StorageTek ACSLS SNMP Agent

User's Guide

Version 2.0

First Edition

312555301

Proprietary Information Statement

The information in this document is confidential and proprietary to Storage Technology Corporation and may be used only under the terms of the product license or nondisclosure agreement. The information in this document, including any associated software program, may not be disclosed, disseminated, or distributed in any manner without the written consent of Storage Technology Corporation.

Limitations on Warranties and Liability

This document neither extends nor creates warranties of any nature, expressed or implied. Storage Technology Corporation cannot accept any responsibility for your use of the information in this document or for your use of any associated software program.You are responsible for backing up your data. You should be careful to ensure that your use of the information complies with all applicable laws, rules, and regulations of the jurisdictions in which it is used.

Warning: No part or portion of this document may be reproduced in any manner or in any form without the written permission of Storage Technology Corporation.

Restricted Rights

Use, duplication, or disclosure by the U.S. Government is subject to restrictions as set forth in subparagraph (c) (1) (ii) of the Rights in Technical Data and Computer Software clause at DFARS 252.227─7013 or subparagraphs (c) (1) and (2) of the Commercial Computer Software — Restricted Rights at 48 CFR 52.227─19, as applicable.

Trademarks

StorageTek is a registered trademark of Storage Technology Corporation.

Other product names mentioned in this manual may be trademarks. They are used for identification purposes only.

First Edition: August 2003
128805

This edition applies to StorageTek ACSLS SNMP Agent - User's Guide Release 2.0. Information in this publication is subject to change. Comments concerning the contents of this manual should be directed to:

StorageTek European Operations
Manager, Information Development
1 rond-point du Général Eisenhower
B.P. 1369, 31106 Toulouse Cedex 1
France

Document Effectivity

EC Number	Date	Doc Kit Number	Edition Type	Effectivity
128805	August 2003	_____	Ist Edition	This document applies to StorageTek ACSLS SNMP Agent, Version 2.0

Table of Contents

Chapter 1. Overview

About the StorageTek ACSLS SNMP Agent

The ACSLS SNMP agent 2.0 monitors tape storage libraries under ACSLS control. This agent runs on the host which is running the ACSLS 7.0 software. It provides ACSLS queried information on the ACS(s) and it's internal components, such as LSMs, CAPs, and drives.

The agent regularly queries the libraries through the ACSLS server and sends asynchronous messages (SNMP Traps) whenever changes are detected in the status of a library or any of its components.

This document provides hints on how to use the agent from a management application/framework (e.g. CA Unicenter, IBM Tivoli NetView, HP OpenView).

Such a management application sends SNMP GET/GET_NEXT requests, retrieves information from the agents, listens to incoming traps and reacts accordingly (e.g. displays the events, sends E-mails or pages the administrators).

More information on SNMP can be found at

`http://www.simple-times.org/index.html`

Intended Users

This document is intended to provide instructions to use the ACSLS SNMP agent.

Prerequisistes

This document does not provide support or expertise on SNMP. It is assumed that the reader and the person that will use the agent is familiar with SNMP.

A Few Words About the Management Information Base (MIB)

A MIB describes the information that is available from the SNMP agent. As the SNMP agent evolves to provide more information from one release to another, the MIB has also been modified to reflect these changes. End-users should check whether the release of the agent that runs on a host is consistent with the associated MIB to ensure that SNMP requests are performed on nodes that are supported by the agent.

The ACSLS agent MIB information is provided in this document. For details, see *Chapter 6. The ACSLS agent MIB.*

Note: During Agent installation, the MIB file is also installed in the Agent directory for easy access. This is particularly handy for users who need to compile it and use it in their management applications.

Supported Platforms

The ACSLS SNMP Agent 2.0 is available for the following operating systems:

Platform
Solaris 8 (32/64 bits)
Solaris 9 (32/64 bits)
AIX 4.3.3
AIX 5.1
Linux Red Hat 8.0 using net-snmp-5.0.6

The behavior of the Agent is the same in each of the above platforms, and this document applies to all of them except for installation and start/stop operations. There are also some differences related to SNMP configurations since the technology used to develop the agent is not the same from one platform to another.

Where to Send SNMP Queries

Hosts Considerations

Before querying the agent, the management application must have a list of the IP addresses where queries should be sent. It is also possible to configure the agent so that it reports all unsolicited events (i.e. SNMP traps) back to the management application, in which case it may not be necessary to establish up-front a list of IP addresses. This particuliar aspect is developed in *Chapter 4. Managing SNMP Traps Generated by the Agent*.

Time-out Considerations

The ACSLS agent gathers the information through the ACSLS server. The MIB information provided can be up to 60 seconds old (default polling loop rate is 60 seconds).

Chapter 2. Fundamental Agent Nodes

Assuming that the IP addresses to be used for queries are identified, the simplest way to detect the agent is to query it.

Agent Status

The node contains the status of the agent:

`AcsAgtStatus.0`

Ultimately, a `running` response (integer value `2`) indicates that the Agent is operational and has finished the initialization process. This means that the ACSLS configuration running has been detected by the agent, and data is made available for queries.

Agent Release

The release of the agent can be used as an indication on the availability of the nodes supported by the agent. To be on the safe side, avoid querying nodes that are not implemented and getting NO_SUCH_NAME responses, it is recommended to query the release of the agent once the management application has queried the status node.

`AcsAgtRelease.0`

Agent Boot Date

The agent reports its boot time with this node:

`AcsAgtBootDate.0`

This information is essential since the data reported by the agent may change from one boot to another. The following text gives an explanation of this.

When the agent starts, it builds its internal model of libraries, drives and CAPs as they are detected. This is done once at boot time. If the hardware configuration changes, it is assumed that the agent was stopped and restarted. The fact that the reported boot date changes lets the management application know that a library or a drive may have been added, deleted, or replaced.

When a management application is to update the information previously retrieved from the agent, it is highly recommended to check this node and compare its

values. On the management side, this will avoid updating a library status with the wrong value.

> **Note:** At the end of an agent's initialization, a Start trap is sent indicating the new boot date OID.

Chapter 3. Information Reported by the ACSLS Agent

The information provided by the ACSLS Agent is strictly identical to the ACSLS Server configuration. The information reported reflects directly the ACSLS view of the hardware.

ACSLS Agent Specific Behavior

The ACSLS agent waits for the start of the CSI component of the ACSLS Server to start.

When the ACSLS connection is broken, the ACSLS agent will purge its MIB table entries (ACS, LSM, Drive, and CAP counts will be equal to `0`). The agent will restart automatically when a new ACSLS connection is detected.

ACS Information

ACS Table

The agent provides an ACS table `acsAcsTable`. This table gives information on all ACSs described on the ACSLS server host running the agent. The information retrieved will concern ACSs exclusively as they were configured when the agent was started, and reflect their conditions at that time.

This table provides the number of entries corresponding to the value returned for the `acsAcsCount.0` node. The management application can either perform as many GET_NEXT requests as possible until the returned OID is not in the scope of the table, or perform as many GET requests as the value returned for `acsAcsCount.0`.

The ACS entry count skips no number in the table. The entry index starts at 1, and ends at the value of `acsAcsCount.0`

Static information Vs. Dynamic information

Part of the information reported for the ACS is static and part of it is dynamic.

Static information will not change until the agent is stopped and restarted. This information includes the ACS index, the ACS ID and the LSM count.

Dynamic information can change at any time and may be queried whenever needed by a management application. This information includes the ACS state and the ACS free cells count.

When is an ACS Not Detected by the ACSLS Agent?

When the agent is started, it will attempt connection with the ACSLS server to retrieve the ACSLS configuration. The agent will fail to start if it cannot connect to the ACSLS Server (using the ACSAPI connection).

If an ACS is not detected, the associated LSMs, drives and CAPs will not be reported either.

LSM Information

LSM Table

The agent provides an LSM table `acsLsmTable`. This table will give information on all the LSMs described on the ACSLS server host running the agent. The information retrieved will concern LSMs exclusively as they were configured when the agent was started and reflect their conditions at that time.

This table provides as many entries as the value returned for the `acsLsmCount.0` node. The management application can either perform as many GET_NEXT requests as possible until the returned OID is not in the scope of the table, or perform as many GET requests as the value returned for acsLsmCount.0.

The LSM entry count skips no number in the table. The entry index starts at 1, and ends at the value of `acsLsmCount.0`

LSMs have two indexes, the ACS index and the LSM index. The ownership of a LSM is derived from the ACS index of the LSM.

Static Information Vs. Dynamic Information

Part of the information reported for the LSM is static, and part of it is dynamic.

Static information will not change until the agent is stopped and restarted. This information will include the LSM and CAP indexes, the LSM and CAP IDs, and the LSM and CAP counts.

Dynamic information can change at any time and may be queried as often as needed by a management application. This information includes LSM state, LSM status and LSM free cells count.

Drive Information

General Information

The drive table provides information about the drives configured in ACSLS during agent initialization. Drives have three indexes, the ACS, LSM, and drive indexes. The ownership of a drive is derived from the drive's ACS and LSM indexes.

All nodes report dynamic information that can change from one query to another.

When Is a Drive Not Detected by the ACSLS Agent?

The drive configuration of a LSM is detected when the agent is started. The agent will report on the drives that are described in the ACSLS Server configuration.

CAP Information

General Information

The agent can report on the cap size, priority, (manual/automatic) mode , state and status, and ACSLS CAP ID. CAPs have three indexes: the ACS, LSM, and CAP

indexes. The ownership of a CAP is derived from the ACS and LSM indexes of the CAP entry.

CAP indexes, and ID information are static. All other nodes report dynamic information.

When is a CAP Not Detected by the ACSLS Agent?

The CAP configuration of a LSM is detected when the agent is started. The agent will report on the CAPs described in the ACSLS Server configuration.

Chapter 4. Managing SNMP Traps Generated by the Agent

Agent Start Trap

Description

Agents will send a trap each time they are started. Each trap will contain the related boot date for information purposes.

This type of trap can be used by a management application to resynchronize its own data model with the information available from the agent. As discussed in the *Agent Boot Date* section of Chapter 2, the agent may have been restarted because of a hardware configuration change on the agent side (e.g. adding or removing a library).

Trap 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"
```

Status Traps

The agent regularly checks for status changes in the ACSLS database state.

Description

These ACSLS agent traps are numbered 21 through 64. There are as many traps defined in the MIB as there are possible statuses returned for a ACS, LSM , drive, or CAP for an ACSLS agent.

The unique number assigned to a trap is determined by the type and status of the component involved.

To facilitate component identification among the collection of components detected by the agent, more information is provided in the trap. This information can be displayed by a management application in an event console.

For ACS, this information includes the ACS state, index (which is the index in the ACS table), and ID.

For LSMs, this information includes the LSM state and status, the ACS index (which is the first index in the LSM table), the LSM index (which is the second index in the LSM table), and the LSM ID.

For drives, this information includes the drive state and status, the ACS index (which is the first index in the drive table), the LSM index (which is the second index in the drive table), the drive index (which is the third index in the drive table), and the drive ID.

For CAPs, this information includes the CAP state and status, the CAP priority, the ACS index (which is the first index in the CAP table), the LSM index (which is the second index in the CAP table), the CAP index (which is the third index in the CAP table), and the CAP ID.

Trap Sample

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
```

Polling Rate of the Agent

The polling rate is the amount of time, in seconds, between two periodic sets of queries performed by the agent that checks for dynamic information.

The polling rate may be changed by setting the value of the `acsTrpCurPollingRate.0` node for the ACSLS agent. The polling rate cannot be set to a value lower than a minimum available by querying `acsTrpMinPollingRate.0` for the ACSLS agent.

The new value will be taken into account as soon as it is set.

Chapter 5. Tools of interest

Several tools can be used to perform SNMP queries.

Management Frameworks

Obviously, network management frameworks such as HP OpenView and IBM Tivoli NetView come with a graphical MIB browser.

They usually provide some SNMP collection features to automatically and regularly query some SNMP nodes, and display the results as graphics.

Note: StorageTek Framework Library Monitor 4.2 is a StorageTek module that is integrated in management frameworks (CA Unicenter, HP OpenView, IBM Tivoli NetView). It produces and displays graphic views of the information provided by the ACSLS Agent and forwards its traps to the management framework.

Windows NT Resource Kit

The Windows NT resource kit provides a utility called “snmputil”.

```
c:\ntreskit> snmputil get my_host public .1.3.6.1.4.1.1211.1.11.1.2.0
```

returns the status of the agent running on `my_host` for instance.

Note: The `snmputil` command expects the OIDs to begin with a dot.

`snmputil` also provides a `walk`. Below is an example of the `walk` command usage and output:

```
C:\NTRESKIT>snmputil walk my_host public .1.3.6.1.4.1.1211.1.11.1
Variable = .iso.org.dod.internet.private.enterprises.1211.1.11.1.1.0
Value    = OCTET STRING - 2.0

Variable = .iso.org.dod.internet.private.enterprises.1211.1.11.1.2.0
Value    = INTEGER - 2

Variable = .iso.org.dod.internet.private.enterprises.1211.1.11.1.3.0
Value    = OCTET STRING - 2001-04-04T09:41:28

Variable = .iso.org.dod.internet.private.enterprises.1211.1.11.1.4.0
Value    = OCTET STRING -

End of MIB subtree.
```

AIX snmpinfo

AIX provides the `/usr/sbin/snmpinfo` command (equivalent to Windows NT's **`snmputil`** command) which allows to perform SNMP GET and SET requests.

Requesting the status of the Agent would look like:

```
/usr/sbin/snmpinfo –m get –h my_host 1.3.6.1.4.1.1211.1.11.1.2.0
```

Net-SNMP utils on Red Hat Linux

The Net-SNMP-utils package provided by Red Hat Linux includes a number of useful commands.

Requesting the status of the Agent would look like:

```
snmpget -v 1 -c public my host .1.3.6.1.4.1.1211.1.11.1.2.0
```

Note: A similar package exists on Solaris platforms but it is not installed by default.

Chapter 6. The ACSLS 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;

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


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

  acsAgentOBJECT IDENTIFIER ::= { acsTapeMonitor 1 }
  acsTrapOBJECT 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
     SYNTAXDisplayString
     ACCESS read-only
     STATUS mandatory
     DESCRIPTION
       "The release of the agent. Format is %d.%d"
     ::= { acsAgent 1 }


   acsAgtStatus OBJECT-TYPE
     SYNTAXINTEGER
       {
       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 cor-
rectly
       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
     SYNTAXDisplayString
     ACCESS read-only
     STATUS mandatory
     DESCRIPTION
       "The date & time when the agent started.
       Format is YYYY-MM-DDThh:mm"
     ::= { acsAgent 3 }

   acsAgtUrl  OBJECT-TYPE
     SYNTAXDisplayString
     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
    STATUSmandatory
    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
    STATUSmandatory
    DESCRIPTION
      "Current polling rate in seconde. The value could
      not be set under acsTrpMinPollingRate"
    ::= { acsTrap 2 }


acsTrpMsg OBJECT-TYPE
    SYNTAX DisplayString
    ACCESS read-only
    STATUSmandatory
    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 Enter-
```

```
prise
       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 diag-
nostic."
    ::= 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 recov-
ery."
    ::= 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 diagnos-
tic."
    ::= 64


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

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


   acsAcsOBJECT 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,pane-
lId,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
```