Promentum™ MPCMM0001 Chassis Management Module and Promentum™ MPCMM0002 Chassis Management Module

SHM/OAM API Reference Manual

May 2008

Document Number: 007-03265-0000

Please visit the RadiSys Web site at www.radisys.com for product information and other resources. Downloads (manuals, release notes, software, etc.) are available via the Technical Support Library product lings at www.radisys.com/support or the product pages at www.radisys.com/products.

I2C is a two-wire communications bus/protocol developed by Philips. SMBus is a subset of the I2C bus/protocol and was developed by Intel. Implementations of the I2C bus/protocol may require licenses from various entities, including Philips Electronics N.V. and North American Philips Corporation.

Contents

Contents

Contents

Contents

Contents

Contents

Revision History

Revision Number	Description	Revision Date
0.1	ALPHA release.	August 2006
0.2	SHM API part removed.	September 2006
0.3	Updated for BETA release	October 2006
0.4	Added shmOamPmProcessIdArrayGet function.	November 2006
0.41	Update out of service cause definition descriptions in HA section.	November 2006
1.0	SRA Release.	December 2006
1.01	Section “High Availability Utility API” removed.	January 2007
1.02	Information on RPC program and function numbers added.	January 2007
1.03	Added new functions: shmOamSensorFruIdLookup shmOamSensorTypeLookup shmOamLunsLookup	March 2007
-0000	Document and part number converted to RadiSys format.	May 2008

§

1 Introduction

1.1 Overview

This document describes the Remote Shelf Management and OAM programming interface used to configure and monitor CMM capabilities and features. The functions described in this document are organized by functionality.

CMM supports Remote Shelf Management and OAM interface. The Shelf Management interface exposes functions that correspond to IPMI commands defined in IPMI / PICMG specifications. The OAM interface defines new functions that cover functionality not defined in PICMG specification, e.g. firmware upgrade, diagnostics. The System Manager application calls Shelf Management and OAM API functions locally from client library. The calls are transported to remote CMM using standard RPC protocol defined in “RFC1057”. The RPC messages are transported over LAN using RMCP packets. The OEM payload mechanism defined in RMCP+ is used to encapsulate RPC into RMCP. This transport option makes it possible to utilize security features defined in RMCP+ which are not present in the RPC protocol itself.

1.2 Shelf Management and OAM API Client Library

The Shelf Management and OAM API client library is a dynamic library, written in the C language. The client library is linked with the System Management application and provides support for establishing a session to the Shelf Management and OAM API Server running on CMM and invoking Shelf Management and OAM functions remotely.

§

2 SHM API

2.1 Data Types definitions

2.1.1 General

2.1.1.1 Boolean type

ShmBoolT

Syntax:

```
typedef enum {
          SHM_FALSE              = 0,
          SHM_TRUE               = 1
} ShmBoolT;
```

Members:

SHM_FALSE

FALSE.

SHM_TRUE

TRUE.

2.1.1.2 SHM API return type

ShmErrorT

Syntax:

```
typedef   unsigned int          ShmErrorT;
```

Description:

Error code can be reported due to IPMI errors, OS errors, transport protocol errors for requests this code handles. The MSB in ShmErrorT defines the error domain. Depending on the error domain remaining 3 bytes carry domain specific error information. The successful status indication is defined as common value for all error domains, it is defined as SHM_CC_OK.

2.1.1.3 IPMI LAN IPv4 address type.

ShmIpAddrTSyntax:

```
typedef   unsigned char         ShmIpAddrT[4];
```

2.1.1.4 SHM Error masks

Syntax:

```
#define  SHM_ERR_MASK_DOMAIN                0xFF000000
#define  SHM_ERR_MASK_GENERAL_ERROR_CODE    0x00FFFFFF
#define  SHM_ERR_MASK_IPMI_NETFN            0x00FF0000
#define  SHM_ERR_MASK_IPMI_CMD              0x0000FF00
#define  SHM_ERR_MASK_IPMI_ERROR_CODE       0x000000FF
```

Members:

SHM_ERR_MASK_DOMAIN

SHM_ERR_MASK_GENERAL_ERROR_CODE

SHM_ERR_MASK_IPMI_NETFN

SHM_ERR_MASK_IPMI_CMD

SHM_ERR_MASK_IPMI_ERROR_CODE

Description:

SHM Error masks.

2.1.1.5 API Error Domains

Syntax:

```
#define   SHM_ERR_DOMAIN_GENERAL   0
#define   SHM_ERR_DOMAIN_IPMI      1
```

Members:

SHM_ERR_DOMAIN_GENERAL

General error domain.

SHM_ERR_DOMAIN_IPMI

IPMI error domain.

Description:

API Error Domains.

2.1.1.6 Common API Error Codes

Syntax:

```
#define   SHM_CC_OK              0
```

Members:

SHM_CC_OK

Operation completed successfully. No errors.

Description:

Common API Error Codes.

2.1.1.7 General Domain error completion codes

Syntax:

```
#define  SHM_CC_NULL                          0x01
#define  SHM_CC_RANGE                         0x02
#define  SHM_CC_EMPTY                         0x03
#define  SHM_CC_NO_SUBSCRIPTION               0x04
#define  SHM_CC_ALREADY_SUBSCRIBED            0x05
#define  SHM_CC_NOT_SUPPORTED                 0x06
#define  SHM_CC_NOT_FOUND                     0x07
#define  SHM_CC_BUFFER_EMPTY                  0x08
#define  SHM_CC_INVALID_SESSION_HANDLE        0x09
#define  SHM_CC_INVALID_USERNAME              0x0A
#define  SHM_CC_SESSION_SLOT_FULL             0x0B
#define  SHM_CC_USER_SLOT_FULL                0x0C
#define  SHM_CC_PRIVILEGE_SLOT_FULL           0x0D
#define  SHM_CC_SESSION_SN_OUT_OF_RANGE       0x0E
#define  SHM_CC_INVALID_SESSION_ID            0x0F
#define  SHM_CC_PRIVILEGE_LEVEL_NOT_AVAILABLE 0x10
#define  SHM_CC_PRIVILEGE_LEVEL_EXCEEDED      0x11
#define  SHM_CC_AUTHENTICATION_NOT_DISABLED   0x12
#define  SHM_CC_ACCESS_MODE_NOT_SUPPORTED     0x13
#define  SHM_CC_LOCKED_KEY                    0x14
#define  SHM_CC_INVALID_PASSWORD              0x15
#define  SHM_CC_INVALID_PASSWORD_SIZE         0x16
#define  SHM_CC_INVALID_PARAMETER             0x17
#define  SHM_CC_READ_ONLY                     0x18
#define  SHM_CC_RECORD_LEN_MISMATCH           0x19
#define  SHM_CC_FRU_DEVICE_BUSY               0x1A
#define  SHM_CC_WRITE_PROTECTED_OFFSET        0x1B
#define  SHM_CC_RECORD_CHANGED                0x1C
#define  SHM_CC_DESTINATION_UNAVAILABLE       0x1D
#define  SHM_CC_TIMEOUT                       0x1E
#define  SHM_CC_MISMATCH                      0x1F
#define  SHM_CC_UNAVAILABLE                   0x20
#define  SHM_CC_ALREADY_USED                  0x21
#define  SHM_CC_ILLEGAL_CMD                   0x22
#define  SHM_CC_ILLEGAL_CMD_FOR_HASTATE       0x23
#define  SHM_CC_INVALID_GROUP                 0x24
#define  SHM_CC_INVALID_COUNTER               0x25
#define  SHM_CC_INVALID_THRESHOLD             0x26
#define  SHM_CC_LAST_ENTRY                    0x27
#define  SHM_CC_INVALID_ENTRY_ID              0x28
#define  SHM_CC_INVALID_FILE                  0x29
#define  SHM_CC_TYPE_UNSPECIFIED              0x2A
#define  SHM_CC_UPDATE_IN_PROGRESS            0x2B
#define  SHM_CC_NO_STANDBY                    0x2C
```


```
#define  SHM_CC_STANDBY_STATE_UNKNOWN          0x2D
#define  SHM_CC_OLDER_FW                       0x2E
#define  SHM_CC_CRITICAL_EVENTS                0x2F
#define  SHM_CC_COMM_FAILED                    0x30
#define  SHM_CC_NOT_SYNCED                     0x31
#define  SHM_CC_DIR_NOT_VALID                  0x32
#define  SHM_CC_FILE_NOT_FOUND                 0x33
#define  SHM_CC_DIR_NOT_ALLOWED                0x34
#define  SHM_CC_ZERO_SIZE                      0x35
#define  SHM_CC_INSUFFICENT_PRIVILEGE          0x36
#define  SHM_CC_ERROR                          0xFF
```

Members:

SHM_CC_NULL

null parameter prohibited.

SHM_CC_RANGE

parameter value incorrect.

SHM_CC_EMPTY

no more data to read.

SHM_CC_NO_SUBSCRIPTION

access prohibited, because session is not subscribed.

SHM_CC_ALREADY_SUBSCRIBED

session already subscribed.

SHM_CC_NOT_SUPPORTED

functionality not supported.

SHM_CC_NOT_FOUND

Requested Sensor, data, or record not present..

SHM_CC_BUFFER_EMPTY

Data not available

Data not available (queue/buffer empty).

SHM_CC_INVALID_SESSION_HANDLE

Invalid session handle.

Invalid session handle. The session handle does not match up with any currently active sessions.

SHM_CC_INVALID_USERNAME

Invalid user name.

SHM_CC_SESSION_SLOT_FULL

No session slot available.

No session slot available (BMC cannot accept any more sessions).

SHM_CC_USER_SLOT_FULL

No slot available for given user.

No slot available for given user. (Limit of user sessions allowed under that name has been reached).

SHM_CC_PRIVILEGE_SLOT_FULL

No slot available to support user due to maximum privilege capability.

No slot available to support user due to maximum privilege capability. (An implementation may only be able to support a certain number of sessions based on what authentication resources are required. For example, if User Level Authentication is disabled, an implementation may be able to allow a larger number of users that are limited to User Level privilege, than users that require higher privilege.)

SHM_CC_SESSION_SN_OUT_OF_RANGE

Session sequence number out-of-range

SHM_CC_INVALID_SESSION_ID

Invalid Session ID in request.

SHM_CC_PRIVILEGE_LEVEL_NOT_AVAILABLE

Requested level not available for this user.

SHM_CC_PRIVILEGE_LEVEL_EXCEEDED

Requested level exceeds Channel and/or User Privilege Limit.

SHM_CC_AUTHENTICATION_NOT_DISABLED

Cannot disable User Level authentication.

SHM_CC_ACCESS_MODE_NOT_SUPPORTED

Access mode not supported.

SHM_CC_LOCKED_KEY

Key is locked.

Cannot perform set / confirm. Key is locked.

SHM_CC_INVALID_PASSWORD

Password test failed, password data does not match stored value.

SHM_CC_INVALID_PASSWORD_SIZE

Password test failed, wrong password size was used.

SHM_CC_INVALID_PARAMETER

Parameter not supported.

SHM_CC_READ_ONLY

Attempt to write read-only parameter.

SHM_CC_RECORD_LEN_MISMATCH

Record rejected due to mismatch between record length in header data and number of bytes written.

Record rejected due to mismatch between record length in header data and number of bytes written. (Verifying the length is an optional operation for the management controller).

SHM_CC_FRU_DEVICE_BUSY

FRU device busy.

FRU device busy. The requested cannot be completed because the implementation of the logical FRU device is in a state where the FRU information is temporarily unavailable.

SHM_CC_WRITE_PROTECTED_OFFSET

Write protected offset.

Write-protected offset. Cannot complete write because one or more bytes of FRU data are to a write-protected offset in the FRU device. Note that an implementation may have allowed a ‘partial write’ of the data to occur.

SHM_CC_RECORD_CHANGED

Record changed.

Record changed. This status is returned if any of the record contents have been altered since the last time the Requester issued the request with 00h for the ‘Offset into SDR’ field.

SHM_CC_DESTINATION_UNAVAILABLE

Destination Unavailable.

Destination unavailable. Cannot deliver request to selected destination.

SHM_CC_TIMEOUT

Timeout occurred.

Timeout occurred.

SHM_CC_MISMATCH

Argument mismatch

SHM_CC_UNAVAILABLE

Resource unavailable

SHM_CC_ALREADY_USED

Already used

SHM_CC_ILLEGAL_CMD

Command illegal for specified sensor or record type.

SHM_CC_ILLEGAL_CMD_FOR_HASTATE

Command illegal for specified HA state.

SHM_CC_INVALID_GROUP

Illegal group.

SHM_CC_INVALID_COUNTER

Invalid counter.

SHM_CC_INVALID_THRESHOLD

Invalid threshold.

SHM_CC_LAST_ENTRY

Last entry reached.

SHM_CC_INVALID_ENTRY_ID

Non-existent entry.

SHM_CC_INVALID_FILE

Invalid file - not exist or no access rights.

SHM_CC_TYPE_UNSPECIFIED

Type unspecified.

SHM_CC_UPDATE_IN_PROGRESS

Update in progress.

SHM_CC_NO_STANDBY

Standby CMM is not physically present.

SHM_CC_STANDBY_STATE_UNKNOWN

There is no standby CMM.

SHM_CC_OLDER_FW

Peer CMM has an older version.

SHM_CC_CRITICAL_EVENTS

Critical events on peer CMM.

SHM_CC_COMM_FAILED

Communication failed.

SHM_CC_NOT_SYNCED

State not synchronized.

SHM_CC_DIR_NOT_VALID

Invalid script pathname.

SHM_CC_FILE_NOT_FOUND

Script file does not exist, has a different name, or is stored in another directory

SHM_CC_DIR_NOT_ALLOWED

Script pathname cannot be a directory.

SHM_CC_ZERO_SIZE

Script has zero bytes size.

SHM_CC_INSUFFICENT_PRIVILEGE

Invalid script execution rights.

SHM_CC_ERROR

Unspecified error.

Description:

General Domain error completion codes.

2.1.1.8 Predefined Record Id's

Syntax:

```
#define   IPMI_RECORD_ID_FIRST    0x0000
#define   IPMI_RECORD_ID_LAST     0xFFFF
```

Members:

IPMI_RECORD_ID_FIRST

First record

IPMI_RECORD_ID_LAST

Last record

Description:

Predefined Record Id's.

2.1.1.9 Record Id.

ShmRecordIdT

Syntax:

```
typedef   unsigned short        ShmRecordIdT;
```

Description:

ID used to uniquely identify SEL/SDR Record instances. The Record ID values 0000h and FFFFh have special meaning in the Event Access commands and must not be used as Record ID values for stored Records.

2.1.1.10 SDR Entry Header

ShmSdrEntryHeaderT

Syntax:

```
typedef struct {
        ShmRecordIdT                    recordId;
        unsigned char                   ver;
        unsigned char                   type;
} ShmSdrEntryHeaderT;
```

Members:

recordId

SDR record id.

ver

Version of the Sensor Model specification that this record is compatible with.

type

Record type.

2.1.1.11 Reservation Id.

ShmReservationIdT

Syntax:

```
typedef   unsigned short        ShmReservationIdT;
```

Description:

ID used to lock/reserve SEL/SDR Repository. The Reservation ID value 0000h is reserved.

2.1.1.12 Sensor Number.

ShmSensorNumT

Syntax:

```
typedef   unsigned char          ShmSensorNumT;
```

2.1.1.13 Sensor Type

ShmSensorTypeT

Syntax:

```
typedef   unsigned char          ShmSensorTypeT;
```

Description:

The type defines sensor type, which is one of IPMI Sensor Type Codes.

2.1.1.14 ATCA Hot Swap States

ShmAtcaHotSwapStateT

Syntax:

```
typedef enum {
          SHM_ATCA_HS_STATE_M0,
          SHM_ATCA_HS_STATE_M1,
          SHM_ATCA_HS_STATE_M2,
          SHM_ATCA_HS_STATE_M3,
          SHM_ATCA_HS_STATE_M4,
          SHM_ATCA_HS_STATE_M5,
          SHM_ATCA_HS_STATE_M6,
          SHM_ATCA_HS_STATE_M7
} ShmAtcaHotSwapStateT;
```

Members:

SHM_ATCA_HS_STATE_M0

M0 - FRU is not installed.
FRU is not installed.

SHM_ATCA_HS_STATE_M1

M1 - FRU is inactive.
FRU is inactive. The FRU is installed and its managing IPM Controller is operational, but payload is not powered. In this state the AdvancedTCA® “Insertion Criteria Met” condition (Section 3.2.4.1.1 in [ATCA]) is not satisfied.

SHM_ATCA_HS_STATE_M2

M2 - FRU activation request.
FRU activation request. The FRU is ready to attempt activation. “Insertion Criteria Met” condition is satisfied. FRU is held in this state if associated ‘Shelf Activation and Power Management record’ in the Shelf FRU Device(s) has ‘Shelf Manager Controlled Activation’ bit cleared.

SHM_ATCA_HS_STATE_M3

M3 - FRU activation in progress.
FRU activation in progress. The Shelf Manager is preparing the FRU to become fully active.

SHM_ATCA_HS_STATE_M4

M4 - FRU active.
FRU active. Normal operational state of the FRU.

SHM_ATCA_HS_STATE_M5

M5 - FRU deactivation request.
FRU deactivation request. The FRU is requesting deactivation permission from the Shelf Manager and the System Manager. “Extraction Criteria Met” condition (Section 3.2.4.1.2 in [ATCA]) is satisfied.

SHM_ATCA_HS_STATE_M6

M6 - FRU deactivation in progress.
FRU deactivation in progress. The FRU is gracefully shutting down its payload and preparing it to be deactivated. Following successful deactivation the FRU would move to the M1 state.

SHM_ATCA_HS_STATE_M7

M7 - Communication Lost.
Communication lost. This is an abnormal state when the Shelf Manager cannot communicate with the FRU and is not aware of its current status.

2.1.1.15 Timestamp

ShmTimestampT

Syntax:

```
typedef   unsigned int          ShmTimestampT
```

Description:

Timestamping is a key part of event logging and tracking changes to the Sensor Data Records and the SDR Repository. The following specifies the format of the seconds-based timestamp used in this document. Time is an unsigned 32-bit value representing the local time as the number of seconds from 00:00:00, January 1, 1970. This format is sufficient to maintain timestamping with 1-second resolution past the year 2100. This is based on a long standing UNIX-based standard for time keeping, which represents time as the number of seconds from 00:00:00, January 1, 1970 GMT. Similar time formats are used in ANSI C. The timestamps used for SDR and SEL records are assumed to be specified in relative local time. That is, the difference between the timestamp does not include the GMT offset. To convert the timestamp to a GMT-based time requires adding the GMT offset for the system. (The GMT offset needs to be obtained from system software level interfaces, there is no provision in the IPMI commands for storing or returning a GMT offset for the system.) Applications may use ANSI C time standard library routines for converting the SEL timestamp reading into other time formats. Be aware that this may require additional steps to account for the system's GMT offset.

2.1.2 Addressing information

2.1.2.1 Session ID

ShmSessionIdT

Syntax:

```
typedef    unsigned int          ShmSessionIdT;
```

2.1.2.2 Application name string max length

Syntax:

```
#define    SHM_APP_NAME_LEN_MAX    30
```

Members:

`SHM_APP_NAME_LEN_MAX`

Description:

Application name string max length.

2.1.2.3 Application name string

ShmAppNameT

Syntax:

```
typedef    char                  ShmAppNameT[SHM_APP_NAME_LEN_MAX];
```

2.1.2.4 User name string max length

Syntax:

```
#define    SHM_USER_NAME_LEN_MAX    16
```

Members:

`SHM_USER_NAME_LEN_MAX`

Description:

User name string max length.

2.1.2.5 User Name string

ShmUserNameT

Syntax:

```
typedef   char                  ShmUserNameT[SHM_USER_NAME_LEN_MAX];
```

2.1.2.6 Max number of system users

Syntax:

```
#define   SHM_USERS_MAX          63
```

Members:

`SHM_USERS_MAX`

Description:

Max number of system users.

2.1.2.7 User password string max length

Syntax:

```
#define  SHM_USER_PASSWORD_LEN_MAX   20
```

Members:

`SHM_USER_PASSWORD_LEN_MAX`

Description:

User password string max length.

2.1.2.8 User password string

ShmUserPasswordT

Syntax:

```
typedef   char          ShmUserPasswordT[SHM_USER_PASSWORD_LEN_MAX];
```

Description:

User password string (all values (0-255) are allowed for every byte.

2.1.2.9 Site Type

ShmPicmgSiteTypeT

Syntax:

```
typedef   unsigned char          ShmPicmgSiteTypeT;
```

Description:

(Table 3-8 PICMG 3.0 spec).

2.1.2.10 Site ID

ShmPicmgSiteIdT

Syntax:

```
typedef   unsigned char          ShmPicmgSiteIdT;
```

2.1.2.11 Predefined PICMG Site Types

Syntax:

```
#define  SHM_SITE_TYPE_ATCA_BOARD             0x00
#define  SHM_SITE_TYPE_PEM                    0x01
#define  SHM_SITE_TYPE_SHELF_FRU_INFO         0x02
#define  SHM_SITE_TYPE_DEDICATED_SHMC         0x03
#define  SHM_PICMG_SITE_TYPE_DEDICATED_SHMC   0x03
#define  SHM_SITE_TYPE_FAN_TRAY               0x04
#define  SHM_SITE_TYPE_FAN_FILTER_TRAY        0x05
#define  SHM_SITE_TYPE_ALARM                  0x06
#define  SHM_SITE_TYPE_MEZZANINE              0x07
#define  SHM_SITE_TYPE_PMC                    0x08
#define  SHM_SITE_TYPE_RTM                    0x09
```

Members:

SHM_SITE_TYPE_ATCA_BOARD

ATCA Board

SHM_SITE_TYPE_PEM

Power Entry Module

SHM_SITE_TYPE_SHELF_FRU_INFO

Shelf FRU Information

SHM_SITE_TYPE_DEDICATED_SHMC

Dedicated ShMC

SHM_PICMG_SITE_TYPE_DEDICATED_SHMC

Dedicated ShMC TBD to be removed

SHM_SITE_TYPE_FAN_TRAY

Fan Tray

SHM_SITE_TYPE_FAN_FILTER_TRAY

Fan Filter Tray

SHM_SITE_TYPE_ALARM

Alarm

SHM_SITE_TYPE_MEZZANINE

AdvancedMC Module(Mezzanine)

SHM_SITE_TYPE_PMC

PMC

SHM_SITE_TYPE_RTM

Rear Transition Module

Description:

Predefined PICMG Site Types.

2.1.2.12 Maximum length of Sensor Id String.

Syntax:

```
#define  SHM_SDR_FRU_SENSOR_STRING_MAX_LEN   16
```

Members:

SHM_SDR_FRU_SENSOR_STRING_MAX_LEN

Description:

Maximum length of Sensor Id String.

2.1.2.13 PICMG Physical Address

ShmPicmgPhyAddrT

Syntax:

```
typedef struct {
        ShmPicmgSiteTypeT                       SiteType;
        ShmPicmgSiteIdT                         SiteId;
} ShmPicmgPhyAddrT;
```

Members:

SiteType

Site Type

SiteId

Site ID

2.1.2.14 FRU Device Id

ShmFruDevIdT

Syntax:

```
typedef   unsigned char         ShmFruDevIdT;
```

2.1.2.15 Entity ID for the device associated with FRU information

ShmFruEntityIdT

Syntax:

```
typedef   unsigned char         ShmFruEntityIdT;
```

2.1.2.16 Instance number for entity

ShmFruEntityInstanceT

Syntax:

```
typedef   unsigned char          ShmFruEntityInstanceT;
```

2.1.2.17 Hardware Address

ShmHwAddrT

Syntax:

```
typedef   unsigned char          ShmHwAddrT;
```

2.1.2.18 IPMB-0 address

ShmIpmbAddrT

Syntax:

```
typedef   unsigned char          ShmIpmbAddrT;
```

2.1.2.19 Assigned IPMB addresses

Syntax:

```
#define  SHM_BROADCAST_IPMB_ADDR          0x00
#define  SHM_ACTIVE_SHM_IPMB_ADDR         0x20
#define  SHM_SLOT1_IPMB_ADDR              0x82
#define  SHM_SLOT2_IPMB_ADDR              0x84
#define  SHM_SLOT3_IPMB_ADDR              0x86
#define  SHM_SLOT4_IPMB_ADDR              0x88
#define  SHM_SLOT5_IPMB_ADDR              0x8A
#define  SHM_SLOT6_IPMB_ADDR              0x8C
#define  SHM_SLOT7_IPMB_ADDR              0x8E
#define  SHM_SLOT8_IPMB_ADDR              0x90
#define  SHM_SLOT9_IPMB_ADDR              0x92
#define  SHM_SLOT10_IPMB_ADDR             0x94
#define  SHM_SLOT11_IPMB_ADDR             0x96
#define  SHM_SLOT12_IPMB_ADDR             0x98
#define  SHM_SLOT13_IPMB_ADDR             0x9A
#define  SHM_SLOT14_IPMB_ADDR             0x9C
#define  SHM_SLOT15_IPMB_ADDR             0x9E
#define  SHM_SLOT16_IPMB_ADDR             0xA0
#define  SHM_TMP_ERROR_REPORT_IPMB_ADDR   0xEE
```

Members:

SHM_BROADCAST_IPMB_ADDR

Broadcast.

SHM_ACTIVE_SHM_IPMB_ADDR

Active ShMC address.

SHM_SLOT1_IPMB_ADDR

Logical slot 1 address.

SHM_SLOT2_IPMB_ADDR

Logical slot 2 address.

SHM_SLOT3_IPMB_ADDR

Logical slot 3 address.

SHM_SLOT4_IPMB_ADDR

Logical slot 4 address.

SHM_SLOT5_IPMB_ADDR

Logical slot 5 address.

SHM_SLOT6_IPMB_ADDR

Logical slot 6 address.

SHM_SLOT7_IPMB_ADDR

Logical slot 7 address.

SHM_SLOT8_IPMB_ADDR

Logical slot 8 address.

SHM_SLOT9_IPMB_ADDR

Logical slot 9 address.

SHM_SLOT10_IPMB_ADDR

Logical slot 10 address.

SHM_SLOT11_IPMB_ADDR

Logical slot 11 address.

SHM_SLOT12_IPMB_ADDR

Logical slot 12 address.

SHM_SLOT13_IPMB_ADDR

Logical slot 13 address.

SHM_SLOT14_IPMB_ADDR

Logical slot 14 address.

SHM_SLOT15_IPMB_ADDR

Logical slot 15 address.

SHM_SLOT16_IPMB_ADDR

Logical slot 16 address.

SHM_TMP_ERROR_REPORT_IPMB_ADDR

Temporary address available for error reporting when the Hardware Address fails parity validation.

Description:

Assigned IPMB addresses.

2.1.2.20 IPMI Logical Unit Number

ShmIpmiLunT

Syntax:

```
typedef   unsigned char          ShmIpmiLunT;
```

2.1.2.21 Logical Unit Numbers

Syntax:

```
#define   SHM_BMC_LUN            0x0
#define   SHM_OEM_LUN1           0x01
#define   SHM_SMS_MSG_LUN        0x10
#define   SHM_OEM_LUN2           0x11
```

Members:

SHM_BMC_LUN

BMC commands and Event Request Messages LUN.

Event Request Messages received on this LUN are routed to the Event Receiver function in the BMC, and automatically logged if SEL logging is enabled.

SHM_OEM_LUN1

OEM LUN 1.

OEM - reserved for BMC implementer / system integrator definition.

SHM_SMS_MSG_LUN

SMS Message LUN.

SMS Message LUN (Intended for messages to System Management Software). Messages received on this LUN are routed to the Receive Message Queue and retrieved using a Read Message command.

SHM_OEM_LUN2

OEM LUN 2.

OEM - reserved for BMC implementer / system integrator definition.

Description:

Logical Unit Numbers.

2.1.2.22 Sensor String ID bytes.

ShmSensorStringBytesT

Syntax:

```
typedef  char  ShmSensorStringBytesT[SHM_SDR_FRU_SENSOR_STRING_MAX_LEN];
```

2.1.2.23 Sensor Instance Modifier.

ShmSensorInstanceModifierT

Syntax:

```
typedef  char  ShmSensorInstanceModifierT[SHM_SDR_FRU_SENSOR_INSTANCE_
               MODIFIER_MAX_LEN];
```

2.1.2.24 Sensor name

ShmSensorStringT

Syntax:

```
typedef struct {
        ShmSensorStringBytesT                   stringBytes;
        unsigned char                           stringTypeLen;
        ShmSensorInstanceModifierT              instanceModifier;
        ShmFruDevIdT                            fruId;
        ShmIpmiLunT                             lun;
        ShmSensorTypeT                          sensorType;
} ShmSensorStringT;
```

Members:

stringBytes

ID String Bytes

stringTypeLen

ID String Type/Length Code

instanceModifier

Instance Modifier

fruId

The (sub)fruId to which the sensor belongs

lun

The LUN to which the sensor belongs

sensorType

Sensor Type

2.1.2.25 Logical Device Id used in LISM configuration file

ShmLogicalDevIdT

Syntax:

```
typedef   int                   ShmLogicalDevIdT;
```

2.1.2.26 Assigned logical device id's

Syntax:

```
#define  SHM_LOGICAL_DEVICE_NUMBER_NOT_FOUND   0x00
#define  SHM_LOGICAL_DEVICE_NUMBER_CHASSIS     0x100
#define  SHM_LOGICAL_DEVICE_NUMBER_CMM         0x101
#define  SHM_LOGICAL_DEVICE_NUMBER_OTHERCMM    0x102
```

Members:

SHM_LOGICAL_DEVICE_NUMBER_NOT_FOUND

not found.

SHM_LOGICAL_DEVICE_NUMBER_CHASSIS

Chassis ID.

SHM_LOGICAL_DEVICE_NUMBER_CMM

'this' CMM.

SHM_LOGICAL_DEVICE_NUMBER_OTHERCMM

'other' CMM.

Description:

Assigned logical device id's.

2.1.2.27 Transport Type

ShmTransportAddrTypeT

Syntax:

```
typedef enum {
         SHM_UDP_TRANSPORT_ADDRESS,
         SHM_TCP_TRANSPORT_ADDRESS,
         SHM_RMCP_PLUS_TRANSPORT_ADDRESS,
         SHM_FIFO_TRANSPORT_ADDRESS
} ShmTransportAddrTypeT;
```

Members:

SHM_UDP_TRANSPORT_ADDRESS

TBD: Temporarily added: for tests only.

SHM_TCP_TRANSPORT_ADDRESS

TBD: Temporarily added: for tests only.

SHM_RMCP_PLUS_TRANSPORT_ADDRESS

RMCP+ transport address.

SHM_FIFO_TRANSPORT_ADDRESS

FIFO (local) transport address.

2.1.2.28 RMCP+ Transport Protocol

ShmRmcpPlusTransportProtoT

Syntax:

```
typedef enum {
        SHM_RMCP_PLUS_TRANSPORT_PROTO_UDP,
        SHM_RMCP_PLUS_TRANSPORT_PROTO_SCTP
} ShmRmcpPlusTransportProtoT;
```

Members:

SHM_RMCP_PLUS_TRANSPORT_PROTO_UDP

UDP, default transport protocol

SHM_RMCP_PLUS_TRANSPORT_PROTO_SCTP

SCTP

2.1.2.29 IPMI/RMCP+ Authentication Level

ShmRmcpPlusAuthLevelT

Syntax:

```
typedef enum {
        SHM_RMCP_PLUS_AUTH_LEVEL_AUTO,
        SHM_RMCP_PLUS_AUTH_LEVEL_CALLBACK,
        SHM_RMCP_PLUS_AUTH_LEVEL_USER,
        SHM_RMCP_PLUS_AUTH_LEVEL_OPERATOR,
        SHM_RMCP_PLUS_AUTH_LEVEL_ADMINISTRATOR,
        SHM_RMCP_PLUS_AUTH_LEVEL_OEM
} ShmRmcpPlusAuthLevelT;
```

Members:

SHM_RMCP_PLUS_AUTH_LEVEL_AUTO

Auto.

SHM_RMCP_PLUS_AUTH_LEVEL_CALLBACK

Callback.

SHM_RMCP_PLUS_AUTH_LEVEL_USER

User.

SHM_RMCP_PLUS_AUTH_LEVEL_OPERATOR

Operator.

SHM_RMCP_PLUS_AUTH_LEVEL_ADMINISTRATOR

Administrator.

SHM_RMCP_PLUS_AUTH_LEVEL_OEM

OEM.

2.1.2.30 RMCP+ Authentication Type

ShmRmcpPlusAuthTypeT

Syntax:

```
typedef enum {
        SHM_RMCP_PLUS_AUTH_TYPE_AUTO,
        SHM_RMCP_PLUS_AUTH_TYPE_NONE,
        SHM_RMCP_PLUS_AUTH_TYPE_HMAC_SHA1,
        SHM_RMCP_PLUS_AUTH_TYPE_HMAC_MD5
} ShmRmcpPlusAuthTypeT;
```

Members:

SHM_RMCP_PLUS_AUTH_TYPE_AUTO

Auto.

SHM_RMCP_PLUS_AUTH_TYPE_NONE

None.

SHM_RMCP_PLUS_AUTH_TYPE_HMAC_SHA1

HMAC_SHA1.

SHM_RMCP_PLUS_AUTH_TYPE_HMAC_MD5

HMAC_SD5

2.1.2.31 RMCP+ Integrity Type

ShmRmcpPlusIntegrityTypeT

Syntax:

```
typedef enum {
       SHM_RMCP_PLUS_INTEGRITY_TYPE_AUTO,
       SHM_RMCP_PLUS_INTEGRITY_TYPE_NONE,
       SHM_RMCP_PLUS_INTEGRITY_TYPE_HMAC_SHA1_96,
       SHM_RMCP_PLUS_INTEGRITY_TYPE_HMAC_MD5_128,
       SHM_RMCP_PLUS_INTEGRITY_TYPE_MD5_128
} ShmRmcpPlusIntegrityTypeT;
```

Members:

SHM_RMCP_PLUS_INTEGRITY_TYPE_AUTO

Auto.

SHM_RMCP_PLUS_INTEGRITY_TYPE_NONE

None.

SHM_RMCP_PLUS_INTEGRITY_TYPE_HMAC_SHA1_96

HMAC_SHA1_96.

SHM_RMCP_PLUS_INTEGRITY_TYPE_HMAC_MD5_128

HMAC_MD5_128

SHM_RMCP_PLUS_INTEGRITY_TYPE_MD5_128

MD5_128

2.1.2.32 RMCP+ Confidentiality Type

ShmRmcpPlusConfidentialityTypeT

Syntax:

```
typedef enum {
      SHM_RMCP_PLUS_CONFIDENTIALITY_TYPE_AUTO,
      SHM_RMCP_PLUS_CONFIDENTIALITY_TYPE_NONE,
      SHM_RMCP_PLUS_CONFIDENTIALITY_TYPE_AES_CBC_128
} ShmRmcpPlusConfidentialityTypeT;
```

Members:

SHM_RMCP_PLUS_CONFIDENTIALITY_TYPE_AUTO

Auto.

SHM_RMCP_PLUS_CONFIDENTIALITY_TYPE_NONE

None.

SHM_RMCP_PLUS_CONFIDENTIALITY_TYPE_AES_CBC_128

AES_CBC_128.

2.1.2.33 RMCP+ Transport Address

ShmRmcpPlusTransportAddrT

Syntax:

```
typedef struct {
       ShmRmcpPlusTransportProtoT          transportProto;
       char                                hostName[20];
       ShmUserNameT                        userName;
       ShmUserPasswordT                    password;
       ShmRmcpPlusAuthLevelT               authLevel;
       ShmRmcpPlusAuthTypeT                authType;
       ShmRmcpPlusIntegrityTypeT           integrityType;
       ShmRmcpPlusConfidentialityTypeT     confidentialityType;
} ShmRmcpPlusTransportAddrT;
```

Members:

transportProto

Transport Protocol

hostName

Host address.

userName

User Name.

password

Password.

authLevel

RMCP+ Authentication Level.

authType

RMCP+ Authentication Type.

integrityType

RMCP+ Integrity Type.

confidentialityType

RMCP+ Confidentiality Type.

2.1.2.34 FIFO Transport Address

ShmLocalFifoTransportAddrT

Syntax:

```
typedef struct {
        ShmUserNameT                    userName;
        char*                           path;
} ShmLocalFifoTransportAddrT;
```

Description:

TBD.

Members:

userName

User Name.

path

FIFO queue name.

2.1.2.35 Transport Address

ShmTransportAddrT

Syntax:

```
typedef struct {
        ShmAppNameT                             appName;
        ShmTransportAddrTypeT                   type;
        union {
            char                                udp;
            char                                tcp;
            ShmRmcpPlusTransportAddrT           rmcpPlus;
            ShmLocalFifoTransportAddrT          localFifo;
        } u;
} ShmTransportAddrT;
```

Description:

Address of the remote host where the remote server is located.

Members:

appName

Application Name.

type

Transport Address type selector.

udp

UDP Transport Address.

Valid for selector: SHM_UDP_TRANSPORT_ADDRESS

tcp

TCP Transport Address.

Valid for selector: SHM_TCP_TRANSPORT_ADDRESS

rmcpPlus

RMCP+ Transport Address.

Valid for selector: SHM_RMCP_PLUS_TRANSPORT_ADDRESS

localFifo

Local FIFO Transport Address.

Valid for selector: SHM_FIFO_TRANSPORT_ADDRESS

2.1.2.36 IPMC Address

ShmIpmcAddrT

Syntax:

```
typedef struct {
        ShmIpmbAddrT                    slaveAddr;
        ShmIpmiLunT                     lun;
} ShmIpmcAddrT;
```

Members:

slaveAddr

IPMB Slave Address.

lun

LUN

2.1.2.37 FRU Location

ShmFruLocationT

Syntax:

```
typedef struct {
        ShmHwAddrT                      hwAddr;
        ShmPicmgPhyAddrT                phyAddr;
        ShmFruDevIdT                    fruId;
} ShmFruLocationT;
```

Description:

Uniquely identifies FRU.

Members:

hwAddr

FRU hardware address.

phyAddr

FRU physical address.

fruId

FRU Id.

2.1.2.38 Maximum length of FRU name

Syntax:

```
#define    SHM_FRU_NAME_MAX_LEN    16
```

Members:

SHM_FRU_NAME_MAX_LEN

Maximum length of FRU name (Device ID string from Device Locator SDR.).

Description:

2.1.2.39 Sensor Instance Modifier.

ShmFruDeviceNameT

Syntax:

```
typedef   char              ShmFruDeviceNameT[SHM_FRU_NAME_MAX_LEN];
```

2.1.2.40 Active FRU location

ShmFruActiveLocationT

Syntax:

```
typedef struct {
        ShmFruLocationT                 fruLocation;
        ShmIpmbAddrT                    ipmbAddr;
        ShmIpmiLunT                     lun;
        ShmFruEntityIdT                 entityId;
        ShmFruEntityInstanceT           entityInstance;
        ShmAtcaHotSwapStateT            hsState;
        ShmFruDeviceNameT               fruName;
} ShmFruActiveLocationT;
```

Description:

Uniquely identifies FRU and provides information on its state.

Members:

fruLocation

FRU location.

ipmbAddr

FRU IPMB-0 address.

lun

LUN

entityId

Entity ID for the device associated with this FRU information

entityInstance

Instance number for entity

hsState

FRU Hot Swap state.

fruName

FRU Name.

2.2 API functions

2.2.1 Session Management

2.2.1.1 Session Open

shmSessionOpen

Syntax:

```
ShmErrorT
shmSessionOpen(
        ShmTransportAddrT                   shelfMgrAddr,
        ShmSessionIdT*                      sessionId);
```

Description:

This function opens an IPMI session for a given domain. Remarks None.

Parameters:

`shelfMgrAddr [in]`

Identifies the address of the remote host where the remote server is located. Client will establish connection with given host.

`sessionId [inout]`

Pointer to a location to store an identifier for the newly opened session. This identifier is used for subsequent access to IPMI resources.

Returns:

Possible return values are:

SHM_CC_OK

is returned on successful completion; otherwise, an error code is returned.

SHM_CC_INVALID_USERNAME

Invalid user name.

SHM_CC_SESSION_SLOT_FULL

No session slot available.

SHM_CC_USER_SLOT_FULL

No slot available for given user. Limit of user sessions allowed under that name has been reached.

SHM_CC_PRIVILEGE_SLOT_FULL

No slot available to support user due to maximum privilege capability.

SHM_CC_PRIVILEGE_LEVEL_NOT_AVAILABLE

Requested level not available for this user.

SHM_CC_PRIVILEGE_LEVEL_EXCEEDED

Requested level exceeds Channel and/or User Privilege Limit.

SHM_CC_ACCESS_MODE_NOT_SUPPORTED

Access mode not supported.

SHM_CC_DESTINATION_UNAVAILABLE

Destination Unavailable.

SHM_CC_TIMEOUT

Timeout occurred.

SHM_CC_ERROR

Unspecified error.

2.2.1.2 Session Close

shmSessionClose

Syntax:

```
ShmErrorT
shmSessionClose(
        ShmSessionIdT                                   sessionId);
```

Description:

This function closes an IPMI session. After closing a session, the SessionId will no longer be valid. Remarks None.

Parameters:

`sessionId [in]`

Session identifier previously obtained using shmSessionOpen().

Returns:

Possible return values are:

`SHM_CC_OK`

is returned on successful completion; otherwise, an error code is returned.

`SHM_CC_INVALID_SESSION_HANDLE`

Invalid session handle. The session handle does not match up with any currently active sessions.

`SHM_CC_TIMEOUT`

Timeout occurred.

`SHM_CC_ERROR`

Unspecified error.

2.2.2 Addressing

2.2.2.1 List Locations

shmOamLocationsList

Syntax:

```
ShmErrorT
shmOamLocationsList(
        ShmSessionIdT                   sessionId,
        unsigned int*                   locationsNum,
        ShmFruLocationT**               locations);
```

Description:

This function returns the list of all locations available in the system. It retrieves this list from the SDR repository module that is running on the LISM module.

Parameters:

`sessionId [in]`

Identifier for a session context previously obtained using shmSessionOpen().

`locationsNum [out]`

Number of locations returned.

`locations [out]`

FRU locations.

Returns:

Possible return values are:

`SHM_CC_OK`

is returned on successful completion; otherwise, appropriate IPMI error code is returned.

2.2.2.2 List Active Locations

shmOamActiveLocationsList

Syntax:

```
ShmErrorT
shmOamActiveLocationsList(
        ShmSessionIdT                   sessionId,
        ShmBoolT                        wait,
        unsigned int*                   locationsNum,
        ShmFruActiveLocationT**         locations);
```

Description:

This function returns the list of active locations available in the system. It retrieves this list from the SDR repository module that is running on the LISM module.

Parameters:

`sessionId [in]`

Identifier for a session context previously obtained using shmSessionOpen().

`wait [in]`

This setting is only valid if SDR repository is being updated at the time when the function is called. When enabled - function waits until the SDR repository update procedure is finished and returns requested entries. When disabled - function returns immediately with an error code indicating that the SDR repository update procedure is in progress.

`locationsNum [out]`

Number of locations returned.

`locations [out]`

Location FRU Physical Address Path.

Returns:

Possible return values are:

`SHM_CC_OK`

is returned on successful completion; otherwise, appropriate IPMI error code is returned.

2.2.2.3 List Location Targets

shmOamLocationTargetsList

Syntax:

```
ShmErrorT
shmOamLocationTargetsList(
        ShmSessionIdT                   sessionId,
        ShmIpmbAddrT                    ipmbAddr,
        unsigned int*                   locationsNum,
        ShmSensorStringT**              targetName);
```

Description:

This function returns the list of targets (i.e. IPMI sensors) hosted on the specific FRU. This list is retrieved from the SDR repository module that is running on the LISM module.

Parameters:

`sessionId [in]`

Identifier for a session context previously obtained using shmSessionOpen().

`ipmbAddr [in]`

IPMB address.

`locationsNum [out]`

Number of targets returned.

`targetName [out]`

Target Name (Id string from the SDR entry).

Returns:

Possible return values are:

`SHM_CC_OK`

is returned on successful completion; otherwise, appropriate IPMI error code is returned.

2.2.2.4 Return location information.

shmOamLocationGet

Syntax:

```
ShmErrorT
shmOamLocationGet(
        ShmSessionIdT                   sessionId,
        ShmFruLocationT*                location,
        ShmBoolT*                       activeShmKnown,
        ShmIpmbAddrT*                   activeShm,
        ShmBoolT*                       standbyShmKnown,
        ShmIpmbAddrT*                   standbyShm);
```

Description:

This function returns this Shelf Manager location information, IPMB address of the known active Shelf Manager and known IPMB address of a standby Shelf Manager.

Parameters:

`sessionId [in]`

Identifier for a session context previously obtained using shmSessionOpen().

`location [out]`

Location info associated with FRU upon which the SHM User is running.

`activeShmKnown [out]`

Active Shelf Manager IPMB address provided in the activeShm parameter.

`activeShm [out]`

Active Shelf Manager IPMB address.

`standbyShmKnown [out]`

Standby Shelf Manager IPMB address provided in the standbyShm parameter.

`standbyShm [out]`

Standby Shelf Manager IPMB address.

Returns:

Possible return values are:

`SHM_CC_OK`

is returned on successful completion; otherwise, appropriate IPMI error code is returned.

2.2.2.5 Get Sensor Id

shmOamSensorNumLookup

Syntax:

```
ShmErrorT
shmOamSensorNumLookup(
        ShmSessionIdT                   sessionId,
        ShmIpmcAddrT                    dstAddr,
        ShmSensorStringT                sensorName,
        ShmSensorNumT*                  sensorNumber);
```

Description:

This function returns Sensor Id for target identified by the name specified as the input parameter. Sensor Id is retrieved from the SDR repository module that is running on the LISM module.

Parameters:

`sessionId [in]`

Identifier for a session context previously obtained using shmSessionOpen().

`dstAddr [in]`

FRU path. Indicates the FRU device for which the command is intended.

`sensorName [in]`

Sensor Name

Info related with sensor name.

`sensorNumber [out]`

Sensor Id

Sensor number.

Returns:

Possible return values are:

`SHM_CC_OK`

is returned on successful completion; otherwise, appropriate IPMI error code is returned.

2.2.2.6 Get Sensor Name

shmOamSensorNameLookup

Syntax:

```
ShmErrorT
shmOamSensorNameLookup(
        ShmSessionIdT                   sessionId,
        ShmIpmcAddrT                    dstAddr,
        ShmSensorNumT                   sensorNumber,
        ShmSensorStringT*               sensorName);
```

Description:

This function returns Sensor name for target identified by the Id specified as the input parameter. Sensor name is retrieved from the SDR repository module that is running on the LISM module.

Parameters:

`sessionId [in]`

Identifier for a session context previously obtained using shmSessionOpen().

`dstAddr [in]`

FRU path. Indicates the FRU device for which the command is intended.

`sensorNumber [in]`

Sensor Id

`sensorName [out]`

Sensor Name

Returns:

Possible return values are:

`SHM_CC_OK`

is returned on successful completion; otherwise, appropriate IPMI error code is returned.

2.2.2.7 Get Sensor related SDR

shmOamSensorSdrGet

Syntax:

```
ShmErrorT
shmOamSensorSdrGet(
        ShmSessionIdT                   sessionId,
        ShmIpmcAddrT                    dstAddr,
        ShmSensorNumT                   sensorNumber,
        ShmSdrEntryHeaderT*             sdrEntryHeader,
        unsigned int*                   recordLength,
        unsigned char**                 recordData);
```

Description:

This command returns the SDR record for a sensor.

Parameters:

`sessionId [in]`

Session identifier previously obtained using shmSessionOpen().

`dstAddr [in]`

Destination address. Provides IPMB address and LUN that identify target destination.

`sensorNumber [in]`

Sensor Id.

`sdrEntryHeader [out]`

SDR Entry Header. Record Body is passed in recordData parameter below.

`recordLength [out]`

Number of bytes of data following the Record Length field.

`recordData [out]`

Record Body. The remaining SDR Record Type specific information for the particular sensor data record.

Returns:

Possible return values are:

`SHM_CC_OK`

is returned on successful completion; otherwise, appropriate IPMI error code is returned.

2.2.2.8 Get IPMB address related to given location

shmOamLogicalDeviceAddressGet

Syntax:

```
ShmErrorT
shmOamLogicalDeviceAddressGet(
        ShmSessionIdT                   sessionId,
        char*                           deviceName,
        ShmIpmbAddrT*                   ipmbAddr);
```

Description:

This function returns address of the specified logical device based on information obtained from configuration file.

Parameters:

`sessionId [in]`

Session identifier previously obtained using shmSessionOpen().

`deviceName [in]`

Location string.

`ipmbAddr [out]`

IPMB address.

Returns:

Possible return values are:

SHM_CC_OK

Successful completion

SHM_CC_NOT_FOUND

Device not found.

SHM_CC_ERROR

Internal error.

2.2.2.9 Get Device Name related to given IPMB address

shmOamLogicalDeviceNameGet

Syntax:

```
ShmErrorT
shmOamLogicalDeviceNameGet(
        ShmSessionIdT                   sessionId,
        ShmIpmbAddrT                    ipmbAddr,
        char**                          deviceName);
```

Description:

This function returns address of the specified logical device based on information obtained from configuration file.

Parameters:

`sessionId [in]`

Session identifier previously obtained using shmSessionOpen().

`ipmbAddr [in]`

IPMB address.

`deviceName [out]`

Location string.

Returns:

Possible return values are:

SHM_CC_OK

Successful completion

SHM_CC_NOT_FOUND

Device not found.

SHM_CC_ERROR

Internal error.

2.2.2.10 Get Device Name related to given alias

shmOamLogicalDeviceAliasToNameConvert

Syntax:

```
ShmErrorT
shmOamLogicalDeviceAliasToNameConvert(
        ShmSessionIdT                   sessionId,
        char*                           aliasName,
        char**                          deviceName);
```

Description:

This function returns name of the specified logical device based on information obtained from LISM configuration file.

Parameters:

`sessionId [in]`

Session identifier previously obtained using shmSessionOpen().

`aliasName [in]`

Alias string.

`deviceName [out]`

Location string.

Returns:

Possible return values are:

SHM_CC_OK

Successful completion.

SHM_CC_ERROR

Internal error.

2.2.2.11 Get Alias related to given device Name

shmOamLogicalDeviceNameToAliasConvert

Syntax:

```
ShmErrorT
shmOamLogicalDeviceNameToAliasConvert(
        ShmSessionIdT                   sessionId,
        char*                           deviceName,
        char**                          aliasName);
```

Description:

This function returns an user-friendly alias of the specified logical device based on information obtained from LISM configuration file.

Parameters:

`sessionId [in]`

Session identifier previously obtained using shmSessionOpen().

`deviceName [in]`

Location string.

`aliasName [out]`

Alias string.

Returns:

Possible return values are:

SHM_CC_OK

Successful completion.

SHM_CC_ERROR

Internal error.

2.2.2.12 Get Device Id related to given IPMB address

shmOamLogicalDeviceIdGet

Syntax:

```
ShmErrorT
shmOamLogicalDeviceIdGet(
          ShmSessionIdT                    sessionId,
          ShmIpmbAddrT                     ipmbAddr,
          ShmLogicalDevIdT*                deviceId);
```

Description:

This function returns logical device id related to given IMPB address based on information obtained from LISM configuration file.

Parameters:

`sessionId [in]`

Session identifier previously obtained using shmSessionOpen().

`ipmbAddr [in]`

IPMB address.

`deviceId [out]`

Location string.

Returns:

Possible return values are:

SHM_CC_OK

Successful completion. (If not found input parameter is returned)

2.2.2.13 Get Sensor Fru ID

shmOamSensorFruIdLookup

Syntax:

```
ShmErrorT
shmOamSensorFruIdLookup(
        ShmSessionIdT                   sessionId,
        ShmIpmcAddrT                    dstAddr,
        ShmSensorNumT                   sensorNumber,
        ShmFruDevIdT*                   fruId);
```

Description:

This function returns FRU ID for target identified by the Id specified as the input parameter. FRU ID is retrieved from the SDR repository module that is running on the LISM module.

Parameters:

`sessionId [in]`

Identifier for a session context previously obtained using shmSessionOpen().

`dstAddr [in]`

FRU path. Indicates the FRU device for which the command is intended.

`sensorNumber [in]`

Sensor Number

`fruId [out]`

FRU ID

Returns:

Possible return values are:

`SHM_CC_OK`

is returned on successful completion; otherwise, appropriate IPMI error code is returned.

2.2.2.14 Get Sensor Type

shmOamSensorTypeLookup

Syntax:

```
ShmErrorT
shmOamSensorTypeLookup(
        ShmSessionIdT                   sessionId,
        ShmIpmcAddrT                    dstAddr,
        ShmSensorNumT                   sensorNumber,
        ShmSensorTypeT*                 sensorType);
```

Description:

This function returns type for target identified by the Id specified as the input parameter. Type is retrieved from the SDR repository module that is running on the LISM module.

Parameters:

`sessionId [in]`

Identifier for a session context previously obtained using shmSessionOpen().

`dstAddr [in]`

FRU path. Indicates the FRU device for which the command is intended.

`sensorNumber [in]`

Sensor Number

`sensorType [out]`

FRU ID

Returns:

Possible return values are:

`SHM_CC_OK`

is returned on successful completion; otherwise, appropriate IPMI error code is returned.

2.2.2.15 Get Lun(s)

shmOamLunsLookup

Syntax:

```
ShmErrorT
shmOamLunsLookup(
        ShmSessionIdT                   sessionId,
        ShmIpmbAddrT                    ipmbAddr,
        ShmSensorNumT                   sensorNumber,
        ShmLunsUseT*                    luns);
```

Description:

This function returns luns for target identified by the Id specified as the input parameter. Luns is retrieved from the SDR repository module that is running on the LISM module.

Parameters:

`sessionId [in]`

Identifier for a session context previously obtained using shmSessionOpen().

`ipmbAddr [in]`

FRU path. Indicates the FRU device for which the command is intended.

`sensorNumber [in]`

Sensor Number

`luns [out]`

luns table

Returns:

Possible return values are:

`SHM_CC_OK`

is returned on successful completion; otherwise, appropriate IPMI error code is returned.

`SHM_CC_NOT_FOUND`

Target not found on any lun.

`SHM_CC_MANY_LUNS`

Target found on at least two LUN's..

3 Common Upgrade API

3.1 Data Types definitions

3.1.1 Common Upgrade Defines

Syntax:

```
#define IMAGE_NAME_LEN                256
#define IMAGE_DESCRIPTION_LEN         1024
#define IMAGE_VERSION_LEN             128
#define TARGET_OS_LEN                 256
#define REPOSITORY_IMAGE_PATH_LEN     1024
```

Members:

IMAGE_NAME_LEN

Maximum length of image name

IMAGE_DESCRIPTION_LEN

Length of image description

IMAGE_VERSION_LEN

Maximum length of version string

TARGET_OS_LEN

Maximum length of target OS name

REPOSITORY_IMAGE_PATH_LEN

Max length of repository image path

Description:

3.1.2 Error codes returned by CU API functions

CUResultT

Syntax:

```
typedef enum {
        CU_SUCCESS                  = 0,
        CU_INVDESTADDRFORMAT        = 1,
        CU_FRUNOTPRESENT            = 2,
        CU_FRUTYPEINV               = 3,
        CU_INVDESTHANDLE            = 4,
        CU_INVIMAGENAME             = 5,
        CU_INVIMAGEINSTANCE         = 6,
        CU_INVSRC                   = 7,
        CU_INVTYPE                  = 8,
        CU_INVPROT                  = 9,
        CU_SRCUNREACHABLE           = 10,
        CU_SRCCORRUPTED             = 11,
        CU_DESTACTIVE               = 12,
        CU_INSUFFSIZE               = 13,
        CU_PROPNOTSET               = 14,
        CU_GETPROPERR               = 15,
        CU_GETPROPPARTIAL           = 16,
        CU_IMAGELOCKED              = 17,
        CU_IMAGENOTLOCKED           = 18,
        CU_RESTARTNOTSUPPORTED      = 19,
        CU_IMAGEVERIFYERR           = 20,
        CU_FUNCNOTSUPPORTED         = 21,
        CU_RESTARTINITIATED         = 22,
        CU_GENERALERR               = 23,
        CU_TIMEOUT                  = 24
} CUResultT;
```

Members:

`CU_SUCCESS`

No error

CU_INVDESTADDRFORMAT

Invalid destination address format

CU_FRUNOTPRESENT

FRU not present

CU_FRUTYPEINV

FRU type specified is invalid

CU_INVDESTHANDLE

Invalid Destination Handle

CU_INVIMAGENAME

Invalid Image Name

CU_INVIMAGEINSTANCE

Invalid Image Instance

CU_INVSRC

Invalid Source Address or Binary Image Name

CU_INVTYPE

Invalid Type of Image

CU_INVPROT

Request Service Protocol Not Supported

CU_SRCUNREACHABLE

Source Address Not Reachable

CU_SRCCORRUPTED

The image at the source address is corrupted.

CU_DESTACTIVE

The destination image is currently active.

CU_INSUFFSIZE

Size of array allocated is insufficient to return image names and instances.

CU_PROPNOTSET

Property not set due to error.

CU_GETPROPERR

Error in retrieving properties

CU_GETPROPPARTIAL

Partial success in retrieving properties

CU_IMAGELOCKED

Images on FRU are already locked - try again later.

CU_IMAGENOTLOCKED

Attempt to unlock Images not locked by destination handle.

CU_RESTARTNOTSUPPORTED

Single image restart not supported.

CU_IMAGEVERIFYERR

Error during image property verification

CU_FUNCNOTSUPPORTED

Function not supported by plugin

CU_RESTARTINITIATED

Image Restart action initiated.

CU_GENERALERR

Catchall - more specific err info lacking.

CU_TIMEOUT

Internal timeout occurred.

3.1.3 Current Image State

ImageCurrentStateT

Syntax:

```
typedef enum {
          IMAGE_CURSTATE_EMPTY,
          IMAGE_CURSTATE_INACTIVE,
          IMAGE_CURSTATE_ACTIVE,
          IMAGE_CURSTATE_OEM1,
          IMAGE_CURSTATE_OEM2,
          IMAGE_CURSTATE_OEM3,
          IMAGE_CURSTATE_OEM4,
          IMAGE_CURSTATE_OEM5
} ImageCurrentStateT;
```

Members:

`IMAGE_CURSTATE_EMPTY`

Image store does not contain an image. This state may also be entered if image fails download or verification.

`IMAGE_CURSTATE_INACTIVE`

Image is currently not executing. A new image may be written. When a new image is written, its current state property is set to this value.

`IMAGE_CURSTATE_ACTIVE`

Image is currently executing. A new image may not be written.

`IMAGE_CURSTATE_OEM1`

OEM defined state

`IMAGE_CURSTATE_OEM2`

OEM defined state

`IMAGE_CURSTATE_OEM3`

OEM defined state

`IMAGE_CURSTATE_OEM4`

OEM defined state

`IMAGE_CURSTATE_OEM5`

OEM defined state

Description:

The enumeration defines possible values of IMAGE_CURRENT_STATE property.

3.1.4 Image Restart

ImageRestartEnumT

Syntax:

```
typedef enum {
          IMAGE_RESTART_FRU_COLD,
          IMAGE_RESTART_FRU_WARM,
          IMAGE_RESTART_IMAGE,
          IMAGE_RESTART_OEM1,
          IMAGE_RESTART_OEM2,
          IMAGE_RESTART_OEM3
} ImageRestartEnumT;
```

Members:

`IMAGE_RESTART_FRU_COLD`

A power cycle of the FRU is required for new image to be reinitialized.

`IMAGE_RESTART_FRU_WARM`

A warm restart of the FRU is required for the new image to be reinitialized.

`IMAGE_RESTART_IMAGE`

Only the image must be restarted for it to be reinitialize.

`IMAGE_RESTART_OEM1`

OEM defined

`IMAGE_RESTART_OEM2`

OEM defined

`IMAGE_RESTART_OEM3`

OEM defined

Description:

The enumeration defines the type of image restart.

3.1.5 Image Verification Result

VerifyRetCodeT

Syntax:

```
typedef enum {
          CU_VERIFY_SUCCESS,
          CU_VERIFY_ERRCHKSUM,
          CU_VERIFY_ERRNOTPRESENT,
          CU_VERIFY_ERRNODEFAULT,
          CU_VERIFY_ERRTOOMANYDEFAULT,
          CU_VERIFY_ERRTOOMANYFALLBACK,
          CU_VERIFY_OEM1,
          CU_VERIFY_OEM2,
          CU_VERIFY_OEM3,
          CU_VERFIY_OEM4,
          CU_VERFIY_OEM5
} VerifyRetCodeT;
```

Members:

`CU_VERIFY_SUCCESS`

Image verification success.

`CU_VERIFY_ERRCHKSUM`

Checksum error

`CU_VERIFY_ERRNOTPRESENT`

Required image not present

`CU_VERIFY_ERRNODEFAULT`

No image instances of this type are defined to be default for next reboot - at least one must be default

`CU_VERIFY_ERRTOOMANYDEFAULT`

Too many image instances of this type are default for next boot

`CU_VERIFY_ERRTOOMANYFALLBACK`

Too many image instances of this type are fallback for next boot.

`CU_VERIFY_OEM1`

OEM defined verification error

`CU_VERIFY_OEM2`

OEM defined verification error

CU_VERIFY_OEM3

OEM defined verification error

CU_VERFIY_OEM4

OEM defined verification error

CU_VERFIY_OEM5

OEM defined verification error

Description:

The enumeration defines possible result of single image verification.

3.1.6 Image Properties

ImagePropEnumT

Syntax:

```
typedef enum {
          IMAGE_HANDLE,
          IMAGE_NAME,
          IMAGE_INSTANCE,
          IMAGE_REQUIRED,
          IMAGE_CURRENT_STATE,
          IMAGE_ROLE_NEXTBOOT,
          IMAGE_START_ADDR,
          IMAGE_END_ADDR,
          IMAGE_LEN_MAX,
          IMAGE_LEN_CUR,
          IMAGE_VERSION_STRING,
          IMAGE_VERSION_MAJOR,
          IMAGE_VERSION_MINOR,
          IMAGE_VERSION_REVISION,
          IMAGE_VERSION_BUILD,
          IMAGE_SERIAL_NUMBER,
          IMAGE_TARGET_OS,
          IMAGE_DESCRIPTION,
          IMAGE_RELEASE_TIME,
          IMAGE_INSTALL_TIME,
          IMAGE_ACTIVATE_TIME,
          IMAGE_RESTART_TYPE,
          IMAGE_INSTALL_TYPE,
          SINGLE_IMAGE_INSTALL
} ImagePropEnumT;
```

Members:

`IMAGE_HANDLE`

Unique Handle for the image

`IMAGE_NAME`

The name property defines the label by which the object is known.

`IMAGE_INSTANCE`

Instance number of image

IMAGE_REQUIRED

Specifies whether this image is required for the proper functioning of the FRU.

IMAGE_CURRENT_STATE

Current state of image

IMAGE_ROLE_NEXTBOOT

Role of image on next boot

IMAGE_START_ADDR

Pointer to start address. This is opaque.

IMAGE_END_ADDR

Pointer to end address.

IMAGE_LEN_MAX

Maximum size of image in bytes.

IMAGE_LEN_CUR

Length of current image in bytes.

IMAGE_VERSION_STRING

A string representing the version info ? e.g. '12.1(3)T'. Since vastly different representations and semantics exist for versions, no interpretation is assumed.

IMAGE_VERSION_MAJOR

The major number component of the version information - for example, '12' from version 12.1.3.1234. This property is defined as a numeric value to allow the determination of 'newer' vs. 'older' releases. A 'newer' major release is indicated by a larger numeric value.

IMAGE_VERSION_MINOR

The minor number component of the version. E.g, '1' from version 12.1.3.1251. This property is defined as a numeric value to allow the determination of 'newer' vs. 'older' releases. A 'newer' minor release is indicated by a larger numeric value.

IMAGE_VERSION_REVISION

The revision or maintenance release component of the version. E.g. '3' from version 12.1.3.1251. This property is defined as a numeric value to allow the determination of 'newer' vs. 'older' releases. A 'newer' revision is indicated by a larger numeric value.

IMAGE_VERSION_BUILD

The build number of the image. E.g. '1251' from version 12.1.3.1251. This property is defined as a numeric value to allow the determination of 'newer' vs. 'older' releases. A 'newer' build is indicated by a larger numeric value

IMAGE_SERIAL_NUMBER

A manufacturer-allocated number used to identify the software.

IMAGE_TARGET_OS

Specifies the target operating systems of the software. Due to the extreme variability in operating systems, this property is defined as a string array.

IMAGE_DESCRIPTION

Textual Description of the Image

IMAGE_RELEASE_TIME

A time value indicating when the image was released by the manufacturer. Lack of a value is acceptable.

IMAGE_INSTALL_TIME

A time value indicating when the image was installed on this managed element. A lack of a value does not indicate that the image is not installed.

IMAGE_ACTIVATE_TIME

A time value indicating when the image was activated.

IMAGE_RESTART_TYPE

The type of restart required to make this image active.

IMAGE_INSTALL_TYPE

The type of installation - normal or forced.

SINGLE_IMAGE_INSTALL

Special OEM upgrade mode, used by CLI.

Description:

The enumeration defines the properties possible for an image

3.1.7 Image Installation Type

ImageInstallTypeT

Syntax:

```
typedef enum {
          INSTALL_TYPE_NORMAL,
          INSTALL_TYPE_FORCE
} ImageInstallTypeT;
```

Members:

INSTALL_TYPE_NORMAL

INSTALL_TYPE_FORCE

3.1.8 Destination Address

DestAddrT

Syntax:

```
typedef   SaHpiEntityPathT        DestAddrT;
```

Description:

HPI Entity Path format is used.

3.1.9 Destination Handle

DestHandleT

Syntax:

```
typedef   unsigned int            DestHandleT;
```

3.1.10 Image Name

ImageNameT

Syntax:

```
typedef   char                ImageNameT[IMAGE_NAME_LEN];
```

3.1.11 Image Instance Number

ImageInstanceT

Syntax:

```
typedef   unsigned int        ImageInstanceT;
```

Description:

0-based value identifying image instance.

Range:

Depends on image type

3.1.12 Image Handle

ImageHandleT

Syntax:

```
typedef   unsigned int        ImageHandleT;
```

Range:

Full

3.1.13 Image Enumeration

ImageEnumT

Syntax:

```
typedef struct {
        ImageNameT                      imageName;
        ImageInstanceT                  imageInstance;
} ImageEnumT;
```

Members:

imageName

Formal name of the image

imageInstance

Instance number of the image

3.1.14 Return code

PropRetCodeT

Syntax:

```
typedef enum {
          PROP_NOERR,
          PROP_ERROR,
          PROP_INVPROP
} PropRetCodeT;
```

Members:

PROP_NOERR

Property retrieved correctly

PROP_ERROR

Error retrieving property

PROP_INVPROP

Invalid property for the image

3.1.15 Time Value

TimerT

Syntax:

```
typedef    uint64_t                    TimerT;
```

Description:

Time in nanoseconds from 00:00:00 January 1, 1970.

3.1.16 Image Role On Next Boot

ImageRoleNextbootT

Syntax:

```
typedef enum {
          IMAGE_NEXTBOOT_DEFAULT,
          IMAGE_NEXTBOOT_FALLBACK,
          IMAGE_NEXTBOOT_ONESHOT,
          IMAGE_NEXTBOOT_EMPTY,
          IMAGE_NEXTBOOT_OEM1,
          IMAGE_NEXTBOOT_OEM2,
          IMAGE_NEXTBOOT_OEM3,
          IMAGE_NEXTBOOT_OEM4,
          IMAGE_NEXTBOOT_OEM5
} ImageRoleNextbootT;
```

Members:

`IMAGE_NEXTBOOT_DEFAULT`

On next boot, this Image will be the default image to be activated.

`IMAGE_NEXTBOOT_FALLBACK`

Image will be fallback image on next boot. Thus, if default image fails to boot, this image will be automatically used. If additional fallbacks are to be designated, use OEMs defined states.

`IMAGE_NEXTBOOT_ONESHOT`

Image will be active on next boot. If it fails to load, the other image will be used next time.

IMAGE_NEXTBOOT_EMPTY

Image will be inactive on next boot.

IMAGE_NEXTBOOT_OEM1

OEM defined state

IMAGE_NEXTBOOT_OEM2

OEM defined state

IMAGE_NEXTBOOT_OEM3

OEM defined state

IMAGE_NEXTBOOT_OEM4

OEM defined state

IMAGE_NEXTBOOT_OEM5

OEM defined state

Description:

The enumeration defines possible values of IMAGE_ROLE_NEXTBOOT property.

3.1.17 Image Property

PropValueT

Syntax:

```
typedef struct {
        ImagePropEnumT                  type;
        union {
            ImageHandleT                imageHandle;
            char                        imageName;
            ImageInstanceT              imageInstance;
            int                         imageRequired;
            ImageCurrentStateT          imageCurrentState;
            ImageRoleNextbootT          imageRoleNextboot;
            unsigned int                imageStartAddr;
            unsigned int                imageEndAddr;
            unsigned int                imageLenMax;
            unsigned int                imageLenCur;
            char                        imageVersionString;
            unsigned int                imageVersionMajor;
            unsigned int                imageVersionMinor;
            unsigned int                imageVersionRevision;
            unsigned int                imageVersionBuild;
            unsigned int                imageSerialNumber;
            char                        imageTargetOs;
            char                        imageDescription;
            TimerT                      imageReleaseTime;
            TimerT                      imageInstallTime;
            TimerT                      imageActivateTime;
            ImageRestartEnumT           imageRestartType;
            ImageInstallTypeT           imageInstallType;
            unsigned int                singleImageInstall;
        } u;
} PropValueT;
```

Members:

type

The name of property to set/get.

imageHandle

Unique Handle for the image

Valid for selector: IMAGE_HANDLE

imageName

The name property defines the label by which the object is known.

Valid for selector: IMAGE_NAME

imageInstance

Instance number of image

Valid for selector: IMAGE_INSTANCE

imageRequired

Specifies whether this image is required for the proper functioning of the FRU.

Valid for selector: IMAGE_REQUIRED

imageCurrentState

Current state of image

Valid for selector: IMAGE_CURRENT_STATE

imageRoleNextboot

Role of image on next boot

Valid for selector: IMAGE_ROLE_NEXTBOOT

imageStartAddr

Pointer to start address. This is opaque.

Valid for selector: IMAGE_START_ADDR

imageEndAddr

Pointer to end address. This is opaque.

Valid for selector: IMAGE_END_ADDR

imageLenMax

Maximum size of image in bytes.

Valid for selector: IMAGE_LEN_MAX

imageLenCur

Length of current image in bytes.

Valid for selector: IMAGE_LEN_CUR

imageVersionString

A string representing the version info ? e.g, '12.1(3)T'. Since vastly different representations and semantics exist for versions, no interpretation is assumed.

Valid for selector: IMAGE_VERSION_STRING

imageVersionMajor

The major number component of the version information - for example, '12' from version 12.1.3.1234. This property is defined as a numeric value to allow the determination of 'newer' vs. 'older' releases. A 'newer' major release is indicated by a larger numeric value.

Valid for selector: IMAGE_VERSION_MAJOR

imageVersionMinor

The minor number component of the version. E.g, '1' from version 12.1.3.1251. This property is defined as a numeric value to allow the determination of 'newer' vs. 'older' releases. A 'newer' minor release is indicated by a larger numeric value.

Valid for selector: IMAGE_VERSION_MINOR

imageVersionRevision

The revision or maintenance release component of the version. E.g. '3' from version 12.1.3.1251. This property is defined as a numeric value to allow the determination of 'newer' vs. 'older' releases. A 'newer' revision is indicated by a larger numeric value.

Valid for selector: IMAGE_VERSION_REVISION

imageVersionBuild

The build number of the image. E.g. '1251' from version 12.1.3.1251. This property is defined as a numeric value to allow the determination of 'newer' vs. 'older' releases. A 'newer' build is indicated by a larger numeric value

Valid for selector: IMAGE_VERSION_BUILD

imageSerialNumber

A manufacturer-allocated number used to identify the software.

Valid for selector: IMAGE_SERIAL_NUMBER

imageTargetOs

Specifies the target operating systems of the software. Due to the extreme variability in operating systems, this property is defined as a string array.

Valid for selector: IMAGE_TARGET_OS

imageDescription

Textual Description of the Image

Valid for selector: IMAGE_DESCRIPTION

imageReleaseTime

A time value indicating when the image was released by the manufacturer. Lack of a value is acceptable.

Valid for selector: IMAGE_RELEASE_TIME

imageInstallTime

A time value indicating when the image was installed on this managed element. A lack of a value does not indicate that the image is not installed.

Valid for selector: IMAGE_INSTALL_TIME

imageActivateTime

A time value indicating when the image was activated.

Valid for selector: IMAGE_ACTIVATE_TIME

imageRestartType

The type of restart required to make this image active.

Valid for selector: IMAGE_RESTART_TYPE

imageInstallType

Required mode of installation.

Valid for selector: IMAGE_INSTALL_TYPE

singleImageInstall

OEM parameters. Valid values - 0 (false) or non-zero (true)

Valid for selector: SINGLE_IMAGE_INSTALL

3.1.18 Simgle Image Property Operation Arguments

ImagePropT

Syntax:

```
typedef struct {
        PropValueT                      propValue;
        PropRetCodeT                    propRetCode;
} ImagePropT;
```

Description:

The data structure gathers arguments for set/get property operation on single image.

Members:

propValue

Property value

propRetCode

Return code indicating success or err for set/get operation

3.1.19 Image Description

ImageDescriptionT

Syntax:

```
typedef   char   ImageDescriptionT[IMAGE_DESCRIPTION_LEN];
```

3.1.20 Traget OS Name

ImageTargetOS

Syntax:

```
typedef   char   ImageTargetOS[TARGET_OS_LEN];
```

3.1.21 Image Version String

ImageVersionT

Syntax:

```
typedef   char   ImageVersionT[IMAGE_VERSION_LEN];
```

3.1.22 Repository Image Path Name

RepositoryImagePathT

Syntax:

```
typedef   char   RepositoryImagePathT[REPOSITORY_IMAGE_PATH_LEN];
```

3.1.23 Image Verfification Result structure

ImageVerifyResultT

Syntax:

```
typedef struct {
        ImageHandleT                    imageHandle;
        VerifyRetCodeT                  verifyRetCode;
} ImageVerifyResultT;
```

Description:

Image Verification Result. Structure containing image verification result

Members:

imageHandle

Image Handle.

verifyRetCode

Image Verification Result.

3.2 API Functions

3.2.1 Create Destination Handle

CUCreateDestHandle

Syntax:

```
int
CUCreateDestHandle(
        DestAddrT                       destAddr,
        DestHandleT*                    destHandle);
```

Description:

Given a destination address, this API returns a unique handle for the destination. This unique destination handle is used in all subsequent calls related to that destination. The destination address is in the HPI Entity Path format and is the address of the field replaceable unit.

Parameters:

`destAddr` [in]

Address of destination in Entity Path format.

`destHandle` [inout]

Unique handle identifier for destination address.

Returns:

CU_SUCCESS

Success

CU_INVDESTADDRFORMAT

The destination address format is invalid.

CU_FRUNOTPRESENT

FRU not found - physical slot is empty.

CU_FRUTYPEINV

FRU type is invalid

CU_GENERALERR

General error, more specific information lacking.

3.2.2 Delete Destination Handle

CUDeleteDestHandle

Syntax:

```
int
CUDeleteDestHandle(
        DestHandleT                                    destHandle);
```

Description:

Given a destination handle, this API removes information about this destination handle. Any future references to this destination handle will be considered invalid.

Parameters:

`destHandle` [in]

Destination handle identifier.

Returns:

CU_SUCCESS

Success

CU_INVDESTHANDLE

No such destination handle exists

CU_GENERALERR

General error, more specific information lacking.

3.2.3 Get Image Handle

CUGetImageHandle

Syntax:

```
int
CUGetImageHandle(
        DestHandleT                     destHandle,
        ImageNameT                      imageName,
        ImageInstanceT                  imageInstance,
        ImageHandleT*                   imageHandle);
```

Description:

Given a destination handle, the image name and the instance number of the image, this function returns a unique handle for the image on that destination. This unique image handle is used in all subsequent calls related to that image. Note the image may or may not currently reside at the destination. The image name in the parameter list is the formal name of the image as opposed to the image filename.

Parameters:

`destHandle` [in]

Destination Handle

`imageName` [in]

Image Formal Name

`imageInstance` [in]

Instance number of the image

`imageHandle` [out]

Unique handle identifier for the image

Returns:

`CU_SUCCESS`

Success

`CU_INVDESTHANDLE`

The destination handle is invalid

`CU_INVIMAGENAME`

The image formal name is invalid

CU_INVIMAGEINSTANCE

The image instance is invalid

3.2.4 Load Image

CULoadImage

Syntax:

```
int
CULoadImage(
        DestHandleT                     destHandle,
        ImageHandleT                    imageHandle,
        RepositoryImagePathT            repositoryImagePath,
        unsigned int                    numRepositoryProps,
        ImagePropT*                     repositoryImageProperties);
```

Description:

This API takes the image from the specified repository location and writes it to the specified target. "Writing" the image may include special actions such as uncompressing the image etc. The target address is specified by DestHandle and ImageHandle pair. DestHandle can be retrieved by a call to CUCreateDestHandle() and the ImageHandle can be retrieved by a call to CUGetImageHandle(). The format of the repository address must be a valid URI as defined in RFC 2396 [6] and must include the file name of the image to be retrieved. The URI specified may contain a scheme which indicates the explicit service and location to be used to retrieve the binary image (e.g. ftp, http etc.).

Parameters:

destHandle [in]

Destination Handle

imageHandle [in]

Image Handle

repositoryImagePath [in]

Address of image repository including image filename

numRepositoryProps [in]

Number of properties of repository image

repositoryImageProperties [in]

Pointer to an array of size NumRepositoryProps that contains read-only properties of repository image

Returns:

CU_SUCCESS

Success

CU_ERRINVSRC

The source address or the binary name of the image is invalid.

CU_ERRINVTYPE

The type of image is invalid

CU_ERRINVPROT

The requested service protocol is not supported.

CU_ERRSRCUNREACHABLE

The source address is unreachable.

CU_ERRSRCCORRUPTED

The image at the source address is corrupted.

CU_ERRDESTACTIVE

The destination image is currently active.

CU_ERRFRUNOTPRESENT

The FRU is no longer present.

3.2.5 Enumerate Images

CUEnumerateImages

Syntax:

```
int
CUEnumerateImages(
        DestHandleT                     destHandle,
        unsigned int*                   numImages,
        ImageEnumT**                    imageEnum);
```

Description:

This API returns all the image names and their instances on the destination. The first parameter, DestHandle is the unique identifier for the destination and is retrieved by a call to CUCreateDestHandle(). The second parameter, NumImages, indicates the size of the ImageEnum array allocated by the caller of the function. However, when the function call returns, the NumImages parameter is updated with the number of images on the destination. If the number of images on the destination is less than or equal to the size of the array allocated, then the ImageEnum array is populated. If the number of images on the destination is greater than the size of the array allocated, the ImageEnum array is not populated. An error is returned in this case and the value of NumImages is parameter is updated with the number of images on the destination. The contents of the ImageEnum array are undefined in this case. The third parameter is a pointer to an array of size NumImages of type ImageEnum. The ImageEnum structure contains the Image name and the instance number of the image.

Parameters:

`destHandle` [in]

Destination Handle

`numImages` [inout]

Number of images

`imageEnum` [inout]

Pointer to ImageEnum array of size NumImages

Returns:

`CU_SUCCESS`

Success

`CU_INSUFFSIZE`

The size of array allocated is insufficient to store image names.

CU_ERRFRUNOTPRESENT

FRU notpresent

3.2.6 Enumerate Image Properties

CUEnumerateImageProperties

Syntax:

```
int
CUEnumerateImageProperties(
        DestHandleT                     destHandle,
        ImageHandleT                    imageHandle,
        unsigned int*                   numProps,
        ImagePropEnumT**                imagePropEnum);
```

Description:

This function returns all the property names defined for a specific target image on the destination. The first parameter, DestHandle is the unique identifier for the destination and is retrieved by a call to CUCreateDestHandle(). The second parameter is the unique handle for the image on the destination and is retrieved by a call to CUGetImageHandle(). The third parameter, pNumProps, indicates the size of the ImagePropEnum array allocated by the caller of the function. However, when the function call returns, the NumProps parameter is updated with the number of properties associated with that image-destination. If the number of properties of the image on the destination is less than or equal to the size of the array allocated, then the ImagePropEnum array is populated. If the number of properties on the image-destination is greater than the size of the array allocated, the ImagePropEnum array is not populated. An error is returned in this case. The fourth parameter is a pointer to an array of size pNumProps of type ImagePropEnum.

Parameters:

destHandle [in]

Destination Handle

imageHandle [in]

Image Handle

numProps [inout]

Number of properties

imagePropEnum [inout]

Pointer to ImageProp array of size NumProps

Returns:

CU_SUCCESS

Success

CU_INSUFFSIZE

The size of array allocated is insufficient to store property names.

3.2.7 Get Image Properties

CUGetImageProperties

Syntax:

```
int
CUGetImageProperties(
        DestHandleT                     destHandle,
        ImageHandleT                    imageHandle,
        unsigned int*                   numProps,
        ImagePropT**                    imageProp);
```

Description:

This function returns requested properties about a specific image stored on the destination. The first parameter, DestHandle is the unique identifier for the destination and is retrieved by a call to CUCreateDestHandle(). The second parameter, ImageHandle is the unique identifier for a specific image on the destination and is retrieved by a call to CUGetImageHandle(). Numprops indicates the number of properties to retrieve. Additionally, the caller of this function is responsible for allocating an array that contains pointers to where the image properties and status are returned. When the function returns, the caller should first check the PropRetCode for each property to determine if the Image property was returned correctly. If the property was not returned correctly, the caller should not process the contents of the property store. The PropRetCode may indicate the following return codes: PROP_NOERR - No error in retrieving properties PROP_INVPROP - Invalid property for the image

Parameters:

destHandle [in]

Destination Handle

imageHandle [in]

Image Handle

numProps [inout]

Number of properties

imageProp [inout]

Pointer to ImageProp array of size NumProps

Returns:

CU_SUCCESS

Success

CU_GETPROPERR

Error in retrieving properties

CU_GETPROPPARTIAL

Partial success in retrieving properties

3.2.8 Set Image Property

CUSetImageProperty

Syntax:

```
int
CUSetImageProperty(
        DestHandleT                     destHandle,
        ImageHandleT                    imageHandle,
        unsigned int*                   numProps,
        ImagePropT**                    imageProp);
```

Description:

This function sets the requested properties about a specific image stored on the destination. The first parameter, DestHandle is the unique identifier for the destination and is retrieved by a call to CUCreateDestHandle(). The second parameter, ImageHandle is the unique identifier for a specific image on the destination and is retrieved by a call to CUGetImageHandle(). Numprops indicates the number of properties that will be set in this call. Additionally, the caller of this function is responsible for allocating an array that contains pointers to where the image properties to be set are stored. If one of the properties passed in for modification is not valid, none of the properties must be changed. And error code, CU_PROPNOTSET, indicating that the function call was not successful is returned. In order to find out which property is the offending one and the reason for error, the PropRetCode member of the ImageProp structure must be consulted.

Parameters:

destHandle [in]

Destination Handle

imageHandle [in]

Image Handle

numProps [inout]

Number of properties

imageProp [inout]

Pointer to ImageProp array of size NumProps

Returns:

CU_SUCCESS

Success

CU_PROPNOTSET

Properties not set due to error

3.2.9 Lock Images

CULockImages

Syntax:

```
int
CULockImages(
        DestHandleT                                    destHandle);
```

Description:

This function locks the images on a specified destination. This is required to ensure atomicity when image properties are changed. Thus, for example, if the default image is to be changed from one instance to another, this function must be called first. Following a call to this function, the image properties for the 2 images may be changed sequentially and then the images unlocked. While the images are locked on an FRU, no other CU function may be called on that FRU. The parameter, DestHandle, is the unique identifier for the destination and is retrieved by a call to CUCreateDestHandle().

Parameters:

destHandle [in]

Destination Handle

Returns:

CU_SUCCESS

Success

CU_IMAGELOCKED

Images on FRU are already locked, try again later.

3.2.10 Unlock Images

CUUnlockImages

Syntax:

```
int
CUUnlockImages(
        DestHandleT                                    destHandle);
```

Description:

This function unlocks the images on a specified destination. This operation is complementary to the lock operation. The lock/unlock pair is required to ensure atomicity when image properties are changed. Thus, for example, if the default image is to be changed from one instance to another, this function must be called first. Following a call to this function, the image properties for the 2 images may be changed sequentially and then the images unlocked. The parameter, DestHandle, is the unique identifier for the destination and is retrieved by a call to CUCreateDestHandle().

Parameters:

destHandle [in]

Destination Handle

Returns:

CU_SUCCESS

Success

CU_IMAGENOTLOCKED

Image not locked by the handle.

3.2.11 Verify Images

CUVerifyImages

Syntax:

```
int
CUVerifyImages(
        DestHandleT                             destHandle,
        unsigned int*                           numImages,
        ImageVerifyResultT**                    imageVerifyResult);
```

Description:

This function validates the properties of the images in the image store. The validation that is done as part of this API call is implementation specific. The first parameter, DestHandle is the unique identifier for the destination and is retrieved by a call to CUCreateDestHandle(). The second parameter is a pointer to a variable NumImages. The variable indicates the size of the ImageVerifyResult array allocated by the caller of the function. However, when the function call returns, the NumImages parameter is updated with the number of images on the destination. If the number of images on the destination is less than or equal to the size of the array allocated, then the ImageVerifyResult array is populated. If the number of images on the destination is greater than the size of the array allocated, the ImageVerifyResult array is not populated. An error, CU_INSUFFSIZE, is returned in this case. The third parameter is a pointer to an array of size NumImages of type ImageVerifyResultT. The ImageVerifyResult structure contains the Image Handle and the verification result code for that image.

Parameters:

`destHandle` [in]

Destination Handle

`numImages` [inout]

Number of images

`imageVerifyResult` [inout]

Pointer to ImageVerifyResult array of size NumImages

Returns:

`CU_SUCCESS`

Success

`CU_INSUFFSIZE`

The size of array allocated is insufficient to store image names.

CU_IMAGEVERIFYERR

Error in image verification - see error associated with each image for details

3.2.12 Image Restart

CUImageRestart

Syntax:

```
int
CUImageRestart(
        DestHandleT                     destHandle,
        ImageHandleT                    imageHandle);
```

Description:

If the restart granularity of the image is just the image under consideration and the image can be restarted without affecting any of the ongoing operations on the system, then this API results in an action. If the restart granularity is an FRU or multiple images or effects the ongoing operations on the FRU, then this API must return CU_RESTARTNOTSUPPORTED. If this function results in an image restart, this function should initiate the restart action and immediately return. The restart should occur after this call is completed.

Parameters:

destHandle [in]

Destination Handle

imageHandle [in]

Image handle

Returns:

CU_RESTARTINITIATED

Restart action initiated.

CU_RESTARTNOTSUPPORTED

Image restart is not supported.

4 Alarm Monitoring API

4.1 Data Structures

4.1.1 Health Event Severities

ShmOamEventSeverity

Syntax:

```
typedef enum {
         SHM_OAM_EVENT_SEVERITY_CRITICAL,
         SHM_OAM_EVENT_SEVERITY_MAJOR,
         SHM_OAM_EVENT_SEVERITY_MINOR,
         SHM_OAM_EVENT_SEVERITY_NORMAL
} ShmOamEventSeverity;
```

Members:

SHM_OAM_EVENT_SEVERITY_CRITICAL

Critical severity

SHM_OAM_EVENT_SEVERITY_MAJOR

Major severity

SHM_OAM_EVENT_SEVERITY_MINOR

Minor severity

SHM_OAM_EVENT_SEVERITY_NORMAL

Normal severity

Description:

The type defines health event severity levels specified in PICMG.

4.1.2 Fault LED Color

ShmOamFaultLEDColor

Syntax:

```
typedef enum {
          SHM_OAM_FAULT_LED_RED,
          SHM_OAM_FAULT_LED_AMBER
} ShmOamFaultLEDColor;
```

Members:

SHM_OAM_FAULT_LED_RED

Red color

SHM_OAM_FAULT_LED_AMBER

Green color

Description:

The type defines fault LED colors.

4.1.3 Compatibility mode

ShmOamCompatibility

Syntax:

```
typedef enum {
          SHM_OAM_COMP_5_2,
          SHM_OAM_COMP_6_1
} ShmOamCompatibility;
```

Members:

SHM_OAM_COMP_5_2

Timnath 5.2 compatibility mode

SHM_OAM_COMP_6_1

Timnath 6.1 compatibility mode

Description:

The type defines compatibility mode values.

4.1.4 Event Description Format

ShmOamEventDescrFormat

Syntax:

```
typedef enum {
          SHM_OAM_DESCR_FORMAT_SEL,
          SHM_OAM_DESCR_FORMAT_HEALTH,
          SHM_OAM_DESCR_FORMAT_SNMP_TRAP
} ShmOamEventDescrFormat;
```

Members:

SHM_OAM_DESCR_FORMAT_SEL

format presented in SEL

SHM_OAM_DESCR_FORMAT_HEALTH

format presented in health commands

SHM_OAM_DESCR_FORMAT_SNMP_TRAP

format presented in snmp traps descriptions

Description:

Type defines format of event description depending on presentation method, SEL, health commands, snmp traps

4.1.5 Alarm Monitor Health Event

ShmOamAlarmMonitorHealthEvent

Syntax:

```
typedef struct {
        ShmSelRecordT                   selRecord;
        ShmBoolT                        acknowledged;
} ShmOamAlarmMonitorHealthEvent;
```

Members:

selRecord

SEL Record containing detailed information about the event.

acknowledged

Indicates whether the event has been acknowledged.

4.2 API Functions

4.2.1 Acknowledge Health Events

shmOamAlarmMonitorHealthEventsAcknowledge

Syntax:

```
ShmErrorT
shmOamAlarmMonitorHealthEventsAcknowledge(
        ShmSessionIdT                   sessionId,
        ShmOamEventSeverity             severity);
```

Description:

The function acknowledges (a.k.a. clears) all events of a specified severity.

Parameters:
`sessionId [in]`

Session ID identifying target LISM
`severity [in]`

Severity of the events to be acknowledged

4.2.2 Acknowledge IPMC Health Events

shmOamAlarmMonitorIpmcHealthEventsAcknowledge

Syntax:

```
ShmErrorT
shmOamAlarmMonitorIpmcHealthEventsAcknowledge(
          ShmSessionIdT                                 sessionId,
          ShmIpmcAddrT                                  ipmc);
```

Description:

The function acknowledges (a.k.a. clears) all events of a specified severity.

Parameters:
`sessionId [in]`

Session ID identifying target LISM
`ipmc [in]`

IPMB address identifying the IPMC.

Returns:

`SHM_CC_OK`

Success

`SHM_CC_INVALID_PARAMETER`

Invalid Parameter

`SHM_CC_ILLEGAL_CMD_FOR_HASTATE`

The LISM instance is not active ShMC.

4.2.3 Acknowledge IPMC Health Events

shmOamAlarmMonitorFruHealthEventsAcknowledge

Syntax:

```
ShmErrorT
shmOamAlarmMonitorFruHealthEventsAcknowledge(
          ShmSessionIdT                                 sessionId,
          ShmIpmcAddrT                                  ipmc,
          ShmFruDevIdT                                  fruId);
```

Description:

The function acknowledges (a.k.a. clears) all events of a specified FRU.

Parameters:

`sessionId [in]`

Session ID identifying target LISM

`ipmc [in]`

IPMB address identifying the IPMC.

`fruId [in]`

FRU ID identifying the FRU

Returns:

`SHM_CC_OK`

Success

`SHM_CC_INVALID_PARAMETER`

Invalid Parameter

`SHM_CC_ILLEGAL_CMD_FOR_HASTATE`

The LISM instance is not active ShMC.

4.2.4 Acknowledge IPMC Health Events

shmOamAlarmMonitorSensorHealthEventsAcknowledge

Syntax:

```
ShmErrorT
shmOamAlarmMonitorSensorHealthEventsAcknowledge(
        ShmSessionIdT                                   sessionId,
        ShmIpmcAddrT                                    ipmc,
        ShmIpmiLunT                                     lun,
        ShmSensorNumT                                   sensorNumber);
```

Description:

The function acknowledges (a.k.a. clears) all events of a specified sensor.

Parameters:

`sessionId [in]`

Session ID identifying target LISM

`ipmc [in]`

IPMB address identifying the IPMC.

`lun [in]`

LUN identifying the device hosting the sensor.

`sensorNumber [in]`

Sensor number identifying the sensor

Returns:

`SHM_CC_OK`

Success

`SHM_CC_INVALID_PARAMETER`

Invalid Parameter

`SHM_CC_ILLEGAL_CMD_FOR_HASTATE`

The LISM instance is not active ShMC.

4.2.5 Get Total Number of Health Events

shmOamAlarmMonitorAllEventsNumberGet

Syntax:

```
ShmErrorT
shmOamAlarmMonitorAllEventsNumberGet(
        ShmSessionIdT                   sessionId,
        unsigned int*                   criticalEventNum,
        unsigned int*                   criticalEventAckNum,
        unsigned int*                   majorEventNum,
        unsigned int*                   majorEventAckNum,
        unsigned int*                   minorEventNum,
        unsigned int*                   minorEventAckNum);
```

Description:

The function returns the number of all health events active on all entities in the shelf an the number of acknowledged events.

Parameters:

`sessionId [in]`

Session ID identifying target LISM

`criticalEventNum [out]`

The function sets the argument to the number of active critical events

`criticalEventAckNum [out]`

The function sets the argument to the number of acknowledged critical events

`majorEventNum [out]`

The function sets the argument to the number of active major events

`majorEventAckNum [out]`

The function sets the argument to the number of acknowledged major events

`minorEventNum [out]`

The function sets the argument to the number of active minor events

`minorEventAckNum [out]`

The function sets the argument to the number of acknowledged minor events

Returns:

`SHM_CC_OK`

Success

SHM_CC_INVALID_PARAMETER

Invalid Parameter

4.2.6 Get Number of Health Events on IPMC

shmOamAlarmMonitorIpmcEventsNumberGet

Syntax:

```
ShmErrorT
shmOamAlarmMonitorIpmcEventsNumberGet(
        ShmSessionIdT                   sessionId,
        ShmIpmcAddrT                    ipmc,
        unsigned int*                   criticalEventNum,
        unsigned int*                   criticalEventAckNum,
        unsigned int*                   majorEventNum,
        unsigned int*                   majorEventAckNum,
        unsigned int*                   minorEventNum,
        unsigned int*                   minorEventAckNum);
```

Description:

The function returns the number of health events active on an IPMC and acknowledged health events.

Parameters:

`sessionId [in]`

Session ID identifying target LISM

`ipmc [in]`

IPMB address identifying the IPMC.

`criticalEventNum [out]`

The function sets the argument to the number of active critical events

`criticalEventAckNum [out]`

The function sets the argument to the number of acknowledged critical events

`majorEventNum [out]`

The function sets the argument to the number of active major events

`majorEventAckNum [out]`

The function sets the argument to the number of acknowledged major events

`minorEventNum [out]`

The function sets the argument to the number of active minor events

minorEventAckNum [out]

The function sets the argument to the number of acknowledged minor events

Returns:

SHM_CC_OK

Success

SHM_CC_INVALID_PARAMETER

Invalid Parameter

4.2.7 Get Number of Health Events on FRU

shmOamAlarmMonitorFruEventsNumberGet

Syntax:

```
ShmErrorT
shmOamAlarmMonitorFruEventsNumberGet(
        ShmSessionIdT                   sessionId,
        ShmIpmcAddrT                    ipmc,
        ShmFruDevIdT                    fruId,
        unsigned int*                   criticalEventNum,
        unsigned int*                   criticalEventAckNum,
        unsigned int*                   majorEventNum,
        unsigned int*                   majorEventAckNum,
        unsigned int*                   minorEventNum,
        unsigned int*                   minorEventAckNum);
```

Description:

The function returns the number of health events active on a FRU and the number of acknowledged health events.

Parameters:

sessionId [in]

Session ID identifying target LISM

ipmc [in]

IPMB address identifying the IPMC.

fruId [in]

FRU ID identifying the FRU

criticalEventNum [out]

The function sets the argument to the number of active critical events

`criticalEventAckNum [out]`

The function sets the argument to the number of acknowledged critical events

`majorEventNum [out]`

The function sets the argument to the number of active major events

`majorEventAckNum [out]`

The function sets the argument to the number of acknowledged major events

`minorEventNum [out]`

The function sets the argument to the number of active minor events

`minorEventAckNum [out]`

The function sets the argument to the number of acknowledged minor events

Returns:

SHM_CC_OK

Success

SHM_CC_INVALID_PARAMETER

Invalid Parameter

4.2.8 Get Number of Health Events on Sensor

shmOamAlarmMonitorSensorEventsNumberGet

Syntax:

```
ShmErrorT
shmOamAlarmMonitorSensorEventsNumberGet(
       ShmSessionIdT                          sessionId,
       ShmIpmcAddrT                           ipmc,
       ShmIpmiLunT                            lun,
       ShmSensorNumT                          sensorNumber,
       unsigned int*                          criticalEventNum,
       unsigned int*                          criticalEventAckNum,
       unsigned int*                          majorEventNum,
       unsigned int*                          majorEventAckNum,
       unsigned int*                          minorEventNum,
       unsigned int*                          minorEventAckNum);
```

Description:

The function returns the number of health events active on a sensor and the number of acknowledged events.

Parameters:

`sessionId [in]`

Session ID identifying target LISM

`ipmc [in]`

IPMB address identifying the IPMC.

`lun [in]`

LUN identifying the device hosting the sensor.

`sensorNumber [in]`

Sensor number identifying the sensor

`criticalEventNum [out]`

The function sets the argument to the number of active critical events

`criticalEventAckNum [out]`

The function sets the argument to the number of acknowledged critical events

`majorEventNum [out]`

The function sets the argument to the number of active major events

`majorEventAckNum [out]`

The function sets the argument to the number of acknowledged major events

`minorEventNum [out]`

The function sets the argument to the number of active minor events

`minorEventAckNum [out]`

The function sets the argument to the number of acknowledged minor events

Returns:

`SHM_CC_OK`

Success

`SHM_CC_INVALID_PARAMETER`

Invalid Parameter

4.2.9 Get All Active Health Events

shmOamAlarmMonitorAllEventsGet

Syntax:

```
ShmErrorT
shmOamAlarmMonitorAllEventsGet(
        ShmSessionIdT                           sessionId,
        unsigned int*                           criticalEventNum,
        ShmOamAlarmMonitorHealthEvent**         criticalEvents,
        unsigned int*                           majorEventNum,
        ShmOamAlarmMonitorHealthEvent**         majorEvents,
        unsigned int*                           minorEventNum,
        ShmOamAlarmMonitorHealthEvent**         minorEvents);
```

Description:

The function returns health events active on all entities in the shelf.

Parameters:

`sessionId [in]`

Session ID identifying target LISM

`criticalEventNum [out]`

The function sets the argument to the number of active critical events

`criticalEvents [out]`

The function sets the argument to pointer to an array of active critical health events. The table is allocated by the function and should be freed using function ShmOamAla.

`majorEventNum [out]`

The function sets the argument to the number of active major events

`majorEvents [out]`

The function sets the argument to pointer to an array of active major health events. The table is allocated by the function and should be freed using function shmOamAlarmMonitorAllEventsGet_free.

`minorEventNum [out]`

The function sets the argument to the number of active minor events

`minorEvents [out]`

The function sets the argument to pointer to an array of active minor health events. The table is allocated by the function and should be freed using function shmOamAlarmMonitorAllEventsGet_free..

Returns:

`SHM_CC_OK`

Success

`SHM_CC_INVALID_PARAMETER`

Invalid Parameter

4.2.10 Get Active Health Events on IPMC

shmOamAlarmMonitorIpmcEventsGet

Syntax:

```
ShmErrorT
shmOamAlarmMonitorIpmcEventsGet(
        ShmSessionIdT                       sessionId,
        ShmIpmcAddrT                        ipmc,
        unsigned int*                       criticalEventNum,
        ShmOamAlarmMonitorHealthEvent**     criticalEvents,
        unsigned int*                       majorEventNum,
        ShmOamAlarmMonitorHealthEvent**     majorEvents,
        unsigned int*                       minorEventNum,
        ShmOamAlarmMonitorHealthEvent**     minorEvents);
```

Description:

The function returns active health events for an IPMC.

Parameters:

`sessionId [in]`

Session ID identifying target LISM

`ipmc [in]`

IPMB address identifying the IPMC.

`criticalEventNum [out]`

The function sets the argument to the number of active critical events

`criticalEvents [out]`

The function sets the argument to pointer to an array of active critical health events. The table is allocated by the function and should be freed using function shmOamAlarmMonitorIpmcEventsGet_free.

`majorEventNum [out]`

The function sets the argument to the number of active major events

`majorEvents [out]`

The function sets the argument to pointer to an array of active major health events. The table is allocated by the function and should be freed using function shmOamAlarmMonitorIpmcEventsGet_free.

`minorEventNum [out]`

The function sets the argument to the number of active minor events

`minorEvents [out]`

The function sets the argument to pointer to an array of active minor health events. The table is allocated by the function and should be freed using function shmOamAlarmMonitorIpmcEventsGet_free.

Returns:

SHM_CC_OK

Success

SHM_CC_INVALID_PARAMETER

Invalid Parameter

4.2.11 Get Active Health Events for FRU

shmOamAlarmMonitorFruEventsGet

Syntax:

```
ShmErrorT
shmOamAlarmMonitorFruEventsGet(
        ShmSessionIdT                           sessionId,
        ShmIpmcAddrT                            ipmc,
        ShmFruDevIdT                            fruId,
        unsigned int*                           criticalEventNum,
        ShmOamAlarmMonitorHealthEvent**         criticalEvents,
        unsigned int*                           majorEventNum,
        ShmOamAlarmMonitorHealthEvent**         majorEvents,
        unsigned int*                           minorEventNum,
        ShmOamAlarmMonitorHealthEvent**         minorEvents);
```

Description:

The function returns the health events active on a FRU.

Parameters:

`sessionId [in]`

Session ID identifying target LISM

`ipmc` `[in]`

IPMB address identifying the IPMC.

`fruId` `[in]`

FRU ID identifying the FRU

`criticalEventNum [out]`

The function sets the argument to the number of active critical events

`criticalEvents [out]`

The function sets the argument to pointer to an array of active critical health events. The table is allocated by the function and should be freed using function shmOamAlarmMonitorFruEventsGet_free.

`majorEventNum [out]`

The function sets the argument to the number of active major events

`majorEvents [out]`

The function sets the argument to pointer to an array of active major health events. The table is allocated by the function and should be freed using function shmOamAlarmMonitorFruEventsGet_free.

`minorEventNum [out]`

The function sets the argument to the number of active minor events

`minorEvents [out]`

The function sets the argument to pointer to an array of active minor health events. The table is allocated by the function and should be freed using function shmOamAlarmMonitorFruEventsGet_free.

Returns:

`SHM_CC_OK`

Success

`SHM_CC_INVALID_PARAMETER`

Invalid Parameter

4.2.12 Get Active Health Events for Sensor

shmOamAlarmMonitorSensorEventsGet

Syntax:

```
ShmErrorT
shmOamAlarmMonitorSensorEventsGet(
        ShmSessionIdT                           sessionId,
        ShmIpmcAddrT                            ipmc,
        ShmIpmiLunT                             lun,
        ShmSensorNumT                           sensorNumber,
        unsigned int*                           criticalEventNum,
        ShmOamAlarmMonitorHealthEvent**         criticalEvents,
        unsigned int*                           majorEventNum,
        ShmOamAlarmMonitorHealthEvent**         majorEvents,
        unsigned int*                           minorEventNum,
        ShmOamAlarmMonitorHealthEvent**         minorEvents);
```

Description:

The function returns the health events active on a sensor.

Parameters:

`sessionId [in]`

Session ID identifying target LISM

`ipmc [in]`

IPMB address identifying the IPMC.

`lun [in]`

LUN identifying the device hosting the sensor.

`sensorNumber [in]`

Sensor number identifying the sensor

`criticalEventNum [out]`

The function sets the argument to the number of active critical events

`criticalEvents [out]`

The function sets the argument to pointer to an array of active critical health events. The table is allocated by the function and should be freed using function shmOamAlarmMonitorSensorEventsGet_free.

`majorEventNum [out]`

The function sets the argument to the number of active major events

`majorEvents [out]`

The function sets the argument to pointer to an array of active major health events. The table is allocated by the function and should be freed using function shmOamAlarmMonitorSensorEventsGet_free.

`minorEventNum [out]`

The function sets the argument to the number of active minor events

`minorEvents [out]`

The function sets the argument to pointer to an array of active minor health events. The table is allocated by the function and should be freed using function shmOamAlarmMonitorSensorEventsGet_free.

Returns:

`SHM_CC_OK`

Success

`SHM_CC_INVALID_PARAMETER`

Invalid Parameter

4.2.13 Get SEL Record Description

shmOamAlarmMonitorEventDescriptionGet

Syntax:

```
ShmErrorT
shmOamAlarmMonitorEventDescriptionGet(
        ShmSessionIdT                   sessionId,
        ShmSelRecordT                   selRecord,
        ShmOamEventDescrFormat          descrFormat,
        char**                          eventDescr,
        unsigned short*                 eventCode);
```

Description:

The function returns description for an SEL record. It handles both events contributing to health and not contributing to health. It also handles OEM SEL records.

Parameters:

`sessionId [in]`

Session ID identifying target LISM

`selRecord [in]`

SEL record.

descrFormat [in]

Requested description format

eventDescr [out]

The function sets the argument to pointer to a zero-terminated string describing the event.

eventCode [out]

The function sets the argument to event code.

4.2.14 Get compatibility mode

shmOamAlarmMonitorCompatibilityModeGet

Syntax:

```
ShmErrorT
shmOamAlarmMonitorCompatibilityModeGet(
          ShmSessionIdT                          sessionId,
          ShmOamCompatibility*                   mode);
```

Description:

The function returns current compatibility mode.

Parameters:

sessionId [in]

Session ID identifying target LISM

mode [out]

The function sets the argument to compatibility mode.

4.2.15 Set compatibility mode

shmOamAlarmMonitorCompatibilityModeSet

Syntax:

```
ShmErrorT
shmOamAlarmMonitorCompatibilityModeSet(
          ShmSessionIdT                          sessionId,
          ShmOamCompatibility                    mode);
```

Description:

The function configures compatibility mode.

Parameters:

`sessionId [in]`

Session ID identifying target LISM

`mode [in]`

The function sets the compatibility

5 Telco Alarm Device Management API

5.1 Data Types definitions

5.1.1 Telco Alarm Severities

ShmOamTadmAlarmSeverity

Syntax:

```
typedef enum {
        SHM_OAM_TADM_ALARM_SEVERITY_CRITICAL,
        SHM_OAM_TADM_ALARM_SEVERITY_MAJOR,
        SHM_OAM_TADM_ALARM_SEVERITY_MINOR
} ShmOamTadmAlarmSeverity;
```

Members:

`SHM_OAM_TADM_ALARM_SEVERITY_CRITICAL`

Critical alarm

`SHM_OAM_TADM_ALARM_SEVERITY_MAJOR`

Major alarm

`SHM_OAM_TADM_ALARM_SEVERITY_MINOR`

Minor alarm

Description:

The type defines alarm severity levels that can be announced using telco alarm output lines.

5.1.2 General Defines

Syntax:

```
#define  SHM_OAM_TDM_MAX_DEV_COUNT  8
```

Members:

SHM_OAM_TDM_MAX_DEV_COUNT

Maximum number of Telco Alarm Devices in a chassis

Description:

General Defines.

5.2 API Functions

5.2.1 Get Telco Alarm Devices

shmOamTadmTelcoAlarmDevicesGet

Syntax:

```
ShmErrorT
shmOamTadmTelcoAlarmDevicesGet(
        ShmSessionIdT                   sessionId,
        unsigned int*                   ipmcCount,
        ShmIpmcAddrT**                  ipmcAddrs);
```

Description:

The function returns a table of known Telco Alarm Devices.

Parameters:

sessionId [in]

Session ID identifying target LISM

ipmcCount [out]

Number of entries in the above table.

ipmcAddrs [out]

Table of addresses of IPMCs hosting the telco alarm devices.

Returns:

SHM_CC_OK

Success

SHM_CC_INVALID_PARAMETER

Invalid Parameter

5.2.2 Set Telco Alarm Output

shmOamTadmTelcoAlarmOutputSet

Syntax:

```
ShmErrorT
shmOamTadmTelcoAlarmOutputSet(
        ShmSessionIdT                   sessionId,
        ShmIpmcAddrT                    ipmcAddr,
        ShmOamTadmAlarmSeverity         alarmSeverity,
        ShmBoolT                        alarmState);
```

Description:

The function sets state of telco alarm output line of a defined severity and on a defined telco alarm device.

Parameters:

sessionId [in]

Session ID identifying target LISM

ipmcAddr [in]

Address of IPMC hosting the telco alarm device.

alarmSeverity [in]

Alarm severity identifying the output line to be set

alarmState [in]

State of the line to be set

Returns:

SHM_CC_OK

Success

SHM_CC_INVALID_PARAMETER

Invalid Parameter

SHM_CC_ILLEGAL_CMD_FOR_HASTATE

The LISM instance is not active ShMC.

5.2.3 Get Telco Alarm Output

shmOamTadmTelcoAlarmOutputGet

Syntax:

```
ShmErrorT
shmOamTadmTelcoAlarmOutputGet(
        ShmSessionIdT                   sessionId,
        ShmIpmcAddrT                    ipmcAddr,
        ShmOamTadmAlarmSeverity         alarmSeverity,
        ShmBoolT*                       alarmState);
```

Description:

The function returns state of telco alarm output line of a defined severity and on a defined telco alarm device.

Parameters:

sessionId [in]

Session ID identifying target LISM

ipmcAddr [in]

Address of IPMC hosting the telco alarm device.

alarmSeverity [in]

Alarm severity identifying the output line

alarmState [out]

Current state of the line

Returns:

SHM_CC_OK

Success

SHM_CC_INVALID_PARAMETER

Invalid Parameter

5.2.4 Set Telco Alarm Silence State

shmOamTadmTelcoAlarmSilenceStateSet

Syntax:

```
ShmErrorT
shmOamTadmTelcoAlarmSilenceStateSet(
        ShmSessionIdT                    sessionId,
        ShmIpmcAddrT                     ipmcAddr,
        ShmBoolT                         silenceState);
```

Description:

The function sets state of telco alarm silence on a defined telco alarm device.

Parameters:

`sessionId` [in]

Session ID identifying target LISM

`ipmcAddr` [in]

Address of IPMC hosting the telco alarm device

`silenceState` [in]

If the value is TRUE, the silence state is activated. Otherwise, it alarm annunciation is restored.

Returns:

`SHM_CC_OK`

Success

`SHM_CC_INVALID_PARAMETER`

Invalid Parameter

5.2.5 Get Telco Alarm Silence State

shmOamTadmTelcoAlarmSilenceStateGet

Syntax:

```
ShmErrorT
shmOamTadmTelcoAlarmSilenceStateGet(
        ShmSessionIdT                   sessionId,
        ShmIpmcAddrT                    ipmcAddr,
        ShmBoolT*                       silenceState);
```

Description:

The function returns state of telco alarm silence on a defined telco alarm device.

Parameters:

sessionId [in]

Session ID identifying target LISM

ipmcAddr [in]

Address of IPMC hosting the telco alarm device

silenceState [out]

If the value is TRUE, the silence state is active. Otherwise, it alarm annunciation is enabled.

Returns:

SHM_CC_OK

Success

SHM_CC_INVALID_PARAMETER

Invalid Parameter

5.2.6 Set Telco Alarm Silence Timeout

shmOamTadmTelcoAlarmSilenceTimeoutSet

Syntax:

```
ShmErrorT
shmOamTadmTelcoAlarmSilenceTimeoutSet(
        ShmSessionIdT                           sessionId,
        unsigned short                          silenceTimeout);
```

Description:

The function sets value of telco alarm silence timeout parameter.

Parameters:

`sessionId` [in]

Session ID identifying target LISM

`silenceTimeout` [in]

New silence timeout in seconds

Returns:

SHM_CC_OK

Success

SHM_CC_INVALID_PARAMETER

Invalid Parameter

SHM_CC_ILLEGAL_CMD_FOR_HASTATE

The LISM instance is not active ShMC.

5.2.7 Get Telco Alarm Silence Timeout

shmOamTadmTelcoAlarmSilenceTimeoutGet

Syntax:

```
ShmErrorT
shmOamTadmTelcoAlarmSilenceTimeoutGet(
        ShmSessionIdT                           sessionId,
        unsigned short*                         silenceTimeout);
```

Description:

The function returns value of telco alarm silence timeout parameter.

Parameters:

`sessionId` [in]

Session ID identifying target LISM

`silenceTimeout` [out]

Silence timeout in seconds

Returns:

`SHM_CC_OK`

Success

`SHM_CC_INVALID_PARAMETER`

Invalid Parameter

6 SNMP Configuration Management API

6.1 API functions

6.1.1 Enables Local IP Address monitoring

shmOamSnmpLocalIpAddrMonitorEnable

Syntax:

```
ShmErrorT
shmOamSnmpLocalIpAddrMonitorEnable(
        ShmSessionIdT                                   sessionId);
```

Description:

Enables Local IP Address Monitor changes in local IP addresses will force SNMP agent reconfiguration. Because Net-SNMP does not support transport layer reconfiguration at runtime, the SNMP agent must be terminated and restarted again. SNMP agent is terminated by LIPAM module and restarted again by the Process Monitor.

Parameters:

`sessionId` [in]

Session identifier previously obtained using shmSessionOpen().

Returns:

`SHM_CC_OK`

6.1.2 Disables Local IP Address monitoring

shmOamSnmpLocalIpAddrMonitorDisable

Syntax:

```
ShmErrorT
shmOamSnmpLocalIpAddrMonitorDisable(
        ShmSessionIdT                                   sessionId);
```

Description:

Disables Local IP Address Monitor changes in local IP address configuration will be ignored. User is responsible for proper transport layer configuration in common configuration file for SNMP agent.

Parameters:

`sessionId` [in]

Session identifier previously obtained using shmSessionOpen().

Returns:

`SHM_CC_OK`

6.1.3 Returns the status of Local IP Address Monitor

shmOamSnmpLocalIpAddrMonitorStatus

Syntax:

```
ShmErrorT
shmOamSnmpLocalIpAddrMonitorStatus(
        ShmSessionIdT                           sessionId,
        ShmBoolT*                               monitorStatus);
```

Description:

Reads the Local IP Address Monitor status.

Parameters:

`sessionId` [in]

Session identifier previously obtained using shmSessionOpen().

`monitorStatus` [out]

Local IP Address Monitor status.

Returns:

`SHM_CC_OK`

7 RMCP Configuration API

7.1 Data Types definitions

7.1.1 Status

ShmOamStatusT

Syntax:

```
typedef enum {
          SHM_OAM_STATUS_DISABLED   = 0,
          SHM_OAM_STATUS_ENABLED    = 1
} ShmOamStatusT;
```

Members:

SHM_OAM_STATUS_DISABLED

Disabled.

SHM_OAM_STATUS_ENABLED

Enabled.

7.2 API functions

7.2.1 Add OEM user permission

shmOamRmcpSvrOemPermittedAdd

Syntax:

```
ShmErrorT
shmOamRmcpSvrOemPermittedAdd(
          ShmSessionIdT                    sessionId,
          unsigned char                    netfn,
          unsigned char                    cmd);
```

Description:

Enables OEM user to execute IPMI command specified by netfn, cmd parameters. Initially all IPMI commands are disabled for OEM users.

Parameters:

`sessionId` [in]

Session identifier previously obtained using shmSessionOpen().

`netfn` [in]

IPMI msg netfn code.

`cmd` [in]

IPMI msg cmd code.

Returns:

`SHM_CC_OK`

7.2.2 Removes permission for OEM

shmOamRmcpSvrOemPermittedDel

Syntax:

```
ShmErrorT
shmOamRmcpSvrOemPermittedDel(
          ShmSessionIdT                          sessionId,
          unsigned char                          netfn,
          unsigned char                          cmd);
```

Description:

Disables OEM user access to IPMI command specified by netfn, cmd parameters.

Parameters:

sessionId [in]

Session identifier previously obtained using shmSessionOpen().

netfn [in]

IPMI msg netfn code.

cmd [in]

IPMI msg cmd code.

Returns:

SHM_CC_OK

7.2.3 Gets OEM user permission

shmOamRmcpSvrOemPermittedGet

Syntax:

```
ShmErrorT
shmOamRmcpSvrOemPermittedGet(
          ShmSessionIdT                    sessionId,
          unsigned char                    netfn,
          unsigned char                    cmd,
          ShmOamStatusT*                   status);
```

Description:

Returns TRUE if OEM user is permitted to execute IPMI command specified by netfn, cmd parameters. Otherwise FALSE is returned.

Parameters:

`sessionId` [in]

Session identifier previously obtained using shmSessionOpen().

`netfn` [in]

IPMI msg netfn code.

`cmd` [in]

IPMI msg cmd code.

`status` [out]

IPMI command status.

Returns:

`SHM_CC_OK`

7.2.4 Configure RMCP server transport protocol

shmOamRmcpSvrTransportProtocolSet

Syntax:

```
ShmErrorT
shmOamRmcpSvrTransportProtocolSet(
          ShmSessionIdT                          sessionId,
          ShmRmcpPlusTransportProtoT             protocol,
          ShmOamStatusT                          status);
```

Parameters:

`sessionId` [in]

Session identifier previously obtained using shmSessionOpen().

`protocol` [in]

RMCP server transport protocol type.

`status` [in]

Operational status for protocol. Changes to the transport protocol configuration are activated after the server restart.

Returns:

`SHM_CC_OK`

7.2.5 Get the RMCP server transport protocol

shmOamRmcpSvrTransportProtocolGet

Syntax:

```
ShmErrorT
shmOamRmcpSvrTransportProtocolGet(
          ShmSessionIdT                          sessionId,
          ShmRmcpPlusTransportProtoT             protocol,
          ShmOamStatusT*                         status);
```

Parameters:

`sessionId` [in]

Session identifier previously obtained using shmSessionOpen().

`protocol` [in]

RMCP server transport protocol type.

`status` [out]

Operational status for protocol.

Returns:

`SHM_CC_OK`

8 Policy Management OAM API

8.1 Data Types definitions

8.1.1 Event Severities

ShmOamPolicyEventSeverityT

Syntax:

```
typedef enum {
        SHM_OAM_POLICY_EVENT_SEVERITY_CRITICAL,
        SHM_OAM_POLICY_EVENT_SEVERITY_MAJOR,
        SHM_OAM_POLICY_EVENT_SEVERITY_MINOR,
        SHM_OAM_POLICY_EVENT_SEVERITY_NORMAL,
        SHM_OAM_POLICY_EVENT_SEVERITY_EXTENDED
} ShmOamPolicyEventSeverityT;
```

Members:

SHM_OAM_POLICY_EVENT_SEVERITY_CRITICAL

Critical severity

SHM_OAM_POLICY_EVENT_SEVERITY_MAJOR

Major severity

SHM_OAM_POLICY_EVENT_SEVERITY_MINOR

Minor severity

SHM_OAM_POLICY_EVENT_SEVERITY_NORMAL

Normal severity

SHM_OAM_POLICY_EVENT_SEVERITY_EXTENDED

Ignored severity, this value is used when Event Code is supplied

Description:

The type defines event severity levels used by Policy Management.

8.1.2 Event Code

ShmOamPolicyEventCodeT

Syntax:

```
typedef   unsigned short        ShmOamPolicyEventCodeT;
```

Description:

The type defines implementation specific event code associated with event.

8.1.3 Action Table Entry ID

ShmOamPolicyActionTableEntryIdT

Syntax:

```
typedef   unsigned short        ShmOamPolicyActionTableEntryIdT;
```

8.1.4 Maximum allowed execution Time (in seconds)

ShmOamPolicyExecutionTimeT

Syntax:

```
typedef   unsigned int          ShmOamPolicyExecutionTimeT;
```

Description:

Use 0 as unlimited time.

8.1.5 Unlimited execution time

Syntax:

```
#define  SHM_OAM_POLICY_UNLIMITED_EXECUTION_TIME   0
```

Members:

`SHM_OAM_POLICY_UNLIMITED_EXECUTION_TIME`

Description:

Unlimited execution time.

8.1.6 User script path length

Syntax:

```
#define   SHM_OAM_POLICY_MAXPATH   256
```

Members:

SHM_OAM_POLICY_MAXPATH

Description:

User script path length.

8.1.7 Wildcard values for filter parameters

Syntax:

```
#define  SHM_OAM_POLICY_ANY_ADDRESS          0xFF
#define  SHM_OAM_POLICY_CMM_ADDRESS          0xFD
#define  SHM_OAM_POLICY_CHASSIS_ADDRESS      0x20
#define  SHM_OAM_POLICY_ANY_LUN              0xFF
#define  SHM_OAM_POLICY_ANY_SENSOR_NUMBER    0xFF
#define  SHM_OAM_POLICY_DUMMY_EVENT_CODE     0
```

Members:

SHM_OAM_POLICY_ANY_ADDRESS

SHM_OAM_POLICY_CMM_ADDRESS

SHM_OAM_POLICY_CHASSIS_ADDRESS

SHM_OAM_POLICY_ANY_LUN

SHM_OAM_POLICY_ANY_SENSOR_NUMBER

SHM_OAM_POLICY_DUMMY_EVENT_CODE

Description:

Wildcard values for filter parameters.

8.1.8 Action Table related constants

Syntax:

```
#define  SHM_OAM_POLICY_FIRST_ENTRY
                                        (ShmOamPolicyActionTableEntryIdT)0x0000
#define  SHM_OAM_POLICY_LAST_ENTRY
                                        (ShmOamPolicyActionTableEntryIdT)0xFFFF
#define  SHM_OAM_POLICY_FREE_ENTRY       SHM_OAM_POLICY_FIRST_ENTRY
#define  SHM_OAM_POLICY_UNSPECIFIED_ENTRY  SHM_OAM_POLICY_LAST_ENTRY
```

Members:

SHM_OAM_POLICY_FIRST_ENTRY

SHM_OAM_POLICY_LAST_ENTRY

SHM_OAM_POLICY_FREE_ENTRY

SHM_OAM_POLICY_UNSPECIFIED_ENTRY

Description:

Action Table related constants.

8.1.9 Event Filter

ShmOamPolicyEventFilterT

Syntax:

```
typedef struct {
        ShmIpmbAddrT                    slaveAddress;
        ShmIpmiLunT                     lun;
        ShmSensorTypeT                  sensorType;
        ShmSensorNumT                   sensorNumber;
        ShmOamPolicyEventSeverityT      severity;
        ShmOamPolicyEventCodeT          eventCode;
} ShmOamPolicyEventFilterT;
```

Description:

This structure encompasses event filter attributes. A match between event and filter occurs when there are event filter matches (exact or wild-carded) for all compared fields in the event message

Members:

`slaveAddress`

Value of the Slave Address from the event message used to match the event with the filter entry. The value 0xff used for wildcard filter matches any address except for 20h.

`lun`

Value of the LUN field from the event message used to match the event with the filter entry. The value 0xff used for wildcard filter matches any LUN field value.

`sensorType`

Value of the Sensor Type field from the event message used to match the event with the filter entry.

`sensorNumber`

Value of the Sensor Number field from the event message used to match the event with the filter entry.

`severity`

Value of the Event Severity associated with an event and used to match the event with the filter entry. Value SHM_OAM_POLICY_EVENT_SEVERITY_EXTENDED is used when Event Code is supplied.

`eventCode`

Value of the Event Code associated with an event and used to match the event with the filter entry. The value 0xffff matches any Event Code value.

8.1.10 Policy Action

ShmOamPolicyActionT

Syntax:

```
typedef struct {
    char                          scriptName[SHM_OAM_POLICY_MAXPATH];
    ShmOamPolicyExecutionTimeT    executionTime;
} ShmOamPolicyActionT;
```

Description:

This structure encompasses user script attributes.

Members:

`scriptName`

Script name that is to be executed upon event occurrence. This can be name of the file that contains the script, a relative pathname (one that begins with directory name and not with "/"), or an absolute pathname beginning with "/".

executionTime

Allowed execution time [s]. If 0 then unlimited time is used.

8.2 API Functions

8.2.1 Add script association

shmOamPolicyActionAdd

Syntax:

```
ShmErrorT
shmOamPolicyActionAdd(
        ShmSessionIdT                       sessionId,
        ShmOamPolicyEventFilterT            filter,
        ShmOamPolicyActionT                 action,
        ShmOamPolicyActionTableEntryIdT*    entryId);
```

Description:

This function makes association between event filter and user defined script.

Parameters:

`sessionId [in]`

Session identifier previously obtained using shmSessionOpen().

`filter [in]`

Event filter.

`action [in]`

User script that shall be executed when an event that matches filter occurs.

entryId `[inout]`

Action Table Entry ID to be added. Value SHM_OAM_POLICY_FREE_ENTRY is used when implementation is to assign entryId.

Returns:

SHM_CC_OK

Returned on successful completion. Otherwise, an error code is returned.

SHM_CC_INVALID_SESSION_ID

Invalid session ID.

SHM_CC_INVALID_ENTRY_ID

Invalid entry ID.

SHM_CC_RANGE

Invalid parameter.

SHM_CC_ALREADY_USED

Association already exists.

SHM_CC_INVALID_FILE_NAME

Invalid script file name.

SHM_CC_DIR_NOT_VALID

Invalid script pathname.

SHM_CC_FILE_NOT_FOUND

Script file does not exist, has a different name, or is stored in another directory.

SHM_CC_DIR_NOT_ALLOWED

Script pathname cannot be a directory.

SHM_CC_ZERO_SIZE

Script has zero bytes size.

SHM_CC_INSUFFICENT_PRIVILEGE

Invalid script execution rights.

SHM_CC_ERROR

Indicates unspecified error.

8.2.2 Remove script association

shmOamPolicyActionRemove

Syntax:

```
ShmErrorT
shmOamPolicyActionRemove(
          ShmSessionIdT                          sessionId,
          ShmOamPolicyActionTableEntryIdT        entryId);
```

Description:

This function removes association between event filter and user defined script.

Parameters:

`sessionId [in]`

Session identifier previously obtained using shmSessionOpen().

`entryId [in]`

Action Table Entry ID to be removed.

Returns:

SHM_CC_OK

Returned on successful completion. Otherwise, an error code is returned.

SHM_CC_INVALID_SESSION_ID

Invalid session ID.

SHM_CC_INVALID_ENTRY_ID

Invalid entry ID.

SHM_CC_ENTITY_NOT_PRESENT

Association not present.

SHM_CC_ERROR

Indicates unspecified error.

8.2.3 Get script association

shmOamPolicyActionGet

Syntax:

```
ShmErrorT
shmOamPolicyActionGet(
          ShmSessionIdT                          sessionId,
          ShmOamPolicyEventFilterT*              filter,
          ShmOamPolicyActionT*                   action,
          ShmOamPolicyActionTableEntryIdT*       entryId);
```

Description:

This function returns association between indicated event filter and user defined script.

Parameters:

`sessionId [in]`

Session identifier previously obtained using shmSessionOpen().

`filter` `[inout]`

Event filter.

`action` `[out]`

Pointer to the action associated with event filter.

`entryId` `[inout]`

Action Table Entry ID to be found. If set to SHM_OAM_POLICY_UNSPECIFIED_ENTRY then filter is passed to the function as a search key. In the opposite case entryId identifies the entry. Filter and action are returned by the function.

Returns:

`SHM_CC_OK`

Returned on successful completion. Otherwise, an error code is returned.

`SHM_CC_INVALID_SESSION_ID`

Invalid session ID.

`SHM_CC_INVALID_ENTRY_ID`

Invalid entry ID.

`SHM_CC_RANGE`

Parameter out of range.

`SHM_CC_SCRIPT_WAS_REMOVED`

Script was deleted or moved without unassociating from event.

`SHM_CC_ERROR`

Indicates unspecified error.

8.2.4 Get next script association

shmOamPolicyActionGetNext

Syntax:

```
ShmErrorT
shmOamPolicyActionGetNext(
          ShmSessionIdT                              sessionId,
          ShmOamPolicyActionTableEntryIdT*           entryId,
          ShmOamPolicyEventFilterT*                  filter,
          ShmOamPolicyActionT*                       action);
```

Description:

This iterator-type function returns the next association between an event filter and a script.

Parameters:

`sessionId [in]`

Session identifier previously obtained using shmSessionOpen().

`entryId [inout]`

Returned Action Table Entry ID. On input it is set to the previous entry ID. It must be set to SHM_OAM_POLICY_FIRST_ENTRY to get the first entry. Function returns SHM_OAM_POLICY_LAST_ENTRY when last entry is set on input.

`filter [inout]`

Returned event filter.

`action [inout]`

Returned action.

Returns:

`SHM_CC_OK`

Returned on successful completion. Otherwise, an error code is returned.

`SHM_CC_INVALID_SESSION_ID`

Invalid session ID.

`SHM_CC_INVALID_ENTRY_ID`

Invalid entry ID.

SHM_CC_ERROR

Indicates unspecified error.

8.2.5 Synchronize user-defined script

shmOamPolicyScriptSync

Syntax:

```
ShmErrorT
shmOamPolicyScriptSync(
        ShmSessionIdT                   sessionId,
        char*                           scriptName);
```

Description:

This function forces synchronization of a user-defined script between active and standby LISM.

Parameters:

`sessionId [in]`

Session identifier previously obtained using shmSessionOpen().

`scriptName [in]`

Name of script to be synchronized.

Returns:

SHM_CC_OK

Returned on successful completion. Otherwise, an error code is returned.

SHM_CC_INVALID_SESSION_ID

Invalid session ID.

SHM_CC_INVALID_FILE_NAME

Invalid script file name.

SHM_CC_DIR_NOT_VALID

Invalid script pathname.

SHM_CC_FILE_NOT_FOUND

Script file does not exist, has a different name, or is stored in another directory.

SHM_CC_DIR_NOT_ALLOWED

Script pathname cannot be a directory.

SHM_CC_ZERO_SIZE

Script has zero bytes size.

SHM_CC_SCRIPT_WAS_REMOVED

Script was deleted or moved without unassociating from event.

SHM_CC_ERROR

Indicates unspecified error.

9 TimeSync API

9.1 Data Types definitions

9.1.1 Maximum supported server index

Syntax:

```
#define  TIME_SYNC_MAX_SERVER_INDEX
```

Members:

TIME_SYNC_MAX_SERVER_INDEX

Description:

Defines maximum supported server index.

9.1.2 Maximum supported listen address index

Syntax:

```
#define  TIME_SYNC_MAX_LISTEN_ADDR_INDEX
```

Members:

TIME_SYNC_MAX_LISTEN_ADDR_INDEX

Description:

Defines maximum supported listen address index.

9.1.3 Maximum supported broadcast address index

Syntax:

```
#define    TIME_SYNC_MAX_BROADCAST_ADDR_INDEX
```

Members:

TIME_SYNC_MAX_BROADCAST_ADDR_INDEX

Description:

Defines maximum supported broadcast address index.

9.1.4 Invalid index value

Syntax:

```
#define    TIME_SYNC_VOID_INDEX
```

Members:

TIME_SYNC_VOID_INDEX

Description:

Defines invalid index value.

9.1.5 Default value for server preferred argument

Syntax:

```
#define    TIME_SYNC_DEFAULT_SERVER_PREFERRED
```

Members:

TIME_SYNC_DEFAULT_SERVER_PREFERRED

Description:

Defines default value for server preferred argument.

9.1.6 Default value for server NTP version

Syntax:

```
#define    TIME_SYNC_DEFAULT_NTP_VERSION
```

Members:

TIME_SYNC_DEFAULT_NTP_VERSION

Description:

Defines default value for server NTP version.

9.1.7 Default value for server min poll

Syntax:

```
#define    TIME_SYNC_DEFAULT_MIN_POLL
```

Members:

TIME_SYNC_DEFAULT_MIN_POLL

Description:

Defines default value for server min poll.

9.1.8 Default value for server max poll

Syntax:

```
#define    TIME_SYNC_DEFAULT_MAX_POLL
```

Members:

TIME_SYNC_DEFAULT_MAX_POLL

Description:

Defines default value for server max poll.

9.1.9 Default value for broadcast interval

Syntax:

```
#define    TIME_SYNC_DEFAULT_BROADCAST_INTERVAL
```

Members:

TIME_SYNC_DEFAULT_BROADCAST_INTERVAL

Description:

Defines default value for broadcast interval.

9.1.10 NTP Version

ShmOamTimeSyncNtpVersionT

Syntax:

```
typedef    unsigned int    ShmOamTimeSyncNtpVersionT;
```

Description:

Identifies protocol version used in communication.

Range:

2..3

9.1.11 Server Index

ShmOamTimeSyncServerIndexT

Syntax:

```
typedef   unsigned int          ShmOamTimeSyncServerIndexT;
```

Description:

Unique number used to identify server.

Range:

`0..TIME_SYNC_MAX_SERVER_INDEX`

9.1.12 Server poll period

ShmOamTimeSyncPollT

Syntax:

```
typedef   unsigned int          ShmOamTimeSyncPollT;
```

Description:

Server poll period specified in seconds as power of 2. For example server poll value of 6 means 64 seconds.

Range:

`4..10`

9.1.13 Server Index

ShmOamTimeSyncListenAddrIndexT

Syntax:

```
typedef   unsigned int          ShmOamTimeSyncListenAddrIndexT;
```

Description:

Unique number used to identify listen address in a table.

Range:

`0..TIME_SYNC_MAX_LISTEN_ADDR_INDEX`

9.1.14 Server Index

ShmOamTimeSyncBroadcastAddrIndexT

Syntax:

```
typedef   unsigned int          ShmOamTimeSyncBroadcastAddrIndexT;
```

Description:

Unique number used to identify listen address in a table.

Range:

`0..TIME_SYNC_MAX_BROADCAST_ADDR_INDEX`

9.1.15 Socket Address

ShmOamSockAddrT

Syntax:

```
typedef struct {
        uint32_t                    address;
        uint16_t                    port;
} ShmOamSockAddrT;
```

Members:

address

address

port

port

9.2 API Functions

9.2.1 Server Table Iterator

shmOamTimeSyncNextServerIndexGet

Syntax:

```
ShmErrorT
shmOamTimeSyncNextServerIndexGet(
        ShmSessionIdT                   sessionId,
        ShmOamTimeSyncServerIndexT      prevServerIdx,
        ShmOamTimeSyncServerIndexT*     nextServerIdx);
```

Description:

Retrieves index of the next configured server.

Parameters:

`sessionId` [in]

Session identifier previously obtained using shmSessionOpen().

`prevServerIdx` [in]

Index of previous server or TIME_SYNC_VOID_INDEX to start indexing.

`nextServerIdx` [out]

Index of the next server or TIME_SYNC_VOID_INDEX if no more servers configured in the table.

Returns:

`SHM_CC_OK`

9.2.2 Get Server Parameters

shmOamTimeSyncServerParamsGet

Syntax:

```
ShmErrorT
shmOamTimeSyncServerParamsGet(
        ShmSessionIdT                   sessionId,
        ShmOamTimeSyncServerIndexT      serverIdx,
        ShmOamSockAddrT*                serverAddress,
        ShmBoolT*                       preferred,
        ShmOamTimeSyncNtpVersionT*      ntpVersion,
        ShmOamTimeSyncPollT*            minServerPoll,
        ShmOamTimeSyncPollT*            maxServerPoll);
```

Description:

Retrieves parameters of the configured server.

Parameters:

sessionId [in]

Session identifier previously obtained using shmSessionOpen().

serverIdx [in]

Index of the server of which parameters to be retrieved.

serverAddress [out]

Server network address.

preferred [out]

Set to true if this is the preferred server.

ntpVersion [out]

NTP version used in communication to this server.

minServerPoll [out]

Minimum polling period for this server.

maxServerPoll [out]

Maximum polling period for this server.

Returns:

`SHM_CC_OK`

9.2.3 Add new server

shmOamTimeSyncServerAdd

Syntax:

```
ShmErrorT
shmOamTimeSyncServerAdd(
        ShmSessionIdT                   sessionId,
        ShmOamTimeSyncServerIndexT      serverIdx,
        ShmOamSockAddrT                 serverAddress,
        ShmBoolT                        preferred,
        ShmOamTimeSyncNtpVersionT       ntpVersion,
        ShmOamTimeSyncPollT             minServerPoll,
        ShmOamTimeSyncPollT             maxServerPoll);
```

Description:

This functions adds a new server to the server table.

Parameters:

`sessionId` [in]

Session identifier previously obtained using shmSessionOpen().

`serverIdx` [in]

Index of the created server.

`serverAddress` [in]

Server network address.

`preferred` [in]

Set to true if this is the preferred server.

`ntpVersion` [in]

NTP version used in communication to this server.

`minServerPoll` [in]

Minimum polling period for this server.

`maxServerPoll` [in]

Maximum polling period for this server.

Returns:

SHM_CC_OK

9.2.4 Delete server configuration

shmOamTimeSyncServerDelete

Syntax:

```
ShmErrorT
shmOamTimeSyncServerDelete(
        ShmSessionIdT                   sessionId,
        ShmOamTimeSyncServerIndexT      serverIdx);
```

Description:

Set configuration parameters for the specified server.

Parameters:

sessionId [in]

Session identifier previously obtained using shmSessionOpen().

serverIdx [in]

Index of the server to be deleted.

Returns:

SHM_CC_OK

9.2.5 Listen Address Table Iterator

shmOamTimeSyncNextListenAddressIndexGet

Syntax:

```
ShmErrorT
shmOamTimeSyncNextListenAddressIndexGet(
        ShmSessionIdT                   sessionId,
        ShmOamTimeSyncListenAddrIndexT  prevAddrIdx,
        ShmOamTimeSyncListenAddrIndexT* nextAddrIdx);
```

Description:

Retrieves index of the next configured listen address.

Parameters:

sessionId [in]

Session identifier previously obtained using shmSessionOpen().

prevAddrIdx [in]

Index of the previous address or TIME_SYNC_VOID_INDEX to start indexing.

nextAddrIdx [out]

Index of the next address or TIME_SYNC_VOID_INDEX if no more addresses configured in the table.

Returns:

SHM_CC_OK

9.2.6 Retrieve Listen Address

shmOamTimeSyncListenAddressGet

Syntax:

```
ShmErrorT
shmOamTimeSyncListenAddressGet(
          ShmSessionIdT                          sessionId,
          ShmOamTimeSyncListenAddrIndexT         addrIdx,
          ShmOamSockAddrT*                       address);
```

Description:

Retrieves configured network address used to listen for NTP requests.

Parameters:

sessionId [in]

Session identifier previously obtained using shmSessionOpen().

addrIdx [in]

Index of the server of which parameters to be retrieved.

address [out]

Network address.

Returns:

SHM_CC_OK

9.2.7 Add new address to listen on

shmOamTimeSyncListenAddrAdd

Syntax:

```
ShmErrorT
shmOamTimeSyncListenAddrAdd(
          ShmSessionIdT                          sessionId,
          ShmOamTimeSyncListenAddrIndexT         addrIdx,
          ShmOamSockAddrT                        address);
```

Description:

This functions adds a new network address to listen on for incoming NTP requests.

Parameters:

sessionId [in]

Session identifier previously obtained using shmSessionOpen().

addrIdx [in]

Index of the created address.

address [in]

Network address.

Returns:

SHM_CC_OK

9.2.8 Delete listen address

shmOamTimeSyncListenAddrDelete

Syntax:

```
ShmErrorT
shmOamTimeSyncListenAddrDelete(
          ShmSessionIdT                          sessionId,
          ShmOamTimeSyncListenAddrIndexT         addrIdx);
```

Description:

Requests to stop listing for the NTP requests on this address.

Parameters:

sessionId [in]

Session identifier previously obtained using shmSessionOpen().

addrIdx [in]

Index of the address to be deleted.

Returns:

SHM_CC_OK

9.2.9 Broadcast Address Table Iterator

shmOamTimeSyncNextBroadcastAddressIndexGet

Syntax:

```
ShmErrorT
shmOamTimeSyncNextBroadcastAddressIndexGet(
          ShmSessionIdT                          sessionId,
          ShmOamTimeSyncBroadcastAddrIndexT      prevAddrIdx,
          ShmOamTimeSyncBroadcastAddrIndexT*     nextAddrIdx);
```

Description:

Retrieves index of the next configured broadcast address.

Parameters:

sessionId [in]

Session identifier previously obtained using shmSessionOpen().

prevAddrIdx [in]

Index of the previous address or TIME_SYNC_VOID_INDEX to start indexing.

nextAddrIdx [out]

Index of the next address or TIME_SYNC_VOID_INDEX if no more addresses configured in the table.

Returns:

SHM_CC_OK

9.2.10 Retrieve Broadcast Address

shmOamTimeSyncBroadcastAddressGet

Syntax:

```
ShmErrorT
shmOamTimeSyncBroadcastAddressGet(
        ShmSessionIdT                          sessionId,
        ShmOamTimeSyncBroadcastAddrIndexT      addrIdx,
        ShmOamSockAddrT*                       address,
        ShmOamTimeSyncPollT*                   broadcastInterval);
```

Description:

Retrieves configured network address used to listen for NTP requests.

Parameters:

sessionId [in]

Session identifier previously obtained using shmSessionOpen().

addrIdx [in]

Index of the address of which parameters to be retrieved.

address [out]

Network address.

broadcastInterval [out]

Interval for sending out brodacst NTP messages to this address.

Returns:

SHM_CC_OK

9.2.11 Add new broadcast address

shmOamTimeSyncBroadcastAddrAdd

Syntax:

```
ShmErrorT
shmOamTimeSyncBroadcastAddrAdd(
        ShmSessionIdT                       sessionId,
        ShmOamTimeSyncBroadcastAddrIndexT   addrIdx,
        ShmOamSockAddrT                     address,
        ShmOamTimeSyncPollT                 broadcastInterval);
```

Description:

This functions adds a new network address to broadcast NTP messages.

Parameters:

sessionId [in]

Session identifier previously obtained using shmSessionOpen().

addrIdx [in]

Index of the created address.

address [in]

Network address.

broadcastInterval [in]

Interval for sending out broadcast NTP messages to this address.

Returns:

SHM_CC_OK

9.2.12 Delete broadcast address

shmOamTimeSyncBroadcastAddrDelete

Syntax:

```
ShmErrorT
shmOamTimeSyncBroadcastAddrDelete(
          ShmSessionIdT                          sessionId,
          ShmOamTimeSyncBroadcastAddrIndexT      addrIdx);
```

Description:

Requests to stop sending the NTP requests on this broadcast address.

Parameters:

`sessionId` [in]

Session identifier previously obtained using shmSessionOpen().

`addrIdx` [in]

Index of the address to be deleted.

Returns:

`SHM_CC_OK`

10 IP Connectivity Manager API

10.1 Data Types definitions

10.1.1 Ethernet Configuration Source

ShmOamIPConnectivityEtherConfigSource

Syntax:

```
typedef enum {
        IP_CONNECTIVITY_ETHER_CONFIG_SOURCE_STATIC,
        IP_CONNECTIVITY_ETHER_CONFIG_SOURCE_DHCP
} ShmOamIPConnectivityEtherConfigSource;
```

Members:

`IP_CONNECTIVITY_ETHER_CONFIG_SOURCE_STATIC`

Indicates static IP configuration kept in Shelf FRU.

`IP_CONNECTIVITY_ETHER_CONFIG_SOURCE_DHCP`

Indicates dynamic IP configuration obtained using DHCP.

Description:

The type defines possible sources of IP configuration of Ethernet interfaces.

10.1.2 Ethernet Interface Direction

ShmOamIPConnectivityEtherDir

Syntax:

```
typedef enum {
        IP_CONNECTIVITY_ETHER_DIR_FRONT,
        IP_CONNECTIVITY_ETHER_DIR_RTM,
        IP_CONNECTIVITY_ETHER_DIR_BACKPLANE,
        IP_CONNECTIVITY_ETHER_DIR_UNSET
} ShmOamIPConnectivityEtherDir;
```

Members:

IP_CONNECTIVITY_ETHER_DIR_FRONT

The interface is directed to the front panel.

IP_CONNECTIVITY_ETHER_DIR_RTM

The interface is directed to the RTM.

IP_CONNECTIVITY_ETHER_DIR_BACKPLANE

The interface is directed to the backplane.

IP_CONNECTIVITY_ETHER_DIR_UNSET

Value set to factory defaults (valid for getting value from the shelf FRU or config file).

Description:

The type defines possible directions where Ethernet interfaces can be routed on Colorado Springs and Wagotire platforms.

10.1.3 Source of configuration

ShmOamIPConnectivityConfigStorage

Syntax:

```
typedef enum {
        IP_CONNECTIVITY_CONFIG_STORAGE_SHELF_FRU,
        IP_CONNECTIVITY_CONFIG_STORAGE_LOCAL,
        IP_CONNECTIVITY_CONFIG_STORAGE_HW
} ShmOamIPConnectivityEtherDir;
```

Members:

`IP_CONNECTIVITY_CONFIG_STORAGE_SHELF_FRU`

The information is read from shelf FRU.

`IP_CONNECTIVITY_CONFIG_STORAGE_LOCAL`

The information is read from cmm internal runtime structures.

`IP_CONNECTIVITY_CONFIG_STORAGE_HW`

The information is read from hardware.

Description:

Source of configuration. The type defines source where configuration is readied from.

10.1.4 Ethernet Interface IP Configuration

ShmOamIpConnectivityEtherIpConfig

Syntax:

```
typedef struct {
        unsigned int                                ipAddress;
        unsigned int                                subnetMask;
        unsigned int                                defaultGateway;
        ShmOamIPConnectivityEtherConfigSource       configSource;
} ShmOamIpConnectivityEtherIpConfig;
```

Description:

x The type contains IP Configuration of an Ethernet interface.

Members:

ipAddress

IP Address assigned to the interface

subnetMask

Subnet mask assigned to the interface

defaultGateway

Default gateway configured for the interface

configSource

Source of IP configuration

10.2 API Functions

10.2.1 Set Ethernet Interface IP Configuration

shmOamIpConnectivitySetEtherConfig

Syntax:

```
ShmErrorT
shmOamIpConnectivitySetEtherConfig(
        ShmSessionIdT                       sessionId,
        unsigned char                       siteNumber,
        unsigned char                       ifIndex,
        ShmOamIpConnectivityEtherIpConfig   ipConfig);
```

Description:

The function sets IP configuration of an Ethernet interface. Note that the parameter is not applied immediately. It is applied after the board reset or when oamIpConnectivityEtherConfigApply function is called.

Parameters:

`sessionId` [in]

Session ID identifying connection to the target LISM

`siteNumber` [in]

Indicates the ShMC board on which the Ethernet interface is located

`ifIndex` [in]

Index identifying the Ethernet interface

`ipConfig` [in]

New IP configuration

Returns:

`SHM_CC_OK`

Returned on successful completion. Otherwise, an error code is returned.

`SHM_CC_INVALID_SESSION_ID`

Invalid session ID.

`SHM_CC_NOT_FOUND`

Interface does not exist

SHM_CC_INVALID_PARAMETER

Invalid IP configuration parameter

SHM_CC_READ_ONLY

Attempt to write read-only parameter

10.2.2 Get Ethernet Interface IP Configuration

shmOamIpConnectivityGetEtherConfig

Syntax:

```
ShmErrorT
shmOamIpConnectivityGetEtherConfig(
        ShmSessionIdT                          sessionId,
        unsigned char                          siteNumber,
        unsigned char                          ifIndex,
        ShmBoolT                               currentData,
        ShmOamIpConnectivityEtherIpConfig*     ipConfig);
```

Description:

The function returns IP configuration of an Ethernet interface.

Parameters:

sessionId [in]

Session ID identifying connection to the target LISM

siteNumber [in]

Indicates the ShMC board on which the Ethernet interface is located

ifIndex [in]

Index identifying the Ethernet interface

currentData [in]

If the parameter is TRUE, the function returns the currently used configuration (i.e., configuration retrieved from the network stack). Otherwise it returns the configuration stored in Shelf FRU (i.e., the settings that will be used after the next restart).

ipConfig [out]

IP configuration

Returns:

SHM_CC_OK

Returned on successful completion. Otherwise, an error code is returned.

SHM_CC_INVALID_SESSION_ID

Invalid session ID.

SHM_CC_NOT_FOUND

Interface does not exist

SHM_CC_ENTITY_NOT_PRESENT

Data not present

10.2.3 Set Ethernet Interface Direction

shmOamIpConnectivitySetEtherDir

Syntax:

```
ShmErrorT
shmOamIpConnectivitySetEtherDir(
          ShmSessionIdT                    sessionId,
          unsigned char                    siteNumber,
          unsigned char                    ifIndex,
          ShmOamIPConnectivityEtherDir     etherDir);
```

Description:

The function sets direction of an Ethernet interface. Note that the parameter is not applied immediately. It is applied after the board reset or when oamIpConnectivityEtherConfigApply function is called.

Parameters:

sessionId [in]

Session ID identifying connection to the target LISM

siteNumber [in]

Indicates the ShMC board on which the Ethernet interface is located

ifIndex [in]

Index identifying the Ethernet interface

etherDir [in]

New Ethernet interface direction

Returns:

SHM_CC_OK

Returned on successful completion. Otherwise, an error code is returned.

SHM_CC_INVALID_SESSION_ID

Invalid session ID.

SHM_CC_NOT_FOUND

Interface does not exist

SHM_CC_INVALID_PARAMETER

Invalid Ethernet interface direction parameter

SHM_CC_READ_ONLY

Attempt to write read-only parameter

10.2.4 Get Ethernet Interface Direction

shmOamIpConnectivityGetEtherDir

Syntax:

```
ShmErrorT
shmOamIpConnectivityGetEtherDir(
        ShmSessionIdT                          sessionId,
        unsigned char                          siteNumber,
        unsigned char                          ifIndex,
        ShmOamIPConnectivityConfigStorage      configStorage,
        ShmOamIPConnectivityEtherDir*          etherDir);
```

Description:

The function returns direction of an Ethernet interface.

Parameters:

sessionId [in]

Session ID identifying connection to the target LISM

siteNumber [in]

Indicates the ShMC board on which the Ethernet interface is located

ifIndex [in]

Index identifying the Ethernet interface

configStorage [in]

Source of configuration to retrieve

etherDir [out]

Ethernet interface direction

Returns:

SHM_CC_OK

Returned on successful completion. Otherwise, an error code is returned.

SHM_CC_INVALID_SESSION_ID

Invalid session ID.

SHM_CC_NOT_FOUND

Interface does not exist

SHM_CC_ENTITY_NOT_PRESENT

Data not present

10.2.5 Apply Ethernet Configuration Settings Immediately

shmOamIpConnectivityEtherConfigApply

Syntax:

```
ShmErrorT
shmOamIpConnectivityEtherConfigApply(
        ShmSessionIdT                                   sessionId);
```

Description:

This function commits all non-committed changes of network configuration into kernel (interface reconfiguration) and/or HW (Ethernet directions).

Parameters:

sessionId [in]

Session ID identifying connection to the target LISM

Returns:

SHM_CC_OK

Returned on successful completion. Otherwise, an error code is returned.

SHM_CC_INVALID_SESSION_ID

Invalid session ID.

11 EKeying Manager API

11.1 API Functions

11.1.1 Get Port State

shmOamEKeyManagerLinkStateGet

Syntax:

```
ShmErrorT
shmOamEKeyManagerLinkStateGet(
        ShmSessionIdT                   sessionId,
        ShmIpmcAddrT                    dstAddr,
        unsigned char                   ifcType,
        unsigned int*                   length,
        ShmLinkInfoRecT**               linkStateTable);
```

Description:

Unlike shmGetPortState(), this function returns information about all links defined by the IPMC.

Parameters:

sessionId [in]

Session identifier previously obtained using shmSessionOpen().

dstAddr [in]

Destination address. Provides IPMB address and LUN that identify target destination.

ifcType [in]

Interface

Describes the interface being queried. { base | fabric | update }

length [out]

The length of the LinkStateTable.

linkStateTable [out]

Table containing link information. When empty, the Channel does not have any links on the channel.

Returns:

Possible return values are:

SHM_CC_OK

is returned on successful completion; otherwise, appropriate IPMI error code is returned.

SHM_CC_NOT_FOUND

Returned when EKey Manager has no records about given IPMC

SHM_CC_UNAVAILABLE

Returned when EKey Manager is collecting information

12 Diagnostics API

12.1 Data Types definitions

12.1.1 Test Result Enumeration

ShmOamDiagnosticsTestResultEnumT

Syntax:

```
typedef enum {
          SHM_DIAGN_TEST_PASSED,
          SHM_DIAGN_TEST_REJECTED,
          SHM_DIAGN_TEST_FAILED,
          SHM_DIAGN_TEST_UNDETERMINED
} ShmOamDiagnosticsTestResultEnumT;
```

Members:

SHM_DIAGN_TEST_PASSED

No error. Test passed.

SHM_DIAGN_TEST_REJECTED

Test cannot be run at that moment.

SHM_DIAGN_TEST_FAILED

Test failed.

SHM_DIAGN_TEST_UNDETERMINED

Unable to determine test result. Operator's action needed.

Description:

Specifies the result of the diagnostic test.

12.1.2 Test Rejection Cause

ShmOamDiagnosticsTestRejectCauseT

Syntax:

```
typedef enum {
          INVALID_HA_STATE,
          IPMB_LINK_OTHER_NOT_OPERATIONAL,
          IPMB_LINK_STATE_CHANGED,
          NO_BUFFER_AVAILABLE,
          UNKNOWN_REASON
} ShmOamDiagnosticsTestRejectCauseT;
```

Members:

`INVALID_HA_STATE`

Test cannot be run because HA state is not Active.

`IPMB_LINK_OTHER_NOT_OPERATIONAL`

Test cannot be run because the other IPMB link is not operational.

`IPMB_LINK_STATE_CHANGED`

Test interrupted because the IPMB link state changed to operational during the diagnostics procedure.

`NO_BUFFER_AVAILABLE`

Test cannot be run buffers are exhausted.

`UNKNOWN_REASON`

Test cannot be run due to unspecified reason.

Description:

Specifies the cause of rejecting diagnostic test.

12.1.3 Test Error Codes

ShmOamDiagnosticsTestErrorT

Syntax:

```
typedef enum {
        INVALID_ADDRESS_ERROR,
        INVALID_DATA_ERROR,
        NO_RESPONSE_ERROR,
        IPMB_DRIVER_ERROR,
        IPMB_INVALID_LINK_ERROR,
        IPMB_LINK_ISOLATED_ERROR,
        IPMB_SETTING_CLOCK_LINE_HIGH_ERROR,
        IPMB_SETTING_CLOCK_LINE_LOW_ERROR,
        IPMB_SETTING_DATA_LINE_HIGH_ERROR,
        IPMB_SETTING_DATA_LINE_LOW_ERROR,
        IPMB_CLOCK_LOW_ERROR,
        UNKNOWN_ERROR
} ShmOamDiagnosticsTestErrorT;
```

Members:

INVALID_ADDRESS_ERROR

Invalid test target address.

INVALID_DATA_ERROR

Invalid test data.

NO_RESPONSE_ERROR

Test target does not respond.

IPMB_DRIVER_ERROR

IPMB driver error.

IPMB_INVALID_LINK_ERROR

Invalid IPMB link number.

IPMB_LINK_ISOLATED_ERROR

IPMB link isolated.

IPMB_SETTING_CLOCK_LINE_HIGH_ERROR

Setting clock line high failed.

IPMB_SETTING_CLOCK_LINE_LOW_ERROR

Setting clock line low failed.

IPMB_SETTING_DATA_LINE_HIGH_ERROR

Setting data line high failed.

IPMB_SETTING_DATA_LINE_LOW_ERROR

Setting data line low failed.

IPMB_CLOCK_LOW_ERROR

Clock low error.

UNKNOWN_ERROR

Unspecified error.

Description:

Specifies diagnostic test error code.

12.1.4 Diagnostics Test Result

ShmOamDiagnosticsTestResultT

Syntax:

```
typedef struct {
        ShmOamDiagnosticsTestResultEnumT        result;
        union {
            ShmOamDiagnosticsTestRejectCauseT   rejectCause;
            ShmOamDiagnosticsTestErrorT         error;
            int                                 empty;
        } u;
} ShmOamDiagnosticsTestResultT;
```

Members:

result

Diagnostics test result.

rejectCause

Test rejection cause.

Valid for selector: SHM_DIAGN_TEST_REJECTED

error

Test error code.

Valid for selector: SHM_DIAGN_TEST_FAILED

empty

Empty.

Valid for selector: SHM_DIAGN_TEST_PASSED

SHM_DIAGN_TEST_UNDETERMINED

12.1.5 IPv4 address

ShmOamInAddrT

Syntax:

```
typedef struct {
        unsigned int                    address;
} ShmOamInAddrT;
```

Members:

`address`

32 bit IPv4 address (host order)

12.2 API functions

12.2.1 Flash Test

shmOamDiagnosticsTestFlash

Syntax:

```
ShmErrorT
shmOamDiagnosticsTestFlash(
        ShmSessionIdT                   sessionId,
        ShmOamDiagnosticsTestResultT*   result);
```

Description:

Test Flash storage device.

Parameters:

`sessionId` [in]

session ID identifying target LISM.

`result` [out]

Result of the test.

Returns:

SHM_CC_OK

Returned on successful completion. Otherwise, an error code is returned.

SHM_CC_INVALID_SESSION_ID

Invalid session ID.

SHM_CC_ERROR

Indicates unspecified error.

12.2.2 Ethernet Test

shmOamDiagnosticsTestEthernet

Syntax:

```
ShmErrorT
shmOamDiagnosticsTestEthernet(
          ShmSessionIdT                          sessionId,
          ShmOamInAddrT                          target,
          ShmOamDiagnosticsTestResultT*          result);
```

Description:

Test Ethernet connectivity by pinging specified location.

Parameters:

sessionId [in]

session ID identifying target LISM.

target [in]

IP address to ping.

result [out]

Result of the test.

Returns:

SHM_CC_OK

Returned on successful completion. Otherwise, an error code is returned.

SHM_CC_INVALID_SESSION_ID

Invalid session ID.

SHM_CC_INVALID_PARAMETER

Invalid parameter.

SHM_CC_ERROR

Indicates unspecified error.

12.2.3 IPMB Test

shmOamDiagnosticsTestIpmb

Syntax:

```
ShmErrorT
shmOamDiagnosticsTestIpmb(
        ShmSessionIdT                       sessionId,
        ShmIpmbLinkNumT                     linkNum,
        unsigned char                       physicalLinkIndicator,
        ShmOamDiagnosticsTestResultT*       result);
```

Description:

IPMB-0 Test.

Parameters:

sessionId [in]

session ID identifying target LISM.

linkNum [in]

IPMB link number.

physicalLinkIndicator [in]

Physical link indicator (0 -IPMB-A, 1 - IPMB-B).

result [out]

Result of the test.

Returns:

SHM_CC_OK

Returned on successful completion. Otherwise, an error code is returned.

SHM_CC_INVALID_SESSION_ID

Invalid session ID.

SHM_CC_INVALID_PARAMETER

Invalid parameter.

SHM_CC_ERROR

Indicates unspecified error.

12.2.4 LED Test

shmOamDiagnosticsTestLeds

Syntax:

```
ShmErrorT
shmOamDiagnosticsTestLeds(
        ShmSessionIdT                   sessionId,
        ShmIpmcAddrT                    destAddr,
        ShmFruDevIdT                    fruId,
        unsigned short                  testDuration,
        ShmOamDiagnosticsTestResultT*   result);
```

Description:

Test if the LEDs work on FRU.

Parameters:

sessionId [in]

session ID identifying target LISM.

destAddr [in]

Destination address.

fruId [in]

FRU Id.

testDuration [in]

Duration of the lamp test [100ms].

result [out]

Result of the test.

Returns:

SHM_CC_OK

Returned on successful completion. Otherwise, an error code is returned.

SHM_CC_INVALID_SESSION_ID

Invalid session ID.

SHM_CC_INVALID_PARAMETER

Invalid parameter.

SHM_CC_ERROR

Indicates unspecified error.

12.2.5 Reboot Reason Enumeration

ShmOamDiagnosticsRebootReasonT

Syntax:

```
typedef enum {
          REBOOT_REASON_UPGRADE,
          REBOOT_REASON_MANUAL_RESET,
          REBOOT_REASON_FRU_CONTROL_RESET,
          REBOOT_REASON_PM_RESET,
          REBOOT_REASON_EJECTOR_OPEN,
          REBOOT_REASON_OS_SHUTDOWN,
          REBOOT_REASON_KERNEL_PANIC
} ShmOamDiagnosticsRebootReasonT;
```

Members:

REBOOT_REASON_UPGRADE

Reboot due to firmware upgrade.

REBOOT_REASON_MANUAL_RESET

Reboot due to user command.

REBOOT_REASON_FRU_CONTROL_RESET

Reboot due to FRU Control IPMI command.

REBOOT_REASON_PM_RESET

Reboot from Process Monitoring.

REBOOT_REASON_EJECTOR_OPEN

Reboot due to ejector latch open.

REBOOT_REASON_OS_SHUTDOWN

Reboot due to Linux shutdown command.

REBOOT_REASON_KERNEL_PANIC

Reboot due to kernel panic.

Description:

Specifies SW reasons for last reboot.

12.2.6 Set Reboot Reason

shmOamDiagnosticsSetRebootReason

Syntax:

```
ShmErrorT
shmOamDiagnosticsSetRebootReason(
          ShmSessionIdT                          sessionId,
          ShmOamDiagnosticsRebootReasonT         reason);
```

Description:

Sets the reason for last reboot.

Parameters:

sessionId [in]

session ID identifying target LISM.

reason [in]

Reboot reason.

Returns:

SHM_CC_OK

Returned on successful completion. Otherwise, an error code is returned.

SHM_CC_INVALID_SESSION_ID

Invalid session ID.

SHM_CC_INVALID_PARAMETER

Invalid parameter.

SHM_CC_ERROR

Indicates unspecified error.

12.2.7 Clear POST Sensor

shmOamDiagnosticsClearPOSTSensor

Syntax:

```
ShmErrorT
shmOamDiagnosticsClearPOSTSensor(
          ShmSessionIdT                                  sessionId,
          ShmIpmcAddrT                                   shmcAddr);
```

Description:

Clears POST sensor state.

Parameters:

`sessionId` [in]

session ID identifying target LISM.

`shmcAddr` [in]

Address of LISM on which POST sensor should be cleared.

Returns:

SHM_CC_OK

Returned on successful completion. Otherwise, an error code is returned.

SHM_CC_INVALID_SESSION_ID

Invalid session ID.

SHM_CC_INVALID_PARAMETER

Invalid parameter.

SHM_CC_ERROR

Indicates unspecified error.

13 Statistics API

13.1 Data Types definitions

13.1.1 Max Length of the Statistics Group/Counter Name

Syntax:

```
#define  SHM_STATISTICS_NAME_SIZE  48
```

Members:

SHM_STATISTICS_NAME_SIZE

Description:

Max. length of the statistics group/counter name.

13.1.2 Counter Value Type

ShmOamCounterValueT

Syntax:

```
typedef  uint64_t  ShmOamCounterValueT;
```

Range:

full

13.1.3 Group/Counter Name Type

ShmOamStatisticsNameT

Syntax:

```
typedef   char        ShmOamStatisticsNameT[SHM_STATISTICS_NAME_SIZE];
```

13.2 API functions

13.2.1 Get Statistics Groups

shmOamStatisticsGetCounterGroups

Syntax:

```
ShmErrorT
shmOamStatisticsGetCounterGroups(
        ShmSessionIdT                   sessionId,
        unsigned int*                   groupNum,
        ShmOamStatisticsNameT**         groupNames);
```

Description:

Get groups of statistics counters present in LISM.

Parameters:

`sessionId` [in]

session ID identifying target LISM.

`groupNum` [out]

Number of entries in the above table.

`groupNames` [out]

Returned table of the group names.

Returns:

`SHM_CC_OK`

Returned on successful completion. Otherwise, an error code is returned.

SHM_CC_INVALID_SESSION_ID

Invalid session ID.

SHM_CC_ERROR

Indicates unspecified error.

13.2.2 Get Counter Descriptions

shmOamStatisticsGetCounterNames

Syntax:

```
ShmErrorT
shmOamStatisticsGetCounterNames(
        ShmSessionIdT                   sessionId,
        char*                           groupName,
        unsigned int*                   counterNum,
        ShmOamStatisticsNameT**         counterNames);
```

Description:

Gets names in the specified counter group.

Parameters:

sessionId [in]

session ID identifying target LISM.

groupName [in]

Name of the counter group.

counterNum [out]

Number of entries in the above table.

counterNames [out]

Returned table of counter names.

Returns:

SHM_CC_OK

Returned on successful completion. Otherwise, an error code is returned.

SHM_CC_INVALID_SESSION_ID

Invalid session ID.

SHM_CC_INVALID_GROUP

Invalid group name.

SHM_CC_ERROR

Indicates unspecified error.

13.2.3 Get Counter Value

shmOamStatisticsGetCounterValue

Syntax:

```
ShmErrorT
shmOamStatisticsGetCounterValue(
        ShmSessionIdT                   sessionId,
        char*                           groupName,
        char*                           counterName,
        ShmBoolT                        resetFlag,
        ShmOamCounterValueT*            counterValue);
```

Description:

Gets the value of the specified counter.

Parameters:

sessionId [in]

session ID identifying target LISM.

groupName [in]

Group name.

counterName [in]

Counter name.

resetFlag [in]

Reset after read flag.

counterValue [out]

Returned counter value.

Returns:

SHM_CC_OK

Returned on successful completion. Otherwise, an error code is returned.

SHM_CC_INVALID_SESSION_ID

Invalid session ID.

SHM_CC_INVALID_GROUP

Invalid group name.

SHM_CC_INVALID_COUNTER

Invalid counter name.

SHM_CC_ERROR

Indicates unspecified error.

13.2.4 Threshold Type Enumeration

ShmOamStatisticsThresholdTypeT

Syntax:

```
typedef enum {
          THRESHOLD_NONE,
          THRESHOLD_UPPER,
          THRESHOLD_LOWER,
          THRESHOLD_BOTH
} ShmOamStatisticsThresholdTypeT;
```

Members:

THRESHOLD_NONE

no threshold

THRESHOLD_UPPER

upper threshold

THRESHOLD_LOWER

lower threshold

THRESHOLD_BOTH

both lower and upper thresholds

Description:

Specifies what thresholds a counter supports.

13.2.5 Get Threshold Types

shmOamStatisticsGetCounterThresholdTypes

Syntax:

```
ShmErrorT
shmOamStatisticsGetCounterThresholdTypes(
        ShmSessionIdT                           sessionId,
        char*                                   groupName,
        char*                                   counterName,
        ShmOamStatisticsThresholdTypeT*         thresholdType);
```

Description:

Gets threshold types supported by the counter.

Parameters:

`sessionId` [in]

session ID identifying target LISM.

`groupName` [in]

Group name.

`counterName` [in]

Counter name.

`thresholdType` [out]

Returned threshold type.

Returns:

`SHM_CC_OK`

Returned on successful completion. Otherwise, an error code is returned.

`SHM_CC_INVALID_SESSION_ID`

Invalid session ID.

`SHM_CC_INVALID_GROUP`

Invalid group name.

`SHM_CC_INVALID_COUNTER`

Invalid counter name.

`SHM_CC_ERROR`

Indicates unspecified error.

13.2.6 Get Counter Threshold

shmOamStatisticsGetCounterThreshold

Syntax:

```
ShmErrorT
shmOamStatisticsGetCounterThreshold(
        ShmSessionIdT                       sessionId,
        char*                               groupName,
        char*                               counterName,
        ShmOamStatisticsThresholdTypeT      thresholdType,
        ShmOamCounterValueT*                thresholdValue);
```

Description:

Gets the current threshold value of the counter.

Parameters:

sessionId [in]

session ID identifying target LISM.

groupName [in]

Group name.

counterName [in]

Counter name.

thresholdType [in]

Specifies whether upper or lower threshold should be returned

thresholdValue [out]

Returned threshold value.

Returns:

SHM_CC_OK

Returned on successful completion. Otherwise, an error code is returned.

SHM_CC_INVALID_SESSION_ID

Invalid session ID.

SHM_CC_INVALID_GROUP

Invalid group name.

SHM_CC_INVALID_COUNTER

Invalid counter name.

SHM_CC_INVALID_THRESHOLD

Invalid threshold type.

SHM_CC_ERROR

Indicates unspecified error.

13.2.7 Set Counter Threshold

shmOamStatisticsSetCounterThreshold

Syntax:

```
ShmErrorT
shmOamStatisticsSetCounterThreshold(
        ShmSessionIdT                       sessionId,
        char*                               groupName,
        char*                               counterName,
        ShmOamStatisticsThresholdTypeT      thresholdType,
        ShmOamCounterValueT                 thresholdValue);
```

Description:

Sets the value of the threshold for the counter.

Parameters:

sessionId [in]

session ID identifying target LISM.

groupName [in]

name of the counter group.

counterName [in]

Counter name.

thresholdType [in]

Specifies whether the upper or lower threshold should be set

thresholdValue [in]

Value of the threshold to set.

Returns:

SHM_CC_OK

Returned on successful completion. Otherwise, an error code is returned.

SHM_CC_INVALID_SESSION_ID

Invalid session ID.

SHM_CC_INVALID_GROUP

Invalid group name.

SHM_CC_INVALID_COUNTER

Invalid counter name.

SHM_CC_INVALID_THRESHOLD

Invalid threshold type.

SHM_CC_ERROR

Indicates unspecified error.

13.2.8 Get Next Counter Get value of the next counter

shmOamStatisticsGetNextCounter

Syntax:

```
ShmErrorT
shmOamStatisticsGetNextCounter(
        ShmSessionIdT                           sessionId,
        char*                                   prevGroupName,
        char*                                   prevCounterName,
        char**                                  groupName,
        char**                                  counterName,
        ShmOamStatisticsThresholdTypeT*         thresholdType,
        ShmOamCounterValueT*                    lowerThresholdValue,
        ShmOamCounterValueT*                    upperThresholdValue,
        ShmOamCounterValueT*                    counterValue);
```

Parameters:

sessionId [in]

session ID identifying target LISM.

prevGroupName [in]

Preceding counter's group name. Set to NULL to retrieve first group.

prevCounterName [in]

Preceding counter name. Set to NULL to retrieve first counter in a group.

groupName [out]

Next counter's group name in lexicographical order.

counterName [out]

Next counter name in lexicographical order.

thresholdType [out]

Returned threshold type.

lowerThresholdValue [out]

Returned lower threshold value.

upperThresholdValue [out]

Returned upper threshold value.

counterValue [out]

Returned counter value.

Returns:

SHM_CC_OK

Returned on successful completion. Otherwise, an error code is returned.

SHM_CC_INVALID_SESSION_ID

Invalid session ID.

SHM_CC_LAST_ENTRY

Last entry encountered.

SHM_CC_ERROR

Indicates unspecified error.

14 Event Manager OAM API

14.1 API functions

14.1.1 Get SEL Size

shmOamEventManagerLogSizeGet

Syntax:

```
ShmErrorT
shmOamEventManagerLogSizeGet(
          ShmSessionIdT                          sessionId,
          unsigned int*                          number);
```

Description:

This function returns number of log records stored in SEL.

Parameters:

`sessionId` [in]

Session identifier previously obtained using shmSessionOpen().

`number` [out]

Number of records in SEL.

Returns:

`SHM_CC_OK`

Successful completion.

14.1.2 Get SEL Capacity

shmOamEventManagerLogCapacityGet

Syntax:

```
ShmErrorT
shmOamEventManagerLogCapacityGet(
        ShmSessionIdT                   sessionId,
        unsigned int*                   confCapacity,
        Unsigned int*                   currCapacity);
```

Description:

This function returns configured max number of log records that can be stored in SEL.

Parameters:

sessionId [in]

Session identifier previously obtained using shmSessionOpen().

confCapacity [out]

Configured Max number of records in SEL.

currCapacity [out]

Current Max number of records in SEL.

Returns:

SHM_CC_OK

Successful completion.

14.1.3 Set SEL Capacity

shmOamEventManagerLogCapacitySet

Syntax:

```
ShmErrorT
shmOamEventManagerLogCapacitySet(
          ShmSessionIdT                          sessionId,
          unsigned int                           number);
```

Description:

This function configures max number of log records that can be stored in SEL.

Parameters:

`sessionId` [in]

Session identifier previously obtained using shmSessionOpen().

`number` [in]

Max number of records in SEL.

Returns:

SHM_CC_OK

Successful completion.

SHM_CC_INVALID_PARAMETER

Invalid parameter value.

14.1.4 Get SEL Archive Information

shmOamEventManagerLogArchiveInfoGet

Syntax:

```
ShmErrorT
shmOamEventManagerLogArchiveInfoGet(
          ShmSessionIdT                                 sessionId,
          unsigned int*                                 maxCopies,
          unsigned int*                                 maxSize);
```

Description:

This function returns configured SEL archive parameters.

Parameters:

`sessionId` [in]

Session identifier previously obtained using shmSessionOpen().

`maxCopies` [out]

Max number of SEL backup copies.

`maxSize` [out]

Max total size of all SEL backup copies.

Returns:

`SHM_CC_OK`

Successful completion.

14.1.5 Set SEL Archive Information

shmOamEventManagerLogArchiveInfoSet

Syntax:

```
ShmErrorT
shmOamEventManagerLogArchiveInfoSet(
          ShmSessionIdT                          sessionId,
          unsigned int                           maxCopies,
          unsigned int                           maxSize);
```

Description:

This function configures SEL archive parameters.

Parameters:

sessionId [in]

Session identifier previously obtained using shmSessionOpen().

maxCopies [in]

Max number of SEL backup copies.

maxSize [in]

Max total size of all SEL backup copies.

Returns:

SHM_CC_OK

Successful completion.

14.1.6 Archive SEL

shmOamEventManagerLogArchive

Syntax:

```
ShmErrorT
shmOamEventManagerLogArchive(
        ShmSessionIdT                                   sessionId);
```

Description:

This function force SEL archive procedure. Current SEL will be closed and archived. New empty SEL will be reopened.

Parameters:

`sessionId` [in]

Session identifier previously obtained using shmSessionOpen().

Returns:

`SHM_CC_OK`

Successful completion.

14.1.7 Find Event Manager Log Entry by Date

shmOamEventManagerLogEntryFindByDate

Syntax:

```
ShmErrorT
shmOamEventManagerLogEntryFindByDate(
        ShmSessionIdT                                   sessionId,
        ShmTimestampT                                   time,
        ShmRecordIdT*                                   eventLogEntryId);
```

Description:

This function finds first event log entry that was added on given date. If found, function returns in the output parameter SEL record Id of the first log entry with matching date. User shall use SHM function shmSelEntryGet() to retrieve log entry. If no matching entry was found, function returns with error code.

Parameters:

`sessionId` [in]

Session identifier previously obtained using shmSessionOpen().

`time` [in]

Time.

`eventLogEntryId` [out]

Record Id of the first event log entry that match given date.

Returns:

`SHM_CC_OK`

Successful completion.

`SHM_CC_NOT_FOUND`

No matching entries was found.

14.1.8 Find Event Manager Log Entry by Severity

shmOamEventManagerLogEntryFindBySeverity

Syntax:

```
ShmErrorT
shmOamEventManagerLogEntryFindBySeverity(
        ShmSessionIdT                           sessionId,
        ShmOamEventSeverity                     severity,
        ShmRecordIdT                            firstEntry,
        ShmRecordIdT*                           firstFoundEntry);
```

Description:

This function finds next event log entry with given severity. If found, function returns in the output parameter SEL record id of the next log entry with matching severity. User shall use SHM function shmSelEntryGet() to retrieve log entry. If no matching entry was found, function returns with error code.

Parameters:

`sessionId` [in]

Session identifier previously obtained using shmSessionOpen().

`severity` [in]

Severity.

`firstEntry` [in]

Record id of the first event log entry to begin search.
`firstFoundEntry` [out]

Record id of the first next log entry that match given severity.

Returns:

`SHM_CC_OK`

Successful completion.

`SHM_CC_NOT_FOUND`

No matching entries was found.

14.1.9 Get SEL Capacity

shmOamEventManagerLogCapacityRangeGet

Syntax:

```
ShmErrorT
shmOamEventManagerLogCapacityRangeGet(
        ShmSessionIdT                           sessionId,
        unsigned int*                           minCapacity,
        unsigned int*                           maxCapacity);
```

Description:

Get SEL Capacity minimum and maximum allowed values.

Parameters:

`sessionId` [in]

Session identifier previously obtained using shmSessionOpen().
`minCapacity` [out]

Minimum allowed capacity of SEL.
`maxCapacity` [out]

Maximum allowed capacity of SEL.

Returns:

`SHM_CC_OK`

Successful completion.

15 Local Sensor API

15.1 Data Types definitions

15.1.1 Local Sensor Indicator

ShmLsrLocalIndicatorT

Syntax:

```
typedef enum {
          SHM_LSR_LOCAL          = 0,
          SHM_LSR_SHELF          = 1
} ShmLsrLocalIndicatorT;
```

Members:

`SHM_LSR_LOCAL`

Used to indicate local sensors available at physical IPMB address.

`SHM_LSR_SHELF`

Used to indicate shelf sensors available at active ShMC address (20h).

Description:

The type is used to indicate location of the sensor (local or shelf).

15.1.2 Entity Instance Type

ShmLsrEntityInstanceTypeT

Syntax:

```
typedef enum {
        SHM_LSR_ENTITY_INSTANCE_PHYSICAL,
        SHM_LSR_ENTITY_INSTANCE_LOGICAL
} ShmLsrEntityInstanceTypeT;
```

Members:

`SHM_LSR_ENTITY_INSTANCE_PHYSICAL`

`SHM_LSR_ENTITY_INSTANCE_LOGICAL`

Description:

The enum is use to indicate entity instance type used in SDR.

15.1.3 Entity Instance Scope

ShmLsrEntityInstanceScopeT

Syntax:

```
typedef enum {
        SHM_LSR_ENTITY_INSTANCE_SYS_REL,
        SHM_LSR_ENTITY_INSTANCE_DEV_REL
} ShmLsrEntityInstanceScopeT;
```

Members:

`SHM_LSR_ENTITY_INSTANCE_SYS_REL`

System Relative Entity Instance

`SHM_LSR_ENTITY_INSTANCE_DEV_REL`

Device Relative Entity Instance

Description:

The enum is use to indicate entity instance scope used in SDR.

15.1.4 ID String Modifier Type

ShmLsrIdStringModifierTypeT

Syntax:

```
typedef enum {
        SHM_LSR_ID_STRING_MODIFIER_NUMERIC  = 0,
        SHM_LSR_ID_STRING_MODIFIER_ALPHA    = 1
} ShmLsrIdStringModifierTypeT;
```

Members:

`SHM_LSR_ID_STRING_MODIFIER_NUMERIC`

`SHM_LSR_ID_STRING_MODIFIER_ALPHA`

15.1.5 LSR Sensor Direction

ShmLsrSensorDirectionT

Syntax:

```
typedef enum {
        SHM_LSR_SENSOR_DIRECTION_UNSPECIFIED = 0,
        SHM_LSR_SENSOR_DIRECTION_INPUT       = 1,
        SHM_LSR_SENSOR_DIRECTION_OUTPUT      = 2
} ShmLsrSensorDirectionT;
```

Members:

SHM_LSR_SENSOR_DIRECTION_UNSPECIFIED

SHM_LSR_SENSOR_DIRECTION_INPUT

SHM_LSR_SENSOR_DIRECTION_OUTPUT

15.1.6 General Sensor Parameters

ShmLsrGeneralSensorDef

Syntax:

```
typedef struct {
        ShmLsrLocalIndicatorT          localIndicator;
        unsigned char                  sensorNumber;
        unsigned char                  entityId;
        ShmLsrEntityInstanceTypeT      entityInstanceType;
        ShmLsrEntityInstanceScopeT     entityInstanceScope;
        unsigned char                  entityInstanceNumber;
        unsigned char                  sensorType;
        unsigned char                  eventReadingTypeCode;
        unsigned char                  oem;
        unsigned char                  sensorId[256];
} ShmLsrGeneralSensorDef;
```

Description:

The definition is common for all sensors.

Members:

localIndicator
sensorNumber
entityId
entityInstanceType
entityInstanceScope
entityInstanceNumber
sensorType
eventReadingTypeCode
oem
sensorId

15.1.7 Sensor Group Parameters

ShmLsrSensorGroupDefT

Syntax:

```
typedef struct {
    unsigned char                   shareCount;
    ShmLsrIdStringModifierTypeT     idStringInstanceModifierType;
    unsigned char                   idStringInstanceModifierOffset;
} ShmLsrSensorGroupDefT;
```

Members:

shareCount

Number of sensors in the group.

idStringInstanceModifierType
idStringInstanceModifierOffset

15.1.8 Event-Only Sensor Parameters

ShmLsrEventOnlySensorDefT

Syntax:

```
typedef struct {
        ShmLsrGeneralSensorDef          generalSensorDef;
        ShmLsrSensorDirectionT          sensorDirection;
} ShmLsrEventOnlySensorDefT;
```

Members:

generalSensorDef

General sensor definition.

sensorDirection

Sensor direction.

15.1.9 Event-Only Sensor Group Parameters

ShmLsrEventOnlySensorGroupDefT

Syntax:

```
typedef struct {
        ShmLsrEventOnlySensorDefT       eventOnlySensorDef;
        ShmLsrSensorGroupDefT           sensorGroupDef;
} ShmLsrEventOnlySensorGroupDefT;
```

Members:

eventOnlySensorDef

Event-Only Sensor Definition.

sensorGroupDef

Sensor Group Definition.

15.1.10 Event Data 2 Content

ShmLsrEventData2ContentT

Syntax:

```
typedef enum {
        DATA2_NOTHING                           = 0,
        DATA2_SEVERITY_EVENT_READING_CODE       = 1,
        DATA2_OEM_VALUE                         = 2,
        DATA2_SENSOR_SPECIFIC_EVENT_EXTENSION   = 3
} ShmLsrEventData2ContentT;
```

Members:

DATA2_NOTHING

DATA2_SEVERITY_EVENT_READING_CODE

DATA2_OEM_VALUE

DATA2_SENSOR_SPECIFIC_EVENT_EXTENSION

15.1.11 Event Data 3 Content

ShmLsrEventData3ContentT

Syntax:

```
typedef enum {
        DATA3_NOTHING                                   = 0,
        DATA3_OEM_VALUE                                 = 2,
        DATA3_SENSOR_SPECIFIC_EVENT_EXTENSION           = 3
} ShmLsrEventData3ContentT;
```

Members:

DATA3_NOTHING

DATA3_OEM_VALUE

DATA3_SENSOR_SPECIFIC_EVENT_EXTENSION

15.2 API functions

15.2.1 Create Event-Only Sensor

shmLsrEventOnlySensorCreate

Syntax:

```
ShmErrorT
shmLsrEventOnlySensorCreate(
        ShmSessionIdT                           sessionId,
        ShmLsrEventOnlySensorDefT               sensorDef);
```

Parameters:

sessionId [in]

Session identifier previously obtained using shmSessionOpen().

sensorDef [in]

Sensor definition.

Returns:

SHM_CC_OK

Success

SHM_CC_ALREADY_USED

Sensor Already Exists

SHM_CC_INVALID_PARAMETER

Invalid Parameter

15.2.2 Create Event-Only Sensor

shmLsrEventOnlySensorCreateIfDoesNotExist

Syntax:

```
ShmErrorT
shmLsrEventOnlySensorCreateIfDoesNotExist(
        ShmSessionIdT                   sessionId,
        ShmLsrEventOnlySensorDefT       sensorDef);
```

Parameters:

sessionId [in]

Session identifier previously obtained using shmSessionOpen().

sensorDef [in]

Sensor definition.

Returns:

SHM_CC_OK

Success

SHM_CC_ALREADY_USED

Sensor Already Exists

SHM_CC_INVALID_PARAMETER

Invalid Parameter

15.2.3 Delete Event-Only Sensor

shmLsrEventOnlySensorDelete

Syntax:

```
ShmErrorT
shmLsrEventOnlySensorDelete(
        ShmSessionIdT                   sessionId,
        ShmLsrLocalIndicatorT           localIndicator,
        unsigned char                   sensorNumber);
```

Parameters:

`sessionId` [in]

Session identifier previously obtained using shmSessionOpen().

`localIndicator` [in]

Identifies the sensor.

`sensorNumber` [in]

Identifies the sensor.

Returns:

SHM_CC_OK

Success

SHM_CC_NOT_FOUND

Sensor Does Not Exist

SHM_CC_INVALID_PARAMETER

Invalid Parameter

SHM_CC_ILLEGAL_CMD

The sensor is not event-only sensor.

15.2.4 Create Event-Only Sensor Group

shmLsrEventOnlySensorGroupCreate

Syntax:

```
ShmErrorT
shmLsrEventOnlySensorGroupCreate(
        ShmSessionIdT                       sessionId,
        ShmLsrEventOnlySensorGroupDefT      sensorGroupDef);
```

Description:

The function creates a group of sensor sharing the same SDR record. The sensor numbers starts with the value specified in the sensor definition and are incremeneted for subsequent sensors.

Parameters:

`sessionId` [in]

Session identifier previously obtained using shmSessionOpen().

`sensorGroupDef` [in]

Sensor group definition.

Returns:

`SHM_CC_OK`

Success

`SHM_CC_ALREADY_USED`

Sensor Already Exists

`SHM_CC_INVALID_PARAMETER`

Invalid Parameter

15.2.5 Delete Event-Only Sensor Group

shmLsrEventOnlySensorGroupDelete

Syntax:

```
ShmErrorT
shmLsrEventOnlySensorGroupDelete(
        ShmSessionIdT                   sessionId,
        ShmLsrLocalIndicatorT           localIndicator,
        unsigned char                   sensorNumber);
```

Parameters:

`sessionId` [in]

Session identifier previously obtained using shmSessionOpen().

`localIndicator` [in]

Identifies the sensor group.

`sensorNumber` [in]

Identifies the sensor group.

Returns:

SHM_CC_OK

Success

SHM_CC_NOT_FOUND

Sensor Group Does Not Exist

SHM_CC_INVALID_PARAMETER

Invalid Parameter

SHM_CC_ILLEGAL_CMD

The sensors are not event-only sensors.

15.2.6 Send Event for Event-Only Sensor

shmLsrEventOnlySensorSendEvent

Syntax:

```
ShmErrorT
shmLsrEventOnlySensorSendEvent(
        ShmSessionIdT                       sessionId,
        ShmLsrLocalIndicatorT               localIndicator,
        unsigned char                       sensorNumber,
        ShmBoolT                            assertion,
        unsigned char                       eventOffset,
        ShmLsrEventData2ContentT            eventData2Content,
        unsigned char                       eventData2,
        ShmLsrEventData3ContentT            eventData3Content,
        unsigned char                       eventData3);
```

Parameters:

`sessionId` [in]

Session identifier previously obtained using shmSessionOpen().

`localIndicator` [in]

Identifies the sensor.

`sensorNumber` [in]

Identifies the sensor.

`assertion` [in]

Indicates value of assertion(TRUE)/deassertion(FALSE) flag.

`eventOffset` [in]

Event Offset field value.

`eventData2Content` [in]

Event Data 2 Content Indicator.

`eventData2` [in]

Event Data 2 field value.

`eventData3Content` [in]

Event Data 3 Content Indicator.

`eventData3` [in]

Event Data 3 field value.

Returns:

SHM_CC_OK

Success

SHM_CC_NOT_FOUND

Sensor Does Not Exist

SHM_CC_INVALID_PARAMETER

Invalid Parameter

SHM_CC_ILLEGAL_CMD

The sensor is not event-only sensor.

SHM_CC_ILLEGAL_CMD_FOR_HASTATE

The LISM instance is not active ShMC.

16 OEM Access Permissions API

16.1 Data Types definitions

16.1.1 Security Access Value Enumeration

ShmOamSecurityAccessValueT

Syntax:

```
typedef enum {
          ACCESS_UNDEFINED = 0,
          ACCESS_ALLOWED = 1,
          ACCESS_DENIED = 2
} ShmOamSecurityAccessValueT;
```

Members:

`ACCESS_UNDEFINED = 0`

Undefined access value.

`ACCESS_ALLOWED = 1`

Access allowed.

`ACCESS_DENIED = 2`

Access denied.

Description:

Specifies the access value granted to the user.

16.2 API functions

16.2.1 Get OEM Access Permission

shmOamSecurityOemPermissionGet

Syntax:

```
ShmErrorT
shmOamSecurityOemPermissionGet(
        ShmSessionIdT                       sessionId,
        int                                 rpcProgramNr,
        int                                 rpcFunctionNr,
        ShmOamSecurityAccessValueT*         value);
```

Description:

Gets the access permissions for OEM role to the specified IPMI-like or OAM function.

Parameters:

`sessionId` [in]

session ID identifying target LISM.

`rpcProgramNr` [in]

RPC program number of the requested function.

`rpcFunctionNr` [in]

RPC function number of the requested function.

`value` [out]

Returned access value.

Returns:

`SHM_CC_OK`

is returned on successful completion; otherwise, appropriate error code is returned.

16.2.2 Get Next OEM Access Permission

shmOamSecurityOemPermissionGetNext

Syntax:

```
ShmErrorT
shmOamSecurityOemPermissionGetNext(
        ShmSessionIdT                       sessionId,
        int*                                rpcProgramNr,
        int*                                rpcFunctionNr,
        ShmOamSecurityAccessValueT*         value);
```

Description:

Gets the access permission value defined in the OEM access permission table that is next to the one specified in rpcProgramNr/rpcFunctionNr.

Parameters:

`sessionId` [in]

session ID identifying target LISM.

`rpcProgramNr` [inout]

RPC program number of the requested function.

`rpcFunctionNr` [inout]

RPC function number of the requested function.

`value` [out]

Returned access value or NULL if there is no value in table after the one specified in rpcProgramNr/rpcFunctionNr.

Returns:

`SHM_CC_OK`

is returned on successful completion; otherwise, appropriate error code is returned.

16.2.3 Set OEM Access Permission

shmOamSecurityOemPermissionSet

Syntax:

```
ShmErrorT
shmOamSecurityOemPermissionSet(
        ShmSessionIdT                       sessionId,
        int                                 rpcProgramNr,
        int                                 rpcFunctionNr,
        ShmOamSecurityAccessValueT          value);
```

Description:

Sets the OEM access permission value to the specified IPMI-like or OAM function for OEM role.

Parameters:

`sessionId` [in]

session ID identifying target LISM.

`rpcProgramNr` [in]

RPC program number of the requested function.

`rpcFunctionNr` [in]

RPC function number of the requested function.

`value` [in]

Access permission value to set for the OEM role.

Returns:

`SHM_CC_OK`

is returned on successful completion; otherwise, appropriate error code is returned.

16.2.4 Reset OEM Access Permission

shmOamSecurityOemPermissionReset

Syntax:

```
ShmErrorT
shmOamSecurityOemPermissionReset(
        ShmSessionIdT                   sessionId,
        int                             rpcProgramNr,
        int                             rpcFunctionNr);
```

Description:

Resets the OEM access permission to the specified IPMI-like or OAM function to the default value.

Parameters:

`sessionId` [in]

session ID identifying target LISM.

`rpcProgramNr` [in]

RPC program number of the requested function.

`rpcFunctionNr` [in]

RPC function number of the requested function.

Returns:

`SHM_CC_OK`

is returned on successful completion; otherwise, appropriate error code is returned.

16.2.5 Get Default OEM Access Permission

shmOamSecurityOemPermissionDefaultGet

Syntax:

```
ShmErrorT
shmOamSecurityOemPermissionDefaultGet(
          ShmSessionIdT                          sessionId,
          ShmOamSecurityAccessValueT*            value);
```

Description:

Gets the default value of the OEM access permission.

Parameters:

`sessionId` [in]

session ID identifying target LISM.

`value` [out]

Returned default OEM access permission value.

Returns:

`SHM_CC_OK`

is returned on successful completion; otherwise, appropriate error code is returned.

16.2.6 Set Default OEM Access Permission

shmOamSecurityOemPermissionDefaultSet

Syntax:

```
ShmErrorT
shmOamSecurityOemPermissionDefaultSet(
          ShmSessionIdT                          sessionId,
          ShmOamSecurityAccessValueT             value);
```

Description:

Sets the default OEM access permission value.

Parameters:

`sessionId` [in]

session ID identifying target LISM.

`value` [in]

Value of the default OEM access permission.

Returns:

`SHM_CC_OK`

is returned on successful completion; otherwise, appropriate error code is returned.

17 User Management OAM API

17.1 API functions

17.1.1 Delete User

shmOamSecurityUserDelete

Syntax:

```
ShmErrorT
shmOamSecurityUserDelete(
        ShmSessionIdT                   sessionId,
        char*                           userName);
```

Description:

Remove the specified user account.

Parameters:

`sessionId` [in]

session ID identifying target LISM.

`userName` [in]

User account to remove.

Returns:

`SHM_CC_OK`

is returned on successful completion; otherwise, appropriate error code is returned.

18 Process Monitoring OAM API

18.1 Data Types definitions

18.1.1 Process Monitoring Process Identifier

ShmPmOamPidT

Syntax:

```
typedef   unsigned char          ShmPmOamPidT;
```

Description:

Unique number used to identify a process. It does not change when a process is started or restarted.

Range:

`1..255`

18.1.2 Max Length of the Process Name

Syntax:

```
#define   SHM_PM_PROCESS_NAME_SIZE   32
```

Members:

SHM_PM_PROCESS_NAME_SIZE

Description:

Max. length of the process name.

18.1.3 Process Name Type

ShmPmOamProcessNameT

Syntax:

```
typedef   char        ShmPmOamProcessNameT[SHM_PM_PROCESS_NAME_SIZE];
```

18.1.4 Process Status

ShmPmOamOpStateT

Syntax:

```
typedef enum {
          SHM_PM_OAM_OP_STATE_OK          = 1,
          SHM_PM_OAM_OP_STATE_FAILED
} ShmPmOamOpStateT;
```

Members:

SHM_PM_OAM_OP_STATE_OK

Process operational (enabled).

SHM_PM_OAM_OP_STATE_FAILED

Process failed (disabled).

18.1.5 Process Administrative State

ShmPmOamAdminStateT

Syntax:

```
typedef enum {
        SHM_PM_OAM_ADMIN_STATE_LOCKED       = 1,
        SHM_PM_OAM_ADMIN_STATUS_UNLOCKED
} ShmPmOamAdminStateT;
```

Members:

`SHM_PM_OAM_ADMIN_STATE_LOCKED`

Process locked.

`SHM_PM_OAM_ADMIN_STATUS_UNLOCKED`

Process unlocked.

18.1.6 Severity Codes

ShmPmOamSeverityT

Syntax:

```
typedef enum {
        SHM_PM_OAM_SEVERITY_CRITICAL    = 1,
        SHM_PM_OAM_SEVERITY_MAJOR,
        SHM_PM_OAM_SEVERITY_MINOR,
        SHM_PM_OAM_SEVERITY_INFO
} ShmPmOamSeverityT;
```

Members:

`SHM_PM_OAM_SEVERITY_CRITICAL`

Critical severity.

`SHM_PM_OAM_SEVERITY_MAJOR`

Major severity.

`SHM_PM_OAM_SEVERITY_MINOR`

Minor severity.

SHM_PM_OAM_SEVERITY_INFO

Informational severity.

Description:

This data type is used to indicate the process severity.

18.1.7 Recovery Type

ShmPmOamRecoveryT

Syntax:

```
typedef enum {
        SHM_PM_OAM_RECOVERY_NO_ACTION                = 1,
        SHM_PM_OAM_RECOVERY_SWITCH_AND_REBOOT,
        SHM_PM_OAM_RECOVERY_RESTART,
        SHM_PM_OAM_RECOVERY_SWITCH_AND_RESTART
} ShmPmOamRecoveryT;
```

Members:

SHM_PM_OAM_RECOVERY_NO_ACTION

Recovery: no action.

SHM_PM_OAM_RECOVERY_SWITCH_AND_REBOOT

Recovery: switchover and reboot.

SHM_PM_OAM_RECOVERY_RESTART

Recovery: process restart.

SHM_PM_OAM_RECOVERY_SWITCH_AND_RESTART

Recovery: switchover and restart.

Description:

This data type is used to describe the process recovery type.

18.1.8 Escalated Recovery Type

ShmPmOamEscalatedRecoveryT

Syntax:

```
typedef enum {
        SHM_PM_OAM_ESCALATED_RECOVERY_NO_ACTION                 = 1,
        SHM_PM_OAM_ESCALATED_RECOVERY_SWITCH_AND_REBOOT
} ShmPmOamEscalatedRecoveryT;
```

Members:

SHM_PM_OAM_ESCALATED_RECOVERY_NO_ACTION

Escalated recovery: no action.

SHM_PM_OAM_ESCALATED_RECOVERY_SWITCH_AND_REBOOT

Escalated recovery: switchover and reboot.

Description:

This data type is used to describe the process escalated recovery type.

18.1.9 Process State

ShmPmOamProcessStateT

Syntax:

```
typedef enum {
            SHM_PM_OAM_PROCESS_STATE_TBD    = 1
} ShmPmOamProcessStateT;
```

Members:

SHM_PM_OAM_PROCESS_STATE_TBD

Process state: TBD.

Description:

This data type is used to describe the process state.

18.1.10 Process Description

ShmPmOamProcessT

Syntax:

```
typedef struct {
        ShmPmOamPidT                    pid;
        ShmPmOamProcessNameT            name;
        ShmPmOamSeverityT               severity;
        ShmPmOamRecoveryT               recovery;
        ShmPmOamEscalatedRecoveryT      escRecovery;
} ShmPmOamProcessT;
```

Description:

This structure describes process monitoring parameters.

Members:

pid

Process ID.

name

Process name.

severity

Process severity.

recovery

Process recovery type.

escRecovery

Process escalated recovery type.

18.1.11 Time Interval

ShmPmOamTimevalT

Syntax:

```
typedef struct {
        int                                     time;
        int                                     suseconds;
} ShmPmOamTimevalT;
```

Members:

time

Time in seconds

suseconds

Time in microseconds

18.2 API functions

18.2.1 Process administrative state set

shmOamPmAdminStateSet

Syntax:

```
ShmErrorT
shmOamPmAdminStateSet(
        ShmSessionIdT                           sessionId,
        ShmPmOamPidT                            pid,
        ShmPmOamAdminStateT                     adminState);
```

Parameters:

sessionId [in]

Session identifier previously obtained using shmSessionOpen().

pid [in]

Process ID.

adminState [in]

Process administrative state. LOCKED or UNLOCKED.

Returns:

SHM_CC_OK

SHM_CC_INVALID_PARAMETER

LOCKED state requested for Process Monitoring process.

SHM_CC_RANGE

Invalid process ID.

18.2.2 Process administrative state get

shmOamPmAdminStateGet

Syntax:

```
ShmErrorT
shmOamPmAdminStateGet(
        ShmSessionIdT                   sessionId,
        int                             pid,
        ShmPmOamAdminStateT*            asminState);
```

Parameters:

`sessionId [in]`

Session identifier previously obtained using shmSessionOpen().

`pid [in]`

Process ID.

`asminState [out]`

Process status. LOCKED or UNLOCKED.

Returns:

SHM_CC_OK

18.2.3 Process Configuration Get

shmOamPmProcessGet

Syntax:

```
ShmErrorT
shmOamPmProcessGet(
        ShmSessionIdT                   sessionId,
        ShmPmOamPidT                    pid,
        ShmPmOamProcessT*               process,
        ShmPmOamAdminStateT*            adminState,
        ShmPmOamOpStateT*               opState,
        ShmPmOamSeverityT*              health,
        char**                          healthEvents);
```

Description:

This call retrieves the process monitoring configuration.

Parameters:

`sessionId [in]`

Session identifier previously obtained using shmSessionOpen().

`pid [in]`

Process ID.

`process [out]`

Process configuration.

`adminState [out]`

Process administrative state. LOCKED or UNLOCKED.

`opState [out]`

Process operational state. ENABLED or DISABLED.

`health [out]`

Process health. Equal to severity of the last process event. Either OK or equal to the process severity.

`healthEvents [out]`

Process health events string.

Returns:

`SHM_CC_OK`

18.2.4 Process ID Array Get

shmOamPmProcessIdArrayGet

Syntax:

```
ShmErrorT
shmOamPmProcessIdArrayGet(
        ShmSessionIdT                   sessionId,
        unsigned int*                   processNum,
        ShmPmOamPidT**                  processArray);
```

Description:

This call retrieves the ID array of existing processes.

Parameters:

`sessionId [in]`

Session identifier previously obtained using shmSessionOpen().

`processNum [out]`

Number of process in the array.

`processArray [out]`

An array of processes monitored by the process monitoring.

Returns:

`SHM_CC_OK`

19 High Avaliability API

19.1 Data Types definitions

19.1.1 Out-of-service Cause Definition

Syntax:

```
#define  SHM_OAM_HA_OOS_CAUSE_USER_COMMAND
#define  SHM_OAM_HA_OOS_CAUSE_EJECTOR_OPEN
#define  SHM_OAM_HA_OOS_CAUSE_IPMB0_LOST
#define  SHM_OAM_HA_OOS_CAUSE_M1_FRU_STATE
#define  SHM_OAM_HA_OOS_CAUSE_SHUTDOWN_REQUEST
#define  SHM_OAM_HA_OOS_CAUSE_ACTIVE_STATE_SEIZED
#define  SHM_OAM_HA_OOS_CAUSE_ELECTION_NO_ROLE
#define  SHM_OAM_HA_OOS_CAUSE_FIM_ELECTION_FAILED
#define  SHM_OAM_HA_OOS_CAUSE_IP_CONNECTIVITY_FAILED
#define  SHM_OAM_HA_OOS_CAUSE_CHASSIS_DETECTION_FAILED
#define  SHM_OAM_HA_OOS_CAUSE_PROCESS_MONITOR_GRACEFUL_REBOOT
#define  SHM_OAM_HA_OOS_CAUSE_PROCESS_MONITOR_REBOOT
#define  SHM_OAM_HA_OOS_CAUSE_FRU_CONTROL_REBOOT
#define  SHM_OAM_HA_OOS_CAUSE_LOCAL_IPMC_NOT_READY
#define  SHM_OAM_HA_OOS_CAUSE_INVALID_LICENSE
```

Members:

SHM_OAM_HA_OOS_CAUSE_USER_COMMAND

User command requested the out-of-service state.

SHM_OAM_HA_OOS_CAUSE_EJECTOR_OPEN

The ejector is open.

SHM_OAM_HA_OOS_CAUSE_IPMB0_LOST

IPMB-0 connectivity lost.

SHM_OAM_HA_OOS_CAUSE_M1_FRU_STATE

Deactivate FRU (M1 FRU state) requested.

SHM_OAM_HA_OOS_CAUSE_SHUTDOWN_REQUEST

Shutdown was requested by OS.

SHM_OAM_HA_OOS_CAUSE_ACTIVE_STATE_SEIZED

The active Shelf Manager 20h address was seized by a peer.

SHM_OAM_HA_OOS_CAUSE_ELECTION_NO_ROLE

No active nor standby role assigned in the election process.

SHM_OAM_HA_OOS_CAUSE_FIM_ELECTION_FAILED

FIM election failed.

SHM_OAM_HA_OOS_CAUSE_IP_CONNECTIVITY_FAILED

IP connectivity lost on the standby CMM.

SHM_OAM_HA_OOS_CAUSE_CHASSIS_DETECTION_FAILED

Chassis type detection failed.

SHM_OAM_HA_OOS_CAUSE_PROCESS_MONITOR_GRACEFUL_REBOOT

Process Monitoring requested CMM graceful reboot as a recovery action type.

SHM_OAM_HA_OOS_CAUSE_PROCESS_MONITOR_REBOOT

Process Monitoring requested CMM reboot as a recovery action type.

SHM_OAM_HA_OOS_CAUSE_FRU_CONTROL_REBOOT

FRU Control reboot request.

SHM_OAM_HA_OOS_CAUSE_LOCAL_IPMC_NOT_READY

Local IPMC is not ready.

SHM_OAM_HA_OOS_CAUSE_INVALID_LICENSE

Invalid license.

Description:

Out-of-service Cause Definition. The bitmasks is used to read the reasons fo the out-of-service readiness state. There can be multiple permanant causes and only one transient cause at the same time for the out-of-service readiness state. Transient causes are: no role assigned in election, FIM election failed or standby IP connectivity failed.

19.1.2 Readiness State

ShmOamHaStateReadinessT

Syntax:

```
typedef enum {
       SHM_OAM_HA_STATE_READINESS_OUT_OF_SERVICE  = 1,
       SHM_OAM_HA_STATE_READINESS_ELECTION,
       SHM_OAM_HA_STATE_READINESS_IN_SERVICE
} ShmOamHaStateReadinessT;
```

Members:

SHM_OAM_HA_STATE_READINESS_OUT_OF_SERVICE

Out-of-service readiness state.

SHM_OAM_HA_STATE_READINESS_ELECTION

Election readiness state.

SHM_OAM_HA_STATE_READINESS_IN_SERVICE

In-service readiness state.

Description:

This data type is used to indicate the readiness state.

19.1.3 High Availability State

ShmOamHaStateHaT

Syntax:

```
typedef enum {
        SHM_OAM_HA_STATE_HA_NONE                = 1,
        SHM_OAM_HA_STATE_HA_ACTIVE_NO_STANDBY,
        SHM_OAM_HA_STATE_HA_ACTIVE,
        SHM_OAM_HA_STATE_HA_STANDBY,
        SHM_OAM_HA_STATE_HA_QUIESCED,
        SHM_OAM_HA_STATE_HA_STOPPING
} ShmOamHaStateHaT;
```

Members:

SHM_OAM_HA_STATE_HA_NONE

HA state does not exists because the readiness state is not in-service.

SHM_OAM_HA_STATE_HA_ACTIVE_NO_STANDBY

Active-no-standby HA state.

SHM_OAM_HA_STATE_HA_ACTIVE

Active HA state.

SHM_OAM_HA_STATE_HA_STANDBY

Standby HA state.

SHM_OAM_HA_STATE_HA_QUIESCED

Quiesced HA state.

SHM_OAM_HA_STATE_HA_STOPPING

Stopping HA state.

Description:

This data type is used to indicate the high availability state. The high availability state exists only within the in-service readiness state.

19.1.4 Presence State

ShmOamHaStatePresenceT

Syntax:

```
typedef enum {
        SHM_OAM_HA_STATE_PRESENCE_NOT_PRESENT  = 0,
        SHM_OAM_HA_STATE_PRESENCE_PRESENT      = 1
} ShmOamHaStatePresenceT;
```

Members:

SHM_OAM_HA_STATE_PRESENCE_NOT_PRESENT

The HW indicates that the CMM is not present.

SHM_OAM_HA_STATE_PRESENCE_PRESENT

The HW indicates that the CMM is present.

Description:

This data type is used to indicate the CMM HW presence.

19.2 API Functions

19.2.1 Get the current readiness state

shmOamHaStateReadinessGet

Syntax:

```
ShmErrorT
shmOamHaStateReadinessGet(
        ShmSessionIdT                       sessionId,
        ShmOamHaStateReadinessT*            readinessState);
```

Parameters:

`sessionId [in]`

Session identifier previously obtained using shmSessionOpen().

`readinessState [out]`

Current readiness state.

Returns:

`SHM_CC_OK`

Current readiness state returned.

19.2.2 Get the current High Availability state

shmOamHaStateHaGet

Syntax:

```
ShmErrorT
shmOamHaStateHaGet(
          ShmSessionIdT                          sessionId,
          ShmOamHaStateHaT*                      haState);
```

Parameters:

`sessionId [in]`

Session identifier previously obtained using shmSessionOpen().

`haState [out]`

Current High Availability state.

Returns:

`SHM_CC_ILLEGAL_CMD_FOR_HASTATE`

Not in the in-service readiness state.

`SHM_CC_OK`

Current readiness state returned.

19.2.3 Get the cause for the out-of-service readiness state

shmOamHaOutOfServiceCauseGet

Syntax:

```
ShmErrorT
shmOamHaOutOfServiceCauseGet(
          ShmSessionIdT                          sessionId,
          unsigned short*                        oosCause);
```

Description:

This request is accepted in the out-of-service readiness state.

Parameters:

`sessionId [in]`

Session identifier previously obtained using shmSessionOpen().

`oosCause [out]`

Bitmask of the causes for the out-of-service state.

Returns:

`SHM_CC_ILLEGAL_CMD_FOR_HASTATE`

Not in the out-of-service readiness state.

`SHM_CC_OK`

Reason returned.

19.2.4 Check the switchover automatic mode

shmOamHaSwitchoverAutomaticModeGet

Syntax:

```
ShmErrorT
shmOamHaSwitchoverAutomaticModeGet(
          ShmSessionIdT                          sessionId,
          ShmBoolT*                              autoMode);
```

Parameters:

`sessionId [in]`

Session identifier previously obtained using shmSessionOpen().

`autoMode [out]`

Flag that determines if automatic switchover mode is enabled. Otherwise, Shelf Manager operates in manual switchover mode.

Returns:

`SHM_CC_OK`

Switchover mode flag returned.

19.2.5 Manual switchover request

shmOamHaSwitchoverManual

Syntax:

```
ShmErrorT
shmOamHaSwitchoverManual(
        ShmSessionIdT                                   sessionId);
```

Description:

The request can be executed on an active CMM when initial synchronization is finished in active HA state. The request is unconditional, i.e. cannot be rejected due to standby CMM worse health. After the switchover is performed, the new active CMM enters manual switchover mode. It means the health checking is switched off and the switchover is performed only due to user request. The new active CMM stays in manual switchover mode until:

- standby CMM is lost, i.e. transition to active-no-standby

- switchover automatic command is issued on the new active CMM

- the new active CMM leaves in-service readiness state

Parameters:

`sessionId [in]`

Session identifier previously obtained using shmSessionOpen().

Returns:

`SHM_CC_ILLEGAL_CMD_FOR_HASTATE`

Invalid HA state.

`SHM_CC_OK`

Switchover done.

19.2.6 Automatic switchover mode request

shmOamHaSwitchoverAutomatic

Syntax:

```
ShmErrorT
shmOamHaSwitchoverAutomatic(
        ShmSessionIdT                                   sessionId);
```

Description:

The request can be executed only on an active CMM. The request switches on peer health comparision, which may result in a switchover.

Parameters:

`sessionId [in]`

Session identifier previously obtained using shmSessionOpen().

Returns:

`SHM_CC_OK`

Automatic switchover mode enabled.

19.2.7 Out-of-service readiness state transition request

shmOamHaOutOfService

Syntax:

```
ShmErrorT
shmOamHaOutOfService(
        ShmSessionIdT                                   sessionId);
```

Description:

This request is accepted in the in-service readiness state. The request initiates the transition to the out-of-service readiness state.

Parameters:

`sessionId [in]`

Session identifier previously obtained using shmSessionOpen().

Returns:

SHM_CC_ILLEGAL_CMD_FOR_HASTATE

Not in the in-service readiness state.

SHM_CC_OK

Transition to the out-of-service readiness state initiated.

19.2.8 In-service readiness state transition request

shmOamHaInService

Syntax:

```
ShmErrorT
shmOamHaInService(
        ShmSessionIdT                                   sessionId);
```

Description:

This request clears the OAM_HA_OOS_CAUSE_USER_COMMAND permanent cause for the out-of-service readiness state. When there are no other permanent causes for the out-of-service state the request causes the CMM to go to the election readiness state. The transient causes for the out-of-service state are cleared when the election readiness state is entered. If elected to be active or standby Shelf Manager, the CMM goes to the in-service readiness state.

Parameters:

`sessionId [in]`

Session identifier previously obtained using shmSessionOpen().

Returns:

SHM_CC_ILLEGAL_CMD_FOR_HASTATE

Not in out-of-service readiness state.

SHM_CC_ERROR

Other reasons force the out-of-service readiness state.

SHM_CC_OK

Transition to the election readiness state initiated.

19.2.9 Peer out-of-service readiness state transition request

shmOamHaPeerOutOfService

Syntax:

```
ShmErrorT
shmOamHaPeerOutOfService(
        ShmSessionIdT                                   sessionId);
```

Description:

This request is forwarded to the peer CMM.

Parameters:

`sessionId [in]`

Session identifier previously obtained using shmSessionOpen().

Returns:

`SHM_CC_OK`

Request forwarded.

19.2.10 Peer in-service readiness state transition request

shmOamHaPeerInService

Syntax:

```
ShmErrorT
shmOamHaPeerInService(
        ShmSessionIdT                                   sessionId);
```

Description:

This request is forwarded to the peer CMM.

Parameters:

`sessionId [in]`

Session identifier previously obtained using shmSessionOpen().

Returns:

`SHM_CC_OK`

Request forwarded.

19.2.11 Peer forced exit transition request

shmOamHaPeerForcedExit

Syntax:

```
ShmErrorT
shmOamHaPeerForcedExit(
        ShmSessionIdT                                   sessionId);
```

Description:

This request is forwarded to the peer CMM.

Parameters:

`sessionId [in]`

Session identifier previously obtained using shmSessionOpen().

Returns:

`SHM_CC_OK`

Request forwarded.

19.2.12 Legacy redundancy state get

shmOamHaLegacyRedundancyStateGet

Syntax:

```
ShmErrorT
shmOamHaLegacyRedundancyStateGet(
        ShmSessionIdT                   sessionId,
        ShmOamHaStateReadinessT*        readinessState,
        ShmOamHaStateHaT*               haState,
        ShmOamHaStatePresenceT*         otherPresence,
        ShmOamHaStateHaT*               otherHaState);
```

Description:

This request returns redundancy state.

Parameters:

`sessionId [in]`

Session identifier previously obtained using shmSessionOpen().

`readinessState [out]`

Current readiness state.

`haState [out]`

Current High Availability state.

`otherPresence [out]`

Other CMM presence.

`otherHaState [out]`

Known current High Availability state of the other CMM.

Returns:

`SHM_CC_OK`

Call success.

19.2.13 Initiate manual switchover

shmOamHaPeerSwitchoverManual

Syntax:

```
ShmErrorT
shmOamHaPeerSwitchoverManual(
        ShmSessionIdT                                   sessionId);
```

Description:

The request is forwarded to the peer active CMM and initiates on it the manual switchover procedure.

Parameters:

`sessionId [in]`

Session identifier previously obtained using shmSessionOpen().

Returns:

`SHM_CC_ILLEGAL_CMD_FOR_HASTATE`

Invalid HA state.

`SHM_CC_OK`

The request forwarded to the peer CMM.

19.2.14 Legacy failover operation request

shmOamHaLegacyFailover

Syntax:

```
ShmErrorT
shmOamHaLegacyFailover(
      ShmSessionIdT                                 sessionId,
      ShmBoolT                                      onlySameOrNewerVersion);
```

Parameters:

`sessionId [in]`

Session identifier previously obtained using shmSessionOpen().

`onlySameOrNewerVersion [in]`

True if failover only to CMM with the same or newer firmware version.

Returns:

SHM_CC_NO_STANDBY

Standby CMM is not physically present.

SHM_CC_STANDBY_STATE_UNKNOWN

There is no standby CMM (this CMM is not in the active HA state) or the active does not have the up to date standby health score.

SHM_CC_OLDER_FW

Standby has an older version and -v any was not specified..

SHM_CC_CRITICAL_EVENTS

Standby has the worse health score..

SHM_CC_ILLEGAL_CMD_FOR_HASTATE

The standby is present but the state is not synchronized.

SHM_CC_COMM_FAILED

Failover failed because peer disconnected.

SHM_CC_NOT_SYNCED

Standby is present but the state is not synchronized.

SHM_CC_OK

Call success.

19.2.15 Legacy standby reboot operation request

shmOamHaLegacyStandbyReboot

Syntax:

```
ShmErrorT
shmOamHaLegacyStandbyReboot(
        ShmSessionIdT                                   sessionId);
```

Description:

Note that the active/standby state is based on the HW line state, not the HA state. In consequence a CMM may be rebooted while being in the election readiness state.

Parameters:

`sessionId [in]`

Session identifier previously obtained using shmSessionOpen().

Returns:

`SHM_CC_NO_STANDBY`

Standby CMM is not physically present.

`SHM_CC_COMM_FAILED`

Communication with standby peer failed.

`SHM_CC_ILLEGAL_CMD_FOR_HASTATE`

Insufficient privilege level to reboot peer.

`SHM_CC_OK`

Call success.

19.2.16 Get current health score

shmOamHaHealthScoreGet

Syntax:

```
ShmErrorT
shmOamHaHealthScoreGet(
        ShmSessionIdT                   sessionId,
        ShmOamHaStateHaT*               haState,
        unsigned char*                  critical,
        unsigned char*                  major,
        unsigned char*                  minor,
        unsigned char*                  otherCritical,
        unsigned char*                  otherMajor,
        unsigned char*                  otherMinor);
```

Description:

In case the HA state is standby or active, also peer CMM health score is returned.

Parameters:

`sessionId [in]`

Session identifier previously obtained using shmSessionOpen().

`haState [out]`

This CMM HA state.

`critical [out]`

This CMM critical health score.

`major [out]`

This CMM major health score.

`minor [out]`

This CMM minor health score.

`otherCritical [out]`

Other CMM critical health score.

`otherMajor [out]`

Other CMM major health score.

`otherMinor [out]`

Other CMM minor health score.

Returns:

`SHM_CC_OK`

Call success.

20 SHM IPMI API Errors

20.1 IPMI Generic completion codes

Syntax:

```
#define IPMI_CC_OK                          0x00
#define IPMI_CC_NODE_BUSY                   0xC0
#define IPMI_CC_INVALID_CMD                 0xC1
#define IPMI_CC_INVALID_CMD_LUN             0xC2
#define IPMI_CC_TIMEOUT                     0xC3
#define IPMI_CC_OUT_OF_SPACE                0xC4
#define IPMI_CC_INVALID_RESERVATION         0xC5
#define IPMI_CC_DATA_TRUNCATED              0xC6
#define IPMI_CC_DATA_BAD_LEN                0xC7
#define IPMI_CC_DATA_TOO_LONG               0xC8
#define IPMI_CC_PARAMETER_OUT_OF_RANGE      0xC9
#define IPMI_CC_UNKNOWN_RESPONSE_DATA_SIZE  0xCA
#define IPMI_CC_ENTITY_NOT_PRESENT          0xCB
#define IPMI_CC_INVALID_DATA_FIELD          0xCC
#define IPMI_CC_ILLEGAL_CMD                 0xCD
#define IPMI_CC_NO_RESPONSE                 0xCE
#define IPMI_CC_DUPLICATE_REQUEST           0xCF
#define IPMI_CC_SDR_UPDATE_MODE             0xD0
#define IPMI_CC_FIRMWARE_UPDATE_MODE        0xD1
#define IPMI_CC_INITIALIZATION_IN_PROGRESS  0xD2
#define IPMI_CC_DESTINATION_UNAVAILABLE     0xD3
#define IPMI_CC_INSUFFICIENT_PRIVILEGE      0xD4
#define IPMI_CC_INVALID_CMD_FOR_HWSTATE     0xD5
#define IPMI_CC_CMD_UNAVAILABLE             0xD6
#define IPMI_CC_UNSPECIFIED                 0xFF
```

Members:

IPMI_CC_OK

Success.

Command completed normally.

IPMI_CC_NODE_BUSY

Node Busy.

Node Busy. Command could not be processed because command processing resources are temporarily unavailable.

IPMI_CC_INVALID_CMD

Invalid Command.

Invalid Command. Used to indicate an unrecognized or unsupported command.

IPMI_CC_INVALID_CMD_LUN

Invalid Command for given LUN.

IPMI_CC_TIMEOUT

Timeout while processing command.

Timeout while processing command. Response unavailable.

IPMI_CC_OUT_OF_SPACE

Out of Space.

Out of space. Command could not be completed because of a lack of storage space required to execute the given command operation.

IPMI_CC_INVALID_RESERVATION

Reservation Canceled or Invalid Reservation ID.

IPMI_CC_DATA_TRUNCATED

Request data truncated.

IPMI_CC_DATA_BAD_LEN

Request data length invalid.

IPMI_CC_DATA_TOO_LONG

Request data field length limit exceeded.

IPMI_CC_PARAMETER_OUT_OF_RANGE

Parameter out of range.

Parameter out of range. One or more parameters in the data field of the Request are out of range. This is different from 'Invalid data field' (CCh) code in that it indicates that the erroneous field(s) has a contiguous range of possible values.

IPMI_CC_UNKNOWN_RESPONSE_DATA_SIZE

Cannot return number of requested data bytes.

IPMI_CC_ENTITY_NOT_PRESENT

Requested Sensor, data, or record not present.

IPMI_CC_INVALID_DATA_FIELD

Invalid data field in Request.

IPMI_CC_ILLEGAL_CMD

Command illegal for specified sensor or record type.

IPMI_CC_NO_RESPONSE

Command response could not be provided.

IPMI_CC_DUPLICATE_REQUEST

Cannot execute duplicated request.

Cannot execute duplicated request. This completion code is for devices which cannot return the response that was returned for the original instance of the request. Such devices should provide separate commands that allow the completion status of the original request to be determined. An Event Receiver does not use this completion code, but returns the 00h completion code in the response to (valid) duplicated requests.

IPMI_CC_SDR_UPDATE_MODE

SDR Repository in update mode.

Command response could not be provided. SDR Repository in update mode.

IPMI_CC_FIRMWARE_UPDATE_MODE

Device in firmware update mode.

Command response could not be provided. Device in firmware update mode.

IPMI_CC_INITIALIZATION_IN_PROGRESS

BMC initialization or initialization agent in progress.

Command response could not be provided. BMC initialization or initialization agent in progress.

IPMI_CC_DESTINATION_UNAVAILABLE

Destination Unavailable.

Destination unavailable. Cannot deliver request to selected destination. E.g. this code can be returned if a request message is targeted to SMS, but receive message queue reception is disabled for the particular channel.

IPMI_CC_INSUFFICIENT_PRIVILEGE

Insufficient privilege level.

Cannot execute command due to insufficient privilege level or other security based restriction (e.g. disabled for ‘firmware firewall’).

IPMI_CC_INVALID_CMD_FOR_HWSTATE

Command, or request parameter(s), not supported in present state.

Cannot execute command. Command, or request parameter(s), not supported in present state.

IPMI_CC_CMD_UNAVAILABLE

Parameter is illegal because command sub-function has been disabled or is unavailable (e.g. disabled for ‘firmware firewall’).

Cannot execute command. Parameter is illegal because command sub-function has been disabled or is unavailable (e.g. disabled for ‘firmware firewall’).

IPMI_CC_UNSPECIFIED

Unspecified error.

Description:

IPMI Generic completion codes.

20.2 IPMI Command-specific completion codes

Syntax:

```
#define IPMI_CC_SET_SYSTEM_BOOT_OPTIONS_UNSUPP_PARAM                0x80
#define IPMI_CC_SET_SYSTEM_BOOT_OPTIONS_INVALID_ATTEMPT             0x81
#define IPMI_CC_SET_SYSTEM_BOOT_OPTIONS_READ_ONLY_PARAM             0x82
#define IPMI_CC_GET_SYSTEM_BOOT_OPTIONS_UNSUPP_PARAM                0x80
#define IPMI_CC_SET_PEF_CONFIG_PARAMS_UNSUP_PARAM                   0x80
#define IPMI_CC_SET_PEF_CONFIG_PARAMS_INVALID_ATTEMPT               0x81
#define IPMI_CC_SET_PEF_CONFIG_PARAMS_READ_ONLY_PARAM               0x82
#define IPMI_CC_GET_PEF_CONFIG_PARAMS_UNSUP_PARAM                   0x80
#define IPMI_CC_SET_LAST_EVENT_ID_SEL_ERASE_IN_PROGRESS             0x81
#define IPMI_CC_GET_LAST_EVENT_ID_SEL_ERASE_IN_PROGRESS             0x81
#define IPMI_CC_ALERT_IMMEDIATE_IPMI_SESSION_ACTIVE                 0x82
#define IPMI_CC_GET_DEVICE_SDR_RECORD_CHANGED                       0x80
#define IPMI_CC_SEND_MSG_INVALID_SESSION_HANDLE                     0x80
#define IPMI_CC_SEND_MSG_LOST_ARBITRATION                           0x81
#define IPMI_CC_SEND_MSG_BUS_ERROR                                  0x82
#define IPMI_CC_SEND_MSG_NAK_ON_WRITE                               0x83
#define IPMI_CC_READ_EVENT_MSG_BUFF_DATA_NOT_AVAILABLE              0x80
#define IPMI_CC_GET_SESSION_CHALLENGE_INVALID_USER_NAME             0x81
#define IPMI_CC_GET_SESSION_CHALLENGE_NULL_USER_NAME                0x82
#define IPMI_CC_ACTIVATE_SESSION_NO_SLOT_AVAILABLE                  0x81
#define IPMI_CC_ACTIVATE_SESSION_NO_SLOT_AVAILABLE_FOR_USER         0x82
#define IPMI_CC_ACTIVATE_SESSION_NO_SLOT_AVAILABLE_TO_PRIVILEGE     0x83
#define IPMI_CC_ACTIVATE_SESSION_SEQNUM_OUT_OF_RANGE                0x84
#define IPMI_CC_ACTIVATE_SESSION_INVALID_SESSION_ID                 0x85
#define IPMI_CC_ACTIVATE_SESSION_PRIVILEGE_LIMIT                    0x86
#define IPMI_CC_SET_SESSION_PRIVELEGE_LEVEL_NOT_AVAIL               0x80
#define IPMI_CC_SET_SESSION_PRIVELEGE_LEVEL_EXCEEDS_PRIVELEGE       0x81
#define IPMI_CC_SET_SESSION_PRIVELEGE_LEVEL_CANNOT_DISABLE          0x82
#define IPMI_CC_SET_CHANNEL_ACCESS_UNSUPP                           0x82
#define IPMI_CC_SET_CHANNEL_ACCESS_MODE_UNSUPP                      0x83
#define IPMI_CC_GET_CHANNEL_ACCESS_UNSUPP                           0x82
#define IPMI_CC_SET_USER_PASSWORD_FAILED                            0x80
#define IPMI_CC_SET_USER_PASSWORD_WRONG_SIZE                        0x81
```


```
#define    IPMI_CC_MASTER_WR_LOST_ARBITRATION                    0x81
#define    IPMI_CC_MASTER_WR_BUS_ERROR                           0x82
#define    IPMI_CC_MASTER_WR_NAK_ON_WRITE                        0x83
#define    IPMI_CC_MASTER_WR_TRUNC_READ                          0x84
#define    IPMI_CC_WRITE_FRU_DATA_WRITE_PROTECTED                0x80
#define    IPMI_CC_SET_CHANNEL_SECURITY_KEY_INSUFFICIENT_BYTES   0x81
#define    IPMI_CC_SET_CHANNEL_SECURITY_KEY_TOO_MANY_BYTES       0x82
#define    IPMI_CC_PARTIAL_ADD_SDR_REJECTED                      0x80
#define    IPMI_CC_GET_SEL_INFO_ERASE_IN_PROGRESS                0x81
#define    IPMI_CC_RESERVE_SEL_ERASE_IN_PROGRESS                 0x81
#define    IPMI_CC_GET_SEL_ENTRY_ERASE_IN_PROGRESS               0x81
#define    IPMI_CC_ADD_SEL_ENTRY_UNSUPP                          0x80
#define    IPMI_CC_ADD_SEL_ENTRY_ERASE_IN_PROGRESS               0x81
#define    IPMI_CC_PARTIAL_ADD_SEL_ENTRY_REJECTED                0x80
#define    IPMI_CC_PARTIAL_ADD_SEL_ENTRY_ERASE_IN_PROGRESS       0x81
#define    IPMI_CC_DELETE_SEL_ENTRY_ENTRY_UNSUPP                 0x80
#define    IPMI_CC_DELETE_SEL_ENTRY_ENTRY_ERASE_IN_PROGRESS      0x81
#define    IPMI_CC_SET_LAN_CONFIG_PARAMS_UNSUPP                  0x80
#define    IPMI_CC_SET_LAN_CONFIG_PARAMS_INVALID_ATTEMPT         0x81
#define    IPMI_CC_GET_LAN_CONFIG_PARAMS_UNSUPP                  0x80
#define    IPMI_CC_SET_CONFIGURATION_UNSUPP                      0x80
#define    IPMI_CC_SET_CONFIGURATION_INVALID_ATTEMPT             0x81
#define    IPMI_CC_SET_CONFIGURATION_READ_ONLY_PARAM             0x82
#define    IPMI_CC_SET_CONFIGURATION_WRITE_ONLY_PARAM            0x83
#define    IPMI_CC_GET_CONFIGURATION_UNSUPP                      0x80
#define    IPMI_CC_SEND_PPP_UDP_PROXY_PKT_LINK_DOWN              0x80
#define    IPMI_CC_SEND_PPP_UDP_PROXY_PKT_IP_DOWN                0x81
#define    IPMI_CC_GET_PPP_UDP_PROXY_PKT_NO_DATA                 0x80
#define    IPMI_CC_CALLBACK_ALERT_IN_PROGRESS                    0x81
#define    IPMI_CC_CALLBACK_SESSION_ACTIVE                       0x82
```

Members:

IPMI_CC_SET_SYSTEM_BOOT_OPTIONS_UNSUPP_PARAM

IPMI_CC_SET_SYSTEM_BOOT_OPTIONS_INVALID_ATTEMPT

IPMI_CC_SET_SYSTEM_BOOT_OPTIONS_READ_ONLY_PARAM

IPMI_CC_GET_SYSTEM_BOOT_OPTIONS_UNSUPP_PARAM

IPMI_CC_SET_PEF_CONFIG_PARAMS_UNSUP_PARAM

IPMI_CC_SET_PEF_CONFIG_PARAMS_INVALID_ATTEMPT

IPMI_CC_SET_PEF_CONFIG_PARAMS_READ_ONLY_PARAM

IPMI_CC_GET_PEF_CONFIG_PARAMS_UNSUP_PARAM

IPMI_CC_SET_LAST_EVENT_ID_SEL_ERASE_IN_PROGRESS

IPMI_CC_GET_LAST_EVENT_ID_SEL_ERASE_IN_PROGRESS

IPMI_CC_ALERT_IMMEDIATE_IPMI_SESSION_ACTIVE

IPMI_CC_GET_DEVICE_SDR_RECORD_CHANGED

IPMI_CC_SEND_MSG_INVALID_SESSION_HANDLE

IPMI_CC_SEND_MSG_LOST_ARBITRATION

IPMI_CC_SEND_MSG_BUS_ERROR

IPMI_CC_SEND_MSG_NAK_ON_WRITE

IPMI_CC_READ_EVENT_MSG_BUFF_DATA_NOT_AVAILABLE

IPMI_CC_GET_SESSION_CHALLENGE_INVALID_USER_NAME

IPMI_CC_GET_SESSION_CHALLENGE_NULL_USER_NAME

IPMI_CC_ACTIVATE_SESSION_NO_SLOT_AVAILABLE

IPMI_CC_ACTIVATE_SESSION_NO_SLOT_AVAILABLE_FOR_USER

IPMI_CC_ACTIVATE_SESSION_NO_SLOT_AVAILABLE_TO_PRIVILEGE

IPMI_CC_ACTIVATE_SESSION_SEQNUM_OUT_OF_RANGE

IPMI_CC_ACTIVATE_SESSION_INVALID_SESSION_ID

IPMI_CC_ACTIVATE_SESSION_PRIVILEGE_LIMIT

IPMI_CC_SET_SESSION_PRIVELEGE_LEVEL_NOT_AVAIL

IPMI_CC_SET_SESSION_PRIVELEGE_LEVEL_EXCEEDS_PRIVELEGE

IPMI_CC_SET_SESSION_PRIVELEGE_LEVEL_CANNOT_DISABLE

IPMI_CC_SET_CHANNEL_ACCESS_UNSUPP

IPMI_CC_SET_CHANNEL_ACCESS_MODE_UNSUPP

IPMI_CC_GET_CHANNEL_ACCESS_UNSUPP

IPMI_CC_SET_USER_PASSWORD_FAILED

IPMI_CC_SET_USER_PASSWORD_WRONG_SIZE

IPMI_CC_MASTER_WR_LOST_ARBITRATION

IPMI_CC_MASTER_WR_BUS_ERROR

IPMI_CC_MASTER_WR_NAK_ON_WRITE

IPMI_CC_MASTER_WR_TRUNC_READ

IPMI_CC_WRITE_FRU_DATA_WRITE_PROTECTED

IPMI_CC_PARTIAL_ADD_SDR_REJECTED

IPMI_CC_GET_SEL_INFO_ERASE_IN_PROGRESS

IPMI_CC_RESERVE_SEL_ERASE_IN_PROGRESS

IPMI_CC_GET_SEL_ENTRY_ERASE_IN_PROGRESS

IPMI_CC_ADD_SEL_ENTRY_UNSUPP

IPMI_CC_ADD_SEL_ENTRY_ERASE_IN_PROGRESS

IPMI_CC_PARTIAL_ADD_SEL_ENTRY_REJECTED

IPMI_CC_PARTIAL_ADD_SEL_ENTRY_ERASE_IN_PROGRESS

IPMI_CC_DELETE_SEL_ENTRY_ENTRY_UNSUPP

IPMI_CC_DELETE_SEL_ENTRY_ENTRY_ERASE_IN_PROGRESS

IPMI_CC_SET_LAN_CONFIG_PARAMS_UNSUPP

IPMI_CC_SET_LAN_CONFIG_PARAMS_INVALID_ATTEMPT

IPMI_CC_GET_LAN_CONFIG_PARAMS_UNSUPP

IPMI_CC_SET_CONFIGURATION_UNSUPP

IPMI_CC_SET_CONFIGURATION_INVALID_ATTEMPT

IPMI_CC_SET_CONFIGURATION_READ_ONLY_PARAM

IPMI_CC_SET_CONFIGURATION_WRITE_ONLY_PARAM

IPMI_CC_GET_CONFIGURATION_UNSUPP

IPMI_CC_SEND_PPP_UDP_PROXY_PKT_LINK_DOWN

IPMI_CC_SEND_PPP_UDP_PROXY_PKT_IP_DOWN

IPMI_CC_GET_PPP_UDP_PROXY_PKT_NO_DATA

IPMI_CC_CALLBACK_ALERT_IN_PROGRESS

IPMI_CC_CALLBACK_SESSION_ACTIVE

Description:

IPMI Command-specific completion codes.

20.3 IPMI Device-specific (OEM) completion codes

Syntax:

```
#define   IPMI_CC_EMP_SESSION_ACTIVE     0x02
#define   IPMI_CC_NON_IPMI_BLADE         0x05
#define   IPMI_CC_INVALID_PASSWORD       0x10
#define   IPMI_CC_DPC_SESSION_ACTIVE     0x11
#define   IPMI_CC_DPC_SESSION_AUTH       0x12
```

Members:

IPMI_CC_EMP_SESSION_ACTIVE

DPC Open Session only

IPMI_CC_NON_IPMI_BLADE

IPMI_CC_INVALID_PASSWORD

DPC Authenticate only

IPMI_CC_DPC_SESSION_ACTIVE

DPC Open Session and EMP Submit Password only

IPMI_CC_DPC_SESSION_AUTH

DPC Open Session only(?)

Description:

IPMI Device-specific (OEM) completion codes.

20.4 NetFn values

Syntax:

```
#define   IPMI_NETFN_CHASSIS          0x00
#define   IPMI_NETFN_BRIDGE           0x02
#define   IPMI_NETFN_SENSOR_EVENT     0x04
#define   IPMI_NETFN_APP              0x06
#define   IPMI_NETFN_FIRMWARE         0x08
#define   IPMI_NETFN_STORAGE          0x0a
#define   IPMI_NETFN_TRANSPORT        0x0c
#define   IPMI_NETFN_PICMG            0x2c
#define   IPMI_NETFN_OEM_GROUP        0x2e
#define   IPMI_NETFN_INTEL_CMM        0x3c
#define   IPMI_NETFN_INTEL_LISM       0x32
```

Members:

IPMI_NETFN_CHASSIS

IPMI_NETFN_BRIDGE

IPMI_NETFN_SENSOR_EVENT

IPMI_NETFN_APP

IPMI_NETFN_FIRMWARE

IPMI_NETFN_STORAGE

IPMI_NETFN_TRANSPORT

IPMI_NETFN_PICMG

IPMI_NETFN_OEM_GROUP

IPMI_NETFN_INTEL_CMM

IPMI_NETFN_INTEL_LISM

Description:

NetFn values.

20.5 Chassis netfn - Chassis Device Commands

Syntax:

```
#define   IPMI_CMD_GET_CHASSIS_CAPABILITIES          0x00
#define   IPMI_CMD_GET_CHASSIS_STATUS                0x01
#define   IPMI_CMD_CHASSIS_CONTROL                   0x02
#define   IPMI_CMD_CHASSIS_RESET                     0x03
#define   IPMI_CMD_CHASSIS_IDENTIFY                  0x04
#define   IPMI_CMD_SET_CHASSIS_CAPABILITIES          0x05
#define   IPMI_CMD_SET_POWER_RESTORE_POLICY          0x06
#define   IPMI_CMD_GET_SYSTEM_RESTART_CAUSE          0x07
#define   IPMI_CMD_SET_SYSTEM_BOOT_OPTIONS           0x08
#define   IPMI_CMD_GET_SYSTEM_BOOT_OPTIONS           0x09
#define   IPMI_CMD_GET_POH_COUNTER                   0x0f
```

Members:

IPMI_CMD_GET_CHASSIS_CAPABILITIES

IPMI_CMD_GET_CHASSIS_STATUS

IPMI_CMD_CHASSIS_CONTROL

IPMI_CMD_CHASSIS_RESET

IPMI_CMD_CHASSIS_IDENTIFY

IPMI_CMD_SET_CHASSIS_CAPABILITIES

IPMI_CMD_SET_POWER_RESTORE_POLICY

IPMI_CMD_GET_SYSTEM_RESTART_CAUSE

IPMI_CMD_SET_SYSTEM_BOOT_OPTIONS

IPMI_CMD_GET_SYSTEM_BOOT_OPTIONS

IPMI_CMD_GET_POH_COUNTER

Description:

Chassis netfn - Chassis Device Commands.

20.6 Bridge netfn - Bridge Management Commands (ICMB)

Syntax:

```
#define     IPMI_CMD_GET_BRIDGE_STATE                0x00
#define     IPMI_CMD_SET_BRIDGE_STATE                0x01
#define     IPMI_CMD_GET_ICMB_ADDRESS                0x02
#define     IPMI_CMD_SET_ICMB_ADDRESS                0x03
#define     IPMI_CMD_SET_BRIDGE_PROXY_ADDRESS        0x04
#define     IPMI_CMD_GET_BRIDGE_STATISTICS           0x05
#define     IPMI_CMD_GET_ICMB_CAPABILITIES           0x06
#define     IPMI_CMD_CLEAR_BRIDGE_STATISTICS         0x08
#define     IPMI_CMD_GET_BRIDGE_PROXY_ADDRESS        0x09
#define     IPMI_CMD_GET_ICMB_CONNECTOR_INFO         0x0a
#define     IPMI_CMD_SET_ICMB_CONNECTOR_INFO         0x0b
#define     IPMI_CMD_SEND_ICMB_CONNECTION_ID         0x0c
```

Members:

IPMI_CMD_GET_BRIDGE_STATE

IPMI_CMD_SET_BRIDGE_STATE

IPMI_CMD_GET_ICMB_ADDRESS

IPMI_CMD_SET_ICMB_ADDRESS

IPMI_CMD_SET_BRIDGE_PROXY_ADDRESS

IPMI_CMD_GET_BRIDGE_STATISTICS

IPMI_CMD_GET_ICMB_CAPABILITIES

IPMI_CMD_CLEAR_BRIDGE_STATISTICS

IPMI_CMD_GET_BRIDGE_PROXY_ADDRESS

IPMI_CMD_GET_ICMB_CONNECTOR_INFO

IPMI_CMD_SET_ICMB_CONNECTOR_INFO

IPMI_CMD_SEND_ICMB_CONNECTION_ID

Description:

Bridge netfn - Bridge Management Commands (ICMB).

20.7 Bridge netfn - Discovery Commands (ICMB)

Syntax:

```
#define    IPMI_CMD_PREPARE_FOR_DISCOVERY          0x10
#define    IPMI_CMD_GET_ADDRESSES                  0x11
#define    IPMI_CMD_SET_DISCOVERED                 0x12
#define    IPMI_CMD_GET_CHASSIS_DEVICE_ID          0x13
#define    IPMI_CMD_SET_CHASSIS_DEVICE_ID          0x14
```

Members:

IPMI_CMD_PREPARE_FOR_DISCOVERY

IPMI_CMD_GET_ADDRESSES

IPMI_CMD_SET_DISCOVERED

IPMI_CMD_GET_CHASSIS_DEVICE_ID

IPMI_CMD_SET_CHASSIS_DEVICE_ID

Description:

Bridge netfn - Discovery Commands (ICMB).

20.8 Bridge netfn - Bridging Commands (ICMB)

Syntax:

```
#define  IPMI_CMD_BRIDGE_REQUEST   0x20
#define  IPMI_CMD_BRIDGE_MESSAGE   0x21
```

Members:

IPMI_CMD_BRIDGE_REQUEST

IPMI_CMD_BRIDGE_MESSAGE

Description:

Bridge netfn - Bridging Commands (ICMB).

20.9 Bridge netfn - Event Commands (ICMB)

Syntax:

```
#define    IPMI_CMD_GET_EVENT_COUNT                    0x30
#define    IPMI_CMD_SET_EVENT_DESTINATION              0x31
#define    IPMI_CMD_SET_EVENT_RECEPTION_STATE          0x32
#define    IPMI_CMD_SEND_ICMB_EVENT_MESSAGE            0x33
#define    IPMI_CMD_GET_EVENT_DESTIATION               0x34
#define    IPMI_CMD_GET_EVENT_RECEPTION_STATE          0x35
```

Members:

IPMI_CMD_GET_EVENT_COUNT

IPMI_CMD_SET_EVENT_DESTINATION

IPMI_CMD_SET_EVENT_RECEPTION_STATE

IPMI_CMD_SEND_ICMB_EVENT_MESSAGE

IPMI_CMD_GET_EVENT_DESTIATION

IPMI_CMD_GET_EVENT_RECEPTION_STATE

Description:

Bridge netfn - Event Commands (ICMB).

20.10 Bridge netfn - Other Bridge Commands

Syntax:

```
#define    IPMI_CMD_ERROR_REPORT    0xff
```

Members:

IPMI_CMD_ERROR_REPORT

Description:

Bridge netfn - Other Bridge Commands.

20.11 Sensor Event netfn - Event Commands

Syntax:

```
#define  IPMI_CMD_SET_EVENT_RECEIVER   0x00
#define  IPMI_CMD_GET_EVENT_RECEIVER   0x01
#define  IPMI_CMD_PLATFORM_EVENT       0x02
```

Members:

IPMI_CMD_SET_EVENT_RECEIVER

IPMI_CMD_GET_EVENT_RECEIVER

IPMI_CMD_PLATFORM_EVENT

Description:

Sensor Event netfn - Event Commands.

20.12 Sensor Event netfn - PEF and Allerting Commands

Syntax:

```
#define    IPMI_CMD_GET_PEF_CAPABILITIES                0x10
#define    IPMI_CMD_ARM_PEF_POSTPONE_TIMER              0x11
#define    IPMI_CMD_SET_PEF_CONFIG_PARMS                0x12
#define    IPMI_CMD_GET_PEF_CONFIG_PARMS                0x13
#define    IPMI_CMD_SET_LAST_PROCESSED_EVENT_ID         0x14
#define    IPMI_CMD_GET_LAST_PROCESSED_EVENT_ID         0x15
#define    IPMI_CMD_ALERT_IMMEDIATE                     0x16
#define    IPMI_CMD_PET_ACKNOWLEDGE                     0x17
```

Members:

IPMI_CMD_GET_PEF_CAPABILITIES

IPMI_CMD_ARM_PEF_POSTPONE_TIMER

IPMI_CMD_SET_PEF_CONFIG_PARMS

IPMI_CMD_GET_PEF_CONFIG_PARMS

IPMI_CMD_SET_LAST_PROCESSED_EVENT_ID

IPMI_CMD_GET_LAST_PROCESSED_EVENT_ID

IPMI_CMD_ALERT_IMMEDIATE

IPMI_CMD_PET_ACKNOWLEDGE

Description:

Sensor Event netfn - PEF and Allerting Commands.

20.13 App netfn - IPM Device Global Commands

Syntax:

```
#define    IPMI_CMD_GET_DEVICE_ID                  0x01
#define    IPMI_CMD_BROADCAST_GET_DEVICE_ID        0x01
#define    IPMI_CMD_COLD_RESET                     0x02
#define    IPMI_CMD_WARM_RESET                     0x03
#define    IPMI_CMD_GET_SELF_TEST_RESULTS          0x04
#define    IPMI_CMD_MANUFACTURING_TEST_ON          0x05
#define    IPMI_CMD_SET_ACPI_POWER_STATE           0x06
#define    IPMI_CMD_GET_ACPI_POWER_STATE           0x07
#define    IPMI_CMD_GET_DEVICE_GUID                0x08
```

Members:

IPMI_CMD_GET_DEVICE_ID

IPMI_CMD_BROADCAST_GET_DEVICE_ID

IPMI_CMD_COLD_RESET

IPMI_CMD_WARM_RESET

IPMI_CMD_GET_SELF_TEST_RESULTS

IPMI_CMD_MANUFACTURING_TEST_ON

IPMI_CMD_SET_ACPI_POWER_STATE

IPMI_CMD_GET_ACPI_POWER_STATE

IPMI_CMD_GET_DEVICE_GUID

Description:

App netfn - IPM Device Global Commands.

20.14 App netfn - BMC Watchdog Timer Commands

Syntax:

```
#define    IPMI_CMD_RESET_WATCHDOG_TIMER          0x22
#define    IPMI_CMD_SET_WATCHDOG_TIMER            0x24
#define    IPMI_CMD_GET_WATCHDOG_TIMER            0x25
```

Members:

IPMI_CMD_RESET_WATCHDOG_TIMER

IPMI_CMD_SET_WATCHDOG_TIMER

IPMI_CMD_GET_WATCHDOG_TIMER

Description:

App netfn - BMC Watchdog Timer Commands.

20.15 App netfn - BMC Device and Messaging Commands

Syntax:

```
#define    IPMI_CMD_SET_BMC_GLOBAL_ENABLES                  0x2e
#define    IPMI_CMD_GET_BMC_GLOBAL_ENABLES                  0x2f
#define    IPMI_CMD_CLEAR_MSG_FLAGS                         0x30
#define    IPMI_CMD_GET_MSG_FLAGS                           0x31
#define    IPMI_CMD_ENABLE_MESSAGE_CHANNEL_RCV              0x32
#define    IPMI_CMD_GET_MSG                                 0x33
#define    IPMI_CMD_SEND_MSG                                0x34
#define    IPMI_CMD_READ_EVENT_MSG_BUFFER                   0x35
#define    IPMI_CMD_GET_BT_INTERFACE_CAPABILITIES           0x36
#define    IPMI_CMD_GET_SYSTEM_GUID                         0x37
#define    IPMI_CMD_GET_CHANNEL_AUTH_CAPABILITIES           0x38
#define    IPMI_CMD_GET_SESSION_CHALLENGE                   0x39
#define    IPMI_CMD_ACTIVATE_SESSION                        0x3a
#define    IPMI_CMD_SET_SESSION_PRIVILEGE                   0x3b
#define    IPMI_CMD_CLOSE_SESSION                           0x3c
#define    IPMI_CMD_GET_SESSION_INFO                        0x3d
#define    IPMI_CMD_GET_AUTHCODE                            0x3f
#define    IPMI_CMD_SET_CHANNEL_ACCESS                      0x40
#define    IPMI_CMD_GET_CHANNEL_ACCESS                      0x41
#define    IPMI_CMD_GET_CHANNEL_INFO                        0x42
#define    IPMI_CMD_SET_USER_ACCESS                         0x43
#define    IPMI_CMD_GET_USER_ACCESS                         0x44
#define    IPMI_CMD_SET_USER_NAME                           0x45
#define    IPMI_CMD_GET_USER_NAME                           0x46
#define    IPMI_CMD_SET_USER_PASSWORD                       0x47
#define    IPMI_CMD_ACTIVATE_PAYLOAD                        0x48
#define    IPMI_CMD_DEACTIVATE_PAYLOAD                      0x49
#define    IPMI_CMD_GET_PAYLOAD_ACTIVATION_STATUS           0x4a
#define    IPMI_CMD_GET_PAYLOAD_INSTANCE_INFO               0x4b
#define    IPMI_CMD_SET_USER_PAYLOAD_ACCESS                 0x4c
#define    IPMI_CMD_GET_USER_PAYLOAD_ACCESS                 0x4d
#define    IPMI_CMD_GET_CHANNEL_PAYLOAD_SUPPORT             0x4e
#define    IPMI_CMD_GET_CHANNEL_PAYLOAD_VERSION             0x4f
#define    IPMI_CMD_GET_CHANNEL_OEM_PAYLOAD_INFO            0x50
#define    IPMI_CMD_MASTER_READ_WRITE                       0x52
#define    IPMI_CMD_GET_CHANNEL_CIPHER_SUITES               0x54
#define    IPMI_CMD_SUSPEND_RESUME_PAYLOAD_ENCRYPTION       0x55
#define    IPMI_CMD_SET_CHANNEL_SECURITY_KEY                0x56
#define    IPMI_CMD_GET_SYSTEM_INTERFACE_CAPABILITIES       0x57
```

Members:

IPMI_CMD_SET_BMC_GLOBAL_ENABLES

IPMI_CMD_GET_BMC_GLOBAL_ENABLES

IPMI_CMD_CLEAR_MSG_FLAGS

IPMI_CMD_GET_MSG_FLAGS

IPMI_CMD_ENABLE_MESSAGE_CHANNEL_RCV

IPMI_CMD_GET_MSG

IPMI_CMD_SEND_MSG

IPMI_CMD_READ_EVENT_MSG_BUFFER

IPMI_CMD_GET_BT_INTERFACE_CAPABILITIES

IPMI_CMD_GET_SYSTEM_GUID

IPMI_CMD_GET_CHANNEL_AUTH_CAPABILITIES

IPMI_CMD_GET_SESSION_CHALLENGE

IPMI_CMD_ACTIVATE_SESSION

IPMI_CMD_SET_SESSION_PRIVILEGE

IPMI_CMD_CLOSE_SESSION

IPMI_CMD_GET_SESSION_INFO

IPMI_CMD_GET_AUTHCODE

IPMI_CMD_SET_CHANNEL_ACCESS

IPMI_CMD_GET_CHANNEL_ACCESS

IPMI_CMD_GET_CHANNEL_INFO

IPMI_CMD_SET_USER_ACCESS

IPMI_CMD_GET_USER_ACCESS

IPMI_CMD_SET_USER_NAME

IPMI_CMD_GET_USER_NAME

IPMI_CMD_SET_USER_PASSWORD

IPMI_CMD_ACTIVATE_PAYLOAD

IPMI_CMD_DEACTIVATE_PAYLOAD

IPMI_CMD_GET_PAYLOAD_ACTIVATION_STATUS

IPMI_CMD_GET_PAYLOAD_INSTANCE_INFO

IPMI_CMD_SET_USER_PAYLOAD_ACCESS

IPMI_CMD_GET_USER_PAYLOAD_ACCESS

IPMI_CMD_GET_CHANNEL_PAYLOAD_SUPPORT

IPMI_CMD_GET_CHANNEL_PAYLOAD_VERSION

IPMI_CMD_GET_CHANNEL_OEM_PAYLOAD_INFO

IPMI_CMD_MASTER_READ_WRITE

IPMI_CMD_GET_CHANNEL_CIPHER_SUITES

IPMI_CMD_SUSPEND_RESUME_PAYLOAD_ENCRYPTION

IPMI_CMD_SET_CHANNEL_SECURITY_KEY

IPMI_CMD_GET_SYSTEM_INTERFACE_CAPABILITIES

Description:

App netfn - BMC Device and Messaging Commands.

20.16 Storage netfn - Sensor Device Commands

Syntax:

```
#define    IPMI_CMD_GET_DEVICE_SDR_INFO                0x20
#define    IPMI_CMD_GET_DEVICE_SDR                     0x21
#define    IPMI_CMD_RESERVE_DEVICE_SDR_REPOSITORY      0x22
#define    IPMI_CMD_GET_SENSOR_READING_FACTORS         0x23
#define    IPMI_CMD_SET_SENSOR_HYSTERESIS              0x24
#define    IPMI_CMD_GET_SENSOR_HYSTERESIS              0x25
#define    IPMI_CMD_SET_SENSOR_THRESHOLD               0x26
#define    IPMI_CMD_GET_SENSOR_THRESHOLD               0x27
#define    IPMI_CMD_SET_SENSOR_EVENT_ENABLE            0x28
#define    IPMI_CMD_GET_SENSOR_EVENT_ENABLE            0x29
#define    IPMI_CMD_REARM_SENSOR_EVENTS                0x2a
#define    IPMI_CMD_GET_SENSOR_EVENT_STATUS            0x2b
#define    IPMI_CMD_GET_SENSOR_READING                 0x2d
#define    IPMI_CMD_SET_SENSOR_TYPE                    0x2e
#define    IPMI_CMD_GET_SENSOR_TYPE                    0x2f
```

Members:

IPMI_CMD_GET_DEVICE_SDR_INFO

IPMI_CMD_GET_DEVICE_SDR

IPMI_CMD_RESERVE_DEVICE_SDR_REPOSITORY

IPMI_CMD_GET_SENSOR_READING_FACTORS

IPMI_CMD_SET_SENSOR_HYSTERESIS

IPMI_CMD_GET_SENSOR_HYSTERESIS

IPMI_CMD_SET_SENSOR_THRESHOLD

IPMI_CMD_GET_SENSOR_THRESHOLD

IPMI_CMD_SET_SENSOR_EVENT_ENABLE

IPMI_CMD_GET_SENSOR_EVENT_ENABLE

IPMI_CMD_REARM_SENSOR_EVENTS

IPMI_CMD_GET_SENSOR_EVENT_STATUS

IPMI_CMD_GET_SENSOR_READING

IPMI_CMD_SET_SENSOR_TYPE

IPMI_CMD_GET_SENSOR_TYPE

Description:

Storage netfn - Sensor Device Commands.

20.17 Storage netfn - FRU Device Commands

Syntax:

```
#define    IPMI_CMD_GET_FRU_INVENTORY_AREA_INFO      0x10
#define    IPMI_CMD_READ_FRU_DATA                    0x11
#define    IPMI_CMD_WRITE_FRU_DATA                   0x12
```

Members:

IPMI_CMD_GET_FRU_INVENTORY_AREA_INFO

IPMI_CMD_READ_FRU_DATA

IPMI_CMD_WRITE_FRU_DATA

Description:

Storage netfn - FRU Device Commands.

20.18 Storage netfn - SDR Device Commands

Syntax:

```
#define   IPMI_CMD_GET_SDR_REPOSITORY_INFO              0x20
#define   IPMI_CMD_GET_SDR_REPOSITORY_ALLOC_INFO        0x21
#define   IPMI_CMD_RESERVE_SDR_REPOSITORY               0x22
#define   IPMI_CMD_GET_SDR                              0x23
#define   IPMI_CMD_ADD_SDR                              0x24
#define   IPMI_CMD_PARTIAL_ADD_SDR                      0x25
#define   IPMI_CMD_DELETE_SDR                           0x26
#define   IPMI_CMD_CLEAR_SDR_REPOSITORY                 0x27
#define   IPMI_CMD_GET_SDR_REPOSITORY_TIME              0x28
#define   IPMI_CMD_SET_SDR_REPOSITORY_TIME              0x29
#define   IPMI_CMD_ENTER_SDR_REPOSITORY_UPDATE          0x2a
#define   IPMI_CMD_EXIT_SDR_REPOSITORY_UPDATE           0x2b
#define   IPMI_CMD_RUN_INITIALIZATION_AGENT             0x2c
```

Members:

IPMI_CMD_GET_SDR_REPOSITORY_INFO

IPMI_CMD_GET_SDR_REPOSITORY_ALLOC_INFO

IPMI_CMD_RESERVE_SDR_REPOSITORY

IPMI_CMD_GET_SDR

IPMI_CMD_ADD_SDR

IPMI_CMD_PARTIAL_ADD_SDR

IPMI_CMD_DELETE_SDR

IPMI_CMD_CLEAR_SDR_REPOSITORY

IPMI_CMD_GET_SDR_REPOSITORY_TIME

IPMI_CMD_SET_SDR_REPOSITORY_TIME

IPMI_CMD_ENTER_SDR_REPOSITORY_UPDATE

IPMI_CMD_EXIT_SDR_REPOSITORY_UPDATE

IPMI_CMD_RUN_INITIALIZATION_AGENT

Description:

Storage netfn - SDR Device Commands.

20.19 Storage netfn - SEL Device Commands

Syntax:

```
#define    IPMI_CMD_GET_SEL_INFO                0x40
#define    IPMI_CMD_GET_SEL_ALLOCATION_INFO     0x41
#define    IPMI_CMD_RESERVE_SEL                 0x42
#define    IPMI_CMD_GET_SEL_ENTRY               0x43
#define    IPMI_CMD_ADD_SEL_ENTRY               0x44
#define    IPMI_CMD_PARTIAL_ADD_SEL_ENTRY       0x45
#define    IPMI_CMD_DELETE_SEL_ENTRY            0x46
#define    IPMI_CMD_CLEAR_SEL                   0x47
#define    IPMI_CMD_GET_SEL_TIME                0x48
#define    IPMI_CMD_SET_SEL_TIME                0x49
#define    IPMI_CMD_GET_AUXILIARY_LOG_STATUS    0x5a
#define    IPMI_CMD_SET_AUXILIARY_LOG_STATUS    0x5b
```

Members:

IPMI_CMD_GET_SEL_INFO

IPMI_CMD_GET_SEL_ALLOCATION_INFO

IPMI_CMD_RESERVE_SEL

IPMI_CMD_GET_SEL_ENTRY

IPMI_CMD_ADD_SEL_ENTRY

IPMI_CMD_PARTIAL_ADD_SEL_ENTRY

IPMI_CMD_DELETE_SEL_ENTRY

IPMI_CMD_CLEAR_SEL

IPMI_CMD_GET_SEL_TIME

IPMI_CMD_SET_SEL_TIME

IPMI_CMD_GET_AUXILIARY_LOG_STATUS

IPMI_CMD_SET_AUXILIARY_LOG_STATUS

Description:

Storage netfn - SEL Device Commands.

20.20 Transport netfn - LAN Device Commands

Syntax:

```
#define   IPMI_CMD_SET_LAN_CONFIG_PARMS          0x01
#define   IPMI_CMD_GET_LAN_CONFIG_PARMS          0x02
#define   IPMI_CMD_SUSPEND_BMC_ARPS              0x03
#define   IPMI_CMD_GET_IP_UDP_RMCP_STATS         0x04
```

Members:

IPMI_CMD_SET_LAN_CONFIG_PARMS

IPMI_CMD_GET_LAN_CONFIG_PARMS

IPMI_CMD_SUSPEND_BMC_ARPS

IPMI_CMD_GET_IP_UDP_RMCP_STATS

Description:

Transport netfn - LAN Device Commands.

20.21 Transport netfn - Serial/Modem Device Commands

Syntax:

```
#define   IPMI_CMD_SET_SERIAL_MODEM_CONFIG                0x10
#define   IPMI_CMD_GET_SERIAL_MODEM_CONFIG                0x11
#define   IPMI_CMD_SET_SERIAL_MODEM_MUX                   0x12
#define   IPMI_CMD_GET_TAP_RESPONSE_CODES                 0x13
#define   IPMI_CMD_SET_PPP_UDP_PROXY_XMIT_DATA            0x14
#define   IPMI_CMD_GET_PPP_UDP_PROXY_XMIT_DATA            0x15
#define   IPMI_CMD_SEND_PPP_UDP_PROXY_PACKET              0x16
#define   IPMI_CMD_GET_PPP_UDP_PROXY_RECV_DATA            0x17
#define   IPMI_CMD_SERIAL_MODEM_CONN_ACTIVE               0x18
#define   IPMI_CMD_CALLBACK                               0x19
#define   IPMI_CMD_SET_USER_CALLBACK_OPTIONS              0x1a
#define   IPMI_CMD_GET_USER_CALLBACK_OPTIONS              0x1b
#define   IPMI_CMD_SOL_ACTIVATING                         0x20
#define   IPMI_CMD_SET_SOL_CONFIGURATION_PARAMETERS       0x21
#define   IPMI_CMD_GET_SOL_CONFIGURATION_PARAMETERS       0x22
```

Members:

IPMI_CMD_SET_SERIAL_MODEM_CONFIG

IPMI_CMD_GET_SERIAL_MODEM_CONFIG

IPMI_CMD_SET_SERIAL_MODEM_MUX

IPMI_CMD_GET_TAP_RESPONSE_CODES

IPMI_CMD_SET_PPP_UDP_PROXY_XMIT_DATA

IPMI_CMD_GET_PPP_UDP_PROXY_XMIT_DATA

IPMI_CMD_SEND_PPP_UDP_PROXY_PACKET

IPMI_CMD_GET_PPP_UDP_PROXY_RECV_DATA

IPMI_CMD_SERIAL_MODEM_CONN_ACTIVE

IPMI_CMD_CALLBACK

IPMI_CMD_SET_USER_CALLBACK_OPTIONS

IPMI_CMD_GET_USER_CALLBACK_OPTIONS

IPMI_CMD_SOL_ACTIVATING

IPMI_CMD_SET_SOL_CONFIGURATION_PARAMETERS

IPMI_CMD_GET_SOL_CONFIGURATION_PARAMETERS

Description:

Transport netfn - Serial/Modem Device Commands.

20.22 PICMG netfn

Syntax:

```
#define    IPMI_CMD_GET_PICMG_PROPERTIES                0x00
#define    IPMI_CMD_GET_ADDRESS_INFO                    0x01
#define    IPMI_CMD_GET_SHELF_ADDRESS_INFO              0x02
#define    IPMI_CMD_SET_SHELF_ADDRESS_INFO              0x03
#define    IPMI_CMD_FRU_CONTROL                         0x04
#define    IPMI_CMD_GET_FRU_LED_PROPERTIES              0x05
#define    IPMI_CMD_GET_LED_COLOR_CAPABILITIES          0x06
#define    IPMI_CMD_SET_FRU_LED_STATE                   0x07
#define    IPMI_CMD_GET_FRU_LED_STATE                   0x08
#define    IPMI_CMD_SET_IPMB_STATE                      0x09
#define    IPMI_CMD_SET_FRU_ACTIVATION_POLICY           0x0A
#define    IPMI_CMD_GET_FRU_ACTIVATION_POLICY           0x0B
#define    IPMI_CMD_SET_FRU_ACTIVATION                  0x0C
#define    IPMI_CMD_GET_DEVICE_LOCATOR_RECORD_ID        0x0D
#define    IPMI_CMD_SET_PORT_STATE                      0x0E
#define    IPMI_CMD_GET_PORT_STATE                      0x0F
#define    IPMI_CMD_COMPUTE_POWER_PROPERTIES            0x10
#define    IPMI_CMD_SET_POWER_LEVEL                     0x11
#define    IPMI_CMD_GET_POWER_LEVEL                     0x12
#define    IPMI_CMD_RENEGOTIATE_POWER                   0x13
#define    IPMI_CMD_GET_FAN_SPEED_PROPERTIES            0x14
#define    IPMI_CMD_SET_FAN_LEVEL                       0x15
#define    IPMI_CMD_GET_FAN_LEVEL                       0x16
#define    IPMI_CMD_BUSED_RESOURCE_CONTROL              0x17
#define    IPMI_CMD_GET_IPMB_LINK_INFO                  0x18
```

Members:

IPMI_CMD_GET_PICMG_PROPERTIES

IPMI_CMD_GET_ADDRESS_INFO

IPMI_CMD_GET_SHELF_ADDRESS_INFO

IPMI_CMD_SET_SHELF_ADDRESS_INFO

IPMI_CMD_FRU_CONTROL

IPMI_CMD_GET_FRU_LED_PROPERTIES

IPMI_CMD_GET_LED_COLOR_CAPABILITIES

IPMI_CMD_SET_FRU_LED_STATE

IPMI_CMD_GET_FRU_LED_STATE

IPMI_CMD_SET_IPMB_STATE

IPMI_CMD_SET_FRU_ACTIVATION_POLICY

IPMI_CMD_GET_FRU_ACTIVATION_POLICY

IPMI_CMD_SET_FRU_ACTIVATION

IPMI_CMD_GET_DEVICE_LOCATOR_RECORD_ID

IPMI_CMD_SET_PORT_STATE

IPMI_CMD_GET_PORT_STATE

IPMI_CMD_COMPUTE_POWER_PROPERTIES

IPMI_CMD_SET_POWER_LEVEL

IPMI_CMD_GET_POWER_LEVEL

IPMI_CMD_RENEGOTIATE_POWER

IPMI_CMD_GET_FAN_SPEED_PROPERTIES

IPMI_CMD_SET_FAN_LEVEL

IPMI_CMD_GET_FAN_LEVEL

IPMI_CMD_BUSED_RESOURCE_CONTROL

IPMI_CMD_GET_IPMB_LINK_INFO

Description:

PICMG netfn.

20.23 PICMG identifier

Syntax:

```
#define   IPMI_PICMG_IDENTIFIER   0x00
```

Members:

IPMI_PICMG_IDENTIFIER

Description:

PICMG identifier.

20.23.1 RadiSys CMM netfn

Syntax:

```
#define   IPMI_CMD_SET_TELCO_ALARM_STATE                0x00
#define   IPMI_CMD_GET_TELCO_ALARM_STATE                0x01
#define   IPMI_CMD_GET_TELCO_ALARM_INPUT_SENSOR_NUM     0x02
          BER
#define   IPMI_CMD_SET_TELCO_ALARM_SILENCE_TIMEOUT      0x03
#define   IPMI_CMD_SET_LISM_STATUS_LED                  0x04
#define   IPMI_CMD_SET_SECONDARY_IPMB_ADDRESS           0x05
#define   IPMI_CMD_GET_PHYSICAL_IPMB_LINK_STATISTIC     0x06
          S
#define   IPMI_CMD_SET_PHYSICAL_IPMB_LINK_STATE         0x07
#define   IPMI_CMD_GET_PHYSICAL_IPMB_LINK_STATE         0x08
#define   IPMI_CMD_SET_I2C_LINK_PULLUPS                 0x09
#define   IPMI_CMD_RUN_PHYSICAL_IPMB_LINK_DIAGNOSTI     0x0A
          CS
#define   IPMI_CMD_LOCAL_LINK_SEGMENT_ISOLATION_NOT     0x0B
          IFICATION
```

Members:

IPMI_CMD_SET_TELCO_ALARM_STATE

IPMI_CMD_GET_TELCO_ALARM_STATE

IPMI_CMD_GET_TELCO_ALARM_INPUT_SENSOR_NUMBER

IPMI_CMD_SET_TELCO_ALARM_SILENCE_TIMEOUT

IPMI_CMD_SET_LISM_STATUS_LED

IPMI_CMD_SET_SECONDARY_IPMB_ADDRESS

IPMI_CMD_GET_PHYSICAL_IPMB_LINK_STATISTICS

IPMI_CMD_SET_PHYSICAL_IPMB_LINK_STATE

IPMI_CMD_GET_PHYSICAL_IPMB_LINK_STATE

IPMI_CMD_SET_I2C_LINK_PULLUPS

IPMI_CMD_RUN_PHYSICAL_IPMB_LINK_DIAGNOSTICS

IPMI_CMD_LOCAL_LINK_SEGMENT_ISOLATION_NOTIFICATION

Description:

RadiSys CMM netfn.

20.23.2 RadiSys LISM netfn

Syntax:

```
#define    IPMI_CMD_LOCAL_IPMC_HS_TRANSITION_NOTIFICATION          0x0C
#define    IPMI_CMD_LOCAL_LISM_DEACTIVATION_CONFIRMATION           0x0D
#define    IPMI_CMD_GET_LOCAL_HOT_SWAP_STATE                       0x0E
#define    IPMI_CMD_GRACEFUL_SYSTEM_SHUTDOWN                       0x0F
#define    IPMI_CMD_IMAGE_STATUS_WORD_READ                         0x10
#define    IPMI_CMD_IMAGE_STATUS_WORD_WRITE                        0x11
#define    IPMI_CMD_GRACEFUL_SYSTEM_SHUTDOWN_ACK                   0x1C
#define    IPMI_CMD_RESET_SENSOR_READING                           0x1D
#define    IPMI_CMD_CLEAR_POST_SENSOR                              0x20
```

Members:

IPMI_CMD_LOCAL_IPMC_HS_TRANSITION_NOTIFICATION
IPMI_CMD_LOCAL_LISM_DEACTIVATION_CONFIRMATION
IPMI_CMD_GET_LOCAL_HOT_SWAP_STATE
IPMI_CMD_GRACEFUL_SYSTEM_SHUTDOWN
IPMI_CMD_IMAGE_STATUS_WORD_READ
IPMI_CMD_IMAGE_STATUS_WORD_WRITE
IPMI_CMD_GRACEFUL_SYSTEM_SHUTDOWN_ACK
IPMI_CMD_CLEAR_POST_SENSOR

Description:

RadiSys LISM netfn.

21 RPC program and function numbers

API defined in this document are exposed as RPC functions. Below RPC specific information for each function is provided. PRC program and function numbers defined below can be used to manage assess permissions using OEM Access Permissions API.

21.1 SHM API

This document defines SHM API version 1.

Table 21-1 SHM API RPC program and function numbers assignments

Function name	RPC Program Number	RPC Function Number
shmSessionOpen	1	101
shmSessionClose	1	102
shmOamLocationsList	1	103
shmOamActiveLocationsList	1	104
shmOamLocationTargetsList	1	105
shmOamLocationGet	1	106
shmOamSensorNumLookup	1	107
shmOamSensorNameLookup	1	108
shmOamSensorSdrGet	1	109
shmOamLogicalDeviceAddressGet	1	110
shmOamLogicalDeviceNameGet	1	111
shmOamLogicalDeviceAliasToNameConvert	1	112
shmOamLogicalDeviceNameToAliasConvert	1	113
shmOamLogicalDeviceIdGet	1	114
shmOamSensorFruIdLookup	1	115
shmOamSensorTypeLookup	1	116
shmOamLunsLookup	1	117

21.2 Common Upgrade API

This document defines Common Upgrade API version 1.

Table 21-2 Common Upgrade API RPC program and function numbers assignments

Function name	RPC program number	RPC function number
CUCreateDestHandle	4	1
CUDeleteDestHandle	4	2
CUGetImageHandle	4	3
CULoadImage	4	4
CUEnumerateImages	4	5
CUEnumerateImageProperties	4	6
CUGetImageProperties	4	7
CUSetImageProperty	4	8
CULockImages	4	9
CUUnlockImages	4	10
CUVerifyImages	4	11
CUImageRestart	4	12

21.3 Alarm Monitor API

This document defines Alarm Monitor API version 1.

Table 21-3 Alarm Monitor API RPC program and function assignments

Function name	RPC program number	RPC function number
shmOamAlarmMonitorHealthEventsAcknowledge	1	601
shmOamAlarmMonitorIpmcHealthEventsAcknowledge	1	611
shmOamAlarmMonitorFruHealthEventsAcknowledge	1	612
shmOamAlarmMonitorSensorHealthEventsAcknowledge	1	613
shmOamAlarmMonitorAllEventsNumberGet	1	602
shmOamAlarmMonitorIpmcEventsNumberGet	1	603
shmOamAlarmMonitorFruEventsNumberGet	1	604
shmOamAlarmMonitorSensorEventsNumberGet	1	605
shmOamAlarmMonitorAllEventsGet	1	606
shmOamAlarmMonitorIpmcEventsGet	1	607
shmOamAlarmMonitorFruEventsGet	1	608
shmOamAlarmMonitorSensorEventsGet	1	609
shmOamAlarmMonitorEventDescriptionGet	1	610
shmOamAlarmMonitorCompatibilityModeGet	1	614
shmOamAlarmMonitorCompatibilityModeSet	1	615

21.4 Telco Alarm Device Management API

This document defines Telco Alarm Device Management API version 1.

Table 21-4 Telco Alarm Device Management API RPC program and function numbers assignments

Function name	RPC program number	RPC function number
shmOamTadmTelcoAlarmDevicesGet	1	2001
shmOamTadmTelcoAlarmOutputSet	1	2002
shmOamTadmTelcoAlarmOutputGet	1	2003
shmOamTadmTelcoAlarmSilenceStateSet	1	2004
shmOamTadmTelcoAlarmSilenceStateGet	1	2005
shmOamTadmTelcoAlarmSilenceTimeoutSet	1	2006
shmOamTadmTelcoAlarmSilenceTimeoutGet	1	2007

21.5 SNMP Configuration Management API

This document defines SNMP Configuration Management API version 1.

Table 21-5 SNMP Configuration Management API RPC program and function numbers assignments

Function name	RPC program number	RPC function number
shmOamSnmpCfgFileSync	1	1901
shmOamSnmpLocalIpAddrMonitorEnable	1	1902
shmOamSnmpLocalIpAddrMonitorDisable	1	1903
shmOamSnmpLocalIpAddrMonitorStatus	1	1904

21.6 RMCP Configuration API

This document defines RMCP Configuration API version 1.

Table 21-6 RMCP Configuration API RPC program and function numbers assignments

Function name	RPC program number	RPC function number
shmOamRmcpSvrOemPermittedAdd	1	1701
shmOamRmcpSvrOemPermittedDel	1	1702
shmOamRmcpSvrOemPermittedGet	1	1703
shmOamRmcpSvrTransportProtocolSet	1	1704
shmOamRmcpSvrTransportProtocolGet	1	1705

21.7 Policy Management OAM API

This document defines Policy Management OAM API version 1.

Table 21-7 Policy Management OAM API RPC program and function numbers assignments

Function name	RPC program number	RPC function number
shmOamPolicyActionAdd	1	1601
shmOamPolicyActionRemove	1	1602
shmOamPolicyActionGet	1	1603
shmOamPolicyActionGetNext	1	1604
shmOamPolicyScriptSync	1	1605

21.8 TimeSync API

This document defines TimeSync API version 1.

Table 21-8 TimeSync API RPC program and function numbers assignments

Function name	RPC program number	RPC function number
shmOamTimeSyncNextServerIndexGet	1	2101
shmOamTimeSyncServerParamsGet	1	2102
shmOamTimeSyncServerAdd	1	2103
shmOamTimeSyncServerDelete	1	2104
shmOamTimeSyncNextListenAddressIndexGet	1	2105
shmOamTimeSyncListenAddressGet	1	2106
shmOamTimeSyncListenAddrAdd	1	2107
shmOamTimeSyncListenAddrDelete	1	2108
shmOamTimeSyncNextBroadcastAddressIndexGet	1	2109
shmOamTimeSyncBroadcastAddressGet	1	2110
shmOamTimeSyncBroadcastAddrAdd	1	2111
shmOamTimeSyncBroadcastAddrDelete	1	2112

21.9 IP Connectivity Manager API

This document defines IP Connectivity Manager API version 1.

Table 21-9 IP Connectivity Manager API RPC program and function numbers assignments

Function name	RPC program number	RPC function number
shmOamIpConnectivitySetEtherConfig	1	1101
shmOamIpConnectivityGetEtherConfig	1	1102
shmOamIpConnectivitySetEtherDir	1	1103
shmOamIpConnectivityGetEtherDir	1	1104
shmOamIpConnectivityEtherConfigApply	1	1105
shmOamIpConnectivitySetActiveIpConfig	1	1106

21.10 E-Keying Manager API

This document defines E-Keying Manager API version 1.

Table 21-10 E-Keying Manager API RPC program and function numbers assignments

Function name	RPC program number	RPC function number
shmOamEKeyManagerLinkStateGet	1	2601

21.11 Diagnostics API

This document defines Diagnostics API version 1.

Table 21-11 Diagnostics API RPC program and function numbers assignments

Function name	RPC program number	RPC function number
shmOamDiagnosticsTestFlash	1	701
shmOamDiagnosticsTestEthernet	1	702
shmOamDiagnosticsTestIpmb	1	703
shmOamDiagnosticsTestLeds	1	704
shmOamDiagnosticsSetRebootReason	1	705
shmOamDiagnosticsClearPOSTSensor	1	706

21.12 Statistics API

This document defines Statistics API version 1.

Table 21-12 Statistics API RPC program and function numbers assignments

Function name	RPC program number	RPC function number
shmOamStatisticsGetCounterGroups	1	708
shmOamStatisticsGetCounterNames	1	709
shmOamStatisticsGetCounterValue	1	710
shmOamStatisticsGetCounterThresholdTypes	1	711
shmOamStatisticsGetCounterThreshold	1	712
shmOamStatisticsSetCounterThreshold	1	713
shmOamStatisticsGetNextCounter	1	714

21.13 Event Manager OAM API

This document defines Event Manager OAM API version 1.

Table 21-13 Event Manager OAM API RPC program and function numbers assignments

Function name	RPC program number	RPC function number
shmOamEventManagerLogSizeGet	1	801
shmOamEventManagerLogCapacityGet	1	802
shmOamEventManagerLogCapacitySet	1	803
shmOamEventManagerLogArchiveInfoGet	1	804
shmOamEventManagerLogArchiveInfoSet	1	805
shmOamEventManagerLogArchive	1	806
shmOamEventManagerLogEntryFindByDate	1	807
shmOamEventManagerLogEntryFindBySeverity	1	808
shmOamEventManagerLogCapacityRangeGet	1	809

21.14 Local Sensor API

This document defines Local Sensor API version 1.

Table 21-14 Local Sensor API RPC program and function numbers assignments

Function name	RPC program number	RPC function number
shmLsrEventOnlySensorCreate	1	2201
shmLsrEventOnlySensorCreateIfDoesNotExist	1	2206
shmLsrEventOnlySensorDelete	1	2202
shmLsrEventOnlySensorGroupCreate	1	2203
shmLsrEventOnlySensorGroupDelete	1	2204
shmLsrEventOnlySensorSendEvent	1	2205

RPC program and function numbers

21.15 OEM Access Permissions API

This document defines OEM Access Permissions API version 1.

Table 21-15 OEM Access Permissions API RPC program and function numbers assignments

Function name	RPC program number	RPC function number
shmOamSecurityOemPermissionGet	1	1801
shmOamSecurityOemPermissionGetNext	1	1802
shmOamSecurityOemPermissionSet	1	1803
shmOamSecurityOemPermissionReset	1	1804
shmOamSecurityOemPermissionDefaultGet	1	1805
shmOamSecurityOemPermissionDefaultSet	1	1806

21.16 User Management OAM API

This document defines User Management OAM API version 1.

Table 21-16 User Management OAM API RPC program and function numbers assignments

Function name	RPC program number	RPC function number
shmOamSecurityUserDelete	1	1807

21.17 Process Monitoring OAM API

This document defines Process Monitoring OAM API version 1.

Table 21-17 Process Monitoring OAM API RPC program and function numbers assignments

Function name	RPC program number	RPC function number
shmOamPmAdminStateSet	1	1501
shmOamPmAdminStateGet	1	1502
shmOamPmProcessGet	1	1503
shmOamPmProcessIdArrayGet	1	1504

21.18 High Availability API

This document defines High Availability API version 1.

Table 21-18 High Availability API RPC program and function number assignments

Function name	RPC program number	RPC function number
shmOamHaStateReadinessGet	1	901
shmOamHaStateHaGet	1	902
shmOamHaOutOfServiceCauseGet	1	903
shmOamHaSwitchoverAutomaticModeGet	1	904
shmOamHaSwitchoverManual	1	905
shmOamHaSwitchoverAutomatic	1	906
shmOamHaOutOfService	1	907
shmOamHaInService	1	908
shmOamHaPeerOutOfService	1	909
shmOamHaPeerInService	1	910
shmOamHaPeerForcedExit	1	911
shmOamHaLegacyRedundancyStateGet	1	912
shmOamHaPeerSwitchoverManual	1	913
shmOamHaLegacyFailover	1	914
shmOamHaLegacyStandbyReboot	1	915
shmOamHaHealthScoreGet	1	916
shmOamHaCmmReadyTimeoutGet	1	917
shmOamHaCmmReadyTimeoutSet	1	918