Chapter 32

Simple Network Management Protocol (SNMP)

Introduction

The Simple Network Management Protocol (SNMP) is the network management protocol of choice for the Internet and IP-based internetworks.

This chapter describes the main features of SNMP Version 1 (SNMPv1), SNMP Version 2c (SNMPv2c) and Version 3 (SNMPv3). It also describes support for SNMP on the switch, and how to configure the switch's SNMP agent. See Appendix C, SNMP MIBs for a detailed description of all MIBs (Management Information Bases) and MIB objects supported by the switch.

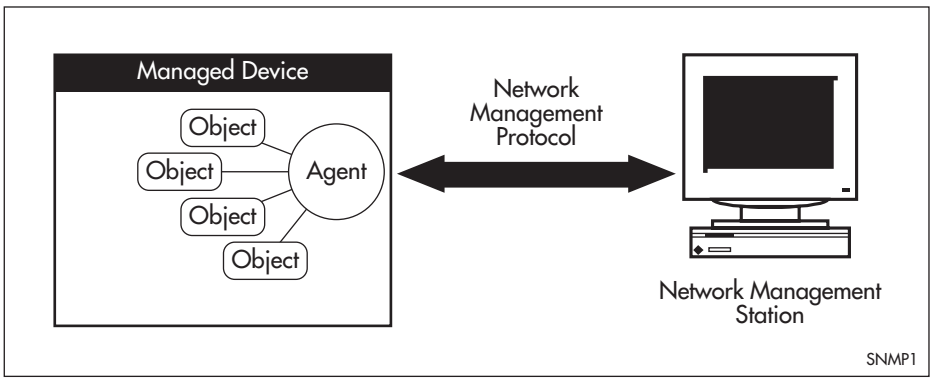
Unless a particular version of SNMP is named, "SNMP" in this chapter refers to versions SNMPv1, SNMPv2c and SNMPv3.

Network Management Framework

A network management system has the following components (Figure 32-1 on page 32-3):

- One or more *managed devices*, each containing an agent that provides the management functions. A managed device may be any computing device with a network capability, for example, a host system, workstation, terminal server, printer, router, switch, bridge, hub or repeater.
- One or more *Network Management Stations* (NMS). An NMS is a host system running a network management protocol and network management applications, enabling the user to manage the network.
- A *network management protocol* used by the NMS and agents to exchange information.

Figure 32-1: Components of a network management system



The *Internet-standard Network Management Framework* is the framework used for network management in the Internet. The framework was originally defined by the following documents:

- RFC 1155, *Structure and identification of management information for TCP/IP-based internets* (referred to as the SMI), details the mechanisms used to describe and name the objects to be managed.
- RFC 1213, *Management Information Base for network management of TCP/IP-based internets: MIB-II* (referred to as MIB-II), defines the core set of managed objects for the Internet suite of protocols. The set of managed objects can be extended by adding other MIBs specific to particular protocols, interfaces or network devices.

- RFC 1157, *A Simple Network Management Protocol (SNMP)*, is the protocol used for communication between management stations and managed devices.

Subsequent documents that have defined SNMPv2c are:

- RFC 1901, *Introduction to Community-based SNMPv2*
- RFC 1902, *Structure of Management Information for Version 2 of the Simple Network Management Protocol (SNMPv2)*
- RFC 1903, *Textual Conventions for Version 2 of the Simple Network Management Protocol (SNMPv2)*
- RFC 1904, *Conformance Statements for Version 2 of the Simple Network Management Protocol*
- RFC 1905, *Protocol Operations for Version 2 of the Simple Network Management Protocol (SNMPv2)*
- RFC 1906, *Transport Mappings for Version 2 of the Simple Network Management Protocol (SNMPv2)*
- RFC 1907, *Management Information Base for Version 2 of the Simple Network Management Protocol (SNMPv2)*
- RFC 2576, *Coexistence between Version 1, Version 2, and Version 3 of the Internet-standard Network Management Framework*
- RFC 2578, *Structure of Management Information Version 2 (SMIv2)*
- RFC 2579, *Textual Conventions for SMIv2*
- RFC 2580, *Conformance Statements for SMIv2*

Subsequent documents that have defined SNMPv3 are:

- RFC 3410, *Introduction and Applicability Statements for Internet Standard Management Framework*
- RFC 3411, *An Architecture for Describing Simple Network Management Protocol (SNMP) Management Frameworks*
- RFC 3412, *Message Processing and Dispatching for the Simple Network Management Protocol (SNMP)*
- RFC 3413, *Simple Network Management Protocol (SNMP) Applications*
- RFC 3414, *User-based Security Model (USM) for version 3 of the Simple Network Management Protocol (SNMPv3)*
- RFC 3415, *View-based Access Control Model (VACM) for the Simple Network Management Protocol (SNMP)*
- RFC 3416, *Version 2 of the Protocol Operations for the Simple Network Management Protocol (SNMP)*
- RFC 3417, *Transport Mappings for the Simple Network Management Protocol (SNMP)*
- RFC 3418, *Management Information Base (MIB) for the Simple Network Management Protocol (SNMP)*

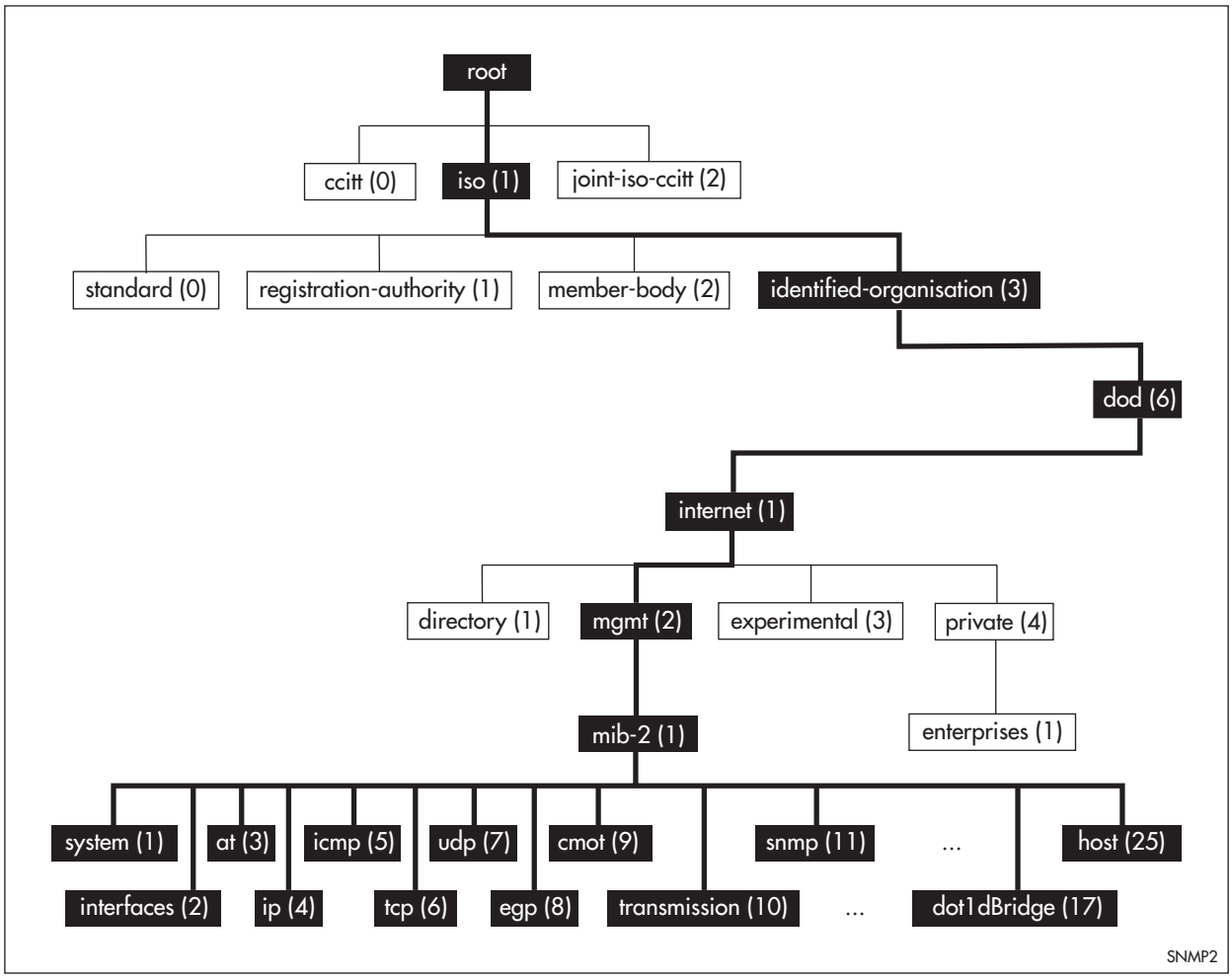
Structure of Management Information

The structure of management information (SMI) defines the schema for a collection of managed objects residing in a virtual store called the *management information base* (MIB). The information in a MIB includes administrative and operational configuration information, as well as counters of system events and activities.

The MIB is organised into a tree-like hierarchy in which nodes are each assigned an identifier consisting of a non-negative integer and an optional brief textual description. The top of the MIB, as it relates to the management of Internet protocols is summarised in Figure 32-2 on page 32-5.

Each managed object is represented by a leaf node and is defined by its name, syntax, access mode, status and description. It can also be specifically identified by its unique position within the tree. This position is expressed as a series of dot-delimited sub-identifiers that start at the root node and end in the sub-identifier at the particular object's leaf node. For example, in Figure 32-2 the object named interfaces would be uniquely identified by the string of individual sub-identifiers, 1.3.6.1.2.1.2.

Figure 32-2: Top levels of the Internet-standard Management Information Base (MIB)



Objects defined in the Internet-standard MIB (MIB-II) reside in the mib(1) sub-tree.

Names

Names are used to identify managed objects, and are hierarchical in nature. An *object identifier* is a globally unique, authoritatively assigned sequence of non-negative integers which traverse the MIB tree from the root to the node containing the object.

Object identifiers may be represented in one of the following forms:

- **Dotted notation** lists the integer values found by traversing the tree from the root to the node in question, separated by dots. For example, the following identifies the MIB-II sub-tree:

  1.3.6.1.2.1

  The following identifies the sysDescr MIB object in the system group of MIB-II:

  1.3.6.1.2.1.1.1

- **Textual notation** lists the textual descriptions found by traversing the tree from the root to the node in question, separated by spaces and enclosed in braces. For example, the following identifies the internet sub-tree:

  { iso org dod 1}

  The name may be abbreviated to a relative form. The following example identifies the first (directory) node of the internet sub-tree:

  { internet 1}

- **Combined notation** lists both the integer values and textual descriptions found by traversing the tree from the root to the node in question. The integer value is placed in parentheses after the textual description. The labels are separated by spaces and enclosed in braces. For example, the following identifies the first (directory) node in the internet sub-tree:

  {iso(1) org(3) dod(6) internet(1) 1}

  The name may be abbreviated to the following:

  directory(1)

Since there is no effective limit to the magnitude of non-negative integers, and no effective limit to the depth of the tree, the MIB provides an unlimited name space.

An object is also usually assigned an *object descriptor*. The object descriptor is a unique, mnemonic, printable string intended for humans to use when discussing the MIB. Examples are sysDescr, ifTable and ipRouteNextHop.

Instances

Objects are just templates for data types. An actual value that can be manipulated by an NMS is an *instance* of an object. An instance is named by appending an *instance identifier* to the end of the object's object identifier. The instance identifier depends on the object's data type:

- If the object is not a column in a table, the instance identifier is 0 (zero). For example, the instance of the sysDescr MIBobject is:

  sysDescr.0
  or 1.3.6.1.2.1.1.1.0

- If the object is a column in a table, the method used to assign an instance identifier varies. Typically, the value of the index column or columns is used.

  The object ifTable in MIB-II contains information about interfaces and is indexed by the interface number, ifIndex. The instance of the ifDescr object for the first interface is:

  ifDescr.1
  or 1.3.6.1.2.1.2.2.1.2.1

  If the index column is an IP address, the entire IP address is used as the instance identifier. The object ipRouteTable in MIB-II contains information about IP routes and is indexed by the destination address, ipRouteDest. The instance of the ipRouteNextHop object for the route 131.203.9.0 is:

  ipRouteNextHop.131.203.9.0
  or 1.3.6.1.2.1.4.21.1.7.131.203.9.0

  If the table has more than one index, the values of all the index columns are combined to form the instance identifier. The object tcpConnTable in MIB-II contains information about existing TCP connections and is indexed by the local IP address (tcpConnLocalAddress), the local port number (tcpConnLocalPort), the remote IP address (tcpConnRemAddress) and the remote port number (tcpConnRemPort) of the TCP connection. The instance of the tcpConnState object for the connection between 131.203.8.36,23 and 131.203.9.197,1066 is:

  tcpConnState.131.203.8.36.23.131.203.9.197.1066
  or 1.3.6.1.2.1.6.13.1.1.131.203.8.36.23.131.203.9.197.1066

Syntax

The syntax of an object describes the abstract data structure corresponding to that object type. For example, INTEGER or OCTET STRING.

Access

The access mode of an object describes the level of access for the object (Table 32-1).

Table 32-1: Access modes for MIB objects

Access	Description
Read-only	The object's value can be read but not set.
Read-write	The object's value can be read and set.
Write-only	The object's value can be set but not read.
Not-accessible	The object's value cannot be read or set.

Status

The status of an object describes the implementation requirements for the object (Table 32-2).

Table 32-2: Status values for MIB objects

Status	Description
Mandatory	Managed devices must implement the object.
Optional	Managed devices may implement the object.
Obsolete	Managed devices need no longer implement the object.
Deprecated	Managed devices should implement the object. However, the object may be deleted from the next version of the MIB. A new object with equal or superior functionality is defined.

Description

The definition of an object may include an optional textual description of the meaning and use of the object. This description is often essential for successful understanding of the object.

The SNMP Protocol

The SNMP protocol provides a mechanism for management entities, or stations, to extract information from the Management Information Base (MIB) of a managed device.

The normal method of accessing information in a MIB is to use a Network Management Station (NMS), typically a PC or workstation, to send commands to the managed device (in this case the switch) using the SNMP protocol.

SNMP can use a number of different protocols as its underlying transport mechanism, but the most common transport protocol, and the only one supported by the switch, is UDP. Therefore the IP module must be enabled and properly configured in order to use SNMP. SNMP trap messages are sent to UDP port 162; all other SNMP messages are sent to UDP port 161. The switch's SNMP agent accepts SNMP messages up to the maximum UDP length the switch can receive.

Other transport mappings have been defined (e.g. OSI [RFC 1418], AppleTalk [RFC 1419] and IPX [RFC 1420]), but the standard transport mapping for the Internet (and the one the switch uses) is UDP. The IP module must be enabled and configured correctly. See Chapter 13, Internet Protocol (IP) for detailed descriptions of the commands required to enable and configure IP.

SNMP Versions

The switch supports SNMP version 1 (SNMPv1), SNMP version 2c (SNMPv2c) and SNMP Version 3 (SNMPv3). The three versions operate similarly. SNMPv2c updated the original protocol, and offered the following main enhancements:

- a new format for trap messages.
- the get-bulk-request PDU allows for the retrieval of large amounts of data, including tables, with one message.
- more error codes mean that error responses to set messages have more detail than is possible with SNMPv1.
- three new exceptions to errors can be returned for get, get-next and get-bulk-request messages. These are: noSuchObject, noSuchInstance, and endOfMibView.

SNMPv3 provides significant enhancements to address the security weaknesses existing in the earlier versions. This is achieved by implementing two new major features:

- Authentication - by using password hashing and time stamping.
- Privacy - by using message encryption.

Support for multiple versions of SNMP is achieved by responding to each SNMP request with a response of the same version. For example, if an SNMPv1 request is sent to the switch, an SNMPv1 response is returned. If an SNMPv2c request is sent, an SNMPv2c response is returned. Therefore, authentication and encryption functions are not invoked when messages are detected as having either an SNMPv1 or SNMPv2c protocol format.

SNMP Messages

The SNMP protocol is termed simple because it has only six operations, or messages—get, get-next, get-response, set, and trap, and SNMPv2c also has the get-bulk-request message (Table 32-4 on page 32-10). The replies from the managed device are processed by the NMS and generally used to provide a graphical representation of the state of the network. The two major SNMP operations available to a management station for interacting with a client are the get and set operations. The SNMP set operator can lead to security breaches, since SNMP is not inherently very secure. When forced to operate in either SNMPv1 or v2 mode, when operating with older management stations for example, care must be taken in the choice and safe-guarding of community names, which are effectively passwords for SNMP. See Appendix C, SNMP MIBs for a description of the switch's implementation of each MIB object with read-write access.

Polling versus Event Notification

SNMP employs a polling paradigm. A Network Management Station (NMS) polls the managed device for information as and when it is required, by sending get-request, get-next-request, and/or get-bulk-request PDUs to the managed device. The managed device responds by returning the requested information in a get-response PDU. The NMS may manipulate objects in the managed device by sending a set-request PDU to the managed device.

The only time that a managed device initiates an exchange of information is in the special case of a trap PDU. A managed device may generate a limited set of traps to notify the NMS of critical events that may affect the ability of the NMS to communicate with the managed device or other managed devices on the network, and therefore to "manage" the network. Such events include the restarting or re-initialisation of a device, a change in the status of a network link (up or down), or an authentication failure.

Message Format for SNMPv1 and SNMPv2c

Figure 32-3 shows the format of an SNMP message for v1 and v2c versions. The function of the fields are described in Table 32-3. Table 32-4 shows the SNMP PDUs, and Table 32-5 shows the generic traps.

Figure 32-3: Format of an SNMP message

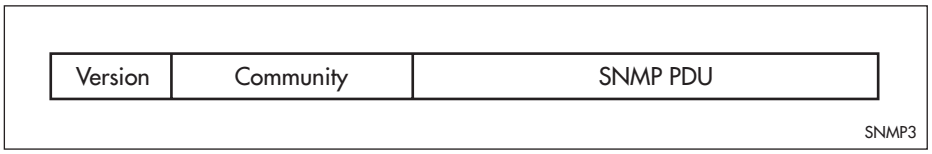


Table 32-3: Fields in an SNMP message

Field	Function
Version	The version of the SNMP protocol. The value is version-1 (0) for the SNMP protocol as defined in RFC 1157, or version-2c (1) for the SNMP protocol as defined in RFC 1902.
Community	The name of an SNMP community, for authentication purposes.
SNMP PDU	An SNMP Protocol Data Unit (PDU).

Table 32-4: SNMP PDUs

PDU	Function
get-request	Sent by an NMS to an agent, to retrieve the value of an object.
get-next-request	Sent by an NMS to an agent, to retrieve the value of the next object in the sub-tree. A sub-tree is traversed by issuing a get-request PDU followed by successive get-next-request PDUs.
get-bulk-request	Sent by an NMS to an agent to request a large amount of data with a single message. This is for SNMPv2c messages.
set-request	Sent by an NMS to an agent, to manipulate the value of an object.
get-response	Sent by an agent to an NMS in response to a get-request, get-next-request, get-bulk-response, or set-request PDU.
trap	Sent by an agent to an NMS to notify the NMS of a extraordinary event.
report	Although not explicitly defined in the RFCs, reports are used for specific purposes such as EngineID discovery and time synchronisation.

Table 32-5: Generic SNMP traps

Value	Meaning
coldStart	The agent is re-initialising itself. Objects may be altered.
warmStart	The agent is re-initialising itself. Objects are not altered.
linkDown	An interface has changed state from up to down.
linkUp	An interface has changed state from down to up.
authenticationFailure	An SNMP message has been received with an invalid community name.
egpNeighborLoss	An EGP peer has transitioned to down state.

SNMP Communities (Version v1 and v2c)

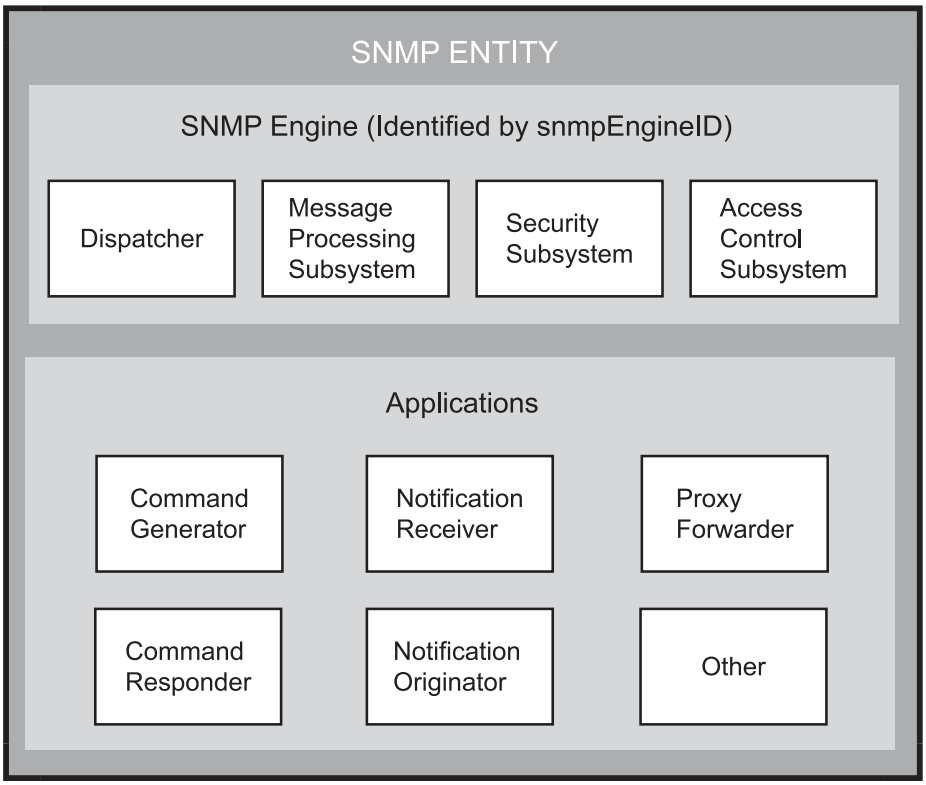
A community is a relationship between an NMS and an agent. The community name is used like a password for a trivial authentication scheme. Both SNMPv1 and SNMPv2c provide security based on the community name only. The concept of communities does not exist for SNMPv3, which instead provides for a far more secure communications method using *entities*, *users*, and *groups*.

Important We strongly recommend removing community membership from all SNMPv3 configured devices to prevent access to them via SNMPv1 and SNMv2c, which could bypass the additional SNMPv3 security features.

SNMPv3 Entities

Entities comprise one of the basic components of the SNMPv3 enhanced architecture. They define the functionality and internal structure of the SNMP managers and agents. An in depth description of entities can be found in RFC 3411, on which the following text is based. SNMPv3 defines two entity types, a *manager* and an *agent*. Both entity types contain two basic components: an *SNMP engine* and a set of *applications*.This concept is illustrated in Figure 32-4 on page 32-12.

Figure 32-4: SNMPv3 entity components



SNMP Engine

The engine provides the basic services to support the agents component applications, in this respect it performs much of the functionality expected of the ISO Session and Presentation layers. These functions include, message transmission and reception, authentication and encryption, and access control to its managed objects database (MIB). The SNMP engine comprises the following components:

- Dispatcher
- Message processing Subsystem
- Security Subsystem
- Access Control Subsystem

The only security subsystem presently supported is the user based security model (USM).

Each SNMP engine is identified by an *snmpEngineID* that must be unique within the management system. A one to one association exists between an engine and the entity that contains it.

Entity Applications

The following applications are defined within the agent applications:

- Command Generator
- Notification Receiver
- Proxy Forwarder
- Command Responder
- Notification Originator
- Other

SNMPv3 Message Protocol Format

Figure 32-5 on page 32-13 and Table 32-6 on page 32-14 explain the protocol format of an SNMPv3 message.

Figure 32-5: SNMPv3 protocol format

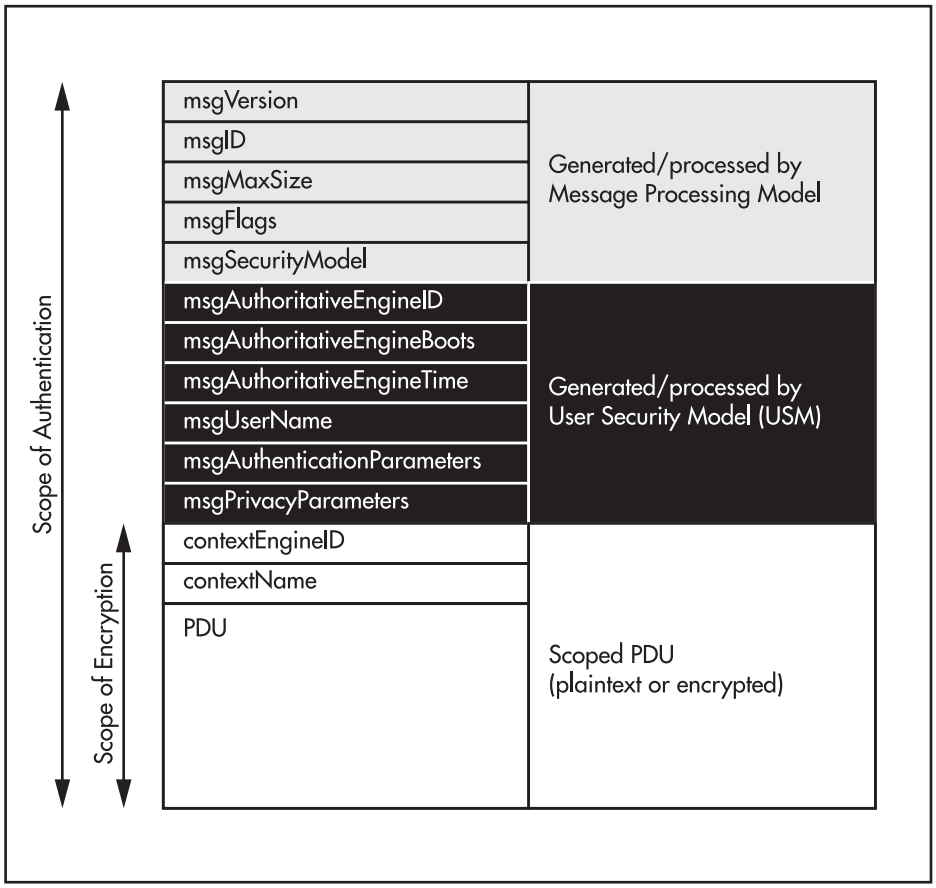


Table 32-6: SNMPv3 PDUs

Value	Meaning
msgVersion	Identifies the message format to be SNMPv3.
msgID	An identifier used between SNMP entities to coordinate message requests and responses. Note that a message response takes the msgID value of the initiating message.
msgMaxSize	Conveys the maximum message size (in octets) supported by the sender of the message. Specified as an integer between 484 and 2^{31}-1.
msgFlags	A single octet whose last three bits indicate the operational mode for privacy, authentication, and report.
msgSecurityModel	An identifier used to indicate the security mode (i.e. SNMPv1, SNMPv2c or SNMPv3 to be used when processing the message. Note that although only the SNMPv3 identifier is accepted by the switch, these earlier version message formats are detected by the msgVersion field and processed appropriately.
msgAuthoritativeEngineID	The ID of the authoritative engine that relates to a particular message, i.e. the source engine ID for Traps, Responses and Reports, and the destination engine for Gets, GetNexts, Sets, and Informs.
msgAuthoritativeEngineBoots	A value that represents the number of times the authoritative engine has rebooted since its installation. Its value has the range 1 to 2^{31}-1.
msgAuthoritativeEngineTime	The number of seconds since the authoritative engine snmpEngineBoots counter was last incremented.
msgUserName	The name of the user (principal) on whose behalf the message is being exchanged.
msgAuthenticationParameters	If the message has been authenticated, this field contains a serialized OCTET STRING representing the first 12 octets of the HMAC-MD5-96 output done over the whole message.
msgPrivacyParameters	For encrypted data, this field contains the "salt" used to create the DES encryption Initialisation Vector (IV).
ContextEngineID	Within a particular administrative domain, this field uniquely identifies an SNMP entity that may realize an instance of a context with a particular contextName.
ContextName	A unique name given to a context within a particular SNMP entity.

SNMPv1 and SNMPv2c on the Switch

Although software levels 2.6.3 and higher support the specific facilities of SNMP v1 and v2, their documentation is available to provide backward compatibility with older network management systems. The far superior security features offered by implementing SNMPv3 should be used wherever possible. See "SNMPv3 on the Switch" on page 32-20.

The switch's implementation of SNMPv1 is based on RFC 1157, *A Simple Network Management Protocol (SNMP)*, and RFC 1812, *Requirements for IP Version 4 Routers*.

The switch's implementation of SNMPv2c is based on the RFCs listed in "Network Management Framework" on page 32-3.

The SNMP agent can be enabled or disabled by using the commands:

```
enable snmp
disable snmp
```

When the SNMP agent is disabled, the agent does not respond to SNMP request messages. The agent is disabled by default. The current state and configuration of the SNMP agent can be displayed by using the command:

```
show snmp
```

SNMP MIB Views for SNMPv1 and SNMPv2c

An SNMP MIB *view* is a arbitrary subset of objects in the MIB. Objects in the view may be from any part of the object name space, and not necessarily the same sub-tree. An *SNMP community profile* is the pairing of an *SNMP access mode* (read-only or read-write) with the access mode defined by the MIB for each object in the view. For each object in the view, the community profile defines the operations that can be performed on the object (Table 32-7).

Table 32-7: Community profiles for objects in a MIB view

SNMP Access Mode	Object Access Defined by MIB			
	Read-Only	**Read-Write**	**Write-Only**	**Not Accessible**
Read-Only	get, get-next, trap	get, get-next, trap	*None*	*None*
Read-Write	get, get-next, trap	get, get-next, set, trap	get, get-next, set, trap(*)	*None*

Pairing an SNMP community with an SNMP community profile determines the level of access that the agent affords to an NMS that is a member of the specified community. When an agent receives an SNMP message, it checks the community name encoded in the message. If the agent knows the community name, the message is deemed to be authentic and the sending SNMP entity is accepted as a member of the community. The community profile associated with the community name then determines the sender's view of the MIB and the operations that can be performed on objects in the view.

SNMP Communities

SNMP communities were introduced into SNMPv1 and retained in version 2c. Although the switch's software still supports communities, this is to provide backward compatibility with legacy management systems. Communities should **not** be used where a secure network is required. Instead, use the secure network features offered by SNMPv3.

An SNMP *community* is a pairing of an SNMP agent with a set of SNMP application entities. Communities are the main configuration item in the switch's implementation of SNMPv1 and v2, and are defined in terms of a list of IP addresses which define the SNMP application entities (trap hosts and management stations) in the community. An SNMP community is created by using the command:

```
create snmp community=name [access={read|write}]
   [traphost=ipadd] [manager=ipadd]
   [open={on|off|yes|no|true|false}] [v1traphost=ipadd]
   [v2ctraphost=ipadd]
```

which defines the name of the community (e.g. "public"), and specifies the IP address of a trap host and/or a management station. This command also specifies the version of SNMP received by trap hosts. A community can be modified by using the command:

```
set snmp community=name [access={read|write}]
   [open={on|off|yes|no|true|false}]
```

Important Community names act as passwords and provide minimal authentication. Any SNMP application entity that knows a community name can read the value of any instance of any object in the MIB implemented in the switch. Any SNMP application entity that knows the name of a community with write access can change the value of any instance of any object in the MIB implemented in the switch, possibly affecting the operation of the switch. For this reason, take care with the security of community names.

An SNMP community is destroyed by using the command:

```
destroy snmp community=name
```

Additional trap hosts and management stations can be added to or removed from a community by using the commands:

```
add snmp community=name [traphost=ipadd] [manager=ipadd]
   [v1traphost=ipadd] [v2ctraphost=ipadd]
delete snmp community=name [traphost=ipadd] [manager=ipadd]
   [v1traphost=ipadd] [v2ctraphost=ipadd]
```

When a trap is generated by the SNMP agent it is forwarded to all trap hosts in all communities. The community name and manager addresses are used to provide trivial authentication. An incoming SNMP message is deemed authentic if it contains a valid community name and originated from an IP address defined as a management station for that community.

An SNMP community, or the generation of traps by the community, can be temporarily enabled or disabled by using the commands:

```
disable snmp community=name [trap]
enable snmp community=name [trap]
```

When a community is disabled, the SNMP agent behaves as if the community does not exist and generates authentication failure traps for messages directed to the disabled community. Information about the configuration of SNMP communities can be displayed by using the command:

```
show snmp community=name
```

The SNMP agent does not support a default community called "public" with read-only access, traps disabled and open access as mandated in RFC 1812, as this is a security hole open for users who wish to use the switch with minimal modification to the default configuration. The default configuration of the switch has no defined communities. Communities must be explicitly created. The defaults for other parameters such as the open access flag and the trap enabled flag also follow the principle of security first, access second.

SNMP *authentication* (for SNMPv1 and v2) is a mechanism whereby an SNMP message is declared to be authentic, that is from an SNMP application entity actually in the community to which the message purports to belong. The mechanism may be trivial or secure. The only form of SNMP authentication implemented by the switch's SNMP agent is trivial authentication. The authentication failure trap may be generated as a result of the failure to authentication an SNMP message. The generation of authentication failure traps may be enabled or disabled by using the commands:

```
enable snmp authenticate_trap
disable snmp authenticate_trap
```

Link up/down traps can be enabled or disabled on a per-interface basis by using the commands:

```
enable interface={ifIndex|interface|dynamic} linktrap
disable interface={ifIndex|interface|dynamic} linktrap
```

where *ifIndex* is the value of ifIndex for the interface in the Interface Table and *interface* is the name of the interface. If link traps are enabled, when an interface changes to or from the "Down" state an SNMP trap is sent to any defined trap hosts. Link traps are disabled by default on the switch. The current settings for link traps can be displayed by using the command:

```
show interface={ifIndex|interface}
```

The maximum number of link traps generated per minute can be set for each static interface or for all dynamic interfaces by using the command:

```
set interface={ifIndex|interface|dynamic} traplimit=1..60
```

See Chapter 12, Interfaces for a detailed description of the commands for configuring and monitoring link up/down traps.

Switch interfaces can be enabled or disabled via SNMP by setting the ifAdminStatus object in the ifTable of MIB-II MIB to 'Up(1)' or 'Down(2)' for the corresponding ifIndex. If it is not possible to change the status of a particular interface the switch returns an SNMP error message.

The switch's implementation of the ifOperStatus object in the ifTable of MIB-II MIB supports two additional values—"Unknown(4)" and "Dormant(5)" (e.g. an inactive dial-on-demand interface).

Important An unauthorised person with knowledge of the appropriate SNMP community name could bring an interface up or down. Community names act as passwords for the SNMP protocol. When creating an SNMP community with write access, take care to select a secure community name and to ensure that only authorised personnel know it.

An SNMP MIB view is a subset of objects in the MIB that pertain to a particular network element. For example, the MIB view of a hub would be the objects relevant to management of the hub, and would not include IP routing table objects, for example. The switch's SNMP agent does not allow the construction of MIB views. The switch supports all relevant objects from all MIBs that it implements.

Note that the switch's standard **set** and **show** commands can also be used to access objects in the MIBs supported by the switch.

Defining Management Stations within Communities

You can add management stations to a community either individually, by entering just its IP address, or you can enter a **range** of management stations by entering an IP address that ends with a '/' character followed by a number between 1 and 32. The number that follows the '/' character operates as an address mask to define a range of addresses for the management stations. The following example shows how to allocate a band of three binary addresses to a portion of the subnet 146.15.1.X

Example In this example we make provision for up to 8 possible management stations within a community called "admin".

1. Decide on the number of management stations that you want to assign to a particular subnet, then decide how many binary digits are required to define this number of addresses. In this case we need up to 8 management stations, so we will assign 3 binary digits (3 binary digits can provide 8 different values). To assign the last 3 binary digits for management stations, we assign a prefix that is a count of all binary digits in the address minus those to be assigned as management stations. In this case the prefix is 29; this being the number of binary digits in an IP address (32) minus the number of digits assigned to the management stations (3).
2. The method used in this step depends on whether or not the community already exists.

   a. If the community called "admin" does not exist, create a **new** community called "admin" and allocate a three binary digit block of addresses to the address subnet 146.15.1.X. To do this, enter the IP address of one of the management stations and attach the management station prefix / 29. Use the command:

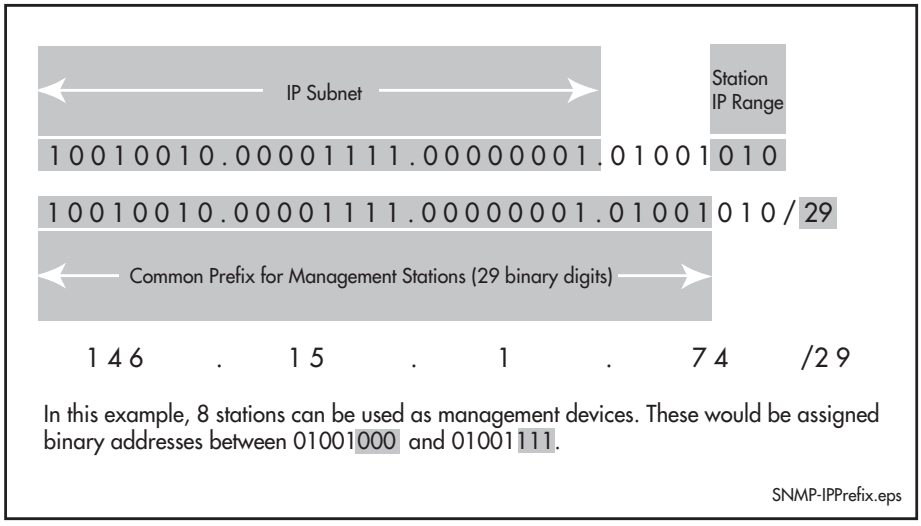
   ```
   create snmp community=admin manager=146.15.1.74/29
   ```

   b. If the community called "admin" already exists, allocate a three binary digit block of addresses to an **existing** community called "admin" with the address subnet 146.15.1.X. To do this, enter the IP address of one of the management stations and attach the management station prefix / 29, use the command:

   ```
   add snmp community=admin manager=146.15.1.74/29
   ```

   The management IP range and prefix used in this example is illustrated in Figure 32-6 on page 32-19.

Figure 32-6: Adding SNMP management stations by assigning an IP prefix



For security reasons, the common management prefix should be larger than the IP subnet. This prevents stations on one subnet from being considered valid management stations on a different subnet.

Configuration Example (SNMPv1 and v2)

This example shows how to configure the switch's SNMP agent. Two network management stations have been set up on a large network. The central NMS (IP address 192.168.11.5) monitors devices on the network and uses SNMP set messages to manage devices on the network. Trap messages are sent to this management station. The regional network management station (IP addresses 192.168.16.1) is used just to monitor devices on the network by using SNMP get messages. Link traps are enabled for all interfaces on this particular switch.

The IP module must be enabled and correctly configured in order to access the SNMP agent in the switch. This is because the IP module handles both the TCP transport functions, and the UDP functions that enable datagrams to transport SNMP messages. See Chapter 13, Internet Protocol (IP) for commands that enable and configure IP.

To configure SNMP

1. **Enable the SNMP agent.**

   Enable the SNMP agent and enable the generation of authenticate failure traps to monitor unauthorised SNMP access.

   ```
   enable snmp
   enable snmp authenticate_trap
   ```

2. **Create a community with write access for the central NMS.**

   Create a community called "private", with write access for use only by the central network management station at 192.168.11.5. All traps are sent to this NMS.

   ```
   create snmp community=private access=write
      traphost=192.168.11.5 manager=192.168.11.5 open=no
   ```

Enable sending of trap messages for this community.

```
enable snmp community=private trap
```

Care must be taken with the security of community names. Do not use the name "private" in your network because it is too obvious. Community names act as passwords and provide only trivial authentication. Any SNMP application entity that knows a community name can read the value of any instance of any object in the MIB implemented in the switch. Any SNMP application entity that knows the name of a community with write access can change the value of any instance of any object in the MIB implemented in the switch, possibly affecting the operation of the switch.

3. **Create a community with read-only access for the regional NMS.**

   Create a community called "public", with read-only access for use by the regional network management station at 192.168.16.1. To define a range of management stations, see "Defining Management Stations within Communities" on page 32-18.

   ```
   create snmp community=public access=read
      manager=192.168.16.1 open=no
   ```

   Enable sending of trap messages for this community.

   ```
   enable snmp community=public trap
   ```

4. **Enable link traps.**

   Enable link traps for the switch's VLAN interfaces.

   ```
   enable interface=vlan1 linktrap
   ```

5. **Check the configuration.**

   Check that the current configuration of the SNMP communities matches the desired configuration:

   ```
   show snmp
   ```

   ```
   show snmp community
   ```

   Check that the interface link up/down traps have been correctly configured:

   ```
   show interface=vlan1
   ```

SNMPv3 on the Switch

SNMPv3 is the third version of the Simple Network Management Protocol. The architecture comprises the following:

- entities that may be either managers, agents, or both
- a management information base (MIB)
- a transport protocol

At least one manager node runs the SNMP management software in every configuration. Managed devices such as routers, servers, and workstations are equipped with an agent software module. The agent provides access to local objects in the MIB that reflect activity and resources at the node. The agent also responds to manager commands to retrieve values from, and set values in the MIB.

SNMP MIB Views for SNMPv3

An SNMP MIB view is a arbitrary subset of objects in the MIB. Objects in the view may be from any part of the object name space, and not necessarily the same sub-tree. Views are created by using the command:

```
add snmp view=view-name {oid=oid-tree|mib=mib-name}
   [type={include|exclude}]
```

For a practical MIB view see "Allied Telesis Enterprise MIB" on page C-4 of Appendix C, SNMP MIBs.

SNMP Defined MIB Names

Table 32-8 lists the MIB names that are defined within the ATR switch. These names can be used in commands instead of using the MIB tree character string.

Table 32-8: SNMP-defined MIB names

Value	Meaning
internet	1.3.6.1
mib-2	1.3.6.1.2.1
system	1.3.6.1.2.1.1
interfaces	1.3.6.1.2.1.2
at	1.3.6.1.2.1.3
ip	1.3.6.1.2.1.4
icmp	1.3.6.1.2.1.5
tcp	1.3.6.1.2.1.6
udp	1.3.6.1.2.1.7
egp	1.3.6.1.2.1.8
transmission	1.3.6.1.2.1.10
snmp	1.3.6.1.2.1.11
bgp	1.3.6.1.2.1.15
rmon	1.3.6.1.2.1.16
bridge	1.3.6.1.2.1.17
host	1.3.6.1.2.1.25
mau	1.3.6.1.2.1.26
if	1.3.6.1.2.1.31
private	1.3.6.1.4
alliedTelesis	1.3.6.1.4.1.207
snmpV2	1.3.6.1.6
snmpModules	1.3.6.1.6.3
snmpFramework	1.3.6.1.6.3.10
snmpMPD	1.3.6.1.6.3.11
snmpTarget	1.3.6.1.6.3.12
snmpUsm	1.3.6.1.6.3.15
snmpVacm	1.3.6.1.6.3.16

SNMP Groups

Groups were introduced as part of SNMPv3. They are the means by which users are assigned their views and access control policy. Groups are created by using the command:

```
add snmp group=group-name
    securitylevel={authnopriv|noauthnopriv|authpriv}
    [readview=view-name] [writeview=view-name]
    [notifyview=view-name]
```

Once a group has been created, users can be added to them. In practice a number of groups would be created, each with varying views and access security requirements. Users would then be added to their most appropriate groups. Each Group name and Security Level pair must be unique within a switch.

SNMP Users

Users were introduced as part of SNMPv3. From a system perspective a user is represented as an entity stored in a table that defines the access and authentication criteria to be applied to access or modify the SNMP MIB data. Users are created by using the command:

```
add snmp user=user-name [group=group-name]
    [authprotocol={none|md5|sha}] [authpassword=password]
    [privprotocol={none|des}] [privpassword=password]
```

SNMP Target Addresses

Target addresses were introduced as part of SNMPv3. They specify the destination and user that receives outgoing notifications such as trap messages. SNMP target address names must be unique within the managed device. Target addresses are created by using the command:

```
add snmp targetaddr=address-name ip=target-ipadd
    [udp=udp-port] params=params-name
```

SNMP Target Params

Target params were introduced as part of SNMPv3. They specify an entry in the snmpTargetParamsTable. SNMP target params names must be unique within the managed device. Target params are created by using the command:

```
add snmp targetparams=params-name
    securitylevel={noauthnopriv|authnopriv|authpriv}
    user=user-name
```

An entry in the target-params table can be used to apply the same security profile to multiple management targets, and is used in conjunction with the **add snmp targetaddr** command on page 32-27 to specify the security profile for a Target Address.

Configuration Example (SNMPv3)

This example shows how to configure the switch's SNMP agent. Two network management stations have been set up on a large network. The central NMS (IP address 192.168.11.5) monitors devices on the network and uses SNMP set messages to manage devices on the network. Trap messages are sent to this management station.

The IP module must be enabled and correctly configured in order to access the SNMP agent in the switch, since the IP module handles the UDP datagrams used to transport SNMP messages. See Chapter 13, Internet Protocol (IP) for a detailed description of the commands required to enable and configure IP.

To configure SNMP

1. **Enable the SNMP agent.**

   Enable the SNMP agent and enable the generation of authenticate failure traps to monitor unauthorised SNMP access.

   ```
   enable snmp
   enable snmp authenticate_trap
   ```

2. **Add SNMP views.**

   You can specify views using their OID or the predefined MIB name.

   ```
   add snmp view=atmib oid=1.3.6.1.2.14 type=include
   add snmp view=atmib mib=alliedtelesis type=include
   ```

3. **Add SNMP group.**

   ```
   add snmp group=ord-user securitylevel=noauthnopriv
      readview=atmib
   add snmp group=admin-user securitylevel=authnopriv
      readview=atmib
      writeview=atmib
      notifyview=atmib
   ```

4. **Add SNMP users.**

   Add users to the groups by using commands such as:

   ```
   add snmp user=ken group=admin-user authprotocol=md5
      authpassword=mercury
   ```

5. **Add SNMP targetparams.**

   ```
   add snmp targetparams=netmonpc securitylevel=authnopriv
      user=ken
   ```

6. **Add SNMP target address.**

   ```
   add snmp targetaddr=target ip=192.168.11.5 udp=162
      params=netmonpc
   ```

Command Reference

This section describes the commands available on the switch to configure and manage the SNMP agent. The IP module must be enabled and correctly configured in order to access the SNMP agent in the switch, since the IP module handles the UDP datagrams used to transport SNMP messages. See Chapter 13, Internet Protocol (IP) for a detailed description of the commands required to enable and configure IP.

The shortest valid command is denoted by capital letters in the Syntax section. See "Conventions" on page xxxviii of About this Software Reference in the front of this manual for details of the conventions used to describe command syntax. See Appendix A, Messages for a complete list of error messages and their meanings.

add snmp community

Syntax

```
ADD SNmp COMmunity=name [TRAPhost=ipadd]
    [MAnager=prefix[/0..32]] [V1traphost=ipadd]
    [V2ctraphost=ipadd]
```

where:

- *name* is a string 1 to 15 characters long. Valid characters are any printable ASCII character. The *name* is case sensitive, that is "Public" is a different name from "public".
- *ipadd* is an IP address in dotted decimal notation.
- *prefix* is an IP address range in dotted decimal notation optionally followed by a "/" character and the number of bits in the network mask.

Description

This command adds a trap host or a management station to a specific SNMP community. Note that for security reasons, communities should not be used when operating SNMPv3.

The **community** parameter specifies the SNMP community. The community must already exist on the switch.

The **traphost**, **v1traphost** and **v2ctraphost** parameters specify a trap host for the SNMP community. This is the IP address of a device to which traps generated by the switch are sent. A community may have more than one trap host, but only one can be specified when the community is created. If the parameter is not specified, the community has no defined trap host. If the **traphost** or **v1traphost** parameter is used to specify a trap host, the switch sends SNMPv1 format traps to this host. If the **v2ctraphost** parameter is used to specify a trap host, the switch sends SNMPv2c format traps to this host. The same trap host can be used for sending version 1 and version 2c traps, but you have to add it to the SNMP community twice.

The **manager** parameter can specify either a single management station or a range of management stations for this SNMP community. If no parameter is specified, then the community will have no management station defined to it. Note that additional management stations or ranges can be created once the community has been created.

Examples

To add the host 192.168.1.1 as both a trap host and a management station to the existing SNMP community “Administration”, use the command:

```
add snmp community=Administration traphost=192.168.1.1
   manager=192.168.1.1
```

To add the host 192.168.1.2 as both a version 1 and a version 2c trap host to the existing SNMP community “Administration”, use the command:

```
add snmp community=Administration traphost=192.168.1.2
   v2ctraphost=192.168.1.2
```

To add a block of 16 management stations to a community named administration, use the command:

```
add snmp community=administration manager=192.168.7.31/28
```

Note that this adds management station addresses of 192.168.7.16 to 192.168.7.31.

Related Commands

create snmp community
delete snmp community
destroy snmp community
disable snmp community
disable snmp community trap
enable snmp community trap
enable snmp community trap
show snmp community

add snmp group

Syntax

```
ADD SNmp GROup=group-name
   SECuritylevel={AUTHnopriv|NOAUthnopriv|AUTHPRiv}
   [READview=view-name] [WRITEview=view-name]
   [NOTifyview=view-name]
```

where:

- *group-name* is a string 1 to 32 characters long. It may contain any printable character and is case sensitive. If *group-name* contains spaces, it must be in double quotes.
- *view-name* is a string 1 to 32 characters long. It may contain any printable character and is case sensitive. If *view-name* contains spaces, it must be in double quotes.

Description

This command is used with SNMP version 3 only, and adds an SNMP group, optionally setting the security level and view access modes for the group. You must specify at least one of the optional parameters **readview**, **writeview**, or **notifyview**.

The **group** parameter names a group to which security and authorisation levels can be applied. The security and access levels defined for the group represent the minimum required of its users in order to gain access.

The practical number of groups that may be added is limited by the memory available within the device.

The **securitylevel** parameter specifies the minimum access and privacy levels for the group. **noauthnopriv** specifies that users belonging to this group need not apply authentication or privacy (encryption). **authnopriv** specifies that users belonging to this group must apply authentication based on either MD5 or SHA algorithms, but they need not apply privacy (encryption). **authpriv** specifies that users belonging to this group must apply authentication based on either MD5 or SHA algorithms, together with encryption based on the DES algorithm. This option provides both security and privacy.

The **readview** parameter specifies the MIB contents that this group can read. Note that this content is specified for each view-name by using the **add snmp view** command. Group members are not able to read any MIB objects unless this parameter is specified.

The **writeview** parameter specifies the MIB contents that this group can modify. Note that this content is specified for each view-name by using the **add snmp view** command. Group members cannot modify MIB objects unless this parameter is specified.

The **notifyview** parameter specifies the notify contents that this group can receive. Note that this content is specified for each view-name by using the **add snmp view** command. Group members cannot read MIB objects unless this parameter is specified. Group members cannot receive notifications unless this parameter is specified.

Examples

To add SNMP group, for ordinary users, use the command:

```
add snmp group=usergroup securitylevel=noauthpriv
   readview=useraccess writeview=useraccess
```

To add SNMP group, for network administrators, use the command:

```
add snmp group=admingroup securitylevel=authpriv
   readview=adminaccess writeview=adminaccess
```

Related Commands

delete snmp group
set snmp group
show snmp group

add snmp targetaddr

Syntax

```
ADD SNmp TARgetaddr=address-name IP=target-ipadd
   [UDP=udp-port] PARams=params-name
```

where:

- *address-name* is a string 1 to 32 characters long. It may contain any printable character and is case sensitive. If *address-name* contains spaces, it must be in double quotes.
- *target ipadd* is an IP address in dotted decimal notation.
- *udp-port* is a decimal number from 1 to 255.
- *params-name* is a string 1 to 32 characters long. It may contain any printable character and is case sensitive. If *params-name* contains spaces, it must be in double quotes.

Description

This command adds an SNMP target address entry to the snmpTargetAddrTable, and is used with SNMP version 3 only.

The **targetaddr** parameter specifies the target parameters for outgoing notifications such as trap messages. SNMP target address names must be unique within the managed device. The practical number of addresses that may be added is limited by the memory available within the device.

The **ip** parameter specifies the IP address to which notifications are sent.

The **udp** parameter specifies the identification number of a UDP port. In accordance with RFC recommendations, the default value is 162.

The **params** parameter specifies the reference to the entry in snmpTargetParamsTable. To create a target **params**, use the **add snmp targetparams** command on page 32-28.

Examples

To add an **snmp targetaddr** called "target" and a **params** called “params” on address "127.0.0.1", use the command:

```
add snmp targetaddr=target ip=127.0.0.1 params=params
```

Related Commands

delete snmp targetaddr
set snmp targetaddr
show snmp targetaddr

add snmp targetparams

Syntax

```
ADD SNmp TARGETParams=params-name
   SECuritylevel={NOAUthnopriv|AUTHnopriv|AUTHPRiv}
   USer=user-name
```

where:

- *params-name* is a string 1 to 32 characters long. It may contain any printable character and is case sensitive. If *params-name* contains spaces, it must be in double quotes.
- *user-name* is a string 1 to 32 characters long. It may contain any printable character and is case sensitive. If *user-name* contains spaces, it must be in double quotes.

Description

This command adds an SNMP target params entry to the snmpTargetParamsTable, and is used with SNMP version 3 only.

The **targetparams** parameter specifies an SNMP target params entry in the snmpTargetParamsTable. An SNMP target params entry with the specified name must not exist in the switch.

The **securitylevel** parameter specifies the security level for this target parameters. **noauthnopriv** specifies that no authentication or privacy is used. **authnopriv** specifies that authentication is based on the MD5 or SHA algorithm. This option provides security but no privacy. **authpriv** specifies that authentication be based on the MD5 or SHA algorithm, and to base encryption on the DES algorithm.

The **user** parameter specifies the SNMP user. Although a user with the specified name need not pre-exist in the switch, it must be added (using the **add snmp user** command) before user access can be enabled.

If the user's authentication and privacy protocols do not support the specified target parameters security level, then the SNMP message is not sent.

Examples

To add an SNMP target params "params" with security level “noAuthNoPriv” for user "test", use the command:

```
add snmp targetparams=params seclevel=noauthnopriv user=test
```

Related Commands

delete snmp targetparams
set snmp targetparams
show snmp targetparams

add snmp user

Syntax

```
ADD SNmp USer=user-name [GROup=group-name]
   [AUTHprotocol={NONE|MD5|SHa}] [AUTHPassword=password]
   [PRIVprotocol={NONE|DES}] [PRIVPassword=password]
```

where:

- *user-name* is a string 1 to 32 characters long. It may contain any printable character and is case sensitive. If *user-name* contains spaces, it must be in double quotes.
- *group-name* is a string 1 to 32 characters long. It may contain any printable character and is case sensitive. If *group-name* contains spaces, it must be in double quotes.
- *password* is a string 8 to 32 characters long. It may contain any printable character and is case sensitive. If *password* contains spaces, it must be in double quotes.

Important When passwords need to be entered on both local and remote stations, they should be entered in such a way as not to compromise system security. Note that Telnet should not be used for this purpose because it transmits passwords across the network in cleartext. Refer to RFC 3414 for more information on key management, or contact your authorised distributor or reseller.

Description

This command is used with SNMP version 3 only, and adds an SNMP user as a member to a specific SNMP group. Additionally it provides the option of selecting an authentication protocol and (where appropriate) an associated password. Similar options are offered for selecting a privacy protocol and password. Note that each SNMP user must be configured on "both" the manager and agent entities. Where passwords are used, these must be the same for both entities.

The **user** parameter specifies the SNMP user. The user name is used to reference the SNMP user in all other SNMP commands. A user with the specified name must not exist in the switch.

The **group** parameter specifies the SNMP group to which the user becomes a member. Although a group with the specified name need not pre-exist in the switch, it must be added (using the **add snmp group** command) before user access can be enabled.

The **authprotocol** parameter specifies the authentication protocol for the SNMP user (either **md5**, **sha**, or **none**). If this parameter is not specified, then the default is none. Note that the authentication specified for the SNMP user must match that specified for its declared group. For example, a user configured with the authentication default of **none**, cannot access a group whose authentication is defined as **md5**.

The **authpassword** parameter specifies the authentication password for the SNMP user. If **authprotocol** is set to **md5** or **sha**, the **authpassword** must be specified.

The **privprotocol** parameter specifies privacy protocol for the SNMP user (either **des** or **none**). If this parameter is not specified, then the default is none. The **privprotocol** cannot be set to **des** if **authprotocol** is set to **none**.

The **privpassword** parameter specifies the privacy password for the SNMP user. If **privprotocol** is set to DES, the **privpassword** must be specified.

Examples To add SNMP user "authuser" as a member of group "usergroup", with authentication protocol "MD5", authentication password "Authpass", privacy protocol "DES" and privacy password "Privpass", use the command:

```
add snmp user=authuser group=usergroup authprotocol=md5
   authpassword=authpass privprotocol=des
   privpassword=Privpass
```

Related Commands **delete snmp user**
set snmp user
show snmp user

add snmp view

Syntax

```
ADD SNmp VIEW=view-name {OID=oid-tree|MIB=mib-name}
    [TYpe={INCLude|EXCLude}]
```

where:

- *vew-name* is a string 1 to 32 characters long. It may contain any printable character and is case sensitive. If *vew-name* contains spaces, it must be in double quotes.
- *oid-tree* is string in a decimal and dot format that is from 1 to 32 sub-identifiers long. For a an explanation of the OID tree and a definition of sub-identifiers, see "Structure of Management Information" on page 32-5.
- *mib-name* is a string 1 to 32 characters long. It may contain any printable character and is case sensitive. If *mib-name* contains spaces, it must be in double quotes. Note that the mib name must be one that is defined to the switch, see "SNMP Defined MIB Names" on page 32-21.

Description

This command adds an SNMP view and specifies the initial sub-tree. Further sub-trees can then be added by specifying a new OID to an existing view.

The **view** parameter specifies the name of the SNMP view. View names are used to reference the SNMP views in all other SNMP commands.

The **oid** parameter specifies the object identifier of the ASN.1 sub-tree to be included or excluded from the view. The sub-tree has to be specified as a character string consisting of numbers, for example, 1.3.6.1.2.1. If this parameter is specified, the **mib** parameter cannot be specified.

The **mib** parameter specifies a predefined MIB node to be included or excluded from the view. If this parameter is specified, the **oid** parameter cannot be specified. See "SNMP Defined MIB Names" on page 32-21.

The **type** parameter specifies whether a particular OID sub-tree entry is included or excluded from the SNMP view.

Examples

To add SNMP view "mib2view" that includes all objects in the MIB-II sub-tree, use the command:

```
add snmp view=mib2view oid=1.3.6.1.2.1 type=include
```

Related Commands

delete snmp view
show snmp view

create snmp community

Syntax

```
CREate SNmp COMmunity=name [ACCess={Read|Write}]
   [TRAPhost=ipadd] [MAnager=prefix[/0..32]]
   [OPen={ON|OFF|YES|NO|True|False}] [V1traphost=ipadd]
   [V2ctraphost=ipadd]
```

where:

- *name* is a string 1 to 15 characters long. It may contain any printable character and is case sensitive. If the string contains spaces, it must be in double quotes.
- *ipadd* is an IP address in dotted decimal notation.
- *prefix* is an IP address range in dotted decimal notation optionally followed by a "/" character and the number of bits in the network mask.

Description

This command creates an SNMP community, optionally setting the access mode for the community and defining a trap host and manager. This command requires a user with security officer privilege when the switch is in security mode.

The **community** parameter specifies the name of the community. The community name is used to reference the SNMP community in all other SNMP commands. A community with the specified name must not already exist in the switch.

The **access** parameter specifies the access mode for this community. If **read** is specified, management stations in this community can only read MIB variables from the switch, that is perform SNMP get or get-next operations. If **write** is specified, management stations in this community can read and write MIB variables, that is perform SNMP set, get and get-next operations. The default is **read**.

The **traphost**, **v1traphost** and **v2ctraphost** parameters specify a trap host for the SNMP community. This is the IP address of a device to which traps generated by the switch are to be sent. A community may have more than one trap host, but only one can be specified when the community is created. If the parameter is not specified, the community has no defined trap host. If the **traphost** or **v1traphost** parameter is used to specify a trap host, the switch sends SNMPv1 format traps to this host. If the **v2ctraphost** parameter is used to specify a trap host, the switch sends SNMPv2c format traps to this host. The same trap host can be used for sending version 1 and version 2c traps, but you have to add it to the SNMP community twice.

The **manager** parameter can specify either a single management station or a range of management stations for this SNMP community. If no parameter is specified, then the community will have no management station defined to it. Note that additional management stations or ranges can be created once the community has been created.

The **open** parameter allows access to this community by any management station, and overrides the management stations defined with the **manager** parameter. The default is **off**.

Additional trap hosts and management stations can be defined for a the community using the **add snmp community** command.

Examples To create an SNMP community called "public" with read only access to all MIB variables from any management station, use the command:

```
cre sn com=public op=on
```

Related Commands

add snmp community
delete snmp community
destroy snmp community
disable snmp
disable snmp community
disable snmp community trap
enable snmp
enable snmp community trap
enable snmp community trap
show snmp community

delete snmp community

Syntax

```
DELete SNmp COMmunity=name [TRAPhost=ipadd]
   [MAnager=prefix[/0..32]] [V1traphost=ipadd]
   [V2ctraphost=ipadd]
```

where:

- *name* is a string 1 to 15 characters long. It may contain any printable character and is case sensitive. If the string contains spaces, it must be in double quotes.
- *ipadd* is an IP address in dotted decimal notation.
- *prefix* is an IP address range in dotted decimal notation optionally followed by a "/" character and the number of bits in the network mask.

Description

This command deletes a trap host or management station from the specified SNMP community.

The **community** parameter specifies the SNMP community. The community must already exist on the switch.

The **traphost**, **v1traphost** or **v2ctraphost** parameters specify a trap host to be deleted for the SNMP community. This is the IP address of a device to which traps generated by the switch are currently sent. The **traphost** or **v1traphost** parameters specify an SNMPv1 trap host. The **v2ctraphost** parameter specifies an SNMPv2c trap host. If both SNMPv1 and SNMPv2c trap hosts exist for the same trap host, you must delete the two versions separately using the same **delete** command.

The **manager** parameter can specify either a single management station or a range of management stations for this SNMP community.

Examples

To delete the host 192.168.1.1 as a trap host from the community named Administration, use the command:

```
del sn com=Administration trap=192.168.1.1
```

Related Commands

add snmp community
create snmp community
destroy snmp community
disable snmp community
disable snmp community trap
enable snmp community trap
enable snmp community trap
show snmp community

delete snmp group

Syntax `DELete SNmp GROup=`*`group-name`*

where *group-name* is a string 1 to 32 characters long. It may contain any printable character and is case sensitive. If *group-name* contains spaces, it must be in double quotes.

Description This command deletes an SNMP group, and is used with SNMP version 3 only.

The **group** parameter specifies the SNMP group. A group with the specified name must already exist in the switch.

Examples To delete SNMP group "usergroup", use the command:

```
del sn gro=usergroup
```

Related Commands
set snmp group
show snmp group

delete snmp targetaddr

Syntax `DELete SNmp TARgetaddr=`*`address-name`*

where *address-name* is a string 1 to 32 characters long. It may contain any printable character and is case sensitive. If *address-name* contains spaces, it must be in double quotes.

Description This command deletes the SNMP target address entry from the snmpTargetAddrTable, and is used with SNMP version 3 only.

The **targetaddr** parameter specifies an SNMP target address entry in the snmpTargetAddrTable. An SNMP target address entry with the specified name must exist in the switch.

Examples To delete SNMP target address "target", use the command:

```
del sn tar=target
```

Related Commands
set snmp targetaddr
show snmp targetaddr

delete snmp targetparams

Syntax `DELete SNmp TARGETParams=`*`params-name`*

where *params-name* is a string 1 to 32 characters long. It may contain any printable character and is case sensitive. If *params-name* contains spaces, it must be in double quotes.

Description This command deletes the SNMP target params entry from the snmpTargetParamsTable, and is used with SNMP version 3 only.

The **targetparams** parameter specifies an SNMP target params entry in the snmpTargetParamsTable. An SNMP target params entry with the specified name must exist in the switch.

Examples To delete SNMP target params "params", use the command:

```
del sn targetp=params
```

Related Commands
add snmp targetparams
set snmp targetparams
show snmp targetparams

delete snmp user

Syntax `DELete SNmp USer=`*`user-name`*

where *user-name* is a string 1 to 32 characters long. It may contain any printable character and is case sensitive. If *user-name* contains spaces, it must be in double quotes.

Description This command deletes an SNMP user and is used with SNMP version 3 only.

The **user** parameter specifies the SNMP user. A user with the specified name must already exist in the switch.

Examples To delete the SNMP user “noauthuser,” use the command:

```
del sn us=noauthuser
```

Related Commands
set snmp user
show snmp user

delete snmp view

Syntax

```
DELete SNmp VIEW=view-name
   {OID=oid-tree|MIB={mib-name|ALL}}
```

where:

- *view-name* is a string 1 to 32 characters long. It may contain any printable character and is case sensitive. If *view-name* contains spaces, it must be in double quotes.
- *oid-tree* is character string in a format of decimal and dot that is from 1 to 32 sub-identifiers long.
- *mib-name* is a string 1 to 32 characters long. It may contain any printable character and is case sensitive. If *mib-name* contains spaces, it must be in double quotes.

Description

This command deletes the specified pair of SNMP view and the sub-tree. A specified pair must already exist in the switch.

The **view** parameter specifies the SNMP view. A view with the specific name must already exist in the switch.

The **oid** parameter specifies the object identifier of the ASN.1 sub-tree to be included or excluded from the view. The sub-tree has to be specified as a character string consisting of numbers, for example, 1.3.6.1.2.1. If this parameter is specified, the **mib** parameter cannot be specified. A sub-tree with the specific **oid** must already exist in the switch.

The **mib** parameter specifies a predefined MIB node to be included or excluded from the view. If this parameter is specified, the **oid** parameter cannot be specified. A sub-tree with the specified MIB node must already exist in the switch. If **all** is specified, then all pairs of the specific SNMP view and associated sub-trees are deleted.

Examples

To delete SNMP view “mib2view”, with all associated sub-trees, use the command:

```
del sn view=mib2view mib=all
```

Related Commands

add snmp view
show snmp view

destroy snmp community

Syntax `DESTroy SNmp COMmunity=`*`name`*

where *name* is a string 1 to 15 characters long. It may contain any printable character and is case sensitive. If *name* contains spaces, it must be in double quotes.

Description This command destroys an existing SNMP community.

The **community** parameter specifies the SNMP community. The community must already exist on the switch.

Related Commands
add snmp community
create snmp community
disable snmp community
disable snmp community trap
enable snmp community trap
enable snmp community trap
show snmp community

disable snmp

Syntax `DISable SNmp`

Description This command disables the switch's SNMP agent, and disables the RMON MIB from gathering information (Appendix C, SNMP MIBs). SNMP packets sent to the switch are treated as unknown protocol packets by the underlying transport layer (UDP) and traps are not generated by the switch.

Related Commands
disable snmp community
disable snmp community trap
enable snmp
enable snmp community trap
enable snmp community trap
show snmp community

disable snmp authenticate_trap

Syntax `DISable SNmp AUTHenticate_trap`

Description This command disables the generation of authentication failure traps by the SNMP agent whenever an SNMP authentication failure occurs.

Related Commands
disable snmp
enable snmp
enable snmp authenticate_trap
show snmp community

disable snmp community

Syntax `DISable SNmp COMmunity=`*`name`*

where *name* is a string 1 to 15 characters long. It may contain any printable character and is case sensitive. If *name* contains spaces, it must be in double quotes.

Description This command disables a particular SNMP community.

The **community** parameter specifies the SNMP community. The community must already exist on the switch. When a community is disabled, packets for the community are processed as if the community does not exist and traps are not generated for the community. The SNMP agent generates an authentication error if a packet is received for a disabled community.

Related Commands
disable snmp
disable snmp community trap
enable snmp
enable snmp community trap
enable snmp community trap
show snmp community
show snmp community

disable snmp community trap

Syntax `DISable SNmp COMmunity=`*`name`* `TRAP`

where *name* is a string 1 to 15 characters long. It may contain any printable character and is case sensitive. If *name* contains spaces, it must be in double quotes.

Description This command disables the generation of trap messages for a community. Trap messages are not sent to the community's trap host(s), but all other SNMP operations proceed normally.

The **community** parameter specifies the SNMP community. The community must already exist on the switch.

Related Commands
disable snmp
disable snmp community
enable snmp
enable snmp community trap
enable snmp community trap
show snmp community
show snmp community

enable snmp

Syntax `ENAble SNmp`

Description This command enables the switch's SNMP agent, and requires a user with security officer privilege when the switch is in security mode. The SNMP agent receives and processes SNMP packets sent to the switch and generate traps.

This command enables the switch's SNMP agent, and enables the RMON MIB to gather information (Appendix C, SNMP MIBs). The SNMP agent receives and processes SNMP packets sent to the switch and generate traps.

By default, the SNMP agent is disabled. This command is required to enable SNMP to operate at boot.

Related Commands
disable snmp
disable snmp community
disable snmp community trap
enable snmp community trap
enable snmp community trap
show snmp community

enable snmp authenticate_trap

Syntax `ENAble SNmp AUTHenticate_trap`

Description This command enables the generation of authentication failure traps by the SNMP agent whenever an SNMP authentication failure occurs.

By default, the generation of authentication traps is disabled. This command is required to enable SNMP authentication failure traps at boot.

Related Commands
disable snmp
disable snmp authenticate_trap
enable snmp
show snmp community

enable snmp community

Syntax `ENAble SNmp COMmunity=`*`name`*

where *name* is a string 1 to 15 characters long. It may contain any printable character and is case sensitive. If *name* contains spaces, it must be in double quotes.

Description This command enables a particular SNMP community. This command requires a user with Security Officer privilege when the switch is in security mode.

The **community** parameter specifies the SNMP community. The community must already exist on the switch. When a community is enabled, the SNMP agent processes SNMP packets for the community and generates traps to trap hosts in the community, if traps are also enabled. SNMP communities are enabled when they are created, but traps are not enabled for the community.

Examples To create an SNMP community and enable it, use the following commands:

```
cre sn com=private trap=192.168.1.1 ma=192.168.1.1
ena sn com=private
```

Related Commands
disable snmp
disable snmp community
disable snmp community trap
enable snmp
enable snmp community trap
show snmp community

enable snmp community trap

Syntax `ENAble SNmp COMmunity=`*`name`* `TRap`

where *name* is a string 1 to 15 characters long. It may contain any printable character and is case sensitive. If *name* contains spaces, it must be in double quotes.

Description This command enables the generation of trap messages for a community. Only traps for the community should be enabled, not the entire operation of the community. Trap messages are sent to the community's trap hosts. This command requires a user with Security Officer privilege when the switch is in security mode.

The **community** parameter specifies the SNMP community. The community must already exist on the switch. When a community is enabled, the SNMP agent processes SNMP packets for the community and generates traps to trap hosts in the community, if traps are also enabled. SNMP communities are enabled when they are created, but traps are not enabled for the community.

Examples To create an SNMP community and enable traps on it, use the following commands:

```
cre sn com=private trap=192.168.1.1 ma=192.168.1.1
ena sn com=private tr
```

Related Commands
disable snmp
disable snmp community
disable snmp community trap
enable snmp
enable snmp community
show snmp community

purge snmp

Syntax `PURge SNmp`

Description This command disables the SNMP agent, and deletes all SNMP configuration elements (communities, groups, target addresses, users and views). This command applies to all versions of SNMP (v1, v2 and v3)

Related Commands **disable snmp**

set snmp asnberpadding

Syntax `SET SNmp ASNberpadding={YES|NO|ON|OFF|True|False}`

Description This command enables or disables the padding of ASN.1/BER encoded integer values with 0x00 in SNMP get-response and trap messages. Padding enables legacy network management systems to correctly decode the integer value. Padding is disabled by default.

The **asnberpadding** parameter specifies whether padding is enabled or disabled. Specify **yes**, **on**, or **true** to pad integer values with 0x00. Specify **no**, **off**, or **false** to stop padding integer values. The default is **off**.

By default, SNMP encodes integer values in the smallest possible number of octets. If an integer value consists of more than one octet, and the first nine bits (the first octet and bit eight of the second octet) are all ones or all zeros, the first octet is omitted from the encoding. This process is repeated until the smallest possible encoding is achieved. For example, a counter32 object with a value of 4289592837 (0xffadfe05) would be encoded as:

```
41 03 ad fe 05
```

where:

`41`	is the identifier for a counter32 integer
`03`	is the length of the content (3 octets)
`ad fe 05`	is the content

When you enable padding, integer values with all ones in the first nine bits are not truncated, but padded with an extra 0x00 octet. Integer values with all zeroes in the first nine bits are still truncated. For example, the counter32 object with a value of 4289592837 (0xffadfe05) would be encoded as:

```
41 05 00 ff ad fe 05
```

where:

`41`	is the identifier for a counter32 integer
`05`	is the length of the content (5 octets)
`00 ff ad fe 05`	is the content

Examples To enable padding of ASN.1/BER encoded integer values, use the command:

```
set sn asn=on
```

Related Commands **show snmp**

set snmp community

Syntax

```
SET SNmp COMmunity=name [ACCess={Read|Write}]
   [OPen={ON|OFF|YES|NO|True|False}]
```

where *name* is a string 1 to 15 characters long. It may contain any printable character and is case sensitive. If *name* contains spaces, it must be in double quotes.

Description

This command modifies the access mode and open access configuration for the specified SNMP community. It requires a user with security officer privilege when the switch is in security mode.

The **community** parameter specifies the name of the community. A community with the specified name must already exist in the switch.

The **access** parameter specifies the access mode for this community. If **read** is specified, management stations in this community can only read MIB variables from the switch, that is perform SNMP get or get-next operations. If **write** is specified, management stations in this community can read and write MIB variables, that is perform SNMP set, get and get-next operations. The default is **read**.

The **open** parameter allows access to this community by any management station, and overrides the management stations defined with the **manager** parameter. The default is **off**.

Examples

To disable access from any management station for an SNMP community called “public”, use the command:

```
set sn com=public op=off
```

Related Commands

create snmp community
destroy snmp community
disable snmp community
disable snmp community trap
enable snmp community
enable snmp community trap
show snmp community

set snmp engineid

Syntax `SET SNmp ENGineid=`*`snmpEngineID`*

where *snmpEngineID* is a string of hexadecimal characters having a minimum length of 5 octets and maximum length of 32 octets. There are two hexadecimal characters in each octet.

Description This command modifies the local snmpEngineID, and is used with SNMP version 3 only.

The **engineid** parameter specifies the snmpEngineID. The value for this object shall not be all zeros or ‘ff’H or the empty (zero length) string.

A user's password (entered on the command line) is converted to an MD5 or SHA security digest. This digest is based on both the password and the local engine ID. The command line password is then destroyed, as required by RFC 3414. Because of this deletion, if the local value of engineID changes, the security digests of SNMPv3 users are invalid, and the users must be recreated.

The default engine ID is a string of 11 octets and is constructed as follows:

- Octets 1 through 4 represent the agent's SNMP management private enterprise number as assigned by the Internet Assigned Numbers Authority (IANA). For example, “000000CF” (decimal 207) is the enterprise number for Allied Telesis in hexadecimal. The most significant bit of octet 1 is always equal to "1", so the first four octets in the default SNMP engine ID is always “800000CF”.
- Octet 5 is always 03 in hexadecimal and indicates that the next set of values represent a MAC address. Octets 6 through 11 comprise the MAC address. The Engine ID must be a unique number among the various SNMP engines in the management domain. Using the default engine ID based on MAC addresses should ensure this uniqueness.

MAC addresses are initially assigned to an interface card by its manufacturer and should be unique. However, it is possible for users to change these values. This text assumes that the MAC addresses contained within the MAC address table are those originally supplied by the card manufacturer.

Examples To add or modify an SNMP local engine ID containing the AlliedTelesis IANA enterprise number and a MAC address of 00-00-CD-12-34-56, use the command:

```
set sn eng=800000cf030000cd123456
```

In this example, the IANA enterprise number, with initial bit set to 1 is 800000CF, the fifth octet is 03, and the MAC address is 0000CD123456.

Related Commands **show snmp**

set snmp group

Syntax

```
SET SNmp GROup=group-name [READview=view-name]
   [WRITEview=view-name] [NOTIFyview=view-name]
```

where:

- *group-name* is a string 1 to 32 characters long. It may contain any printable character and is case sensitive. If *group-name* contains spaces, it must be in double quotes.
- *view-name* is a string 1 to 32 characters long. It may contain any printable character and is case sensitive. If *group-name* contains spaces, it must be in double quotes.

Description

This command modifies an SNMP group parameters, and is used with SNMP version 3 only.

The **group** parameter specifies the SNMP group. A group with the specified name must already exist in the switch.

The **readview** parameter specifies the MIB contents that this group can read.

The **writeview** parameter specifies the MIB contents that this group can modify.

The **notifyview** parameter specifies the notify contents that this group can receive.

Examples

To set read view “user-view” in SNMP group “usergroup”, use the command:

```
set sn gro=usergroup read=user-view
```

Related Commands

add snmp group
delete snmp group
show snmp group

set snmp local

Syntax `SET SNmp LOCal={NONE|1..15} [VERsion={V1|V2|V3|ALL}]`

Description This command sets the local interface to be used with a particular version of SNMP. Once set, the IP address of the local interface specified is used as the source IP address for all SNMP packets of the version specified.

The **version** parameter specifies the version of SNMP packets to which the local interface applies. The default is **all**.

The **local** parameter specifies a local interface to be used as the source IP address for all packets of a particular SNMP version that the switch generates and sends. The local interface IP address is also be used as the SNMP agent IP address in these outgoing packets. The local interface must already be configured and fall in the range 1-15. If no local interface has been set for SNMP, the switch selects a source address on the basis of the route. This means the source address is the IP address from which the SNMP packet is issued.

Examples To set the local interface 5 for SNMPv3 packets, use the command:

```
set sn loc=5 ver=v3
```

Related Commands show snmp

set snmp targetaddr

Syntax

```
SET SNmp TARgetaddr=address-name [IP=target-ipadd]
   [UDP=udp-port] [PARams=params-name]
```

where:

- *address-name* is a string 1 to 32 characters long. It may contain any printable character and is case sensitive. If *address-name* contains spaces, these must be in double quotes.
- *target-ipadd* is an IP address in dotted decimal notation.
- *udp-port* is a decimal number from 1 to 255.
- *params-name* is a string 1 to 32 characters long. It may contain any printable character and is case sensitive. If *params-name* contains spaces, it must be in double quotes.

Description

This command modifies the SNMP target address entry in the snmpTargetAddrTable, and is used with SNMP version 3 only.

The **targetaddr** parameter specifies the SNMP target address entry in the snmpTargetAddrTable. An SNMP target address entry with the specified name must already exist in the switch.

The **ip** parameter specifies the IP address to which notification are sent.

The **udp** parameter specifies the identification number of a UDP port. If this parameter is not specified then the default value of 162 is used.

The **params** parameter specifies the reference to the entry in snmpTargetParamsTable.

Examples

To modify the SNMP target address "target" to use the IP address "1272.0.0.1", use the command:

```
set sn tar=target ip=127.0.0.1
```

Related Commands

add snmp targetaddr
delete snmp targetaddr
show snmp targetaddr

set snmp targetparams

Syntax

```
SET SNmp TARGETParams=params-name
   [SECuritylevel={NOAUthnopriv|AUTHnopriv|AUTHPRiv}]
   [USer=user-name]
```

where:

- *params-name* is a string 1 to 32 characters long. It may contain any printable character and is case sensitive. If *params-name* contains spaces, it must be in double quotes.
- *user-name* is a string 1 to 32 characters long. It may contain any printable character and is case sensitive. If *user-name* contains spaces, it must be in double quotes.

Description

This command modifies the SNMP target params entry in the snmpTargetParamsTable, and is used with SNMP version 3 only.

The **targetparams** parameter specifies an SNMP target params entry in the snmpTargetParamsTable. An SNMP target params entry with the specified name must exist in the switch.

The **securitylevel** parameter specifies the security level for this target parameters. **noauthnopriv** specifies that no authentication or privacy is used. **authnopriv** specifies that authentication be based on the MD5 or SHA algorithm. This option provides security but no privacy. **authpriv** specifies that authentication be based on the MD5 or SHA algorithm, and encryption is to be based on the DES algorithm.

The **user** parameter specifies the SNMP user. A user with the specified name does not need to exist in the switch. It can be added later.

If the user's authentication and privacy protocols do not support the specified target parameters security level, then the SNMP message is not sent.

Examples

To modify the user in the SNMP target params "params" to use user "test", use command:

```
set sn targetp=params us=test
```

Related Commands

add snmp targetparams
delete snmp targetparams
show snmp targetparams

set snmp trapdelay

Syntax `SET SNmp TRapdelay=10..600`

Description This command allows you to configure a delay on transmitting SNMP traps at start up. Any traps that occur before the delay expires are held in a queue. Once the delay expires, the switch transmits all queued SNMP traps.

SNMP traps are delayed by a minimum of 10 seconds at start up. This allows time for the links to come up on the switch. Use this command to extend the delay if you sometimes find that this is not enough time for other network protocols to converge and open up transmission paths to the SNMP management station.

The **trapdelay** parameter specifies the time delay, in seconds, before the switch transmits SNMP traps. The default is 10 seconds.

Examples To configure a delay of one minute before SNMP traps are transmitted at start up, use the command:

```
set sn tr=60
```

Related Commands **enable snmp**
show snmp

set snmp user

Syntax

```
SET SNmp USer=user-name [GROup=group-name]
   [AUTHprotocol={NONE|MD5|SHa}] [AUTHPAssword=password]
   [PRIVprotocol={NONE|DES}] [PRIVPAssword=password]
```

where:

- *user-name* is a string 1 to 32 characters long. It may contain any printable character, and is case sensitive. If *user-name* contains spaces, it must be in double quotes.
- *group-name* is a string 1 to 32 characters long. It may contain any printable character, and is case sensitive. If *group-nam*e contains spaces, it must be in double quotes.
- *password* is a string 8 to 32 characters long. It may contain any printable character, and is case sensitive. If *password* contains spaces, it must be in double quotes.

Description

This command modifies an existing SNMP user details, and is used with SNMP version 3 only.

The **user** parameter specifies the SNMP user. A user with the specified name must already exist in the switch.

The **group** parameter specifies the SNMP group. Although a group with the specified name need not pre-exist in the switch, it must be added (using the **add snmp group** command) before user access can be enabled.

The **authprotocol** parameter specifies the authentication protocol for the SNMP user (either **md5**, **sha**, or **none**).

The **authpassword** parameter specifies the authentication password for the SNMP user. If **authprotocol** is set to **md5** or **sha**, the **authpassword** must be specified.

The **privprotocol** parameter specifies privacy protocol for the SNMP user (either **des** or **none**). The **privprotocol** cannot be set to **des** if the **authprotocol** is set to **none**.

The **privpassword** parameter specifies the encryption password for the SNMP user. If **privprotocol** is set to **des**, the **privpassword** must be specified.

If authentication and privacy are turned on and you change the authentication type, i.e. **md5** to **sha**, you are prompted to re-enter a privacy password. This is because the password is used when calculating the key (or digest). However, once the key is created, the password is deleted and thus no longer available if a new key is required.

Examples

To set the authentication password for the SNMP user “Sam” to “authpassword” and to use the MD5 authentication protocol, use the command:

```
set sn us=Sam auth=md5 authpa=authpassword
```

Related Commands

add snmp user
delete snmp user
show snmp user

show snmp

Syntax `SHow SNmp`

Description This command displays information about the switch's SNMP agent (Figure 32-7, Table 32-9 on page 32-53).

Figure 32-7: Example output from the **show snmp** command

```
SNMP configuration:
  Status ........................... Enabled
  ASN.01 BER Padding ............... Off
  Authentication failure traps .... Enabled
  Trap start up delay .............. 10 (secs)
  Local Interface SNMPv1 ........... Not Set
  Local Interface SNMPv2 ........... local5
  Local Interface SNMPv3 ........... local2

  Community ........................ public
    Access ......................... read-only
    Status ......................... Enabled
    Traps .......................... Enabled
    Open access .................... Yes
  Community ........................ Administration
    Access ......................... read-write
    Status ......................... Disabled
    Traps .......................... Disabled
    Open access .................... No

SNMPv3 engine information
    snmpEngineID ................... 800000cf03aabbccdd11
    snmpEngineBoots ................ 5
    snmpEngineTime ................. 3

SNMP counters:
  inPkts ........................... 0       outPkts ........................... 0
  inBadVersions .................... 0       outTooBigs ........................ 0
  inBadCommunityNames .............. 0       outNoSuchNames .................... 0
  inBadCommunityUses ............... 0       outBadValues ...................... 0
  inASNParseErrs ................... 0       outGenErrs ........................ 0
  inTooBigs ........................ 0       outGetRequests .................... 0
  inNoSuchNames .................... 0       outGetNexts ....................... 0
  inBadValues ...................... 0       outSetRequests .................... 0
  inReadOnlys ...................... 0       outGetResponses ................... 0
  inGenErrs ........................ 0       outTraps .......................... 0
  inTotalReqVars ................... 0
  inTotalSetVars ................... 0
  inGetRequests .................... 0
  inGetNexts ....................... 0
  inSetRequests .................... 0
  inGetResponses ................... 0
  inTraps .......................... 0

SNMPv3 counters:
  UnsupportedSecLevels ............. 0       UnknownSecurityModels ............. 0
  NotInTimeWindows ................. 0       InvalidMsgs ....................... 0
  UnknownUserNames ................. 0       UnknownPDUhandlers ................ 0
  UnknownEngineIDs ................. 0
  WrongDigests ..................... 0
  DecryptionErrors ................. 0
```

Table 32-9: Parameters in output of the **show snmp** command

Parameter	Meaning
Status	Whether the SNMP agent or the specified community is enabled.
ASN.01 BER Padding	Whether ASN.1/BER encoded integer values are padded with a leading 0x00 octet; one of "Off" or "On".
Authentication failure traps	Whether the SNMP agent is enabled to generate a trap on an authentication failure for an incoming SNMP packet.
Trap start up delay	The delay on transmitting SNMP traps at start up. Any traps that occur before the delay expires are sent once the delay ends.
Local Interface	The local interface used as the source in outgoing SNMP messages.
Community	The name of an SNMP community on the switch.
Access	Whether access rights for the SNMP community is read-only or read-write.
Status	Whether the community is enabled or disabled.
Traps	Whether the community generates traps.
Open access	Whether the SNMP community is open to access from all IP addresses.
snmpEngineID	The ID assigned to the local SNMP engineID.
snmpEngineBoots	A count of the number of times the SNMP engine has been re-booted or re-initialized since snmpEngineID was last configured.
snmpEngineTime	The number of seconds since the snmpEngineBoots counter was last incremented.
inPkts	The number of SNMP packets received by the switch.
inBadVersions	The number of SNMP packets with a bad version field received by the switch.
inBadCommunityNames	The total number of SNMP PDUs delivered to the SNMP agent that used an unknown SNMP community name.
inBadCommunityUses	The total number of SNMP PDUs delivered to the SNMP agent that represented an SNMP operation not allowed by the SNMP community name in the PDU.
inASNParseErrs	The total number of ASN.1 parsing errors, either in encoding or syntax, encountered by the SNMP agent when decoding received SNMP PDUs.
inTooBigs	The total number of valid SNMP PDUs delivered to the SNMP agent for which the value of the errorStatus component was tooBig.
inNoSuchNames	The number of SNMP packets received with an error status of nosuchname.
inBadValues	The number of SNMP packets received with an error status of badvalue.
inReadOnlys	The number of SNMP packets received with an error status of readonly.
inGenErrs	The number of SNMP packets received with an error status of generr.
inTotalReqVars	The total number of SNMP MIB objects requested.

Table 32-9: Parameters in output of the **show snmp** command (Continued)

Parameter	Meaning
inTotalSetVars	The total number of SNMP MIB objects which were changed.
inGetRequests	The number of SNMP Get Request packets received by the switch.
inGetNexts	The number of SNMP Get Next Request packets received by the switch.
inSetRequests	The number of SNMP Set Request packets received by the switch.
inGetResponses	The number of SNMP Get Response packets received by the switch.
inTraps	The number of SNMP trap message packets received by the switch.
outPkts	The number of SNMP packets transmitted by the switch.
outTooBigs	The number of SNMP packets transmitted with an error status of toobig.
outNoSuchNames	The number of SNMP packets transmitted with an error status of nosuchname.
outBadValues	The number of SNMP packets transmitted with an error status of badvalue.
outGenErrs	The number of SNMP packets transmitted with an error status of generror.
outGetRequests	The number of SNMP Get Request response packets transmitted by the switch
outGetNexts	The number of Get Next response packets transmitted by the switch.
outSetRequests	The number of Set Request packets transmitted by the switch.
outGetResponses	The number of SNMP Get response packets transmitted.
outTraps	The number of SNMP Traps transmitted by the switch.
UnknownSecurityModels	The number of SNMP packets received with a requested for unknown security level.
InvalidMsgs	The number of SNMP packets with invalid components in the SNMP message.
UnknownPDUHandlers	The number of SNMP packets with an unknown PDU.
UnsupportedSecLevels	The number of SNMP packets with an unsupported security level.
NotInTimeWindows	The number of SNMP packets outside the SNMP engine time window.
UnknownUserNames	The number of SNMP packets with an unknown user name.
UnknownEngineIDs	The number of SNMP packets with an unknown SNMP. engineID
WrongDigests	The number of SNMP packets with wrong digest value.
DecryptionErrors	The number of SNMP packets that could not be decrypted.

Related Commands

create snmp community
disable snmp
disable snmp authenticate_trap
disable snmp community
disable snmp community trap
enable snmp
enable snmp authenticate_trap
enable snmp community
enable snmp community trap
set snmp asnberpadding
set snmp community
show snmp community
show snmp group
show snmp user
show snmp targetaddr
show snmp view

show snmp community

Syntax `SHow SNmp COMmunity=`*`name`*

where *name* is a string 1 to 15 characters long. It may contain any printable character and is case sensitive. If the string contains spaces, it must be in double quotes.

Description This command displays information about a single SNMP community (Figure 32-8, Table 32-10).

The **community** parameter specifies the name of the community. A community with the specified name must already exist in the switch.

Figure 32-8: Example output from the **show snmp community** command

```
SNMP community information:
  Name ......................... public
  Access ....................... read-only
  Status ....................... Enabled
  Traps ........................ Enabled
  Open access .................. Yes
  Manager ...................... 192.168.1.1
  Manager ...................... 192.168.5.3
  Trap host .................... 192.168.1.1
  Trap host .................... 192.168.6.23
  V2c trap host ................ 192.168.6.23
```

Table 32-10: Parameters in the output of the **show snmp community** command

Parameter	Meaning
Name	Name of the community. This identifies the community and appears in SNMP messages for this community.
Access	Whether access rights for the SNMP community is read-only or read-write.
Status	Whether the community is enabled or disabled.
Traps	Whether the community generates trap messages.
Open access	Whether the community is open to access from all IP addresses.
Manager	IP address of a management station or IP range of stations that can access this switch using this community.
Trap host	IP address of a trap host to which traps for this community is sent.
V2c trap host	IP address of a trap host to which SNMP version 2c traps for this community is sent.

Related Commands

create snmp community
disable snmp community
disable snmp community trap
enable snmp community
enable snmp community trap
show snmp
show snmp group
show snmp user
show snmp targetaddr
show snmp view

show snmp group

Syntax `SHow SNmp GROup[=group-name]`

where *group-name* is a string 1 to 32 characters long. It may contain any printable character, and is case sensitive. If *group-name* contains spaces, it must be in double quotes.

Description This command is used with SNMP version 3 only, and displays information about a one or more SNMP groups (Figure 32-9, Table 32-11).

The **group** parameter specifies the SNMP group. A group with the specified name must already exist in the switch. If *group-name* is not specified, then all existing groups are presented.

Figure 32-9: Example output from the **show snmp group** command

```
SNMP group information:
  Group Name ............... usergroup
    Security Level ........ noAuthNoPriv
    Read View ............. read-view
    Write View ............ write-view
    Notification View ..... notification-view
    Row Status..............active
```

Table 32-11: Parameters in output of the **show snmp group** command

Parameter	Meaning
Group Name	Name of the SNMP group.
MinimumSecurity Level	Whether the minimum security level for the SNMP group is noAuthNoPriv, authNoPriv, or authPriv.
Read View	Name of a predefined view that gives read rights to the members of the specified SNMP group.
Write View	Name of a predefined view that gives write rights to the members of the specified SNMP group.
Notification View	Name of a predefined view that gives notification rights to the members of the specified SNMP group.
Row Status	Whether the status of the displayed group is active, not in service, or not ready.

Examples To display information on the group called “adminusers”, use the command:

```
show snmp group=adminusers
```

To display information on all groups, use the command:

```
show snmp group
```

Related Commands
add snmp group
delete snmp group
set snmp group

show snmp targetaddr

Syntax `SHow SNmp TARgetaddr[=address-name]`

where *address-name* is a string 1 to 32 characters long. It may contain any printable character, and is case sensitive. If *address-name* contains spaces, it must be in double quotes.

Description This command displays information about a one of more SNMP target addresses, and is used with SNMP version 3 only (Figure 32-10, Table 32-12).

The **targetaddr** parameter specifies the SNMP target address name. A target address with the specified name must already exist in the switch. If the target *address-name* is not specified, then all existing target addresses are presented.

Figure 32-10: Example output from the **show snmp targetaddr** command

```
SNMP target addresss information:
  Target Address Name ..... target
    IP address ............ 172.20.70.1
    UDP port .............. 162
    Target Address Params.. user
    Row Status ............ active
```

Table 32-12: Parameters in output of the **show snmp targetaddr** command

Parameter	Meaning
Target Address Name	Name of the SNMP target address.
IP address	IP address of SNMP target address.
UDP port	UDP port of SNMP target address.
Target Address Parameter	Reference to the entry specified in the snmpTargetParamsTable.
Row Status	Whether the status of the displayed SNMP target address is active, not in service, or not ready.

Example To show the target address for the target called "Adminserver" use the command:

```
show snmp targetaddr=Adminserver
```

Related Commands

add snmp targetaddr
delete snmp targetaddr
set snmp targetaddr

show snmp targetparams

Syntax `SHow SNmp TARGETParams[=params-name]`

where *params-name* is a string 1 to 32 characters long. It may contain any printable character and is case sensitive. If *params-name* contains spaces, it must be in double quotes.

Description This command displays information about a single SNMP target params and is used on version 3 only (Figure 32-11, Table 32-13).

The **targetparams** parameter specifies an SNMP target params entry in the snmpTargetParamsTable. An SNMP target params entry with the specified name must exist in the switch. If the target *params-name* is not specified, all existing target params are presented.

Figure 32-11: Example output from the **show snmp targetparams** command

```
SNMP target params information:
  Target Params Name ..... params
    Security Level ....... authPriv
    User Name ............ test
    Row Status ........... active
```

Table 32-13: Parameters in output of the **show snmp targetparams** command

Parameter	Meaning
Target Params Name	Name of the SNMP target parameters
Security Level	Whether the security level for the SNMP messages to be sent to the target address is noAuthNoPriv, authNoPriv, or authPrive.
User Name	Name of user who receives notification on target address
Row Status	Whether the status of the displayed SNMP target address is active, not in service, or not ready.

Example To display the target parameters for the target called "Adminserverparams" use the command:

```
sh sn targetp=Adminserverparams
```

Related Commands **delete snmp targetparams**
set snmp targetparams

show snmp user

Syntax `SHow SNmp USer[=`*`user-name`*`]`

where *user-name* is a string 1 to 32 characters long. It may contain any printable character and is case sensitive. If *user-name* contains spaces, it must be in double quotes.

Description This command displays information about one or more SNMP users, and is used with SNMP version 3 only (Figure 32-12, Table 32-11).

The **user** parameter specifies the SNMP user. A user with the specified name must already exist in the switch. If *user-name* is not specified, then all existing users are presented.

Figure 32-12: Example output from the **show snmp user** command

```
SNMP user information:
  User Name ............... user
    Group Name ............ usergroup
    Auth Protocol ......... MD5
    Priv Protocol ......... DES
    Row Status ............ active
```

Table 32-14: Parameters in output of the **show snmp user** command

Parameter	Meaning
User Name	Name of the SNMP user.
Group Name	Name of the SNMP group.
Auth Protocol	Whether the authentication protocol for the SNMP user is MD5, SHA, or none.
Priv Protocol	Whether the privacy protocol for the SNMP user is DES or none.
Row Status	Whether the status of the displayed SNMP user is active, not in service, or not ready.

Example To display the details for a user called “Sam”, use the command:

```
show snmp user=Sam
```

To display the details of all users, use the command:

```
show snmp user
```

Related Commands
add snmp user
delete snmp user
set snmp user

show snmp view

Syntax `SHow SNmp VIEW[=`*`view-name`*`]`

where *view-name* is a string 1 to 32 characters long. It may contain any printable character and is case sensitive. If *view-name* contains spaces, it must be in double quotes.

Description This command displays information about specified SNMP view (Figure 32-13, Figure 32-14, Table 32-15 on page 32-62).

The **view** parameter specifies the SNMP view. A view with the specified name must already exist in the switch. If *view-name* is not specified, a list of all existing views is presented.

Figure 32-13: Example output from the **show snmp view** command

```
SNMP View information:
  SNMP View Name(s):
    readview
    writeview
    myview
```

Figure 32-14: Example output from the **show snmp view=readview** command

```
View Name ........ readview
  OID ............ 1.3.6.1.2.1
    MIB .......... mib-2
    Type ......... include
    Row Status ... active
  OID ............ 1.3.6.1.2.1.16
    MIB .......... rmon
    Type ......... exclude
    Row Status ... active
  OID ............ 1.3.6.5.6.7.8
    MIB .......... -
    Type ......... exclude
    Row Status ... active
```

Table 32-15: Parameters in output of the **show snmp view** command

Parameter	Meaning
View name	Name of the SNMP view.
OID	Object identifier of the sub-tree to be included or excluded from the view.
MIB	Predefined MIB name of the sub-tree.
Type	Whether the type for this sub-tree is include or exclude.
Row Status	Whether the status of the displayed SNMP view is active, not in service, or not ready.

Example To display the view details for the view called “Myview” use the command:

```
show snmp view=Myview
```

To display the details of all views, use the command:

```
show snmp view
```

Related Commands

add snmp view
delete snmp view
show snmp view