Chapter 38

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 router, and how to configure the router's SNMP agent. See Chapter B, SNMP MIBs for a detailed description of all MIBs (Management Information Bases) and MIB objects supported by the router.

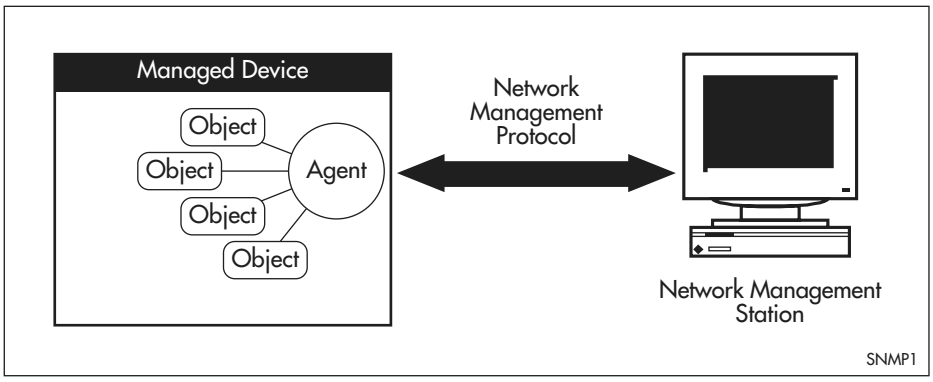
Unless a particular version of SNMP is named, "SNMP" in this chapter refers to versions SNMPv1, SNMPv2c and SNMPv3.

Network Management Framework

A network management system has three components (Figure 38-1 on page 38-3):

- 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 38-1: 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 three 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)*" (referred to as 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)*"

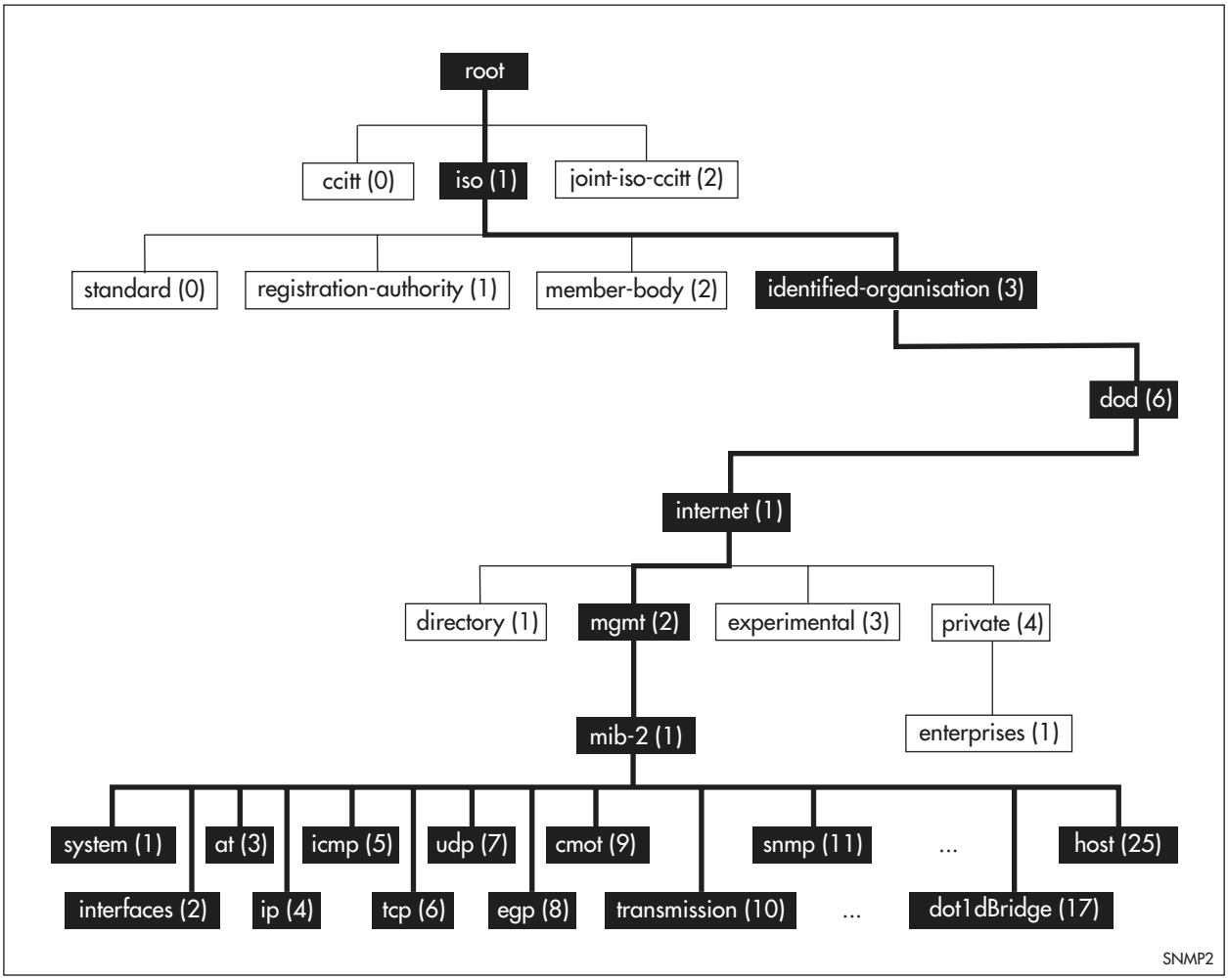
Structure of Management Information

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 38-2 on page 38-5.

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 38-2 the object named interfaces would be uniquely identified by the string of individual sub-identifiers, 1.3.6.1.2.1.2.

Figure 38-2: The 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* 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 following example 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* object 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 38-1 on page 38-7).

Table 38-1: 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 38-2 on page 38-8).

Table 38-2: 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 router) 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 router, 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 router's SNMP agent accepts SNMP messages up to the maximum UDP length the router can receive.

Note 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 used by the router) is UDP. The IP module must be enabled and configured correctly. See Chapter 14, Internet Protocol (IP) for detailed descriptions of the commands required to enable and configure IP.

SNMP Versions

The router 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 38-4 on page 38-10). 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 Chapter B, SNMP MIBs for a description of the router'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 may initiate an exchange of information is 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 38-3 on page 38-10 shows the format of an SNMP message for v1 and v2c versions. The function of the fields are described in Table 38-3 on page 38-10. There are five different SNMP PDUs (Table 38-4 on page 38-10) and six generic traps (Table 38-5 on page 38-11).

Figure 38-3: Format of an SNMP message

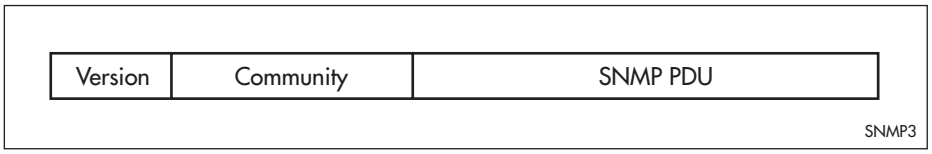


Table 38-3: Fields in an SNMP message

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 38-4: 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 38-5: 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)

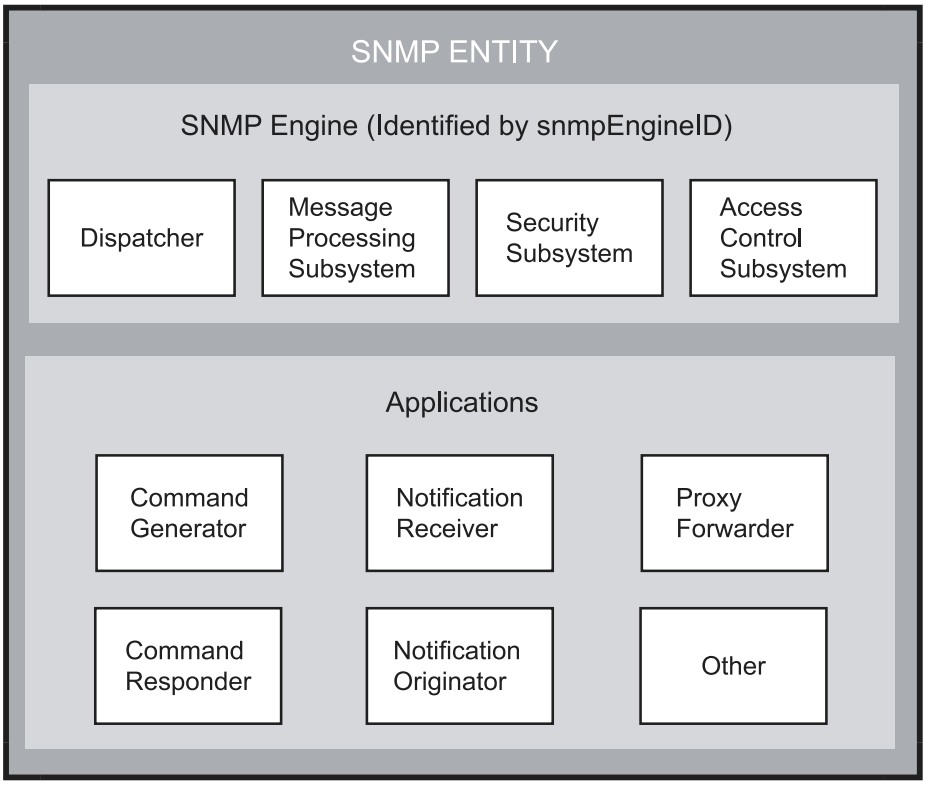
A community is a relationship between an NMS and an agent. The community name is used like a password for a trivial authentication scheme. Both SNMPv1 and SNMPv2c provide security based on the community name only. The concept of communities does not exist for SNMPv3, which instead provides for a far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

Note Removing community membership from all SNMPv3 configured devices is strongly recommended. This is to prevent alternative access to these devices via SNMPv1 and SNMv2c that 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 38-4 on page 38-12.

Figure 38-4: 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

Note 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 38-5 on page 38-13 and Table 38-6 on page 38-13.explain the protocol format of an SNMPv3 message.

Figure 38-5: SNMPv3 Protocol Format

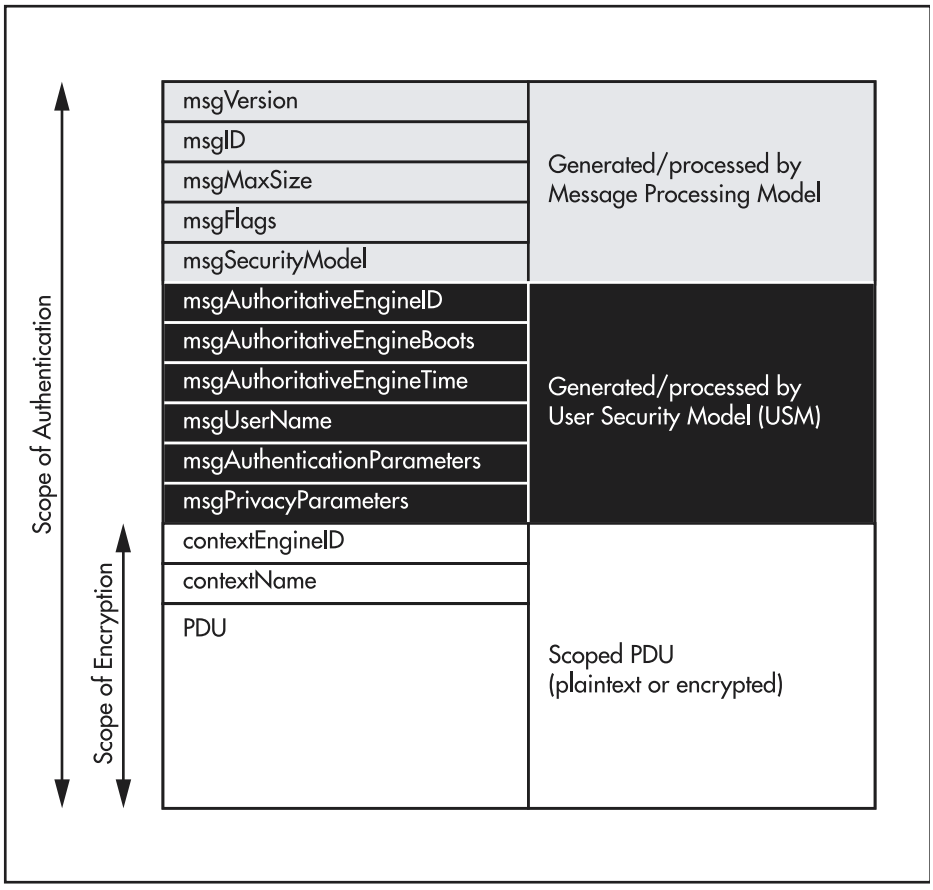


Table 38-6: SNMPv3 PDUs

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.

Table 38-6: SNMPv3 PDUs (continued)

Value	Meaning
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 router, 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 Router

Although current software levels, 2.6.3 and higher, still support the specific facilities of SNMP v1 and v2, their documentation and use is supplied primarily 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 Router on page 19.

The router'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 router's implementation of SNMPv2c is based on the RFCs listed in "Network Management Framework" on page 38-3.

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 38-7 on page 38-15).

Table 38-7: 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 router's software still supports communities, this is to provide backward compatibility with legacy management systems. For security reasons communities should NOT be used within an SNMPv3 environment.

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 router'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}]
```

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 router. 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 router, possibly affecting the operation of the router. For this reason, care must be taken with the security of community names.

Users are strongly advised not to use SNMP communities where secure network management is required, but instead use the secure network features offered by SNMPv3.

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
```

Note 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 router with minimal modification to the default configuration. The default configuration of the router 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 router'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 router. 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 7, Interfaces for a detailed description of the commands for configuring and monitoring link up/down traps.

Router 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 router returns an SNMP error message.

The router'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).

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. Care should be taken when creating an SNMP community with write access to select a secure community name and to ensure that this name is known only to authorised personnel.

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 router's SNMP agent does not allow the construction of MIB views. The router supports all relevant objects from all MIBs that it implements.

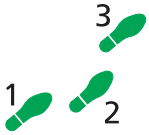
Note The router's standard SET and SHOW commands can also be used to access objects in the MIBs supported by the router.

Configuration Example (SNMPv1 and v2)

The following example illustrates the steps required to configure the router's SNMP agent. In this example, two network management stations have been set up on a large network. The central NMS (IP address 192.168.11.5) is used to both monitor devices on the network and use SNMP *set* messages to manage the 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 using SNMP *get* messages. Link traps are enabled for all interfaces on this particular router.

Note Some interface and port types mentioned in this example may not be supported on your router. The interface and port types that are available vary depending on your product's model, and whether an expansion unit (PIC, NSM) is installed. For more information, see the AR400 Series Router Hardware Reference.

Note The IP module must be enabled and correctly configured in order to access the SNMP agent in the router, since the IP module handles the UDP datagrams used to transport SNMP messages. See Chapter 14, 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. **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
   ```

Do not use the name "private" in a real 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 router. 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 router, possibly affecting the operation of the router. For this reason, care must be taken with the security of community names.

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.

```
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.

This router has static interfaces ppp0, fr3 and x25t0. Additional dynamic interfaces may be created and destroyed as the result of ISDN or ACC calls. Enable link traps for these interfaces, and set a limit of 30 traps per minute for dynamic interfaces.

5. Enable link traps.

If the router has a ppp0 static interface, additional dynamic interfaces may be created and destroyed as the result of ACC calls. Enable link traps for these interfaces, and set a limit of 30 traps per minute for dynamic interfaces.

```
ENABLE INTERFACE=ppp0 LINKTRAP
ENABLE INTERFACE=fr3 LINKTRAP
ENABLE INTERFACE=x25t0 LINKTRAP
ENABLE INTERFACE=DYNAMIC LINKTRAP
SET INTERFACE=DYNAMIC TRAPLIMIT=30
```

6. 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=ppp0
SHOW INTERFACE=fr3
SHOW INTERFACE=x25t0
SHOW INTERFACE
```

SNMPv3 on the Router

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

- entities - that may be either managers, agents, or a combination of 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 Telesyn Enterprise MIB" on page B-3 of Chapter B, SNMP MIBs.

SNMP Defined MIB Names

Table 38-8 on page 38-20 lists the MIB names that are defined within the ATR router. These names can be used in commands instead of using the MIB tree character string.

Table 38-8: 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
alliedTelesyn	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

Table 38-8: SNMP Defined Mib Names (continued)

Value	Meaning
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.

Note 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:

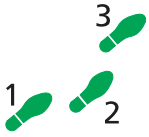
```
ADD SNMP TARGETPARAMS=params-name
    SECURITYLEVEL={NOAUTHNOPRIV|AUTHNOPRIV|AUTHPRIV}
    USER=user-name
```

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 38-26 to specify the security profile for a Target Address.

Configuration Example (SNMPv3)

The following example illustrates the steps required to configure the router's SNMP agent. In this example, two network management stations have been set up on a large network. The central NMS (IP address 192.168.11.5) is used to both monitor devices on the network and use SNMP *set* messages to manage the devices on the network. Trap messages are sent to this management station.

Note The IP module must be enabled and correctly configured in order to access the SNMP agent in the router, since the IP module handles the UDP datagrams used to transport SNMP messages. See Chapter 14, 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=alliedTelesyn 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, using commands like:

```
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 router 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 router, since the IP module handles the UDP datagrams used to transport SNMP messages. See Chapter 14, Internet Protocol (IP) for a detailed description of the commands required to enable and configure IP.

See "Conventions" on page xcv of Preface 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=ipadd]
    [V1TRAPHOST=ipadd] [V2CTRAPHOST=ipadd]
```

where:

- *name* is a string 1 to 15 characters long. It may contain any printable ASCII character and is case sensitive.
- *ipadd* is an IP address in dotted decimal notation.

Description

This command adds a trap host or a management station to a specific SNMP community.

Note 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 router.

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 router 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 router sends SNMPv1 format traps to this host. If the **v2ctraphost** parameter is used to specify a trap host, the router 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 specifies a management station for this SNMP community. This is the IP address of a device from which SNMP requests with the community name is deemed to be authentic. A community may have more than one management station.

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
```

See Also
create snmp community
destroy snmp community
destroy snmp community
disable snmp community
enable snmp community
show snmp community
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]
```

Note You must specify at least one of the optional parameters **readview**, **writeview** or **notifyview**. See the parameter descriptions below for more information.

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.

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
```

See Also

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 38-27.

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
```

See Also

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 router.

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 router, 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
```

See Also

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.

When passwords need to be entered on both the local and remote stations, these should be entered in such a way as not to compromise the system security. Note that telnet should "not" be used for this purpose, as this transmits the passwords across the network in cleartext. Refer to RFC 3414 for more information on key management. For more information on key management contact your authorised Allied Telesyn 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 router.

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 router, 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
```

See Also **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 38-5.
- *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 router, see SNMP Defined MIB Names on page 20.

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 38-20.

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
```

See Also delete snmp view
show snmp view

create snmp community

Syntax

```
CREATE SNMP COMMUNITY=name [ACCESS={READ|WRITE}]
    [TRAPHOST=ipadd] [MANAGER=ipadd]
    [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.

Description This command creates an SNMP community, optionally setting the access mode for the community and defining a trap host and manager.

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 router.

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 router, 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 router 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 router sends SNMPv1 format traps to this host. If the **v2ctraphost** parameter is used to specify a trap host, the router 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 specifies a management station for this SNMP community. This is the IP address of a device from which SNMP requests with the community name is deemed to be authentic. A community can have more than one management station, but only one can be specified when the community is created. If the parameter is not specified, the community has no defined management station.

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 on page 38-23.

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

```
create snmp community=public open=on
```

See Also
add snmp community
destroy snmp community
destroy snmp community
disable snmp
disable snmp community
enable snmp
enable snmp community
show snmp community
show snmp community

delete snmp community

Syntax

```
DELETE SNMP COMMUNITY=name [TRAPHOST=ipadd]
   [MANAGER=ipadd] [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 *name* contains spaces, it must be in double quotes.
- *ipadd* is an IP address in dotted decimal notation.

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 router.

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 router 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 specifies a single management station for this SNMP community. This is the IP address of a device from which SNMP requests received with the community name are deemed to be authentic.

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

```
delete snmp community=administration traphost=192.168.1.1
```

See Also
add snmp community
create snmp community
destroy snmp community
disable snmp community
enable snmp community
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 router.

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

```
delete snmp group=usergroup
```

See Also
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 router.

Examples To delete SNMP target address “target”, use the command:

```
deleted snmp targetaddr=target
```

See Also **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 router.

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

```
deleted snmp targetparams=params
```

See Also **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 router.

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

```
delete snmp user=noauthuser
```

See Also **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 router.

The **view** parameter specifies the SNMP view. A view with the specified name must already exist in the router.

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 specified OID must already exist in the router.

Note 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 router. If ALL is specified, then all pairs of the specified SNMP view and associated the sub-trees are deleted.

Examples

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

```
delete snmp view=mib2view mib=all
```

See Also

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 router.

See Also **add snmp community**
create snmp community
disable snmp community
enable snmp community
show snmp community

disable snmp

Syntax `DISABLE SNMP`

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

See Also **disable snmp community**
enable snmp
enable snmp community
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.

See Also **disable snmp**
enable snmp
enable snmp authenticate_trap
show snmp community

disable snmp community

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 a particular SNMP community or disables the generation of trap messages for the community.

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

The **trap** parameter specifies that only traps for the community should be disabled, not the entire operation of the community. Trap messages are not sent to the community's trap host(s), but all other SNMP operations proceed normally.

See Also **disable snmp**
enable snmp
enable snmp community
show snmp community
show snmp community

enable snmp

Syntax `ENABLE SNMP`

Description This command enables the router's SNMP agent. The SNMP agent receives and processes SNMP packets sent to the router and generate traps.

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

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

See Also **disable snmp**
disable snmp community
enable snmp community
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.

See Also **disable snmp**
disable snmp authenticate_trap
enable snmp
show snmp community

enable snmp community

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 a particular SNMP community or enables the generation of trap messages for the community.

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

The **trap** parameter specifies that only traps for the community should be enabled, not the entire operation of the community. Trap messages are sent to the community's trap host(s).

Note For security reasons, this command are accepted only if the user has Security Officer privilege.

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

```
create snmp community=private traphost=192.168.1.1
   manager=192.168.1.1
enable snmp community=private trap
```

See Also **disable snmp**
disable snmp community
enable snmp
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)

See Also disable 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.

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

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 router, 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.

Note For security reasons, this command are accepted only if the user has Security Officer privilege.

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

```
set snmp community=public open=off
```

See Also **create snmp community**
destroy snmp community
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.

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

Note 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.

- 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 Telesyn 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.

Note 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 AlliedTelesyn IANA enterprise number and a MAC address of "00-00-CD-12-34-56", use the command:

```
set snmp engineid=800000cf030000cd123456
```

Note 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.

See Also 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 router.

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 snmp group=user-group readview=user-view
```

See Also

add snmp group
delete snmp group
show snmp group

set snmp local

Syntax

```
SET SNMP LOCal={none|1..15}[VERson={V1|V2|V3|ALL}]
```

Where

- V1, V2 and V3 represent the version of the SNMP packets.

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 will be used as the source IP address for all SNMP packets of the version specified.

The **version** parameter specifies which version of SNMP packets that the local interface will apply to. 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 will select a source address on the bases of the route, i.e. the source address will be the IP address that the SNMP packet will be issued from.

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

```
set snmp 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, 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 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 router.

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 snmp targetaddr=target ip=127.0.0.1
```

See Also **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 router.

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 router. It can be added later.

Note 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 snmp targetparams=params user=test
```

See Also

add snmp targetparams
delete snmp targetparams
show snmp targetparams

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-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.

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 router.

The **group** parameter specifies the SNMP group. Although a group with the specified name need not pre-exist in the router, 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.

Note 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 snmp user=sam authprotocol=md5 authpassword=authpassword
```

See Also **add snmp user**
delete snmp user
show snmp user

show snmp

Syntax

```
SHOW SNMP
```

Description This command displays information about the router’s SNMP agent. See Figure 38-6 on page 38-45, and Table 38-9 on page 38-45.

Figure 38-6: Example output from the SHOW SNMP command

```
SNMP configuration:
  Status ........................ Enabled
  Authentication failure traps .... Enabled
  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
Unknown UserNames................ 0       UnknownPDUhandlers................0
UnknownEngineIDs ................ 0
WrongDigests .................... 0
DecryptionErrors ................ 0
```

Table 38-9: Parameters in the output of the **show snmp** command

Parameter	Meaning
Status	Whether the SNMP agent or the specified community is enabled or disabled.
Authentication failure traps	Whether the SNMP agent is enabled to generate a trap on an authentication failure for an incoming SNMP packet.

Table 38-9: Parameters in the output of the **show snmp** command (continued)

Parameter	Meaning
Local Interface	The interface used as the source in outgoing SNMP messages.
Community	The name of an SNMP community on the router.
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 router.
inBadVersions	The number of SNMP packets with a bad version field received by the router.
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.
inTotalSetVars	The total number of SNMP MIB objects which were changed.
inGetRequests	The number of SNMP Get Request packets received by the router.
inGetNexts	The number of SNMP Get Next Request packets received by the router.
inSetRequests	The number of SNMP Set Request packets received by the router.

Table 38-9: Parameters in the output of the **show snmp** command (continued)

Parameter	Meaning
inGetResponses	The number of SNMP Get Response packets received by the router.
inTraps	The number of SNMP trap message packets received by the router.
outPkts	The number of SNMP packets transmitted by the router.
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 router
outGetNexts	The number of Get Next response packets transmitted by the router.
outSetRequests	The number of Set Request packets transmitted by the router.
outGetResponses	The number of SNMP Get response packets transmitted.
outTraps	The number of SNMP Traps transmitted by the router.
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.

See Also **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 ASCII character and is case sensitive.

Description This command displays information about a single SNMP community. See Figure 38-7 on page 38-48, Table 38-10 on page 38-48.

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

Figure 38-7: 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 38-10: Parameters in the output of the **show snmp community** command

Parameter	Meaning
Name	The 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	The IP address of a management station that can access this router using this community.
Trap host	The IP address of a trap host to which traps for this community is sent.
V2c trap host	The IP address of a trap host to which SNMP version 2c traps for this community is sent.

See Also **show snmp community**

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. See Figure 38-8 on page 38-49 and Table 38-11 on page 38-49.

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

Figure 38-8: 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 38-11: Parameters in the output of the SHOW SNMP GROUP command

Parameter	Meaning
Group Name	The name of the SNMP group.
MinimumSecurity Level	The minimum security level for the SNMP group; either "noAuthNoPriv", "authNoPriv" or "authPriv".
Read View	The name of a predefined view that gives read rights to the members of the specified SNMP group.
Write View	The name of a predefined view that gives write rights to the members of the specified SNMP group.
Notification View	The name of a predefined view that gives notification rights to the members of the specified SNMP group.
Row Status	The status of the displayed group, either "active", "notInService" or "notReady".

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
```

See Also
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. See Figure 38-9 on page 38-50 and Table 38-12 on page 38-50.

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

Figure 38-9: Example output from the **show snmp group** 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 38-12: Parameters in the output of the **show snmp targetaddr** command

Parameter	Meaning
Target Address Name	The name of the SNMP target address
IP address	The IP address of SNMP target address
UDP port	The UDP port of SNMP target address
Target Address Parameter	The reference to the entry specified in the *snmpTargetParamsTable*
Row Status	The status of the displayed SNMP target address, either "active", "notInService", or "notReady"

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

```
show snmp targetaddr=adminserver
```

See Also **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. See Figure 38-10 on page 38-51, and Table 38-13 on page 38-51.

The **targetparams** parameter specifies a SNMP target params entry in the *snmpTargetParamsTable*. A SNMP target params entry with the specified name must exist in the router. If the target params-name is not specified all existing target params will be presented.

Figure 38-10: 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 38-13: Parameters in the output of the **show snmp group** command

Parameter	Meaning
Target Params Name	The name of the SNMP target parameters
Security Level	The security level for the SNMP messages to be sent to the target address; one of "noAuthNoPriv", "authNoPriv" or "authPrive"
User Name	The name of user who will receive notification on target address
Row Status	The status of the displayed SNMP target address, one of "active", "notInService" or "notReady"

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

```
show snmp targetparams=adminserverparams
```

See Also **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. See Figure 38-11 on page 38-52, and Table 38-11 on page 38-49.

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

Figure 38-11: 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 38-14: Parameters in the output of the **show snmp user** command

Parameter	Meaning
User Name	The name of the SNMP user
Group Name	The name of the SNMP group
Auth Protocol	The authentication protocol for the SNMP user; either "MD5","SHA", or "None"
Priv Protocol	The privacy protocol for the SNMP user; either "DES" or "None"
Row Status	The status of the displayed SNMP user, either "active", "notInService", or "notReady"

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
```

See Also **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. See Figure 38-12 on page 38-53, and Figure 38-13 on page 38-53.

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

Figure 38-12: Example output from the **show snmp view** command

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

Figure 38-13: 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 38-15: Parameters in the output of the **show snmp view** command

Parameter	Meaning
View name	The name of the SNMP view
OID	The object identifier of the sub-tree to be included or excluded from the view
MIB	The predefined MIB name of the sub-tree
Type	The type for this sub-tree, either “include” or “exclude”
Row Status	The status of the displayed SNMP view, either “active”, “notInService”, or “notReady”

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
```

See Also **show snmp view**
delete snmp view
add snmp view