To: T10 Technical Committee
From: Steven Fairchild, HP (steve.Fairchild@hp.com)
Date: 3 September 2004
Subject: Common Storage Management Interface for SAS Proposal

The purpose of this proposal is to present a storage management interface for the Serial Attached SCSI (SAS) infrastructure.

This document was originally proposed to T11.5 for inclusion in the SM-HBA API efforts. The T11.5 committee recommended that the scope of the proposal might be addressed more effectively under the direction of T10.

The emerging SAS standard provides an opportunity to standardize the software interfaces involved in the management of SAS HBAs and infrastructure.

Outlined in this proposal is a Common Storage Management Interface (CSMI) that defines a set of functionality that is deployed as part of an OS Specific Interface (i.e. OS driver).

The CSMI functionality defined in this proposal includes;

- Descriptive information about the OS driver, version and capabilities
- Descriptive information about the HBA configuration, hardware addressing, ROM and BIOS support
- HBA operational status
- HBA Firmware download support
- Descriptive information about HBA based RAID support
- Descriptive information about HBA based RAID configurations
- Descriptive information about the SAS/SATA phys available on the HBA
- Controlling functions for the SAS/SATA phys available on the HBA, link rate range, signal level, enable/disable options
- Information on any available link error counters for each phy
- Passthru support for commands to devices that support each of the SAS protocols, SMP, SSP, STP/SATA or SATA devices emulated as SCSI devices
- Descriptive information to support SATA devices directly attached to HBAs
- Address conversion functions to convert from SAS addressing to OS addressing or from OS addressing to SAS addressing
- Task management support (for SSP devices)
- Controlling functions for signal level characterization of phys on the HBA
- Descriptive information for connectors on the HBA

Each of the functions is defined in detail as behaviors available as IOCTL functions provided by an OS specific driver.

Since the OS driver interface varies from OS to OS, no attempt is made to use a single calling methodology, instead the basic premise of the CSMI functionality is that the

IOCTL payload is common across all OS driver interfaces. This allows applications to maximize code reuse across supported OS'es.

The CSMI document that follows, provides OS specific details, potential implementations and support for three OS'es; Windows, Linux and Netware. The intention is not to limit to these three OS'es, but to use them as templates for defining support for future OS'es.

Since in many ways, the SATA infrastructure can be viewed as a subset of SAS solution, the CSMI document presented here also incorporates SATA HBAs as part of the solution. That is the reason it is presented as a common management proposal.

The recommendation of this proposal and the accompanying CSMI document is to recommend that T10 establish a working group to define a low level interface standard that could be used by upper level standards such as the SM-HBA API being defined in the T11.5 committee.

Table of Contents

1 Introduction

This document is intended to define a Common Storage Management Interface (CSMI) composed of a set of control codes, definitions, data structures and return codes that a Windows, Linux or Netware driver shall implement to provide a standard mechanism for accessing the physical components within a Serial Attached SCSI or Serial ATA domain.

The CSMI described is applicable to installations supporting Microsoft Windows platforms, Linux platforms and NetWare platforms.

The interface allows management and test utilities to access the physical components within a Serial Attached SCSI or Serial ATA domain.

2 Platform Requirements

All OS platforms implementing this specification shall adhere to the following:

- An accessible device node shall be available for the controller, even if no physical devices are registered with the SCSI Subsystem.
- Shall allow sending commands to any storage / enclosure device connected to the controller.

Additional requirements specific to the OS platform may also be necessary.

2.1 Windows

- Shall support Microsoft Windows 2000, XP, and 2003 family platforms.
- Shall support the Windows driver registry entry, **MaximumSGList**, with a value of 0xFF.
- The SATA and SAS pass-through IOCTLS shall all support buffered IO to the limit defined by **MaximumSGList**.

2.2 Linux

- Shall support 2.4.x, 2.5.x, and 2.6.x Linux kernels.
- Shall be made open source and the source made available to software developers.
- Shall be made available as patches, RPM and driver diskettes.
- Shall provide driver version information via modinfo call.
- Controller firmware shall be EDD3.0 complaint.

2.3 NetWare

- Shall NetWare versions 6.5 and greater

3 Submitting Control Codes

The CSMI control codes and submission mechanism is dependent on the OS platform, but is typically based on device I/O controls, or IOCTLs. The definitions, data structures, return codes and functional behavior are intended to be independent of the OS platform.

3.1 Windows

On the Windows platform, CSMI is defined as a set of control codes that are submitted using the ***DeviceIoControl*** function call with the **IOCTL_SCSI_MINIPORT** I/O control code. The specific CSMI functional behavior is executed by providing the associated CSMI control code and data structure within the **SRB_IO_CONTROL** structure passed as the input buffer.

To maintain cross platform compatibility and reusability, the **SRB_IO_CONTROL** has an alias of IOCTL_HEADER in the CSMI data structures. The **SRB_IO_CONTROL** uses the standard Windows data types for its members. For reference, the Windows data types in the **SRB_IO_CONTROL** correspond to the following CSMI data types:

- __u8 == **UCHAR**
- **__i8 == CHAR**
- **__u16 == USHORT**
- __u32 == **ULONG**

If a CSMI control code is not supported then the ***DeviceIoControl*** will return a one (1) indicating success and the IoctlHeader.ReturnCode will contain **CSMI_SAS_STATUS_BAD_CNTL_CODE.**

If the CSMI buffer provided is too small, then the ***DeviceIoControl*** will return a one (1) indicating success and the IoctlHeader.ReturnCode will contain **CSMI_SAS_STATUS_INVALID_PARAMETER.**

3.1.1 Example

```
// this example does not include the definitions of all the variables used
// begin by creating the device name to open, the portNumber referenced is
// the identifier for the miniport driver

sprintf(deviceName,
        "\\.\Scsi%d:",
        portNumber);

// get a handle to the miniport driver
handle = CreateFile(deviceName,
                    GENERIC_READ | GENERIC_WRITE,
                    FILE_SHARE_READ | FILE_SHARE_WRITE,
                    NULL,
                    OPEN_EXISTING,
                    NULL,
                    NULL);

// use handle to issue the desired IOCTL (SAS phy info in this example)
info = (CSMI_SAS_PHY_INFO_BUFFER *)
        calloc(1,
               sizeof(CSMI_SAS_PHY_INFO_BUFFER));

info->IoctlHeader.HeaderLength = sizeof(IOCTL_HEADER);
info->IoctlHeader.Timeout = CSMI_SAS_TIMEOUT;
info->IoctlHeader.ControlCode = CC_CSMI_SAS_GET_PHY_INFO;
info->IoctlHeader.Length = sizeof(CSMI_SAS_PHY_INFO_BUFFER) -
                           sizeof(IOCTL_HEADER);

memcpy(&info->IoctlHeader.Signature,
       CSMI_SAS_SIGNATURE,
       sizeof(CSMI_SAS_SIGNATURE));

success = DeviceIoControl(handle,
                          IOCTL_SCSI_MINIPORT,
                          info,
                          sizeof(CSMI_SAS_PHY_INFO_BUFFER),
                          info,
                          sizeof(CSMI_SAS_PHY_INFO_BUFFER),
                          &bytesReturned,
                          NULL);

// process the result of the IOCTL using; success, GetLastError(), info->ReturnCode,
// bytesReturned and the IOCTL content.
```

3.2 Linux

On the Linux platform, CSMI is defined as a set of I/O control codes that are submitted using the ***ioctl*** function call. The specific CSMI functional behavior is executed by providing the associated CSMI data structure passed as part of the input buffer. The input buffer also contains a header data structure. See IOCTL_HEADER.

3.2.1 Example

```
// this example does not include the definitions of all the variables used
// open the device node for the HBA, eg. hba, to obtain a file descriptor.
fd = open("/dev/hba",
          O_RDWR);

// use the file descriptor to issue the desired IOCTL (SAS phy info in this example)
info = (CSMI_SAS_PHY_INFO_BUFFER *)
       calloc(1,
              sizeof(CSMI_SAS_PHY_INFO_BUFFER));

info->IoctlHeader.Timeout = CSMI_SAS_TIMEOUT;

success = ioctl(fd,
                CC_CSMI_SAS_GET_PHY_INFO,
                 info);

// process the result of the IOCTL using; success, errno, ReturnCode and the IOCTL content.
```

3.3 NetWare

On the NetWare platform, CSMI is defined as a set of I/O control codes that are submitted using the NPA_HACB_Passthru( ) API.

3.3.1 Example

The follow source code example illustrates a generic function to issue all CSMI IOCTL commands within a NetWare Loadable Module (NLM.)

```
// this example does not include the definitions of all the variables used

LONG CSMI_API(LONG AdapObjId,      // Media Manager adapter ID for target HBA
              LONG DevId,          // Media Manager device ID (-1 if HBA)
              LONG lCSMIcmd,       // CSMI IOCLT command
              void* pvBuff,        // pointer to CSMI cmd specific structure
              LONG lBuffSize)      // size of CSMI cmd data structure buffer
{
  LONG        cCode;
  LONG        lDeviceHandle = 0;
  HACB        hacbNW;


   // Clear HACB structure.
   memset((void *)&hacbNW, 0, sizeof( HACB ) );

   // Get the device handle if Device Object provided
   if (DevId != -1)
   {
      cCode =  NPA_Return_DeviceHandle(DevId, &lDeviceHandle);
      if (MM_OK != cCode)
         return (cCode);
     hacbNW.deviceHandle = lDeviceHandle;
   }

   // Setup HACB
   hacbNW.Command.Host.functio = lCSMIcmd;
   hacbNW.controlInfo = NO_FREEZE_QUEUE | PRIORITY_HACB | DATA_DIRECTION_OUT;
   hacbNW.HACBType = 0;
   hacbNW.timeoutAmount=15;
   hacbNW.vDataBufferPtr = (void *) &pvBuff;
   hacbNW.dataBufferLength = (lBuffSize + ALIGN_FACTOR) & ~ALIGN_FACTOR ;

   hacbNW.vErrorSenseBufferPtr = (void *) &pvBuff;
   hacbNW.errorSenseBufferLength = (lBuffSize + ALIGN_FACTOR) & ~ALIGN_FACTOR;

   // issue IOCTL
   cCode = NPA_HACB_Passthru( AdapObjId, (VU_HACB*)&hacbNW );

   // return HACB return codes to sender
   return( cCode | hacb.HACBCompletion);
}
```

4 Buffer Header

The CSMI data structure that defines the CSMI functional behavior uses a platform specific header structure. To allow a shared 'C' style header file to define CSMI, the header structure is named **IOCTL_HEADER.** While the name is the same across OS platforms, the actual content of the **IOCTL_HEADER** data structure is unique to the OS platform. The application must be aware of the OS platform in order to properly access the elements of the **IOCTL_HEADER** structure.

4.1 Windows

The CSMI **IOCTL_HEADER** is an alias for the SRB_IO_CONTROL data structure on the Windows platforms. Refer to the Microsoft Windows Driver Development Kit for more information. The information provided here is for convenience only.

4.1.1 Input

The SRB_IO_CONTROL data structure with the following initialized members;

- HeaderLength. Length of the header structure. Must be sizeof(**IOCTL_HEADER**).
- Signature. Namespace signature, dependent on the CSMI control code used. See Security and Enabling Features.
- Timeout. Time (in seconds) to wait before the CSMI functional behavior is considered to have failed. See Timeouts.
- ControlCode. Indicates which CSMI functional behavior to execute. See Control Codes.
- ReturnCode. Initialized to 0.
- Length. Length of the CSMI data structure buffer excluding **IOCTL_HEADER**. At a minimum this should be the (sizeof(**CSMI_SAS_xxxx_xxxx_BUFFER**) – sizeof(**SRB_IO_CONTROL**)) associated with the CSMI control code. A larger buffer may be supplied.

4.1.2 Output

A SRB_IO_CONTROL data structure with the returned data;

- HeaderLength. Same as input.
- Signature. Same as input.
- Timeout. Same as input.
- ControlCode. Same as input.
- ReturnCode. Indicates the resulting status of the CSMI functional behavior. See Return Codes.
- Length. Same as input.

4.1.3 Structure Definitions

4.1.3.1 SRB_IO_CONTROL

```
typedef struct _SRB_IO_CONTROL {
        ULONG HeaderLength;
        UCHAR Signature[8];
        ULONG Timeout;
        ULONG ControlCode;
        ULONG ReturnCode;
        ULONG Length;
} SRB_IO_CONTROL,
  *PSRB_IO_CONTROL;
```

4.2 Linux

The **IOCTL_HEADER** is a reference to the typedef of the struct *_IOCTL_HEADER* on the Linux platform.

4.2.1 Input

The IOCTL_HEADER data structure with the following initialized members;

- IOControllerNumber. The I/O controller number for drivers that support multiple I/O controllers (adapters).
- Length. Length of the CSMI data structure buffer including **IOCTL_HEADER**. At a minimum this should be the sizeof(**CSMI_SAS_xxxx_xxxx_BUFFER**) associated with the CSMI control code. A larger buffer may be supplied.
- ReturnCode. Initialized to 0.
- Timeout. Time (in seconds) to wait before the CSMI functional behavior is considered to have failed. See Timeouts.
- Direction. Indicates the direction of data flow as part of the CSMI functional behavior. A **CSMI_SAS_DATA_READ** is used to indicate that data will be returned as part of the CSMI functional behavior. A **CSMI_SAS_DATA_WRITE** is used to indicate that data will be provided as part of the CSMI functional behavior.

4.2.2 Output

The IOCTL_HEADER data structure with the following initialized members;

- IOControllerNumber. Same as input.
- Length. Same as input.
- ReturnCode. Indicates the resulting status of the CSMI functional behavior. See Return Codes.
- Timeout. Same as input.
- Direction. Same as input.

4.2.3 Structure Definitions

4.2.3.1 IOCTL_HEADER

```
typedef struct _IOCTL_HEADER {
  __u32 IOControllerNumber;
  __u32 Length;
  __u32 ReturnCode;
  __u32 Timeout;
  __u16 Direction;
} IOCTL_HEADER,
  *PIOCTL_HEADER;
```

4.3 NetWare

The **IOCTL_HEADER** is a reference to the typedef of the struct *_IOCTL_HEADER* on the NetWare platform. Definition of this data structure is provided below. Unlike the Windows, or Linux versions, this datat structure will be minimual since most IOCTL details are already contained within the NetWare Peripheral Architecture (NWPA) Host bus Adapter Control Block (HACB) structure. Please Refer to Novell's Developer's Kit (NDK) for complete information on NWPA and HACB definitions. The information provided here is for convenience only.

4.3.1 HACB Usage

Application NLMs rely on the use of Novell's Media Manager (MM) for discovery of drivers, adapters and devices. As well, driver pass through calls will be used where necessary to identify and acquire hardware-device specific information. Where passthroughs and MM calls cannot deliver required information, a vendor unique set of Host Bus Adapter Control Block (HACB) IOCtrls (CSMI) are defined.

The NWPA specification allows for a number of methods for implementing vendor unique HACB calls. One method is with the HACB field *hacbType* set to the Novell assigned Adapter Module ID of the HAM receiving such HACB. A second method is with the HACB field *hacbType* set to zero (0). In both of these cases the command union area of the HACB will allow for Vendor Unique functions definitions. The method defined here in will be for the hacbType set to zero.

The HACB contains a command union area of 28 bytes to define each vendor unique IOCTL. For convenience to the reader, the following data structure defines the required HACB command block union that shall be utilized with HP's CSMI:

```
HACB Command Block Union Area
union {
   struct /*HACB Type = 0: Host Adapter Cmd */
   {
     LONG   function;
     LONG   parameter0;
     LONG   parameter1;
     LONG   parameter2;
     BYTE   reserved[12];
   }host;
```

With the first 4 bytes (function) used for IOCTL definition, there remains only 24 bytes of space to define additional parameters. These 24 bytes are insufficient to allow for a common usage of the HP CSMI IOCTLs across supported operating system platforms. The following will allow for the usage of the CSMI IOCTLs within the confines of NetWare's Peripheral Architecture.

All NetWare CSMI IOCTLs will be issued with the data direction bit set to WRITE within the `controlInfo` field, (e.g. 0x00000002.) The CSMI IOCTL data structure buffer will always be sent to the driver in the the HACB *vDataBufferPtr. Upon IOCTL return (to a HACB WRITE) the driver will send the CSMI IOCTL data structure buffer back to the calling application using the HACB *vErrorSenseBufferPtr..

Note:
By design the NWPA HACB process is uni-directional; thus when a READ IOCTL is issued, the memory referenced by the HACB pdataBufferPointer is only to be used for READING data from the driver, not for transporting data to the driver. And when a WRITE IOCTL is issued, the HACB databuffer pointer is only to be used for sending information

to the driver, not for reading data from the driver. However, the vErrorSenseBufferPtr is always available as a data transport by the driver for both READ and WRITE operations.

4.3.2 Input

For each CSMI command, the following HACB data structure members will be set as follows:

HACB field	Desription / Required Values
devicehandle	NWPA supplied handle for a specific registered device. Obtained via NPA_Return_DeviceHandle( )
hacbType	Time (in seconds) to wait before the CSMI functional behavior is considered to have failed. See Timeouts.
controlInfo	HACB control flags. Must include 0x00000002 (WRITE)
dataBufferLen	WORD aligned length of the CSMI command data structure buffer
vDataBufferPtr	Virtual address to the CSMI command data structure buffer. The data structure is CSMI IOCTL command dependent.
errorSenseBufferLen	Same as dataBufferLen
vErrorSenseBufferPtr	Same as vDataBufferPtr
command.host.function	Defines the CSMI IOCTL command

4.3.3 Output

For each CSMI command, the driver will return information within the following HACB data structure fields:

HACB field	Desription / Required Values
hacbCompletion	Per NWPA specifications, the return status of this HACB.
errorSenseBufferLen	WORD aligned length of CSMI command data structure buffer.
vErrorSenseBufferPtr	Virtual address to the CSMI command data structure buffer.

4.3.4 Structure Definitions

4.3.4.1 IOCTL_HEADER

```
typedef struct _IOCTL_HEADER {
  long          lLength;          // size CSMI IOCTL specific command data structure
  unsigned long ulReturnCode;     // CSMI return code
  } IOCTL_HEADER
```

5 Security and Enabling Features

Since the CSMI control codes allow a management application to access the underlying physical layers of a storage solution, an important aspect of the implementation is to protect against unauthorized access. The security and enabling features are OS platform specific. The basis of any implementation is to prevent a user application from using the CSMI control codes to access the CSMI functional behaviors without restricting authorized system applications. The implementation would ideally provide the following levels of access to the CSMI functional behaviors:

- **No access**, CSMI functional behaviors would not be accessible by any user or system application.
- **Restricted access,** CSMI read only functional behaviors would be accessible by a system application. It is important to note that Restricted access could also expose the CSMI read only functional behaviors to user applications on some OS platforms.
- **Limited access**, CSMI read only functional behaviors and firmware download functional behaviors would be accessible by a system application. It is important to note that Limited access could also expose the CSMI read only functional behaviors and firmware download functional behaviors to user applications on some OS platforms.
- **Full access;** CSMI read and write functional behaviors would be accessible by a system application. It is important to note that Full access could also expose the CSMI read and write functional behaviors to user applications on some OS platforms.

The CSMI control codes may also need an enabling namespace signature to ensure that the code definitions do not collide with other standard control codes or vendor unique codes. The use of the signature is platform specific.

5.1 Windows

Since the IOCTL_SCSI_MINIPORT I/O control is not adequately protected on all Windows platforms, the driver implementing CSMI control codes will use the DriverParameter registry value of the miniport driver registry definition to identify which CSMI functional behaviors are allowed. The CSMI security registry value will be delineated from existing DriverParameter regristry values by using a semicolon (‘;”). For example if the DriverParameter value already contains “Something=here”, then after adding the CSMI security, the DriverParameter value will contain “Something=here;CSMI=Full;”. Refer to the Microsoft Driver Development Kit for more information on the miniport registry usage. If registry value content shall be identified as valid only if the CSMI descriptor matches exactly the ASCII string shown in Table 1.

Table 1: Windows Registry, Security Access

Security Access	DriverParameter Registry Value
None	“CSMI=None;” ASCII string
Restricted	“CSMI=Restricted;” ASCII string
Limited	No CSMI ASCII string or “CSMI=Limited;” ASCII string
Full	“CSMI=Full;” ASCII string

The CSMI control codes, enabling namespace signature and associated security access are listed in Table 2.

Table 2: Windows Namespace Signatures

CSMI Control Code	Enabling Namespace Signature	Security Access
CC_CSMI_SAS_GET_DRIVER_INFO	CSMI_ALL_SIGNATURE	Limited
CC_CSMI_SAS_GET_CNTLR_CONFIG	CSMI_ALL_SIGNATURE	Limited
CC_CSMI_SAS_GET_CNTLR_STATUS	CSMI_ALL_SIGNATURE	Limited
CC_CSMI_SAS_FIRMWARE_DOWNLOAD	CSMI_ALL_SIGNATURE	Limited
CC_CSMI_SAS_GET_RAID_INFO	CSMI_RAID_SIGNATURE	Limited
CC_CSMI_SAS_GET_RAID_CONFIG	CSMI_RAID_SIGNATURE	Limited
CC_CSMI_SAS_GET_PHY_INFO	CSMI_SAS_SIGNATURE	Limited
CC_CSMI_SAS_SET_PHY_INFO	CSMI_SAS_SIGNATURE	Full
CC_CSMI_SAS_GET_LINK_ERRORS	CSMI_SAS_SIGNATURE	Limited
CC_CSMI_SAS_SMP_PASSTHROUGH	CSMI_SAS_SIGNATURE	Full
CC_CSMI_SAS_SSP_PASSTHROUGH	CSMI_SAS_SIGNATURE	Full
CC_CSMI_SAS_STP_PASSTHROUGH	CSMI_SAS_SIGNATURE	Full
CC_CSMI_SAS_GET_SATA_SIGNATURE	CSMI_SAS_SIGNATURE	Limited
CC_CSMI_SAS_GET_SCSI_ADDRESS	CSMI_SAS_SIGNATURE	Limited
CC_CSMI_SAS_GET_DEVICE_ADDRESS	CSMI_SAS_SIGNATURE	Limited
CC_CSMI_SAS_TASK_MANAGEMENT	CSMI_SAS_SIGNATURE	Full
CC_CSMI_SAS_GET_CONNECTOR_INFO	CSMI_SAS_SIGNATURE	Limited
CC_CSMI_SAS_PHY_CONTROL	CSMI_PHY_SIGNATURE	Full

5.1.1 Windows Port Driver

For Windows port drivers that implement CSMI an alternate form of the registry security key may be used. A port driver may use the Parameters key of CSMI with a numeric value to set the security level. See Table 3

Table 3: Windows Registry, Port Driver Security Access

Security Access	HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\ PortDriverName\Parameters\CSMI = dword value
None	0 dword value
Restricted	1 dword value
Limited	No CSMI key or 2 dword value
Full	3 dword value

5.2 Linux

Since the CSMI functional behaviors may only be issued by an application with root security access, no specific protection mechanisms are required on the Linux platforms. There is also not

a provision for the namespace signature, because the CSMI control codes on the Linux platform should prevent a namespace collision.

5.3 NetWare

Since the CSMI functional behaviors may only be issued by an application with administrative security access, no specific protection mechanisms are required on the NetWare platforms.

6 Control Codes

The CSMI control code may be provided as an element of a buffer structure that is submitted as part of the device I/O control or as an argument to the device I/O control call. The exact method is platform specific (see Submitting CSMI Control Codes). In either case, a CSMI data structure buffer is provided as the content of the device I/O control call. The CSMI data structure buffer has the general form of;

```
typedef struct _CSMI_SAS_xxxx_xxxx_BUFFER {
        IOCTL_HEADER IoctlHeader;
        …
} CSMI_SAS_xxxx_xxxx_BUFFER, *PCSMI_SAS_xxxx_xxxx_BUFFER;
```

The CSMI data structure buffer provides as input, the necessary information to specify the CSMI functional behavior desired. It also provides space for any resulting data requested by the CSMI functional behavior. The application using the CSMI control codes must ensure that enough memory has been allocated to contain any requested data. If the memory provided is too small a CSMI error will be returned from the device I/O control call. See Submitting Control Codes.

6.1 CC_CSMI_SAS_GET_DRIVER_INFO
(SATA, SAS)

6.1.1 Behavior
This CSMI functional behavior requests descriptive and version information for the driver associated with a storage controller. The information returned shall be consistent with any file information provided on the platform OS for the driver. Any driver that implements this specification must support this behavior.

6.1.2 Input
A CSMI_SAS_DRIVER_INFO_BUFFER data structure with the following initialized members;
- IoctlHeader. See IOCTL_HEADER for the platform specific initialization.
- Information. Initialized to 0's.

6.1.3 Output
A CSMI_SAS_DRIVER_INFO_BUFFER data structure with the returned data;
- IoctlHeader.ReturnCode. See Return Codes.
- Information.szName. Name of the binary driver. May contain a string of up to 80 ASCII characters and a null termination. Shall reference the base name of the driver, without a file extension.
- Information.szDescription. Description of the driver. May contain a string of up to 80 ASCII characters and a null termination. Shall reference the vendor, product family and model information.
- Information.usMajorRevision. Major revision of the driver.
- Information.usMinorRevision. Minor revision of the driver.
- Information.usBuildRevision. Build revision of the driver.
- Information.usReleaseRevision. Release revision of the driver.
- Information.usCSMIMajorRevision. Revision of the CSMI specification that the driver supports. The driver shall return the constant **CSMI_MAJOR_REVISION**.
- Information.usCSMIMinorRevision. Revision of the CSMI specification that the driver supports. The driver shall return the constant **CSMI_MINOR_REVISION**.

6.1.4 Structure Definitions

6.1.4.1 CSMI_SAS_DRIVER_INFO

```
typedef struct _CSMI_SAS_DRIVER_INFO {
  __u8  szName[81];
  __u8  szDescription[81];
  __u16 usMajorRevision;
  __u16 usMinorRevision;
  __u16 usBuildRevision;
  __u16 usReleaseRevision;
  __u16 usCSMIMajorRevision;
  __u16 usCSMIMinorRevision;
 } CSMI_SAS_DRIVER_INFO,
 *PCSMI_SAS_DRIVER_INFO;
```

6.1.4.2 CSMI_SAS_DRIVER_INFO_BUFFER

```
typedef struct _CSMI_SAS_DRIVER_INFO_BUFFER {
  IOCTL_HEADER IoctlHeader;
  CSMI_SAS_DRIVER_INFO Information;
} CSMI_SAS_DRIVER_INFO_BUFFER,
  *PCSMI_SAS_DRIVER_INFO_BUFFER;
```

6.2 CC_CSMI_SAS_GET_CNTLR_CONFIG
(SATA, SAS)

6.2.1 Behavior
This CSMI functional behavior requests descriptive and version information for the hardware, firmware and boot BIOS associated with a storage controller. Any driver that implements this specification must support this behavior.

6.2.2 Input
A CSMI_SAS_CNTLR_CONFIG_BUFFER data structure with the following initialized members;
- IoctlHeader, see IOCTL_HEADER for the platform specific initialization.
- Configuration, initialized to 0's.

6.2.3 Output
A CSMI_SAS_CNTLR_CONFIG_BUFFER data structure with the returned data;
- IoctlHeader.ReturnCode. See Return Codes.
- Configuration.ulBaseIoAddress, Base I/O Address of the controller. If the controller has more than one base I/O address, this field will specify the lowest one used.
- Configuration.BaseMemoryAddress, Base memory address of the controller. If the controller has more than one base memory address, this field will specify the lowest one used.
- Configuration.ulBoardID, 32-bit subsystem ID from the controller's PCI configuration space. Bits 0 – 15 contain the subsystem vendor ID and bits 16 – 31 contain the subsystem ID as defined by the PCI specification.
- Configuration.usSlotNumber, The physical slot number of the controller in the system. If the driver cannot determine the physical slot number, the constant **SLOT_NUMBER_UNKNOWN** will be returned.
- Configuration.bControllerClass, Specify the class of the controller. For SAS and SATA controllers implementing this specification, it is the constant **CSMI_SAS_CNTLR_CLASS_HBA**.
- Configuration.bIoBusType. System I/O bus type of the controller. Shall be one of the following;
  - **CSMI_SAS_BUS_TYPE_PCI**, if the host bus adapter is in a PCI slot
  - **CSMI_SAS_BUS_TYPE_PCMCIA**, if the host bus adapter is in a PCMCIA slot
- Configuration.BusAddress. The I/O bus address of the controller, if applicable.
- Configuration.szSerialNumber. Controller serial number. May contain a string of up to 80 ASCII characters and a null termination. Shall reference the serial number of the controller. If the serial number cannot be determined, then the field shall be 0 filled.
- Configuration.usMajorRevision. Major revision of the controller firmware. If the controller firmware cannot be determined, then the field shall return 0.
- Configuration.usMinorRevision. Minor revision of the controller firmware. If the controller firmware cannot be determined, then the field shall return 0.
- Configuration.usBuildRevision. Build revision of the controller firmware. If the controller firmware cannot be determined, then the field shall return 0.
- Configuration.usReleaseRevision. Release revision of the controller firmware. If the controller firmware cannot be determined, then the field shall return 0.
- Configuration.usBIOSMajorRevision. Major revision of the controller boot BIOS. If the controller boot BIOS cannot be determined, then the field shall return 0.
- Configuration.usBIOSMinorRevision. Minor revision of the controller boot BIOS. If the controller boot BIOS cannot be determined, then the field shall return 0.

- Configuration.usBIOSBuildRevision. Build revision of the controller boot BIOS. If the controller boot BIOS cannot be determined, then the field shall return 0.
- Configuration.usBIOSReleaseRevision. Release revision of the controller boot BIOS. If the controller boot BIOS cannot be determined, then the field shall return 0.
- uControllerFlags. Controller sub class definition. One or more of the following constants may be used;
  - **CSMI_SAS_CNTLR_SAS_HBA,** controller is a SAS HBA
  - **CSMI_SAS_CNTLR_SAS_RAID,** controller is a SAS HBA with RAID support
  - **CSMI_SAS_CNTLR_SATA_HBA,** controller is a SATA HBA
  - **CSMI_SAS_CNTLR_SATA_RAID,** controller is a SATA HBA with RAID support
  - **CSMI_SAS_CNTLR_FWD_SUPPORT**, controller supports firmware download CSMI control code; **CC_CSMI_SAS_FIRMWARE_DOWNLOAD**.
  - **CSMI_SAS_CNTLR_FWD_ONLINE**, controller supports online update of firmware.
  - **CSMI_SAS_CNTLR_FWD_SRESET**, controller requires soft reset to initiate a firmware update. The driver will manage coordinating the download to ensure outstanding IOs are not impacted.
  - **CSMI_SAS_CNTLR_FWD_HRESET**, controller requires a hard reset to initiate a firmware update. The driver will force the controller to a power-up state and re-initialize the controller as necessary.
  - **CSMI_SAS_CNTLR_FWD_RROM**, controller supports a redundant copy of the ROM image.
- Configuration.usRromMajorRevision. Major revision of the redundant controller firmware. If the redundant controller firmware cannot be determined, then the field shall return 0.
- Configuration.usRromMinorRevision. Minor revision of the redundant controller firmware. If the redundant controller firmware cannot be determined, then the field shall return 0.
- Configuration.usRromBuildRevision. Build revision of the redundant controller firmware. If the redundant controller firmware cannot be determined, then the field shall return 0.
- Configuration.usRromReleaseRevision. Release revision of the redundant controller firmware. If the redundant controller firmware cannot be determined, then the field shall return 0.
- Configuration.usRromBIOSMajorRevision. Major revision of the redundant controller boot BIOS. If the redundant controller boot BIOS cannot be determined, then the field shall return 0.
- Configuration.usRromBIOSMinorRevision. Minor revision of the redundant controller boot BIOS. If the redundant controller boot BIOS cannot be determined, then the field shall return 0.
- Configuration.usRromBIOSBuildRevision. Build revision of the redundant controller boot BIOS. If the redundant controller boot BIOS cannot be determined, then the field shall return 0.
- Configuration.usRromBIOSReleaseRevision. Release revision of the redundant controller boot BIOS. If the redundant controller boot BIOS cannot be determined, then the field shall return 0.

6.2.4 Structure Definitions

6.2.4.1 CSMI_SAS_PCI_BUS_ADDRESS

```
typedef struct _CSMI_SAS_PCI_BUS_ADDRESS {
  __u8  bBusNumber;
  __u8  bDeviceNumber;
  __u8  bFunctionNumber;
  __u8  bReserved;
} CSMI_SAS_PCI_BUS_ADDRESS,
  *PCSMI_SAS_PCI_BUS_ADDRESS;
```

6.2.4.2 CSMI_SAS_IO_BUS_ADDRESS

```
typedef union _CSMI_SAS_IO_BUS_ADDRESS {
  CSMI_SAS_PCI_BUS_ADDRESS PciAddress;
  __u8  bReserved[32];
} CSMI_SAS_IO_BUS_ADDRESS,
  *PCSMI_SAS_IO_BUS_ADDRESS;
```

6.2.4.3 CSMI_SAS_CNTLR_CONFIG

```
typedef struct _CSMI_SAS_CNTLR_CONFIG {
  __u32 uBaseIoAddress;
  struct {
    __u32 uLowPart;
    __u32 uHighPart;
  } BaseMemoryAddress;
  __u32 uBoardID;
  __u16 usSlotNumber;
  __u8  bControllerClass;
  __u8  bIoBusType;
  CSMI_SAS_IO_BUS_ADDRESS BusAddress;
  __u8  szSerialNumber[81];
  __u16 usMajorRevision;
  __u16 usMinorRevision;
  __u16 usBuildRevision;
  __u16 usReleaseRevision;
  __u16 usBIOSMajorRevision;
  __u16 usBIOSMinorRevision;
  __u16 usBIOSBuildRevision;
  __u16 usBIOSReleaseRevision;
  __u32 uControllerFlags;
  __u16 usRromMajorRevision;
  __u16 usRromMinorRevision;
  __u16 usRromBuildRevision;
  __u16 usRromReleaseRevision;
  __u16 usRromBIOSMajorRevision;
  __u16 usRromBIOSMinorRevision;
  __u16 usRromBIOSBuildRevision;
  __u16 usRromBIOSReleaseRevision;
  __u8  bReserved[7];
} CSMI_SAS_CNTLR_CONFIG,
  *PCSMI_SAS_CNTLR_CONFIG;
```

6.2.4.4 CSMI_SAS_CNTLR_CONFIG_BUFFER

```
typedef struct _CSMI_SAS_CNTLR_CONFIG_BUFFER {
  IOCTL_HEADER IoctlHeader;
  CSMI_SAS_CNTLR_CONFIG Configuration;
} CSMI_SAS_CNTLR_CONFIG_BUFFER,
  *PCSMI_SAS_CNTLR_CONFIG_BUFFER;
```

6.3 CC_CSMI_SAS_GET_CNTLR_STATUS

(SATA, SAS)

6.3.1 Behavior

This CSMI functional behavior requests the current status of the controller. Any driver that implements this specification must support this behavior.

6.3.2 Input

A CSMI_SAS_CNTLR_STATUS_BUFFER data structure with the following initialized members;

- IoctlHeader, see IOCTL_HEADER for the platform specific initialization.
- Status, initialized to 0's.

6.3.3 Output

A CSMI_SAS_CNTLR_STATUS_BUFFER data structure with the returned data;

- IoctlHeader.ReturnCode. See Return Codes.
- Status.uStatus. Current status of the controller. Shall contain one of the following values;
  - **CSMI_SAS_CNTLR_STATUS_GOOD**, operating normally.
  - **CSMI_SAS_CNTLR_STATUS_FAILED**, the controller has failed. No I/O is allowed to the controller in this state.
  - **CSMI_SAS_CNTLR_STATUS_OFFLINE**, the controller is in a transitional state and is currently inaccessible. It has not failed, but no I/O is allowed to the controller in this state.
  - **CSMI_SAS_CNTLR_STATUS_POWEROFF**, the controller slot is powered off. It may have been failed before but currently does not have power to the slot.
- Status.uOfflineReason. The reason code for a **CSMI_SAS_CNTLR_STATUS_OFFLINE** value in uStatus. Shall contain one of the following values;
  - **CSMI_SAS_OFFLINE_REASON_NO_REASON**, unknown reason.
  - **CSMI_SAS_OFFLINE_REASON_INITIALIZING**, the driver is in the process of initializing the controller and bringing it online.
  - **CSMI_SAS_OFFLINE_REASON_BACKSIDE_BUS_DEGRADED**, the physical interface to the SAS or SATA domain is in a degraded state.
  - **CSMI_SAS_OFFLINE_REASON_BACKSIDE_BUS_FAILURE** the physical interface to the SAS or SATA domain has failed.

6.3.4 Structure Definitions

6.3.4.1 CSMI_SAS_CNTLR_STATUS

```
typedef struct _CSMI_SAS_CNTLR_STATUS {
  __u32 uStatus;
  __u32 uOfflineReason;
  __u8  bReserved[28];
} CSMI_SAS_CNTLR_STATUS,
  *PCSMI_SAS_CNTLR_STATUS;
```

6.3.4.2 CSMI_SAS_CNTLR_STATUS_BUFFER

```
typedef struct _CSMI_SAS_CNTLR_STATUS_BUFFER {
  IOCTL_HEADER IoctlHeader;
  CSMI_SAS_CNTLR_STATUS Status;
} CSMI_SAS_CNTLR_STATUS_BUFFER,
  *PCSMI_SAS_CNTLR_STATUS_BUFFER;
```

6.4 CC_CSMI_SAS_FIRMWARE_DOWNLOAD

(SATA, SAS) (OPTIONAL/RECOMMENDED)

6.4.1 Behavior

This CSMI functional behavior allows the controller firmware to be updated online. This is an optional behavior. The driver indicates support for this behavior in the **CC_CSMI_SAS_CNTLR_CONFIG** control. For the behavior to be successful, the uControllerFlags field in the **CSMI_SAS_CNTLR_CONFIG** structure must have **CSMI_SAS_CNTLR_FWD_SUPPORT** and **CSMI_SAS_CNTLR_FWD_ONLINE** set. One of **CSMI_SAS_CNTLR_FWD_SRESET** or **CSMI_SAS_CNTLR_FWD_HRESET** must be set depending on the upgrade reset behavior. The controller is responsible for validating the integrity of the ROM image before attempting to upgrade.

6.4.2 Input

A CSMI_SAS_FIRMWARE_DOWNLOAD_BUFFER data structure with the following initialized members;

- IoctlHeader, see IOCTL_HEADER for the platform specific initialization.
- Information, initialized to 0's.
- Information.uBufferLength. Shall be set to the length of the ROM image being downloaded in uDataBuffer.
- Information.bDownloadFlags. Control for the firmware download operation. Shall contain one or more of the following values;
  - **CSMI_SAS_FWD_VALIDATE**, validate the download image, but do not upgrade the image. If this operation cannot be supported, then return with Information.uStatus set to **CSMI_SAS_FWD_REJECT.**
  - **CSMI_SAS_FWD_SOFT_RESET**, download operation will initiate a soft reset to the controller after the ROM image has been upgraded. The driver will manage all I/O until the controller has returned to a ready state. If a soft reset is insufficient to complete a firmware download operation then Information.uStatus will return with **CSMI_SAS_FWD_REJECT** and the upgrade operation will not be initiated.
  - **CSMI_SAS_FWD_HARD_RESET**, download operation will initiate a hard reset to the controller after the ROM image has been upgraded. The driver will suspend all I/O until the controller has returned to a ready state. If a hard reset is insufficient to complete a firmware download operation then Information.uStatus will return with **CSMI_SAS_FWD_REJECT** and the upgrade operation will not be initiated.
- bDataBuffer. Contains the ROM image that is being written to the controller.

6.4.3 Output

A CSMI_SAS_FIRMWARE_DOWNLOAD_BUFFER data structure with the returned data;

- IoctlHeader.ReturnCode. See Return Codes.
- Information.uStatus. Status of the firmware download operation. Shall contain one of the following values;
  - **CSMI_SAS_FWD_SUCCESS**, download operation was successful.
  - **CSMI_SAS_FWD_FAILED**, download operation has failed. No I/O is allowed to the controller in this state.
  - **CSMI_SAS_FWD_USING_RROM**, download operation has failed and the controller is using the redundant ROM image.
  - **CSMI_SAS_FWD_REJECT**, download operation was rejected. The ROM image was corrupted, incorrect for this controller or validation is not supported.
  - **CSMI_SAS_FWD_DOWNREV**, download operation was successful, however the ROM image was an earlier revision than the executing image.
- Information.uSeverity. The severity code for the uStatus. Shall contain one of the following values;
  - **CSMI_SAS_FWD_INFORMATION**, uStatus is informational only.
  - **CSMI_SAS_FWD_WARNING**, uStatus is indicating a condition that may be helpful for diagnostic purposes.
  - **CSMI_SAS_FWD_ERROR**, uStatus is indicating a recoverable error condition.
  - **CSMI_SAS_FWD_FATAL,** uStatus is indicating a fatal error condition.

6.4.4 Structure Definitions

6.4.4.1 CSMI_SAS_FIRMWARE_DOWNLOAD

```
typedef struct _CSMI_SAS_FIRMWARE_DOWNLOAD {
  __u32 uBufferLength;
  __u32 uDownloadFlags;
  __u8   bReserved[32];
  __u16 usStatus;
  __u16 usSeverity;
} CSMI_SAS_FIRMWARE_DOWNLOAD,
  *PCSMI_SAS_FIRMWARE_DOWNLOAD;
```

6.4.4.2 CSMI_SAS_FIRMWARE_DOWNLOAD_BUFFER

```
typedef struct _CSMI_SAS_FIRMWARE_DOWNLOAD_BUFFER {
   IOCTL_HEADER IoctlHeader;
   CSMI_SAS_FIRMWARE_DOWNLOAD Information;
   __u8 bDataBuffer[1];
} CSMI_SAS_FIRMWARE_DOWNLOAD_BUFFER,
  *PCSMI_SAS_FIRMWARE_DOWNLOAD_BUFFER;
```

6.5 CC_CSMI_SAS_GET_RAID_INFO

(SATA, SAS)

6.5.1 Behavior

This CSMI functional behavior requests information on the number of RAID volumes and number of physical drives on a controller. The RAID solution may be implemented within the driver or as firmware on the controller. If the uControllerFlags in the CSMI_SAS_CNTLR_CONFIG structure indicates that the controller supports RAID, then the driver that implements this specification must support this CSMI functional behavior; otherwise the driver may respond to this control code with a generic IO error (see Submitting CSMI Control Codes).

6.5.2 Input

A CSMI_SAS_RAID_INFO_BUFFER data structure with the following initialized members;

- IoctlHeader, see IOCTL_HEADER for the platform specific initialization.
- Information, initialized to 0's.

6.5.3 Output

A CSMI_SAS_RAID_INFO_BUFFER data structure with the returned data;

- IoctlHeader.ReturnCode. See Return Codes.
- Information.uNumRaidSets. Number of logical RAID volumes (or sets) currently defined. If no volumes (or sets) have been defined, then a 0 value is returned.
- Information.uMaxDrivesPerSet. Maximum number of physical drives within a logical RAID volume. This may be an absolute maximum or the actual maximum currently defined for all volumes. This value will be used to allocate memory for the **CC_CSMI_SAS_GET_RAID_CONFIG** CSMI functional behavior.

6.5.4 Structure Definitions

6.5.4.1 CSMI_SAS_RAID_INFO

```
typedef struct _CSMI_SAS_RAID_INFO {
  __u32 uNumRaidSets;
  __u32 uMaxDrivesPerSet;
  __u8  bReserved[92];
} CSMI_SAS_RAID_INFO,
  *PCSMI_SAS_RAID_INFO;
```

6.5.4.2 CSMI_SAS_RAID_INFO_BUFFER

```
typedef struct _CSMI_SAS_RAID_INFO_BUFFER {
  IOCTL_HEADER IoctlHeader;
  CSMI_SAS_RAID_INFO Information;
} CSMI_SAS_RAID_INFO_BUFFER,
  *PCSMI_SAS_RAID_INFO_BUFFER;
```

6.6 CC_CSMI_SAS_GET_RAID_CONFIG

(SATA, SAS)

6.6.1 Behavior

This CSMI functional behavior requests information for a specified RAID set on a controller that supports RAID. To obtain the information for all the logical RAID sets defined; this CSMI functional behavior must be called for each RAID set of the controller. If the uControllerFlags in the CSMI_SAS_CNTLR_CONFIG structure indicates that the controller supports RAID; then the driver that implements this specification must support this CSMI functional behavior; otherwise the driver may respond to this control code with a generic IO error (see Submitting CSMI Control Codes).

6.6.2 Input

A CSMI_SAS_RAID_CONFIG_BUFFER data structure with the following initialized members;

- IoctlHeader, see IOCTL_HEADER for the platform specific initialization.
- Configuration, initialized to 0's.
- Configuration.uRaidSetIndex. Contains the number of the RAID set for which information is being requested. A calling routine would increment this value to enumerate the information for all RAID sets. If this value exceeds the number of RAID sets (see CC_CSMI_SAS_GET_RAID_INFO), then the IoctlHeader.ReturnCode shall be set to: **CSMI_SAS_RAID_SET_OUT_OF_RANGE**.

6.6.3 Output

A CSMI_SAS_RAID_CONFIG_BUFFER data structure with the returned data;

- IoctlHeader.ReturnCode. See Return Codes.
- Configuration.uRaidSetIndex. Same as input.
- Configuration.uCapacity. Contains the capacity of the RAID set in megabytes (MB = 1024 * 1024 bytes).
- Configuration.uStripSize. Contains the stripsize of the RAID set in kilobytes (KB = 1024 bytes).
- Configuration.bRaidType. Contains the basic RAID type of the RAID set. Shall be one of:
  - **CSMI_SAS_RAID_TYPE_NONE**, indicates the RAID set is composed of a single drive.
  - **CSMI_SAS_RAID_TYPE_0**, indicates the RAID set is a striped set, with no fault tolerance.
  - **CSMI_SAS_RAID_TYPE_1**, indicates the RAID set is a mirrored set.
  - **CSMI_SAS_RAID_TYPE_10**, indicates the RAID set is a striped mirror set.
  - **CSMI_SAS_RAID_TYPE_5**, indicates the RAID set is a parity set.
  - **CSMI_SAS_RAID_TYPE_15**, indicates the RAID set is an advanced parity set.
  - **CSMI_SAS_RAID_TYPE_OTHER**, indicates the RAID set type configuration does not match the standard types.
- Configuration.bStatus. Contains the status of the RAID set. Shall be one of:
  - **CSMI_SAS_RAID_SET_STATUS_OK**, indicates the RAID set is operational.
  - **CSMI_SAS_RAID_SET_STATUS_DEGRADED**, indicates the RAID set is no longer functioning in a fault tolerant mode.
  - **CSMI_SAS_RAID_SET_STATUS_REBUILDING**, indicates the RAID set is rebuilding. This implies a degraded operation. Once the rebuild completes successfully, the status will change to **CSMI_SAS_RAID_SET_STATUS_OK**. If the rebuilding process fails, the status will be updated appropriately.

  - **CSMI_SAS_RAID_SET_STATUS_FAILED**, indicates the RAID set has failed. There is no guarantee on the operational behavior of the RAID set and data loss has occurred or is imminent.
- Configuraton.bInformation. Contains clarifying information for bStatus results. The actual content depends on the bStatus result. Shall be:
  - If bStatus == **CSMI_SAS_RAID_SET_STATUS_OK**, then bInformation will be 0.
  - If bStatus == **CSMI_SAS_RAID_SET_STATUS_DEGRADED**, then bInformation will contain the failed drive index number.
  - If bStatus == **CSMI_SAS_RAID_SET_STATUS_REBUILDING**, then bInformation will contain the percentage complete. The value will be in the range of 0 to 100 (0h to 64h).
  - If bStatus == **CSMI_SAS_RAID_SET_STATUS_FAILED**, then bInformation will be 0 or vendor specific. Since the failure modes could include drive or controller failures, bInformation may provide a vendor specific error code to indicate which component led to the failed status.
- Configuration.bDriveCount. Contains the number of drives in the RAID set and in turn the number of **CSMI_SAS_RAID_DRIVES** data structures that will exist.
- Configuration.Drives[n]. Depending on bDriveCount, there will be from 0 to (bDriveCount – 1), **CSMI_SAS_RAID_DRIVES** data structures that can be indexed to provide information on each physical drive in the array.
- Configuration.Drives[n].bModel. Contains up to 40 ASCII characters of the drive model designation. This may not be a null terminated character string. The model designation shall be filled from the low order bytes to the high order bytes, with padding of 0's in any unused high order bytes. For SAS drives, the model is the concatenation of the vendor identification and product identification fields from a standard INQUIRY. For SATA drives, the model is from the IDENTIFY or IDENTIFY PACKET command response.
- Configuration.Drives[n].bFirmware. Contains up to 8 ASCII characters of the drive firmware designation. This may not be a null terminated character string. The firmware designation shall be filled from the low order bytes to the high order bytes, with padding of 0's in any unused high order bytes.
- Configuration.Drives[n].bSerialNumber. Contains up to 40 ASCII characters of the drive serial number designation. This is may not be a null terminated character string. The serial number designation shall be filled from the low order bytes to the high order bytes, with padding of 0's in any unused high order bytes.
- Configuration.Drives[n].bSASAddress. Contains the SAS address of the physical drive. If the drive does not have a SASAddress as is the case with a directly attached SATA drive, then this field shall be 0 filled.
- Configuration.Drives[n].bSASLun. Contains the SAS Lun of the physical drive. If the drive does not have a SAS Lun as is the case with a directly attached SATA drive, then this field shall be 0 filled.
- Configuration.Drives[n].bDriveStatus. Contains the status of the physical drive. Shall be one of:
  - **CSMI_SAS_DRIVE_STATUS_OK**, indicates the physical drive is operational.
  - **CSMI_SAS_DRIVE_STATUS_DEGRADED**, indicates the physical drive has posted a SMART notification to the controller.
  - **CSMI_SAS_DRIVE_STATUS_REBUILDING**, indicates the physical drive is the target drive of a RAID set rebuild. Once the rebuild completes successfully, the status will change to **CSMI_SAS_DRIVE_STATUS_OK**. If the rebuilding process fails, the status will be updated appropriately.
  - **CSMI_SAS_DRIVE_STATUS_FAILED**, indicates the physical drive has posted unrecoverable errors to the controller or has triggered a vendor specific action to remove the physical drive from the RAID set. There is no guarantee on the operational behavior of the drive and data loss has occurred or is imminent.
- Configuration.Drives[n].bDriveUsage. Contains whether the physical drive is part of the RAID set. Shall be one of:

- **CSMI_SAS_DRIVE_CONFIG_NOT_USED**, indicates the physical drive is not part of a RAID set.
- **CSMI_SAS_DRIVE_CONFIG_MEMBER**, indicates the physical drive is part of this RAID set.
- **CSMI_SAS_DRIVE_CONFIG_SPARE**, indicates the physical drive is part of this RAID set as a hot swap spare.

6.6.4 Structure Definitions

6.6.4.1 CSMI_SAS_RAID_DRIVES

```
typedef struct _CSMI_SAS_RAID_DRIVES {
  __u8  bModel[40];
  __u8  bFirmware[8];
  __u8  bSerialNumber[40];
  __u8  bSASAddress[8];
  __u8  bSASLun[8];
  __u8  bDriveStatus;
  __u8  bDriveUsage;
  __u8  bReserved[22];
} CSMI_SAS_RAID_DRIVES,
  *PCSMI_SAS_RAID_DRIVES;
```

6.6.4.2 CSMI_SAS_RAID_CONFIG

```
typedef struct _CSMI_SAS_RAID_CONFIG {
  __u32 uRaidSetIndex;
  __u32 uCapacity;
  __u32 uStripeSize;
  __u8  bRaidType;
  __u8  bStatus;
  __u8  bInformation;
  __u8  bDriveCount;
  __u8  bReserved[20];
  CSMI_SAS_RAID_DRIVES Drives[1];
} CSMI_SAS_RAID_CONFIG,
  *PCSMI_SAS_RAID_CONFIG;
```

6.6.4.3 CSMI_SAS_RAID_CONFIG_BUFFER

```
typedef struct _CSMI_SAS_RAID_CONFIG_BUFFER {
  IOCTL_HEADER IoctlHeader;
  CSMI_SAS_RAID_CONFIG Configuration;
} CSMI_SAS_RAID_CONFIG_BUFFER,
  *PCSMI_SAS_RAID_CONFIG_BUFFER;
```

6.7 CC_CSMI_SAS_GET_PHY_INFO

(SATA, SAS)

6.7.1 Behavior

This CSMI functional behavior requests information about the physical characteristics and interconnect to the SATA or SAS domain. Any driver that implements this specification must support this behavior.

6.7.2 Input

A CSMI_SAS_PHY_INFO_BUFFER data structure with the following initialized members;

- IoctlHeader, see IOCTL_HEADER for the platform specific initialization.
- Information, initialized to 0's.

6.7.3 Output

A CSMI_SAS_PHY_INFO_BUFFER data structure with the returned data;

- IoctlHeader.ReturnCode. See Return Codes.
- Information.bNumberOfPhys. Contains the number of phys (real or virtual) supported by this controller. It is possible for a controller and/or driver to contain a virtual phy that supports one or more of the SAS protocols. A management or test application shall not assume that all phys are real.
- Information.Phy[0 - 31]. Contains 32 data structures each of which defines the physical characteristics and provides information on the device attached to each interconnect.
- Information.Phy[n].Identify. Contains a data structure with information that will be transferred to the attached device during a link reset sequence as defined in the SAS specification. If the controller is a SATA implementation, then the link reset sequence is not transmitted, but the content of this data structure will define this controller as a SATA solution.
- Information.Phy[n].Identify.bDeviceType. Contains the SAS device type. Shall be one of the following:
  - **CSMI_SAS_PHY_UNUSED**, indicates that the phy cannot be attached to a physical device.
  - **CSMI_SAS_END_DEVICE**, indicates that the phy will have the characteristics of a SAS end device. A SATA controller would define a SATA device as an end device.
- Information.Phy[n].Identify.bInitiatorPortProtocol. Contains information on which SAS initiator protocols are supported by this initiator on this phy. Shall be one or more of the following:
  - **CSMI_SAS_PROTOCOL_SATA**, indicates the controller may support a directly attached SATA device. This protocol bit is used to notify the management or test application about the SATA capabilities of a controller, it will be masked out from the data provided during a SAS link reset sequence. A SAS controller may support this protocol. A SATA controller must support this protocol.
  - **CSMI_SAS_PROTOCOL_SMP**, indicates the controller may support connection to a SAS expander device. A SAS controller must support this protocol. A SATA controller may support this protocol.
  - **CSMI_SAS_PROTOCOL_STP**, indicates the controller may support connection to a tunneled SATA device. A SAS or SATA controller may support this protocol.
  - **CSMI_SAS_PROTOCOL_SSP**, indicate the controller may support connection to a SAS end device. A SAS controller must support this protocol. If a SATA controller supports this protocol, then it is by definition a SAS controller.

- Information.Phy[n].Identify.bTargetPortProtocol. Contains information on which SAS target protocols are supported by this initiator on this phy. Initiators are not required to support any target protocols, so this field would typically be 0. However an initiator may have target capabilities and in that event, this field shall be one or more of the following:
  - **CSMI_SAS_PROTOCOL_SATA**, indicates the controller may respond as a SATA device. This protocol bit is used to notify the management or test application about the SATA capabilities of a controller, it will be masked out from the data provided during a SAS link reset sequence. A SAS or SATA controller may support this protocol.
  - **CSMI_SAS_PROTOCOL_SMP**, indicates the controller may respond as a SAS expander device. A SAS or SATA controller may support this protocol.
  - **CSMI_SAS_PROTOCOL_STP**, indicates the controller may respond as a tunneled SATA device. A SAS or SATA controller may support this protocol.
  - **CSMI_SAS_PROTOCOL_SSP**, indicate the controller may respond as a SAS end device. A SAS controller may support this protocol.
- Information.Phy[n].Identify.bSASAddress. Contains the SAS address in MSB order. A SATA controller would return a 0 for this field.
- Information.Phy[n].Identify.bPhyIdentifier. Contains the phy identifier of this phy. The range of the value must be from 0 to (bNumberOfPhys – 1). This value is restricted to a maximum of 254 (FEh), because FFh is a reserved identifier used to indicate a “don’t care” for other CSMI functional behaviors.
- Information.Phy[n].bPortIdentifier. Contains the port identifier associated with this phy. The range of the value must be from 0 to (bNumberOfPhys - 1). Multiple phys may be associated with the same port, because of wide links in SAS. For example, a 4 wide link (phys 0 - 3) from an initiator to an expander would all reference one port (port 0).
- Information.Phy[n].bNegotiatedLinkRate. Contains the current link rate of this phy. Shall be one of the following:
  - **CSMI_SAS_LINK_RATE_UNKNOWN**, indicates the link may currently be unconnected, or that a link rate does not apply, as is the case with a virtual phy.
  - **CSMI_SAS_PHY_DISABLED**, indicates the phy has been disabled.
  - **CSMI_SAS_LINK_RATE_FAILED**, indicates that a link rate negotiation has failed. In this case, there appears to be a device connected, because the link reset sequence has been initiated, but communication was not established.
  - **CSMI_SAS_SATA_SPINUP_HOLD**, indicates that a link has detected a SATA device attached and is in a wait state to release a spin-up hold. A SATA drive will use this mechanism to stage the power surges associated with spin-up.
  - **CSMI_SAS_LINK_RATE_1_5_GBPS**, indicates that a link was established at 1.5 Gb/s.
  - **CSMI_SAS_LINK_RATE_3_0_GBPS**, indicates that a link was established at 3.0 Gb/s.
- Information.Phy[n].bMinimumLinkRate. Contains the minimum link rate for this phy. This field incorporates information for both the programmed and hardware link rate. Shall be one of the following:
  - **CSMI_SAS_LINK_RATE_1_5_GBPS**, indicates the minimum link rate for this phy is 1.5 Gb/s.
  - **CSMI_SAS_LINK_RATE_3_0_GBPS**, indicates the minimum link rate for this phy is 3.0 Gb/s.

  In combination with one of the following:
  - **CSMI_SAS_PROGRAMMED_LINK_RATE_1_5_GBPS**, indicates the minimum link rate programmed for this phy is 1.5 Gb/s.
    CSMI_SAS_PROGRAMMED_LINK_RATE_3_0_GBPS, indicates the minimum link rate programmed for this phy is 3.0 Gb/s.

- Information.Phy[n].bMaximumLinkRate. Contains the maximum link rate for this phy. This field incorporates information for both the programmed and hardware link rate. Shall be one of the following:
  - **CSMI_SAS_LINK_RATE_1_5_GBPS**, indicates the maximum link rate for this phy is 1.5 Gb/s.
  - **CSMI_SAS_LINK_RATE_3_0_GBPS**, indicates the maximum link rate for this phy is 3.0 Gb/s.

  In combination with one of the following:
  - **CSMI_SAS_PROGRAMMED_LINK_RATE_1_5_GBPS**, indicates the maximum link rate programmed for this phy is 1.5 Gb/s.
  - **CSMI_SAS_PROGRAMMED_LINK_RATE_3_0_GBPS**, indicates the maximum link rate programmed for this phy is 3.0 Gb/s.
- Information.Phy[n].bPhyChangeCount. Contains the current count of BROADCAST(CHANGE) primitives received on this phy. This count needs to be updated according to the SAS specfication. A SATA controller shall update this count anytime a hotplug event is detected. If the SATA controller does not support hotplug detection, then this value shall remain 0.
- Information.Phy[n].bAutoDiscover. Contains the current state of the discover process for the SAS Domain. The auto-discover process may begin at power on or may be initiated by an OS platform specific method. For example, an **IOCTL_SCSI_RESCAN_BUS** I/O control function on Windows will initiate auto-discover. Auto-discovery can only be interrupted in a vendor specific manner.
  - **CSMI_SAS_DISCOVER_NOT_SUPPORTED**, indicates that auto-discover is not supported. A SATA controller shall set this value.
  - **CSMI_SAS_DISCOVER_NOT_STARTED**, indicates that auto-discover is supported, but has not begun.
  - **CSMI_SAS_DISCOVER_IN_PROGRESS**, indicates that the auto-discover process is in progress. Any address translation or routing errors that occur during this period shall be retried.
  - **CSMI_SAS_DISCOVER_COMPLETE**, indicates that the auto-discover process has completed successfully.
  - **CSMI_SAS_DISCOVER_ERROR**, indicates that the auto-discover process has completed with a vendor unique or topology error. The driver may have a vendor unique mechanism to determine where the error occurred. A management or test application may need to examine the topology to determine the cause of the error. Address translation, routing errors, or command errors may result if this state is entered.
- Information.Phy[n].Attached. Contains a data structure with information that defines the attached device. If the attached device is a SATA device, then the controller will generate a pseudo representation of the information.
- Information.Phy[n].Attached.bDeviceType. Contains the SAS device type attached to this phy. Shall be one of the following:
  - **CSMI_SAS_NO_DEVICE_ATTACHED**, indicates that the phy is not currently attached to a device. A SATA controller will set this value if no device is attached.
  - **CSMI_SAS_END_DEVICE**, indicates that the phy is connected to a SAS target or a SATA device. A SATA controller will set this value if a device is attached.
  - **CSMI_SAS_EDGE_EXPANDER_DEVICE**, indicate that the phy is connected to a SAS edge expander.
  - **CSMI_SAS_FANOUT_EXPANDER_DEVICE**, indicate that the phy is connected to a SAS fanout expander.

- Information.Phy[n].Attached.bInitiatorPortProtocol. Contains information on which SAS initiator protocols are supported by the attached device. May be one or more of the following:
  - **CSMI_SAS_PROTOCOL_SATA**, indicates the attached device is a directly attached SATA host device. Currently if this bit is set it would indicate a topology error.
  - **CSMI_SAS_PROTOCOL_SMP**, indicates the attached device supports generating SMP requests.
  - **CSMI_SAS_PROTOCOL_STP**, indicates the attached device supports generating STP commands.
  - **CSMI_SAS_PROTOCOL_SSP**, indicate the attached device supports generating SSP commands.
- Information.Phy[n].Attached.bTargetPortProtocol. Contains information on which SAS target protocols are supported by the attached device. Shall be one or more of the following:
  - **CSMI_SAS_PROTOCOL_SATA**, indicates the attached device is a directly attached SATA device.
  - **CSMI_SAS_PROTOCOL_SMP**, indicates the attached device supports receiving SMP requests.
  - **CSMI_SAS_PROTOCOL_STP**, indicates the attached device support receiving STP commands.
  - **CSMI_SAS_PROTOCOL_SSP**, indicate the attached device supports receiving SSP commands.
- Information.Phy[n].Attached.bSASAddress. Contains the SAS address of the attached device. If the device is a SATA device, then this field may be 0.
- Information.Phy[n].Attached.bPhyIdentifier. Contains the phy identifier of the attached device. If the device is a SATA device, then this field shall be 0.

6.7.4 Structure Definitions

6.7.4.1 CSMI_SAS_IDENTIFY

```
typedef struct _CSMI_SAS_IDENTIFY {
  __u8  bDeviceType;
  __u8  bRestricted;
  __u8  bInitiatorPortProtocol;
  __u8  bTargetPortProtocol;
  __u8  bRestricted2[8];
  __u8  bSASAddress[8];
  __u8  bPhyIdentifier;
  __u8  bSignalClass;
  __u8  bReserved[6];
} CSMI_SAS_IDENTIFY,
  *PCSMI_SAS_IDENTIFY;
```

6.7.4.2 CSMI_SAS_PHY_ENTITY

```
typedef struct _CSMI_SAS_PHY_ENTITY {
  CSMI_SAS_IDENTIFY Identify;
  __u8  bPortIdentifier;
  __u8  bNegotiatedLinkRate;
  __u8  bMinimumLinkRate;
  __u8  bMaximumLinkRate;
  __u8  bPhyChangeCount;
  __u8  bAutoDiscover;
  __u8  bReserved[2];
```

```
    CSMI_SAS_IDENTIFY Attached;
  } CSMI_SAS_PHY_ENTITY,
   *PCSMI_SAS_PHY_ENTITY;
```

6.7.4.3 CSMI_SAS_PHY_INFO

```
typedef struct _CSMI_SAS_PHY_INFO {
    __u8  bNumberOfPhys;
    __u8  bReserved[3];
    CSMI_SAS_PHY_ENTITY Phy[32];
} CSMI_SAS_PHY_INFO,
  *PCSMI_SAS_PHY_INFO;
```

6.7.4.4 CSMI_SAS_PHY_INFO_BUFFER

```
typedef struct _CSMI_SAS_PHY_INFO_BUFFER {
    IOCTL_HEADER IoctlHeader;
    CSMI_SAS_PHY_INFO Information;
} CSMI_SAS_PHY_INFO_BUFFER,
  *PCSMI_SAS_PHY_INFO_BUFFER;
```

6.8 CC_CSMI_SAS_SET_PHY_INFO
(SATA, SAS)

6.8.1 Behavior

This CSMI functional behavior requests that the physical characteristics of a phy be changed. Any driver that implements this specification must support this behavior. Even though this CSMI functional behavior must be supported, the changing of physical characteristics is not required.

A driver may choose to post an IoctlHeader.ReturnCode of **CSMI_SAS_PHY_INFO_NOT_CHANGEABLE** and not implement any further behavior.

Upon completion of a phy characteristic change, a driver shall post an IoctlHeader.ReturnCode of **CSMI_SAS_PHY_INFO_CHANGED** and a link reset sequence will be initiated.

6.8.1.1 Signal Class

Signal Class is a mechanism to allow initiators and expanders to provide a hint to end devices on the type of transport media between the HBA and end device. Using this hint, the end device can adjust the phy signaling level appropriate to compensate for the potential loss through the transport media. There are three classes of transport media; direct cable, server backplane, and enclosure backplane (see Table 4).

Table 4: Signal Class Characteristics

Class 0	**Unknown Interconnect**
	• The system is not providing any hint of the transport characteristics
Class 1	**Direct Cable**
	• No more than 2 interconnects • Maximum of 2 dB loss • Maximum cable length of 8 m
Class 2	**Server Backplane and Cable**
	• No more than 3 interconnects • Maximum of 5 dB loss • Maximum cable length of 1 m • Maximum backplane length of 0.5 m with FR4 loss characteristics
Class 3	**Enclosure Backplane, Intermediate Media and Cable**
	• 3 or more interconnects • Maximum of 6 dB loss • Maximum cable length of 8 m • Maximum backplane length of 0.5 m with FR4 loss characteristics • Additional transition through an intermediate media

For initiators to support the Signal Class concept they must be capable of modifying the Identify Address Frame they originate. The Signal Class redefines a reserved field in byte 21 to support a Signal Class field in byte 21. The Identify Address Frame change is shown in Table 5. The bSignalClass field allows changing the signal class hint the HBA will provide.

Table 5: Identify Address Frame

Bit / Byte	7	6	5	4	3	2	1	0
0	Restricted	Device Type			Address Frame Type (0h)			
1	Restricted							
2	Reserved				SSP Initiator Port	STP Initiator Port	SMP Initiator Port	Restricted
3	Reserved				SSP Target Port	STP Target Port	SMP Target Port	Restricted
4 – 11	Restricted							
12 –19	SAS Address							
20	Phy Identifier							
21	Reserved				Signal Class			
22 – 27	Reserved							
28 – 31	CRC							

Note: The Signal Class proposal has not been proposed to the T10 SAS working group at this time. This requirement may change based on the acceptance of the proposal when submitted.

6.8.2 Input

A CSMI_SAS_SET_PHY_INFO_BUFFER data structure with the following initialized members;

- IoctlHeader, see IOCTL_HEADER for the platform specific initialization.
- Information, initialized to 0's.
- Information.bPhyIdentifier. Contains the phy identifier of the phy to modify.
- Information.bNegotiatedLinkRate. Contains the directive to negotiate a new link rate or to disable the phy. Shall be one of the following:
  - **CSMI_SAS_LINK_RATE_NEGOTIATE**, indicates to negotiate a new link rate constrained by the bProgrammedMinimumLInkRate and bProgrammedMaximumLinkRate values provided.
  - **CSMI_SAS_LINK_RATE_PHY_DISABLED**, indicates that the phy shall be disabled. The values for bProgrammedMinimumLInkRate and bProgrammedMaximumLinkRate will be updated after the phy has been disabled. A link reset sequence will not occur.
- Information.bProgrammedMinimumLinkRate. Contains the value used to update the minimum programmed link rate for this phy. If the value is outside the range of the hardware minimum and maximum link rates or greater than bProgrammedMaximumLinkRate then IoctlHeader.ReturnCode = **CSMI_SAS_LINK_RATE_OUT_OF_RANGE** is returned. Shall be one of the following:
  - **CSMI_SAS_PROGRAMMED_LINK_RATE_UNCHANGED**, indicates that the programmed minimum link rate shall be unchanged.
  - **CSMI_SAS_PROGRAMMED_LINK_RATE_1_5_GBPS**, indicates that the programmed minimum link rate shall be updated to 1.5 Gb/s.

    - **CSMI_SAS_PROGRAMMED_LINK_RATE_3_0_GBPS**, indicates that the programmed minimum link rate shall be updated to 3.0 Gb/s.
- Information.bProgrammedMaximumLinkRate. Contains the value used to update the maximum programmed link rate for this phy. If the value is outside the range of the hardware minimum and maximum link rates or less than bProgrammedMinimumLinkRate then IoctlHeader.ReturnCode = **CSMI_SAS_LINK_RATE_OUT_OF_RANGE** is returned. Shall be one of the following:
    - **CSMI_SAS_PROGRAMMED_LINK_RATE_UNCHANGED**, indicates that the programmed maximum link rate shall be unchanged.
    - **CSMI_SAS_PROGRAMMED_LINK_RATE_1_5_GBPS**, indicates that the programmed maximum link rate shall be updated to 1.5 Gb/s.
    - **CSMI_SAS_PROGRAMMED_LINK_RATE_3_0_GBPS**, indicates that the programmed maximum link rate shall be updated to 3.0 Gb/s.
- Information.bSignalClass. Contains one of the following:
    - CSMI_SAS_SIGNAL_CLASS_UNKNOWN, indicates that the Signal Class field in the Identify frame from the initiator shall be set to 0.
    - CSMI_SAS_SIGNAL_CLASS_DIRECT, indicates that the Signal Class field in the Identify frame from the initiator shall be set to 1.
    - CSMI_SAS_SIGNAL_CLASS_SERVER, indicates that the Signal Class field in the Identify frame from the initiator shall be set to 2.
    - CSMI_SAS_SIGNAL_CLASS_ENCLOSURE, indicates that the Signal Class field in the Identify frame from the initiator shall be set to 3.

6.8.3 Output

A CSMI_SAS_SET_PHY_INFO_BUFFER data structure with the returned data;

- IoctlHeader.ReturnCode. See Return Codes.
- Information.bPhyIdentifier. Same as input.
- Information.bNegotiatedLinkRate. Same as input.
- Information.bProgrammedMinimumLinkRate. Same as input.
- Information.bProgrammedMaximumLinkRate. Same as input.
- Information.bSignalClass. Same as input.

6.8.4 Structure Definitions

6.8.4.1 CSMI_SAS_SET_PHY_INFO

```
typedef struct _CSMI_SAS_SET_PHY_INFO {
  __u8  bPhyIdentifier;
  __u8  bNegotiatedLinkRate;
  __u8  bProgrammedMinimumLinkRate;
  __u8  bProgrammedMaximumLinkRate;
  __u8  bSignalClass;
  __u8  bReserved[3];
} CSMI_SAS_SET_PHY_INFO,
  *PCSMI_SAS_SET_PHY_INFO;
```

6.8.4.2 CSMI_SAS_SET_PHY_INFO_BUFFER

```
typedef struct _CSMI_SAS_SET_PHY_INFO_BUFFER {
  IOCTL_HEADER IoctlHeader;
  CSMI_SAS_SET_PHY_INFO Information;
} CSMI_SAS_SET_PHY_INFO_BUFFER,
  *PCSMI_SAS_SET_PHY_INFO_BUFFER;
```

6.9 CC_CSMI_SAS_GET_LINK_ERRORS (SAS)

6.9.1 Behavior

This CSMI functional behavior requests information on the link errors associated with a specific phy. Any driver that implements this specification must support this behavior. If the controller cannot support tracking of one or more of the errors indicated, then the associated error counter shall contain 0. If the controller can track one or more of the errors indicated, then the reset flag must be supported. If the controller cannot support any of the link error counters, then it may post an IoctlHeader.ReturnCode of **CSMI_SAS_STATUS_FAILED**.

6.9.2 Input

A CSMI_SAS_LINK_ERRORS_BUFFER data structure with the following initialized members;

- IoctlHeader, see IOCTL_HEADER for the platform specific initialization.
- Information, initialized to 0's.
- Information.bPhyIdentifier. Contains the phy identifier of the phy to return link error information. If the phy identifier specified is to an unsupported or non-existing phy, then an IoctlHeader.ReturnCode of CSMI_SAS_PHY_DOES_NOT_EXIST is returned.
- Information.bResetCounts. Contains a flag to reset the error counts on return. Shall one of the following:
  - **CSMI_SAS_LINK_ERROR_DONT_RESET_COUNTS**, indicates that the error counts shall not be reset.
  - **CSMI_SAS_LINK_ERROR_RESET_COUNTS**, indicates that the error counts shall be reset.

6.9.3 Output

A CSMI_SAS_LINK_ERRORS_BUFFER data structure with the returned data;

- IoctlHeader.ReturnCode. See Return Codes.
- Information.bPhyIdentifier. Same as input.
- Information.bResetCounts. Same as input.
- Information.uInvalidDwordCount. Refer to the SAS specification, SMP REPORT PHY ERROR LOG for information on what is expected for this counter.
- Information.uRunningDisparityErrorCount. Refer to the SAS specification, SMP REPORT PHY ERROR LOG for information on what is expected for this counter.
- Information.uLossOfDwordSyncCount. Refer to the SAS specification, SMP REPORT PHY ERROR LOG for information on what is expected for this counter.
- Information.uPhyResetProblemCount. Refer to the SAS specification, SMP REPORT PHY ERROR LOG for information on what is expected for this counter.

6.9.4 Structure Definitions

6.9.4.1 CSMI_SAS_LINK_ERRORS

```
typedef struct _CSMI_SAS_LINK_ERRORS {
  __u8  bPhyIdentifier;
  __u8  bResetCounts;
  __u8  bReserved[2];
  __u32 uInvalidDwordCount;
  __u32 uRunningDisparityErrorCount;
  __u32 uLossOfDwordSyncCount;
  __u32 uPhyResetProblemCount;
} CSMI_SAS_LINK_ERRORS,
  *PCSMI_SAS_LINK_ERRORS;
```

6.9.4.2 CSMI_SAS_LINK_ERRORS_BUFFER

```
typedef struct _CSMI_SAS_LINK_ERRORS_BUFFER {
  IOCTL_HEADER IoctlHeader;
  CSMI_SAS_LINK_ERRORS Information;
} CSMI_SAS_LINK_ERRORS_BUFFER,
  *PCSMI_SAS_LINK_ERRORS_BUFFER;
```

6.10 CC_CSMI_SAS_SMP_PASSTHRU
(SAS)

6.10.1 Behavior

This CSMI functional behavior provides a method of sending generic SMP requests to a specific SAS address. Any driver that implements this specification and supports the SMP protocol must support this behavior; otherwise the driver may respond to this control code with a generic IO error (see Submitting Control Codes).

6.10.1.1 Security

A driver may post IoctlHeader.ReturnCode = CSMI_SAS_SCSI_WRITE_ATTEMPTED, if the security level is insufficient to complete the function requested (see Security and Enabling Features). Vendor unique functions are only allowed if Full access is enabled. The security access required for the standard SMP functions is provided in Table 6. As SMP standard functions are added the security shall be applied appropriately.

Table 6: SMP Functions, Security Access

SMP Function	Function Code	Security Access
REPORT GENERAL	00H	RESTRICTED
REPORT MANUFACTURER INFORMATION	01H	RESTRICTED
DISCOVER	10H	RESTRICTED
REPORT PHY ERROR LOG	11H	RESTRICTED
REPORT PHY SATA	12H	RESTRICTED
REPORT ROUTE INFORMATION	13H	RESTRICTED
CONFIGURE ROUTE INFORMATION	90H	FULL
PHY CONTROL	91H	FULL
VENDOR SPECIFIC	ALL OTHERS	FULL

6.10.2 Input

A CSMI_SAS_SMP_PASSTHRU_BUFFER data structure with the following initialized members;

- IoctlHeader, see IOCTL_HEADER for the platform specific initialization.
- Parameters, initialized to 0's.
- Parameters.bPhyIdentifier. Contains the phy identifier that specifies which phy shall be used to issue the request. The value must be in the range of 0 to 254 (0 to FEh) or be the directive **CSMI_SAS_USE_PORT_IDENTIFIER**. The driver may generate an error due to the phy identifier for the following reasons:
  - The phy identifier cannot be selected because the driver does not support sending SMP requests to a phy identifier. The driver may support sending SMP requests to a bPortIdentifier only. The IoctlHeader.ReturnCode would be **CSMI_SAS_PHY_CANNOT_BE_SELECTED**.
  - The phy identifier is out of range of valid phys. The IoctlHeader.ReturnCode would be **CSMI_SAS_PHY_DOES_NOT_EXIST**.
  - The phy identifier cannot be associated with bPortIdentifier. If bPhyIdentifier is intended to reference the phy and the bPortIdentifier value is not **CSMI_SAS_IGNORE_PORT**, then the bPhyIdentifier and bPortIdentifier must have the proper association. The IoctlHeader.ReturnCode would be **CSMI_SAS_PHY_DOES_NOT_MATCH_PORT**.
  - The phy identifier has a value of **CSMI_SAS_USE_PORT_IDENTIFIER** and the bPortIdentifier has a value of **CSMI_SAS_IGNORE_PORT.** The driver cannot

determine where to send the SMP request. Either bPhyIdentifier shall reference a valid phy or bPortIdentifier must reference a valid port. The IoctlHeader.ReturnCode would be **CSMI_SAS_SELECT_PHY_OR_PORT**.

- Parameters.bPortIdentifier. Contains the port identifier that specifies which port shall be used to issue the request. The value must be in the range of 0 to 254 (0 to FEh) or be the directive **CSMI_SAS_IGNORE_PORT**. The driver may generate an error due to the port identifier for the following reasons:
  - The port identifier cannot be selected because the driver does not support sending SMP requests to a port identifier. The driver may support sending SMP requests to a bPhyIdentifier only. The IoctlHeader.ReturnCode would be **CSMI_SAS_PORT_CANNOT_BE_SELECTED**.
  - The port identifier is out of range of valid ports. The IoctlHeader.ReturnCode would be **CSMI_SAS_PORT_DOES_NOT_EXIST**.
  - The port identifier cannot be associated with bPhyIdentifier. If bPortIdentifier is intended to reference the port and the bPhyIdentifier value is not **CSMI_SAS_USE_PORT_IDENTIFIER**, then the bPortIdentifier and bPhyIdentifier must have the proper association. The IoctlHeader.ReturnCode would be **CSMI_SAS_PHY_DOES_NOT_MATCH_PORT**.
  - The port identifier has a value of **CSMI_SAS_IGNORE_PORT** and the bPhyIdentifier has a value of **CSMI_SAS_USE_PORT_IDENTIFIER.** The driver cannot determine where to send the SMP request. Either bPhyIdentifier shall reference a valid phy or bPortIdentifier must reference a valid port. The IoctlHeader.ReturnCode would be **CSMI_SAS_SELECT_PHY_OR_PORT**.
- Parameters.bConnectionRate. Contains the connection rate directive for the driver connection manager. Shall be one of the following:
  - CSMI_SAS_LINK_RATE_NEGOTIATED, indicates that the connection shall be opened at the highest allowable negotiated rate for the destination device. The resulting rate will be the lowest common denominator of link rates along a connection pathway.
  - CSMI_SAS_LINK_RATE_1_5_GBPS, indicates that the connection shall be attempted at 1.5 Gb/s.
  - CSMI_SAS_LINK_RATE_3_0_GBPS, indicates that the connection shall be attempted at 3.0 Gb/s. This connection rate may not succeed if an intermediate link is less than 3.0 Gb/s.
- Parameters.bDestinationSASAddress. Contains the SAS address of the destination device in MSB order.
- Parameters.uRequestLength. Contains the length of the function specific content in Parameters.Request. The length shall be in LSB order and shall not include the CRC bytes. The driver will be responsible for appending the proper CRC to the request at the uRequestLength offset using the function specific content.
- Parameters.Request. Contains the function specific content for the SMP request.
- Parameters.Request.bFrameType. Shall be initialized to 40h.
- Parameters.Request.bFunction. Contains the SMP function to request. Refer to the SAS specification.
- Parameters.Request.bAdditionalRequestBytes. Contains the payload bytes for the SMP function requested. The data shall be in MSB order. Any unused bytes shall be filled with 0's.

6.10.3 Output

A CSMI_SAS_SMP_PASSTHRU_BUFFER data structure with the returned data;

- IoctlHeader.ReturnCode. See Return Codes.
- Parameters.bPhyIdentifier. Same as input.
- Parameters.bPortIdentifier. Same as input.
- Parameters.bConnectionRate. Same as input.

- Parameters.bDestinationSASAddress. Same as input.
- Parameters.uRequestLength. Same as input.
- Parameters.Request. Same as input.
- Parameters.bConnectionStatus. Contains the results of the open connection attempt. Shall be one of the following:
  - **CSMI_SAS_OPEN_ACCEPT**, indicate the connection was successful and the request was submitted.
  - **CSMI_SAS_OPEN_REJECT_BAD_DESTINATION**, indicates the destination address was not found because the destination address was improper, no request was submitted.
  - **CSMI_SAS_OPEN_REJECT_RATE_NOT_SUPPORTED**, indicates the requested link rate could not be used, no request was submitted.
  - **CSMI_SAS_OPEN_REJECT_NO_DESTINATION**, indicates the destination address was not found, no request was submitted.
  - **CSMI_SAS_OPEN_REJECT_PATHWAY_BLOCKED**, see SAS specification, no request was submitted.
  - **CSMI_SAS_OPEN_REJECT_PROTOCOL_NOT_SUPPORTED**, indicates the destination device does not support the protocol requested, no request was submitted.
  - **CSMI_SAS_OPEN_REJECT_RESERVE_ABANDON**, see SAS specification, no request was submitted.
  - **CSMI_SAS_OPEN_REJECT_RESERVE_CONTINUE**, see SAS specification, no request was submitted.
  - **CSMI_SAS_OPEN_REJECT_RESERVE_INITIALIZE**, see SAS specification, no request was submitted.
  - **CSMI_SAS_OPEN_REJECT_RESERVE_STOP**, see SAS specification, no request was submitted.
  - **CSMI_SAS_OPEN_REJECT_RETRY**, see SAS specification, no request was submitted.
  - **CSMI_SAS_OPEN_REJECT_STP_RESOURCES_BUSY**, see SAS specification, no request was submitted.
  - **CSMI_SAS_OPEN_REJECT_WRONG_DESTINATION**, see SAS specification, no request was submitted.
- Parameters.bResponseBytes. Contains the number of valid bytes in the Parameters.Response data structure. The CRC bytes of the response my optionally be included in the count.
- Parameters.Response. Contains the function specific response. See the SAS specification.
- Parameters.Response.bFrameType. Contains the SMP response type. Shall be 41h.
- Parameters.Response.bFunction. Contains the SMP function.
- Parameters.Response.bFunctionResult. Contains the SMP function result.
- Parameters.Response.bAdditionalRequestBytes. Contains the payload bytes for the SMP function response. The data shall be in MSB order. Any trailing unused bytes shall be filled with 0's.

6.10.4 Structure Definitions

6.10.4.1 CSMI_SAS_SMP_REQUEST

```
typedef struct _CSMI_SAS_SMP_REQUEST {
  __u8  bFrameType;
  __u8  bFunction;
  __u8  bReserved[2];
  __u8  bAdditionalRequestBytes[1016];
```

```
} CSMI_SAS_SMP_REQUEST,
  *PCSMI_SAS_SMP_REQUEST;
```

6.10.4.2 CSMI_SAS_SMP_RESPONSE

```
typedef struct _CSMI_SAS_SMP_RESPONSE {
    __u8  bFrameType;
    __u8  bFunction;
    __u8  bFunctionResult;
    __u8  bReserved;
    __u8  bAdditionalResponseBytes[1016];
} CSMI_SAS_SMP_RESPONSE,
  *PCSMI_SAS_SMP_RESPONSE;
```

6.10.4.3 CSMI_SAS_SMP_PASSTHRU

```
typedef struct _CSMI_SAS_SMP_PASSTHRU {
    __u8  bPhyIdentifier;
    __u8  bPortIdentifier;
    __u8  bConnectionRate;
    __u8  bReserved;
    __u8  bDestinationSASAddress[8];
    __u32 uRequestLength;
    CSMI_SAS_SMP_REQUEST Request;
    __u8  bConnectionStatus;
    __u8  bReserved2[3];
    __u32 uResponseBytes;
    CSMI_SAS_SMP_RESPONSE Response;
} CSMI_SAS_SMP_PASSTHRU,
  *PCSMI_SAS_SMP_PASSTHRU;
```

6.10.4.4 CSMI_SAS_SMP_PASSTHRU_BUFFER

```
typedef struct _CSMI_SMP_PASSTHRU_BUFFER {
    IOCTL_HEADER IoctlHeader;
    CSMI_SAS_SMP_PASSTHRU Parameters;
} CSMI_SAS_SMP_PASSTHRU_BUFFER,
  *PCSMI_SAS_SMP_PASSTHRU_BUFFER;
```

6.11 CC_CSMI_SAS_SSP_PASSTHRU
(SAS)

6.11.1 Behavior

This CSMI functional behavior provides a method of sending generic SSP requests to a specific SAS address. Any driver that implements this specification and supports the SSP protocol must support this behavior; otherwise the driver may respond to this control code with a generic IO error (see Submitting Control Codes).

6.11.1.1 Security

A driver may post IoctrlHeader.ReturnCode = CSMI_SAS_SCSI_WRITE_ATTEMPTED, if the security level is insufficient to complete the command requested (see Security and Enabling Features). Vendor unique commands are only allowed if Full access is enabled. The security access required for the common SCSI command set of direct access devices is provided in Table 7. As SCSI standard commands are added the security shall be applied appropriately.

Table 7: SCSI commands for direct access devices, Security Access

SCSI Command	Command Code	Security Access
FORMAT UNIT	04h	FULL
INQUIRY	12h	RESTRICTED
LOG SELECT	4Ch	FULL
LOG SENSE	4Dh	RESTRICTED
MODE SELECT(06)	15h	FULL
MODE SELECT(10)	55h	FULL
MODE SENSE(06)	1Ah	RESTRICTED
MODE SENSE(10)	5Ah	RESTRICTED
PERSISTENT RESERVE IN	5Eh	FULL
PERSISTENT RESERVE OUT	5Fh	FULL
READ BUFFER	3Ch	RESTRICTED
READ CAPACITY	25h	RESTRICTED
READ DEFECT DATA(10)	37h	RESTRICTED
READ(06)	08h	RESTRICTED
READ(10)	28h	RESTRICTED
REASSIGN BLOCKS	07h	FULL
RECEIVE DIAGNOSTIC RESULTS	1Ch	FULL
RELEASE(06)	17h	FULL
RELEASE(10)	57h	FULL
REPORT LUNS	A0h	RESTRICTED
REQUEST SENSE	03h	RESTRICTED
RESERVE(06)	16h	FULL
RESERVE(10)	56h	FULL
REZERO UNIT	01h	RESTRICTED
SEEK(06)	0Bh	RESTRICTED
SEEK(10)	2Bh	RESTRICTED
SEND DIAGNOSTIC	1Dh	FULL
STOP START UNIT	1Bh	FULL
SYNCHRONIZE CACHE	35h	FULL
TEST UNIT READY	00h	RESTRICTED
VERIFY	13h	RESTRICTED
VERIFY(10)	2Fh	RESTRICTED
WRITE AND VERIFY(10)	2Eh	FULL
WRITE BUFFER	3Bh	LIMITED(1)
WRITE LONG	3Fh	FULL
WRITE(06)	0Ah	FULL
WRITE(10)	2Ah	FULL

Note 1: To support download of microcode and for link validation, the WRITE BUFFER command requires Limited access. The end device is responsible for ensuring that any download operation performed is validated with proper vendor, model and checksum associations.

6.11.2 Input

A CSMI_SAS_SSP_PASSTHRU_BUFFER data structure with the following initialized members;

- IoctlHeader, see IOCTL_HEADER for the platform specific initialization.
- Parameters. Initialized to 0's.
- Status. Initialized to 0's.
- Parameters.bPhyIdentifier. Contains the phy identifier that specifies which phy shall be used to issue the request. The value must be in the range of 0 to 254 (0 to FEh) or be the directive **CSMI_SAS_USE_PORT_IDENTIFIER**. The driver may generate an error due to the phy identifier for the following reasons:
  - The phy identifier cannot be selected because the driver does not support sending SMP requests to a phy identifier. The driver may support sending SMP requests to a bPortIdentifier only. The IoctlHeader.ReturnCode would be **CSMI_SAS_PHY_CANNOT_BE_SELECTED**.
  - The phy identifier is out of range of valid phys. The IoctlHeader.ReturnCode would be **CSMI_SAS_PHY_DOES_NOT_EXIST**.
  - The phy identifier cannot be associated with bPortIdentifier. If bPhyIdentifier is intended to reference the phy and the bPortIdentifier value is not **CSMI_SAS_IGNORE_PORT**, then the bPhyIdentifier and bPortIdentifier must have the proper association. The IoctlHeader.ReturnCode would be **CSMI_SAS_PHY_DOES_NOT_MATCH_PORT**.
  - The phy identifier has a value of **CSMI_SAS_USE_PORT_IDENTIFIER** and the bPortIdentifier has a value of **CSMI_SAS_IGNORE_PORT.** The driver cannot determine where to send the SMP request. Either bPhyIdentifier shall reference a valid phy or bPortIdentifier must reference a valid port. The IoctlHeader.ReturnCode would be **CSMI_SAS_SELECT_PHY_OR_PORT**.
- Parameters.bPortIdentifier. Contains the port identifier that specifies which port shall be used to issue the request. The value must be in the range of 0 to 254 (0 to FEh) or be the directive **CSMI_SAS_IGNORE_PORT**. The driver may generate an error due to the port identifier for the following reasons:
  - The port identifier cannot be selected because the driver does not support sending SMP requests to a port identifier. The driver may support sending SMP requests to a bPhyIdentifier only. The IoctlHeader.ReturnCode would be **CSMI_SAS_PORT_CANNOT_BE_SELECTED**.
  - The port identifier is out of range of valid ports. The IoctlHeader.ReturnCode would be **CSMI_SAS_PORT_DOES_NOT_EXIST**.
  - The port identifier cannot be associated with bPhyIdentifier. If bPortIdentifier is intended to reference the port and the bPhyIdentifier value is not **CSMI_SAS_USE_PORT_IDENTIFIER**, then the bPortIdentifier and bPhyIdentifier must have the proper association. The IoctlHeader.ReturnCode would be **CSMI_SAS_PHY_DOES_NOT_MATCH_PORT**.
  - The port identifier has a value of **CSMI_SAS_IGNORE_PORT** and the bPhyIdentifier has a value of **CSMI_SAS_USE_PORT_IDENTIFIER.** The driver cannot determine where to send the SMP request. Either bPhyIdentifier shall reference a valid phy or bPortIdentifier must reference a valid port. The IoctlHeader.ReturnCode would be **CSMI_SAS_SELECT_PHY_OR_PORT**.

- Parameters.bConnectionRate. Contains the connection rate directive for the driver connection manager. Shall be one of the following:
  - **CSMI_SAS_LINK_RATE_NEGOTIATED**, indicates that the connection shall be opened at the highest allowable negotiated rate for the destination device. The resulting rate will be the lowest common denominator of link rates along a connection pathway.
  - **CSMI_SAS_LINK_RATE_1_5_GBPS**, indicates that the connection shall be attempted at 1.5 Gb/s.
  - **CSMI_SAS_LINK_RATE_3_0_GBPS**, indicates that the connection shall be attempted at 3.0 Gb/s. This connection rate may not succeed if an intermediate link is less than 3.0 Gb/s.
- Parameters.bDestinationSASAddress. Contains the SAS address of the destination device in MSB order.
- Parameters.bLun. Contains the LUN of the target physical device to address. Equivalent to the SSP Logical Unit Number in an SSP information unit.
- Parameters.bCDBLength. Contains the length of the CDB defined.
- Parameters.bAdditionalCDBLength. Contains the length of valid dwords in bAdditionalCDB. Must be in the range of 0 to 6.
- Parameters.bCDB. Contains the CDB bytes for the specific SCSI command to send.
- Parameters.uFlags. Contains the directive that tells the SSP link and transport layers whether the command is expected to send or receive data. Shall be one or more of the following:
  - **CSMI_SAS_SSP_READ**, indicates that the data transfer will be from the destination device.
  - **CSMI_SAS_SSP_WRITE**, indicates that the data transfer will be to the destination device.
  - **CSMI_SAS_SSP_UNSPECIFIED**, indicates that there will be no data transfer, or the data transfer direction is unknown and any data received until the ITL nexus is completed shall be retained.
  - **CSMI_SAS_SSP_TASK_ATTRIBUTE_SIMPLE,** indicates that the Task attribute for the SSP command information unit shall be set to SIMPLE.
  - **CSMI_SAS_SSP_TASK_ATTRIBUTE_HEAD_OF_QUEUE,** indicates that the Task attribute for the SSP command information unit shall be set to HEAD_OF_QUEUE.
  - **CSMI_SAS_SSP_TASK_ATTRIBUTE_ORDERED,** indicates that the Task attribute for the SSP command information unit shall be set to ORDERED.
  - **CSMI_SAS_SSP_TASK_ATTRIBUTE_ACA,** indicates that the Task attribute for the SSP command information unit shall be set to ACA.

  Note: Only one of the Task attribute flags may be included in uFlags. If more than one is specified, then all are ignored and the result will be the equivalent of choosing CSMI_SAS_SSP_TASK_ATTRIBUTE_SIMPLE.
- Parameters.bAdditionalCDB. Contains any additional CDB bytes.
- Parameters.uDataLength. Contains the length of bDataBuffer in LSB order.
- Status. Initialized to 0's.
- bDataBuffer. Contains any data that is being written to the device or will provide a memory space for any data that is being read from the device.

6.11.3 Output

A CSMI_SAS_SSP_PASSTHRU_BUFFER data structure with the returned data;

- IoctlHeader.ReturnCode. See Return Codes.
- Parameters.bPhyIdentifier. Same as input.
- Parameters.bPortIdentifier. Same as input.
- Parameters.bConnectionRate. Same as input.

- Parameters.bDestinationSASAddress. Same as input.
- Parameters.bLun. Same as input.
- Parameters.bCDBLength. Same as input.
- Parameters.bAdditionalCDBLength. Same as input.
- Parameters.bCDB. Same as input.
- Parameters.uFlags. Same as input.
- Parameters.bAdditionalCDB. Same as input.
- Parameters.uDataLength. Same as input.
- Status. Contains the SSP status structure for the SSP command.
- Status.bConnectionStatus. Contains the results of the open connection attempt. Shall be one of the following:
  - **CSMI_SAS_OPEN_ACCEPT**, indicate the connection was successful and the request was submitted.
  - **CSMI_SAS_OPEN_REJECT_BAD_DESTINATION**, indicates the destination address was not found because the destination address was improper, no request was submitted.
  - **CSMI_SAS_OPEN_REJECT_RATE_NOT_SUPPORTED**, indicates the requested link rate could not be used, no request was submitted.
  - **CSMI_SAS_OPEN_REJECT_NO_DESTINATION**, indicates the destination address was not found, no request was submitted.
  - **CSMI_SAS_OPEN_REJECT_PATHWAY_BLOCKED**, see SAS specification, no request was submitted.
  - **CSMI_SAS_OPEN_REJECT_PROTOCOL_NOT_SUPPORTED**, indicates the destination device does not support the protocol requested, no request was submitted.
  - **CSMI_SAS_OPEN_REJECT_RESERVE_ABANDON**, see SAS specification, no request was submitted.
  - **CSMI_SAS_OPEN_REJECT_RESERVE_CONTINUE**, see SAS specification, no request was submitted.
  - **CSMI_SAS_OPEN_REJECT_RESERVE_INITIALIZE**, see SAS specification, no request was submitted.
  - **CSMI_SAS_OPEN_REJECT_RESERVE_STOP**, see SAS specification, no request was submitted.
  - **CSMI_SAS_OPEN_REJECT_RETRY**, see SAS specification, no request was submitted.
  - **CSMI_SAS_OPEN_REJECT_STP_RESOURCES_BUSY**, see SAS specification, no request was submitted.
  - **CSMI_SAS_OPEN_REJECT_WRONG_DESTINATION**, see SAS specification, no request was submitted.
- Status.bDataPresent. Contains the directives that indicate what has been returned in bResponse. Shall be one of the following:
  - **CSMI_SAS_SSP_NO_DATA_PRESENT**, see SCSI specification.
  - **CSMI_SAS_SSP_RESPONSE_DATA_PRESENT**, see SCSI specification.
  - **CSMI_SAS_SSP_SENSE_DATA_PRESENT**, see SCSI specification.
- Status.bStatus. Contains the SCSI status code.
- Status.bResponseLength. Contains the number of valid bytes in the bResponse field in MSB order.
- Status.bResponse. Contains the response bytes in MSB order. The interpretation of the data depends on the directive in the bDataPresent field.
- Status.uDataBytes. Contains the number of valid bytes in bDataBuffer, in LSB order.
- bDataBuffer. Contains any data that is being written to the device or contains any data that has been read from the device.

6.11.4 Structure Definitions

6.11.4.1 CSMI_SAS_SSP_PASSTHRU

```
typedef struct _CSMI_SAS_SSP_PASSTHRU {
  __u8  bPhyIdentifier;
  __u8  bPortIdentifier;
  __u8  bConnectionRate;
  __u8  bReserved;
  __u8  bDestinationSASAddress[8];
  __u8  bLun[8];
  __u8  bCDBLength;
  __u8  bAdditionalCDBLength;
  __u8  bReserved2[2];
  __u8  bCDB[16];
  __u32 uFlags;
  __u8  bAdditionalCDB[24];
  __u32 uDataLength;
} CSMI_SAS_SSP_PASSTHRU,
  *PCSMI_SAS_SSP_PASSTHRU;
```

6.11.4.2 CSMI_SAS_SSP_PASSTHRU_STATUS

```
typedef struct _CSMI_SAS_SSP_PASSTHRU_STATUS {
  __u8  bConnectionStatus;
  __u8  bReserved[3];
  __u8  bDataPresent;
  __u8  bStatus;
  __u8  bReponseLength[2];
  __u8  bResponse[256];
  __u32 uDataBytes;
} CSMI_SAS_SSP_PASSTHRU_STATUS,
  *PCSMI_SAS_SSP_PASSTHRU_STATUS;
```

6.11.4.3 CSMI_SAS_SSP_PASSTHRU_BUFFER

```
typedef struct _CSMI_SAS_SSP_PASSTHRU_BUFFER {
  IOCTL_HEADER IoctlHeader;
  CSMI_SAS_SSP_PASSTHRU Parameters;
  CSMI_SAS_SSP_PASSTHRU_STATUS Status;
  __u8  bDataBuffer[1];
} CSMI_SAS_SSP_PASSTHRU_BUFFER,
  *PCSMI_SAS_SSP_PASSTHRU_BUFFER;
```

6.12 CC_CSMI_SAS_STP_PASSTHRU

(SATA, SAS)

6.12.1 Behavior

This CSMI functional behavior provides a method of sending generic STP or SATA commands to a specific SAS address. Any driver that implements this specification and supports the STP or SATA protocols must support this behavior; otherwise the driver may respond to this control code with a generic IO error (see Submitting Control Codes). A driver that emulates STP or SATA devices as SCSI devices may require that commands directed to STP or SATA devices be directed to the SCSI link and transport layer. To facilitate sending generic STP or SATA commands with that restriction an alternative mechanism using a special SCSI command to wrap SATA commands may be provided (see **Error! Reference source not found.**). A driver can direct the upper level application to use the alternative method by posting IoctrlHeader.ReturnCode = CSMI_SAS_SCSI_EMULATION.

6.12.1.1 Security

A driver may post IoctrlHeader.ReturnCode = CSMI_SAS_SCSI_WRITE_ATTEMPTED, if the security level is insufficient to complete the command requested (see Security and Enabling Features). Vendor unique commands are only allowed if Full access is enabled. The security access required for the drive ATA command set is provided in Table 8. As ATA standard commands or devices are added the security shall be applied appropriately.

Table 8: ATA commands, Security Access

ATA Command	Command Code	Security Access
CHECK POWER MODE	E5H	RESTRICTED
DEVICE CONFIGURATION	B1H	FULL
DEVICE RESET	08H	FULL
DOWNLOAD MICROCODE	92H	LIMITED (1)
EXECUTE DEVICE DIAGNOSTIC	90H	RESTRICTED
FLUSH CACHE	E7H	RESTRICTED
FLUSH CACHE EXT	EAH	RESTRICTED
IDENTIFY DEVICE	ECH	RESTRICTED
IDENTIFY PACKET DEVICE	A1H	RESTRICTED
IDLE	E3H	FULL
IDLE IMMEDIATE	E1H	FULL
NOP	00H	RESTRICTED
PACKET	A0H	MIXED (2)
READ BUFFER	E4H	RESTRICTED
READ DMA	C8H	RESTRICTED
READ DMA EXT	25H	RESTRICTED
READ DMA QUEUED	C7H	RESTRICTED
READ DMA QUEUED EXT	26H	RESTRICTED
READ LOG EXT	2FH	RESTRICTED
READ MULTIPLE	C4H	RESTRICTED
READ MULTIPLE EXT	29H	RESTRICTED
READ NATIVE MAX ADDRESS	F8H	RESTRICTED
READ NATIVE MAX ADDRESS EXT	27H	RESTRICTED

(continued)

Table 5: ATA commands, Security Access, *(continued)*

ATA Command	Command Code	Security Access
READ SECTOR(S)	20H	RESTRICTED
READ SECTOR(S) EXT	24H	RESTRICTED
READ VERIFY SECTOR(S)	40H	RESTRICTED
READ VERIFY SECTOR(S) EXT	42H	RESTRICTED
SECURITY DISABLE PASSWORD	F6H	FULL
SECURITY ERASE PREPARE	F3H	FULL
SECURITY ERASE UNIT	F4H	FULL
SECURITY FREEZE LOCK	F5H	FULL
SECURITY SET PASSWORD	F1H	FULL
SECURITY UNLOCK	F2H	FULL
SEEK	70H	RESTRICTED
SERVICE	A2H	FULL
SET FEATURES	EFH	FULL
SET MAX ADDRESS	F9H	FULL
SET MAX ADDRESS EXT	37H	FULL
SET MULTIPLE MODE	C6H	FULL
SLEEP	E6H	FULL
SMART	B0H	RESTRICTED
STANDBY	E2H	FULL
STANDBY IMMEDIATE	E0H	FULL
WRITE BUFFER	E8H	LIMITED (3)
WRITE DMA	CAH	FULL
WRITE DMA EXT	35H	FULL
WRITE DMA FUA EXT	3DH	FULL
WRITE DMA QUEUED	CCH	FULL
WRITE DMA QUEUED EXT	36H	FULL
WRITE DMA QUEUED FUA EXT	3EH	FULL
WRITE LOG EXT	3FH	FULL
WRITE MULTIPLE	C5H	FULL
WRITE MULTIPLE EXT	39H	FULL
WRITE MULTIPLE FUA EXT	CEH	FULL
WRITE SECTOR(S)	30H	FULL
WRITE SECTOR(S) EXT	34H	FULL
VENDOR SPECIFIC	ALL OTHERS	FULL

Note 1: To support download of microcode, the DOWNLOAD MICROCODE command requires Limited access. The end device is responsible for ensuring that any download operation performed is validated with proper vendor, model and checksum associations.
Note 2: See Table 7, for SCSI commands.
Note 3: To support link verification, the WRITE BUFFER command requires Limited access.

6.12.2 Input

A CSMI_SAS_STP_PASSTHRU_BUFFER data structure with the following initialized members;

- IoctlHeader, see IOCTL_HEADER for the platform specific initialization.
- Parameters, initialized to 0's.
- Parameters.bPhyIdentifier. Contains the phy identifier that specifies which phy shall be used to issue the request. The value must be in the range of 0 to 254 (0 to FEh) or be the directive **CSMI_SAS_USE_PORT_IDENTIFIER**. The driver may generate an error due to the phy identifier for the following reasons:
  - The phy identifier cannot be selected because the driver does not support sending SMP requests to a phy identifier. The driver may support sending SMP requests to a bPortIdentifier only. The IoctlHeader.ReturnCode would be **CSMI_SAS_PHY_CANNOT_BE_SELECTED**.
  - The phy identifier is out of range of valid phys. The IoctlHeader.ReturnCode would be **CSMI_SAS_PHY_DOES_NOT_EXIST**.
  - The phy identifier cannot be associated with bPortIdentifier. If bPhyIdentifier is intended to reference the phy and the bPortIdentifier value is not **CSMI_SAS_IGNORE_PORT**, then the bPhyIdentifier and bPortIdentifier must have the proper association. The IoctlHeader.ReturnCode would be **CSMI_SAS_PHY_DOES_NOT_MATCH_PORT**.
  - The phy identifier has a value of **CSMI_SAS_USE_PORT_IDENTIFIER** and the bPortIdentifier has a value of **CSMI_SAS_IGNORE_PORT.** The driver cannot determine where to send the SMP request. Either bPhyIdentifier shall reference a valid phy or bPortIdentifier must reference a valid port. The IoctlHeader.ReturnCode would be **CSMI_SAS_SELECT_PHY_OR_PORT**.
- Parameters.bPortIdentifier. Contains the port identifier that specifies which port shall be used to issue the request. The value must be in the range of 0 to 254 (0 to FEh) or be the directive **CSMI_SAS_IGNORE_PORT**. The driver may generate an error due to the port identifier for the following reasons:
  - The port identifier cannot be selected because the driver does not support sending SMP requests to a port identifier. The driver may support sending SMP requests to a bPhyIdentifier only. The IoctlHeader.ReturnCode would be **CSMI_SAS_PORT_CANNOT_BE_SELECTED**.
  - The port identifier is out of range of valid ports. The IoctlHeader.ReturnCode would be **CSMI_SAS_PORT_DOES_NOT_EXIST**.
  - The port identifier cannot be associated with bPhyIdentifier. If bPortIdentifier is intended to reference the port and the bPhyIdentifier value is not **CSMI_SAS_USE_PORT_IDENTIFIER**, then the bPortIdentifier and bPhyIdentifier must have the proper association. The IoctlHeader.ReturnCode would be **CSMI_SAS_PHY_DOES_NOT_MATCH_PORT**.
  - The port identifier has a value of **CSMI_SAS_IGNORE_PORT** and the bPhyIdentifier has a value of **CSMI_SAS_USE_PORT_IDENTIFIER.** The driver cannot determine where to send the SMP request. Either bPhyIdentifier shall reference a valid phy or bPortIdentifier must reference a valid port. The IoctlHeader.ReturnCode would be **CSMI_SAS_SELECT_PHY_OR_PORT**.

- Parameters.bConnectionRate. Contains the connection rate directive for the driver connection manager. Shall be one of the following:
  - **CSMI_SAS_LINK_RATE_NEGOTIATED**, indicates that the connection shall be opened at the highest allowable negotiated rate for the destination device. The resulting rate will be the lowest common denominator of link rates along a connection pathway.
  - **CSMI_SAS_LINK_RATE_1_5_GBPS**, indicates that the connection shall be attempted at 1.5 Gb/s.
  - **CSMI_SAS_LINK_RATE_3_0_GBPS**, indicates that the connection shall be attempted at 3.0 Gb/s. This connection rate may not succeed if an intermediate link is less than 3.0 Gb/s.
- Parameters.bDestinationSASAddress. Contains the SAS address of the destination device in MSB order.
- Parameters.bCommandFIS. Contains the SATA command FIS (27h). See the SATA specification.
- Parameters.uFlags. Contains the directive that tells the STP or SATA link and transport layers whether the command is expected to send or receive data. Shall be one or more of the following:
  - **CSMI_SAS_STP_READ**, indicates that the data transfer will be from the destination device.
  - **CSMI_SAS_STP_WRITE**, indicates that the data transfer will be to the destination device.
  - **CSMI_SAS_STP_UNSPECIFIED**, indicates that there will be no data transfer, or the data transfer direction is unknown and any data received shall be retained.
  - **CSMI_SAS_STP_PIO**, indicates the command follows the SATA PIO state machine for completion.
  - **CSMI_SAS_STP_DMA**, indicates the command follows the SATA DMA state machine for completion.
  - **CSMI_SAS_STP_PACKET**, indicates the command follows the SATA packet state machine for completion.
  - **CSMI_SAS_STP_DMA_QUEUED**, indicates the command follows the SATA DMA queued state machine for completion.
  - **CSMI_SAS_STP_EXECUTE_DIAG**, indicates the command follows the SATA execute diagnostic state machine for completion.
  - **CSMI_SAS_STP_RESET_DEVICE**, indicates that a soft reset is being performed.
- bDataBuffer. Contains any data that is being written to the device or will provide a memory space for any data that is being read from the device.

6.12.3 Output

A CSMI_SAS_STP_PASSTHRU_BUFFER data structure with the returned data;

- IoctlHeader.ReturnCode. See Return Codes.
- Parameters.bPhyIdentifier. Same as input.
- Parameters.bPortIdentifier. Same as input.
- Parameters.bConnectionRate. Same as input.
- Parameters.bDestinationSASAddress. Same as input.
- Status. Contains the STP status structure for the STP or SATA command.
- Status.bConnectionStatus. Contains the results of the open connection attempt. Shall be one of the following:
  - **CSMI_SAS_OPEN_ACCEPT**, indicate the connection was successful and the request was submitted.
  - **CSMI_SAS_OPEN_REJECT_BAD_DESTINATION**, indicates the destination address was not found because the destination address was improper, no request was submitted.

- **CSMI_SAS_OPEN_REJECT_RATE_NOT_SUPPORTED**, indicates the requested link rate could not be used, no request was submitted.
- **CSMI_SAS_OPEN_REJECT_NO_DESTINATION**, indicates the destination address was not found, no request was submitted.
- **CSMI_SAS_OPEN_REJECT_PATHWAY_BLOCKED**, see SAS specification, no request was submitted.
- **CSMI_SAS_OPEN_REJECT_PROTOCOL_NOT_SUPPORTED**, indicates the destination device does not support the protocol requested, no request was submitted.
- **CSMI_SAS_OPEN_REJECT_RESERVE_ABANDON**, see SAS specification, no request was submitted.
- **CSMI_SAS_OPEN_REJECT_RESERVE_CONTINUE**, see SAS specification, no request was submitted.
- **CSMI_SAS_OPEN_REJECT_RESERVE_INITIALIZE**, see SAS specification, no request was submitted.
- **CSMI_SAS_OPEN_REJECT_RESERVE_STOP**, see SAS specification, no request was submitted.
- **CSMI_SAS_OPEN_REJECT_RETRY**, see SAS specification, no request was submitted.
- **CSMI_SAS_OPEN_REJECT_STP_RESOURCES_BUSY**, see SAS specification, no request was submitted.
- **CSMI_SAS_OPEN_REJECT_WRONG_DESTINATION**, see SAS specification, no request was submitted.

- Status.bStatusFIS. Contains the SATA status FIS (34h). See SATA specification.
- Status.uSCR. Contains the status control registers. The content of uSCR will be updated at the completion of the command. Register level polling is not intended.
- Status.uDataBytes. Contains the number of valid bytes in bDataBuffer, in LSB order.
- bDataBuffer. Contains any data that is being written to the device or contains any data that has been read from the device.

6.12.4 Structure Definitions

6.12.4.1 CSMI_SAS_STP_PASSTHRU

```
typedef struct _CSMI_SAS_STP_PASSTHRU {
  __u8  bPhyIdentifier;
  __u8  bPortIdentifier;
  __u8  bConnectionRate;
  __u8  bReserved;
  __u8  bDestinationSASAddress[8];
  __u8  bReserved2[4];
  __u8  bCommandFIS[20];
  __u32 uFlags;
  __u32 uDataLength;
} CSMI_SAS_STP_PASSTHRU,
  *PCSMI_SAS_STP_PASSTHRU;
```

6.12.4.2 CSMI_SAS_STP_PASSTHRU_STATUS

```
typedef struct _CSMI_SAS_STP_PASSTHRU_STATUS {
  __u8  bConnectionStatus;
  __u8  bReserved[3];
  __u8  bStatusFIS[20];
  __u32 uSCR[16];
  __u32 uDataBytes;
} CSMI_SAS_STP_PASSTHRU_STATUS,
  *PCSMI_SAS_STP_PASSTHRU_STATUS;
```

6.12.4.3 CSMI_SAS_STP_PASSTHRU_BUFFER

```
typedef struct _CSMI_SAS_STP_PASSTHRU_BUFFER {
  IOCTL_HEADER IoctlHeader;
  CSMI_SAS_STP_PASSTHRU Parameters;
  CSMI_SAS_STP_PASSTHRU_STATUS Status;
  __u8  bDataBuffer[1];
} CSMI_SAS_STP_PASSTHRU_BUFFER,
  *PCSMI_SAS_STP_PASSTHRU_BUFFER;
```

6.13 CC_CSMI_SAS_GET_SATA_SIGNATURE

(SATA, SAS)

6.13.1 Behavior

This CSMI functional behavior provides a method of obtaining the initial SATA signature (the initial Register Device to Host FIS) from a directly attached SATA device. The signature may be used to identify whether a SATA device supports the PACKET command set or whether it is a unique SATA device (like a port multiplier). Any driver that implements this specification and supports directly attached SATA devices must support this behavior; otherwise the driver may respond to this control code with a generic IO error (see Submitting Control Codes).

6.13.2 Input

A CSMI_SAS_SATA_SIGNATURE_BUFFER data structure with the following initialized members;

- IoctlHeader, see IOCTL_HEADER for the platform specific initialization.
- Signature, initialized to 0's.
- Signature.bPhyIdentifier. Contains the phy identifier that is being queried for a SATA signature. The driver may generate an error due to the phy identifier for the following reasons:
  - The phy does not have a SATA device directly attached or the phy has not completed the link reset sequence. The IoctlHeader.ReturnCode would be **CSMI_SAS_NO_SATA_DEVICE**.
  - The phy has not received the initial register device to host FIS from the SATA device. The IoctlHeader.ReturnCode would be **CSMI_SAS_NO_SATA_SIGNATURE**.
  - The phy does not exist. The IoctlHeader.ReturnCode would be **CSMI_SAS_PHY_DOES_NOT_EXIST**.

6.13.3 Output

A CSMI_SAS_SATA_SIGNATURE_BUFFER data structure with the returned data;

- IoctlHeader.ReturnCode. See Return Codes.
- Signature.bPhyIdentifier. Same as input.
- Signature.bSignatureFIS. Contains the initial register device to host FIS (34h) from the SATA device. Only the signature bytes are required to be valid, the remainder of the FIS may be 0 filled. If the FIS type is valid, (i.e. 34h) then the entire FIS is assumed to be valid (i.e. as returned from the device).

6.13.4 Structure Definitions

6.13.4.1 CSMI_SAS_SATA_SIGNATURE

```
typedef struct _CSMI_SAS_SATA_SIGNATURE {
  __u8  bPhyIdentifier;
  __u8  bReserved[3];
  __u8  bSignatureFIS[20];
} CSMI_SAS_SATA_SIGNATURE,
  *PCSMI_SAS_SATA_SIGNATURE;
```

6.13.4.2 CSMI_SAS_SATA_SIGNATURE_BUFFER

```
typedef struct _CSMI_SAS_SATA_SIGNATURE_BUFFER {
  IOCTL_HEADER IoctlHeader;
  CSMI_SAS_SATA_SIGNATURE Signature;
} CSMI_SAS_SATA_SIGNATURE_BUFFER,
  *PCSMI_SAS_SATA_SIGNATURE_BUFFER;
```

6.14 CC_CSMI_SAS_GET_SCSI_ADDRESS (SAS)

6.14.1 Behavior

This CSMI functional behavior provides a method of obtaining the OS specific platform address for a SAS address. Any driver that implements this specification must support this behavior. The driver may generate an error on this request for the following reasons:

- The SAS address is to an expander device. The IoctlHeader.ReturnCode would be **CSMI_SAS_NOT_AN_END_DEVICE**.
- The SAS address does not have an associated OS specific address. The IoctlHeader.ReturnCode would be **CSMI_SAS_NO_SCSI_ADDRESS**.

6.14.2 Input

A CSMI_SAS_GET_SCSI_ADDRESS_BUFFER data structure with the following initialized members;

- IoctlHeader. See IOCTL_HEADER for the platform specific initialization.
- bSASAddress. Contains the SAS address of the device, in MSB order.
- bSASLun. Contains the SAS Lun of the device, in MSB order.
- bHostIndex. Initialized to 0.
- bPathId. Initialized to 0.
- bTargetId. Initialized to 0.
- bLun. Initialized to 0.

6.14.3 Output

A CSMI_SAS_GET_SCSI_ADDRESS_BUFFER data structure with the returned data;

- IoctlHeader.ReturnCode. See Return Codes.
- bSASAddress. Same as input.
- bSASLun. Same as input.
- bHostIndex. Contains the enumerated index of the driver instance (for example, the n value in “SCSIn” under Windows). An FFh indicates the value is invalid.
- bPathId. Contains the path identifier for the device.
- bTargetId. Contains the target identifier for the device.
- bLun. Contains the Lun identifier for the device.

6.14.4 Structure Definitions

6.14.4.1 CSMI_SAS_GET_SCSI_ADDRESS_BUFFER

```
typedef struct _CSMI_SAS_GET_SCSI_ADDRESS_BUFFER {
  IOCTL_HEADER IoctlHeader;
  __u8  bSASAddress[8];
  __u8  bSASLun[8];
  __u8  bHostIndex;
  __u8  bPathId;
  __u8  bTargetId;
  __u8  bLun;
} CSMI_SAS_GET_SCSI_ADDRESS_BUFFER,
  *PCSMI_SAS_GET_SCSI_ADDRESS_BUFFER;
```

6.15 CC_CSMI_SAS_GET_DEVICE_ADDRESS (SAS)

6.15.1 Behavior

This CSMI functional behavior provides a method of obtaining the SAS address of a device from an OS specific platform address. Any driver that implements this specification must support this behavior. The driver may generate an error on this request for the following reasons:

- The OS specific platform address does not have a SAS address. The IoctrlHeader.ReturnCode would be **CSMI_SAS_NO_DEVICE_ADDRESS**.

6.15.2 Input

A CSMI_SAS_GET_DEVICE_ADDRESS_BUFFER data structure with the following initialized members;

- IoctlHeader. See IOCTL_HEADER for the platform specific initialization.
- bHostIndex. Contains the enumerated index of the driver instance (for example, the n value in “SCSIn” under Windows). An FFh indicates the value is invalid.
- bPathId. Contains the path identifier for the device.
- bTargetId. Contains the target identifier for the device.
- bLun. Contains the Lun identifier for the device.
- bSASAddress. Initialized to 0.
- bSASLun. Initialized to 0.

6.15.3 Output

A CSMI_SAS_GET_DEVICE_ADDRESS_BUFFER data structure with the returned data;

- IoctlHeader.ReturnCode. See Return Codes.
- bHostIndex. Same as input.
- bPathId. Same as input.
- bTargetId. Same as input.
- bLun. Same as input.
- bSASAddress. Contains the SAS address of the device, in MSB order.
- bSASLun. Contains the SAS Lun of the device, in MSB order.

6.15.4 Structure Definitions

6.15.4.1 CSMI_SAS_GET_DEVICE_ADDRESS_BUFFER

```
typedef struct _CSMI_SAS_GET_DEVICE_ADDRESS_BUFFER {
  IOCTL_HEADER IoctlHeader;
  __u8  bHostIndex;
  __u8  bPathId;
  __u8  bTargetId;
  __u8  bLun;
  __u8  bSASAddress[8];
  __u8  bSASLun[8];
} CSMI_SAS_GET_DEVICE_ADDRESS_BUFFER,
  *PCSMI_SAS_GET_DEVICE_ADDRESS_BUFFER;
```

6.16 CC_CSMI_SAS_TASK_MANAGEMENT (SAS)

6.16.1 Behavior

This CSMI functional behavior provides a method of sending a Hard Reset Sequence or Task Information Unit to the specified OS specific platform address.

6.16.1.1 Security

A driver may post IoctlHeader.ReturnCode = CSMI_SAS_SCSI_WRITE_ATTEMPTED, if the security level is insufficient to complete the command requested (see Security and Enabling Features). The security level required for issuing any task management or reset function is; FULL.

6.16.2 Input

A CSMI_SAS_SSP_TASK_IU_BUFFER data structure with the following initialized members;

- IoctlHeader. See IOCTL_HEADER for the platform specific initialization.
- Parameters.bHostIndex. Contains the enumerated index of the driver instance (for example, the n value in “SCSIn” under Windows). An FFh indicates the value is invalid.
- Parameters.bPathId. Contains the path identifier for the device.
- Parameters.bTargetId. Contains the target identifier for the device.
- Parameters.bLun. Contains the Lun identifier for the device.
- Parameters.uFlags. Contains one or more of the following;
  - CSMI_SAS_TASK_IU. When set the uQueueTag and bTaskManagementFunction fields must contain the information to be provided in a Task information unit. If set, the CSMI_SAS_HARD_RESET_SEQUENCE may not be set.
  - CSMI_SAS_HARD_RESET_SEQUENCE. When set, will issue a Hard reset sequence as defined in SAS to the OS specific platform address. There shall no delay inserted by the driver after issuing the Hard reset sequence. If set, the CSMI_SAS_TASK_IU may not be set.
  - CSMI_SAS_SUPPRESS_RESULT. Optional flag when set, the OS low-level driver will suppress reporting the task management event to the upper level driver.
- Parameters.uQueueTag. Contains the OS specific queue tag value that will identify the Task information unit tag.
- Parameters.bTaskManagementFunction. Contains one of the following SSP task management functions (refer to the SAS specification for details on expected behavior);
  - CSMI_SAS_SSP_ABORT_TASK
  - CSMI_SAS_SSP_ABORT_TASK_SET
  - CSMI_SAS_SSP_CLEAR_TASK_SET
  - CSMI_SAS_SSP_LOGICAL_UNIT_RESET
  - CSMI_SAS_SSP_CLEAR_ACA
  - CSMI_SAS_SSP_QUERY_TASK
- Parameters.uInformation. Contains application specific information for reason the Task management function is being sent. May be one of the following;
  - CSMI_SAS_SSP_TEST. Used to indicate the Task management request was sent as part of a general test requirement.
  - CSMI_SAS_SSP_EXCEEDED. Used to indicate the Task management request was sent to terminate an outstanding command that has exceeded a time limit.
  - CSMI_SAS_SSP_DEMAND. Used to indicate the Task management request was sent on demand from an application.

- CSMI_SAS_SSP_TRIGGER. Used to indicate the Task management request is being used as a trigger event by an application.
- Status. Initialized to 0's.

6.16.3 Output

A CSMI_SAS_SSP_TASK_IU_BUFFER data structure with the returned data;

- IoctlHeader.ReturnCode. See Return Codes.
- Parameters.bHostIndex. Same as input.
- Parameters.bPathId. Same as input.
- Parameters.bTargetId. Same as input.
- Parameters.bLun. Same as input.
- Parameters.uFlags. Same as input.
- Parameters.uQueueTag. Same as input.
- Parameters.bTaskManagementFunction. Same as input.
- Parameters.uInformation. Same as input.
- Status. Contains the SSP status structure for the SSP command.
- Status.bConnectionStatus. Contains the results of the open connection attempt. Shall be one of the following:
  - **CSMI_SAS_OPEN_ACCEPT**, indicate the connection was successful and the request was submitted.
  - **CSMI_SAS_OPEN_REJECT_BAD_DESTINATION**, indicates the destination address was not found because the destination address was improper, no request was submitted.
  - **CSMI_SAS_OPEN_REJECT_RATE_NOT_SUPPORTED**, indicates the requested link rate could not be used, no request was submitted.
  - **CSMI_SAS_OPEN_REJECT_NO_DESTINATION**, indicates the destination address was not found, no request was submitted.
  - **CSMI_SAS_OPEN_REJECT_PATHWAY_BLOCKED**, see SAS specification, no request was submitted.
  - **CSMI_SAS_OPEN_REJECT_PROTOCOL_NOT_SUPPORTED**, indicates the destination device does not support the protocol requested, no request was submitted.
  - **CSMI_SAS_OPEN_REJECT_RESERVE_ABANDON**, see SAS specification, no request was submitted.
  - **CSMI_SAS_OPEN_REJECT_RESERVE_CONTINUE**, see SAS specification, no request was submitted.
  - **CSMI_SAS_OPEN_REJECT_RESERVE_INITIALIZE**, see SAS specification, no request was submitted.
  - **CSMI_SAS_OPEN_REJECT_RESERVE_STOP**, see SAS specification, no request was submitted.
  - **CSMI_SAS_OPEN_REJECT_RETRY**, see SAS specification, no request was submitted.
  - **CSMI_SAS_OPEN_REJECT_STP_RESOURCES_BUSY**, see SAS specification, no request was submitted.
  - **CSMI_SAS_OPEN_REJECT_WRONG_DESTINATION**, see SAS specification, no request was submitted.
- Status.bDataPresent. Contains the directives that indicate what has been returned in bResponse. Shall be one of the following:
  - **CSMI_SAS_SSP_NO_DATA_PRESENT**, see SCSI specification.
  - **CSMI_SAS_SSP_RESPONSE_DATA_PRESENT**, see SCSI specification.
  - **CSMI_SAS_SSP_SENSE_DATA_PRESENT**, see SCSI specification.
- Status.bStatus. Contains the SCSI status code.
- Status.bResponseLength. Contains the number of valid bytes in the bResponse field in MSB order.

- Status.bResponse. Contains the response bytes in MSB order. The interpretation of the data depends on the directive in the bDataPresent field.
- Status.uDataBytes. Shall be 0.

6.16.4 Structure Definitions

6.16.4.1 CSMI_SAS_SSP_TASK_IU

```
typedef struct _CSMI_SAS_SSP_TASK_IU {
  __u8  bHostIndex;
  __u8  bPathId;
  __u8  bTargetId;
  __u8  bLun;
  __u32  uFlags;
  __u32  uQueueTag;
  __u32 uReserved;
  __u8 bTaskManagementFunction;
  __u8 bReserved[7];
  __u32 uInformation;
} CSMI_SAS_SSP_TASK_IU,
  *PCSMI_SAS_SSP_TASK_IU;
```

6.16.4.2 CSMI_SAS_SSP_TASK_IU_BUFFER

```
typedef struct _CSMI_SAS_SSP_TASK_IU_BUFFER {
   IOCTL_HEADER IoctlHeader;
   CSMI_SAS_SSP_TASK_IU Parameters;
   CSMI_SAS_SSP_PASSTHRU_STATUS Status;
} CSMI_SAS_SSP_TASK_IU_BUFFER,
  *PCSMI_SAS_SSP_TASK_IU_BUFFER;
```

6.17 CC_CSMI_SAS_PHY_CONTROL

(SAS) (Recommended)

6.17.1 Behavior

This CSMI functional behavior provides a method of determining and setting the phy characteristics of the controller. The phy control features include; low level reset control, SATA port selection control, phy signal control, and phy pattern generation. Since this behavior supports functions that are tightly coupled with hardware implementations, full support for every phy signal control is not required. If the hardware is capable of supporting a specific phy signal control then the associated function shall be supported.

6.17.1.1 Security

A driver may post IoctrlHeader.ReturnCode = CSMI_SAS_SCSI_WRITE_ATTEMPTED, if the security level is insufficient to complete the command requested (see Security and Enabling Features). The security level required for issuing the phy control function is; FULL.

6.17.1.2 Spinup Behavior Model

The CSMI functional behavior supports controls that may affect the spinup behavior of devices. The programming model used to define this behavior assumes that the device spinup window is global for the controller. This means that when end devices are directly connected to the controller across the controller phys there is only one window of opportunity to spinup a device. This prevents power supply overload conditions caused by multiple devices spinning up at the same time. The model further assumes that the spinup window is enabled by a token that is passed from phy to phy starting with phy 0 and wrapping around from the last phy back to phy 0. The spinup rate defined in this CSMI behavior is intended to specify the time the token wait before being passed to the next phy in the loop. As an example if a controller has 4 phys and the spinup rate is set for 3 seconds, then;

- At 0 seconds.
  - Phy 0 will output a NOTIFY(SPINUP) primitive for SAS or send a COMWAKE in response to a COMINIT for SATA.
  - Phy 1, will remain idle
  - Phy 2, will remain idle
  - Phy 3, will remain idle
- At 3 seconds
  - Phy 0, will remain idle
  - Phy 1 will output a NOTIFY(SPINUP) primitive for SAS or send a COMWAKE in response to a COMINIT for SATA.
  - Phy 2, will remain idle
  - Phy 3, will remain idle
- At 6 seconds
  - Phy 0, will remain idle
  - Phy 1, will remain idle
  - Phy 2 will output a NOTIFY(SPINUP) primitive for SAS or send a COMWAKE in response to a COMINIT for SATA.
  - Phy 3, will remain idle
- At 9 seconds
  - Phy 0, will remain idle
  - Phy 1, will remain idle
  - Phy 2, will remain idle
  - Phy 3 will output a NOTIFY(SPINUP) primitive for SAS or send a COMWAKE in response to a COMINIT for SATA.
- At 12 seconds, the 0 second behavior repeats and so on…

From this example a 3 second rate translates into a minimum 9 second waiting period for all devices to be given an opportunity to spinup.

A model that is not global in nature shall also be supported by this CSMI behavior, but may have a different interaction between the spinup rate provided and the actual device ready times.

6.17.1.3 Phy Signal Control Behavior Model

The CSMI functional behavior supports controls that may affect communication with end devices.

The controller driver may choose to limit the possible range of controls to ensure excessive voltages are not generated by the phy. The limiting is not required, but is highly recommended.

The programming model assumes that any signal level changes will occur when the phy is in an inactive state and will be followed by either a link reset sequence or a hard link reset sequence.

The pattern generation behavior assumes that the controller receivers will ignore any input from the end device (if any) and simply provide a constant stream of data based on the pattern requested.

If the controller has any active IO outstanding at the time a pattern generation behavior is requested an IoctlHeader.ReturnCode = CSMI_SAS_STATUS_FAILED is returned and the requested function is aborted.

6.17.2 Input

A CSMI_SAS_PHY_CONTROL_BUFFER data structure with the following initialized members;

- IoctlHeader. See IOCTL_HEADER for the platform specific initialization.
- uFunction. Contains the function to perform and may be one of the following:
  - CSMI_SAS_PC_LINK_RESET, indicates that the specified phy shall perform a link reset sequence. The phy identifier (see bPhyIdentifier) specifies which phy shall participate in this function. Depending on the remaining parameters in the structure one of the following behaviors is performed;
    - If the length of control (see bLengthOfControl), number of controls (see bNumberOfControls) and control structure (see Control) properly define one or more phy controls, then after going to the common mode state and prior to initiating the first COMRESET the phy control(s) shall be used to update the current phy settings.
    - If the length of control, number of controls, and control structure do not properly define one or more phy controls, then the IoctlHeader.ReturnCode = CSMI_SAS_STATUS_INVALID_PARAMETER is returned and link reset sequence is not performed.
    - If the length of control, number of controls and control structure are all 0 filled, then a link reset sequence is performed without altering the current phy settings.
  - CSMI_SAS_PC_HARD_RESET, indicates that the specified phy shall perform a hard link reset sequence. The phy identifier (see bPhyIdentifier) specifies which phy shall participate in this function. Depending on the remaining parameters in the structure one of the following behaviors is performed;
    - If the length of control (see bLengthOfControl), number of controls (see bNumberOfControls) and control structure (see Control) properly define one or more phy controls, then after going to the common mode state and prior to initiating the first COMRESET the phy control(s) shall be used to update the current phy settings.

        - If the length of control, number of controls, and control structure do not properly define one or more phy controls, then the IoctlHeader.ReturnCode = CSMI_SAS_STATUS_INVALID_PARAMETER is returned and link reset sequence is not performed.
        - If the length of control, number of controls and control structure are all 0 filled, then a hard link reset sequence is performed without altering the current phy settings.
    - CSMI_SAS_PC_PHY_DISABLE, indicates that the specified phy shall be disabled. The phy identifier (see bPhyIdentifier) specifies which phy shall participate in this function. The length of control (see bLengthOfControl) the number of controls (see bNumberOfControls) and control (see Control) structures shall all be 0 filled.
    - CSMI_SAS_PC_GET_PHY_SETTINGS, indicates that the necessary number of CSMI_SAS_PHY_CONTROL structures shall be updated to reflect the current phy settings for each control type (see bType) and rate (see bRate) supported. For example if the SAS controller supports SATA and SAS devices at 1.5 Gbps and 3.0 Gbps link rates, then 4 CSMI_SAS_PHY_CONTROL structures shall be returned. The order of the structures returned is not defined. The phy identifier (see bPhyIdentifier) specifies which phy shall participate in this function. The length of control (see bLengthOfControl) the number of controls (see bNumberOfControls) and control (see Control) structures shall all be 0 filled on input.
- bPhyIdentifier. Contains the phy identifier of the phy to control or query.
- usLengthOfControl. Contains the length of the phy control structure. If the length is required for the function and is incorrect then the IoctlHeader.ReturnCode = CSMI_SAS_STATUS_INVALID_PARAMETER is returned and the value will be updated to reflect the correct length on return.
- bNumberOfControls. Contains the number of CSMI_SAS_PHY_CONTROL elements in the Control array. If the number of controls is required for the function and is incorrect then the IoctlHeader.ReturnCode = CSMI_SAS_STATUS_INVALID_PARAMETER is returned and the value will be updated to reflect the correct number of controls on return.
- uLinkFlags. Contains flags that define basic link behavior and may be one or more of the following;
    - CSMI_SAS_PHY_ACTIVATE_CONTROL, indicates that the link behavior provided shall be performed. If this flag is not set, then the link behavior will not be modified during input or are not active during output.
    - CSMI_SAS_PHY_UPDATE_SPINUP_RATE, indicates that the spinup rate (see uSpinupRate) shall be used to alter the repetition rate of NOTIFY(SPINUP) primitives for SAS or the release interval of COMWAKE in response to a COMINIT for SATA. Note; the notify spinup rate may be global in nature across all phys, so the application must compensate for this by validating the resulting value by using the CSMI_SAS_PC_GET_PHY_SETTINGS after updating all phys. If this flag is not supported, then the IoctlHeader.ReturnCode = CSMI_SAS_STATUS_INVALID_PARAMETER is returned and no change occurs.
    - CSMI_SAS_PHY_AUTO_COMWAKE, indicates that there is no release interval for COMWAKE in response to a COMINIT for SATA. This means that a SATA drive will be released to spinup immediately. If set in conjunction with CSMI_SAS_PHY_UPDATE_SPINUP_RATE, then the IoctlHeader.ReturnCode = CSMI_SAS_STATUS_INVALID_PARAMETER is returned and no change occurs. If this flag is not supported, then the IoctlHeader.ReturnCode = CSMI_SAS_STATUS_INVALID_PARAMETER is returned and no change occurs.
- bSpinupRate. Contains the repetition rate at which the NOTIFY(SPINUP) primitive is generated on this phy for SAS devices or the release interval of COMWAKE in response

to a COMINIT for SATA.. The value is in seconds. A 0 value indicates that the NOTIFY(SPINUP) primitive generation is disabled for SAS or COMWAKE is not released in response to a COMINIT for SATA. The result is that a device shall stay in the non-spinup state indefinitely. If the value is out of range, then the IoctlHeader.ReturnCode = CSMI_SAS_STATUS_INVALID_PARAMETER is returned and the maximum value of the spinup rate is set.

- uVendorUnique. Contains vendor unique information. Vendor is responsible for positively detecting the validity of the data provided. Shall be initialized to 0 when not providing vendor unique information.
- Control[]. The elements of this structure contain phy signal controls If the control structure is required for the function and is incorrect then the IoctlHeader.ReturnCode = CSMI_SAS_STATUS_INVALID_PARAMETER is returned.
- Control[].bType. Contains the device type of the control and may be one of the following:
  - CSMI_SAS_SATA, indicates that the phy settings shall be applied when a SATA device is attached to the phy.
  - CSMI_SAS_SAS, indicates that the phy settings shall be applied when a SAS device is attached to the phy.
- Control[].bRate. Contains the link rate for which the setting or control applies and may be one of the following;
  - CSMI_SAS_LINK_RATE_UNKNOWN, indicates that the structure content is unknown on input or invalid on output.
  - CSMI_SAS_LINK_RATE_1_5_GPBS, indicates that the structure content is valid for a 1.5 Gbps link rate.
  - CSMI_SAS_LINK_RATE_3_0_GBPS, indicates that the structure content is valid for a 3.0 Gbps link rate.
- Control[].uVendorUnique. Contains vendor unique information. Vendor is responsible for positively detecting the validity of the data provided. Shall be initialized to 0 when not providing vendor unique information.
- Control[].uTransmitterFlags. Contains flags that define the transmitter characteristics or link characteristics. The value may be one of the following;
  - CSMI_SAS_PHY_PREEMPHASIS_DISABLED, indicates that preemphasis on the transmitter shall be disabled during input or is disabled during output.
- Control[].bTransmitterAmplitude. Contains the step offset from the default setting that the transmitter shall use to establish the transmitter driver voltage amplitude. The field value shall be treated as a 2's-complement signed value that can range from –128 to +127. If the step requested is out of range of the transmitter capability an IoctlHeader.ReturnCode = CSMI_SAS_STATUS_INVALID_PARAMETER is returned. A value of 0 shall always be accepted, even if the control is not supported.
- Control[].bTransmitterPreemphasis. Contains the step offset from the default setting that the transmitter shall use to establish the transmitter driver voltage preemphasis. The field value shall be treated as a 2's-complement signed value that can range from –128 to +127. If the step requested is out of range of the transmitter capability an IoctlHeader.ReturnCode = CSMI_SAS_STATUS_INVALID_PARAMETER is returned. A value of 0 shall always be accepted, even if the control is not supported.
- Control[].bTransmitterSlewRate. Contains the step offset from the default setting that the transmitter shall use to establish the transmitter driver voltage slew rate. The field value shall be treated as a 2's-complement signed value that can range from –128 to +127. If the step requested is out of range of the transmitter capability an IoctlHeader.ReturnCode = CSMI_SAS_STATUS_INVALID_PARAMETER is returned. A value of 0 shall always be accepted, even if the control is not supported.
- Control[].bTransmitterReserved. Set to 0.
- Control[].bTransmitterVendorUnique. Contains vendor unique information. Vendor is responsible for positively detecting the validity of the data provided. Shall be initialized to 0 when not providing vendor unique information.

- Control[].bReceiverFlags. Contains flags that define the receiver characteristics or link characteristics. The value may be one or more of the following;
  - CSMI_SAS_PHY_ACTIVATE_CONTROL, indicates that the receiver controls provided shall be updated to the current settings. If this flag is not set, then the receiver controls will not be modified during input or are not active during output.
  - CSMI_SAS_PHY_EQUALIZATION_DISABLED, indicates that any receiver equalization shall be disabled during input or is disabled during output.
- Control[].bReceiverThreshold. Contains the step offset from the default setting that the receiver shall use to establish the receiver signal detection threshold. The field value shall be treated as a 2's-complement signed value that can range from –128 to +127. If the step requested is out of range of the receiver capability an IoctlHeader.ReturnCode = CSMI_SAS_STATUS_INVALID_PARAMETER is returned. A value of 0 shall always be accepted, even if the control is not supported.
- Control[].bReceiverEqualizationGain. Contains the step offset from the default setting that the receiver shall use to establish the receiver signal equalization gain. The field value shall be treated as a 2's-complement signed value that can range from –128 to +127. If the step requested is out of range of the receiver capability an IoctlHeader.ReturnCode = CSMI_SAS_STATUS_INVALID_PARAMETER is returned. A value of 0 shall always be accepted, even if the control is not supported.
- Control[].bReceiverReserved. Set to 0.
- Control[].bReceiverVendorUnique. Contains vendor unique information. Vendor is responsible for positively detecting the validity of the data provided. Shall be initialized to 0 when not providing vendor unique information.
- Control[].uPatternFlags. Contains flags that define whether the phy shall enter a pattern generation mode. The value may be one or more of the following;
  - CSMI_SAS_PHY_ACTIVATE_CONTROL, indicates that the pattern generation mode shall be activated. If this flag is not set, then the pattern generation mode is not activited during input or is not active during output. If this flag is set, then the phy will remain in pattern generation mode until another link reset is initiated. Only a single phy control may have this bit set at any one time. If multiple phy controls have this bit set, then an IoctlHeader.ReturnCode = CSMI_SAS_STATUS_INVALID_PARAMETER is returned and pattern generation is aborted.
  - CSMI_SAS_PHY_FIXED_PATTERN, indicates that the fixed pattern shall be used in pattern generation. This bit may not be used in conjunction with the CSMI_SAS_PHY_USER_PATTERN bit. If both are set then an IoctlHeader.ReturnCode = CSMI_SAS_STATUS_INVALID_PARAMETER is returned and pattern generation is aborted.
  - CSMI_SAS_PHY_DISABLE_SCRAMBLING, indicates that the phy shall disable data scrambling during input or data scrambling is disabled during output.
  - CSMI_SAS_PHY_DISABLE_ALIGN indicates that the phy shall disable align insertion during input or align insertion is disabled during output.
  - CSMI_SAS_PHY DISABLE_SSC, indicates that the phy shall disable spread spectrum clocking during input or spread spectrum clocking is disabled during output.
- Control[].bFixedPattern. Contains the SAS or SATA specification pattern and may be one of the following:
  - CSMI_SAS_PHY_CJPAT, indicates that the pattern used shall be the CJPAT as defined in the SATA and/or SAS specification.
  - CSMI_SAS_PHY_ALIGN, indicates that the pattern used shall be the ALIGN[1] repeated value.
- Control[].bUserPatternLength. Contains the length in bytes of the user pattern buffer (see bUserPattern). The value must be less than the number of elements in the user pattern buffer.

- Control[].UserPatternBuffer[]. Contains an array of CSMI_SAS_CHARACTER elements that define the user data pattern. If the user pattern length and CSMI_SAS_PHY_USER_PATTERN bit are not set then this array shall be 0 filled. If the type flags (see bTypeFlags) for the user pattern are not supported or the user pattern is not supported then an IoctlHeader.ReturnCode = CSMI_SAS_STATUS_INVALID_PARAMETER is returned and pattern generation is aborted.
- Control[].UserPatternBuffer[].bTypeFlags. Contains flags that define the type of character to generate and may be one or more of the following;
  - CSMI_SAS_PHY_POSITIVE_DISPARITY, indicates that the character shall have a running disparity that is positive.
  - CSMI_SAS_PHY_NEGATIVE_DISPARITY, indicates that the character shall have a running disparity that is negative.
  - CSMI_SAS_PHY_CONTROL_CHARACTER, indicates that the character shall be encoded as a control character.
- Control[].UserPatternBuffer[].bValue. Contains the base value used to generate the character.

6.17.3 Output

A CSMI_SAS_PHY_CONTROL_BUFFER data structure with the returned data;

- IoctlHeader. Same as input.
- uFunction. Same as input.
- bPhyIdentifier. Same as input
- bLengthOfControl. See input definition.
- bNumberOfControls. See input definition.
- uVendorUnique. Contains vendor unique information.
- Control[]. See input definition.
- Control[].bType. See input definition.
- Control[].bRate. See input definition.
- Control[].uVendorUnique. Contains vendor unique information.
- Control[].uTransmitterFlags. See input definition.
- Control[].bTransmitterAmplitude. See input definition.
- Control[].bTransmitterPreemphasis. See input definition.
- Control[].bTransmitterSlewRate. See input definition.
- Control[].bTransmitterReserved. Same as input.
- Control[].bTransmitterVendorUnique. Contains vendor unique information.
- Control[].bReceiverFlags. See input definition.
- Control[].bReceiverThreshold. See input definition.
- Control[].bReceiverEqualizationGain. See input definition.
- Control[].bReceiverReserved. Same as input.
- Control[].bReceiverVendorUnique. Contains vendor unique information.
- Control[].uPatternFlags. See input definition.
- Control[].bFixedPattern. See input definition.
- Control[].bUserPatternLength. See input definition.
- Control[].bUserPatternBuffer. See input definition.

6.17.4 Structure Definitions

6.17.4.1 CSMI_SAS_PHY_CONTROL

```
typedef struct _CSMI_SAS_PHY_CONTROL {
  __u8  bType;
  __u8  bRate;
```

```
  __u8  bReserved[6];
  __u32 uVendorUnique[8];
  __u32 uTransmitterFlags;
  __i8  bTransmitAmplitude;
  __i8  bTransmitterPreemphasis;
  __i8  bTransmitterSlewRate;
  __i8  bTransmitterReserved[13];
  __u8  bTransmitterVendorUnique[64];
  __u32 uReceiverFlags;
  __i8  bReceiverThreshold;
  __i8  bReceiverEqualizationGain;
  __i8  bReceiverReserved[14];
  __u8  bReceiverVendorUnique[64];
  __u8  bFixedPattern;
  __u8  bUserPatternLength;
  __u8  bPatternReserved[6];
  __u8  bUserPattern[32];
} CSMI_SAS_PHY_CONTROL,
  *PCSMI_SAS_PHY_CONTROL;
```

6.17.4.2 CSMI_SAS_PHY_CONTROL_BUFFER

```
typedef struct _CSMI_SAS_PHY_CONTROL_BUFFER {
  IOCTL_HEADER IoctlHeader;
  __u32 uFunction;
  __u8  bPhyIdentifier;
  __u16  usLengthOfControls;
  __u8  bNumberOfControls;
  __u8  bReserved[4];
  __u32 uLinkFlags;
  __u8 bSpinupRate;
  __u8 bLinkReserved[7];
  __u32 uVendorUnique[8];
  CSMI_SAS_PHY_CONTROL Control[1];
} CSMI_SAS_PHY_CONTROL_BUFFER,
  *PCSMI_SAS_PHY_CONTROL_BUFFER;
```

6.18 CC_CSMI_SAS_GET_CONNECTOR_INFO

(SATA, SAS)

6.18.1 Behavior

This CSMI functional behavior provides a method for obtaining the connector information for a controller. Any driver that implements this specification must support this behavior.

6.18.2 Input

A CSMI_SAS_GET_CONNECTOR_INFO_BUFFER data structure with the following initialized members;

- IoctlHeader. See IOCTL_HEADER for the platform specific initialization.
- Reference[0 - 31]. Initialized to 0.

6.18.3 Output

A CSMI_SAS_GET_CONNECTOR_INFO_BUFFER data structure with the returned data;

- IoctlHeader.ReturnCode. See Return Codes.
- Reference[0 - 31]. Contains the reference structure for up to 32 phys. The number of valid reference structures corresponds to the number of phys defined in the CC_CSMI_SAS_GET_PHY_INFO functional behavior.
- Reference.bConnector. Contains a null terminated ASCII string that is the reference designator for the component that provides physical connectivity for the phy.
- Reference.uPinout. Contains the pinout identifier for the phy in the connector component and will be one or move of the following:
  - CSMI_SAS_CON_UNKNOWN. Indicates that the phy pinout is unknown.
  - CSMI_SAS_CON_SFF_8482. Indicates that the phy is pinned out as a single lane SFF-8482 connector.
  - CSMI_SAS_CON_SFF_8470_LANE_1. Indicates that the phy is pinned out as lane 1 in an SFF-8470, 4 lane connector.
  - CSMI_SAS_CON_SFF_8470_LANE_2. Indicates that the phy is pinned out as lane 2 in an SFF-8470, 4 lane connector.
  - CSMI_SAS_CON_SFF_8470_LANE_3. Indicates that the phy is pinned out as lane 3 in an SFF-8470, 4 lane connector.
  - CSMI_SAS_CON_SFF_8470_LANE_4. Indicates that the phy is pinned out as lane 4 in an SFF-8470, 4 lane connector.
  - CSMI_SAS_CON_SFF_8484_LANE_1. Indicates that the phy is pinned out as lane 1 in an SFF-8484, 4 lane connector.
  - CSMI_SAS_CON_SFF_8484_LANE_2. Indicates that the phy is pinned out as lane 2 in an SFF-8484, 4 lane connector.
  - CSMI_SAS_CON_SFF_8484_LANE_3. Indicates that the phy is pinned out as lane 3 in an SFF-8484, 4 lane connector.
  - CSMI_SAS_CON_SFF_8484_LANE_4. Indicates that the phy is pinned out as lane 4 in an SFF-8484, 4 lane connector.
- Reference.bLocation. Contains the location identifier for the connector and will be one or more of the following:
  - CSMI_SAS_CON_UNKNOWN. Indicates that the connector location is unknown.
  - CSMI_SAS_CON_INTERNAL. Indicates that the connector is positioned for connecting to devices internal to a system.
  - CSMI_SAS_CON_EXTERNAL. Indicates that the connector is positioned for connecting to devices external to a system.

- CSMI_SAS_CON_SWITCHABLE. Indicates that the phy is switchable between an internal or external connector.
- CSMI_SAS_CON_AUTO. Indicates that the phy will auto detect activity on an internal or external connector and switch.

6.18.4 Structure Definitions

6.18.4.1 CSMI_SAS_GET_CONNECTOR_INFO

```
typedef struct _CSMI_SAS_GET_CONNECTOR_INFO {
  __u32 uPinout;
  __u8  bConnector[16];
  __u8  bLocation;
  __u8  bReserved[15];
} CSMI_SAS_CONNECTOR_INFO,
  *PCSMI_SAS_CONNECTOR_INFO;
```

6.18.4.2 CSMI_SAS GET_CONNECTOR_INFO BUFFER

```
typedef struct _CSMI_SAS_CONNECTOR_INFO_BUFFER {
  IOCTL_HEADER IoctlHeader;
  CSMI_SAS_CONNECTOR_INFO Reference[32];
} CSMI_SAS_CONNECTOR_INFO_BUFFER,
  *PCSMI_SAS_CONNECTOR_INFO_BUFFER;
```

7 SCSI Emulation

7.1 Vendor Unique ATA Passthru

In some implementations, it is more convenient for the driver to provide access to the STP and SATA device as an emulated SCSI device. In these implementations the CC_CSMI_SAS_STP_PASSTHRU functional behavior may not be available. To allow native ATA commands to be generically passed down to the emulation layer, a special purpose SCSI command is provided. This command wraps an ATA command as a SCSI CDB.

Table 9: Vendor Unique ATA Passthru

	7	6	5	4	3	2	1	0
0	Operation Code (E0h)							
1	Word/Block	Reserved	Reserved	Reserved	Protocol			
2	ATA Command							
3	Features Ext							
4	Features							
5	Sector Count Ext							
6	Sector Count							
7	LBA High Ext							
8	LBA Mid Ext							
9	LBA Low Ext							
10	LBA High							
11	LBA Mid							
12	LBA Low							
13	Device							
14	Word/Block Count High							
15	Word/Block Count Low							

The definitions of each CDB byte are provided below:

- Operation Code, vendor unique (E0h), defines a 16 byte CDB targeted at the emulation layer of the HBA stack.
- Protocol, defines the SATA protocol state machine to use for executing the ATA Command:
  - No Data, 0h
  - PIO Read, 1h
  - PIO Write, 2h
  - DMA Read, 3h
  - DMA Write, 4h
  - Packet Read, 5h
  - Packet Write, 6h
  - Queued DMA Read, 7h
  - Queued DMA Write, 8h
- Word/Block, defines whether the Word/Block Count is counting blocks (1) or words (0).
- ATA Command, defines the ATA Command byte
- Features Ext, contains the “current” value of the Features field when 48 bit addressing mode is enabled.
- Features, contains the ATA Features register value.

- Sector Count Ext, contains the “previous” value of the Sector Count field when 48 bit addressing mode is enabled.
- Sector Count, contains the ATA Sector Count register value.
- LBA High Ext, contains the “previous” value of the LBA High field when 48 bit addressing mode is enabled.
- LBA Mid Ext, contains the “previous” value of the LBA Mid field when 48 bit addressing mode is enabled.
- LBA Low Ext, contains the “previous” value of the LBA Low field when 48 bit addressing mode is enabled.
- LBA High, contains the ATA LBA High register value.
- LBA Mid, contains the ATA LBA Mid register value.
- LBA Low, contains the ATA LBA Low register value.
- Device, contains the ATA Device register value.
- Word/Block Count High, along with Word/Block Count Low; defines the expected number of words or blocks in the data transfer associated with the command. A block is 512 bytes.
- Word/Block Count Low, see Word/Block Count High.

7.1.1 Security

A driver may post IoctrlHeader.ReturnCode = CSMI_SAS_SCSI_WRITE_ATTEMPTED, if the security level is insufficient to complete the command requested (see Security and Enabling Features). Vendor unique commands are only allowed if Full access is enabled. The security access required for the drive ATA command set is provided in Table 8. As ATA standard commands or devices are added the security shall be applied appropriately.

8 Timeouts

One of the following constants may be provided in the *Timeout* field of the IOCTL_HEADER structure on submission of the device I/O control call. These are recommended values only and may need to be adjusted based on actual implementations.

Table 10: Timeout Defaults

CSMI Control Code	Timeout
CC_CSMI_SAS_GET_DRIVER_INFO	CSMI_ALL_TIMEOUT
CC_CSMI_SAS_GET_CNTLR_CONFIG	CSMI_ALL_TIMEOUT
CC_CSMI_SAS_GET_CNTLR_STATUS	CSMI_ALL_TIMEOUT
CC_CSMI_SAS_FIRMWARE_DOWNLOAD	CSMI_ALL_TIMEOUT
CC_CSMI_SAS_GET_RAID_INFO	CSMI_RAID_TIMEOUT
CC_CSMI_SAS_GET_RAID_CONFIG	CSMI_RAID_TIMEOUT
CC_CSMI_SAS_GET_PHY_INFO	CSMI_SAS_TIMEOUT
CC_CSMI_SAS_SET_PHY_INFO	CSMI_SAS_TIMEOUT
CC_CSMI_SAS_GET_LINK_ERRORS	CSMI_SAS_TIMEOUT
CC_CSMI_SAS_SMP_PASSTHROUGH	CSMI_SAS_TIMEOUT
CC_CSMI_SAS_SSP_PASSTHROUGH	CSMI_SAS_TIMEOUT
CC_CSMI_SAS_STP_PASSTHROUGH	CSMI_SAS_TIMEOUT
CC_CSMI_SAS_GET_SATA_SIGNATURE	CSMI_SAS_TIMEOUT
CC_CSMI_SAS_GET_SCSI_ADDRESS	CSMI_SAS_TIMEOUT
CC_CSMI_SAS_GET_DEVICE_ADDRESS	CSMI_SAS_TIMEOUT
CC_CSMI_SAS_TASK_MANAGEMENT	CSMI_SAS_TIMEOUT
CC_CSMI_SAS_GET_CONNECTOR_INFO	CSMI_SAS_TIMEOUT
CC_CSMI_SAS_PHY_CONTROL	CSMI_SAS_TIMEOUT

9 Return Codes

One of the following constants may be provided in the *ReturnCode* field of the IOCTL_HEADER structure on completion of the device I/O control call.

Table 11: Return Codes

CSMI Return Code (CSMI_SAS_…)	Returned by CSMI Control Code (CC_CSMI_SAS_…)	Description
STATUS_SUCCESS	GET_DRIVER_INFO GET_CNTLR_CONFIG GET_CNTLR_STATUS FIRMWARE_DOWNLOAD GET_RAID_INFO GET_RAID_CONFIG GET_PHY_INFO SET_PHY_INFO GET_LINK_ERRORS SMP_PASSTHROUGH SSP_PASSTHROUGH STP_PASSTHROUGH GET_SATA_SIGNATURE GET_SCSI_ADDRESS GET_DEVICE_ADDRESS TASK_MANAGEMENT GET_CONNECTOR_INFO PHY_CONTROL	CSMI functional behavior specified by the CSMI control code completed successfully.
STATUS_FAILED	GET_DRIVER_INFO GET_CNTLR_CONFIG GET_CNTLR_STATUS GET_RAID_INFO GET_RAID_CONFIG GET_PHY_INFO SET_PHY_INFO GET_LINK_ERRORS SMP_PASSTHROUGH SSP_PASSTHROUGH STP_PASSTHROUGH GET_SATA_SIGNATURE GET_SCSI_ADDRESS GET_DEVICE_ADDRESS TASK_MANAGEMENT GET_CONNECTOR_INFO PHY_CONTROL	CSMI functional behavior specified by the CSMI control code failed to complete. This is the non-specific default for an error condition that does not meet a more specific definition.
BAD_CNTL_CODE	Any reserved code	The CSMI control code is invalid or unknown.

CSMI Return Code (CSMI_SAS_…)	Returned by CSMI Control Code (CC_CSMI_SAS_…)	Description
INVALID_PARAMETER	GET_DRIVER_INFO GET_CNTLR_CONFIG GET_CNTLR_STATUS FIRMWARE_DOWNLOAD GET_RAID_INFO GET_RAID_CONFIG GET_PHY_INFO SET_PHY_INFO GET_LINK_ERRORS SMP_PASSTHROUGH SSP_PASSTHROUGH STP_PASSTHROUGH GET_SATA_SIGNATURE GET_SCSI_ADDRESS GET_DEVICE_ADDRESS TASK_MANAGEMENT PHY_CONTROL	The CSMI data structure contained an invalid parameter on input. No additional information is provided.
WRITE_ATTEMPTED	SET_PHY_INFO SMP_PASSTHROUGH SSP_PASSTHROUGH STP_PASSTHROUGH TASK_MANAGEMENT SET_PHY_INFO PHY_CONTROL	The CSMI data structure contained a directive to write information to the physical device and the security provisions do not allow the operation.
RAID_SET_OUT_OF_RANGE	GET_RAID_CONFIG	URaidSetIndex is out of range.
PHY_INFO_CHANGED	SET_PHY_INFO	Phy information was successfully changed.
PHY_INFO_NOT_CHANGEABLE	SET_PHY_INFO	Phy information could not be changed. Indicates that the driver does not support changing the phy information.
LINK_RATE_OUT_OF_RANGE	SET_PHY_INFO PHY_CONTROL	The link rate was not supported by the hardware.
PHY_DOES_NOT_EXIST	SET_PHY_INFO GET_LINK_ERRORS SMP_PASSTHROUGH SSP_PASSTHROUGH STP_PASSTHROUGH GET_SATA_SIGNATURE PHY_CONTROL	Specified phy does not exist
PHY_DOES_NOT_MATCH_PORT	SMP_PASSTHROUGH SSP_PASSTHROUGH STP_PASSTHROUGH	The phy and port combination does not exist
PHY_CANNOT_BE_SELECTED	SMP_PASSTHROUGH SSP_PASSTHROUGH STP_PASSTHROUGH	Specified phy cannot be selected
SELECT_PHY_OR_PORT	SMP_PASSTHROUGH SSP_PASSTHROUGH STP_PASSTHROUGH	Return code indicating that either phy or port needs to be selected
PORT_DOES_NOT_EXIST	SMP_PASSTHROUGH SSP_PASSTHROUGH STP_PASSTHROUGH	Specified port does not exist
PORT_CANNOT_BE_SELECTED	SMP_PASSTHROUGH SSP_PASSTHROUGH STP_PASSTHROUGH	Specified port cannot be selected

CSMI Return Code (CSMI_SAS_…)	Returned by CSMI Control Code (CC_CSMI_SAS_…)	Description
CONNECTION_FAILED	SMP_PASSTHROUGH SSP_PASSTHROUGH STP_PASSTHROUGH TASK_MANAGEMENT	Connection failed.
NO_SATA_DEVICE	GET_SATA_SIGNATURE	Specified phy is not connected to a SATA device or has not completed a SATA OOB sequence.
NO_SATA_SIGNATURE	GET_SATA_SIGNATURE	Specified phy has not received the initial register device to host FIS from the SATA device
SCSI_EMULATION	STP_PASSTHROUGH	Use the SCSI emulation CDB for passing SATA commands
NOT_AN_END_DEVICE	GET_SCSI_ADDRESS TASK_MANAGEMENT	The OS specific platform address cannot be returned because the device is not an end device
NO_SCSI_ADDRESS	GET_SCSI_ADDRESS TASK_MANAGEMENT	No OS specific platform address was found for this SAS address
NO_DEVICE_ADDRESS	GET_DEVICE_ADDRESS TASK_MANAGEMENT	No SAS address was found for this OS specific platform address

10 Reference Header File – CSMISAS.H

```
/*****************************************************************************

Module Name:

   CSMISAS.H


Abstract:

   This file contains constants and data structure definitions used by drivers
   that support the Common Storage Management Interface specification for
   SAS or SATA in either the Windows or Linux.

   This shall be considered as a reference implementation only.  Changes may
   be necessary to accommodate a specific build environment or target OS.

*****************************************************************************/

#ifndef _CSMI_SAS_H_
#define _CSMI_SAS_H_

// CSMI Specification Revision, the intent is that all versions of the
// specification will be backward compatible after the 1.00 release.
// Major revision number, corresponds to xxxx. of CSMI specification
// Minor revision number, corresponds to .xxxx of CSMI specification
#define CSMI_MAJOR_REVISION    0
#define CSMI_MINOR_REVISION    81

/*****************************************************************************/
/* TARGET OS LINUX SPECIFIC CODE                                             */
/*****************************************************************************/

#ifdef _linux

// Linux base types

#include <linux/types.h>

// pack definition

#define CSMI_SAS_BEGIN_PACK(x)     pack(x)
#define CSMI_SAS_END_PACK          pack()

// IOCTL Control Codes
// (IoctlHeader.ControlCode)

// Control Codes prior to 0.77

// Control Codes requiring CSMI_ALL_SIGNATURE

// #define CC_CSMI_SAS_GET_DRIVER_INFO    0x12345678
// #define CC_CSMI_SAS_GET_CNTLR_CONFIG   0x23456781
// #define CC_CSMI_SAS_GET_CNTLR_STATUS   0x34567812
// #define CC_CSMI_SAS_FIRMWARE_DOWNLOAD  0x92345678

// Control Codes requiring CSMI_RAID_SIGNATURE

// #define CC_CSMI_SAS_GET_RAID_INFO      0x45678123
// #define CC_CSMI_SAS_GET_RAID_CONFIG    0x56781234
```

```
// Control Codes requiring CSMI_SAS_SIGNATURE

// #define CC_CSMI_SAS_GET_PHY_INFO       0x67812345
// #define CC_CSMI_SAS_SET_PHY_INFO       0x78123456
// #define CC_CSMI_SAS_GET_LINK_ERRORS    0x81234567
// #define CC_CSMI_SAS_SMP_PASSTHRU       0xA1234567
// #define CC_CSMI_SAS_SSP_PASSTHRU       0xB1234567
// #define CC_CSMI_SAS_STP_PASSTHRU       0xC1234567
// #define CC_CSMI_SAS_GET_SATA_SIGNATURE 0xD1234567
// #define CC_CSMI_SAS_GET_SCSI_ADDRESS   0xE1234567
// #define CC_CSMI_SAS_GET_DEVICE_ADDRESS 0xF1234567
// #define CC_CSMI_SAS_TASK_MANAGEMENT    0xA2345678

// Control Codes for 0.77 and later

// Control Codes requiring CSMI_ALL_SIGNATURE

#define CC_CSMI_SAS_GET_DRIVER_INFO    0xCC770001
#define CC_CSMI_SAS_GET_CNTLR_CONFIG   0xCC770002
#define CC_CSMI_SAS_GET_CNTLR_STATUS   0xCC770003
#define CC_CSMI_SAS_FIRMWARE_DOWNLOAD  0xCC770004

// Control Codes requiring CSMI_RAID_SIGNATURE

#define CC_CSMI_SAS_GET_RAID_INFO      0xCC77000A
#define CC_CSMI_SAS_GET_RAID_CONFIG    0xCC77000B

// Control Codes requiring CSMI_SAS_SIGNATURE

#define CC_CSMI_SAS_GET_PHY_INFO       0xCC770014
#define CC_CSMI_SAS_SET_PHY_INFO       0xCC770015
#define CC_CSMI_SAS_GET_LINK_ERRORS    0xCC770016
#define CC_CSMI_SAS_SMP_PASSTHRU       0xCC770017
#define CC_CSMI_SAS_SSP_PASSTHRU       0xCC770018
#define CC_CSMI_SAS_STP_PASSTHRU       0xCC770019
#define CC_CSMI_SAS_GET_SATA_SIGNATURE 0xCC770020
#define CC_CSMI_SAS_GET_SCSI_ADDRESS   0xCC770021
#define CC_CSMI_SAS_GET_DEVICE_ADDRESS 0xCC770022
#define CC_CSMI_SAS_TASK_MANAGEMENT    0xCC770023
#define CC_CSMI_SAS_GET_CONNECTOR_INFO 0xCC770024

// Control Codes requiring CSMI_PHY_SIGNATURE

#define CC_CSMI_SAS_PHY_CONTROL        0xCC77003C

#pragma CSMI_SAS_BEGIN_PACK(8)

// IOCTL_HEADER
typedef struct _IOCTL_HEADER {
    __u32 IOControllerNumber;
        __u32 Length;
        __u32 ReturnCode;
        __u32 Timeout;
        __u16 Direction;
} IOCTL_HEADER, *PIOCTL_HEADER;

#pragma CSMI_SAS_END_PACK

#endif

/***************************************************************************/
/* TARGET OS WINDOWS SPECIFIC CODE                                         */
/***************************************************************************/
```

```
#ifdef _WIN32

// windows IOCTL definitions

#ifndef _NTDDSCSIH_
#include <ntddscsi.h>
#endif

// pack definition

#if defined _MSC_VER
   #define CSMI_SAS_BEGIN_PACK(x)    pack(push,x)
   #define CSMI_SAS_END_PACK         pack(pop)
#elif defined __BORLANDC__
   #define CSMI_SAS_BEGIN_PACK(x)    option -a##x
   #define CSMI_SAS_END_PACK         option -a.
#else
   #error "CSMISAS.H - Must externally define a pack compiler designator."
#endif

// base types

#define __u8    unsigned char
#define __u32   unsigned long
#define __u16   unsigned short

#define __i8    char

// IOCTL Control Codes
// (IoctlHeader.ControlCode)

// Control Codes requiring CSMI_ALL_SIGNATURE

#define CC_CSMI_SAS_GET_DRIVER_INFO    1
#define CC_CSMI_SAS_GET_CNTLR_CONFIG   2
#define CC_CSMI_SAS_GET_CNTLR_STATUS   3
#define CC_CSMI_SAS_FIRMWARE_DOWNLOAD  4

// Control Codes requiring CSMI_RAID_SIGNATURE

#define CC_CSMI_SAS_GET_RAID_INFO      10
#define CC_CSMI_SAS_GET_RAID_CONFIG    11

// Control Codes requiring CSMI_SAS_SIGNATURE

#define CC_CSMI_SAS_GET_PHY_INFO       20
#define CC_CSMI_SAS_SET_PHY_INFO       21
#define CC_CSMI_SAS_GET_LINK_ERRORS    22
#define CC_CSMI_SAS_SMP_PASSTHRU       23
#define CC_CSMI_SAS_SSP_PASSTHRU       24
#define CC_CSMI_SAS_STP_PASSTHRU       25
#define CC_CSMI_SAS_GET_SATA_SIGNATURE 26
#define CC_CSMI_SAS_GET_SCSI_ADDRESS   27
#define CC_CSMI_SAS_GET_DEVICE_ADDRESS 28
#define CC_CSMI_SAS_TASK_MANAGEMENT    29
#define CC_CSMI_SAS_GET_CONNECTOR_INFO 30

// Control Codes requiring CSMI_PHY_SIGNATURE

#define CC_CSMI_SAS_PHY_CONTROL        60

#define IOCTL_HEADER SRB_IO_CONTROL
```

```
#define PIOCTL_HEADER PSRB_IO_CONTROL

#endif

/*****************************************************************************/
/* TARGET OS NOT DEFINED ERROR                                               */
/*****************************************************************************/

#if (!_WIN32 && !_linux)
   #error "Unknown target OS."
#endif

/*****************************************************************************/
/* OS INDEPENDENT CODE                                                       */
/*****************************************************************************/

/* * * * * * * * * * Class Independent IOCTL Constants * * * * * * * * * */

// Return codes for all IOCTL's regardless of class
// (IoctlHeader.ReturnCode)

#define CSMI_SAS_STATUS_SUCCESS              0
#define CSMI_SAS_STATUS_FAILED               1
#define CSMI_SAS_STATUS_BAD_CNTL_CODE        2
#define CSMI_SAS_STATUS_INVALID_PARAMETER    3
#define CSMI_SAS_STATUS_WRITE_ATTEMPTED      4

// Signature value
// (IoctlHeader.Signature)

#define CSMI_ALL_SIGNATURE    "CSMIALL"

// Timeout value default of 60 seconds
// (IoctlHeader.Timeout)

#define CSMI_ALL_TIMEOUT      60

//  Direction values for data flow on this IOCTL
// (IoctlHeader.Direction, Linux only)
#define CSMI_SAS_DATA_READ    0
#define CSMI_SAS_DATA_WRITE   1

// I/O Bus Types
// ISA and EISA bus types are not supported
// (bIoBusType)

#define CSMI_SAS_BUS_TYPE_PCI       3
#define CSMI_SAS_BUS_TYPE_PCMCIA    4

// Controller Status
// (uStatus)

#define CSMI_SAS_CNTLR_STATUS_GOOD     1
#define CSMI_SAS_CNTLR_STATUS_FAILED   2
#define CSMI_SAS_CNTLR_STATUS_OFFLINE  3
#define CSMI_SAS_CNTLR_STATUS_POWEROFF 4

// Offline Status Reason
// (uOfflineReason)

#define CSMI_SAS_OFFLINE_REASON_NO_REASON             0
#define CSMI_SAS_OFFLINE_REASON_INITIALIZING          1
#define CSMI_SAS_OFFLINE_REASON_BACKSIDE_BUS_DEGRADED 2
```

```
#define CSMI_SAS_OFFLINE_REASON_BACKSIDE_BUS_FAILURE  3

// Controller Class
// (bControllerClass)

#define CSMI_SAS_CNTLR_CLASS_HBA    5

// Controller Flag bits
// (uControllerFlags)

#define CSMI_SAS_CNTLR_SAS_HBA   0x00000001
#define CSMI_SAS_CNTLR_SAS_RAID  0x00000002
#define CSMI_SAS_CNTLR_SATA_HBA  0x00000004
#define CSMI_SAS_CNTLR_SATA_RAID 0x00000008

// for firmware download
#define CSMI_SAS_CNTLR_FWD_SUPPORT  0x00010000
#define CSMI_SAS_CNTLR_FWD_ONLINE   0x00020000
#define CSMI_SAS_CNTLR_FWD_SRESET   0x00040000
#define CSMI_SAS_CNTLR_FWD_HRESET   0x00080000
#define CSMI_SAS_CNTLR_FWD_RROM     0x00100000

// Download Flag bits
// (uDownloadFlags)
#define CSMI_SAS_FWD_VALIDATE       0x00000001
#define CSMI_SAS_FWD_SOFT_RESET     0x00000002
#define CSMI_SAS_FWD_HARD_RESET     0x00000004

// Firmware Download Status
// (usStatus)
#define CSMI_SAS_FWD_SUCCESS        0
#define CSMI_SAS_FWD_FAILED         1
#define CSMI_SAS_FWD_USING_RROM     2
#define CSMI_SAS_FWD_REJECT         3
#define CSMI_SAS_FWD_DOWNREV        4

// Firmware Download Severity
// (usSeverity>
#define CSMI_SAS_FWD_INFORMATION    0
#define CSMI_SAS_FWD_WARNING        1
#define CSMI_SAS_FWD_ERROR          2
#define CSMI_SAS_FWD_FATAL          3

/* * * * * * * * * SAS RAID Class IOCTL Constants  * * * * * * * * * */

// Return codes for the RAID IOCTL's regardless of class
// (IoctlHeader.ControlCode)

#define CSMI_SAS_RAID_SET_OUT_OF_RANGE       1000

// Signature value
// (IoctlHeader.Signature)

#define CSMI_RAID_SIGNATURE    "CSMIARY"

// Timeout value default of 60 seconds
// (IoctlHeader.Timeout)

#define CSMI_RAID_TIMEOUT      60

// RAID Types
// (bRaidType)
#define CSMI_SAS_RAID_TYPE_NONE     0
```

```
#define CSMI_SAS_RAID_TYPE_0        1
#define CSMI_SAS_RAID_TYPE_1        2
#define CSMI_SAS_RAID_TYPE_10       3
#define CSMI_SAS_RAID_TYPE_5        4
#define CSMI_SAS_RAID_TYPE_15       5
#define CSMI_SAS_RAID_TYPE_OTHER    255

// RAID Status
// (bStatus)
#define CSMI_SAS_RAID_SET_STATUS_OK          0
#define CSMI_SAS_RAID_SET_STATUS_DEGRADED    1
#define CSMI_SAS_RAID_SET_STATUS_REBUILDING  2
#define CSMI_SAS_RAID_SET_STATUS_FAILED      3

// RAID Drive Status
// (bDriveStatus)
#define CSMI_SAS_DRIVE_STATUS_OK          0
#define CSMI_SAS_DRIVE_STATUS_REBUILDING  1
#define CSMI_SAS_DRIVE_STATUS_FAILED      2
#define CSMI_SAS_DRIVE_STATUS_DEGRADED    3

// RAID Drive Usage
// (bDriveUsage)
#define CSMI_SAS_DRIVE_CONFIG_NOT_USED 0
#define CSMI_SAS_DRIVE_CONFIG_MEMBER   1
#define CSMI_SAS_DRIVE_CONFIG_SPARE    2

/* * * * * * * * * * SAS HBA Class IOCTL Constants * * * * * * * * * */

// Return codes for SAS IOCTL's
// (IoctlHeader.ReturnCode)

#define CSMI_SAS_PHY_INFO_CHANGED           CSMI_SAS_STATUS_SUCCESS
#define CSMI_SAS_PHY_INFO_NOT_CHANGEABLE    2000
#define CSMI_SAS_LINK_RATE_OUT_OF_RANGE     2001

#define CSMI_SAS_PHY_DOES_NOT_EXIST         2002
#define CSMI_SAS_PHY_DOES_NOT_MATCH_PORT    2003
#define CSMI_SAS_PHY_CANNOT_BE_SELECTED     2004
#define CSMI_SAS_SELECT_PHY_OR_PORT         2005
#define CSMI_SAS_PORT_DOES_NOT_EXIST        2006
#define CSMI_SAS_PORT_CANNOT_BE_SELECTED    2007
#define CSMI_SAS_CONNECTION_FAILED          2008

#define CSMI_SAS_NO_SATA_DEVICE             2009
#define CSMI_SAS_NO_SATA_SIGNATURE          2010
#define CSMI_SAS_SCSI_EMULATION             2011
#define CSMI_SAS_NOT_AN_END_DEVICE          2012
#define CSMI_SAS_NO_SCSI_ADDRESS            2013
#define CSMI_SAS_NO_DEVICE_ADDRESS          2014

// Signature value
// (IoctlHeader.Signature)

#define CSMI_SAS_SIGNATURE    "CSMISAS"

// Timeout value default of 60 seconds
// (IoctlHeader.Timeout)

#define CSMI_SAS_TIMEOUT      60

// Device types
// (bDeviceType)
```

```
#define CSMI_SAS_PHY_UNUSED                0x00
#define CSMI_SAS_NO_DEVICE_ATTACHED        0x00
#define CSMI_SAS_END_DEVICE                0x10
#define CSMI_SAS_EDGE_EXPANDER_DEVICE      0x20
#define CSMI_SAS_FANOUT_EXPANDER_DEVICE    0x30

// Protocol options
// (bInitiatorPortProtocol, bTargetPortProtocol)

#define CSMI_SAS_PROTOCOL_SATA    0x01
#define CSMI_SAS_PROTOCOL_SMP     0x02
#define CSMI_SAS_PROTOCOL_STP     0x04
#define CSMI_SAS_PROTOCOL_SSP     0x08

// Negotiated and hardware link rates
// (bNegotiatedLinkRate, bMinimumLinkRate, bMaximumLinkRate)

#define CSMI_SAS_LINK_RATE_UNKNOWN  0x00
#define CSMI_SAS_PHY_DISABLED       0x01
#define CSMI_SAS_LINK_RATE_FAILED   0x02
#define CSMI_SAS_SATA_SPINUP_HOLD   0x03
#define CSMI_SAS_LINK_RATE_1_5_GBPS 0x08
#define CSMI_SAS_LINK_RATE_3_0_GBPS 0x09

// Discover state
// (bAutoDiscover)

#define CSMI_SAS_DISCOVER_NOT_SUPPORTED   0x00
#define CSMI_SAS_DISCOVER_NOT_STARTED     0x01
#define CSMI_SAS_DISCOVER_IN_PROGRESS     0x02
#define CSMI_SAS_DISCOVER_COMPLETE        0x03
#define CSMI_SAS_DISCOVER_ERROR           0x04

// Programmed link rates
// (bMinimumLinkRate, bMaximumLinkRate)
// (bProgrammedMinimumLinkRate, bProgrammedMaximumLinkRate)

#define CSMI_SAS_PROGRAMMED_LINK_RATE_UNCHANGED 0x00
#define CSMI_SAS_PROGRAMMED_LINK_RATE_1_5_GBPS  0x08
#define CSMI_SAS_PROGRAMMED_LINK_RATE_3_0_GBPS  0x09

// Link rate
// (bNegotiatedLinkRate in CSMI_SAS_SET_PHY_INFO)

#define CSMI_SAS_LINK_RATE_NEGOTIATE      0x00
#define CSMI_SAS_LINK_RATE_PHY_DISABLED   0x01

// Signal class
// (bSignalClass in CSMI_SAS_SET_PHY_INFO)

#define CSMI_SAS_SIGNAL_CLASS_UNKNOWN     0x00
#define CSMI_SAS_SIGNAL_CLASS_DIRECT      0x01
#define CSMI_SAS_SIGNAL_CLASS_SERVER      0x02
#define CSMI_SAS_SIGNAL_CLASS_ENCLOSURE   0x03

// Link error reset
// (bResetCounts)

#define CSMI_SAS_LINK_ERROR_DONT_RESET_COUNTS   0x00
#define CSMI_SAS_LINK_ERROR_RESET_COUNTS        0x01

// Phy identifier
```

```
// (bPhyIdentifier)

#define CSMI_SAS_USE_PORT_IDENTIFIER   0xFF

// Port identifier
// (bPortIdentifier)

#define CSMI_SAS_IGNORE_PORT           0xFF

// Programmed link rates
// (bConnectionRate)

#define CSMI_SAS_LINK_RATE_NEGOTIATED  0x00
#define CSMI_SAS_LINK_RATE_1_5_GBPS    0x08
#define CSMI_SAS_LINK_RATE_3_0_GBPS    0x09

// Connection status
// (bConnectionStatus)

#define CSMI_SAS_OPEN_ACCEPT                        0
#define CSMI_SAS_OPEN_REJECT_BAD_DESTINATION        1
#define CSMI_SAS_OPEN_REJECT_RATE_NOT_SUPPORTED     2
#define CSMI_SAS_OPEN_REJECT_NO_DESTINATION         3
#define CSMI_SAS_OPEN_REJECT_PATHWAY_BLOCKED        4
#define CSMI_SAS_OPEN_REJECT_PROTOCOL_NOT_SUPPORTED 5
#define CSMI_SAS_OPEN_REJECT_RESERVE_ABANDON        6
#define CSMI_SAS_OPEN_REJECT_RESERVE_CONTINUE       7
#define CSMI_SAS_OPEN_REJECT_RESERVE_INITIALIZE     8
#define CSMI_SAS_OPEN_REJECT_RESERVE_STOP           9
#define CSMI_SAS_OPEN_REJECT_RETRY                  10
#define CSMI_SAS_OPEN_REJECT_STP_RESOURCES_BUSY     11
#define CSMI_SAS_OPEN_REJECT_WRONG_DESTINATION      12

// SSP Flags
// (uFlags)

#define CSMI_SAS_SSP_READ          0x00000001
#define CSMI_SAS_SSP_WRITE         0x00000002
#define CSMI_SAS_SSP_UNSPECIFIED   0x00000004

#define CSMI_SAS_SSP_TASK_ATTRIBUTE_SIMPLE         0x00000000
#define CSMI_SAS_SSP_TASK_ATTRIBUTE_HEAD_OF_QUEUE  0x00000010
#define CSMI_SAS_SSP_TASK_ATTRIBUTE_ORDERED        0x00000020
#define CSMI_SAS_SSP_TASK_ATTRIBUTE_ACA            0x00000040

// SSP Data present
// (bDataPresent)

#define CSMI_SAS_SSP_NO_DATA_PRESENT         0x00
#define CSMI_SAS_SSP_RESPONSE_DATA_PRESENT   0x01
#define CSMI_SAS_SSP_SENSE_DATA_PRESENT      0x02

// STP Flags
// (uFlags)

#define CSMI_SAS_STP_READ          0x00000001
#define CSMI_SAS_STP_WRITE         0x00000002
#define CSMI_SAS_STP_UNSPECIFIED   0x00000004
#define CSMI_SAS_STP_PIO           0x00000010
#define CSMI_SAS_STP_DMA           0x00000020
#define CSMI_SAS_STP_PACKET        0x00000040
#define CSMI_SAS_STP_DMA_QUEUED    0x00000080
#define CSMI_SAS_STP_EXECUTE_DIAG  0x00000100
```

```
#define CSMI_SAS_STP_RESET_DEVICE   0x00000200

// Task Management Flags
// (uFlags)

#define CSMI_SAS_TASK_IU                0x00000001
#define CSMI_SAS_HARD_RESET_SEQUENCE    0x00000002
#define CSMI_SAS_SUPPRESS_RESULT        0x00000004

// Task Management Functions
// (bTaskManagement)

#define CSMI_SAS_SSP_ABORT_TASK           0x01
#define CSMI_SAS_SSP_ABORT_TASK_SET       0x02
#define CSMI_SAS_SSP_CLEAR_TASK_SET       0x04
#define CSMI_SAS_SSP_LOGICAL_UNIT_RESET   0x08
#define CSMI_SAS_SSP_CLEAR_ACA            0x40
#define CSMI_SAS_SSP_QUERY_TASK           0x80

// Task Management Information
// (uInformation)

#define CSMI_SAS_SSP_TEST           1
#define CSMI_SAS_SSP_EXCEEDED       2
#define CSMI_SAS_SSP_DEMAND         3
#define CSMI_SAS_SSP_TRIGGER        4

// Connector Pinout Information
// (uPinout)

#define CSMI_SAS_CON_UNKNOWN              0x00000001
#define CSMI_SAS_CON_SFF_8482             0x00000002
#define CSMI_SAS_CON_SFF_8470_LANE_1      0x00000100
#define CSMI_SAS_CON_SFF_8470_LANE_2      0x00000200
#define CSMI_SAS_CON_SFF_8470_LANE_3      0x00000400
#define CSMI_SAS_CON_SFF_8470_LANE_4      0x00000800
#define CSMI_SAS_CON_SFF_8484_LANE_1      0x00010000
#define CSMI_SAS_CON_SFF_8484_LANE_2      0x00020000
#define CSMI_SAS_CON_SFF_8484_LANE_3      0x00040000
#define CSMI_SAS_CON_SFF_8484_LANE_4      0x00080000

// Connector Location Information
// (bLocation)

// same as uPinout above...
// #define CSMI_SAS_CON_UNKNOWN             0x01
#define CSMI_SAS_CON_INTERNAL            0x02
#define CSMI_SAS_CON_EXTERNAL            0x04
#define CSMI_SAS_CON_SWITCHABLE          0x08
#define CSMI_SAS_CON_AUTO                0x10

/* * * * * * * * SAS Phy Control Class IOCTL Constants * * * * * * * * */

// Return codes for SAS Phy Control IOCTL's
// (IoctlHeader.ReturnCode)

// Signature value
// (IoctlHeader.Signature)

#define CSMI_PHY_SIGNATURE    "CSMIPHY"

// Phy Control Functions
// (bFunction)
```

```
// values 0x00 to 0xFF are consistent in definition with the SMP PHY CONTROL
// function defined in the SAS spec
#define CSMI_SAS_PC_NOP                     0x00000000
#define CSMI_SAS_PC_LINK_RESET              0x00000001
#define CSMI_SAS_PC_HARD_RESET              0x00000002
#define CSMI_SAS_PC_PHY_DISABLE             0x00000003
// 0x04 to 0xFF reserved...
#define CSMI_SAS_PC_GET_PHY_SETTINGS        0x00000100

// Link Flags
#define CSMI_SAS_PHY_ACTIVATE_CONTROL       0x00000001
#define CSMI_SAS_PHY_UPDATE_SPINUP_RATE     0x00000002
#define CSMI_SAS_PHY_AUTO_COMWAKE           0x00000004

// Device Types for Phy Settings
// (bType)
#define CSMI_SAS_UNDEFINED 0x00
#define CSMI_SAS_SATA      0x01
#define CSMI_SAS_SAS       0x02

// Transmitter Flags
// (uTransmitterFlags)
#define CSMI_SAS_PHY_PREEMPHASIS_DISABLED     0x00000001

// Receiver Flags
// (uReceiverFlags)
#define CSMI_SAS_PHY_EQUALIZATION_DISABLED    0x00000001

// Pattern Flags
// (uPatternFlags)
#define CSMI_SAS_PHY_FIXED_PATTERN            0x00000001
#define CSMI_SAS_PHY_DISABLE_SCRAMBLING       0x00000002
#define CSMI_SAS_PHY_DISABLE_ALIGN            0x00000004
#define CSMI_SAS_PHY_DISABLE_SSC              0x00000008

// Fixed Patterns
// (bFixedPattern)
#define CSMI_SAS_PHY_CJPAT                    0x00000001
#define CSMI_SAS_PHY_ALIGN                    0x00000002

// Type Flags
// (bTypeFlags)
#define CSMI_SAS_PHY_POSITIVE_DISPARITY       0x01
#define CSMI_SAS_PHY_NEGATIVE_DISPARITY       0x02
#define CSMI_SAS_PHY_CONTROL_CHARACTER        0x04

// Miscellaneous
#define SLOT_NUMBER_UNKNOWN    0xFFFF

/***************************************************************************/
/* DATA STRUCTURES                                                         */
/***************************************************************************/

/* * * * * * * * * * Class Independent Structures * * * * * * * * * */

#pragma CSMI_SAS_BEGIN_PACK(8)

// CC_CSMI_SAS_DRIVER_INFO

typedef struct _CSMI_SAS_DRIVER_INFO {
   __u8  szName[81];
   __u8  szDescription[81];
```

```
    __u16 usMajorRevision;
    __u16 usMinorRevision;
    __u16 usBuildRevision;
    __u16 usReleaseRevision;
    __u16 usCSMIMajorRevision;
    __u16 usCSMIMinorRevision;
} CSMI_SAS_DRIVER_INFO,
  *PCSMI_SAS_DRIVER_INFO;

typedef struct _CSMI_SAS_DRIVER_INFO_BUFFER {
    IOCTL_HEADER IoctlHeader;
    CSMI_SAS_DRIVER_INFO Information;
} CSMI_SAS_DRIVER_INFO_BUFFER,
  *PCSMI_SAS_DRIVER_INFO_BUFFER;

// CC_CSMI_SAS_CNTLR_CONFIGURATION

typedef struct _CSMI_SAS_PCI_BUS_ADDRESS {
    __u8  bBusNumber;
    __u8  bDeviceNumber;
    __u8  bFunctionNumber;
    __u8  bReserved;
} CSMI_SAS_PCI_BUS_ADDRESS,
  *PCSMI_SAS_PCI_BUS_ADDRESS;

typedef union _CSMI_SAS_IO_BUS_ADDRESS {
    CSMI_SAS_PCI_BUS_ADDRESS PciAddress;
    __u8  bReserved[32];
} CSMI_SAS_IO_BUS_ADDRESS,
  *PCSMI_SAS_IO_BUS_ADDRESS;

typedef struct _CSMI_SAS_CNTLR_CONFIG {
    __u32 uBaseIoAddress;
    struct {
        __u32 uLowPart;
        __u32 uHighPart;
    } BaseMemoryAddress;
    __u32 uBoardID;
    __u16 usSlotNumber;
    __u8  bControllerClass;
    __u8  bIoBusType;
    CSMI_SAS_IO_BUS_ADDRESS BusAddress;
    __u8  szSerialNumber[81];
    __u16 usMajorRevision;
    __u16 usMinorRevision;
    __u16 usBuildRevision;
    __u16 usReleaseRevision;
    __u16 usBIOSMajorRevision;
    __u16 usBIOSMinorRevision;
    __u16 usBIOSBuildRevision;
    __u16 usBIOSReleaseRevision;
    __u32 uControllerFlags;
    __u16 usRromMajorRevision;
    __u16 usRromMinorRevision;
    __u16 usRromBuildRevision;
    __u16 usRromReleaseRevision;
    __u16 usRromBIOSMajorRevision;
    __u16 usRromBIOSMinorRevision;
    __u16 usRromBIOSBuildRevision;
    __u16 usRromBIOSReleaseRevision;
    __u8  bReserved[7];
} CSMI_SAS_CNTLR_CONFIG,
  *PCSMI_SAS_CNTLR_CONFIG;
```

```
typedef struct _CSMI_SAS_CNTLR_CONFIG_BUFFER {
    IOCTL_HEADER IoctlHeader;
    CSMI_SAS_CNTLR_CONFIG Configuration;
} CSMI_SAS_CNTLR_CONFIG_BUFFER,
  *PCSMI_SAS_CNTLR_CONFIG_BUFFER;

// CC_CSMI_SAS_CNTLR_STATUS

typedef struct _CSMI_SAS_CNTLR_STATUS {
    __u32 uStatus;
    __u32 uOfflineReason;
    __u8  bReserved[28];
} CSMI_SAS_CNTLR_STATUS,
  *PCSMI_SAS_CNTLR_STATUS;

typedef struct _CSMI_SAS_CNTLR_STATUS_BUFFER {
    IOCTL_HEADER IoctlHeader;
    CSMI_SAS_CNTLR_STATUS Status;
} CSMI_SAS_CNTLR_STATUS_BUFFER,
  *PCSMI_SAS_CNTLR_STATUS_BUFFER;

// CC_CSMI_SAS_FIRMWARE_DOWNLOAD

typedef struct _CSMI_SAS_FIRMWARE_DOWNLOAD {
    __u32 uBufferLength;
    __u32 uDownloadFlags;
    __u8  bReserved[32];
    __u16 usStatus;
    __u16 usSeverity;
} CSMI_SAS_FIRMWARE_DOWNLOAD,
  *PCSMI_SAS_FIRMWARE_DOWNLOAD;

typedef struct _CSMI_SAS_FIRMWARE_DOWNLOAD_BUFFER {
    IOCTL_HEADER IoctlHeader;
    CSMI_SAS_FIRMWARE_DOWNLOAD Information;
    __u8  bDataBuffer[1];
} CSMI_SAS_FIRMWARE_DOWNLOAD_BUFFER,
  *PCSMI_SAS_FIRMWARE_DOWNLOAD_BUFFER;

// CC_CSMI_SAS_RAID_INFO

typedef struct _CSMI_SAS_RAID_INFO {
    __u32 uNumRaidSets;
    __u32 uMaxDrivesPerSet;
    __u8  bReserved[92];
} CSMI_SAS_RAID_INFO,
  *PCSMI_SAS_RAID_INFO;

typedef struct _CSMI_SAS_RAID_INFO_BUFFER {
    IOCTL_HEADER IoctlHeader;
    CSMI_SAS_RAID_INFO Information;
} CSMI_SAS_RAID_INFO_BUFFER,
  *PCSMI_SAS_RAID_INFO_BUFFER;

// CC_CSMI_SAS_GET_RAID_CONFIG

typedef struct _CSMI_SAS_RAID_DRIVES {
    __u8  bModel[40];
    __u8  bFirmware[8];
    __u8  bSerialNumber[40];
    __u8  bSASAddress[8];
    __u8  bSASLun[8];
```

```
    __u8  bDriveStatus;
    __u8  bDriveUsage;
    __u8  bReserved[30];
} CSMI_SAS_RAID_DRIVES,
   *PCSMI_SAS_RAID_DRIVES;

typedef struct _CSMI_SAS_RAID_CONFIG {
    __u32 uRaidSetIndex;
    __u32 uCapacity;
    __u32 uStripeSize;
    __u8  bRaidType;
    __u8  bStatus;
    __u8  bInformation;
    __u8  bDriveCount;
    __u8  bReserved[20];
    CSMI_SAS_RAID_DRIVES Drives[1];
} CSMI_SAS_RAID_CONFIG,
   *PCSMI_SAS_RAID_CONFIG;

typedef struct _CSMI_SAS_RAID_CONFIG_BUFFER {
    IOCTL_HEADER IoctlHeader;
    CSMI_SAS_RAID_CONFIG Configuration;
} CSMI_SAS_RAID_CONFIG_BUFFER,
  *PCSMI_SAS_RAID_CONFIG_BUFFER;


/* * * * * * * * * * SAS HBA Class Structures * * * * * * * * * */

// CC_CSMI_SAS_GET_PHY_INFO

typedef struct _CSMI_SAS_IDENTIFY {
    __u8  bDeviceType;
    __u8  bRestricted;
    __u8  bInitiatorPortProtocol;
    __u8  bTargetPortProtocol;
    __u8  bRestricted2[8];
    __u8  bSASAddress[8];
    __u8  bPhyIdentifier;
    __u8  bSignalClass;
    __u8  bReserved[6];
} CSMI_SAS_IDENTIFY,
  *PCSMI_SAS_IDENTIFY;

typedef struct _CSMI_SAS_PHY_ENTITY {
    CSMI_SAS_IDENTIFY Identify;
    __u8  bPortIdentifier;
    __u8  bNegotiatedLinkRate;
    __u8  bMinimumLinkRate;
    __u8  bMaximumLinkRate;
    __u8  bPhyChangeCount;
    __u8  bAutoDiscover;
    __u8  bReserved[2];
    CSMI_SAS_IDENTIFY Attached;
} CSMI_SAS_PHY_ENTITY,
  *PCSMI_SAS_PHY_ENTITY;

typedef struct _CSMI_SAS_PHY_INFO {
    __u8  bNumberOfPhys;
    __u8  bReserved[3];
    CSMI_SAS_PHY_ENTITY Phy[32];
} CSMI_SAS_PHY_INFO,
  *PCSMI_SAS_PHY_INFO;
```

```
typedef struct _CSMI_SAS_PHY_INFO_BUFFER {
   IOCTL_HEADER IoctlHeader;
   CSMI_SAS_PHY_INFO Information;
} CSMI_SAS_PHY_INFO_BUFFER,
  *PCSMI_SAS_PHY_INFO_BUFFER;

// CC_CSMI_SAS_SET_PHY_INFO

typedef struct _CSMI_SAS_SET_PHY_INFO {
   __u8  bPhyIdentifier;
   __u8  bNegotiatedLinkRate;
   __u8  bProgrammedMinimumLinkRate;
   __u8  bProgrammedMaximumLinkRate;
   __u8  bSignalClass;
   __u8  bReserved[3];
} CSMI_SAS_SET_PHY_INFO,
  *PCSMI_SAS_SET_PHY_INFO;

typedef struct _CSMI_SAS_SET_PHY_INFO_BUFFER {
   IOCTL_HEADER IoctlHeader;
   CSMI_SAS_SET_PHY_INFO Information;
} CSMI_SAS_SET_PHY_INFO_BUFFER,
  *PCSMI_SAS_SET_PHY_INFO_BUFFER;

// CC_CSMI_SAS_GET_LINK_ERRORS

typedef struct _CSMI_SAS_LINK_ERRORS {
   __u8  bPhyIdentifier;
   __u8  bResetCounts;
   __u8  bReserved[2];
   __u32 uInvalidDwordCount;
   __u32 uRunningDisparityErrorCount;
   __u32 uLossOfDwordSyncCount;
   __u32 uPhyResetProblemCount;
} CSMI_SAS_LINK_ERRORS,
  *PCSMI_SAS_LINK_ERRORS;

typedef struct _CSMI_SAS_LINK_ERRORS_BUFFER {
   IOCTL_HEADER IoctlHeader;
   CSMI_SAS_LINK_ERRORS Information;
} CSMI_SAS_LINK_ERRORS_BUFFER,
  *PCSMI_SAS_LINK_ERRORS_BUFFER;

// CC_CSMI_SAS_SMP_PASSTHRU

typedef struct _CSMI_SAS_SMP_REQUEST {
   __u8  bFrameType;
   __u8  bFunction;
   __u8  bReserved[2];
   __u8  bAdditionalRequestBytes[1016];
} CSMI_SAS_SMP_REQUEST,
  *PCSMI_SAS_SMP_REQUEST;

typedef struct _CSMI_SAS_SMP_RESPONSE {
   __u8  bFrameType;
   __u8  bFunction;
   __u8  bFunctionResult;
   __u8  bReserved;
   __u8  bAdditionalResponseBytes[1016];
} CSMI_SAS_SMP_RESPONSE,
  *PCSMI_SAS_SMP_RESPONSE;

typedef struct _CSMI_SAS_SMP_PASSTHRU {
```

```
    __u8  bPhyIdentifier;
    __u8  bPortIdentifier;
    __u8  bConnectionRate;
    __u8  bReserved;
    __u8  bDestinationSASAddress[8];
    __u32 uRequestLength;
    CSMI_SAS_SMP_REQUEST Request;
    __u8  bConnectionStatus;
    __u8  bReserved2[3];
    __u32 uResponseBytes;
    CSMI_SAS_SMP_RESPONSE Response;
} CSMI_SAS_SMP_PASSTHRU,
  *PCSMI_SAS_SMP_PASSTHRU;

typedef struct _CSMI_SAS_SMP_PASSTHRU_BUFFER {
    IOCTL_HEADER IoctlHeader;
    CSMI_SAS_SMP_PASSTHRU Parameters;
} CSMI_SAS_SMP_PASSTHRU_BUFFER,
  *PCSMI_SAS_SMP_PASSTHRU_BUFFER;

// CC_CSMI_SAS_SSP_PASSTHRU

typedef struct _CSMI_SAS_SSP_PASSTHRU {
    __u8  bPhyIdentifier;
    __u8  bPortIdentifier;
    __u8  bConnectionRate;
    __u8  bReserved;
    __u8  bDestinationSASAddress[8];
    __u8  bLun[8];
    __u8  bCDBLength;
    __u8  bAdditionalCDBLength;
    __u8  bReserved2[2];
    __u8  bCDB[16];
    __u32 uFlags;
    __u8  bAdditionalCDB[24];
    __u32 uDataLength;
} CSMI_SAS_SSP_PASSTHRU,
  *PCSMI_SAS_SSP_PASSTHRU;

typedef struct _CSMI_SAS_SSP_PASSTHRU_STATUS {
    __u8  bConnectionStatus;
    __u8  bReserved[3];
    __u8  bDataPresent;
    __u8  bStatus;
    __u8  bResponseLength[2];
    __u8  bResponse[256];
    __u32 uDataBytes;
} CSMI_SAS_SSP_PASSTHRU_STATUS,
  *PCSMI_SAS_SSP_PASSTHRU_STATUS;

typedef struct _CSMI_SAS_SSP_PASSTHRU_BUFFER {
    IOCTL_HEADER IoctlHeader;
    CSMI_SAS_SSP_PASSTHRU Parameters;
    CSMI_SAS_SSP_PASSTHRU_STATUS Status;
    __u8  bDataBuffer[1];
} CSMI_SAS_SSP_PASSTHRU_BUFFER,
  *PCSMI_SAS_SSP_PASSTHRU_BUFFER;

// CC_CSMI_SAS_STP_PASSTHRU

typedef struct _CSMI_SAS_STP_PASSTHRU {
    __u8  bPhyIdentifier;
    __u8  bPortIdentifier;
```

```
    __u8  bConnectionRate;
    __u8  bReserved;
    __u8  bDestinationSASAddress[8];
    __u8  bReserved2[4];
    __u8  bCommandFIS[20];
    __u32 uFlags;
    __u32 uDataLength;
} CSMI_SAS_STP_PASSTHRU,
  *PCSMI_SAS_STP_PASSTHRU;

typedef struct _CSMI_SAS_STP_PASSTHRU_STATUS {
    __u8  bConnectionStatus;
    __u8  bReserved[3];
    __u8  bStatusFIS[20];
    __u32 uSCR[16];
    __u32 uDataBytes;
} CSMI_SAS_STP_PASSTHRU_STATUS,
  *PCSMI_SAS_STP_PASSTHRU_STATUS;

typedef struct _CSMI_SAS_STP_PASSTHRU_BUFFER {
    IOCTL_HEADER IoctlHeader;
    CSMI_SAS_STP_PASSTHRU Parameters;
    CSMI_SAS_STP_PASSTHRU_STATUS Status;
    __u8  bDataBuffer[1];
} CSMI_SAS_STP_PASSTHRU_BUFFER,
  *PCSMI_SAS_STP_PASSTHRU_BUFFER;

// CC_CSMI_SAS_GET_SATA_SIGNATURE

typedef struct _CSMI_SAS_SATA_SIGNATURE {
    __u8  bPhyIdentifier;
    __u8  bReserved[3];
    __u8  bSignatureFIS[20];
} CSMI_SAS_SATA_SIGNATURE,
  *PCSMI_SAS_SATA_SIGNATURE;

typedef struct _CSMI_SAS_SATA_SIGNATURE_BUFFER {
    IOCTL_HEADER IoctlHeader;
    CSMI_SAS_SATA_SIGNATURE Signature;
} CSMI_SAS_SATA_SIGNATURE_BUFFER,
  *PCSMI_SAS_SATA_SIGNATURE_BUFFER;

// CC_CSMI_SAS_GET_SCSI_ADDRESS

typedef struct _CSMI_SAS_GET_SCSI_ADDRESS_BUFFER {
    IOCTL_HEADER IoctlHeader;
    __u8  bSASAddress[8];
    __u8  bSASLun[8];
    __u8  bHostIndex;
    __u8  bPathId;
    __u8  bTargetId;
    __u8  bLun;
} CSMI_SAS_GET_SCSI_ADDRESS_BUFFER,
   *PCSMI_SAS_GET_SCSI_ADDRESS_BUFFER;

// CC_CSMI_SAS_GET_DEVICE_ADDRESS

typedef struct _CSMI_SAS_GET_DEVICE_ADDRESS_BUFFER {
    IOCTL_HEADER IoctlHeader;
    __u8  bHostIndex;
    __u8  bPathId;
    __u8  bTargetId;
    __u8  bLun;
```

```
    __u8  bSASAddress[8];
    __u8  bSASLun[8];
} CSMI_SAS_GET_DEVICE_ADDRESS_BUFFER,
  *PCSMI_SAS_GET_DEVICE_ADDRESS_BUFFER;

// CC_CSMI_SAS_TASK_MANAGEMENT

typedef struct _CSMI_SAS_SSP_TASK_IU {
    __u8  bHostIndex;
    __u8  bPathId;
    __u8  bTargetId;
    __u8  bLun;
    __u32 uFlags;
    __u32 uQueueTag;
    __u32 uReserved;
    __u8  bTaskManagementFunction;
    __u8  bReserved[7];
    __u32 uInformation;
} CSMI_SAS_SSP_TASK_IU,
  *PCSMI_SAS_SSP_TASK_IU;

typedef struct _CSMI_SAS_SSP_TASK_IU_BUFFER {
    IOCTL_HEADER IoctlHeader;
    CSMI_SAS_SSP_TASK_IU Parameters;
    CSMI_SAS_SSP_PASSTHRU_STATUS Status;
} CSMI_SAS_SSP_TASK_IU_BUFFER,
  *PCSMI_SAS_SSP_TASK_IU_BUFFER;

// CC_CSMI_SAS_GET_CONNECTOR_INFO

typedef struct _CSMI_SAS_GET_CONNECTOR_INFO {
    __u32 uPinout;
    __u8  bConnector[16];
    __u8  bLocation;
    __u8  bReserved[15];
} CSMI_SAS_CONNECTOR_INFO,
  *PCSMI_SAS_CONNECTOR_INFO;

typedef struct _CSMI_SAS_CONNECTOR_INFO_BUFFER {
    IOCTL_HEADER IoctlHeader;
    CSMI_SAS_CONNECTOR_INFO Reference[32];
} CSMI_SAS_CONNECTOR_INFO_BUFFER,
  *PCSMI_SAS_CONNECTOR_INFO_BUFFER;

// CC_CSMI_SAS_PHY_CONTROL

typedef struct _CSMI_SAS_CHARACTER {
    __u8  bTypeFlags;
    __u8  bValue;
} CSMI_SAS_CHARACTER,
  *PCSMI_SAS_CHARACTER;

typedef struct _CSMI_SAS_PHY_CONTROL {
    __u8  bType;
    __u8  bRate;
    __u8  bReserved[6];
    __u32 uVendorUnique[8];
    __u32 uTransmitterFlags;
    __i8  bTransmitAmplitude;
    __i8  bTransmitterPreemphasis;
    __i8  bTransmitterSlewRate;
    __i8  bTransmitterReserved[13];
    __u8  bTransmitterVendorUnique[64];
```

```
    __u32 uReceiverFlags;
    __i8  bReceiverThreshold;
    __i8  bReceiverEqualizationGain;
    __i8  bReceiverReserved[14];
    __u8  bReceiverVendorUnique[64];
    __u32 uPatternFlags;
    __u8  bFixedPattern;
    __u8  bUserPatternLength;
    __u8  bPatternReserved[6];
    CSMI_SAS_CHARACTER UserPatternBuffer[16];
} CSMI_SAS_PHY_CONTROL,
  *PCSMI_SAS_PHY_CONTROL;

typedef struct _CSMI_SAS_PHY_CONTROL_BUFFER {
    IOCTL_HEADER IoctlHeader;
    __u32 uFunction;
    __u8  bPhyIdentifier;
    __u16 usLengthOfControl;
    __u8  bNumberOfControls;
    __u8  bReserved[4];
    __u32 uLinkFlags;
    __u8  bSpinupRate;
    __u8  bLinkReserved[7];
    __u32 uVendorUnique[8];
    CSMI_SAS_PHY_CONTROL Control[1];
} CSMI_SAS_PHY_CONTROL_BUFFER,
  *PCSMI_SAS_PHY_CONTROL_BUFFER;

#pragma CSMI_SAS_END_PACK

#endif // _CSMI_SAS_H_
```