StorageTek Automated Cartridge System Library Software

SNMP Agent

Installation and User's Guide for Solaris

Version 2.1.1

Part Number: E22184-01
Mar ch 2011

Submit comments about this document to STP_FEEDBACK_US@ORACLE.COM.

StorageTek Automated Cartridge System Library Software SNMP Agent for Soloaris Installation and User's Guide
Part Number:E22184-01

Revision History

Date	Revision	Description
March 2011	E22184-01	This document supports the ACSLS SNMP Agent 2.1.1 for the Solaris 10 SPARC and x64 platforms.

Contents

Preface

This manual provides instructions for the installation and use of the Oracle's StorageTek Automated Cartridge System Library Software (ACSLS) Simple Network Management Protocol (SNMP) Agent. The ACSLS SNMP Agent is also called the Agent in this book.

This book is intended for system or storage administrators who are responsible for monitoring library events.

It is assumed that the reader is familiar with SNMP and only limited general SNMP support is provided in this manual. For further information pertaining to the various SNMP master agents on Solaris, please consult the following documents:

For the SEA Proxy, see the "Solstice Enterprise Agent's (SEA) User's Guide"
http://www.sun.com/software/entagents/docs/UG/user.guide.pdf

"Net-SNMP release 5.4"
http://net-snmp.sourceforge.net/docs/readmefiles.html

Organization of This Guide

This guide contains the following chapters:

- Chapter 1, "Overview" on page 1 gives an introduction to the ACSLS SNMP Agent and how it works with the Management Information Base (MIB).
- Chapter 2, "Installation" on page 3 provides instructions on how to: install the ACSLS SNMP Agent for the first time; start and stop the agent; and uninstall the Agent software.
- Chapter 3, "Configuring the ACSLS Agent" on page 7 explains how to define Agent communities and trap destinations for successful communication with SNMP client applications.
- Chapter 4, "Operating the ACSLS Agent" on page 15 describes the overall behavior and use of the Agent, provides an overview of the MIB, and details specific information that is conveyed by the Agent. This chapter also provides a summary of ACSLS SNMP traps and lists some common client applications that are used to monitor the MIB and receive traps.

- Chapter 5, "Troubleshooting" on page 21 lists the tools, logs, and trace points that are available for troubleshooting. It describes the most common issues that may arise with the Agent and suggests probable solutions.
- Chapter 6, "The Agent MIB" on page 27

Typographic Conventions

Typeface*	Meaning	Examples
AaBbCc123	The names of commands, files, and directories; on-screen computer output	Edit your .login file. Use ls -a to list all files. % You have mail.
AaBbCc123	What you type, when contrasted with on-screen computer output.	% su Password:
AaBbCc123	Book titles, new words or terms, or words to be emphasized. Replace command-line variables with real names or values.	Read Chapter 6 in the User's Guide. These are called class options. You **must** be superuser to do this. To delete a file, type rm *filename*.

* The settings on your browser might differ from these settings.

CHAPTER 1

Overview

SNMP (Simple Network Management Protocol) is an industry-accepted model for collecting operating status from a wide variety of information technology hardware and software nodes within a data center. Each node is equipped with agent server software that communicates to a client. The client is typically a management application that listens for incoming traps and provides comprehensive status displays on a service console. This SNMP client retrieves status information from scores of server agents across the data center.

The role of each Agent is to expose status information to the client about the set of objects that it manages. All of the managed objects are represented in a Management Information Base (MIB). The client management application is in touch with multiple Agents reporting their respective MIBs. The management application can report on the status of each object in the entire data center. It can react to problems or status changes by sending an E-mail message to an administrator or by paging an appropriate support technician. Such a management application communicates with the ACSLS Agent by means of GET, GET_NEXT, and SET requests and it listens for TRAP messages sent from the agent. These communication packets conform to SNMP-V1 protocol.

The ACSLS SNMP Agent is responsible for objects defined within the ACS-TAPE-MONITOR-MIB. It maintains status information about storage libraries under ACSLS control and it exposes the ACS-TAPE-MONITOR-MIB database to the management application, communicating any status changes of the various objects to the management console. The ACSLS Agent works behind a Solaris SNMP Master Agent whose SNMP domain reaches to the various subsystems running on the Solaris system.

The ACSLS SNMP Agent 2.1 (hereafter called the Agent) is intended to run in a Solaris-10 environment on a host that is running ACSLS 8.0 software. The Agent provides ACSLS-queried information about the monitored ACSs and their internal components (such as LSMs, CAPs, and drives). The Agent regularly queries the libraries through the ACSLS server and sends asynchronous messages (SNMP traps) to registered clients whenever changes are detected in the status of a library or any of its components. Standard SNMP Agents listen for requests on UDP port 161 and send traps through UDP port 162. The port assignments are adjustable for administrators who require unique and secure network configurations.

More information on SNMP can be found at http://www.simple-times.org/index.html.

This manual provides installation, configuration, and operation instructions for the ACSLS SNMP Agent. The document offers hints on how to use the Agent from a SNMP management application. A troubleshooting chapter is provided to offer guidance for restoring operation of the Agent in situations arising from common problems.

CHAPTER 2

Installation

This chapter describes the installation procedure for the ACSLS SNMP Agent on Solaris 10. ACSNMP version 2.1 is available for both SPARC and X86 Solaris platforms running ACSLS version 8.0 or later.

Version 2.1 of the SNMP Agent is available for download from the Support Web. Navigate to the ACSLS Software Web directory and look for the following packages:

- ACSNMP for SPARC Solaris 10 STKacsnmp.2.1.S.tar.gz
- ACSNMP for X86 Solaris 10 STKacsnmp.2.1.X.tar.gz

Installing the Agent

▼ Installing the agent

1. Download the package and transfer it to the /opt directory on your ACSLS server.
2. Login as **root**.
3. Extract the package:

   # cd /opt

   # gunzip STKacsnmp.2.1S.tar.gz

   # tar –xvf STKacsnmp.2.1.S.tar
4. Install the package:

   # pkgadd -d .

   This operation displays a list of packages available in the current directory.
5. From the menu, select the package **STKacsnmp** .

   For ease of support, it is recommended that you select the default installation directory, **/export/home**.

The Installed Package

If you followed the default installation procedure, you find the files associated with the Agent in the directory **/export/home/ACSNMP**. You can verify the installed directory using the command, **pkginfo –r STKacsnmp**.

Hereafter any reference in this document to **$ACSNMP_HOME** equates to the directory path: **pkginfo –r STKacsnmp/ACSNMP**

In most cases, this translates to **/export/home/ACSNMP**. In all cases, you must be user **root** to run ACSNMP utilities.

In addition to the Agent software, you find several utilities to assist you as you configure and test the Agent, as well as the Agent MIB in the file, **AcslsMib.mib**. This file is provided for use by your system management application.

Numerous ACSLS SNMP Agent files are also installed in system directories for purposes of registering the ACSLS Agent with the Solaris Master Agent, and for starting up the Agent automatically.

Starting and Stopping the Agent

The ACSLS SNMP Agent is started automatically by the Solaris Service Management Facility (SMF) process during a system boot. The ACSLS Agent is intended to run as a background process as long as all dependencies are satisfied. These dependencies include the System Management Agent, the Solstice Enterprise Agent and the ACSLS application. SMF will monitor all of these dependencies and stop and restart the ACSLS Agent as needed. When ACSLS stops, the Agent also stops. The Agent restarts automatically when ACSLS restarts.

To start the ACSLS Agent for the first time, it is necessary to register the Agent with the parent agents in Solaris. To do this, run the command **agentRegister**.

$ACNMP_HOME/agentRegister

This operation restarts the Solstice Enterprise Agent (SEA) and the System Management Agent (SMA), alerting them of the new ACSLS Agent. It also registers and enables the new acsnmp service with SMF. Once acsnmp is enabled, SMF manages the service thereafter.

To stop the Agent service, use the SMF command:

svcadm disable acsnmp

Uninstalling the ACSNMP Software Package

▼ To uninstall the ACSLS SNMP Agent:

1. Stop the ACSLS SNMP Agent.

   svcadm disable acsnmp

2. Remove the package.

   pkgrm STKacsnmp

CHAPTER 3

Configuring the ACSLS Agent

This chapter describes the use of tools provided for use in configuring ACSLS SNMP communication and access control. All of these utilities require **root** user access. These utilities are found in the ACSNMP directory which is typically installed under **/export/home/**. To verify the actual directory path: **pkginfo -r STKacsnmp**.

SNMP configuration is largely a matter of aligning community names and host names for access to a specific MIB. When configuring SNMP for access to the ACSLS MIB there are several questions to ask:

- Who (what community) is submitting *get* requests?
- Who (what community) is submitting *set* requests?
- From what machines are *set* or *get* requests to be submitted?
- Who (what community) is listening for trap messages?
- To what machines must the trap messages be sent?

A community name is like a user name. This name is embedded within each SNMP request packet. Since a common user id can be used across the network, it is called a *community*. Different user communities may have different levels of access. Some communities may be able to *read* or *get* information. Others may be able to *write* or *set* parameters in the MIB. Some may be able to both *read* and *write*.

The conventional SNMP community names are *public* and *private*. Typically, the *public* community is given read access, and the *private* community can both read and write. If security is a concern, an administrator can use unconventional names other than *public* and *private*.

General Configuration

Declaring SNMP User Communities

The ACSLS Agent cannot connect to the SNMP Master Agent unless configured communities and their privileges match from one layer to the next. Where there is a mismatch between the ACSLS Agent and the Master Agent, the ACSLS Agent behaves as if the Master Agent is not running. The SNMP ports, the community, and the permissions of that community must be consistent.

▼ To Declare the Agent's Community

1. Declare the various communities in the following access control files.

   /etc/snmp/conf/snmpdx.acl (Master agent access)
   /etc/snmp/conf/AcslsAgtd.acl (ACSLS agent access)

   In most cases, it is not necessary to manipulate these files directly since there are two utilities in the ACSNMP directory that can be used to declare the communities and trap destinations.

   - By default, the ACSLS agent community is declared as *public*, but you can declare a different community name using the following expression:

     AcslsAgtdSnmpConf -c <*community name*>

   - If more than one community is to be declared, then enclose the community string in quotes, separating the community names with a comma:

     AcslsAgtdSnmpConf -c "public, private"

   - If you choose to use a community name other than public or private, be sure to declare that community in the following configuration files:

     /etc/snmp/conf/snmpd.conf
     /etc/snmp/conf/snmpdx.acl
     /etc/sma/snmp/snmpd.conf

2. Verify the changes you made in the ACSLS Agent using the **-d** (display) parameter:

   AcslsAgtdSnmpConf -d

   Generally, these communities have SNMP access from any remote machine. If you wish to limit ACSLS SNMP access to one or more specific machines, it is necessary to edit the **/etc/snmp/conf/AcslsAgtd.acl** file, listing the host names of each qualified machine in the **managers** field. By default, an asterisk (*) is inserted in the **managers** field to enable access to all hosts.

```
acl = {
      {
          communities = public, private
          access = read-write
          managers = *
      }
}
```

By removing the asterisk and inserting one or more host names, you can limit access to those specific hosts.

Port Configuration

By convention, SNMP agents use port **161** for general (set/get) operations and **162** for trap messages. If another application is using one or both of these ports, you must specify a different port. To do this, use **AcslsAgtdSnmpConf**.

▼ To change the listener port:

1. Run **AcslsAgtdSnmpConf –p <*listener port number*>**.

   The default port is 161.

   To verify the default port in the file **/etc/services**, issue the following command:

   grep snmp /etc/services

   If you change this port number from the default, make sure the port you specify agrees with any changes that may have been specified in the master agent startup script. To quickly verify the string that launches the master agent, use the following grep expression: **grep {SNMP_BIN} /lib/svc/method/svc-snmpdx**

   If the string that launches the master agent includes **-p <*port number*>**, this implies that the default request port, 161, is not being used. You can redefine the port as follows.

   - From the **$ACSNMP_HOME** directory, issue the following command:

     AcslsAgtdSnmpConf –p < *port number*>

     Make sure the port number you configure for the ACSLS agent agrees with the port number that was defined for the master agent.

     By default, the port number for SNMP trap is **162**. If your system requires a different port for traps, you can change the trap port as follows:

     AcslsAgtdSnmpConf -t <*trap port*>

   - To verify the changes you have made, use the **-d** (display) parameter

     AcslsAgtdSnmpConf -d

2. Restart the ACSLS Agent.

Trap Configuration

To declare which host machines are to receive trap messages from the agent, use the trap destination utility in the ACSNMP directory. You can *add* a hostname using the following expression:

AcslsAgtdTrapDest -a <*hostname*>

Any hostname you declare should be registered in NIS or in your local **/etc/hosts** file. This utility allows you to add only one hostname at a time, but you can repeat the command as many times to include as many hosts that you require. To get a *list* of the trap destination hosts that have been configured, use the command:

AcslsAgtdTrapDest -l

If you wish to *remove* a specific trap destination host, use the same utility with the **-r** parameter:

AcslsAgtdTrapDest -r <*hostname*>

Note – It is necessary to restart the ACSLS Agent after making changes to the trap configuration using the command: $ACSNMP_HOME/agentRegister

---

Configuring for *get* and *set* Permissions

SNMP files are configured by default to provide read-only access to the communities specified in "Declaring SNMP User Communities" on page 8. The ACSLS MIB provides for remote access to three settable parameters. See "Setting Properties of the Agent" on page 11. To manipulate these parameters remotely, you need to change the configuration to allow read-write access for your community.

The ACSLS SNMP agent on Solaris communicates through the System Management Agent. To open *read-write* access to specific communities through this agent, it is necessary to edit the System Management Agent configuration file:

/etc/sma/snmp/snmpd.conf

This file is used to configure general permissions for communities accessing the System Management Agent. Communities who are submitting only *get* requests should be declared as a *rocommunity* for read-only access. For example:

rocommunity public

Communities that are submitting *set* requests must be declared as a *rwcommunity* for read-write access. For example:

rwcommunity private

A community that is declared as an *rwcommunity* can submit to both *get* and *set* SNMP requests.

If you wish to limit *set* capabilities to a specific community on a specific machine, you must declare the machine host name on the same line:

rwcommunity <*community name*> <*hostname*>

You can also limit the ability to *set* parameters to a specific parameter in the MIB. To do this, simply list the MIB OID on the same line:

rwcommunity <*community name*> <*hostname*> <*OID*>

Example:

rwcommunity private daffy 1.3.6.1.4.1.1211.1.11.2.4.0

In this example, the system allows the *private* community on the machine *daffy* to change only the report level of SNMP traps.

Restarting the SNMP Agent

Any time port changes or community changes are made to the configuration files, the changes must be registered with the Solstice Enterprise Agent and the System Master Agent. Use the command *agentRegister*.

$ACSNMP_HOME/agentRegister

Testing your changes

StorageTek has provided a tool in the ACSNMP directory that submits a simple *get* request for each community that has been declared.

▼ To run the utility:

1. Run **$ACSNMP_HOME /agentStatus**

   This utility:

   - Checks to see that the SNMP master agent and the System Management Agent are running.
   - Reveals the ACSLS permission levels for each configured community.
   - Displays the specific hosts that can communicate with the ACSLS SNMP Agent.
   - For each community, this utility verifies access by requesting the version number of the agent.
   - Verifies whether the agent has established communication with ACSLS.
   - Lists all of the trap destination hosts that have been configured and verifies whether each is reachable.
   - If there are any problems in your configuration, this tool can lend possible assistance in identifying the source of the problem.

   Another tool in the ACSNMP directory reveals a complete list of all ACSLS OID's, listing each numeric ID, its human-readable translation, and the value that was returned for that object.

   To run this tool:

   $ACSNMP_HOME/translate

   When ACSLS is running, this utility reveals all of the objects and their OIDs throughout the entire ACSLS MIB.

Setting Properties of the Agent

There are three settable parameters within the ACSLS MIB that determine certain operating properties of the agent. These include:

acsAgtUrl **1.3.6.1.4.1.1211.1.11.1.4.0**
acsTrpCurPollingRate **1.3.6.1.4.1.1211.1.11.2.2.0**
acsTrpLogReportLevel **1.3.6.1.4.1.1211.1.11.2.4.0**

- **acsAgtUrl**

The **acsAgtUrl** can be used by a management application to identify the ACSLS agent. This parameter may have little or no use by some applications and its value in most cases is blank.

You can set this value using either of two methods:

- You can edit the file AcslsAgt.url in the ACSNMP directory, assigning the url to AGENT_URL_ENTRY. With this value set, the URL is established in the MIB when the ACSLS agent starts up.
- If the agent is currently running, you can set the URL into the MIB using the SNMP 'set' command:

  /usr/sfw/bin/snmpset -v1 -c private localhost 1.3.6.1.4.1.1211.1.11.1.4.0 s <*url expression*>

  The `-v1` parameter specifies version-1 SNMP protocol. The ACSLS Agent uses only version-1 SNMP packets and it is necessary to include this parameter. The "s" between the OID and the url_expression is a data type (STRING) declaration and is required when setting a string value. The expression following "-c" is the community identifier. If the command fails, you should use **agentStatus** to double-check the R/W permissions for the community name that you specify here.

  To verify the URL change, submit a *get* request for that OID:

  /usr/sfw/bin/**snmpget** -v1 -c public localhost 1.3.6.1.4.1.1211.1.11.1.4.0

- **acsTrpCurPollingRate**

  The **acsTrpCurPollingRate** is the period of time between SNMP probes from the agent to the ACSLS server. There is a happy medium to which an administrator would aim when setting this value. By polling ACSLS frequently, the agent is assured of fresh, up-to-date information. By polling too frequently, the agent has the potential to impact library performance, interfering with activity between ACSLS and its client applications. A reasonable polling rate is typically between 15 and 60 seconds.

  You can set the current polling rate by two different methods.

  - You can edit the AcslsAgtd.cfg file in the ACSNMP directory, adjusting the "CURR_RATE" parameter.

    Once this file is changed, you need to stop and start the ACSLS agent using

    $ACSNMP_HOME/agentRegister

  - You can dynamically change the current polling rate using an SNMP set command:

    /usr/sfw/bin/snmpset -v1 -c private localhost 1.3.6.1.4.1.1211.1.11.2.2.0 i <*number_of_seconds*>

    As described above, the **-v1** parameter defines the packet level and **-c** identifies the community. The **i** between the OID and the actual number of seconds is a type (INTEGER) declaration and is required when setting an integer value. The actual number you specify must fall in the range between fifteen seconds and sixty seconds. If you specify a number outside of this range, the set request is ignored by the agent.

  If the command fails, you can use **agentStatus** to double-check the R/W permissions for the community name that you specify here. To verify the change, use a 'get' request: **/usr/sfw/bin/snmpget -v1 -c public localhost 1.3.6.1.4.1.1211.1.11.2.2.0**

If there is a compelling reason to set this value lower than 15 seconds, you must first change the minimum polling rate to a value less than 15 seconds. The only way to adjust **acsTrpMinPollingRate** is to edit the **AcslsAgtd.cfg** file in the ACSNMP directory, adjusting the "MIN_RATE" parameter. Once this file is changed, you need to stop and start the ACSLS agent using: **$ACSNMP_HOME/agentRegister**.

- **acsTrpLogReportLevel**

  The **acsTrpLogReportLevel** defines the type and level of verbosity for trap messages. There are five possible levels defined:

1	silent	No trap messages are sent.
2	error	Only error messages are sent.
3	warning	Error messages and status changes to 'offline' are sent.
4	info	Error messages and all status changes are sent.
5	unclassified	Errors, status changes, and informational messages are sent.

The default setting is “5” (unclassified). To change the value of this parameter, use the SNMP **set** command:

/usr/sfw/bin/**snmpset** -v1 -c private localhost 1.3.6.1.4.1.1211.1.11.2.4.0 i <*level*>

As described above, the **-v1** parameter defines the packet level and **-c** identifies the community. The **i** between the OID and the specified level is a type (INTEGER) declaration and is required when setting an integer value. The actual number you specify must fall between one and five. If you specify a level outside this range, the set request will be ignored by the agent.

If the command fails, you can use 'agentStatus” to double-check the R/W permissions for the community name that you specify here. To verify the change, use a 'get' request:

/usr/sfw/bin/**snmpget** -v1 -c public localhost 1.3.6.1.4.1.1211.1.11.2.4.0

Setting the Log Trace Level

The Agent logs internal events, such as entering a function or returning a null pointer, are in `AcslsAgtd.log` log files located in the Agent home installation directory.

The Agent generates the first log file called `AcslsAgtd.log`. When the log file reaches 300 KB, it is rolled over to a backup file called AcslsAgtd.log.0. The AcslsAgtd.log file is then flushed to leave room for a new 300 KB worth of information. The size of the two log files put together never exceeds 60 KB.

Four trace levels are available:

- SILENT produces no trace information

- ERROR traces errors only

- WARNING provides both error and warning information

- DEBUG traces errors, warnings and all the Agent’s operations (all messages are recorded)

Setting the Log Trace Level

The trace level is set with the environment variable **ACS_TRACE**. This variable is defined in the service startup method for acsnmp:

/lib/svc/method/svc-acsnmp

The variable **ACS_TRACE** can be set for any of the following values:

DEBUG, WARNING, ERROR or SILENT

Note – The default trace level is WARNING.

Setting the log trace level to DEBUG accelerates log file roll over.

Note – You must restart the acsnmp service in order for the logging changes to take effect: **$ACSNMP_HOME/agentRegister**

CHAPTER 4

Operating the ACSLS Agent

Once the Agent has been installed and configured, there is little in the way of operation or maintenance. The Agent starts and stops automatically. It responds to *get* and *set* requests from remote management applications. It also sends traps to registered clients to report ACSLS and library operational events.

Agent Behavior

The ACSLS SNMP Agent operates in a continuous loop, polling the status of ACSLS devices, checking for any status changes, and sending a corresponding trap to registered trap recipients with each relevant change in status. The periodicity of this loop is between 15 and 60 seconds, determined by the MIB parameter, **acsTrpCurPollingRate**, which is described in Chapter 3, "Configuring the ACSLS Agent".

If ACSLS software should go down for any reason, the SNMP Agent (AcslsAgtd) will also go down. When ACSLS is reactivated, the Solaris Service Management Facility (SMF) will automatically launch the Agent. Should the Agent process be killed for any reason, SMF will reactivate the process. SMF also watches over the Agent's API client to ACSLS (**snmpssi**).

To manually bring down the ACSLS SNMP Agent, use the Solaris svcadm command:

svcadm disable acsnmp

This command allows you to gently shut down all of the Agent components, including **AcslsAgt** and **snmpssi**.

The ACSLS MIB

The Management Information Base (MIB) is a machine-readable document that lists all object IDs (OIDs) associated with the Agent and defines all of the information that is maintained about each object. The ACS-TAPE-MONITOR-MIB is included in the ACSNMP home directory under the file name **AcslsMib.mib**. It is a text file that can be copied and transferred to another machine for use by an SNMP management

application. Typically such applications compile each MIB under its management and uses the information as a means to translate OIDs to the actual objects being monitored by the Agent. You can review a copy of the ACSLS MIB in Chapter 6, "The Agent MIB".

The MIB defines an actual database that is maintained by the ACSLS Agent. The ACSLS Agent database contains information about library resources. It includes a current count of configured ACSs, LSMs, Drives, and CAPS in the library. It also records the type and the location of each configured tape drive and it knows the size of each CAP in the library. All of this information remains stable and unchanged in the MIB database until ACSLS has been reconfigured and the Agent has restarted.

The MIB database also maintains dynamic information when an object status changes from moment to moment. The Agent keeps track of the number of free cells in each ACS and LSM. The database is updated dynamically as resources are varied offline and back online, or as drives are placed in use or as they become available. If a volume is mounted to a drive, the Agent records the drive status and the volume ID in the MIB database. If a CAP is opened or as CAP priority changes, this information is maintained by the Agent.

When the ACSLS connection is broken, the Agent purges its MIB table entries. ACS, LSM, drive, and CAP counts are changed to 0 (zero). When a new ACSLS connection is detected, the Agent restarts and a new database is created from the fresh information reported by ACSLS.

To view a complete list of translated OIDs for your ACSLS system and to see the current status of each MIB object, you can use the **translate** utility

To view a complete list of translated OIDs for your ACSLS system and to see the current status of each MIB object, you can use the translate utility:

$ACSNMP_HOME/translate

The default behavior of translate displays alpha-numeric OIDs. To view the OIDs in their strictly numeric form, use the **-n** option:

$ACSNMP_HOME/translate -n

Similarly, you can quickly *walk* the ACSLS MIB on the local machine using the *walker* utility:

$ACSNMP_HOME/walker -n

SNMP Traps

A *trap* is an informational message that is sent to registered clients whenever the status of an object has changed. This information can be displayed by a management application in an event console or it can trigger an action defined by the administrator of the management application. There are as many traps defined in the MIB as there are possible statuses returned for the Agent or for the library resources that are monitored by the Agent.

Trap Samples

Agent Start Trap

Agents send a trap every time they are started. These are called *start* traps. Each start trap contains the related boot date for information purposes.

Example

```
ACSLS Agent:
Trap Number = 11
Enterprise OID = 1.3.6.1.4.1.1211.1.11
acsAgtBootDate.0 :
                    OID = 1.3.6.1.4.1.1211.1.11.1.3.0
                    Value : "2002-02-21T05:01"
```

This type of trap can be used by a management application to re-synchronize its own data model with the information available from the Agent. There are numerous reasons to restart the Agent but one typical reason is to allow the Agent to record any hardware configuration changes made to the attached library. Consequently, when receiving this trap, the management application would update the information pertaining to the ACS-TAPE-MONITOR-MIB.

Status Traps

Status traps alert the client that a status change has occurred on a component within the MIB. To facilitate component identification among the collection of components detected by the Agent, object identification along with status information is provided in the trap. For ACS, this information includes the ACS state, index, and ID. For LSMs, this information includes the LSM state and status, the ACS index, the LSM index, and the LSM ID. For drives, this information includes the drive state and status, the ACS index, the LSM index the drive index, and the drive ID. For CAPs, this information includes the CAP state and status, the CAP priority, the ACS index the LSM index, the CAP index, and the CAP ID.

The following example shows the structure of a typical status trap.

Example: Drive State Offline

```
Trap Number = 51
Enterprise OID = 1.3.6.1.4.1.1211.1.11
acsDriveId.1.2.15
                    OID = 1.3.6.1.4.1.1211.1.11.3.3.2.1.4.1.2.15
                    Value: “0, 1, 10, 2”
acsDriveState.1.2.15
                    OID = 1.3.6.1.4.1.1211.1.11.3.3.2.1.6.1.2.15
                    Value: 3
acsDriveStatus.1.2.15
                    OID = 1.3.6.1.4.1.1211.1.11.3.3.2.1.5.1.2.15
                    Value: 1
acsDriveAcsIndex.1.2.15
```

```
                OID = 1.3.6.1.4.1.1211.1.11.3.3.2.1.1.1.2.15
                Value: 1
acsDriveLsmIndex.1.2.15
                OID = 1.3.6.1.4.1.1211.1.11.3.3.2.1.2.1.2.15
                Value: 2
acsDriveIndex.1.2.15
                OID = 1.3.6.1.4.1.1211.1.11.3.3.2.1.3.1.2.15
                Value: 15
```

SNMP Client Utilities

The information provided by the ACSLS SNMP Agent is viewable only by means of an SNMP client application. Such applications are designed to submit SNMP queries (e.g. get, getnext, walk), to change (set) variable parameters in the Agent, and ultimately, to listen for trap messages from the Agent. There is a wide range of commercial applications for network management including HP OpenView, IBM Tivoli NetView, and CA Unicenter. What follows in this section are a few pointers to commonly available tools on standard systems.

NetSNMP

The most widely-accessible client application is NetSNMP. This is an open-source, public domain application that is freely available on all popular platforms including Solaris, AIX, HP-UX, Linux, MacOS, and Windows. NetSNMP is bundled as a standard package with Solaris-1

Common NetSNMP commands on Solaris reside in /usr/sfw/bin. These include snmpget, snmpgetnext, snmpbulkget, snmpwalk, and snmpset. NetSNMP is a command-line application and each command contains a structure of parameters. For example, to view a specific OID from the ACSLS MIB, you can use the snmpget command as follows:

snmpget -v1 –m <*MIB pathname*> -c public <*hostname*> acsAgtRelease.0

In this example, we are requesting the release level of the ACSLS Agent. The SNMP packet version (v1) is specified since the ACSLS Agent supports only **v1** packets. The path to the ACSLS MIB is specified. You would copy the MIB file from the ACSLS machine to any remote machine if you intend to submit MIB-referenced SNMP queries from that machine. The community name and hostname are also passed as arguments. Finally the object id is specified. In this example, we are asking for the first instance **acsAgtRelease**. The snmpget utility consults the MIB to translate the object id to its actual numeric value.

If you know the translated OID, you can pass its numeric value directly in a command string that does not require a MIB lookup:

snmpget –v1 –c public <*hostname*> 1.3.6.1.4.1.1211.1.11.1.1.0

Hint: You can get a complete list of ACSLS Agent OIDs using **./translate –n** in the ACSNMP directory of the ACSLS machine.

A NetSNMP trap listener utility resides in /usr/sfw/sbin.

snmptrapd -P –m <*MIB pathname(s)*>

In this example, traps that are sent to the machine from which you launched the trap daemon are displayed to standard error of your shell environment.

For more information about NetSNMP, see the SourceForge web site:

http://net-snmp.sourceforge.net/

Windows NT Resource Kit

The Windows NT resource kit provides a utility called *snmputil*.

c:\ntreskit>**snmputil get** <*hostname*> public .1.3.6.1.4.1.12.11.1.11.1.2.0

The command in this example returns the status of the Agent. Of course, you need to know the OID of the object of your interest and notice that the OID expression in this application always begins with a dot. (You can get a complete list of ACSLS Agent OIDs using *translate -n* in the ACSNMP directory of the ACSLS machine). The Windows *snmptutil* also provides a *walk* function.

AIX snmpinfo

AIX provides the /usr/sbin/snmpinfo command (equivalent to the Windows NT *snmputil* command) which performs SNMPGET and SET requests.

To request the status of the Agent using snmpinfo, the command would be as follows:

/usr/sbin/**snmpinfo -m get -h** <*hostname*> 1.3.6.1.4.1.1211.1.11.1.2.0

The snmpinfo utility supports SNMP *get*, *getnext*, *set*, and *dump* operations.

CHAPTER 5

Troubleshooting

This chapter summarizes various diagnostic tools that have been provided and itemizes a number of issues that might arise during the operation of the ACSLS SNMP Agent. With each issue, an explanatory context is provided along with a suggested course of action.

Tools

A general purpose diagnostic tool is provided that can verify Agent configuration and check for multiple dependencies. To run the tool, go to the ACSNMP home directory and run:

agentStatus

This utility will check the following dependencies:

- Verifies ACSLS is running.
- Verifies the Solstice Enterprise Master Agent, *snmpdx* is running.
- Verifies the System Management Agent, *snmpd* is running.
- Verifies SNMP requests are allowed from the local host.
- Checks which communities are configured to submit requests.
- Verifies that each community is known to the System Management Agent.
- Verifies *read-only* or *read-write* MIB access to each community.
- Exercises a *read-only* operation for each community.
- Looks to see what hosts have been defined as trap recipients.
- Confirms that each host is known to the Enterprise Master Agent.
- Pings each trap destination host to verify the host is reachable.

If any of these dependencies is lacking, the tool display an appropriate error message. Otherwise, it displays the information that it has confirmed.

Other tools

To review the details of the Agent package installed on your machine.

pkginfo -STKacsnmp

To locate the directory where the Agent has been installed.

pkginfo -r STKacsnmp

To stop the ACSLS Agent

svcadm disable acsmp

To start the ACSLS Agent

svcadm enable acsmp

Execution Issues

Solstice Enterprise Master Agent Does Not Start

The Master Agent may fail to start if it cannot parse information contained in its configuration files (see "Configuring the ACSLS Agent" on page 7). When the Master Agent cannot interpret configuration data in these files, a parse error message will be recorded in the system log, **/var/adm/messages**. Check to be sure that any changes you have made to the configuration files are correct and accurate.

If the hostname of a trap destination cannot be resolved, the Master Agent can fail to start. Make sure that the host is reachable for any trap recipient that has been configured.

The Library Configuration Has Changed

Whenever changes are made to the library hardware, you should reconfigure the ACSLS server (refer to the *ACSLS 8.0 Installation, Configuration and Administration Guide*) and restart the ACSLS server. The Agent will restart when ACSLS restarts and the hardware changes are applied to the ACS-TAPE-MONITOR-MIB.

Stopping the ACSLS Server is not always required when adding, removing or changing tape drives. (Refer to the *ACSLS 8.0 Installation, Configuration and Administration Guide* for details on the *config drives* command.) If the tape drive configuration is changed without restarting ACSLS, you should register the changes with the master agents in order to apply the changes to the ACS-TAPE-MONITOR-MIB.

$ACSNMP_HOME/agentRegister

ACSLS Library Server Not Running

If ACSLS is not running, the ACSLS Agent does not run. The agent remains inactive until ACSLS restarts. The Agent starts automatically after ACSLS starts.

Trouble Removing the Agent

The Agent must be stopped before it can be removed. Any attempt to run **pkgrm STKacsnmp** while the Agent is running fails with the message:

ERROR, the Agent is running

To stop the Agent, use:

svcadm disable acsnmp

SNMP trap errors

If a configured SNMP management application fails to receive traps from the ACSLS Agent, you can verify the trap destination using **agentStatus**.

If **agentStatus** reports that the trap destination is **ok**, you should check whether the destination host machine can be configured with firewall software. You should also check to confirm that the application is listening on the appropriate port. The default trap listener port is **162**. If the client is configured to listen on a different port, you can reconfigure the defined trap port of the Agent using *AcslsAgtSnmpConf*:

AcslsAgtSnmpConf -t <*port*>

If *agentStatus* reports that the trap destination is not configured, then check the file **/etc/snmp/conf/snmpdx.acl**. Make sure that the desired trap hostname is listed among the hosts in the trap parameters, and that the trap communities are properly defined.

If *agentStatus* reports that the trap destination is **unreachable**, then check to see that the hostname is defined in your NIS database or in your local **/etc/hosts** file. You should use a standard "ping" to verify that the remote host is reachable from the Agent machine. If "ping" fails, you should resolve any routing issues on your local network.

If *agentStatus* reports ***none*** for the trap destination, use the *AcslsAgtdTrapDest* routine to add the appropriate host name.

AcslsAgtdTrapDest -a <*hostname*>

You need to restart the Agent if you have made any configuration changes to the trap destination list or to the port.

Note – When restarting the Agent, the SNMP start trap is sent with the boot date.

SNMP Requests Generate SNMP Timeouts

If the Agent is running but there is no sign of activity in response to a *get* or *set* request, you should use *agentStatus* to verify the community configuration. Go to the ACSNMP directory and issue the command.

agentStatus

For *get* requests, make sure that the specified community has *read-only* or *read-write* access. For *set* requests, make sure that the specified community has *read-write* access. If *agentStatus* does not confirm the proper access for the community in question, refer to the configuration procedure in "Configuring the ACSLS Agent" on page 7.

SNMP Requests Generate “Connection refused” Error

The *connection refused* error will be returned whenever the SNMP Manager is trying to retrieve information on an unauthorized port. This is usually due to a difference between the SNMP port used by the Master Agent and the port used by the ACSLS Agent.

The default port is **161**. If you change this port number from the default, make sure the port you specify agrees with any changes that may have been specified in the master agent startup script.

grep SNMP_BIN /lib/svc/method/svc-snmpdx

If the string that launches the master agent includes **-p <*port number*>**, this implies that the default request port, 161, is not being used. You can redefine the port as follows. From the ACSNMP home directory, issue the following command from the **$ACSNMP_HOME** directory:

AcslsAgtdSnmpConf -p < *port number*>

The port number you configure for the ACSLS agent should agree with the port number that was defined for the master agent. Once you have changed the port configuration, you should restart the ACSLS Agent.

ACSLS SNMP Agent Starts but then Stops after One Minute

If the **ACSNMP/AcslsAgtd.log** file includes messages, “failed to connect to Master Agent”, this is a sign that the ACSLS Agent could not determine whether the Master Agent was active. Either of the following could account for this:

- The Master Agent is not running.
- The Master Agent and the ACSLS Agent do not use the same SNMP request port.

You can verify the common port using the procedure described just above.

To restart the master agent, use the following procedure:

$**ACSNMP_HOME/agentRegister**

Log Files

When troubleshooting problems with the ACSLS Agent, be sure to look at clues that are left behind in the Agent log. This log file resides in the **$ACSNMP_HOME** directory under the name **AcslsAgt.log**.

If the Service Management Facility is unable to start acsnmp for any reason, it will place the service in 'maintenance' mode. You can view the service status using 'svcs'.

svcs acsnmp

If the acsnmp service is in maintenance mode, you want to view the service startup log. To discover the pathname for the logfile, use the **-l** option:

svcs -l acsnmp

Another useful file to inspect when SNMP fails to start is the system messages file, **/var/adm/messages**.

Master Agent Tracing

In the event that the logs provide no clues to the problem you are experiencing, you can extract trace information from request/response traffic to the defined port of the System Management Agent (snmpd). To capture a trace of SNMP packets (in hexidecimal format), do the following:

1. Disable the System Management Agent

   svcadm disable sma

2. Edit the sma service method, adding "-d-L" of the snmpdx invocation string.

   /usr/sfw/sbin/snmpd -d -L

   The **-d** option places the master agent into debug mode. The -L option instructs the agent to send debug traffic to standard error.

3. Restart the master agent.

   svcadm disable sma

   - You can find the path to the trace log using **svcs -l sma**:

     /var/svc/log/application-management-sma:default.log

     The get/set packets bound to the Master Agent are logged in this file.

   - You can generate request traffic by restarting the System Management Agent.

     svcadm restart sma

     When the System Management Agent restarts, you should see traffic activity in the Master Agent trace log. If you do not see trace activity, this points to a problem at the system level between the System Management Agent and the Solstice Enterprise Agent (SEA Proxy). Review any changes you may have made in the files **/etc/sma/snmp/snmpd.conf** and **/etc/snmp/conf/snmpdx.conf**. For further information, consult the *Solaris System Management Agent Administration Guide*.

     http://dlc.sun.com/pdf/817-3000/817-3000.pdf

     If the trace window shows transaction activity, try sending a command from the ACSLS Agent. One easy way to do this is to run the agentStatus utility from the $ACSNMP_HOME directory. This simple test utility submits a single snmpget request through the interface to the master agent. If the trace window reveals activity, it confirms good communication via the request port (161) between the ACSLS Agent and the System Management Agent. The information shown in the trace may offer further hints by posting error messages that are relevant to the root of the problem.

If the trace window remains static in response to the agentStatus command, it points to communication problems between the ACSLS Agent and the System Management Agent. Review any changes you may have made in the files /etc/snmp/conf/AcslsAgt.acl, /etc/snmp/conf/snmpdx.acl and /etc/snmp/conf/snmpd.conf.

Essential Commands

The following table provides a list the most commonly used commands for the SNMP Agent.

svcadm enable acsnmp	Starts the ACSLS agent.
ACSNMP/agentRegister	Registers changes to the ACSLS agent and restarts.
ACSNMP/agentRegister	Restarts both master agents and the ACSLS agent.
svcadm disable acsnmp	Stops the ACSLS agent.
ACSNMP/AcslsAgtdTrapDest	Adds or removes trap target addresses.
ACSNMP/AcslsAgtdSnmpConf	Configures Agent community and ports.
ACSNMPagentStatus	General purpose ACSLS Agent diagnostic utility.
ACSNMP/walker	Simple test probe of the entire ACSLS MIB.
ACSNMP/translate	Shows all ACSLS OIDs and their values.

Note – ACSNMP is the installation directory for the ACSLS Agent, **$ACSNMP_HOME**. The default directory is **/export/home/ACSNMP**.

CHAPTER 6

The Agent MIB

```
-- ------------------------------------------------------------------
-- ACS-TAPE-MONITOR-MIB Release 1.0 draft 3
-- ------------------------------------------------------------------
-- 1.0 d3   :  reviewed document : information of the MIB is no more
-- providing LMU count information.
-- ------------------------------------------------------------------
-- 1.0 d2   :  reviewed document : information of the MIB could be
--             provided through the ACS API 5.4.
-- ------------------------------------------------------------------
-- 1.0 d1   :  Initial draft
-- ------------------------------------------------------------------
ACS-TAPE-MONITOR-MIB DEFINITIONS ::= BEGIN
     IMPORTS
             enterprises, OBJECT-TYPE, IpAddress
                   FROM RFC1155-SMI
                   DisplayString
                   FROM RFC1213-MIB;

     storagetek   OBJECT IDENTIFIER ::= { enterprises 1211 }
     products     OBJECT IDENTIFIER ::= { storagetek  1  }


-- Official Product number
     acsTapeMonitorOBJECT IDENTIFIER ::= { products 11 }

     acsAgentOBJECT IDENTIFIER ::= { acsTapeMonitor 1 }
     acsTrap      OBJECT IDENTIFIER ::= { acsTapeMonitor 2 }
     acsHardwareOBJECT IDENTIFIER ::= { acsTapeMonitor 3 }


     --
     -- ACS Tape Library Monitor types definition
     --
```

```
AcsBoolean ::= INTEGER {
                    true (1),
                    false (2)
                    }

-- ------------------------------------------
--
-- acsAgent sub tree
--
-- ------------------------------------------

acsAgtRelease  OBJECT-TYPE
       SYNTAX DisplayString
       ACCESS read-only
       STATUS mandatory
       DESCRIPTION
              "The release of the agent. Format is %d.%d"
       ::= { acsAgent 1 }


acsAgtStatus OBJECT-TYPE
       SYNTAX INTEGER
              {
              initializing (1) ,
              running (2),
              expiring (3),
              expired (4)
              }
       ACCESS read-only
       STATUS mandatory
       DESCRIPTION
              "When starting and initializing internal variables,
              the agent is >initializing< and may not answer correctly
              to request. Status of the agent should be queried at the
              beginning of every session and respond >running<. When
              license expires in less than 3 days, the status is
              >expiring<. When license is no more valid, the status of
              the agent switch to >expired<."
       ::= { acsAgent 2 }

acsAgtBootDate  OBJECT-TYPE
       SYNTAX DisplayString
       ACCESS read-only
       STATUS mandatory
       DESCRIPTION
```

```
                "The date & time when the agent started.
                Format is YYYY-MM-DDThh:mm"
        ::= { acsAgent 3 }

acsAgtUrl  OBJECT-TYPE
        SYNTAX DisplayString
        ACCESS read-write
        STATUS mandatory
        DESCRIPTION
                "Url to be provided at the agent level-Can be used
                for library web based management purpose.
                This item will exist only if it exists a Web based
                library  management application for an ACS or an LSM.
                "
        ::= { acsAgent 4 }


-- ---------------------------------------------
--
--     acsTrap sub tree
--
-- ---------------------------------------------

acsTrpMinPollingRate OBJECT-TYPE
        SYNTAX INTEGER
        ACCESS read-only
        STATUS mandatory
        DESCRIPTION
                "Minimum polling rate in second.
                The value is read
                from a file when the agent starts"
        ::= { acsTrap 1 }

acsTrpCurPollingRate OBJECT-TYPE
        SYNTAX INTEGER
        ACCESS read-write
        STATUS mandatory
        DESCRIPTION
                "Current polling rate in seconde. The value could
                not be set under acsTrpMinPollingRate"
        ::= { acsTrap 2 }


acsTrpMsg OBJECT-TYPE
        SYNTAX DisplayString
```

```
        ACCESS read-only
        STATUS mandatory
        DESCRIPTION
                "A trap displayString varbind"
        ::= { acsTrap 3 }

 acsTrpLogReportLevelOBJECT-TYPE
        SYNTAX  INTEGER {
                silent (1),
                error (2),
                warning (3),
                info (4),
                unclassified(5)
        }
        ACCESS  read-write
        STATUS  mandatory
        DESCRIPTION
                "Log message reporting level preferred by the Enterprise
                Management Framework destinations.
                'silent' means no message is sent to the framework.
                'error' means only error messages are sent.
                'warning' means errors+warnings are sent.
                'info' means  errors+warnings+information messages are sent.
                'unclassified' means every messages stored into
                the library are sent to the framework"
        ::= { acsTrap 4 }




 -- TRAP DEFINITIONS


 -- Message traps : Traps 1 to 4
 --
 -- In what follows, the traps should contain the message
 -- to be displayed into the event console of the framework
-- There is one trap defined per possible severity level of the
-- messages

 acsTrpErr TRAP-TYPE
 ENTERPRISE acsTapeMonitor
 VARIABLES { acsTrpMsg }
 DESCRIPTION
        "A error trap message"
```

```
::= 1

acsTrpWar TRAP-TYPE
ENTERPRISE acsTapeMonitor
VARIABLES { acsTrpMsg }
DESCRIPTION
       "A warning trap message"
::= 2

acsTrpInfo TRAP-TYPE
ENTERPRISE acsTapeMonitor
VARIABLES { acsTrpMsg }
DESCRIPTION
       "An info trap message"
::= 3

acsTrpUncl TRAP-TYPE
ENTERPRISE acsTapeMonitor
VARIABLES { acsTrpMsg }
DESCRIPTION
 "An unclassified trap message"
::= 4


--
--   Agent Boot : cold start trap
--
acsAgentStart TRAP-TYPE
ENTERPRISE acsTapeMonitor
VARIABLES { acsAgtBootDate }
DESCRIPTION
       "This trap is sent when the agent starts"
::= 11




-- acs status related traps : Traps 20 to 24
--
-- These traps are sent when the status of the acs changes.


acsTrpAcsStateOnline TRAP-TYPE
 ENTERPRISE acsTapeMonitor
 VARIABLES { acsAcsId,
```

```
                        acsAcsState,
--                      acsAcsStatus,
--                      acsAcsAlias,
                        acsAcsIndex
                         }
     DESCRIPTION
            " This trap is sent when an Acs become online."
    ::= 20

   acsTrpAcsStateOffline TRAP-TYPE
     ENTERPRISE acsTapeMonitor
     VARIABLES { acsAcsId,
                        acsAcsState,
--                      acsAcsStatus,
--                      acsAcsAlias,
                        acsAcsIndex
                         }
     DESCRIPTION
            " This trap is sent when an Acs become offline."
    ::= 21

    acsTrpAcsStateOfflinePending TRAP-TYPE
     ENTERPRISE acsTapeMonitor
     VARIABLES { acsAcsId,
                        acsAcsState,
--                      acsAcsStatus,
--                      acsAcsAlias,
                        acsAcsIndex
                         }
     DESCRIPTION
            " This trap is sent when an Acs is  offline pending."
    ::= 22


   acsTrpAcsStateRecovery TRAP-TYPE
     ENTERPRISE acsTapeMonitor
     VARIABLES { acsAcsId,
                        acsAcsState,
--                      acsAcsStatus,
--                      acsAcsAlias,
                        acsAcsIndex
                         }
     DESCRIPTION
            " This trap is sent when an Acs is starting a recovery."
    ::= 23
```

```
   acsTrpAcsStateDiagnostic TRAP-TYPE
     ENTERPRISE acsTapeMonitor
     VARIABLES { acsAcsId,
                         acsAcsState,
--                       acsAcsStatus,
--                       acsAcsAlias,
                         acsAcsIndex
                          }
     DESCRIPTION
            " This trap is sent when an Acs enters in a diagnostic phase."
    ::= 24



     -- lsm status related traps : Traps 30 to 35
     --
     -- These traps are sent when the status of the lsm changes.

     acsTrpLsmStateOnline TRAP-TYPE
     ENTERPRISE acsTapeMonitor
     VARIABLES { acsLsmId,
                         acsLsmState,
                         acsLsmStatus,
                         acsLsmAcsIndex,
                         acsLsmIndex
                          }
     DESCRIPTION
            " This trap is sent when Lsm state change to Online state.

            "
    ::= 30

     acsTrpLsmStateOffline TRAP-TYPE
     ENTERPRISE acsTapeMonitor
     VARIABLES { acsLsmId,
                         acsLsmState,
                         acsLsmStatus,
                         acsLsmAcsIndex,
                         acsLsmIndex
                          }
     DESCRIPTION
            " This trap is sent when Lsm state change to Offline state.

            "
```

```
::= 31

 acsTrpLsmStateOfflinePending TRAP-TYPE
 ENTERPRISE acsTapeMonitor
 VARIABLES { acsLsmId,
                    acsLsmState,
                    acsLsmStatus,
                    acsLsmAcsIndex,
                    acsLsmIndex
                     }
 DESCRIPTION
       " This trap is sent when Lsm state change to Offline Pending state.

       "
::= 32

 acsTrpLsmStateRecovery TRAP-TYPE
 ENTERPRISE acsTapeMonitor
 VARIABLES { acsLsmId,
                    acsLsmState,
                    acsLsmStatus,
                    acsLsmAcsIndex,
                    acsLsmIndex
                     }
 DESCRIPTION
       " This trap is sent when Lsm state change to Recovery state.

       "
::= 33

 acsTrpLsmStateDiagnostic TRAP-TYPE
 ENTERPRISE acsTapeMonitor
 VARIABLES { acsLsmId,
                    acsLsmState,
                    acsLsmStatus,
                    acsLsmAcsIndex,
                    acsLsmIndex
                     }
 DESCRIPTION
       " This trap is sent when Lsm state change to Diagnostic state.

       "
::= 34
```

```
  -- Drive status traps : Traps 50 to 52
  --
  -- These traps are sent when the status of a drive changes

acsTrpDriveStateOnline TRAP-TYPE
  ENTERPRISE acsTapeMonitor
  VARIABLES { acsDriveId,
                      acsDriveState,
                      acsDriveStatus,
                      acsDriveAcsIndex,
                      acsDriveLsmIndex,
                      acsDriveIndex
                }
  DESCRIPTION
         " This trap is sent when a drive state change to online."
 ::= 50

acsTrpDriveStateOffline TRAP-TYPE
  ENTERPRISE acsTapeMonitor
  VARIABLES { acsDriveId,
                      acsDriveState,
                      acsDriveStatus,
                      acsDriveAcsIndex,
                      acsDriveLsmIndex,
                      acsDriveIndex
                }
  DESCRIPTION
         " This trap is sent when a drive state change to offline."
 ::= 51

acsTrpDriveStateDiagnostic TRAP-TYPE
  ENTERPRISE acsTapeMonitor
  VARIABLES { acsDriveId,
                      acsDriveState,
                      acsDriveStatus,
                      acsDriveAcsIndex,
                      acsDriveLsmIndex,
                      acsDriveIndex
                }
  DESCRIPTION
         " This trap is sent when a drive state change to diagnostic."
 ::= 52
```

```
  -- CAP status traps : Traps 60 to 64
  --
  -- These traps are sent when the status of a CAP changes

acsTrpCapStateOnline TRAP-TYPE
  ENTERPRISE acsTapeMonitor
  VARIABLES { acsCapId,
                      acsCapState,
                      acsCapStatus,
                      acsCapPriority,
                      acsCapAcsIndex,
                      acsCapLsmIndex,
                      acsCapIndex
                }
  DESCRIPTION
         " This trap is sent when a cap become online."
 ::= 60

acsTrpCapStateOffline TRAP-TYPE
  ENTERPRISE acsTapeMonitor
  VARIABLES { acsCapId,
                      acsCapState,
                      acsCapStatus,
                      acsCapPriority,
                      acsCapAcsIndex,
                      acsCapLsmIndex,
                      acsCapIndex
                }
  DESCRIPTION
         " This trap is sent when a cap state change to offline."
 ::= 61

acsTrpCapStateOfflinePending TRAP-TYPE
  ENTERPRISE acsTapeMonitor
  VARIABLES { acsCapId,
                      acsCapState,
                      acsCapStatus,
                      acsCapPriority,
                      acsCapAcsIndex,
                      acsCapLsmIndex,
                      acsCapIndex
                }
  DESCRIPTION
         " This trap is sent when a cap state change to offline pending."
```

```
 ::= 62

acsTrpCapStateRecovery TRAP-TYPE
  ENTERPRISE acsTapeMonitor
  VARIABLES { acsCapId,
                      acsCapState,
                      acsCapStatus,
                      acsCapPriority,
                      acsCapAcsIndex,
                      acsCapLsmIndex,
                      acsCapIndex
                }
  DESCRIPTION
         " This trap is sent when a cap state change to recovery."
 ::= 63

acsTrpCapStateDiagnostic TRAP-TYPE
  ENTERPRISE acsTapeMonitor
  VARIABLES { acsCapId,
                      acsCapState,
                      acsCapStatus,
                      acsCapPriority,
                      acsCapAcsIndex,
                      acsCapLsmIndex,
                      acsCapIndex
                }
  DESCRIPTION
         " This trap is sent when a cap state change to diagnostic."
 ::= 64


  -- ------------------------------------------
  --
  --     acs library hardware sub tree
  --
  -- ------------------------------------------

  --
  --
  --     acs sub tree
  --
  --


  acsAcs OBJECT IDENTIFIER ::= { acsHardware 1 }
```

```
acsAcsCount OBJECT-TYPE
       SYNTAX  INTEGER
       ACCESS  read-only
       STATUS  mandatory
       DESCRIPTION
              "Count of the ACS in the ACS library table"
       ::= { acsAcs 1 }

acsAcsTable OBJECT-TYPE
       SYNTAX  SEQUENCE OF AcsAcsEntry
       ACCESS  not-accessible
       STATUS  mandatory
       DESCRIPTION
              "This is a table of ACS library detected through ACSLS"
::= {  acsAcs 2  }

acsAcsEntry   OBJECT-TYPE
       SYNTAX  AcsAcsEntry
       ACCESS  not-accessible
       STATUS  mandatory
       DESCRIPTION
              "An entry in the acs library table"
       INDEX { acsAcsIndex }
::= { acsAcsTable 1  }

AcsAcsEntry ::=
       SEQUENCE {
              acsAcsIndex
                     INTEGER,
              acsAcsId
                     DisplayString,
              acsAcsState
                     INTEGER (1..5),
              acsAcsFreeCellsCount
                     INTEGER,
              acsAcsLsmCount
                     INTEGER
              }

acsAcsIndex   OBJECT-TYPE
       SYNTAX  INTEGER
       ACCESS  read-only
       STATUS  mandatory
       DESCRIPTION
```

```
            "Integer index of the acs"
::= { acsAcsEntry 1  }

acsAcsId   OBJECT-TYPE
       SYNTAX  DisplayString
       ACCESS  read-only
       STATUS  mandatory
       DESCRIPTION
              "ACSLS API  Acs identifier : acsId "
::= { acsAcsEntry 2  }

acsAcsState   OBJECT-TYPE
       SYNTAX  INTEGER {
                     diagnostic(1),
                     online (2),
                     offline (3),
                     offpending(4),
                     recovery (5)
                     }
       ACCESS  read-only
       STATUS  mandatory
       DESCRIPTION
              "The state of the Acs :
                     STATE_DIAGNOSTIC
                     STATE_ONLINE
                     STATE_OFFLINE
                     STATE_OFFLINE_PENDING
                     STATE_RECOVERY
              "
::= { acsAcsEntry 3  }

acsAcsFreeCellsCount   OBJECT-TYPE
       SYNTAX  INTEGER
       ACCESS  read-only
       STATUS  mandatory
       DESCRIPTION
              "Count of the free cells of this acs
                           "
::= { acsAcsEntry 4  }

acsAcsLsmCount   OBJECT-TYPE
       SYNTAX  INTEGER
       ACCESS  read-only
       STATUS  mandatory
       DESCRIPTION
```

```
                "Count of LSM forming the ACS"
::= { acsAcsEntry 5  }


acsLsm OBJECT IDENTIFIER ::= { acsHardware 2 }

acsLsmCount OBJECT-TYPE
       SYNTAX  INTEGER
       ACCESS  read-only
       STATUS  mandatory
       DESCRIPTION
              "Count of the LSM in the LSM library table.
              (sum of all the LSM in the ACS library)
                     "
       ::= { acsLsm 1 }

acsLsmTable OBJECT-TYPE
       SYNTAX  SEQUENCE OF AcsLsmEntry
       ACCESS  not-accessible
       STATUS  mandatory
       DESCRIPTION
              "This is a table of LSM detected through ACSLS"
::= {  acsLsm 2  }

acsLsmEntry   OBJECT-TYPE
       SYNTAX  AcsLsmEntry
       ACCESS  not-accessible
       STATUS  mandatory
       DESCRIPTION
              "An entry in the LSM  table"
       INDEX { acsLsmIndex,acsLsmAcsIndex }
::= { acsLsmTable 1  }

AcsLsmEntry ::=
       SEQUENCE {
              acsLsmAcsIndex
                     INTEGER,
              acsLsmIndex
                     INTEGER,
              acsLsmId
                     DisplayString,
              acsLsmState
                     INTEGER (1..5),
              acsLsmStatus
                     INTEGER (1..6),
```

```
                acsLsmFreeCellsCount
                        INTEGER,
                acsLsmCapCount
                        INTEGER,
                acsLsmDriveCount
                        INTEGER
                }

acsLsmAcsIndex   OBJECT-TYPE
        SYNTAX  INTEGER
        ACCESS  read-only
        STATUS  mandatory
        DESCRIPTION
                "Integer index of the acs (where this lsm is) "
::= { acsLsmEntry 1  }

acsLsmIndex   OBJECT-TYPE
        SYNTAX  INTEGER
        ACCESS  read-only
        STATUS  mandatory
        DESCRIPTION
                "Integer index of the lsm"
::= { acsLsmEntry 2  }

acsLsmId   OBJECT-TYPE
        SYNTAX  DisplayString
        ACCESS  read-only
        STATUS  mandatory
        DESCRIPTION
                "ACSLS Lsm identifier :acsId,lsmId"
::= { acsLsmEntry 3  }

acsLsmStatus   OBJECT-TYPE
        SYNTAX  INTEGER {
                        audit-act(1),
                        cap-available (2),
                        eject-act (3),
                        enter-act(4),
                        acs-not-in-lib(5),
                        lsm-not-in-lib(6)
                        }
        ACCESS  read-only
        STATUS  mandatory
        DESCRIPTION
                "The status of the Lsm."
```

```
::= { acsLsmEntry 4  }

acsLsmState   OBJECT-TYPE
       SYNTAX  INTEGER {
                     diagnostic (1),
                     online (2),
                     offline (3),
                     offpending (4),
                     recovery (5)
                     }
       ACCESS  read-only
       STATUS  mandatory
       DESCRIPTION
              "The state of the lsm :
                     STATE_DIAGNOSTIC
                     STATE_ONLINE
                     STATE_OFFLINE
                     STATE_OFFLINE_PENDING
                     STATE_RECOVERY
              "
::= { acsLsmEntry 5  }

acsLsmFreeCellsCount   OBJECT-TYPE
       SYNTAX  INTEGER
       ACCESS  read-only
       STATUS  mandatory
       DESCRIPTION
              "Count of the free cells of this lsm
                           "
::= { acsLsmEntry 6  }

acsLsmCapCount   OBJECT-TYPE
       SYNTAX  INTEGER
       ACCESS  read-only
       STATUS  mandatory
       DESCRIPTION
              "Count of Caps within the lsm"
::= { acsLsmEntry 7  }

acsLsmDriveCount   OBJECT-TYPE
       SYNTAX  INTEGER
       ACCESS  read-only
       STATUS  mandatory
       DESCRIPTION
              "Count of Drives within the lsm"
```

```
::= { acsLsmEntry 8  }



acsDrive OBJECT IDENTIFIER ::= { acsHardware 3 }

acsDriveCount OBJECT-TYPE
       SYNTAX  INTEGER
       ACCESS  read-only
       STATUS  mandatory
       DESCRIPTION
              "Count of the Drive in the Drive library table.
              (sum of all the drive within the whole ACS Library)
                     "
       ::= { acsDrive 1 }

acsDriveTable OBJECT-TYPE
       SYNTAX  SEQUENCE OF AcsDriveEntry
       ACCESS  not-accessible
       STATUS  mandatory
       DESCRIPTION
              "This is a table of Drives detected through ACSLS"
::= {  acsDrive 2  }

acsDriveEntry   OBJECT-TYPE
       SYNTAX  AcsDriveEntry
       ACCESS  not-accessible
       STATUS  mandatory
       DESCRIPTION
              "An entry in the Drive  table"
       INDEX { acsDriveIndex,acsDriveLsmIndex,acsDriveAcsIndex }
::= { acsDriveTable 1  }

AcsDriveEntry ::=
       SEQUENCE {
              acsDriveAcsIndex
                     INTEGER,
              acsDriveLsmIndex
                     INTEGER,
              acsDriveIndex
                     INTEGER,
              acsDriveId
                     DisplayString,
              acsDriveStatus
                     INTEGER (1..4),
```

```
                acsDriveState
                        INTEGER (1..3),
                acsDriveTypeText
                        DisplayString,
                acsDriveVolLabel
                        DisplayString,
                acsDriveVolTypeText
                        DisplayString
                }

acsDriveAcsIndex   OBJECT-TYPE
        SYNTAX  INTEGER
        ACCESS  read-only
        STATUS  mandatory
        DESCRIPTION
                "Integer index of the acs (where this drive is) "
::= { acsDriveEntry 1  }

acsDriveLsmIndex   OBJECT-TYPE
        SYNTAX  INTEGER
        ACCESS  read-only
        STATUS  mandatory
        DESCRIPTION
                "Integer index of the lsm (where this drive is ) "
::= { acsDriveEntry 2  }

acsDriveIndex   OBJECT-TYPE
        SYNTAX  INTEGER
        ACCESS  read-only
        STATUS  mandatory
        DESCRIPTION
                "Integer index of the drive"
::= { acsDriveEntry 3  }

acsDriveId   OBJECT-TYPE
        SYNTAX  DisplayString
        ACCESS  read-only
        STATUS  mandatory
        DESCRIPTION
                "ACSLS Drive identifier : acsId,lsmId,panelId,driveId"
::= { acsDriveEntry 4  }

acsDriveStatus   OBJECT-TYPE
        SYNTAX  INTEGER {
                        available(1),
```

```
                    drive-in-use(2),
                    drive-not-in-lib(3),
                    drive-not-in-lsm(4)
                    }
        ACCESS  read-only
        STATUS  mandatory
        DESCRIPTION
                "The status of the drive .
                "
::= { acsDriveEntry 5  }

acsDriveState   OBJECT-TYPE
        SYNTAX  INTEGER {
                    diagnostic(1),
                    online (2),
                    offline (3)
                    }
        ACCESS  read-only
        STATUS  mandatory
        DESCRIPTION
                "The state of the drive :
                    STATE_DIAGNOSTIC
                    STATE_ONLINE
                    STATE_OFFLINE
                "
::= { acsDriveEntry 6  }

acsDriveVolLabelOBJECT-TYPE
        SYNTAX  DisplayString
        ACCESS  read-only
        STATUS  mandatory
        DESCRIPTION
                "The label of the cartridge present into the drive,
                '------' if the drive is empty"
        ::= { acsDriveEntry 7  }

acsDriveTypeTextOBJECT-TYPE
        SYNTAX  DisplayString
        ACCESS  read-only
        STATUS  mandatory
        DESCRIPTION
                "Textual type of the drive"
        ::= { acsDriveEntry 8  }

acsDriveVolTypeTextOBJECT-TYPE
```

```
        SYNTAX  DisplayString
        ACCESS  read-only
        STATUS  mandatory
        DESCRIPTION
                "Textual type of the volume present into the drive"
        ::= { acsDriveEntry 9  }




acsCap OBJECT IDENTIFIER ::= { acsHardware 4 }


acsCapCount OBJECT-TYPE
        SYNTAX  INTEGER
        ACCESS  read-only
        STATUS  mandatory
        DESCRIPTION
                "Count of the Cap in the Cap library table.
                (sum of all the drive within the whole ACS Library)
                        "
        ::= { acsCap 1 }

acsCapTable OBJECT-TYPE
        SYNTAX  SEQUENCE OF AcsCapEntry
        ACCESS  not-accessible
        STATUS  mandatory
        DESCRIPTION
                "This is a table of Caps detected through ACSLS"
::= {  acsCap 2  }

acsCapEntry   OBJECT-TYPE
        SYNTAX  AcsCapEntry
        ACCESS  not-accessible
        STATUS  mandatory
        DESCRIPTION
                "An entry in the Cap  table"
        INDEX { acsCapIndex,acsCapLsmIndex,acsCapAcsIndex }
::= { acsCapTable 1  }

AcsCapEntry ::=
        SEQUENCE {
                acsCapAcsIndex
```

```
                INTEGER,
            acsCapLsmIndex
                INTEGER,
            acsCapIndex
                INTEGER,
            acsCapId
                DisplayString,
            acsCapStatus
                INTEGER (1..5),
            acsCapState
                INTEGER (1..5),
            acsCapPriority
                INTEGER,
            acsCapSize
                INTEGER,
            acsCapMode
                INTEGER (1..3)
            }

acsCapAcsIndex   OBJECT-TYPE
       SYNTAX  INTEGER
       ACCESS  read-only
       STATUS  mandatory
       DESCRIPTION
              "Integer index of the acs (where this cap is) "
::= { acsCapEntry 1  }

acsCapLsmIndex   OBJECT-TYPE
       SYNTAX  INTEGER
       ACCESS  read-only
       STATUS  mandatory
       DESCRIPTION
              "Integer index of the lsm (where this cap is ) "
::= { acsCapEntry 2  }

acsCapIndex   OBJECT-TYPE
       SYNTAX  INTEGER
       ACCESS  read-only
       STATUS  mandatory
       DESCRIPTION
              "Integer index of the cap"
::= { acsCapEntry 3  }

acsCapId   OBJECT-TYPE
       SYNTAX  DisplayString
```

```
        ACCESS  read-only
        STATUS  mandatory
        DESCRIPTION
                "ACSLS Cap identifier : acsId,lsmId,capId"
::= { acsCapEntry 4  }

acsCapStatus   OBJECT-TYPE
        SYNTAX  INTEGER {
                        audit-act(1),
                        available(2),
                        eject-act(3),
                        enter-act(4),
                        cap-not-in-lib(5)
                       }
        ACCESS  read-only
        STATUS  mandatory
        DESCRIPTION
                "The status of the cap .
                "
::= { acsCapEntry 5  }

acsCapState   OBJECT-TYPE
        SYNTAX  INTEGER {
                      diagnostic(1),
                      online (2),
                      offline (3),
                      offpending(4),
                      recovery (5)
                      }
        ACCESS  read-only
        STATUS  mandatory
        DESCRIPTION
                "The state of the cap :
                      STATE_DIAGNOSTIC
                      STATE_ONLINE
                      STATE_OFFLINE
                      STATE_OFFLINE_PENDING
                      STATE_RECOVERY
                "
::= { acsCapEntry 6  }

acsCapPriority OBJECT-TYPE
        SYNTAX  INTEGER
        ACCESS  read-only
        STATUS  mandatory
```

```
        DESCRIPTION
                "Count of cell of priority assigned to the cap"
        ::= { acsCapEntry 7  }

    acsCapSize OBJECT-TYPE
        SYNTAX  INTEGER
        ACCESS  read-only
        STATUS  mandatory
        DESCRIPTION
                "Count of cell of the cap"
        ::= { acsCapEntry 8  }

    acsCapModeOBJECT-TYPE
        SYNTAX  INTEGER {
                unknown(1),
                manual (2),
                automatic (3)
        }
        ACCESS  read-only
        STATUS  mandatory
        DESCRIPTION
                "Report whether the CAP is in manual mode or not"
        ::= { acsCapEntry 9  }


END
```