Reference Guide

AudioCodes Gateway & Session Border Controller Series

Simple Network Management Protocol

Version 7.4

audiocodes

Notice

Date Published: September-09-2024

WEEE EU Directive

Pursuant to the WEEE EU Directive, electronic and electrical waste must not be disposed of with unsorted waste. Please contact your local recycling authority for disposal of this product.

Security Vulnerabilities

All security vulnerabilities should be reported to vulnerability@audiocodes.com.

Customer Support

Customer technical support and services are provided by AudioCodes or by an authorized AudioCodes Service Partner. For more information on how to buy technical support for AudioCodes products and for contact information, please visit our website at https://www.audiocodes.com/services-support/maintenance-and-support.

Documentation Feedback

AudioCodes continually strives to produce high quality documentation. If you have any comments (suggestions or errors) regarding this document, please fill out the Documentation Feedback form on our website at https://online.audiocodes.com/documentation-feedback.

Stay in the Loop with AudioCodes

Related Documentation

SBC-Gateway Series SNMP Alarm Reference Guide
SBC-Gateway Performance Monitoring Reference Guide
MP-1288 High-Density Analog Media Gateway User's Manual
Mediant 500 Gateway & E-SBC User's Manual
Mediant 500L Gateway & E-SBC User's Manual
Mediant 800 Gateway & E-SBC User's Manual
Mediant 1000B Gateway and E-SBC User's Manual
Mediant 2600 E-SBC User's Manual
Mediant 4000 SBC User's Manual
Mediant 9000 SBC User's Manual
Mediant Software SBC User's Manual

Document Revision Record

LTRT	Description
52465	Initial document release for Ver. 7.4.
52466	Cleared alarms added
52469	SNMPSysOid and SNMPTrapEnterpriseOid removed
52470	SNMP performance monitoring introduction added with 64-bit counter note.

Table of Contents

1 Introduction 1

2 SNMP Overview 2

SNMP Standards and Objects 2

SNMP Message Standard 2

SNMP MIB Objects 3

SNMP Extensibility Feature 4

Supported MIBs 4

SNMP Interface Details 8

SNMP Community Names 9

Configuring Community Strings via the Web 9

Configuring Community Strings via the ini File 9

Configuring Community Strings via SNMP 10

SNMPv3 USM Users 11

Configuring SNMPv3 Users via ini File 12

Configuring SNMPv3 Users via SNMP 13

Trusted Managers 14

Configuring Trusted Managers via ini File 15

Configuring Trusted Managers via SNMP 15

SNMP Ports 16

Multiple SNMP Trap Destinations 16

Configuring Trap Managers via Host Name 17

Configuring Trap Managers via ini File 17

Configuring SNMP Engine ID 18

Configuring Trap Managers via SNMP 19

3 Carrier-Grade Alarm System 21

Active Alarm Table 21

Alarm History 21

3 Topology MIB Objects 22

Physical Entity (RFC 2737) 22

IF-MIB (RFC 2863) 22

Ethernet Interface 22

DS1 Interface 24

3 File Management 26

Downloading a File to the Device 26

Uploading and Deleting a File 26

3 Performance Monitoring 28

4 SNMP Traps 29

Standard Traps 29

Proprietary Traps 29

Trap Varbinds 30

SNMP Alarms in Syslog 35
Cleared Alarms 36
5 Advanced SNMP Features 37
SNMP NAT Traversal 37
Systems 37
High-Availability Systems 38
SNMP Administrative State Control 38
6 Getting Started with SNMP 40
Basic SNMP Configuration Setup 40
Configuring SNMP Port 40
Configuring Trap Managers (Trap Destination) 40
Configuring Trap Destination Port 42
Configuring Trusted Managers 42
Getting Acquainted with AudioCodes MIBs 44
Traps and Alarms 46
Device Configuration 46
Carrier Grade Alarm (CGA) 47

1 Introduction

This document provides an overview and supplementary information on Simple Network Management Protocol (SNMP) based management for AudioCodes session border controllers (SBC) and media gateways (referred to in this document as *device*).

- The **SNMP MIB manual (files)** is supplied in the Software Release Package delivered with the device.
- For a description of the device's SNMP traps (alarms and events), refer to the *SBC-Gateway SNMP Alarm Reference Guide*.
- For a description of the device's performance monitoring parameters (including SNMP), refer to the *SBC-Gateway Performance Monitoring Reference Guide*.
- For configuring SNMP through the web interface, see the device's *User's Manual*.
- For large deployments (for example, multiple devices in globally distributed enterprise offices) that need to be managed by central personnel, it is recommended to use AudioCodes One Voice Operations Center (OVOC). OVOC is not included in the device's supplied package. Contact AudioCodes for more information on its OVOC solution for large VoIP deployments.

2 SNMP Overview

Simple Network Management Protocol (SNMP) is a standards-based network control protocol for managing elements in a network. The SNMP Manager, usually implemented by a third-party Network Management System (NMS) or AudioCodes One Voice Operations Center (OVOC), connects to an SNMP Agent (embedded on a remote Network Element (NE) to perform network element Operation, Administration, Maintenance, and Provisioning (OAMP).

Both the SNMP Manager and the NE refer to the same database to retrieve information or configure parameters. This database is referred to as the Management Information Base (MIB), and is a set of statistical and control values. Apart from the standard MIBs documented in IETF RFCs, SNMP additionally enables the use of proprietary MIBs, containing non-standard information set (specific functionality provided by the Network Element).

Directives, issued by the SNMP Manager to an SNMP Agent, consist of the identifiers of SNMP variables (referred to as MIB object identifiers or MIB variables) along with instructions to either get the value for that identifier, or set the identifier to a new value (configuration). The SNMP Agent can also send unsolicited events towards an EMS, called SNMP traps.

The definitions of MIB variables supported by a particular agent are incorporated in descriptor files, written in Abstract Syntax Notation (ASN.1) format, made available to EMS client programs so that they can become aware of MIB variables and their usage.

The device contains an embedded SNMP Agent supporting both general network MIBs (such as the IP MIB), VoP-specific MIBs (such as RTP) and proprietary MIBs (acGateway, acAlarm, acMedia, acControl, and acAnalog MIBs) enabling a deeper probe into the interworking of the device. All supported MIB files are supplied to customers as part of the release.

SNMP Standards and Objects

This section discusses the SNMP standards and SNMP objects.

SNMP Message Standard

Four types of SNMP messages are defined:

- **Get:** A request that returns the value of a named object.
- **Get-Next:** A request that returns the next name (and value) of the "next" object supported by a network device given a valid SNMP name.
- **Set:** A request that sets a named object to a specific value.
- **Trap:** A message generated asynchronously by network devices. It notifies the network manager of a problem apart from the polling of the device.

Each of these message types fulfills a particular requirement of network managers:

- **Get Request:** Specific values can be fetched via the "get" request to determine the performance and state of the device. Typically, many different values and parameters can be determined via SNMP without the overhead associated with logging into the device, or establishing a TCP connection with the device.

- **Get Next Request:** Enables the SNMP standard network managers to "walk" through all SNMP values of a device (via the "get-next" request) to determine all names and values that a device supports.
- **Get-Bulk:** Extends the functionality of GETNEXT by allowing multiple values to be returned for selected items in the request. This is accomplished by beginning with the first SNMP object to be fetched, fetching the next name with a "get-next", and repeating this operation.
- **Set Request:** The SNMP standard provides a action method for a device (via the "set" request) to accomplish activities such as disabling interfaces, disconnecting users, clearing registers, etc. This provides a way of configuring and controlling network devices via SNMP.
- **Trap Message:** The SNMP standard furnishes a mechanism for a device to "reach out" to a network manager on their own (via the “trap" message) to notify or alert the manager of a problem with the device. This typically requires each device on the network to be configured to issue SNMP traps to one or more network devices that are awaiting these traps.

The above message types are all encoded into messages referred to as "Protocol Data Units" (PDUs) that are interchanged between SNMP devices.

SNMP MIB Objects

The SNMP MIB is arranged in a tree-structure, similar to a disk directory structure of files. The top-level SNMP branch begins with the ISO "internet" directory, which contains four main SNMP branches:

- **"mgmt":** Contains the standard SNMP objects usually supported (at least in part) by all network devices.
- **“private":** Contains those "extended" SNMP objects defined by network equipment vendors.
- **"experimental" and "directory":** Also defined within the "internet" root directory, are usually devoid of any meaningful data or objects.

The "tree" structure described above is an integral part of the SNMP standard, though the most pertinent parts of the tree are the "leaf" objects of the tree that provide actual management data regarding the device. Generally, SNMP leaf objects can be partitioned into two similar but slightly different types that reflect the organization of the tree structure:

- **Discrete MIB Objects:** Contain one precise piece of management data. These objects are often distinguished from "Table" items (below) by adding a “.0" (dot-zero) extension to their names. The operator must merely know the name of the object and no other information.
- **Table MIB Objects:** Contain multiple pieces of management data. These objects are distinguished from "Discrete" items (above) by requiring a "." (dot) extension to their

names that uniquely distinguishes the particular value being referenced. The "." (dot) extension is the "instance" number of an SNMP object. For "Discrete" objects, this instance number is zero. For "Table" objects, this instance number is the index into the SNMP table. SNMP tables are special types of SNMP objects, which allow parallel arrays of information to be supported. Tables are distinguished from scalar objects, such that tables can grow without bounds. For example, SNMP defines the "ifDescr" object (as a standard SNMP object) that indicates the text description of each interface supported by a particular device. Since network devices can be configured with more than one interface, this object can only be represented as an array.

By convention, SNMP objects are always grouped in an "Entry" directory, within an object with a "Table" suffix. (The "ifDescr" object described above resides in the "ifEntry" directory contained in the "ifTable" directory).

SNMP Extensibility Feature

One of the principal components of an SNMP manager is a MIB Compiler, which allows new MIB objects to be added to the management system. When a MIB is compiled into an SNMP manager, the manager is made "aware" of new objects that are supported by agents on the network. The concept is similar to adding a new schema to a database.

Typically, when a MIB is compiled into the system, the manager creates new folders or directories that correspond to the objects. These folders or directories can typically be viewed with a "MIB Browser", which is a traditional SNMP management tool incorporated into virtually all network management systems.

The act of compiling the MIB allows the manager to know about the special objects supported by the agent and access these objects as part of the standard object set.

Supported MIBs

The device contains an embedded SNMP agent supporting the MIBs listed below. A description in HTML format for all supported MIBs can be found in the MIBs directory in the release package.

- **Standard MIB (MIB-2):** The various SNMP values in the standard MIB are defined in RFC 1213. The standard MIB includes various objects to measure and monitor IP activity, TCP activity, UDP activity, IP routes, TCP connections, interfaces, IPv4 address to physical address mapping (ipNetToMediaTable), and general system description.
    - The standard icmpStatsTable and icmpMsgStatsTable under MIB-2 support ICMP statistics for both IPv4 and IPv6.
    - The inetCidrRouteTable (from the standard IP-FORWARD-MIB) supports both IPv4 and IPv6.
- **System MIB (under MIB-2):** Standard system group: sysDescr, sysObjectID, sysUpTime, sysContact, sysName, sysLocation, and sysServices. You can replace the value of sysObjectID.0 with a variable value using the ini file parameter [SNMPEnterpriseOID]. This

parameter is polled during startup and overwrites the standard sysObjectID. SNMPSysName is an administratively assigned name for this managed node. By convention, this is the node's fully-qualified domain name (FQDN). If the name is unknown, the value is the zero-length string. If the [HostName] ini file parameter is configured, its value overwrites the value of SNMPSysName.

- **RTP MIB:** The MIB is supported according to RFC 2959. It contains objects relevant to the RTP streams generated and terminated by the device and to the RTCP information related to these streams.

The inverse tables are not supported.

- **Notification Log MIB:** Standard MIB (RFC 3014 - iso.org.dod.internet.mgmt.mib-2) supported for implementation of Carrier Grade Alarms.
- **Alarm MIB:** IETF MIB (RFC 3877) Supported as part of the implementation of Carrier Grade Alarms.
- **SNMP Target MIB:** (RFC 2273) Allows for configuration of trap destinations and trusted managers.
- **SNMP MIB:** (RFC 3418) Allows support for the coldStart and authenticationFailure traps.
- **SNMP Framework MIB:** (RFC 3411).
- **SNMP Usm MIB:** (RFC 3414) Implements the user-based Security Model.
- **SNMP Vacm MIB:** (RFC 3415) Implements the view-based Access Control Model.
- **SNMP Community MIB:** (RFC 3584) Implements community string management.
- **ipForward MIB:** (RFC 2096) Fully supported.
- **RTCP-XR:** (RFC) implements the following partial support:
  - The rtcpXrCallQualityTable is fully supported.
  - In the rtcpXrHistoryTable, support of the RCQ objects is provided only with no more than 3 intervals, 15 minutes long each.
  - Supports the rtcpXrVoipThresholdViolation trap.
- **ds1 MIB:** supports the following:
  - dsx1ConfigTable: partially supports the following objects with SET and GET applied:
    - dsx1LineCoding
    - dsx1LoopbackConfig
    - dsx1LineStatusChangeTrapEnable
    - dsx1CircuitIdentifier

  All other objects in this table support GET only.

  - dsx1CurrentTable

- dsx1IntervalTable
- dsx1TotalTable
- dsx1LineStatusChange trap

- **SONET MIB:** (RFC 3592) implements the following partial support:
  - In the SonetMediumTable, the following objects are supported:
    - SonetMediumType
    - SonetMediumLineCoding
    - SonetMediumLineType
    - SonetMediumCircuitIdentifier
    - sonetMediumLoopbackConfig
  - In the SonetSectionCurrentTable, the following objects are supported:
    - lsonetSectionCurrentStatus
    - sonetSectionCurrentESs
    - sonetSectionCurrentSESs
    - sonetSectionCurrentSEFSs
    - sonetSectionCurrentCVs
  - In the SonetLineCurrentTable, the following objects are supported:
    - sonetLineCurrentStatus
    - sonetLineCurrentESs
    - sonetLineCurrentSESs
    - sonetLineCurrentCVs
    - sonetLineCurrentUASs
  - sonetSectionIntervalTable
  - sonetLineIntervalTable
  - sonetPathCurrentTable
  - sonetPathIntervalTable
- **acPSTN MIB:**
  - acSonetSDHTable: currently has one entry (acSonetSDHFbrGrpMappingType) for selecting a low path mapping type. Relevant only for PSTN applications. (Refer to the MIB for more details.)
- **acSystem MIB:**
  - acSysTransmissionType: sets the transmission type to optical or DS3 (T3).

In addition to the standard MIBs, the complete product series contains proprietary MIBs:

- **AC-TYPES MIB:** lists the known types defined by the complete product series. This is referred to by the sysObjectID object in the MIB-II.
- **AcBoard MIB:** includes the acTrap group.

Each proprietary MIB contains a Configuration subtree for configuring the related parameters. In some, there also are Status and Action subtrees.

- **AcAnalog MIB**
- **acControl MIB**
- **acMedia MIB**
- **acSystem MIB**
- **acSysInterfaceStatusTable:** supports the networking multiple interfaces feature status. This table reflects all the device's active interfaces. The lines indices consist of both the Entry Index and the Type Index. The table contains the following columns:
    - Entry Index - related Interface index in the interface configuration table (if the table is empty,i.e., there is only single IP address, the index appears with 0)
    - Type Index - 1 for IP Address and 2 for IPv6 Link-Local Address
    - Application Types - type assigned to the interface
    - Status Mode - interface configuration mode
    - IP Address - IP address (either IPv4 or IPv6) for this interface
    - Prefix Length - number of ‘1’ bits in this interface’s net mask
    - Gateway - default gateway
    - Vlan ID - VLAN ID of this interface
    - Name - interface’s name
    - Primary DNS Server IP Address - IP address of primary DNS server for this interface
    - Secondary DNS Server IP Address - IP address of secondary DNS server for this interface
- **acSysModuleTable**
- **acIPMediaChannelsresourcesTable:** IPMedia channels information such as Module ID and DSP Channels Reserved
- **acPSTN MIB**
- **acGateway MIB:** This proprietary MIB contains objects related to configuration of the SIP device. This MIB complements the other proprietary MIBs. The acGateway MIB includes the following groups:
    - Common: parameters common to both SIP and H.323.

- SIP: SIP only parameters.

- **AcAlarm:** This is a proprietary carrier-grade alarm MIB. It is a simpler implementation of the notificationLogMIB and the IETF suggested alarmMIB (both supported).

  The acAlarm MIB has the following groups:

  - **ActiveAlarm:** straight forward (single indexed) table listing all currently active Alarms together with their bindings (the Alarm bindings are defined in acAlarm. acAlarmVarbinds and also in acBoard.acTrap. acBoardTrapDefinitions. oid_1_3_6_1_4_1_5003_9_10_1_21_2_0).
  - **acAlarmHistory:** straight forward (single indexed) table listing all recently sent Alarms together with their bindings (the Alarm bindings are defined in acAlarm. acAlarmVarbinds and also in acBoard.acTrap. acBoardTrapDefinitions. oid_1_3_6_1_4_1_5003_9_10_1_21_2_0).

  The table size can be altered by one of the following:

  - notificationLogMIB.notificationLogMIBObjects.nlmConfig.nlmConfigGlobalEntryLimit
  - noti-fic-ationLo-gMIB.no-tific-ationLo-gMIBOb-jects.nlmConfig.nlmConfigLogTable.nlmConfigLogEntry.nlmConfigLogEntryLimit.

  The table size (i.e., number of contained alarms) can be any value between 10 and 1,000 (default is 500)100 (default is 100)

- A detailed explanation of each parameter can be viewed in the MIB Description field.
- A detailed description in HTML format of all MIBs can be found in the MIBs directory (included in the Release package).
- Not all groups in the MIB are implemented.
- MIB Objects that are marked as 'obsolete' are not implemented.
- When a parameter is Set to a new value via SNMP, the change may affect device functionality immediately or may require that the device be soft restart for the change to take effect. This depends on the parameter type.
- The current (updated) device configuration parameters are configured on the device provided the user doesn't load an ini file to the device after restart. Loading an ini file after restart overrides the updated parameters.

SNMP Interface Details

This subsection describes details of the SNMP interface needed when developing an Element Management System (EMS) for any AudioCodes devices, or to manage a device with a MIB browser.

There are several alternatives for SNMP security:

- SNMPv2c community strings
- SNMPv3 User-based Security Model (USM) users
- SNMP encoded over IPSec
- Various combinations of the above

Currently, both SNMP and ini file commands and downloads are not encrypted. For ini file encoding, refer to the device's *User's Manual*.

SNMP Community Names

By default, the device uses a single, read-only community string of "public" and a single read-write community string of "private". Up to five read-only community strings and up to five read-write community strings, and a single trap community string can be configured. Each community string must be associated with one of the following predefined groups:

Table 2-1: SNMP Predefined Groups

Group	Get Access	Set Access	Sends Traps
ReadGroup	Yes	No	Yes
ReadWriteGroup	Yes	Yes	Yes
TrapGroup	No	No	Yes

Configuring Community Strings via the Web

For detailed information on configuring community strings through the Web interface, refer to the device's *User's Manual*.

Configuring Community Strings via the ini File

The following ini file parameters are used to configure community strings:

- SNMPREADONLYCOMMUNITYSTRING_<x> = '#######'
- SNMPREADWRITECOMMUNITYSTRING_<x> = '#######'

Where *<x>* is a number from 0 through 4. Note that the '#' character represents any alphanumeric character. The maximum length of the string is 19 characters that can include only the following:

- Upper- and lower-case letters (a to z, and A to Z)
- Numbers (0 to 9)

- Hyphen (-)
- Underline (_)

Configuring Community Strings via SNMP

To configure community strings, the EMS must use the standard snmpCommunityMIB. To configure the trap community string, the EMS must also use the snmpTargetMIB.

➢ **To add a read-only v2user community string:**

1. Add a new row to the snmpCommunityTable with CommunityName v2user.
2. Add a row to the vacmSecurityToGroupTable for SecurityName v2user, GroupName ReadGroup and SecurityModel snmpv2c.

➢ **To delete the read-only v2user community string:**

1. If v2user is being used as the trap community string, follow the procedure for changing the trap community string. (See below.)
2. Delete the snmpCommunityTable row with CommunityName v2user.
3. Delete the vacmSecurityToGroupTable row for SecurityName v2user, GroupName ReadGroup and SecurityModel snmpv2c.

➢ **To add a read-write v2admin community string:**

1. Add a new row to the snmpCommunityTable with CommunityName v2admin.
2. Add a row to the vacmSecurityToGroupTable for SecurityName v2admin, GroupName ReadWriteGroup and SecurityModel snmpv2c.

➢ **To delete the read-write v2admin community string:**

1. If v2admin is being used as the trap community string, follow the procedure for changing the trap community string. (See below.)
2. Delete the snmpCommunityTable row with a CommunityName of v2admin and GroupName of ReadWriteGroup.

➢ **To change the only read-write community string from v2admin to v2mgr:**

1. Follow the procedure above to add a read-write community string to a row for v2mgr.
2. Set up the EM such that subsequent set requests use the new community string, v2mgr.
3. If v2admin is being used as the trap community string, follow the procedure to change the trap community string. (See below.)
4. Follow the procedure above to delete a read-write community name in the row for v2admin.

The following procedure assumes that a row already exists in the snmpCommunityTable for the new trap community string. The trap community string can be part of the TrapGroup, ReadGroup, or ReadWriteGroup. If the trap community string is used solely for sending traps (recommended), then it should be made part of the TrapGroup.

➢ **To change the trap community string:**

1. Add a row to the vacmSecurityToGroupTable with these values: SecurityModel=2, SecurityName=the new trap community string, GroupName=TrapGroup, ReadGroup or ReadWriteGroup. The SecurityModel and SecurityName objects are row indices.

⚠ You must add GroupName and RowStatus on the same set.

2. Modify the SecurityName field in the appropriate row of the snmpTargetParamsTable.
3. Remove the row from the vacmSecurityToGroupTable with SecurityName=the old trap community string.

SNMPv3 USM Users

You can configure up to 10 User-based Security Model (USM) users (referred to as SNMPv3 user). Each SNMPv3 user can be configured to one of the following security levels:

Table 2-2: SNMPv3 Security Levels

Security Levels	Authentication	Privacy
noAuthNoPriv(1)	none	none
authNoPriv(2)	MD5 or SHA-1	none
authPriv(3)	MD5 or SHA-1	DES, 3DES, AES128, AES192, or AES256

Each SNMPv3 user must be associated with one of the predefined groups listed in the following table:

Table 2-3: SNMPv3 Predefined Groups

Group	Get Access	Set Access	Sends Traps	Security Level
ReadGroup1	Yes	No	Yes	noAuthNoPriv(1)
ReadWriteGroup1	Yes	Yes	Yes	noAuthNoPriv(1)
TrapGroup1	No	No	Yes	noAuthNoPriv(1)
ReadGroup2	Yes	No	Yes	authNoPriv(2)

Group	Get Access	Set Access	Sends Traps	Security Level
ReadWriteGroup2	Yes	Yes	Yes	authNoPriv(2)
TrapGroup2	No	No	Yes	authNoPriv(2)
ReadGroup3	Yes	No	Yes	authPriv(3)
ReadWriteGroup3	Yes	Yes	Yes	authPriv(3)
TrapGroup3	No	No	Yes	authPriv(3)

The first (initial) SNMPv3 user can only be configured through a management interface other than SNMP (i.e., Web interface, configuration ini file, or CLI). Once configured, additional users can be configured through the SNMP interface as well.

Configuring SNMPv3 Users via ini File

Use the [SNMPUsers] ini file table parameter to add, modify, and delete SNMPv3 users. The [SNMPUsers] ini table is a hidden parameter. Therefore, when you load the ini file to the device using the Web interface, the table is not included in the generated file.

Table 2-4: SNMPv3 Table Columns Description

Parameter	Description	Default
Row number	Table index. Its valid range is 0 to 9.	N/A
SNMPUsers_Username	Name of the v3 user. Must be unique. The maximum length is 32 characters.	N/A
SNMPUsers_ AuthProtocol	Authentication protocol to be used for this user. Possible values are 0 (none), 1 (MD5), 2 (SHA-1)	0
SNMPUsers_ PrivProtocol	Privacy protocol to be used for this user. Possible values are 0 (none), 1 (DES), 2 (3DES), 3 (AES128), 4 (AES192), 5 (AES256)	0
SNMPUsers_AuthKey	Authentication key.	""
SNMPUsers_PrivKey	Privacy key.	""
SNMPUsers_Group	The group that this user is associated with. Possible values are 0 (read-only group), 1 (read-write group), and 2 (trap group). The actual group will be ReadGroup<sl>, ReadWriteGroup<sl> or TrapGroup<sl> where <sl> is the SecurityLevel	0

Parameter	Description	Default
	(1=noAuthNoPriv, 2=authNoPriv, 3=authPriv)	

Keys can be entered in the form of a text password or in the form of a localized key in hex format. If using a text password, then it should be at least 8 characters in length. Below is an example showing the format of a localized key:

```
26:60:d8:7d:0d:4a:d6:8c:02:73:dd:22:96:a2:69:df
```

The following sample configuration creates three SNMPv3 USM users.

```
[ SNMPUsers ]
FORMAT SNMPUsers_Index = SNMPUsers_Username, SNMPUsers_
AuthProtocol, SNMPUsers_PrivProtocol, SNMPUsers_AuthKey, SNMPUsers_
PrivKey, SNMPUsers_Group;
SNMPUsers 0 = v3user, 0, 0, -, -, 0;
SNMPUsers 1 = v3admin1, 1, 0, myauthkey, -, 1;
SNMPUsers 2 = v3admin2, 2, 1, myauthkey, myprivkey, 1;
[ \SNMPUsers ]
```

The example above creates three SNMPv3 users:

- The user v3user is set up for a security level of noAuthNoPriv(1) and is associated with ReadGroup1.
- The user v3admin1 is setup for a security level of authNoPriv(2), with authentication protocol MD5. The authentication text password is “myauthkey” and the user is associated with ReadWriteGroup2.
- The user v3admin2 is setup for a security level of authPriv(3), with authentication protocol SHA-1 and privacy protocol DES. The authentication text password is “myauthkey”, the privacy text password is “myprivkey”, and the user is associated with ReadWriteGroup3.

Configuring SNMPv3 Users via SNMP

To configure SNMPv3 users, the EMS must use the standard snmpUsmMIB and the snmpVacmMIB.

➢ To add a read-only, noAuthNoPriv SNMPv3 user, v3user:

1. Clone the row with the same security level. After the clone step, the status of the row will be notReady(3).
2. Activate the row. That is, set the row status to active(1).
3. Add a row to the vacmSecurityToGroupTable for SecurityName v3user, GroupName ReadGroup1 and SecurityModel usm(3).

A row with the same security level (noAuthNoPriv) must already exist in the usmUserTable. (see the usmUserTable for details).

➢ **To delete the read-only, noAuthNoPriv SNMPv3 user, v3user:**

1. If v3user is associated with a trap destination, follow the procedure for associating a different user to that trap destination. (See below.)
2. Delete the vacmSecurityToGroupTable row for SecurityName v3user, GroupName ReadGroup1 and SecurityModel usm.
3. Delete the row in the usmUserTable for v3user.

➢ **To add a read-write, authPriv SNMPv3 user, v3admin1:**

1. Clone the row with the same security level.
2. Change the authentication key and privacy key.
3. Activate the row. That is, set the row status to active(1).
4. Add a row to the vacmSecurityToGroupTable for SecurityName v3admin1, GroupName ReadWriteGroup3 and SecurityModel usm(3).

A row with the same security level (authPriv) must already exist in the usmUserTable (see the usmUserTable for details).

➢ **To delete the read-write, authPriv SNMPv3 user, v3admin1:**

1. If v3admin1 is associated with a trap destination, follow the procedure for associating a different user to that trap destination. (See below.)
2. Delete the vacmSecurityToGroupTable row for SecurityName v3admin1, GroupName ReadWriteGroup1 and SecurityModel usm.
3. Delete the row in the usmUserTable for v3admin1.

Trusted Managers

By default, the SNMP agent accepts Get and Set requests from any IP address, as long as the correct community string is used in the request. Security can be enhanced implementing Trusted Managers. A Trusted Manager is an IP address from which the SNMP agent accepts and processes Get and Set requests. An element management can be used to configure up to five Trusted Managers.

The concept of Trusted Managers is considered to be a weak form of security and therefore is not a required part of SNMPv3 security, which uses authentication and privacy. Trusted Managers for the devices' SNMP agent are applicable only for SNMPv2c users. An exception to this is when the community string is not the default string (‘public’/’private’), at which time Trusted Managers are applicable for SNMPV2c users alongside SNMPv3 users.

⚠ If Trusted Managers are defined, then all community strings work from all Trusted Managers. In other words, there is no way to associate a community string with specific Trusted Managers.

Configuring Trusted Managers via ini File

To set the Trusted Managers table from start up, write the following in the ini file:

```
SNMPTRUSTEDMGR_X = D.D.D.D
```

Where X is any integer between 0 and 4 (0 sets the first table entry, 1 sets the second and so on), and D is an integer between 0 and 255.

Configuring Trusted Managers via SNMP

To configure Trusted Managers, the Element Management System (EMS) must use the SNMP-COMMUNITY-MIB and snmpCommunityMIB and the snmpTargetMIB.

The following procedure assumes the following: at least one configured read-write community; currently no Trusted Managers; TransportTag for columns for all snmpCommunityTable rows are currently empty.

➢ **To add the first Trusted Manager:**

1. Add a row to the snmpTargetAddrTable with these values: Name=mgr0, TagList=MGR, Params=v2cparams.
2. Add a row to the snmpTargetAddrExtTable table with these values: Name=mgr0, snmpTargetAddrTMask=255.255.255.255:0. The agent does not allow creation of a row in this table unless a corresponding row exists in the snmpTargetAddrTable.
3. Set the value of the TransportTag field on each non-TrapGroup row in the snmpCommunityTable to MGR.

The following procedure assumes the following: at least one configured read-write community; currently one or more Trusted Managers; TransportTag for columns for all rows in the snmpCommunityTable are currently set to MGR. This procedure must be done from one of the existing Trusted Managers.

➢ **To add a subsequent Trusted Manager:**

1. Add a row to the snmpTargetAddrTable with these values: Name=mgrN, TagList=MGR, Params=v2cparams, where N is an unused number between 0 and 4.

2. Add a row to the snmpTargetAddrExtTable table with these values: Name=mgrN, snmpTargetAddrTMask=255.255.255.255:0.

An alternative to the above procedure is to set the snmpTargetAddrTMask column while you are creating other rows in the table.

The following procedure assumes the following: at least one configured read-write community; currently two or more Trusted Managers; taglist for columns for all rows in the snmpCommunityTable are currently set to MGR. This procedure must be done from one of the existing trusted managers, but not the one that is being deleted.

➢ To delete a Trusted Manager (not the last one):

- Remove the appropriate row from the snmpTargetAddrTable.

The change takes effect immediately. The deleted trusted manager cannot access the device. The agent automatically removes the row in the snmpTargetAddrExtTable.

The following procedure assumes the following: at least one configured read-write community; currently only one Trusted Manager; taglist for columns for all rows in the snmpCommunityTable are currently set to MGR. This procedure must be done from the final Trusted Manager.

➢ To delete the last Trusted Manager:

1. Set the value of the TransportTag field on each row in the snmpCommunityTable to the empty string.
2. Remove the appropriate row from the snmpTargetAddrTable.

The change takes effect immediately. All managers can now access the device. The agent automatically removes the row in the snmpTargetAddrExtTable.

SNMP Ports

The SNMP Request Port is 161 and the SNMP Trap Port is 162. These port numbers for SNMP requests and responses can be changed, by using the [SNMPPort] ini file parameter. The valid value is any valid UDP port number. The default is 161 (recommended).

Multiple SNMP Trap Destinations

An agent can send traps to up to five managers. For each manager you need to define the manager IP address and trap receiving port along with enabling the sending to that manager. You can also associate a trap destination with a specific SNMPv3 USM user. Traps are sent to this trap destination using the SNMPv3 format and the authentication and privacy protocol configured for that user.

To configure the Trap Managers table, use one of the following methods:

- Web interface (refer to the device's User's Manual)
- ini file (see Configuring Trap Managers via ini File on the next page)

- SNMP (see Configuring Trap Managers via SNMP on page 19)

Configuring Trap Managers via Host Name

One of the five available SNMP managers can be defined using the manager's host name (i.e., FQDN). This can be configured using the ini file parameter [SNMPTrapManagerHostName].

When this parameter value is defined for this trap, the device at start up tries to resolve the host name. Once the name is resolved (i.e., the IP address is found), the resolved IP address replaces the last entry of the trap manager table (defined by the parameter [SNMPManagerTableIP_x]) and the last trap manager entry of snmpTargetAddrTable in the snmpTargetMIB. The port is 162 (unless specified otherwise). The row is marked as 'used' and the sending is 'enabled'.

When using 'host name' resolution, any changes made by the user to this row in either MIBs are overwritten by the device when a resolving is redone (once an hour).

Some traps may be lost until the name resolving is complete.

Configuring Trap Managers via ini File

In the ini file, the following parameters can be set to enable or disable the sending of SNMP traps. Multiple trap destinations can be supported on the device by setting multiple trap destinations in the ini file.

- SNMPManagerTrapSendingEnable_<x>: indicates whether or not traps are to be sent to the specified SNMP trap manager. A value of ‘1’ means that it is enabled, while a value of ‘0’ means disabled. The <x> represents a number 0, 1, or 2, which is the array element index. Currently, up to five SNMP trap managers is supported.
- SNMPManagerTrapUser_<x>: indicates to send an SNMPv2 trap using the trap user community string configured with the SNMPTrapCommunityString parameter. You may instead specify an SNMPv3 user name.

The following is an example of entries in the ini file regarding SNMP. The device can be configured to send to multiple trap destinations.

```
; SNMP trap destinations
; The device maintains a table of trap destinations containing 5
; rows. The rows are numbered 0..4. Each block of 5 items below
; applies to a row in the table.
;
; To configure one of the rows, uncomment all 5 lines in that
; block. Supply an IP address and if necessary, change the port
; number.
;
; To delete a trap destination, set ISUSED to 0.
```

```
;
;SNMPManagerTableIP_0=
;SNMPManagerTrapPort_0=162
;SNMPManagerIsUsed_0=1
;SNMPManagerTrapSendingEnable_0=1
;SNMPManagerTrapUser_0=''
;
;SNMPManagerTableIP_1=
;SNMPManagerTrapPort_1=162
;SNMPManagerIsUsed_1=1
;SNMPManagerTrapSendingEnable_1=1
;SNMPMANAGERTRAPUSER_1=''
;
;SNMPManagerTableIP_2=
;SNMPManagerTrapPort_2=162
;SNMPManagerIsUsed_2=1
;SNMPManagerTrapSendingEnable_2=1
;SNMPManagerTrapUser_2=''
;
;SNMPManagerTableIP_3=
;SNMPManagerTrapPort_3=162
;SNMPManagerIsUsed_3=1
;SNMPManagerTrapSendingEnable_3=1
;SNMPManagerTrapUser_3=''
;
;SNMPMANAGERTABLEIP_4=
;SNMPManagerTrapPort_4=162
;SNMPManagerIsUsed_4=1
;SNMPManagerTrapSendingEnable_4=1
;SNMPManagerTrapUser_4=''
```

The ‘trap manager host name’ is configured via SNMPTrapManagerHostName. For example:

```
;SNMPTrapManagerHostName = 'myMananger.corp.MyCompany.com'
```

The same information that is configurable in the ini file can also be configured via the acBoardMIB.

Configuring SNMP Engine ID

The [SNMPEngineIDString] ini file parameter configures the SNMP engine ID. The ID can be a string of up to 36 characters. Once defined, the device must be restarted for the parameter to take effect.

The default value is 00:00:00:00:00:00:00:00:00:00:00:00 (12 Hex characters). The provided key must be set with 12 Hex values delimited by ':'.

If the supplied key does not pass validation of the 12 Hex values input or it is set with the default value, the engine ID is then generated, according to RFC 3411.

Before setting this parameter, all SNMPv3 users must be deleted, otherwise the configuration is ignored.

When the device operates in HA mode, the SNMPEngineIDString parameter has the same value for both active and redundant devices (i.e., system identifier). If the devices return to Standalone mode (i.e., non-HA mode), you must configure the parameter to a NULL value (i.e., no value) on both devices. When the devices restart to the standalone mode, each device automatically sets this parameter to a unique value based on its serial number (S/N).

Configuring Trap Managers via SNMP

The snmpTargetMIB interface is available for configuring trap managers.

➢ **To add an SNMPv2 trap destination:**

- Add a row to the snmpTargetAddrTable with these values: Name=trapN, TagList=AC_TRAP, Params=v2cparams, where N is an unused number between 0 and 4

All changes to the trap destination configuration take effect immediately.

➢ **To add an SNMPv3 trap destination:**

1. Add a row to the snmpTargetAddrTable with these values: Name=trapN, TagList=AC_TRAP, Params=usm<user>, where N is an unused number between 0 and 4, and <user> is the name of the SNMPv3 that this user is associated with.
2. If a row does not already exist for this combination of user and SecurityLevel, add a row to the snmpTargetParamsTable with these values: Name=usm<user>, MPModel=3(SNMPv3), SecurityModel=3 (usm), SecurityName=<user>, SecurityLevel=M, where M is either 1 (noAuthNoPriv), 2(authNoPriv) or 3(authPriv).

All changes to the trap destination configuration take effect immediately.

➢ **To delete a trap destination:**

- Remove the appropriate row from the snmpTargetAddrTable.
- If this is the last trap destination associated with this user and security level, you could also delete the appropriate row from the snmpTargetParamsTable.

➢ To modify a trap destination:

You can change the IP address and or port number for an existing trap destination. The same effect can be achieved by removing a row and adding a new row.

- Modify the IP address and/or port number for the appropriate row in the snmpTargetAddrTable.

➢ To disable a trap destination:

- Change TagList on the appropriate row in the snmpTargetAddrTable to the empty string.

➢ To enable a trap destination:

- Change TagList on the appropriate row in the snmpTargetAddrTable to ‘AC_TRAP’.
- Change TagList on the appropriate row in the snmpTargetAddrTable to "AC_TRAP".

3 Carrier-Grade Alarm System

The basic alarm system has been extended to a carrier-grade alarm system. A carrier-grade alarm system provides a reliable alarm reporting mechanism that takes into account EMS outages, network outages, and transport mechanism such as SNMP over UDP.

A carrier-grade alarm system is characterized by the following:

- The device allows an EMS to determine which alarms are currently active in the device. That is, the device maintains an active alarm table.
- The device allows an EMS to detect lost alarms and clear notifications [sequence number in trap, current sequence number MIB object]
- The device allows an EMS to recover lost alarm raise and clear notifications [maintains a log history]
- The device sends a cold start trap to indicate that it is starting. This allows the EMS to synchronize its view of the device's active alarms.

When the SNMP alarm traps are sent, the carrier-grade alarm system does not add or delete alarm traps as part of the feature. This system provides the mechanism for viewing of history and current active alarm information.

Active Alarm Table

The device maintains an active alarm table to allow an EMS to determine which alarms are currently active in the device. Two views of the active alarm table are supported by the agent:

- acActiveAlarmTable in the enterprise AcAlarm
- alarmActiveTable and alarmActiveVariableTable in the IETF standard AcAlarm MIB (rooted in the MIB tree)

The acActiveAlarmTable is a simple, one-row per alarm table that is easy to view with a MIB browser.

Alarm History

The device maintains a history of alarms that have been sent and traps that have been cleared to allow an EMS to recover any lost raise or clear traps. Two views of the alarm history table are supported by the agent:

- acAlarmHistoryTable in the enterprise AcAlarm - a simple, one-row per alarm table, that is easy to view with a MIB browser.
- nlmLogTable and nlmLogVariableTable in the standard NOTIFICATION-LOG-MIB

3 Topology MIB Objects

This section describes the topology of the MIB objects.

Physical Entity (RFC 2737)

The following groups are supported:

- entityPhysical group: Describes the physical entities managed by a single agent.
- entityMapping group: Describes the associations between the physical entities, logical entities, interfaces, and non-interface ports managed by a single agent.
- entityGeneral group: Describes general system attributes shared by potentially all types of entities managed by a single agent.
- entityNotifications group: Contains status indication notifications.

IF-MIB (RFC 2863)

The following interface types are presented in the ifTable:

- ethernetCsmacd(6): for all Ethernet-like interfaces, regardless of speed, as per RFC 3635
- ds1(18): DS1-MIB
- voiceFXO(101): Voice Foreign Exchange Office
- voiceFXS(102): Voice Foreign Exchange Station

The numbers in the brackets above refer to the IANA's interface-number.

For each interface type, the following objects are supported:

Ethernet Interface

Table 3-1: Ethernet Interface

ifTable & ifXTable	Value
ifIndex	Constructed as defined in the device's Index format.
ifDescr	Ethernet interface.
ifType	ethernetCsmacd(6)
ifMtu	1500
ifSpeed	acSysEthernetFirstPortSpeed in bits per second 0 since it’s GBE - refer to ifHighSpeed.
ifPhysAddress	00-90-8F plus acSysIdSerialNumber in hex.Will be same

ifTable & ifXTable	Value
	for both dual ports.
ifAdminStatus	Always UP. [Read Only] - Write access is not required by the standard. Support for 'testing' is not required.
ifOperStatus	Up or Down corresponding to acAnalogFxsFxoType where Unknown is equal to Down.
ifLastChange	The value of sysUpTime at the time the interface entered its current operational state.
ifInOctets	The number of octets in valid MAC frames received on this interface, including the MAC header and FCS. This does include the number of octets in valid MAC Control frames received on this interface.
ifInUcastPkts	As defined in IfMIB.
ifInDiscards	As defined in IfMIB.
ifInErrors	The sum for this interface of dot3StatsAlignmentErrors, dot3StatsFCSErrors, dot3StatsFrameTooLongs, and dot3StatsInternalMacReceiveErrors.
ifInUnknownProtos	As defined in IfMIB.
ifOutOctets	The number of octets transmitted in valid MAC frames on this interface, including the MAC header and FCS. This does include the number of octets in valid MAC Control frames transmitted on this interface.
ifOutUcastPkts	As defined in IfMIB.
ifOutDiscards	As defined in IfMIB.
ifOutErrors	The sum for this interface of: dot3StatsSQETestErrors, dot3StatsLateCollisions, dot3StatsExcessiveCollisions, dot3StatsInternalMacTransmitErrors and dot3StatsCarrierSenseErrors.
ifName	Ethernet port #1 or# 2 Gb Ethernet Port 5/n, where n is the port number
ifInMulticastPkts	As defined in IfMIB.

ifTable & ifXTable	Value
ifInBroadcastPkts	As defined in IfMIB.
ifOutMulticastPkts	As defined in IfMIB.
ifOutBroadcastPkts	As defined in IfMIB.
ifHCInOctets ifHCOutOctets	64-bit versions of counters. Required for ethernet-like interfaces that are capable of operating at 20 Mb/s or faster, even if the interface is currently operating at less than 20 Mb/s.
ifHCInUcastPkts ifHCInMulticastPkts ifHCInBroadcastPkts ifHCOutUcastPkts ifHCOutMulticastPkts ifHCOutBroadcastPkts	64-bit versions of packet counters. Required for ethernet-like interfaces that are capable of operating at 640 Mb/s or faster, even if the interface is currently operating at less than 640 Mb/s. Therefore, will be constant zero.
ifLinkUpDownTrapEnable	Set to disabled (2). Refer to [RFC 2863].
ifHighSpeed	100010 or 100 according to acSysEthernetFirstPortSpeed
ifPromiscuousMode	Constant False. [R/O]
ifConnectorPresent	Constant True.
ifAlias	An 'alias' name for the interface as specified by a network manager (NVM)
ifCounterDiscontinuityTime	As defined in IfMIB.

DS1 Interface

The DS1 interface is applicable only to digital PSTN interfaces.

Table 3-2: DS1 Digital Interface

ifTable	Value
ifDescr	Digital DS1 interface.
ifType	ds1(18).
ifMtu	Constant zero.

ifTable	Value
ifSpeed	DS1 = 1544000, or E1 = 2048000, according to dsx1LineType
ifPhysAddress	The value of the Circuit Identifier [dsx1CircuitIdentifier]. If no Circuit Identifier has been assigned this object should have an octet string with zero length.
ifAdminStatus	Trunk's Lock & Unlock during run time. In initialization process we need to refer the Admin-Status parameter.
ifOperStatus	Up or Down, according to the operation status.
ifLastChange	The value of sysUpTime at the time the interface entered its current operational state.
ifXTable	**Value**
ifName	Digital# acTrunkIndex
ifLinkUpDownTrapEnable	Set to disabled(2)
ifHighSpeed	Speed of line in Megabits per second: 2
ifConnectorPresent	Set to true(1) normally, except for cases such as DS1/E1 over AAL1/ATM where false(2) is appropriate
ifCounterDiscontinuityTime	Always zero.

3 File Management

SNMP supports file download, upload, and removal.

Downloading a File to the Device

The file URL is set in the appropriate MIB object under the acSysHTTPClient subtree (refer to the subtree objects description for the URL form). The download can be scheduled using the acSysHTTPClientAutoUpdatePredefinedTime and acSysHTTPClientAutoUpdateFrequency objects. It can also be a manual process using acSysActionSetAutoUpdate. In this case (only) and as long as one URL is set at a time, the result can be viewed in acSysActionSetAutoUpdateActionResult. In both cases, the acHTTPDownloadResult trap is sent, indicating the success or failure of the process.

acSysActionSetActionId can be set to any value and can be used to indicate an action performed by a certain manager.

A successful process also ends with the file name in the appropriate object under the acSysFile subtree or in the acCASFileTable or the acAuxiliaryFiles subtree, along with the URL being erased from the object under the acSysHTTPClient subtree.

- The action result (both in the acSysActionSetAutoUpdateActionResult object and acHTTPDownloadResult trap) for the Voice Prompt and XML indicates only that the file reached the device and has no indication on the application's ability to parse the file.
- The action result in acSysActionSetAutoUpdateActionResult is reliable as long as only one file is downloaded at a time.

Uploading and Deleting a File

File upload is the procedure of sending a file from the device to the manager. Deleting a file is erasing it from the device, an offline action that requires a restart for it to be applied. The acSysUpload subtree holds all relevant objects.

- acSysUploadFileURI indicates the file name and location along with the file transfer protocol (HTTP or NFS), for example, "http:\\server\filename.txt".
- acSysUploadFileType and acSysUploadFileNumber are used to determine the file to be uploaded along with its instance when relevant (for CAS or Video Font).
- acSysUploadActionID is at the disposal of the manager and can be used to indicate that a certain manager has performed the action.
- acSysUploadActionType determines the action that occurs and triggers it off at the same time.

File upload using SNMP is supported only for ini files; file removal using SNMP is supported for all files except ini files.

3 Performance Monitoring

The device measures performance at fixed sampling intervals. You can poll (SNMP Get) the device for these performance measurements (performance monitoring parameters) using a third-party, performance monitoring systems through an SNMP interface.

For more information on performance monitoring, refer to the SBC-Gateway Performance Monitoring Reference Guide.

For SNMP management, the device supports SNMPv1, SNMPv2, and SNMPv3. SNMPv2 or SNMPv3 is required to query 64-bit counters as SNMPv1 doesn't support 64-bit counters (per RFC 2233). Therefore, to ensure that your SNMP Get requests for performance monitoring parameters are successful, it's recommended to use SNMPv2 or SNMPv3.

4 SNMP Traps

This section provides an overview of the SNMP traps.

For a description of the device's SNMP traps (alarms and events), refer to the *SBC-Gateway SNMP Alarm Reference Guide*.

Standard Traps

The device also supports the following standard traps:

- authenticationFailure
- coldStart: The device supports a cold start trap to indicate that the device is starting up. This allows the EMS to synchronize its view of the device's active alarms. In fact, two different traps are sent at start-up:
  - Standard coldStart trap: iso(1).org(3).dod(6).internet(1). snmpV2(6). snmpModules(3). snmpMIB(1). snmpMIBObjects(1). snmpTraps(5). coldStart(1) sent at system initialization.
  - Enterprise acBoardEvBoardStarted: generated at the end of system initialization. This is more of an "application-level" cold start sent after all the initializing process is over and all the modules are ready
- linkDown
- linkup
- entConfigChange
- dsx1LineStatusChange (Applicable only to Digital Series)

Proprietary Traps

This section provides information on proprietary SNMP traps supported by the device. There is a separation between traps that are alarms and traps that are not (i.e., events or logs). All traps have the same structure made up of the same 16 varbinds (Variable Binding), i.e., 1.3.6.1.4.1.5003.9.10.1.21.1. For a list of the varbinds, see Trap Varbinds on the next page.

The source varbind is composed of a string that details the device component from which the trap is being sent (forwarded by the hierarchy in which it resides). For example, an alarm from an SS7 link has the following string in its source varbind: acBoard#1/SS7#0/SS7Link#6. The SS7 link number is specified as 6 and is part of the only SS7 module in the device that is placed in slot number 1 (in a chassis) and is the module to which this trap relates. For devices where there are no chassis options, the slot number is always 1.

Full proprietary trap definitions and trap varbinds are found in AcBoard MIB and AcAlarm MIB.

All traps are sent from the SNMP port (default 161).

Trap Varbinds

Trap varbinds are sent with each proprietary SNMP trap. Refer to the AcBoard MIB for more information on these varbinds.

Table 4-1: Trap Varbinds for Proprietary SNMP Traps

Trap Varbind	Description
acBoardTrapGlobalsName (1)	Alarm or event number. The number value is obtained from the last digit(s) of the OID of the sent trap, and then subtracted by 1. For example, for the trap acBoardEthernetLinkAlarm, which has an OID of 1.3.6.1.4.1.5003.9.10.1.21.2.0.10, the value of the varbind is 9 (i.e., 10 – 1). The value is an integer from 0 to 1000.
acBoardTrapGlobalsTextualDescription (2)	Description of the reported issue. The value is an octet string of up to 200 characters.
acBoardTrapGlobalsSource (3)	The source of the issue. For example, Trunk#1 or Entity1#x. The value is an octet string of up to 100 characters.
acBoardTrapGlobalsSeverity (4)	Active alarm severity on the device: ■ noAlarm(0) ■ indeterminate(1) ■ warning(2) ■ minor(3) ■ major(4) ■ critical(5)
AcBoardTrapGlobalsUniqID (5)	Consecutive number count of trap since device was powered on. The number is managed separately for alarms and events. For example, you may have an alarm whose value is 1 and an event whose value is 1.

Trap Varbind	Description
	The value is an integer from 0 to 32000.
acBoardTrapGlobalsType (6)	■ other(0) ■ communicationsAlarm(1) ■ qualityOfServiceAlarm(2) ■ processingErrorAlarm(3) ■ equipmentAlarm(4) ■ environmentalAlarm(5) ■ integrityViolation(6) ■ operationalViolation(7) ■ physicalViolation(8) ■ securityServiceOrMechanismViolation(9) ■ timeDomainViolation(10)
acBoardTrapGlobalsProbableCause (7)	■ other(0) ■ adapterError(1) ■ applicationSubsystemFailure(2) ■ bandwidthReduced(3) ■ callEstablishmentError(4) ■ communicationsProtocolError(5) ■ communicationsSubsystemFailure(6) ■ configurationOrCustomizationError(7) ■ congestion(8) ■ corruptData(9) ■ cpuCyclesLimitExceeded(10) ■ dataSetOrModemError(11) ■ degradedSignal(12) ■ dteDceInterfaceError(13) ■ enclosureDoorOpen(14) ■ equipmentMalfunction(15) ■ excessiveVibration(16)

Trap Varbind	Description
	■ fileError(17) ■ fireDetected(18) ■ floodDetected(19) ■ framingError(20) ■ heatingVentCoolingSystemProblem(21) ■ humidityUnacceptable(22) ■ inputOutputDeviceError(23) ■ inputDeviceError(24) ■ lanError(25) ■ leakDetected(26) ■ localNodeTransmissionError(27) ■ lossOfFrame(28) ■ lossOfSignal(29) ■ materialSupplyExhausted(30) ■ multiplexerProblem(31) ■ outOfMemory(32) ■ ouputDeviceError(33) ■ performanceDegraded(34) ■ powerProblem(35) ■ pressureUnacceptable(36) ■ processorProblem(37) ■ pumpFailure(38) ■ queueSizeExceeded(39) ■ receiveFailure(40) ■ receiverFailure(41) ■ remoteNodeTransmissionError(42) ■ resourceAtOrNearingCapacity(43) ■ responseTimeExecessive(44) ■ retransmissionRateExcessive(45) ■ softwareError(46)

Trap Varbind	Description
	■ softwareProgramAbnormallyTerminated (47) ■ softwareProgramError(48) ■ storageCapacityProblem(49) ■ temperatureUnacceptable(50) ■ thresholdCrossed(51) ■ timingProblem(52) ■ toxicLeakDetected(53) ■ transmitFailure(54) ■ transmitterFailure(55) ■ underlyingResourceUnavailable(56) ■ versionMismatch(57) ■ authenticationFailure(58) ■ breachOfConfidentiality(59) ■ cableTamper(60) ■ delayedInformation(61) ■ denialOfService(62) ■ duplicateInformation(63) ■ informationMissing(64) ■ informationModificationDetected(65) ■ informationOutOfSequence(66) ■ intrusionDetection(67) ■ keyExpired(68) ■ nonRepudiationFailure(69) ■ outOfHoursActivity(70) ■ outOfService(71) ■ proceduralError(72) ■ unauthorizedAccessAttempt(73) ■ unexpectedInformation(74)
acBoardTrapGlobalsAdditionalInfo1 (8)	Provides additional information regarding the

Trap Varbind	Description
	reported trap. The value is an octet string of up to 100 characters.
acBoardTrapGlobalsAdditionalInfo2 (9)	Provides additional information regarding the reported trap. The value is an octet string of up to 100 characters.
acBoardTrapGlobalsAdditionalInfo3 (10)	Provides additional information regarding the reported trap. The value is an octet string of up to 100 characters.
acBoardTrapGlobalsDateAndTime (11)	Date and time the trap was sent.
acBoardTrapGlobalsSystemSeverity (12)	The highest alarm severity sent by the device when the trap was sent: ■ cleared(0) ■ indeterminate(1) ■ warning(2) ■ minor(3) ■ major(4) ■ critical(5)
acBoardTrapGlobalsDeviceName (13)	Name of the device. The value is an octet string of up to 100 characters. **Note:** The device sends an empty string "\0". AudioCodes OVOC provides the proper string value when it sends it northbound.
acBoardTrapGlobalsDeviceInfo (14)	Device information. The value is an octet string of up to 100 characters. **Note:** The device sends an empty string "\0". AudioCodes OVOC provides the proper string value when it sends it northbound.
acBoardTrapGlobalsDeviceDescription	Device description.

Trap Varbind	Description
(15)	The value is an octet string of up to 100 characters. **Note:** The device sends an empty string "\0". AudioCodes OVOC provides the proper string value when it sends it northbound.
acBoardTrapGlobalsSystemSerialNumber (16)	The Serial Number of the device that sent the trap. The value is an octet string of up to 255 characters.

SNMP Alarms in Syslog

SNMP alarms are sent to the Syslog server using the following format.

- **Sent alarms:** RAISE-ALARM: <Alarm Name>; Textual Description: <Textual Description>; Severity <Alarm Severity>; Source <Alarm Source>; Unique ID: <Alarm Unique ID >.

  If additional information exists in the alarm, the following are also added: Additional Info1:/ Additional Info2:/ Additional Info3

  The message severity is as follows:

Table 4-2: Message Severity

ITU Perceived Severity (SNMP Alarm's Severity)	AudioCodes Syslog Severity
Critical	RecoverableMsg
Major	RecoverableMsg
Minor	RecoverableMsg
Warning	Notice
Indeterminate	Notice
Cleared	Notice

- **Cleared alarm:**

  CLEAR-ALARM: <Alarm Name>; Textual Description: <Textual Description>; Severity <Alarm Severity>; Source <Alarm Source>; Unique ID: <Alarm Unique ID >; If exists Additional Info1:/ Additional Info2:/ Additional Info3:

Cleared Alarms

When the device clears an alarm, it adds the prefix "Alarm cleared:" to the alarm's original text description. For example, when an Ethernet link alarm is cleared, the following alarm description is sent: "Alarm cleared: Ethernet link alarm. LAN port number 8 is down.".

5 Advanced SNMP Features

This section describes advanced SNMP features.

SNMP NAT Traversal

A NAT placed between the device and the element manager calls for traversal solutions:

- **Trap source port:** all traps are sent from the SNMP port (default is 161). A manager receiving these traps can use the binding information (in the UDP layer) to traverse the NAT back to the device. The trap destination address (port and IP) are as configured in the snmpTargetMIB.
- **acKeepAliveTrap:** this trap is designed to be a constant life signal from the device to the manager, allowing the manager NAT traversal at all times. The acBoardTrapGlobalsAdditionalInfo1 varbind has the device's serial number.

  The destination port (i.e., the manager port for this trap), can be set to be different than the port to which all other traps are sent. To do this, use the acSysSNMPKeepAliveTrapPort object in the acSystem MIB or the KeepAliveTrapPort ini file parameter.
  The Trap is instigated in three ways:

  - Via an ini file parameter [SendKeepAliveTrap] = [1]. This ensures that the trap is continuously sent. The frequency is set via the 9/10 of the [NATBindingDefaultTimeout] parameter or MIB object acSysSTUNBindingLifeTime.
  - After the STUN client has discovered a NAT (any NAT).
  - If the STUN client cannot contact a STUN server.

> The two latter options require the STUN client be enabled (ini file parameter [EnableSTUN]). In addition, once the acKeepAlive trap is instigated it does not stop.

- The manager can view the NAT type in the MIB: audioCodes(5003).acProducts (9).acBoardMibs(10).acSystem(10).acSystemStatus(2).acSysNetwork(6).acSysNAT (2).acSysNATType(1)
- The manager also has access to the STUN client configuration: audioCodes (5003).acProducts(9).acBoardMibs(10).acSystem(10).acSystemConfiguration (1).acSysNetworkConfig(3).acSysNATTraversal(6).acSysSTUN(21)
- acNATTraversalAlarm: When the NAT is placed in front of a device that is identified as a symmetric NAT, this alarm is sent. It is cleared when a non-symmetric NAT or no NAT replaces the symmetric one.

Systems

For the management of a system (a chassis with more than one type of module running), the acSystem/acSystemChassis subtree in the acSystem MIB should be used:

- The first few objects are scalars that are read-only objects for the dry-contacts' state.
- acSysModuleTable: A table containing mostly status information that describes the modules in the system. In addition, the table can be used to restart an entire system, restart a redundant module or perform switchover when for devices supporting HA.
- acSysFanTrayTable: A status-only table with the fan tray's state. Objects in the table indicate the specific state of the individual fans within the fan tray.
- acSysPowerSupplyTable: A status-only table with the states of the two power supplies.

The above tables are complemented by the following alarm traps (as defined in the acBoard MIB). For more details, see Getting Started with SNMP on page 40.

- acFanTrayAlarm: Fault in the fan tray or fan tray missing (see Fan Tray Alarm).
- acPowerSupplyAlarm: Fault in one of the power supply modules or power supply module is missing (see Power Supply Alarm).

High-Availability Systems

For the management of the High Availability (HA) systems, use the acSysChassis MIB subtree (as in the above section). The acSysModuleTable gives the HA state of the system. This includes defining which modules are active and which are in standby mode (redundant). The table also enables to read some of the statuses of the redundant modules (such as SW version, HW version, temperature, license key list, etc.). Restarting the system, restarting the redundant module, and performing switchover are performed done using this table.

Complementing the above are the following alarm traps (as defined in the acBoard MIB):

- acHASystemFaultAlarm: the HA is faulty and therefore, there is no HA.
- acHASystemConfigMismatchAlarm: configuration to the modules in the HA system us uneven causing instability.
- acHASystemSwitchOverAlarm: a switchover from the active to the redundant module has occurred.

SNMP Administrative State Control

Node maintenance for the device is provided via an SNMP interface. The acBoardMIB provides two parameters for graceful and forced shutdowns of the device. These parameters are in the acBoardMIB as follows:

- **acSysActionAdminState:** Read-write MIB object. When a GET request is sent for this object, the agent returns the current device administrative state - determines the device's desired operational state:
    - **locked (0):** Shutdown the device in the time frame set by acSysActionAdminStateLockTimeout.

- **shuttingDown (1):** (read-only) Graceful shutdown is being performed - existing calls are allowed to complete, but no new calls are allowed.
- **unlocked (2):** The device is in service.

On a SET request, the manager supplies the required administrative state, either locked(0) or unlocked(2). When the device changes to either shuttingDown or locked state, an adminStateChange alarm is sent. When the device changes to an unlocked state, the adminStateChange alarm is cleared.

- **acSysActionAdminStateLockTimeout:** Defines the time remaining (in seconds) for the shutdown to complete:
  - **0:** immediate shutdown and calls are terminated (forced lock)
  - **1:** waits until all calls are terminated (i.e., perform a Graceful shutdown)
  - **> 0:** the number of seconds to wait before the graceful shutdown turns into a force lock

The acSysActionAdminStateLockTimeout must be set before the acSysActionAdminState.

6 Getting Started with SNMP

This section provides a getting started for quickly setting up the device for management using AudioCodes SNMP MIBs.

Basic SNMP Configuration Setup

This subsection provides a description of the required SNMP configuration when first accessing the SNMP agent running on the device.

To access the device's SNMP agent, there are a few parameters that can be configured if you don't want to use default settings. The SNMP agent default settings include the following:

- SNMP agent is enabled.
- Port 161 in the agent is used for SNMP GET/SET commands.
- No default trap managers are defined and therefore, the device does not send traps.
- The trap destination port is 162.
- The SNMP agent is accessible to all SNMP managers (i.e., no trusted managers).
- SNMP protocol version is SNMPv2c with 'public' and 'private' as the read-only and read-write community strings, respectively.

Configuring these SNMP attributes is described in the following subsections:

Configuring SNMP Port

To configure the agent's SNMP port:

- ini file:

```
SNMPPort = <x>
; where 'x' is the port number
```

- CLI:

```
(config-system)# snmp settings
(snmp)# port
```

Configuring Trap Managers (Trap Destination)

Configuring Trap Managers (i.e., trap destinations) includes defining IP address and port. This configuration corresponds to the snmpTargetAddrTable. The agent supports up to five separate trap destinations. For each manager, you need to set the manager IP address and trap-receiving port along with enabling the sending to that manager.

In addition, you can associate a trap destination with a specific SNMPv3 USM user. Traps will be sent to that trap destination using the SNMPv3 format and the authentication and privacy protocol configured for that user.

- ini File: two options that can be used separately or together:
  - Explicit IP address:

```
SNMPMANAGERTABLEIP_x=<IP address>
SNMPMANAGERISUSED_x=1
SNMPMANAGERTRAPSENDINGENABLE_x=1
SNMPMANAGERTRAPPORT_x=162 ;(optional)
Where x is the entry index from 0 to 4
```

  - Manager host name:

```
SNMPTrapManagerHostName = <'host name on network'>
```

    For example: 'myMananger.corp.MyCompany.com'

    The host name is translated into the IP address using DNS resolution and is then defined as the fifth (last) trap manager. Until the address is resolved, some traps are expected to be lost.

- This option also requires you to configure the DNS server IP address (in the IP Interfaces table).
- This option results in the fifth manager being overrun by the resolved IP address. Online changes to the Manager table will also be overrun.

- SNMP: The trap managers are SET using the SNMPTargetMIB MIB onbject.
  - To add an SNMPv2 trap destination: Add a row to the snmpTargetAddrTable with these values:
    - Name=trapN, where N is an unused number between 0 and 4.
    - TagList=AC_TRAP
    - Params=v2cparamsm

    All changes to the trap destination configuration take effect immediately.
  - To add an SNMPv3 trap destination:

    i. Add a row to the snmpTargetAddrTable with these values: Name=trapN, >, where N is an unused number between 0 and 4, and <user> is the name of the SNMPv3 that this user is associated with:
    TagList=AC_TRAP
    Params=usm<user>

ii. If a row does not already exist for this combination of user and SecurityLevel, add a row to the snmpTargetParamsTable with this values:
Name=usm<user>
MPModel=3(SNMPv3)
SecurityModel=3 (usm)
SecurityName=<user>
SecurityLevel=M, where M is either 1(noAuthNoPriv), 2(authNoPriv) or 3(authPriv)

- To delete a trap destination:
  i. Remove the appropriate row from the snmpTargetAddrTable.
  ii. If this is the last trap destination associated with this user and security level, you can also delete the appropriate row from the snmpTargetParamsTable.
- To modify a trap destination, change the IP address and or port number for the appropriate row in the snmpTargetAddrTable for an existing trap destination. The same effect can be achieved by removing a row and adding a new row.
- To disable a trap destination, change TagList on the appropriate row in the snmpTargetAddrTable to the empty string.
- To enable a trap destination, change TagList on the appropriate row in the snmpTargetAddrTable to "AC_TRAP".

■ Web Interface: SNMP Trap Destinations table (Setup menu > Administration tab > SNMP folder > SNMP Trap Destinations). The check box on the left indicates if the row is used. The three columns are used to set IP address, port and enable trap sending. The SNMPv3 Users table configures trap users.

- To add a trap user: Click New, and then configure the user. The five columns include name, authentication protocol, privacy protocol, authentication key and privacy key. After configuring the columns, click Apply.
- To delete a row: Select the corresponding index field, and then click Delete.

■ CLI:

```
(config-system)# snmp trap-destination
```

Configuring Trap Destination Port

For configuring the trap destination port, see Configuring Trap Managers (Trap Destination) on page 40.

Configuring Trusted Managers

The configuration of trusted managers determines which managers can access the device. You can define up to five trusted managers.

- The concept of trusted managers is a weak form of security and is therefore, not a required part of SNMPv3 security, which uses authentication and privacy.
- Trusted managers are therefore, not supported in SNMPv3 – thus they apply only when the device is set to use SNMPv2c.
- If trusted managers are defined, then all community strings work from all trusted managers. That is, there is no way to associate a community string with particular trusted managers.

The configuration can be done via ini file, SNMP and Web.

- ini file: SNMPTRUSTEDMGR_x = <IP address>, where x is the entry index 0 to 4.
- SNMP: To configure Trusted Managers, the EM must use the SNMP-COMMUNITY-MIB, snmpCommunityMIB, and snmpTargetMIB.
    - To add the first Trusted Manager: This procedure assumes that there is at least one configured read-write community. There are currently no Trusted Managers. The TransportTag for columns for all snmpCommunityTable rows are currently empty.
        i. Add a row to the snmpTargetAddrTable with these values:
        Name=mgr0
        TagList=MGR
        Params=v2cparams.

        ii. Add a row to the snmpTargetAddrExtTable table with these values:
        Name=mgr0
        snmpTargetAddrTMask=255.255.255.255:0.

        The agent does not allow creation of a row in this table unless a corresponding row exists in the snmpTargetAddrTable.

        iii. Set the value of the TransportTag field on each non-TrapGroup row in the snmpCommunityTable to MGR.
    - To add a subsequent Trusted Manager: This procedure assumes that there is at least one configured read-write community. There are currently one or more Trusted Managers. The TransportTag for columns for all rows in the snmpCommunityTable are currently set to MGR. This procedure must be done from one of the existing Trusted Managers.
        i. Add a row to the snmpTargetAddrTable with these values:
        Name=mgrN, where N is an unused number between 0 and 4.
        TagList=MGR
        Params=v2cparams

        ii. Add a row to the snmpTargetAddrExtTable table with these values:
        Name=mgrN
        snmpTargetAddrTMask=255.255.255.255:0.

        An alternative to the above procedure is to set the snmpTargetAddrTMask column while you are creating other rows in the table.

- To delete a Trusted Manager (not the final one): This procedure assumes that there is at least one configured read-write community. There are currently two or more Trusted Managers. The taglist for columns for all rows in the snmpCommunityTable are currently set to MGR. This procedure must be done from one of the existing trusted managers, but not the one that is being deleted. Remove the appropriate row from the snmpTargetAddrTable; The change takes effect immediately. The deleted trusted manager cannot access the device. The agent automatically removes the row in the snmpTargetAddrExtTable.
- To delete the final Trusted Manager: This procedure assumes that there is at least one configured read-write community. There is currently only one Trusted Manager. The taglist for columns for all rows in the snmpCommunityTable are currently set to MGR. This procedure must be done from the final Trusted Manager.
  i. Set the value of the TransportTag field on each row in the snmpCommunityTable to the empty string.
  ii. Remove the appropriate row from the snmpTargetAddrTable; The change takes effect immediately. All managers can now access the device. The agent automatically removes the row in the snmpTargetAddrExtTable.

- Web interface: SNMP Trusted Managers table (Setup menu > Administration tab > SNMP folder > SNMP Trusted Managers). Click the Apply button for applying your configuration. Use the check boxes for deleting.
- CLI:

```
(config-system)# snmp settings
(snmp)# trusted-managers
```

Getting Acquainted with AudioCodes MIBs

AudioCodes proprietary MIBs are located in the AudioCodes subtree (OID 1.3.6.1.4.1.5003). A classification within the subtree separates the MIBs according to the following:

- **Configuration and status MIBs – in the acBoardMibs subtree**. The different MIB modules are grouped according to different virtual modules of the device. In general, the division is as follows (a more detailed breakdown of the MIBs is discussed below):
  - acBoard MIB: proprietary traps.
  - acGateway MIB: SIP control protocol specific objects. This MIB’s structure is unlike the other configuration and status MIBs.
  - acMedia MIB: DSP and media related objects. This MIB includes the configuration and status of DSP, voice, modem, fax, RTP/RTCP related objects.
  - acControl MIB: mostly MEGACO and MGCP CP related objects. A number of objects are also related to SIP. The MIB is divided into subtrees that are common to both MEGACO

and MGCP (amongst these are also the SIP relevant objects) and subtrees that are specific to the different CPs.

- acAnalog MIB: all objects in this MIB are related only to the configuration, status and line testing or resetting of analog interfaces..
- acPSTN MIB: configuration and status of trunk related objects only. Most of the MIB objects are trunk specific. .
- acSystem MIB: configuration and status of a wide range of general objects along with chassis related objects and a variety of actions that can be instigated.

- **Performance monitoring MIBs – in the acPerformance subtree**. The different MIB modules are grouped according to different virtual modules of the device. In general, the division is as follows (a more detailed breakdown of the MIBs is discussed below):
  - acPMMedia, acPMControl, acPMAnalog, acPMPSTN, acPMSystem: module specific parameters performance monitoring MIBs
  - acPMMediaServer MIB: performance monitoring specifically for MediaServer related parameters (IVR, BCT, Conference and Trunk-Testing)
  - acPerfH323SIPGateway MIB: performance specific for SIP CP devices. This MIB's structure is unlike the other performance monitoring MIBs.
- **Proprietary Carrier Grade Alarm MIB – in the acFault subtree**:
  - acAlarm: a proprietary simplification of the standard notificationLogMIB and alarmMIB (both are also supported)

The structure of the different MIBs is similar, depending on the subtree in which they reside. The MIBs in the acBoardMibs subtree have a very similar structure (except the acBoard and acGateway MIBs). Each MIB can be made up of four major subtrees:

- Configuration subtree: mostly read-write objects, tables and scalars. The relevant module's configuration is done via these objects.
- Status subtree: read-only objects, tables and scalars. Module status is collected by these objects.
- Action subtree: read-write objects that are used to instigate actions on the device (such as restart, save configuration, and so on) and read-only objects used to receive the actions' results.
- Chassis subtree (in acSystem MIB only): read-write and read-only objects related to chassis control and management (this includes, fan trays, power supply modules, PSTN IF modules, etc').

The acBoard MIB contains some deprecated objects and current proprietary trap definitions.

The acGateway MIB contains only the configuration subtree which in return is divided into common, SIP and H323 subtrees. The H323 subtree is mostly deprecated or obsolete.

Traps and Alarms

The device supports standard traps and proprietary traps. Most of the proprietary traps are alarm traps, that is, they can be sent and cleared. Thus, they are referred to as alarm traps. All the standard traps are non-alarm traps, referred to as log traps.

The proprietary traps are defined under the acBoardTrapDefinitions subtree.

The supported standard MIB traps include the following:

- coldStart
- authenticationFailure
- linkDown
- linkup
- dsx1LineStatusChange
- rtcpXrVoipThresholdViolation
- dsx3LineStatusChange
- entConfigChange

This subsection describes the device's configuration so that traps are sent out to user-defined managers under SNMPv2c or SNMPv3. It continues with an explanation on the 'carrier grade alarm' abilities and usage.

Device Configuration

For a device to send traps to specified managers, the most basic configuration are the trap targets. More advanced configuration includes the Trap Community String or traps over SNMPv3.

- Destination IP address and port (see Basic SNMP Configuration Setup on page 40)
- Trap Community String: The default Trap Community String is 'trapuser'. There is only 1 for the entire device.
    - INI file: SNMPTRAPCOMMUNITYSTRING = <your community string here>.
    - SNMP: add a new community string to the snmpCommunityTable. To associate the traps to the new Community String change the snmpTargetParamsSecurityName in the snmpTargetParamsTable so it coincides with the snmpCommunitySecurityName object. If you wish, you can remove the older Trap Community String from snmpCommunityTable (however, it is not mandatory).
    - Web: SNMP Community Settings page (Setup menu > Administration tab > SNMP folder > SNMP Community Settings). Use the Apply button to apply your configuration. You can't delete the Trap Community String, only modify its value.
    - CLI:

```
(config-system)# snmp trap
(snmp-trap)# community-string
```

- SNMPv3 Settings: When using SNMPv3 settings it is important to note that by default the trap configuration remains such that the traps are sent out in SNMPv2c mode. To have traps sent out in SNMPv3, you can use either ini file or SNMP:
    - INI file: amongst the SNMPv3 users ensure that you also define a trap user (the value of 2 in the SNMPUsers_Group indicates the trap user). For example: you can have the SNMP users table defined with a read-write user, 'rwmd5des' with MD5 authentication and DES privacy, along with a trap user, 'tmd5no' with SHA authentication and DES privacy:

```
[ SNMPUsers ]
FORMAT SNMPUsers_Index = SNMPUsers_Username, SNMPUsers_
AuthProtocol, SNMPUsers_PrivProtocol, SNMPUsers_AuthKey,
SNMPUsers_PrivKey, SNMPUsers_Group;
SNMPUsers 1 = rwmd5des, 1, 1, myauthkey, myprivkey, 1;
SNMPUsers 2 = tshades, 2, 1, myauthkey, myprivkey, 2
[ \SNMPUsers ]
```

- If you define a trap user only, the device runs in SNMPv3 mode but will not be accessible as there are no defined read-write or even read-only users.
- If you define non-default community strings (SNMPv2c), you need to access the device via SNMPv2c.

Along with this configuration, you also need to associate the trap targets (managers) with the user:

```
SNMPMANAGERTRAPUSER_x=tshades
```

where x is the target index and can be between 0 and 4.

Any targets that are defined in the ini file where this last parameter isn't defined, receives SNMPv2c traps.

- SNMP: change snmpTargetAddrParams object to the user of your choice adding the letters 'usm' as prefix (ensure it's a trap user). For example, the 'tshades' user should be added as 'usmtshades'.

Carrier Grade Alarm (CGA)

A carrier-grade alarm system provides a reliable alarm reporting mechanism that takes into account element management system outages, network outages, and transport mechanism such as SNMP over UDP.

A carrier-grade alarm system is characterized by the following:

- The device allows a manager to determine which alarms are currently active in the device. That is, the device maintains an active alarm table.
- The device allows a manager to detect lost alarms and clear notifications (sequence number in trap, current sequence number MIB object).
- The device allows a manager to recover lost alarm raise and clear notifications (maintains a log history).
- The device sends a cold start trap to indicate that it is starting. This allows the manager to synchronize its view of the device's active alarms.

When SNMP alarm traps are sent, the carrier-grade alarm system does not add or delete alarm traps as part of the feature. This system provides the mechanism for viewing history and current active alarm information.

As part of CGA, the device supports the following:

- Active Alarm Table: The device maintains an active alarm table to allow an OVOC to determine which alarms are currently active in the device. Two views of the active alarm table are supported by the agent:
    - acActiveAlarmTable in the proprietary AcAlarm MIB (this is a simple, one-row per alarm table that is easy to view with a MIB browser)
    - alarmActiveTable and alarmActiveVariableTable in the IETF standard AcAlarm MIB (rooted in the MIB tree)
- Alarm History: The device maintains a history of alarms that have been sent and traps that have been cleared to allow an OVOC to recover any lost sent or cleared traps. Two views of the alarm history table are supported by the agent:
    - acAlarmHistoryTable in the proprietary AcAlarm MIB (this is a simple, one-row per alarm table that is easy to view with a MIB browser)
    - nlmLogTable and nlmLogVariableTable in the standard NOTIFICATION-LOG-MIB

This page is intentionally left blank.

International Headquarters

6 Ofra Haza Street

Naimi Park

Or Yehuda, 6032303, Israel

Tel: +972-3-976-4000

Fax: +972-3-976-4040

AudioCodes Inc.

80 Kingsbridge Rd

Piscataway, NJ 08854, USA

Tel: +1-732-469-0880

Fax: +1-732-469-2298

Contact us: https://www.audiocodes.com/corporate/offices-worldwide

Website: https://www.audiocodes.com/

Documentation Feedback: https://online.audiocodes.com/documentation-feedback

Document #: LTRT-52470