Chapter 28

Simple Network Management Protocol (SNMP)

Introduction

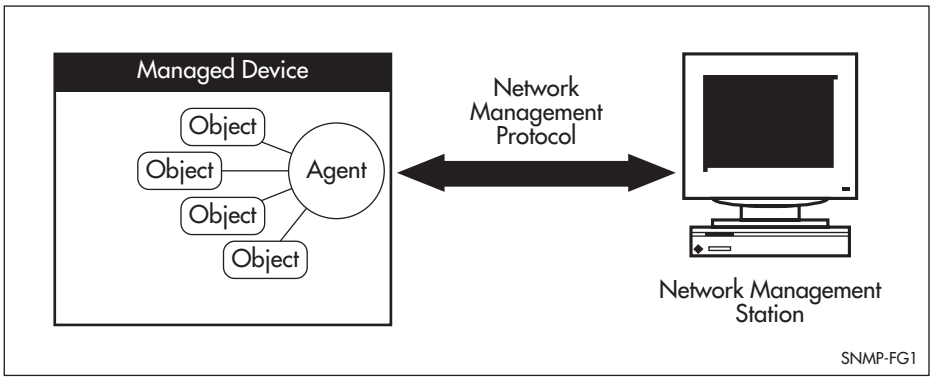
The Simple Network Management Protocol (SNMP) is the network management protocol of choice for the Internet and IP-based internetworks. This chapter describes the main features of SNMP, support for SNMP on the router, and how to configure the router's SNMP agent. See *Appendix C, SNMP MIBs* for a detailed description of all MIBs (Management Information Bases) and MIB objects supported by the router.

Network Management Framework

A network management system has three components (Figure 28-1 on page 28-2):

- One or more *managed devices*, each containing an agent which 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, 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 28-1: Components of a network management system.



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

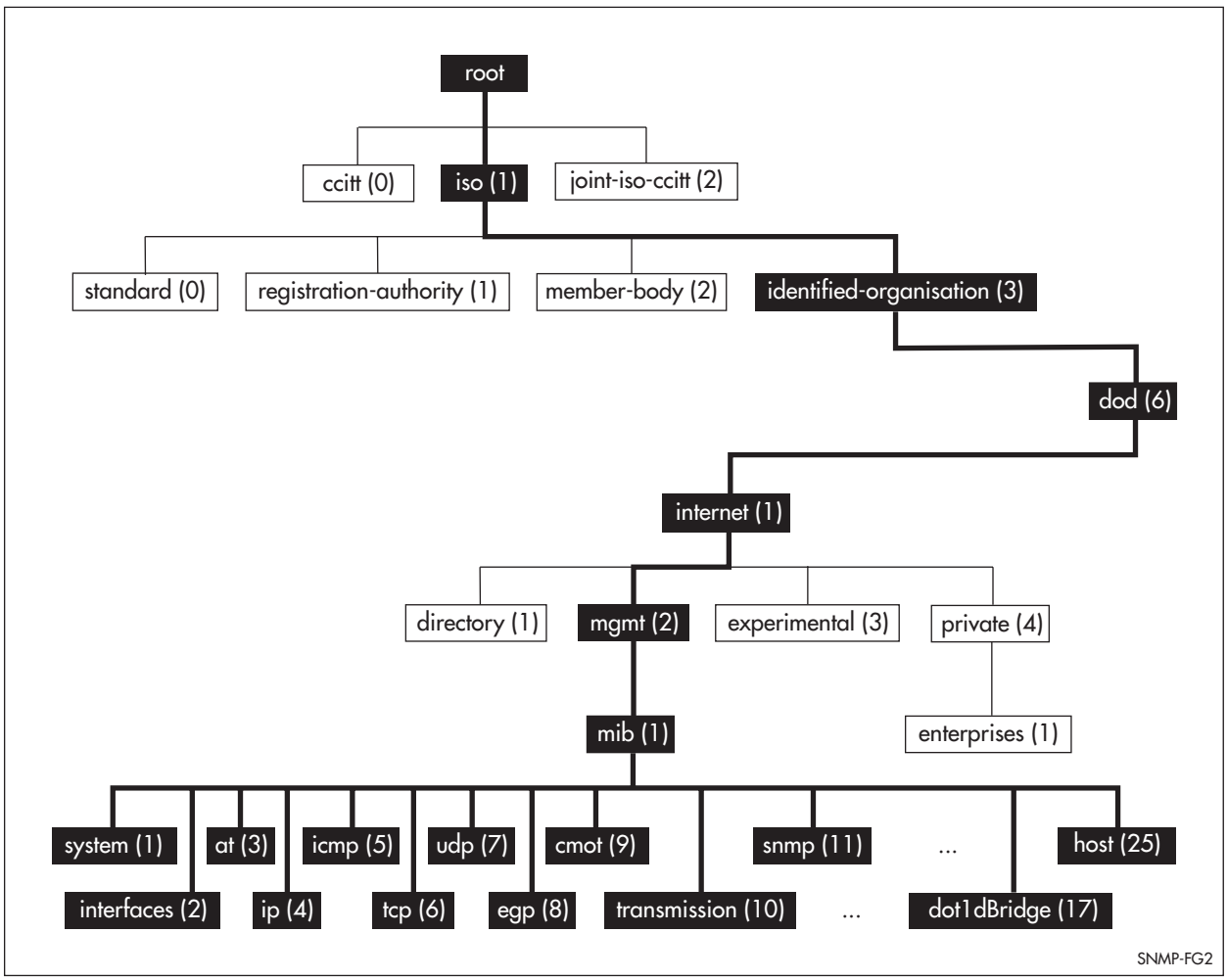
- RFC 1155, "*Structure and identification of management information for TCP/IP-based internets*" (referred to as the SMI), details the mechanisms used to describe and name the objects to be managed.
- RFC 1213, "*Management Information Base for network management of TCP/IP-based internets: MIB-II*" (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.

Structure of Management Information

The 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. Each node in the tree has a label 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 28-2 on page 28-3.

Figure 28-2: 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.

Managed objects are the leaf nodes in the tree. Each managed object is defined by its name, syntax, access mode, status and description.

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 three 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:

  1.3.6.1.2.1

  identifies the MIB-II sub-tree, and:

  1.3.6.1.2.1.1.1

  identifies the *sysDescr* object in the system group of MIB-II.

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

  { iso org dod 1 }

  identifies the *internet* sub-tree. The name may be abbreviated to a relative form:

  { internet 1 }

  identifies the first (*directory*) node of the *internet* sub-tree.

- **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:

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

  identifies the first (*directory*) node in the *internet* sub-tree. The name may be abbreviated to:

  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 28-1 on page 28-5).

Table 28-1: 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 can not be read or set.

Status

The status of an object describes the implementation requirements for the object (Table 28-2 on page 28-6).

Table 28-2: 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.

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.

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 8, Internet Protocol (IP) for detailed descriptions of the commands required to enable and configure IP.

SNMP Messages

The SNMP protocol is termed *simple* because it has only five operations, or messages—*get*, *get-next*, *get-response*, *set*, and *trap* (Table 28-4 on page 28-7). 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. Care must be taken in the choice and safe-guarding of community names, which are effectively passwords for SNMP. See *Appendix C, SNMP MIBs* for a description of the router's implementation of each MIB object with read-write access.

Figure 28-3 on page 28-7 shows the format of an SNMP message. The function of the fields are described in Table 28-3 on page 28-7. There are five different SNMP PDUs (Table 28-4 on page 28-7) and seven generic traps (Table 28-5 on page 28-7).

Figure 28-3: Format of an SNMP message.

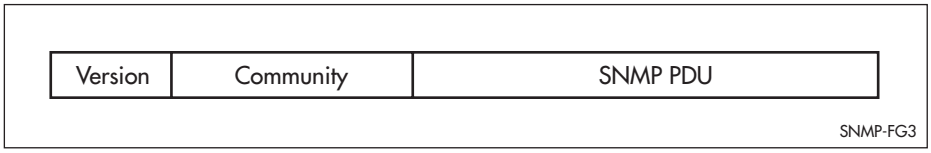


Table 28-3: Fields in an SNMP message.

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

Table 28-4: 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.
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 or set-request PDU.
trap	Sent by an agent to an NMS, to notify the NMS of a extraordinary event.

Table 28-5: Generic SNMP traps.

Value	Meaning
coldStart (0)	The agent is re-initialising itself. Objects may be altered.
warmStart (1)	The agent is re-initialising itself. Objects will not be altered.
linkDown (2)	An interface has changed state from “Up” to “Down”.
linkUp (3)	An interface has changed state from “Down” to “Up”.
authenticationFailure (4)	An SNMP message has been received with an invalid community name.
egpNeighborLoss (5)	An EGP peer has transitioned to state “Down”.
enterpriseSpecific (6)	Some other enterprise-specific trap.

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* and/or *get-next-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.

Communities and Views

A *community* is a relationship between an NMS and an agent. The community name is used like a password for a trivial authentication scheme.

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 28-6 on page 28-8).

Table 28-6: 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*

A pairing of an SNMP community and 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 an authentic SNMP message 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.

Support for SNMP

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

The router's SNMP agent can be enabled or disabled using the commands:

```
ENABLE SNMP
DISABLE SNMP
```

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

```
SHOW SNMP
```

An SNMP *community* is a pairing of an SNMP agent with a set of SNMP application entities.

SNMP communities are the main configuration item in the router's implementation of SNMP, 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 using the command:

```
CREATE SNMP COMMUNITY=name [ACCESS={READ|WRITE}]
    [TRAPHOST=ipadd] [MANAGER=ipadd]
    [OPEN={ON|OFF|YES|NO|TRUE|FALSE}]
```

which defines the name of the community (e.g. "public"), and specifies the IP address of a trap host and/or a management station. A community can be modified 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.

An SNMP community is destroyed using the command:

```
DESTROY SNMP COMMUNITY=name
```

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

```
ADD SNMP COMMUNITY=name [TRAPHOST=ipadd] [MANAGER=ipadd]
DELETE SNMP COMMUNITY=name [TRAPHOST=ipadd] [MANAGER=ipadd]
```

When a trap is generated by the SNMP agent it is forwarded to all the trap hosts assigned to the community. 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 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 will generate authentication failure traps for messages directed to the disabled community. Information about the configuration of SNMP communities can be displayed using the command:

```
SHOW SNMP COMMUNITY=name
```

The SNMP agent does not support a default community called "public" with read-only access, traps disabled and open access as mandated in RFC 1812, as this is a security hole open for users who wish to use the 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* 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 my be enabled or disabled 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, 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 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, using the command:

```
SET INTERFACE={ifIndex|interface|DYNAMIC} TRAPLIMIT=1..60
```

See *Chapter 2, 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 will return 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.

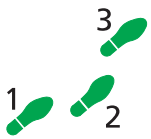
An SNMP MIB *view* is a subset of objects in the MIB that pertain to a particular network element. For example, the MIB view of a hub would be the objects relevant to management of the hub, and would not include IP routing table objects, for example. The router's SNMP agent does not allow the construction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

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

Configuration Example

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) will be used to both monitor devices on the network and use SNMP *set* messages to manage the devices on the network. Trap messages will be sent to this management station. The regional network management station (IP addresses 192.168.16.1) will be used just to monitor devices on the network using SNMP *get* messages. Link traps will be enabled for all interfaces on this particular router.

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 8, 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 will be sent to this NMS.

   ```
   CREATE SNMP COMMUNITY=private ACCESS=WRITE
      TRAPHOST=192.168.11.5 MANAGER=192.168.11.5 OPEN=NO
   ```

Do not use the name "private" in a real network—its too obvious! Use a different name! 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
```

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.

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

5. Check the configuration.

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

```
SHOW SNMP
SHOW SNMP COMMUNITY
```

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

```
SHOW INTERFACE=ppp0
SHOW INTERFACE=fr3
SHOW INTERFACE=x25t0
SHOW INTERFACE
```

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 8, Internet Protocol (IP)* for a detailed description of the commands required to enable and configure IP.

See "*Conventions*" on page lxvii 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`*`]`

where:

- *name* is a character string, 1 to 15 characters in length. Valid characters are uppercase letter (A–Z), lowercase letters (a–z) and digits (0–9). *name* is case-sensitive, that is "Public" is a different name from "public".
- *ipadd* is an IP address in dotted decimal notation.

Description This command adds a trap host or a management station to the specified SNMP community.

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

The TRAPHOST parameter specifies a trap host for the SNMP community. This is the IP address of a device to which traps generated by the router will be sent. A community may have more than one trap host.

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 will be 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
```

See Also CREATE SNMP COMMUNITY
DELETE SNMP COMMUNITY
DESTROY SNMP COMMUNITY
DISABLE SNMP COMMUNITY
ENABLE SNMP COMMUNITY
SET SNMP COMMUNITY
SHOW SNMP COMMUNITY

CREATE SNMP COMMUNITY

Syntax

```
CREATE SNMP COMMUNITY=name [ACCESS={READ|WRITE}]
    [TRAPHOST=ipadd] [MANAGER=ipadd]
    [OPEN={ON|OFF|YES|NO|TRUE|FALSE}]
```

where:

- *name* is a character string, 1 to 15 characters in length. Valid characters are uppercase letter (A–Z), lowercase letters (a–z) and digits (0–9). *name* is case-sensitive, that is "Public" is a different name from "public".
- *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 parameter specifies a trap host for the SNMP community. This is the IP address of a device to which traps generated by the router will 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 will have no defined trap host.

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 will be 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 will have 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.

For security reasons this command will only be accepted if the user has SECURITY OFFICER privilege.

Examples

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

```
SET SNMP COMMUNITY=public OPEN=ON
```

See Also ADD SNMP COMMUNITY
DELETE SNMP COMMUNITY
DESTROY SNMP COMMUNITY
DISABLE SNMP
DISABLE SNMP COMMUNITY
ENABLE SNMP
ENABLE SNMP COMMUNITY
SET SNMP COMMUNITY
SHOW SNMP COMMUNITY

DELETE SNMP COMMUNITY

Syntax

```
DELETE SNMP COMMUNITY=name [TRAPHOST=ipadd]
   [MANAGER=ipadd]
```

where:

- *name* is a character string, 1 to 15 characters in length. Valid characters are uppercase letter (A–Z), lowercase letters (a–z) and digits (0–9). *name* is case-sensitive, that is "Public" is a different name from "public".
- *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 parameter specifies a trap host for the SNMP community. This is the IP address of a device to which traps generated by the router are currently sent.

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
SET SNMP COMMUNITY
SHOW SNMP COMMUNITY

DESTROY SNMP COMMUNITY

Syntax `DESTROY SNMP COMMUNITY=`*`name`*

where:

- *name* is a character string, 1 to 15 characters in length. Valid characters are uppercase letter (A–Z), lowercase letters (a–z) and digits (0–9). *name* is case-sensitive, that is "Public" is a different name from "public".

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
SET SNMP COMMUNITY
SHOW SNMP COMMUNITY

DISABLE SNMP

Syntax `DISABLE SNMP`

Description This command disables the router's SNMP agent. SNMP packets sent to the router will be treated as unknown protocol packets by the underlying transport layer (UDP) and traps will not be generated by the router.

See Also DISABLE SNMP COMMUNITY
ENABLE SNMP
ENABLE SNMP COMMUNITY
SHOW SNMP
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

DISABLE SNMP COMMUNITY

Syntax `DISABLE SNMP COMMUNITY=`*`name`* `[TRAP]`

where:

- *name* is a character string, 1 to 15 characters in length. Valid characters are uppercase letter (A–Z), lowercase letters (a–z) and digits (0–9). *name* is case-sensitive, that is "Public" is a different name from "public".

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 will not be generated for the community. The SNMP agent will generate 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 will not be sent to the community's trap host(s), but all other SNMP operations will proceed as normal.

See Also DISABLE SNMP
ENABLE SNMP
ENABLE SNMP COMMUNITY
SHOW SNMP
SHOW SNMP COMMUNITY

ENABLE SNMP

Syntax `ENABLE SNMP`

Description This command enables the router's SNMP agent. The SNMP agent will receive and process 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
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

ENABLE SNMP COMMUNITY

Syntax `ENABLE SNMP COMMUNITY=`*`name`* `[TRAP]`

where:

- *name* is a character string, 1 to 15 characters in length. Valid characters are uppercase letter (A–Z), lowercase letters (a–z) and digits (0–9). *name* is case-sensitive, that is "Public" is a different name from "public".

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 will be sent to the community's trap host(s).

For security reasons this command will only be accepted 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
SHOW SNMP COMMUNITY

SET SNMP COMMUNITY

Syntax

```
SET SNMP COMMUNITY=name [ACCESS={READ|WRITE}]
   [OPEN={ON|OFF|YES|NO|TRUE|FALSE}]
```

where:

- *name* is a character string, 1 to 15 characters in length. Valid characters are uppercase letter (A–Z), lowercase letters (a–z) and digits (0–9). *name* is case-sensitive, that is "Public" is a different name from "public".

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.

For security reasons this command will only be accepted 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

SHOW SNMP

Syntax SHOW SNMP

Description This command displays information about the router's SNMP agent (Figure 28-4 on page 28-20, Table 28-7 on page 28-20).

Figure 28-4: Example output from the SHOW SNMP command.

```
SNMP configuration:
  Status ..................... Enabled
  Authentication failure traps .... Enabled
  Community ..................... public
    Access ...................... read-only
    Status ...................... Enabled
    Traps ....................... Enabled
    Open access ................. Yes
  Community ..................... Administration
    Access ...................... read-write
    Status ...................... Disabled
    Traps ....................... Disabled
    Open access ................. No

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
```

Table 28-7: Parameters displayed in the output of the SHOW SNMP command.

Parameter	Meaning
Status	The status of the SNMP agent or the specified community; one of "Enabled" or "Disabled".
Authentication failure traps	Whether or not the SNMP agent will generate a trap on an authentication failure for an incoming SNMP packet; one of "Enabled" or "Disabled".
Community	The name of an SNMP community on the router.
Access	The access rights for the SNMP community; one of "read-only" or "read-write".
Status	The status of the community; one of "Enabled" or "Disabled".

Table 28-7: Parameters displayed in the output of the SHOW SNMP command. (Continued)

Parameter	Meaning
Traps	Whether or not the community will generate traps; one of "Enabled" or "Disabled".
Open access	Whether or not the SNMP community is open to access from all IP addresses; one of "Yes" or "No".
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.
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.

Table 28-7: Parameters displayed in the output of the SHOW SNMP command. (Continued)

Parameter	Meaning
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.

See Also SHOW SNMP COMMUNITY

SHOW SNMP COMMUNITY

Syntax `SHOW SNMP COMMUNITY=`*`name`*

where:

- *name* is a character string, 1 to 15 characters in length. Valid characters are uppercase letter (A–Z), lowercase letters (a–z) and digits (0–9). *name* is case-sensitive, that is "Public" is a different name from "public".

Description This command displays information about a single SNMP community (Figure 28-5 on page 28-22, Table 28-8 on page 28-23).

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

Figure 28-5: 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
```

Table 28-8: Parameters displayed 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	The access rights for the SNMP community; one of "read-only" or "read-write".
Status	The status of the community; one of "Enabled" or "Disabled".
Traps	Whether or not the community generates trap messages; one of "Enabled" or "Disabled".
Open access	Whether or not the community is open to access from all IP addresses; one of "Yes" or "No".
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 will be sent.

See Also SHOW SNMP