• Object: The things SNMP gathers information about. Examples are Interface status, or CPU utilization.

• Polling: An NMS will poll, or ask devices for their status regularly.

• Set: SNMP can use a Set command to change settings on a device.

• Trap: SNMP sends a trap, or unsolicited message sent from an agent to a management station when some important event is detected. This is the opposite of polling.

• Variable: Variables are the actual status of an object – up/down, 90% CPU used, etc.

• Version 1: Original version of SNMP, community strings sent in plain text, very weak security.

• Version 2c: SNMP v2c was developed to fix some of the problems in v1. However multiple versions were developed, none truly addressing the problems with v1. V2c is the most common flavour, and has enhanced protocol handling over v1, resulting in slightly improved operations. However, security is still an issue because it uses plain-text community strings.

• Version 3: The newest version of SNMP, v3 supports full security and authentication. Should be used if possible, especially on untrusted networks.

This section provides examples of how to use the following SNMP commands:

| **Topics** |  |
| --- | --- |
| **Description** | **Links** |
| Command examples | * [snmpget Command](https://docs.oracle.com/cd/E19201-01/820-6413-13/SNMP_commands_reference_appendix.html" \l "50446362_67519)  * [snmpwalk Command](https://docs.oracle.com/cd/E19201-01/820-6413-13/SNMP_commands_reference_appendix.html" \l "50446362_54136)  * [snmpbulkwalk Command](https://docs.oracle.com/cd/E19201-01/820-6413-13/SNMP_commands_reference_appendix.html" \l "50446362_61760)  * [snmptable Command](https://docs.oracle.com/cd/E19201-01/820-6413-13/SNMP_commands_reference_appendix.html" \l "50446362_24951)  * [snmpset Command](https://docs.oracle.com/cd/E19201-01/820-6413-13/SNMP_commands_reference_appendix.html" \l "50446362_51144)  * [snmptrapd Command](https://docs.oracle.com/cd/E19201-01/820-6413-13/SNMP_commands_reference_appendix.html" \l "50446362_61509) |

|  |
| --- |
| **Note -**All command examples given in this section are executed on the SNMP management station, unless instructions indicate otherwise. |

|  |
| --- |
| **Note -**The example SNMP commands presented in this appendix are based on the Net-SNMP sample applications and, therefore, will only work as presented if you have Net-SNMP and the Net-SNMP sample applications installed. |

Before using SNMP, be sure to install the ILOM MIBs files in the directory where net-snmp tools load MIBs or where your SNMP tool of choice loads MIBs. See the following URL for additional information on net-snmp:

[http://net-snmp.sourceforge.net/wiki/index.php/](http://net-snmp.sourceforge.net/wiki/index.php/TUT:Using_and_loading_MIBS)

[TUT:Using\_and\_loading\_MIBS](http://net-snmp.sourceforge.net/wiki/index.php/TUT:Using_and_loading_MIBS)

**snmpget Command**

**snmpget -mALL -v1 -cpublic** *snmp\_agent\_Ip\_address* **sysName.0**

As stated in the description of the sysName.0 MIB object in the SNMPv2-MIB, this command returns an administratively assigned name for this managed node. By convention, this is the node’s fully-qualified domain name. If the name is unknown, the value returned is the zero-length string.

For example:

|  |
| --- |
| % **snmpget -v2c -cprivate -mALL** *snmp\_agent\_Ip\_address* **sysName.0 sysObjectID.0 ilomCtrlDateAndTime.0**  SNMPv2-MIB::sysName.0 = STRING: SUNSPHOSTNAME  SNMPv2-MIB::sysObjectID.0 = OID: SUN-ILOM-SMI-MIB::sunILOMSystems  SUN-ILOM-CONTROL-MIB::ilomCtrlDateAndTime.0 = STRING: 2007-12-10,20:33:32.0 |

In addition to the sysName.0 object, this command displays the content of the sysObjectID.0 and the ilomCtrlDateAndTime.0 MIB objects. Notice that the MIB file name is given for each MIB object as part of the reply.

The following descriptions of the MIB objects are taken from the MIB files.

* sysName - An administratively assigned name for this managed node. By convention, this is the node’s fully-qualified domain name. If the name is unknown, the value is the zero-length string.
* sysObjectID - The vendor’s authoritative identification of the network management subsystem contained in the entity. This value is allocated within the SMI enterprises sub-tree (1.3.6.1.4.1) and provides an easy and unambiguous means for determining ‘what kind of box’ is being managed.
* ilomCtrlDataAndTime - The date and time of the device.

**snmpwalk Command**

The snmpwalk command performs a sequence of chained GETNEXT requests automatically. It is a work saving command. Rather than having to issue a series of snmpgetnext requests, one for each object ID, or node, in a sub-tree, you can simply issue one snmpwalk request on the root node of the sub-tree and the command gets the value of every node in the sub-tree.

For example:

|  |
| --- |
| % **snmpwalk -mALL -v1 -cpublic** *snmp\_agent\_Ip\_address* **system**  SNMPv2-MIB::sysDescr.0 = STRING: ILOM machine custom description  SNMPv2-MIB::sysObjectID.0 = OID: SUN-ILOM-SMI-MIB::sunILOMSystems  DISMAN-EVENT-MIB::sysUpTimeInstance = Timeticks: (16439826) 1 day, 21:39:58.26  SNMPv2-MIB::sysContact.0 = STRING: set via snmp test  SNMPv2-MIB::sysName.0 = STRING: SUNSPHOSTNAME  SNMPv2-MIB::sysLocation.0 = STRING:  SNMPv2-MIB::sysServices.0 = INTEGER: 72  SNMPv2-MIB::sysORLastChange.0 = Timeticks: (14) 0:00:00.14  SNMPv2-MIB::sysORID.1 = OID: IF-MIB::ifMIB  SNMPv2-MIB::sysORID.2 = OID: SNMPv2-MIB::snmpMIB  SNMPv2-MIB::sysORID.3 = OID: TCP-MIB::tcpMIB  SNMPv2-MIB::sysORID.4 = OID: RFC1213-MIB::ip  SNMPv2-MIB::sysORID.5 = OID: UDP-MIB::udpMIB  SNMPv2-MIB::sysORID.6 = OID: SNMP-VIEW-BASED-ACM-MIB::vacmBasicGroup  SNMPv2-MIB::sysORID.7 = OID: SNMP-FRAMEWORK-MIB::snmpFrameworkMIBCompliance  SNMPv2-MIB::sysORID.8 = OID: SNMP-MPD-MIB::snmpMPDCompliance  SNMPv2-MIB::sysORID.9 = OID: SNMP-USER-BASED-SM-MIB::usmMIBCompliance  SNMPv2-MIB::sysORDescr.1 = STRING: The MIB module to describe generic objects for network interface sub-layers  SNMPv2-MIB::sysORDescr.2 = STRING: The MIB module for SNMPv2 entities  SNMPv2-MIB::sysORDescr.3 = STRING: The MIB module for managing TCP implementations  SNMPv2-MIB::sysORDescr.4 = STRING: The MIB module for managing IP and ICMP implementations  SNMPv2-MIB::sysORDescr.5 = STRING: The MIB module for managing UDP implementations  SNMPv2-MIB::sysORDescr.6 = STRING: View-based Access Control Model for SNMP.  SNMPv2-MIB::sysORDescr.7 = STRING: The SNMP Management Architecture MIB.  SNMPv2-MIB::sysORDescr.8 = STRING: The MIB for Message Processing and Dispatching.  SNMPv2-MIB::sysORDescr.9 = STRING: The management information definitions for the SNMP User-based Security Model.  SNMPv2-MIB::sysORUpTime.1 = Timeticks: (1) 0:00:00.01  SNMPv2-MIB::sysORUpTime.2 = Timeticks: (2) 0:00:00.02  SNMPv2-MIB::sysORUpTime.3 = Timeticks: (2) 0:00:00.02  SNMPv2-MIB::sysORUpTime.4 = Timeticks: (2) 0:00:00.02  SNMPv2-MIB::sysORUpTime.5 = Timeticks: (2) 0:00:00.02  SNMPv2-MIB::sysORUpTime.6 = Timeticks: (2) 0:00:00.02  SNMPv2-MIB::sysORUpTime.7 = Timeticks: (14) 0:00:00.14  SNMPv2-MIB::sysORUpTime.8 = Timeticks: (14) 0:00:00.14  SNMPv2-MIB::sysORUpTime.9 = Timeticks: (14) 0:00:00.14 |

**snmptable Command**

The snmptable command retrieves the contents of an SNMP table and displays the contents in a tabular format, that is, one table row at a time, such that the resulting output resembles the table being retrieved. This is contrasted with the snmpwalk command, which displays the contents of the table one column at a time.

Here is an example of the snmptable command:

|  |
| --- |
| % **snmptable -mALL -v2c -cprivate** *snmp\_agent\_Ip\_address* **sysORTable**  SNMP table: SNMPv2-MIB::sysORTable  sysORID  sysORDescr  sysORUpTime  IF-MIB::ifMIB  The MIB module to describe generic objects for network interface sub-layers.  0:0:00:00.01  SNMPv2-MIB::snmpMIB  The MIB module for SN MPv2 entities.  0:0:00:00.02  TCP-MIB::tcpMIB  The MIB module for managing TCP implementations.  0:0:00:00.02  RFC1213-MIB::ip  The MIB module for managing IP and ICMP implementations.  0:0:00:00.02  UDP-MIB::udpMIB  The MIB module for managing UDP implementations.  0:0:00:00.02  SNMP-VIEW-BASED-ACM-MIB::vacmBasicGroup  View-based Access Control Model for SNMP.  0:0:00:00.02  SNMP-FRAMEWORK-MIB::snmpFrameworkMIBCompliance  The SNMP Management Architecture MIB.  0:0:00:00.14  SNMP-MPD-MIB::snmpMPDCompliance  The MIB for Message Processing and Dispatching.  0:0:00:00.14  SNMP-USER-BASED-SM-MIB::usmMIBCompliance  The management information definitions for the SNMP User-based Security Model.  0:0:00:00.14 |

**snmpset Command**

While the syntax of the snmpset command is similar to that of the snmpget command, the commands are quite different. The snmpget command merely reads the value of the specified object ID, while the snmpset command writes the value specified to the object ID. Further, along with the value to be written to the object ID, you must also specify the data type of the object ID in the snmpset command because SNMP objects support more than one data type.

The following example shows how use of the snmpget and snmpset commands together. The sequence of steps is as follows:

**1. Use the**snmpget**command to check to current value of the MIB object.**

**2. Use the**snmpset**command to change the value of the MIB object.**

**3. Use the**snmpget**command to verify that the MIB object was in fact changed to the requested value.**

|  |
| --- |
| % **snmpget -mALL -v2c -cprivate** *snmp\_agent\_Ip\_address* **ilomCtrlHttpEnabled.0**  SUN-ILOM-CONTROL-MIB::ilomCtrlHttpEnabled.0 = INTEGER: false(2)  % **snmpset -mALL -v2c -cprivate** *snmp\_agent\_Ip\_address* **ilomCtrlHttpEnabled.0 i 1**  SUN-ILOM-CONTROL-MIB::ilomCtrlHttpEnabled.0 = INTEGER: true(1)  % **snmpget -mALL -v2c -cprivate** *snmp\_agent\_Ip\_address* **ilomCtrlHttpEnabled.0**  SUN-ILOM-CONTROL-MIB::ilomCtrlHttpEnabled.0 = INTEGER: true(1) |

Note that if you try to execute this snmpset command using a public community, instead of private, it will not work. This is because the private community has write permission, but the public community does not. The Reason code returned by the command does not make this clear because it simply states that the object is not writable.

Here is an example:

|  |
| --- |
| % **snmpset -mALL -v2c -cpublic** *snmp\_agent\_Ip\_address* **ilomCtrlHttpEnabled.0 i 1**  Error in packet.  Reason: notWritable (That object does not support modification) |

**snmptrapd Command**

snmptrapd is an SNMP application that receives and logs SNMP Trap and Inform messages. Before your system can receive such messages, you must configure the trap daemon to listen for these messages.

To configure a trap daemon, perform the following steps:

**1. Configure an SNMP trap destination.**

For an example, see [Configuring an snmptrapd Daemon](https://docs.oracle.com/cd/E19201-01/820-6413-13/SNMP_commands_reference_appendix.html#50446362_71644).

**2. Start the trap receiver application, snmptrapd.**

For an example, see [Starting the Trap Daemon](https://docs.oracle.com/cd/E19201-01/820-6413-13/SNMP_commands_reference_appendix.html#50446362_17772).

**3. Generate a test trap to verify that traps are being sent by the agent and received by the trap receiver.**

For an example, see [Testing the Trap Daemon](https://docs.oracle.com/cd/E19201-01/820-6413-13/SNMP_commands_reference_appendix.html#50446362_70207).

|  |
| --- |
| **procedure icon  Configuring an snmptrapd Daemon** |

The following example shows how to use the snmpset command to configure an snmptrapd daemon:

|  |
| --- |
| % **snmpset -mALL -v2c -cprivate** *snmp\_agent\_Ip\_address* **ilomCtrlAlertSeverity.1 i 2 ilomCtrlAlertType.1 i 2 ilomCtrlAlertDestinationIP.1 a** *dest\_Ip\_address*  SUN-ILOM-CONTROL-MIB::ilomCtrlAlertSeverity.1 = INTEGER: critical(21)  SUN-ILOM-CONTROL-MIB::ilomCtrlAlertType.1 = INTEGER: snmptrap(2)  SUN-ILOM-CONTROL-MIB::ilomCtrlAlertDestinationIP.1 = IpAddress: dest\_Ip\_address |

|  |
| --- |
| **procedure icon  Starting the Trap Daemon** |

The following example shows how to use the snmptrapd command start a trap daemon:

|  |
| --- |
| % **snmptrapd -mALL -Lo -f -t -OvQ -e -F "%H.%J.%K:%W:%w %q from %A:%V,% %v\n"**  2007-11-29 13:21:07 NET-SNMP version 5.2.3 Started. |

|  |
| --- |
| **procedure icon  Testing the Trap Daemon** |

While the daemon is running, log in to the CLI on the host that is running the SNMP agent and type the following command:

-> **set /SP/alertmgmt/rules testalert=true**

|  |
| --- |
| **Note -**It is important to test the trap daemon to make sure it is configured properly. |

The following screen shows a sample output when a testalert trap is received:

|  |
| --- |
| SUN-ILOM-CONTROL-MIB::ilom.103.2.1.20.0 = STRING: "This is a test trap" |

Network Management Using Telnet/ Cll and TL1

Two major network management protocols: CLlandTLl . CLl is an old protocol but is still widely used. We have explained some of the measures that have been taken to allow CLl to remain in use for so long. The aim has been to allow the reader to understand the role of CLl and TLl and also to have a basic technical understanding of these protocols. CommandLineInterface(CLI) has traditionally been an essential interface to any network device making CLI the de facto interface for communication with devices via a serial port. The originators of CLI intended it to be a low-overhead way of managing devices. This is reflected in its lightweight design. CLI has been around for so long , it is still the most prevalent interface for management Connectivity to devices.

Device management can be a very big task when it involves managing devices with large numbers of commands. The AdventNet CLI API simplifies this via APIs, which hide the complexity of the commands from the user. The API also allows applications to be built to manage any type of CLI device and to perform network management tasks such as monitoring network status, collecting network statistics and issuing network device control commands. The CLI API also allows the CLI protocol to be run over any transport protocol.

The CLI APIs can be used to build a management console for device management where:

• Devices are remotely managed from a terminal over IP in an Ethernet based network, where the terminal is connected to a data switch via Telnet or TCP.

• A terminal has local access via a serial port to a telecom switch.

• A remote terminal has remote HypertextTransferProtocol(HTTP) access to a Wide AreaNetwork(WAN) router.

CLIbrowser

CLI Browser is a graphical user interface (GUI)-based network management application, which can manage any CLI device. Its features include the ability to load and use files, Command Set and Data Set, with different sets of input messages and Different configuration settings.

Application developers use Extensible Markup Language(XML) to write Command Set and Data Set files for commands that are to be sent to the device. The Command Set and Data Set files are loaded into the CLI Browser from the data directory. Telnet is used as a standard transport protocol to query remote devices. Thus, Telnet and CLI work together wherever CLI is used.

**Telnet** is an [application protocol](http://en.wikipedia.org/wiki/Application_protocol) used on the [Internet](http://en.wikipedia.org/wiki/Internet) or [local area networks](http://en.wikipedia.org/wiki/Local_Area_Network) to provide a bidirectional interactive text-oriented communication facility using a virtual [terminal](http://en.wikipedia.org/wiki/Text_terminal) connection over TCP.

Historically, Telnet provided access to a [command-line interface](http://en.wikipedia.org/wiki/Command-line_interface) (usually, of an [operating system](http://en.wikipedia.org/wiki/Operating_system)) on a remote host. Most network equipment and [operating systems](http://en.wikipedia.org/wiki/Operating_system) with a [TCP/IP stack](http://en.wikipedia.org/wiki/TCP/IP_stack) support a Telnet service for remote configuration.

CLI Agent uses Telnet as one of the standard protocols for communication between the manager application and the CLI devices, such as switches and routers. Telnet stands for Telecommunications Network. This protocol provides a way for users (clients) to connect to multi-user computers (servers) on the Internet, whether in the next building or across the other side of the world. In most cases, users use Telnet to communicate with a remote login service.

The Telnet protocol gives the ability to connect to a machine, by giving commands and instructions interactively to that machine. In such a case, the local system becomes transparent to the user, who gets the feeling that he is connected directly to the remote computer. The commands typed by the user are transmitted directly to the remote machine and the response from the remote machine is displayed on the user’s monitor screen.

On the Internet, the ability to connect with another machine is made possible by the Transmission Control Protocol (TCP), which enables two machines to transmit data back and forth in a manner coherent to the operating systems of each device, and the Internet Protocol (IP),

IntroductiontoTL1

Transaction Language One(TU) is the most widely used management protocol in telecommunications. It manages most broadband and access networks and is increasingly being used for newer management applications, such as end-to-end service provisioning.

**Transaction Language 1** (TL1) is a widely used management protocol in [telecommunications](http://en.wikipedia.org/wiki/Telecommunications). It is a cross-vendor, cross-technology [man-machine language](http://en.wikipedia.org/wiki/MML_%28language%29), and is widely used to manage optical ([SONET](http://en.wikipedia.org/wiki/SONET)) and broadband access infrastructure in [North America](http://en.wikipedia.org/wiki/North_America). TL1 is used in the input and output messages that pass between [Operations Support Systems](http://en.wikipedia.org/wiki/Operations_support_system) (OSSs) and Network Elements (NEs).

TL1 software provides an ASCII-based protocol that allows the exchange of messages between an Operations Support System (OSS) and an Operating System. The TL1 software also allows the exchange of messages between the OSS and the network equipment that connects to the Operations System (OS).

**Transport Methods**

The TL1 software uses the following methods for transporting messages:

1. X.25 packet network (Network Management)
2. Transmission Control Protocol/Internet Protocol (TCP/IP) Telnet socket

The three basic requirements for a management interface are:

1. The ability to develop new interfaces quickly

2. The ability to upgrade the management interface to an existing element as new

Features are added to it:

3. The ability of a deployed interface to keep up with performance and usability

Requirements of the management systems and network elements that it connects.

As with SNMP and CLI, the TU protocol facilitates communication between a Managed device, a device with a TU agent and a TUmanager. The TU agent on the managed device serves to provide access to data stored on the managed device.

The TUmanager uses this access to monitor and control the managed device. The TU Browser is a GUI application that can be used to query TU devices. It allows the user to view and operate on data that are available through a TU agent on a managed device, and manage the device.

Some of the functions that can be performed through the TU Browser are:

• multiple command set and data set loading

• setting the parser options for a particular session

• support for sending messages as strings

• support for receiving partial TU messages

• support for debugging TLl messages

• support for checking drop out connections

• support for receiving Autonomous messages.

The TLl Browser can be used to debug the messages transferred between the TLl Browser and a TLl device. The debug window can be used to check messages sent and received between the TLl Browser and agent.