Technical Guide

# Simple Network Management Protocol (SNMP)

FEATURE OVERVIEW AND CONFIGURATION GUIDE

## Introduction

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

This guide describes the main features of SNMP Version 1(SNMPv1), SNMP Version 2c (SNMPv2c) and Version 3 (SNMPv3). It also describes support for SNMP on the switch, and how to configure the switch's SNMP agent.

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

## Products and software version that apply to this guide

This guide applies to all AlliedWare Plus™ products, running version **5.4.4** or later.

Feature support may change in later software versions. For the latest information, see the following documents:

- The product's Datasheet
- The AlliedWare Plus Datasheet
- The product's Command Reference

These documents are available from the above links on our website at alliedtelesis.com.

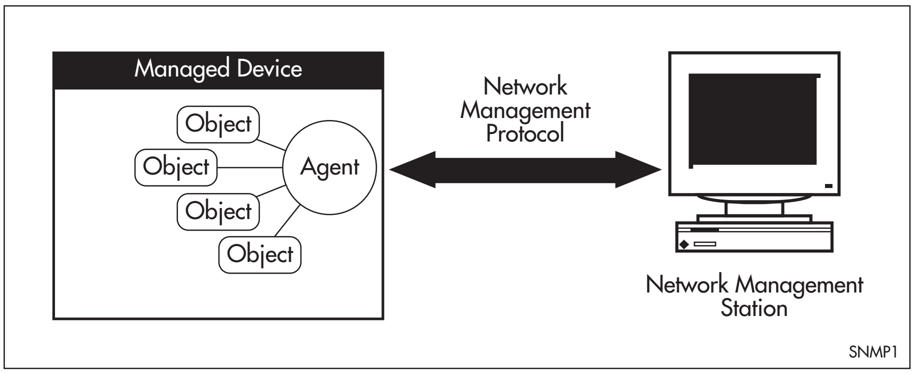
## Content

# Network Management Framework

A network management system has the following components:

- 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 1: Components of a network management system**



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

- RFC 1155, *Structure and identification of management information for TCP/IP based internets* (referred to as the SMI), details the mechanisms used to describe and name the objects to be managed.
- RFC 1213, *Management Information Base for network management of TCP/ IP-based internets: MIB-II* (referred to as MIB-II), defines the core set of managed objects for the Internet suite of protocols. The set of managed objects can be extended by adding other MIBs specific to particular protocols, interfaces or network devices.
- RFC 1157, *A Simple Network Management Protocol (SNMP)*, is the protocol used for communication between management stations and managed devices.

Subsequent documents that have defined SNMPv2c are:

- RFC 1901, *Introduction to Community-based SNMPv2*
- RFC 1902, *Structure of Management Information for Version 2 of the Simple Network Management Protocol (SNMPv2)*
- RFC 1903, *Textual Conventions for Version 2 of the Simple Network Management Protocol (SNMPv2)*
- RFC 1904, *Conformance Statements for Version 2 of the Simple Network Management Protocol*

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

Subsequent documents that have defined SNMPv3 are:

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

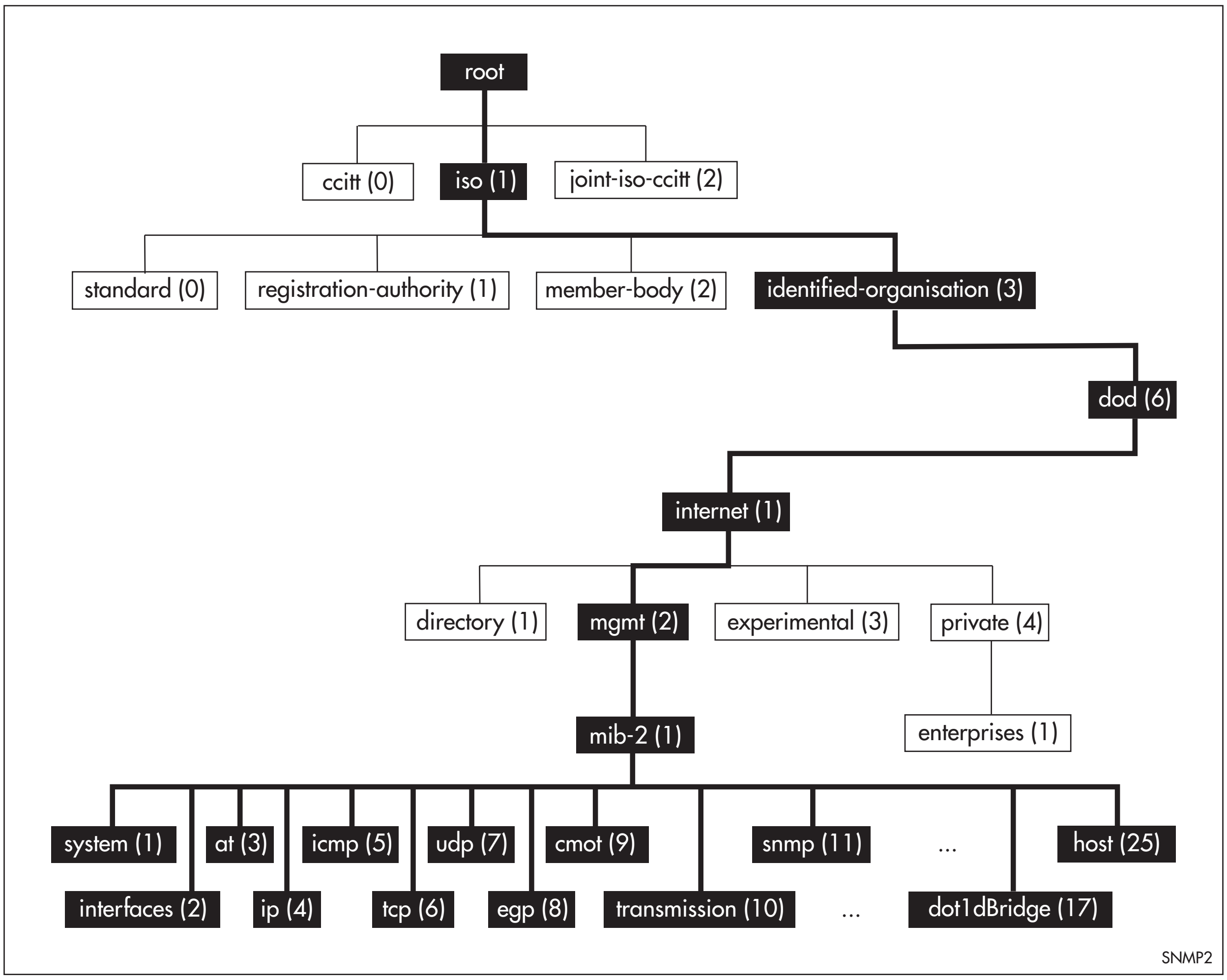
## Structure of Management Information

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

The MIB is organized 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.

Each managed object is represented by a leaf node of the tree 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 2 the object named interfaces would be uniquely identified by the string of individual sub-identifiers, 1.3.6.1.2.1.2.

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



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

## Names

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

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

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

  ```
  1.3.6.1.2.1
  ```

  The following identifies the sysDescr 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. The 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.

## 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 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 2: Status values for MIB objects**

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

## Description

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

# The SNMP Protocol

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

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

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

## SNMP versions

The switch supports SNMP version 1 (SNMPv1), SNMP version 2c (SNMPv2c) and SNMP Version 3 (SNMPv3). The three versions operate similarly.

SNMPv2c updated the original protocol, and offered the following main enhancements:

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

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

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

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

## SNMP messages

The SNMP protocol is termed 'simple' because it has only six operations, or messages—get, get-next, get-response, set, and trap, and SNMPv2c also has the get-bulk-request message. 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.

## Polling versus event notification

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

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

Table 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 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. SNMP PDU Version Community |
| 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 synchronization. |

Table 5: Generic SNMP traps

| VALUE | MEANING |
|---|---|
| coldStart | The agent is re-initializing itself. Objects may be altered. |
| warmStart | The agent is re-initializing 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. |

# SNMPv1 and SNMPv2c

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

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.

## SNMP MIB views for SNMPv1 and SNMPv2c

An SNMP MIB view is an 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.

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

## SNMP communities

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

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

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

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.

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.

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

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

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

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

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

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

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

## Defining management stations within communities

You can add management stations to a community either individually, by entering just its IP address, or you can enter a range of management stations by entering an IP address that ends with a '/' character followed by a number between 1 and 32.

The number that follows the '/' character operates as an address mask to define a range of addresses for the management stations. The following example shows how to allocate a band of three binary addresses to a portion of the subnet 146.15.1.X

Example

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

**Step 1: Assign management stations.**

Decide on the number of management stations that you want to assign to a particular subnet, then decide how many binary digits are required to define this number of addresses. In this case we need up to 8 management stations, so we will assign 3 binary digits (3 binary digits can provide 8 different values). To assign the last 3 binary digits for management stations, we assign a prefix that is a count of all binary digits in the address minus those to be assigned as management stations. In this case the prefix is 29; this being the number of binary digits in an IP address (32) minus the number of digits assigned to the management stations (3).

**Step 2: Allocate address block to 'admin' community.**

The method used in this step depends on whether or not the community already exists.

- If the community called "admin" does not exist, create a new community called "admin" and allocate a three binary digit block of addresses to the address subnet 146.15.1.X.
- If the community called "admin" already exists, allocate a three binary digit block of addresses to an existing community called "admin" with the address subnet 146.15.1.X.

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

## Configuration example (SNMPv1 and v2)

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

IP and VLANs must be correctly configured in order to access the SNMP agent in the switch. This is because the IP module handles both the TCP transport functions, and the UDP functions that enable datagrams to transport SNMP messages.

To configure SNMP

**Step 1: Enable the SNMP agent.**

Enable the SNMP agent and enable the generation of authenticate failure traps to monitor unauthorized SNMP access. SNMP is enabled by default in AlliedWare Plus.

```
awplus(config)# snmp-server enable trap auth
```

### Step 2: Create a community with write access for the central NMS.

Create a write access community called "example1rw" for use by the central network management station at 192.168.11.5 Use an ACL to give the central NMS SNMP access to the switch using that community name.

```
awplus(config)# access-list 66 permit 192.168.11.5
awplus(config)# snmp-server community example1rw rw 66
```

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

SNMP V1 or V2c provide very minimal security. If security is a concern, you should use SNMPv3.

### Step 3: Create a community with read-only access for the regional NMS.

Create a read-only access community called "example2ro" for use by the regional network management station at 192.168.16.1. Use an ACL to give the regional NMS SNMP access to the switch using that community name.

```
awplus(config)# access-list 67 permit 192.168.16.1
awplus(config)# snmp-server community example2ro ro 67
```

### Step 4: Enable link traps.

Enable link traps for the desired interfaces. In this example, the NSMs are in VLAN 2 and VLAN 3 and other ports are in VLAN 1 for simplicity.

```
awplus(config)# interface vlan1-3
awplus(config-if)# snmp trap link-status
```

Note that link traps on VLANs are sent when the last port in the VLAN goes down. You will only see a trap for a VLAN if the trap host is in a different VLAN.

You can also enable link traps on channel groups and switch ports. For example, to enable traps on a range of switch ports:

```
awplus(config)# int port1.0.4-1.0.6
awplus(config)# int port1.1.5-1.1.7
awplus(config-if)# snmp trap link-status
```

You can also enable link traps on channel groups and switch ports. For example, to enable traps on a range of switch ports:

### Step 5: Configure trap hosts.

Specify the IP address or addresses that the traps will get sent to. In this example, traps will be sent to both NMSes

```
awplus(config)# snmp-server host 192.168.11.5 version 2c example1rw
awplus(config)# snmp-server host 192.168.16.1 version 2c example2ro
```

### Step 6: Check the configuration.

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

```
awplus# show snmp-server
awplus# show snmp-server community
awplus# show run snmp
```

This is the output of the **show snmp-server community** command for this example:

```
SNMP community information:
  Community Name ........... example1rw
    Access ................. Read-write
    View ................... none
  Community Name ........... example2ro
    Access ................. Read-only
    View ................... none
```

This is the output of the **show run snmp** command for this example:

```
no snmp-server ip
snmp-server enable trap auth
snmp-server community example1rw rw 66
snmp-server community example2ro 67
snmp-server host 192.168.1.2 version 2c example1rw
snmp-server host 192.168.2.2 version 2c example2ro
!
```

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

```
awplus# show interface vlan1-3
```

This is the output of the **show interface** command for this example:

```
Interface vlan1
  Scope: both
  Link is UP, administrative state is UP
  Hardware is VLAN, address is 0009.41fd.c029
  index 201 metric 1 mtu 1500
  arp ageing timeout 300
  <UP,BROADCAST,RUNNING,MULTICAST>
  VRF Binding: Not bound
  SNMP link-status traps: Sending (suppressed after 20 traps in 60 sec)
  Bandwidth 1g
    input packets 4061, bytes 277043, dropped 0, multicast   packets 3690
    output packets 190, bytes 18123, multicast packets 0 broadcast packets 0
Interface vlan2
  Scope: both
  Link is DOWN, administrative state is UP
  Hardware is VLAN, address is 0009.41fd.c029
  IPv4 address 192.168.11.50/24 broadcast 192.168.11.255
  index 202 metric 1 mtu 1500
  arp ageing timeout 300
  <UP,BROADCAST,MULTICAST>
  VRF Binding: Not bound
  SNMP link-status traps: Sending (suppressed after 20 traps in 60 sec)
  Bandwidth 1g
    input packets 568, bytes 42309, dropped 0, multicast packets 0
    output packets 183, bytes 18078, multicast packets 0 broadcast packets 0
Interface vlan3
  Scope: both
  Link is DOWN, administrative state is UP
  Hardware is VLAN, address is 0009.41fd.c029
  IPv4 address 192.168.16.50/24 broadcast 192.168.16.255
  index 203 metric 1 mtu 1500
  arp ageing timeout 300
  <UP,BROADCAST,MULTICAST>
  VRF Binding: Not bound
  SNMP link-status traps: Sending (suppressed after 20 traps in 60 sec)
    input packets 0, bytes 0, dropped 0, multicast packets 0
    output packets 0, bytes 0, multicast packets 0 broadcast packets 0
```

## SNMPv3

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

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

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

## 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.

### SNMP engine

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

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

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

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

### Entity applications

The following applications are defined within the agent applications:

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

## 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.

## SNMP groups

Groups were introduced as part of SNMPv3. They are the means by which users are assigned their views and access control policy. Once a group has been created, users can be added to them. In practice a number of groups would be created, each with varying views and access security requirements. Users would then be added to their most appropriate groups. Each Group name and Security Level pair must be unique within a switch.

## SNMP users

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

Note: SNMP **Target Addresses** and **Target Params** in SNMPv3 are not currently supported in Alliedware Plus software

## SNMPv3 message protocol format

Table 6: SNMPv3 PDUs

| VALUE | MEANING |
|---|---|
| msgVersion | Identifies the message format to be SNMPv3. |
| msgID | An identifier used between SNMP entities to coordinate message requests and responses. Note that a message response takes the msgID value of the initiating message. |
| msgMaxSize | Conveys the maximum message size (in octets) an integer between 484 and $2^{31}$-1, supported by the sender of the message. Specified as msgFlags. A single octet whose last three bits indicate the operational mode for privacy, authentication, and report. |
| msgSecurityModel | An identifier used to indicate the security mode (i.e. SNMPv1, SNMPv2c or SNMPv3 to be used when processing the message. Note that although only the SNMPv3 identifier is accepted by the switch, these earlier version message formats are detected by the msgVersion field and processed appropriately. |
| msgAuthoritativeEngineID | The ID of the authoritative engine that relates to a particular message, i.e. the source engine ID for Traps, Responses and Reports, and the destination engine for Gets, GetNexts, Sets, and Informs. |
| msgAuthoritativeEngineBoots | A value that represents the number of times the authoritative engine has rebooted since its installation. Its value has the range 1 to $2^{31}$-1. |
| msgAuthoritativeEngineTime | The number of seconds since the authoritative engine snmpEngineBoots counter was last incremented. |

Table 6: SNMPv3 PDUs (Continued)

| VALUE | MEANING |
|---|---|
| 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 Initialization 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. |

## Configuration example (SNMPv3)

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

The IP module must be enabled and correctly configured in order to access the SNMP agent in the switch, since the IP module handles the UDP datagrams used to transport SNMP messages.

To configure SNMP

### Step 1: Enable the SNMP agent.

Enable the SNMP agent and enable the generation of authenticate failure traps to monitor unauthorized SNMP access. SNMP is enabled by default in AlliedWare Plus.

### Step 2: Add SNMP views.

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

```
awplus(config)# snmp-server view atmib 1.3.6.1.2.14 included
awplus(config)# snmp-server view atmib alliedtelesis included
```

### Step 3: Add SNMP group.

```
awplus(config)# snmp-server group ord-user noauth read atmib
awplus(config)# snmp-server group admin-user auth read atmib write
atmib notify atmi
```

### Step 4: Add SNMP users.

Add users to the groups by using commands such as:

```
awplus(config)# snmp-server user ken admin-user auth md5 mercury
```

# Using SNMP to Manage Files and Software

The Allied Telesis Enterprise MIB includes objects for managing files and software on the switch. This section includes procedures for using MIB objects on the switch to perform some common tasks, via an SNMP management application:

- "Copy a file to or from a TFTP server" on page 21
- "Upgrade software and configuration files" on page 22

## Copy a file to or from a TFTP server

Use this procedure to copy a file (for example, a software version file) to the switch from a TFTP server, or to copy a file (for example, a configuration file) from the switch to a TFTP server. The MIB objects in this procedure reside in the module atFilev2 { modules 600 }, with object ID 1.3.6.1.4.1.207.8.4.4.4.600**.** Other MIB objects can be used in a similar way for moving and deleting files on the switch.

Table 7: Procedure for copying a file to or from a device using a TFTP server

| | DO THIS ... | BY SETTING OR READING THIS MIB OBJECT ... | WHOSE OBJECT ID IS ... | TO THIS VALUE ... |
|---|---|---|---|---|
| 1. | If the source device is part of a stack, set the stack ID.<br>For a standalone switch, keep the default value, 1. | atFilev2SourceStackId | { atFilev2Operation 1 } | <stack-id> |
| 2. | If the destination device is part of a stack, set the stack ID. | atFilev2DestinationStackId | { atFilev2Operation 4 } | <stack-id> |
| 3. | Set the source device. | atFilev2SourceDevice | { atFilev2Operation 2 } | 4 (TFTP) or<br>1 (Flash) |
| 4. | Set the destination device. | atFilev2DestinationDevice | { atFilev2Operation 5 } | 4 (TFTP) or<br>1 (Flash) |
| 5. | Set the source filename. Include the path (if any) but not the device. | atFilev2SourceFileName | { atFilev2Operation 3 } | <*source-filename*><br>e.g. /awp/config/admin.cfg |
| 6. | Set the destination filename. Include the path (if any) but not the device. | atFilev2DestinationFileName | { atFilev2Operation 6 } | <*dest-filename*><br>e.g. /config/admin.cfg |
| 7. | Set the IP address of the TFTP server. | atFilev2TftpIPAddr | { atFilev2Tftp_4 1 } | <ip-addr> |
| 8. | Check that no other transfer is in progress, and that the required parameters have been set. | atFilev2CopyBegin | { atFilev2Operation 7 } | Read: idle |
| 9. | Start the file transfer. | atFilev2CopyBegin | { atFilev2Operation 7 } | Set: 1 |

Table 7: Procedure for copying a file to or from a device using a TFTP server (Continued)

| | DO THIS ... | BY SETTING OR READING THIS MIB OBJECT ... | WHOSE OBJECT ID IS ... | TO THIS VALUE ... |
|---|---|---|---|---|
| 10. | Monitor file transfer progress. | atFilev2CopyBegin | { atFilev2Operation 7 } | Read:<br>In progress:<br>copying <*src*> --> <*dst*><br>or<br>Success:<br>copy <*src*> --> <*dst*> success<br>or<br>Failure:<br>copy <*src*> --> <*dst*> failure: <*err-msg*> |

## Upgrade software and configuration files

Use this procedure to upgrade to a new software version and boot configuration file. For detailed descriptions of the MIB objects used in this procedure, and other MIB objects for managing software installation and configuration files, see the products command reference.

Table 8: Procedure for upgrading to a new software version and boot configuration

| | DO THIS ... | BY READING OR SETTING THIS MIB OBJECT ... | WHOSE OBJECT ID IS ... | TO THIS VALUE ... |
|---|---|---|---|---|
| 1. | Check that you have enough flash memory for the currently running software file, the new software version file, and any configuration scripts required. | | | |
| 2. | Check the version and name of the software currently running. | currSoftVersion<br>currSoftName | 1.3.6.1.4.1.207.8.4.4.4.500.2.1.1<br>1.3.6.1.4.1.207.8.4.4.4.500.2.1.2 | Read:<br><*software-name*><br><*software-version*> |
| 3. | If you do not already have the currently running software as a software version file in flash, save the currently running software with a file name to the flash root. | currSoftSaveToFile | 1.3.6.1.4.1.207.8.4.4.4.500.2.1.4 | Set:<br><*backup*-filename.rel> |

Table 8: Procedure for upgrading to a new software version and boot configuration (Continued)

| | DO THIS ... | BY READING OR SETTING THIS MIB OBJECT ... | WHOSE OBJECT ID IS ... | TO THIS VALUE ... |
|---|---|---|---|---|
| 4. | Check that the file saved successfully. (The most common failures result from lack of flash memory space.) | currSoftSaveStatus | 1.3.6.1.4.1.207.8.4.4.4.500.2.1.5 | Read:<br>■ 1 (idle) - there is no release file save operation in progress<br>■ 2 (success) - the last release file save operation completed successfully<br>■ 3 (failure) - the last release file save operation failed<br>■ 4 (saving) - a release file save operation is currently in progress |
| 5. | Copy the new software version file to flash memory on the device | See Table 7. | | |
| 6. | Set the new release file to be the current release that the device will install and run the next time it restarts. Include the path. | nextBootPath | 1.3.6.1.4.1.207.8.4.4.4.500.2.2.2 | Set:<br>*<next-filename>*<br><br>e.g.:<br>**flash:/release.rel** |
| 7. | Check the version of release file set to install next. | nextBootVersion | 1.3.6.1.4.1.207.8.4.4.4.500.2.2.1 | Read:<br>*<software-version>* |
| 8. | Set the previous release file to be the backup release that the device will install and run if the device fails to boot successfully with the new release file.<br>Include the path. | bckpPath | 1.3.6.1.4.1.207.8.4.4.4.500.2.3.2 | Set:<br>*<backup-filename>*<br><br>e.g.:<br>**flash:/release.rel** |
| 9. | Check the version of backup release file. | bckpVersion | 1.3.6.1.4.1.207.8.4.4.4.500.2.3.1 | Read:<br>*<software-version>* |
| 10. | If necessary, copy a configuration file to the device (Table 7), or save the current running configuration to a file in the root directory of flash. To save the running configuration, specify the filename, but not a device or path. | See Table 7.<br>or<br>runCnfgSaveAs | 1.3.6.1.4.1.207.8.4.4.4.500.3.1.1 | Set:<br>*<filename.cfg>*<br>e.g.:<br>myconfig.cfg |

Table 8: Procedure for upgrading to a new software version and boot configuration (Continued)

| | DO THIS ... | BY READING OR SETTING THIS MIB OBJECT ... | WHOSE OBJECT ID IS ... | TO THIS VALUE ... |
|---|---|---|---|---|
| 11. | Check and if necessary set the file the device will use for configuration when it restarts.<br>Include the full path. | bootCnfgPath | 1.3.6.1.4.1.207.8.4.4.4.500.3.2.1 | Read/set:<br>*<filename.cfg>*<br><br>e.g.:<br>flash:/myconfig.cfg |
| 12. | Check that a boot configuration file matching the boot configuration path exists. | bootCnfgExists | 1.3.6.1.4.1.207.8.4.4.4.500.3.2.2 | Read:<br>TRUE (1)<br>or<br>FALSE (2) |
| 13. | Check that the default configuration file<br>flash:/default.cfg exists. | dfltCnfgExists | 1.3.6.1.4.1.207.8.4.4.4.500.3.3.2 | Read:<br>TRUE (1)<br>or<br>FALSE (2) |
| 14. | Restart the device. | restartDevice | 1.3.6.1.4.1.207.8.4.4.4.500.1 | 1 |