CISS
Command Interface for SCSI-3 Support Open Specification

Version 1.04

Valence Number 1

COMPAQ
Inspiration Technology

Version	Date	Changes
1.04	11/27/00	Introduced Open version of CISS specification.

1. General Description

The intent of this document is to define an interface between hosts and intelligent host bus controllers consistent with SCSI-3 addressing, command, and error status mechanisms. The specification defines a command passing protocol. It is beyond the scope of this document to define behaviors associated with physical operation such as device timeouts.

This document presents the command structure along with details on the addressing scheme, supported commands and supported messages. The last part of this document presents possible physical layers over which this command interface may be implemented. The physical layer implemented in a particular controller is dependent on the bridge interface present.

Unless otherwise stated all physical memory constructs are considered to be contiguous.

2. Normative References

- T10 Specifications
    - SCSI Architecture Model – 2 (SAM-2)
    - SCSI Primary Commands – 2 (SPC-2)
    - SCSI Controller Commands – 2 (SCC-2)
- Peripheral Component Interconnect (PCI) Bus Specification revision 2.1
- IEEE Standard for Communicating Among Processors and Peripherals Using Shared Memory (Direct Memory Access – DMA) – IEEE 1212.1
- Compaq Host-Based PCI Array Controller Firmware Specification, April 21, 1999

3. Command Structure

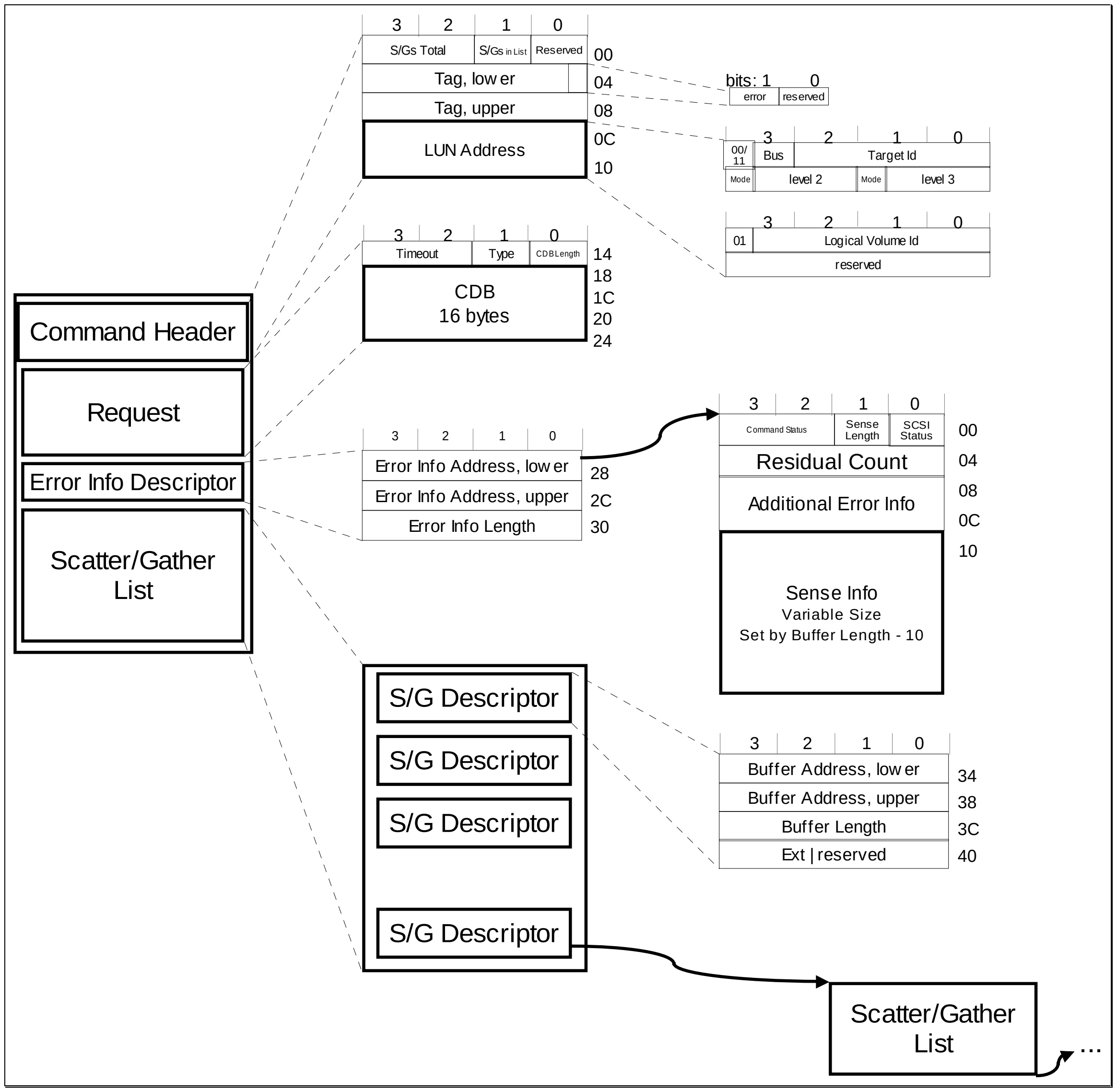

Figure 1 Command Format

The command structure is composed of 4 sections: header, request, error info descriptor, and scatter/gather list. The command structure itself is physically contiguous in memory, the error information is pointed to from within this structure and the scatter/gather lists may be chained.

3.1. Command Header

The **command header** defines the command length and addressing.

Scatter/Gather's Total indicates the total number of scatter/gather descriptors that are contained in all lists, as part of the command structure and chained.

Scatter/Gather's in List indicates the number of scatter/gather descriptors that are contiguously contained in the command structure.

Tag is 8-byte value that allows drivers a convenient method to locate the command structure upon completion. The driver may use this field any way that is convenient to re-access the command, i.e. logical address, array index. The Tag must be unique for every command outstanding in the controller as it is used to reference commands for aborts, etc. The Tag field is the primary completion response, this is the field that the controller will post to the host to indicate command completion. The lowest two bits have special meaning. Bit 1 is set to 1 in the response phase to indicate that an error has occurred during command processing and the host should check the error info block for further information. Bit 0 is reserved. Physical layers may use it as needed. Check with your physical layer definition for correct usage.

The **Logical unit number address** block is an eight-byte structure. It can be used in three modes of addressing, Logical Volume, Peripheral Device, and Mask Peripheral Device, specified by the upper two bits of the structure. Peripheral Device addressing is mode 00, 6 bits are used to specify a bus and the remaining 24 bits are used to specify a unique peripheral device on that bus. The next two bytes are two levels of SCSI-3 LUN addresses to allow addressing of lower tier devices. Mask Peripheral Device addressing is mode 11, it is identical to Peripheral Device addressing and simply gives an indication to the port driver to not export the LUN for logical I/O purpose. Logical Volume addressing is mode 01, 30 bits are used to address the controller’s logical volumes. The remaining 4 bytes are unused. The controller is accessed in mask peripheral mode with a bus value of zero and target id of zero. Hard drives that are candidates to be in RAID sets managed by the controller are referenced in level 2 of the LUN field with the first level being mask peripheral, bus=0, target id=0. The host except in configuration and management cases should not manipulate these drives.

3.2. Request Block

A command may have exactly one **request block**. The request block contains a command descriptor block, command time out, command type, and CDB length.

Timeout is a 2-byte field designating the number of seconds that a command should be given in normal condition for completion. The timer starts when the command is submitted to the controller’s target, it gives the controller a guideline for which it may timeout the command and submits a retry. This is not a guarantee that requests will complete within the specified amount of time. The actual request may take longer. If the host no longer wants to wait for a command to complete it may send an abort to remove that command from controller execution. A zero value in the timeout field indicates that the controller should not time the command out.

The **Type** field indicates the command type in the command descriptor to follow. The upper two bits of the Type field indicate direction. Bits 3- 5 indicate the task attribute for the command. The lower 3 bits indicate whether the command is a command, a message, or others to be defined later.

Table 1 Type/Direction Field

	Description	7	6	5	4	3	2	1	0
Direction	**None**	**0**	**0**						
	Write	**0**	**1**						
	Read	**1**	**0**						
	Reserved	**1**	**1**						
Attribute	**Untagged**			**0**	**0**	**0**			
	Simple			**1**	**0**	**0**			
	Head of Queue			**1**	**0**	**1**			
	Ordered			**1**	**1**	**0**			
	Auto Contingent Allegiance			**1**	**1**	**1**			
Type	**Command**						**0**	**0**	**0**
	Message						**0**	**0**	**1**
	Reserved						To be defined in future		

CDB length designates the size of the actual CDB. The CDB field is always a fixed size of 16 bytes; the command placed in this field might not take up all 16 bytes.

The next block is 16 bytes of data that depend on the Type field for its contents. Typically it is a **command descriptor block** of standard SCSI format of up to 16 bytes, supported commands are detailed later in this document. Otherwise it can be a message block, also defined elsewhere in this spec. The standard format for SCSI CDBs is shown in the following table:

Byte	**Field**	**Field breakdown**							
		7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
0	*Operation code*	Group Code			Command Code				
1	*Command specific parameters*								
n-1									
N	*Control*	Vendor Specific		Reserved			NACA	Link	Flag

The SCSI definitions for Group codes are:

Group Code	**Meaning**
0	6 byte commands
1	10 byte commands
2	10 byte commands
3	Reserved
4	16 byte commands
5	12 byte commands
6	Vendor specific

7	Vendor specific

The vendor specific group code of 6 is used for vendor unique commands in CISS. These commands start at 0xC0.

Command linking is not supported; the Link bit must be zero. Flag bit must be zero.

3.3. Error Info

The **error info descriptor** is a modified scatter/gather descriptor pointing to the error information block. The error info block can be located anywhere in 64-bit physical memory, the sense buffer size can be host defined based on the Error Info Length minus the starting address of the sense info.

The error info block is only valid if the error bit in the Tag field returned by the controller has the error bit set. The controller will not access this block at all if no error occurs.

The **command status** is a 2-byte field giving an overall status of the completed command. A zero value in this field indicates that the command completed successfully.

Table 2 Command Status

Value	**Error**	**Description**
0	Success	Command completed successfully.
1	Target Status	Command completed, but error has occurred.
2	Data Underrun	Command completed with data underrun.
3	Data Overrun	Data overrun has occurred.
4	Invalid Command	One or more fields in command have invalid data.
5	Protocol Error	Error occurred in communication phase accessing backside device.
6	Hardware Error	Error in hardware.
7	Connection Lost	Connection cannot be reestablished with backside device.
8	Command Aborted	Host initiated abort – successful.
9	Command Abort Failed	Host initiated abort – failed.
10	Command Unsolicited Abort	Controller could not complete command decided to abort it.
11	Command Timeout	Command did not complete controller gave up.
12	Command Unabortable	Host initiated abort – command in state of processing on the controller that is not abortable.
13-65535	Reserved	

Sense length is a 1-byte field designating the amount of the sense info buffer that is valid.

SCSI status is a 1-byte field. The values for this field are SCSI standard.

Table 3 SCSI Status

SCSI Status	Value (hex)
Good	00
Check Condition	02
Condition Met	04
Busy	08
Intermediate	10
Intermediate-Condition Met	14
Reservation conflict	18
Command terminated	22
Queue full	28
ACA active	30

Note that the host will never receive a SCSI Status of Good as the controller will not fill out this field at all when a good completion occurs.

Residual count is a 4-byte value that tells the size of the residual data when an under-run or over-run occurs. Residual count is a positive number used in conjunction with the Command Status, where Command Status indicates either under-run or over-run.

Additional error info is an 8-byte field for troubleshooting purposes. It can contain error codes for various drivers in a given controller.

The **sense info buffer** contains the results of a request sense. The amount of data that is valid agrees with the sense length value. The buffer size is host specified based on the error info length minus the starting offset of the sense info buffer.

3.4. Scatter/Gather List

The **scatter/gather list** has zero or more **scatter/gather elements**. The scatter/gather descriptors follow the IEEE 1212.1 Block Vector Structure specification. Within each scatter/gather element are a 4-byte buffer length and an 8-byte buffer address. The address can represent anywhere in 64-bit addressable space. There is also a 4-byte reserved field, for alignment sake, with the most significant bit defined as the Extension Bit. The Extension Bit designates the descriptor as pointing to a chained buffer of scatter/gather descriptors. Only the last scatter/gather descriptor may chain, it does not have to chain. A chained scatter/gather list may chain to another scatter/gather list. The end of the scatter/gather list is realized by matching the scatter/gather count.

A buffer length of zero signifies that no data is transferred for that scatter/gather element. It does not signify end of list nor have any other special meaning.

A buffer address of all F's, lower equals 0xFFFFFFFF and upper equals 0xFFFFFFFF, signifies bit bucketing of that scatter/gather element's data. The controller performs the operation indicated in the command but data buffer is not in host memory, data may be discarded immediately. One or more elements containing this address may occur anywhere in a scatter/gather list and has no further special meaning.

4. Logical Unit Number Addressing

Logical Unit Number addressing is separated between logical and physical devices. Both use the eight byte LUN field in the command structure but have different definitions for the field as laid out earlier. Only logical volumes of RAIDed drives are accessed through the Logical LUN field, mode bits b10, and all physical units, array controllers, hard drives, tape, etc are access through the Physical LUN field, mode bits b00 or b11. In the physical LUN field, level 0 represent add devices directly accessible to the controller. For a Fibre infrastructure, level 0 represents all devices logged into the controller. Level 1 and 2 are used to access devices behind this device. In addition, all physical devices that may not be exported to upper host OS layers for the purpose of logical I/O are accessed through the Physical LUN field with mode bits indicating the device as Masked (b11).

The controller may act as a Storage Array Conversion Layer (SACL), as defined in SCSI Controller Commands-2. Literally this means that logical unit identifiers and logical block addresses in might not equal logical unit identifiers and logical block addresses out. Also single logical unit identifiers and logical block addresses may be converted to multiple different logical unit identifiers and logical block addresses. Practically this means that the controller may group and remap its targets for the view it presents to the host.

For the purpose of discovery, the host uses Report Logical LUNs to discover logical drives and Report Physical LUNs to discover physical devices. These commands are discussed in section 1.1 of this specification.

The following diagram depicts CISS' LUN worldview:

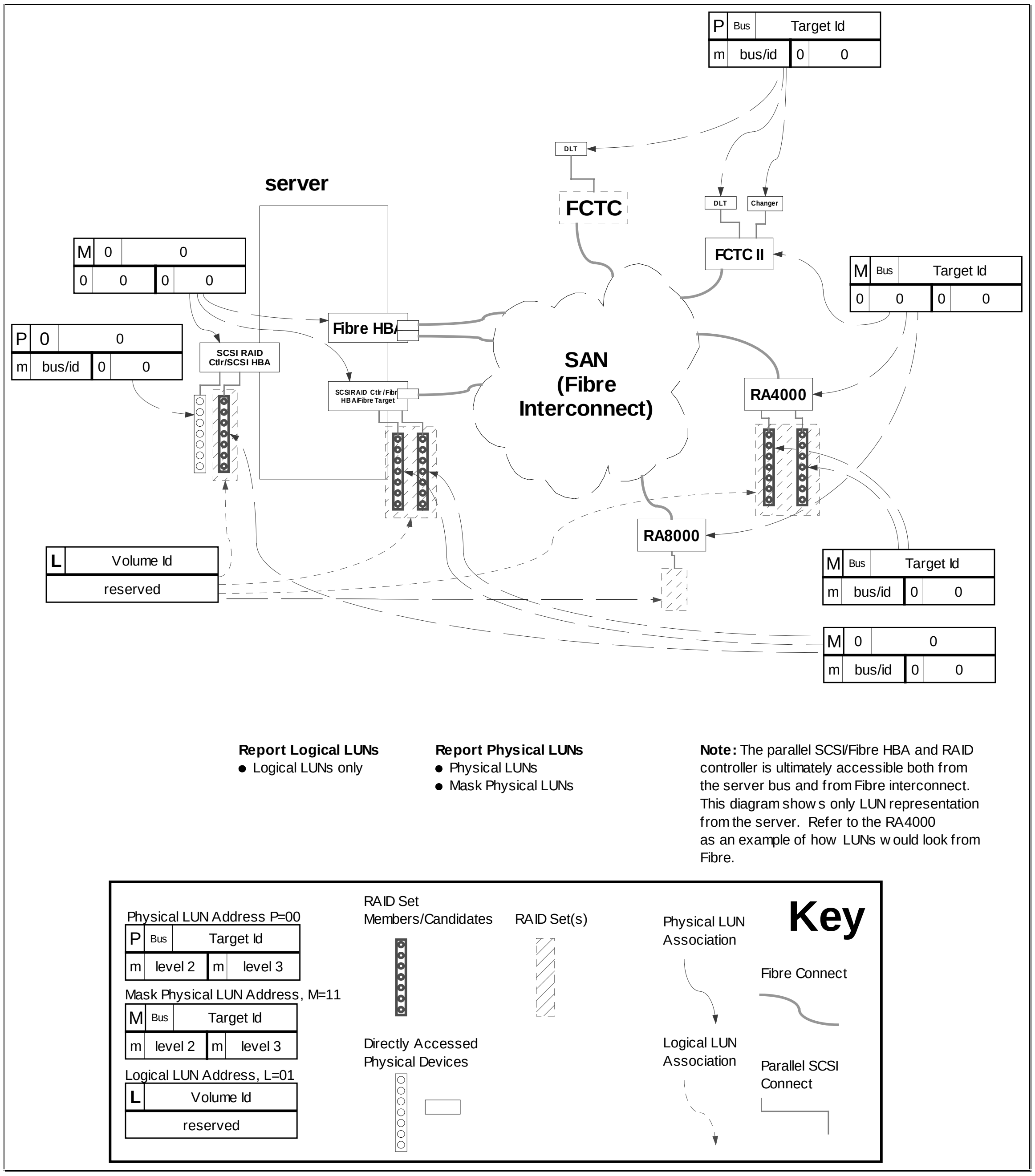
server
Fibre HBA
SCSI RAID Ctlr/SCSI HBA
SCSI RAID Ctlr/Fibre HBA/Fibre Target
SAN (Fibre Interconnect)
FCTC
FCTC II
DLT
Changer
RA4000
RA8000
P Bus Target Id
m bus/id 0 0
M 0 0
M Bus Target Id
L Volume Id
reserved
Report Logical LUNs
● Logical LUNs only
Report Physical LUNs
● Physical LUNs
● Mask Physical LUNs
Note: The parallel SCSI/Fibre HBA and RAID controller is ultimately accessible both from the server bus and from Fibre interconnect. This diagram shows only LUN representation from the server. Refer to the RA4000 as an example of how LUNs would look from Fibre.
Key
Physical LUN Address P=00
P Bus Target Id
m level 2 m level 3
Mask Physical LUN Address, M=11
M Bus Target Id
m level 2 m level 3
Logical LUN Address, L=01
L Volume Id
reserved
RAID Set Members/Candidates
RAID Set(s)
Directly Accessed Physical Devices
Physical LUN Association
Logical LUN Association
Fibre Connect
Parallel SCSI Connect

Figure 2 CISS' LUN Worldview

5.

5. Task Sets

CISS implements the SCSI-3 model of task sets as defined in the SCSI Architecture Model – 2 Specification. Each target, logical or physical, has exactly one task set. Commands may not be linked so each task is represented by exactly one command. Task attributes are specified in each command in the Type field of the command structure.

A CISS controller supporting task sets will indicate so by the basic queue (BQue in SPC-2) and command queue (CmdQue in SPC-2) bits in inquiry command responses. The basic queue bit set indicates that:

a) Only the task attribute of Simple is supported;

b) The controller may reorder command execution sequence in any manner, the application client is responsible for data integrity;

c) All blocked commands in the task set shall be aborted after an ACA or CA condition is cleared;

d) It is not possible to disable tagged queuing; and

e) Support for the Clear Task Set Reset message is optional.

The basic queue bit may be set only if the command queue bit is cleared, both set is an illegal combination. The basic queue bit and command queue bit both zeroed indicates that no command queuing is supported. Command queue bit set indicates that the full command queuing with all task attributes are supported.

Task attributes have the following definitions, from SAM-2:

Non-Tagged – No attributes, not considered part of task set.

Simple – A task having the Simple attribute shall be accepted into the task set in the Dormant state. The task shall not enter the Enabled state until all older Head of Queue and older Ordered tasks in the task set have ended.

Ordered – A task having the Ordered attribute shall be accepted into the task set in the Dormant state. The task shall not enter the Enabled state until all older tasks in the task set have ended.

Head of Queue – A task having the Head of Queue attribute shall be accepted into the task set in the Enabled state.

Auto Contingent Allegiance (ACA) – A task having the ACA attribute shall be accepted into the task set in the Enabled state. There shall be no more than one ACA task per task set. Not supported in CISS.

6. Controller

6.1. Identification

CISS controllers may be recognized by settings in the PCI Device Configuration Header. Vendor Id will typically indicate Compaq, Device Id will indicate the interface chip, and Subsystem Id will distinguish the controller product. The configuration header will appear as follows:

	3	2	1	0
00	Device Id		Vendor Id	
Defined per Bridge ASIC	B060		0E11	
04	Status Register		Command Register	
	----		----	
08	Class Code	Sub-Class Code	Programming I/F	Revision Id
Common values, not exclusive list	01 – Mass Storage	04 – RAID		
	…			
2C	Subsystem Id		Subsystem Vendor Id	
Defined per product as needed	4070 – Smart Array 5300		0E11	

6.2. Initialization

CISS controllers are initialized in the following manner:

a) Determine that PCI device supports CISS by matching the Subsystem Id and Subsystem Vendor Id registers in the PCI Device Configuration Header with known CISS controller values.

b) Initialize first 64-bytes of the PCI Device Configuration Header in standard PCI fashion.

c) Check CISS Configuration Table for Transport Methods supported by controller.

d) Select Transport Method to use.

e) Initialize CISS Configuration Table.

f) Begin I/O.

6.3. Hard Reset

CISS controllers may be hard reset via their Power Management new capabilities registers. These registers are in the PCI Device Configuration Registers.

To locate these registers:

a) Check the Primary PCI Status Register, 2-bytes at offset 06h. Bit 4 set indicates that the device supports New Capabilities.

b) Read the Capabilities List Pointer, 1-byte at offset 34h. This register has the offset for the first item in the new capabilities linked list.

Each new capabilities element has the following:

3	**2**	**1**	**0**
Capability specific		Next Capabilities Pointer	Capabilities ID
Possibly more capability specific fields			

c) Check Capabilities ID for Power Management Capabilities ID, 01h.

  a. If match then found Power Management Capabilities registers.

b. Else, get offset for next Capabilities ID from the Next Capabilities Pointer register. Continue through the new capabilities linked list until the Power Management Capabilities registers are found or the Next Capabilities Pointer is zero indicating end of list. If no Power Management Capabilities, then out of luck.

The Power management capabilities registers look like:

3	2	1	0
Power Management Capabilities		Next Capabilities Pointer	Capabilities ID
Power Management Data Register	Power Management Control/Status Register, Bridge Support Extensions	Power Management Control/Status Register	

d) The Power Management Control/Status Register (CSR) controls the power state of the device. The normal operating state is D0, CSR=00h. The software off state is D3, CSR=03h. To reset the controller place the interface device in D3 then to D0, this causes a secondary PCI reset, which will reset the controller. The PCI Configuration Registers will not be preserved.

e) Save the PCI Configuration Registers, offsets 00h through 60h.

f) Write 00h to the Power Management CSR to turn off the interface device.

g) Write 03h to the Power Management CSR to turn the interface device back on.

h) Restore the PCI Configuration Registers, offsets 00h through 60h. It is important to restore the command register, 16-bits at offset 04h, last. Do not restore the configuration status register, 16-bits at offset 06h.

7. SCSI CDBs

This command interface is SCSI-3 compatible. This allows for a common interface that could be used for many different devices. Also is friendly to OS drivers that largely have to interface to a higher level SCSI layer.

7.1. Mandatory Commands

A controller that supports CISS is considered to be a SCSI storage array controller. Therefore, it must support mandatory commands as per *SCSI Primary Commands – 2 (SPC-2)* and *SCSI Controller Commands – 2 (SCC-2)*. The exception to this is configuration type commands and Report LUNs. SCSI configuration commands do not seem well suited to the needs of our controllers. Report LUNs does not fit well into our controller model. Commands Report Logical LUNs and Report Physical LUNs, 0xC2 and 0xC3 respectively, are substituted. Commands that are not mandatory by SCSI-3 may be mandatory for CISS. The mandatory commands are listed in the following table.

Command	Operation Code (hex)	Comments
Inquiry	12	
Read Buffer	3C	Shall be used to read controller's firmware. The controller's firmware buffer id shall be 0. Mode 010b, data, is mandatory.
Request Sense	03	As CISS uses auto-sense the response to this normally is "No Sense". However, Request Sense has specific meaning when used with Copy, Compare, and Copy and Verify. When used with these commands the response will contain

		the appropriate information.
Test Unit Ready	00	
Write Buffer	3B	Shall be used to upgrade controller firmware. The controller's firmware buffer id shall be 0. Modes 101b, Download micro code and save, and 111b, Download micro code with offsets and save, are mandatory, selection for use depends on the hosts data transfer capacity limitations. Binary firmware images are required to be protected with an 8-bit checksum, such that if all bytes of the image are summed byte wise the result is zero. The controller shall verify this checksum prior to committing the firmware change.
Report Logical LUNs	C2	Vendor unique command to discover logical devices.
Report Physical LUNs	C3	Vendor unique command to discover physical devices.

7.2. Mandatory Commands for Block Storage Devices

A CISS supporting controller that acts as or front-ends a SCSI block storage device will additionally support the following commands.

Command	**Operation Code (hex)**	**Comments**
Read (6)	08	
Read (10)	28	
Read (12)	A8	
Read Capacity	25	
Release (10)	57	
Receive Diagnostic Results	1C	Supported for SES
Reserve (10)	56	
Send Diagnostic	1D	Supported for SES
Write (6)	0A	
Write (10)	2A	
Write (12)	AA	

7.3. Additional Supported Commands

Any SCSI command may be used. It is up to individual controllers or targets addressed by the command as to whether the particular opcode is supported.

Vendor specific opcodes hex 26 and hex 27 are defined in this interface to be *Array Controller Read* and *Array Controller Write*, respectively. These opcodes are used for compatibility to previous generation controller commands. Many of these commands will be leveraged for physical actions such as configuration. Refer to the Compaq Array Controller Specification for further details on these commands.

Vendor specific opcodes hex C0 and C1 support commands new for the CISS interface. C0 specifies read type commands and C1 specifies write and control commands. Commands will be added for these opcodes as needed. The CDB format is:

BYTE	FIELD
0	Operation Code (C0h/C1h)
1	Command
2	Reserved
3	
4	Detail
5	
6	
7	
8	Length
9	
10	
11	
12	Control

Command specifies unique commands within the C0 and C1 opcodes. Detail specifies command unique details. Length specifies the data portion size if any associated with the command.

7.4. Inquiry Vital Product Data Pages

The following Inquiry Vital Product Data Pages are acceptable for a CISS controller. Page Code 00h is required.

7.4.1. Page Code 00h (Supported vital product data pages)

Bit / Byte	7	6	5	4	3	2	1	0
0	PERIPHERAL QUALIFIER			PERIPHERAL DEVICE TYPE				
1	PAGE CODE (00h)							
2	Reserved							
3	Page Length (n-3)							
4	SUPPORTED PAGE LIST							
N								

The Supported Vital Product data pages provide a list of the vital product data codes supported by the target or logical drive.

7.4.1.1. Peripheral Qualifier and Peripheral Device Type fields
Identifies the device currently connected to the logical unit.

7.4.1.2. Page Code field
The PAGE CODE field contains the same value as in the Page or Operation code field of the INQUIRY command descriptor block.

7.4.1.3. Page Length field
Indicates the length of the following page data.

7.4.1.4. Supported Page List field
Contains a list of all vital product data page codes implemented for the target or logical unit in ascending order beginning with page code 00h.

7.4.2. Page Code 83h (Device Identification)

Bit Byte	7	6	5	4	3	2	1	0
0	PERIPHERAL QUALIFIER			PERIPHERAL DEVICE TYPE				
1	PAGE CODE (83h)							
2	Reserved							
3	PAGE Length (n-3)							
	Identification Descriptor list							
4	Identification descriptor (first)							
	.							
	.							
	Identification descriptor (last)							
N								

The device identification page provides the means to retrieve zero or more identification descriptors applying to the logical unit. Logical units may have more than one identification descriptor. Two Identification Descriptors are used for the logical unit. The first provides the Unique ID for the logical unit, and the second provides the Group Number of the logical unit.

7.4.2.1. Peripheral Qualifier and Peripheral Device Type fields
Identifies the device currently connected to the logical unit.

7.4.2.2. Page Code field
The PAGE CODE field contains the same value as in the Page or Operation code field of the INQUIRY command descriptor block.

7.4.2.3. Page Length field
Indicates the length of the following page data.

7.4.2.4. Identification descriptor

Bit Byte	7	6	5	4	3	2	1	0
0	Reserved			CODE SET				
1	Reserved		ASSOCIATION		IDENTIFIER TYPE			
2	Reserved							
3	IDENTIFIER LENGTH (n-3)							
4	(MSB)							
N	IDENTIFIER							(LSB)

Each Identification Descriptor contains information identifying the logical unit.

- **CODE SET field** -The Code Set field specifies the code set used for the identifier field.
  - 0h – Reserved
  - 1h – The identifier field shall contain binary values

  - 2h – The identifier field shall contain ASCII graphic codes
  - 3h-Fh – Reserved
- **ASSOCIATION field** - The Association field specifies the entity which the Identifier is associated.
  - 0h – The Identifier field is associated with the addressed physical or logical device.
  - 1h – The Identifier field is associated with the port that received the request.
  - 2h-3h – Reserved
- **IDENTIFIER TYPE field -** The Identifier Type field specifies the format and assignment for the Identifier.
  - 0h – No assignment authority was used and consequently there is no guarantee that the Identifier is globally unique. (i.e., the identifier is vendor-specific).
  - 1h – The first 8 bytes of the identifier field are a Vendor ID.
  - 2h – The identifier field contains a Canonical IEEE Extended Unique Identifier, 64-bit.
  - 3h – The identifier field contains a FC-PH, FC-PH3 or FC-FS Name_Identifier.
  - 4h – If the ASSOCIATION value is 1h, the Identifier value contains a four-byte binary number Identifying the port relative to other ports in the device using the values shown bellow.
    - 0h – Reserved
    - 1h – Relative port 1, also known as port A
    - 2h – Relative port 2, also known as port B
    - 3h—7FFFFFFFh – Relative port 3 through 2 147 483 647
    - 80000000h-FFFFFFFFh – Reserved
- **IDENTIFIER LENGTH field -** The Identifier Length field specifies the length in bytes of the Identifier field.
- **IDENTIFIER -** The Identifier field contains the Identifier as described by the ASSOCIATION, IDENTIFIER TYPE, CODE SET, and IDENTIFIER LENGTH fields.

7.4.3. Page Code C0h (Vendor specific, Physical Device Identification)

Bit Byte	7	6	5	4	3	2	1	0
0	PERIPHERAL QUALIFIER			PERIPHERAL DEVICE TYPE				
1	PAGE CODE (C0h)							

2	Reserved
3	PAGE Length (n-3)
	Identification Descriptor list
4	Identification descriptor (first)
	.
	.
	Identification descriptor (last)
n	

The vendor specific device identification page provides the means to retrieve zero or more identification descriptors applying to the physical device. Physical devices may have more than one Identification Descriptor. Multiple Identification Descriptors are used for the physical device. The first provides the Node Name for the device, the next are the Port Name for each of the ports on the device.

7.4.3.1. Peripheral Qualifier and Peripheral Device Type fields

Identifies the device currently connected to the logical unit.

7.4.3.2. Page Code field

The PAGE CODE field contains the same value as in the Page or Operation code field of the INQUIRY command descriptor block.

7.4.3.3. Page Length field

Indicates the length of the following page data.

7.4.3.4. Identification descriptor

Bit Byte	7	6	5	4	3	2	1	0
0	Reserved			CODE SET				
1	Reserved		ASSOCIATION		IDENTIFIER TYPE			
2	Reserved							
3	IDENTIFIER LENGTH (n-3)							
4	(MSB)							
n	IDENTIFIER							(LSB)

Each Identification descriptor contains information identifying the physical device.

- **CODE SET field** - The Code Set field specifies the code set used for the identifier field.
  - 0h – Reserved
  - 1h – The identifier field shall contain binary values
  - 2h – The identifier field shall contain ASCII graphic codes
  - 3h-Fh – Reserved
- **ASSOCIATION field -** The Association field specifies the entity which the Identifier is associated.
  - 0h – The Identifier field is associated with the addressed physical or logical device.
  - 1h – The Identifier field is associated with the port that received the request.
  - 2h-3h – Reserved
- **IDENTIFIER TYPE field -** The Identifier Type field specifies the format and assignment for the Identifier.
  - 0h – No assignment authority was used and consequently there is no guarantee that the Identifier is globally unique. (i.e., the identifier is vendor-specific).
  - 1h – The first 8 bytes of the identifier field are a Vendor ID.
  - 2h – The identifier field contains a Canonical IEEE Extended Unique Identifier, 64-bit.
  - 3h – The identifier field contains a FC-PH, FC-PH3 or FC-FS Name_Identifier.

- 4h – If the ASSOCIATION value is 1h, the Identifier value contains a four-byte binary number Identifying the port relative to other ports in the device using the values shown below.
  - 0h – Reserved
  - 1h – Relative port 1, also known as port A
  - 2h – Relative port 2, also known as port B
  - 3h—7FFFFFFFh – Relative port 3 through 2 147 483 647
  - 80000000h-FFFFFFFFh – Reserved
- **IDENTIFIER LENGTH field -** The Identifier Length field specifies the length in bytes of the Identifier field.
- **IDENTIFIER -** The Identifier field contains the Identifier as described by the ASSOCIATION, IDENTIFIER TYPE, CODE SET, and IDENTIFIER LENGTH fields.

7.4.4. Page Code C1h (Vendor specific, Logical Drive Geometry)

Bit / Byte	7	6	5	4	3	2	1	0
0	PERIPHERAL QUALIFIER			PERIPHERAL DEVICE TYPE				
1	PAGE CODE (C1h)							
2	Reserved							
3	PAGE Length (n-3)							
	Logical Drive Parameters							
4	(MSB)							
5	Number of Cylinders (LSB)							
6	Number of Heads							
7	Number of Sectors							
8	Fault Tolerance							
9	Reserved							
10	Reserved							
11	Reserved							
n								

The Logical Drive Geometry page specifies parameters for direct-access devices.

7.4.4.1. Peripheral Qualifier and Peripheral Device Type fields

Identifies the device currently connected to the logical unit.

7.4.4.2. Page Code field

The PAGE CODE field contains the same value as in the Page or Operation code field of the INQUIRY command descriptor block.

7.4.4.3. Page Length field

Indicates the length of the following page data.

7.4.4.4. Fault Tolerance field

Indicates the fault tolerance of the logical drive

- 0h – No fault tolerance (“RAID 0”)
- 1h - Data Guard (RAID 4)
- 2h - Mirroring (RAID 1)
- 3h - Distributed Data Guard (RAID 5)
- 4h - RAID 5+1
- 5h – Advanced Data Guard (RAID ADG)
- 0xFF - Invalid data

7.4.4.5. Number of Cylinders field

Indicates the number of cylinders used for data storage.

7.4.4.6. Number of Heads field

Indicates the number of heads used for data storage.

7.4.4.7. Number of Sectors field

Indicates the number of sectors.

Note1: When the Fault tolerance field is set to 0xFF, the data from byte 4 to byte 8 should be considered invalid.

7.5. Asynchronous Event Reporting

Asynchronous Event Reporting is accomplished by the host posting a specific command to the controller, the controller holds this command until an event has occurred. This is done with the Notify on Event command hex D0 of opcode hex C0. The host may alternatively poll for commands by setting the synchronous bit in this command. To cancel the asynchronous event notifier, simply abort the command.

7.5.1. Notify on Event Command

CDB format:

BYTE	FIELD
0	CISS Read (C0h)
1	Notify on Event (D0h)
2 3	Reserved
4 5 6 7	Bits 16-31(MSB): Timeout for asynchronous events in seconds Bits 4-15: Reserved. Bit 3: If set, host will not receive notification of any events that occurred prior to the current time. Bit 2: If set, controller event pointer is reset to point to oldest event logged – may be non-volatile. Bit 1: If set, events are read, in the order in which they were saved. Bit 0 (LSB): 0 – asynchronous mode 1 – synchronous mode
8	512

9	
10	
11	
12	Control

Return Data:

Byte	Name	Meaning
0-3	Relative Controller Time	This is a time stamp in seconds since power has been applied to the controller or since the last controller reset, whichever occurred more recently.
4-5	Event Class Code.	See Event Code Table, next.
6-7	Event Sub-Class Code.	See Event Code Table, next.
8-9	Event Detail Code.	See Event Code Table, next.
10-73	Event Specific Data Fields.	The Event Specific data fields are reserved for specific data structures associated with certain events. The meaning of the fields in the data structure is dependent upon the particular event.
74-153	Null-Terminated ASCII Message.	The Null-Terminated ASCII Message is optionally (determined by the controller) returned to the host for specific user messages.
154-157	Event tag	This value provides a unique numeric identifier for each logged event. Useful in a multi-host environment to prevent duplicate reporting of single events. NOTE: Events returned from the Event Notifier Protocol class will have this field set to 0.
158-159	Event time (month and day)	Month and day in the 0xmmdd format. Date is valid only if controller has not been reset after the last system boot; if controller was reset, the event time fields will all be 0. Valid only for controllers with real-time clocks, zeroes otherwise.
160-161	Event time (year)	Valid only for controllers with real-time clocks, zeroes otherwise.
162-165	Event time (hour/min/sec)	This is the time in seconds since midnight. It is reset to zero at midnight. Time is not returned if the controller is reset after boot. Valid only for controllers with real-time clocks, zeroes otherwise.
166-167	Pre-power up time	Indicates that this event occurred prior to the controller powering up. Most useful for HBA type controllers where the event occurred at the target device prior to the HBA powering up. This field is in seconds.
168-175	Device address	8-byte LUN structure of target device that experienced event.
176-511	Reserved	Pad to 1 block.

Event codes are listed in Appendix B in this specification.

7.6.

7.6. Discovery Commands

7.6.1. Report Logical LUNs Command

Command to discover logical devices. The response to this command is a list of 8-byte unique handles or logical unit numbers that are accessible by the host. Only logical drives will appear in this list.

Command:

Byte	Field
0	Operation Code (C2h)
1	Reserved
2	Reserved
3	Reserved
4	Reserved
5	Reserved
6	(MSB)
7	Allocation Length in Bytes
8	
9	(LSB)
10	Reserved
11	Control

Response:

Byte	Field
0	(MSB)
1	LUN List Length (N-7)
2	
3	(LSB)
4	(MSB)
5	Reserved
6	
7	(LSB)
LUN List	

Offset/byte
3
2
1
0
8
01
Logical Volume Id
12
reserved
…
N-7
01
Logical Volume Id
N-3
reserved

The LUN list length field shall contain the length in bytes of the LUN list that is available to be transferred. The LUN list length is the number of logical unit numbers reported multiplied by eight. If the allocation length in the command descriptor block is too small to transfer information about all configured logical units, the LUN list length value shall not be adjusted to reflect the truncation. Note that the response is similar in format to the SCSI Report LUNs command with the exception that the LUN fields match the LUN format in the CISS command structure.

7.6.2. Report Physical LUNs Command

Command to discover physical devices. This command will return a list of 8-byte peripheral addresses for physical devices. All peripheral devices that are accessible to the host will appear in this list be they Physical or Mask Physical devices. This includes devices behind a bridge.

Command:

BYTE	FIELD
0	Operation Code (C3h)
1	Reserved
2	Reserved

3	Reserved
4	Reserved
5	Reserved
6	(MSB) Allocation Length in Bytes
7	
8	
9	(LSB)
10	Reserved
11	Control

Response:

Byte	Field
0	(MSB) LUN List Length in Bytes (N-7)
1	
2	
3	(LSB)
4	(MSB) Reserved
5	
6	
7	(LSB)
LUN List	

Offset/byte
3
2
1
0
8
00
Bus
Target Id
12
M
Level 2
M
Level 3
…
N-7
00
Bus
Target Id
N-3
M
Level 2
M
Level 3

The LUN list length field shall contain the length in bytes of the LUN list that is available to be transferred. The LUN list length is the number of logical unit numbers reported multiplied by eight. If the allocation length in the command descriptor block is too small to transfer information about all configured logical units, the LUN list length value shall not be adjusted to reflect the truncation. Note that the response is similar in format to the SCSI Report LUNs command with the exception that the LUN fields match the LUN format in the CISS command structure.

8. Messages

Messages provide a way to send non-CDB style directives.

8.1. Supported Messages

Message	Opcode (hex)
Abort	00
Reset	01
Scan	02
No-op	03

8.2. Description

The descriptions to follow define the format of the CDB field in the command structure. An escalation policy for Aborts/Resets is as follows in order of increasing granularity:

- Abort Task – aborts only command specified.
- Abort Task Set – aborts all commands outstanding to specified target, target may be logical or physical.
- Clear ACA – clears auto contingent allegiance condition. ACA condition is not supported by CISS controllers, though may be supported by intelligent devices accessed through a CISS controller.
- Clear Task Set – aborts all commands outstanding to specified target, target may be logical or physical. In the CISS implementation Abort Task Set and Clear Task Set shall be equivalent.
- Reset Logical Unit – aborts all tasks in the task set and propagate to all dependent logical units.
- Reset Target – hardware reset of specified target.
- Reset Bus – hardware reset of all devices on bus.
- Reset Controller – hardware reset of controller.

All commands affected by aborts and resets must be completed to the host before the completion for the abort or reset is posted to the host. The host can then be confident that the action is complete.

The Scan message is a directive to scan the indicated entity. Scan allows the host to assert its belief that one or more targets exists, usually newly created, of which the controller is unaware. The controller should pay heed to the hosts asserted belief by checking for itself the indicated bus or target. The host must learn of the controller's target awareness with the discovery commands, Report Logical/Physical LUNS.

The No-op message is a mechanism for the host to check the health of the controllers command deliver/response path.

8.2.1. Abort – 0x00

Used to abort commands. The target is specified in the LUN field for abort task set, clear ACA, and clear task set. The id field is used to specify the command tag for an Abort Task.

BYTE	MEANING	EXPLANATION		
0	Opcode	0x00 – Indicates an Abort message.		
1	Type	**Type**	**Code (hex)**	**Explanation**
		Task	00	Abort command
		Task Set	01	Aborts all commands to a target
		Clear ACA	02	Clears ACA condition
		Clear Task Set	03	Aborts all commands to target
2-3	Reserved	Set to zero.		
4-11	Task	Tag of command to abort; Abort Task. Unused otherwise.		
12-15	Reserved	Set to zero.		

8.2.2. Reset – 0x01

Message to reset one of many objects. The target is specified in the LUN field.

BYTE	MEANING	EXPLANATION		
0	Opcode	0x01 – Indicates a Reset message.		
1	Type	**Type**	**Code (hex)**	**Explanation**
		Controller	00	Reset controller
		Bus	01	Reset bus
		Target	03	Reset target
		Logical Unit	04	Aborts all tasks in task set and propagates to all dependent logical units
2-15	Reserved	Set to zero		

8.2.3. Scan – 0x02

Message to scan one of many objects. The target is specified in the LUN field.

BYTE	MEANING	EXPLANATION		
0	Opcode	0x02 – Indicates a Scan message.		
1	Type	**Type**	**Code (hex)**	**Explanation**
		All	00	Scan all busses
		Bus	01	Scan bus
		Target	03	Scan target
		Logical Unit	04	Scan logical unit
2-15	Reserved	Set to zero		

8.2.4. No-op – 0x03

Simply prompts controller to reply. No error is possible.

BYTE	MEANING	EXPLANATION
0	Opcode	0x03 – Indicates a No-op message.
1-15	Reserved	Set to zero

9. Transport Layer

This section describes the transport layers supported by the command interface. CISS' transport layer has a table in controller memory that is used to communicate interface version and initialization information.

The methods described work for Compaq's current internal line of bridges. Differences between the bridges are noted in the register descriptions.

Controller operations and registers only accessed by controller are shaded for clarity.

9.1. Configuration Table

The Configuration Table is used to initialize the command interface. Its fields are spelled out in the table below. Its location is given by doing a mem cycle read from offset 0xB4 (CTCfgOffset) and offset 0xB8 (CTMemOffset) from the I2O BAR. CTCfgOffset identifies which config offset is used to determine the base address for the table in the lower 16 bits and a width identifier in the upper 16 bits. CTMemOffset is used to determine the memory offset from this base address. For example, if CTCfgOffset contains 0x00000014 and CTMemOffset contains 0x00000000, then the Configuration Table is located at offset 0x00000000 from the 32-bit Memory BAR 1. If CTCfgOffset contains 0x00010010 and CTMemOffset contains 0x00004000, then the Configuration Table is located at offset 0x00004000 from the 64-bit Memory BAR 0. The method of access is for the host to read the table to ensure controller has passed a certain power-up initialization point. It then may read the fields it desires. To write the table the host must respect the implied read only nature of certain fields; there is no explicit write protection. Once the host sets the table it sets bit 0 in the Inbound Doorbell Register, effectively interrupting the controller to inform of the update. The host must then check the Inbound Doorbell Status until the controller clears it. The host may now assume that the controller will operate in the directed capacity.

Offset (hex)	Field	Write Access	Details
00	ASCII Signature- "CISS"	Controller	Indicates coherent table
04	Specification Valence Number	Controller	Used to determine compatibility with versions of this specification
08	Supported Transport Methods	Controller	Bit field to indicate transport methods supported by controller. Bit 0 set indicates that the controller has initiated to the point it will accept commands. **Bit** **Transport Method** 0 Indicate ready to accept commands 1 Simple Method 2-31 Reserved for future use
0C	Active Transport Method	Controller	Bit field to indicate current active transport method.
10	Requested Transport Method	Host	Bit field to indicate transport method requested.
14	Command upper 32-bit address	Host	If controller hardware requires commands to be in 4-Gig naturally bound region, this field specifies the upper 32-bit address of this region.
18	Coalesce Interrupt Delay	Host	Time, in microseconds, to delay interrupt to host. Zero means not to delay based on time. An interrupt is passed to host when the either of the Delay or Count parameters is satisfied. Default value is 0.
1C	Coalesce Interrupt Count	Host	Number of interrupts required before passing to host. Zero means not to coalesce based on count. One means to pass all interrupts directly to the host. An interrupt is passed to host when the either of the Delay or Count parameters is satisfied. Default value is 1. **Note:** Values of zero for both Delay and Count specify controller will be used in a polled mode with interrupts disabled. Hosts **must** set appropriately.
20	Maximum outstanding commands supported	Controller	Number of commands the controller can have issued to it at any given instant.

24	Bus Types	Controller	Bit field to indicate backside bus types on this controller. **Bit** **Bus Type** 0 Parallel SCSI – Ultra 2 1 Parallel SCSI – Ultra 3 2-7 Reserved for future Parallel SCSI renditions 8 Fibre – 1 Gig-baud 9 Fibre- 2 Gig-baud 10-15 Reserved for future Fibre renditions 16-31 Reserved for future
28	Reserved	N/A	
2C – 3B	Reserved	N/A	
3C	Heartbeat Counter	Controller	Controller updates this running counter every second as an indicator to the host that it is running. The upper four bits in this field mirror the gas pedal LEDs indicating busyness of the controller in 25% increments.

9.1.1.

9.1.1.

9.1.1. Register Definitions

9.1.1.1. PCI Memory Base Address Register 1

32 bits wide, config address 0x14

This configuration register contains the base address of the Configuration Table. Controller should not allow pre-fetch to be enabled in this register.

9.1.1.2. Inbound Doorbell Register

32 bits wide, address 0x20, read/write

The host writes this register to set the inbound doorbell interrupt to the controller. Writing a one to any bit in this register sets an interrupt to the controller. Bit 0 is used for the Configuration Table. Reading this register will give status of interrupts set.

9.1.1.3. Inbound Doorbell Clear Register

32 bits wide, address 0x70, not accessible by host

The controller writes a one to a bit in this register to clear the corresponding inbound doorbell interrupt. Bit 0 is used for the Configuration Table.

9.2. Simple Method

9.2.1. Operation

The Simple Method uses the controller's Inbound Post Queue for command submission and Outbound Post Queue for command response. The host builds a command in host memory. The command's starting address must be on a 4-byte boundary. The physical address of the command is written to the Inbound Post Queue Register on the controller. This physical address is posted to the FIFO in controller memory at the location referred to by the Inbound Post List Tail Pointer. The Inbound Post List FIFO becomes non-empty causing an interrupt to be driven onboard the controller to the processor. The processor reacts to the interrupt, reads the value at the location referred to in the Inbound Post List FIFO by the Inbound Post List Head Pointer. This value is a physical address of a command in host memory. The processor then fetches one 4-byte entity from host memory at this physical address. This value is the length of the command. The processor then fetches the number of 4-byte entities designated by this length from host memory. Now the processor can process the command. The processor must increment the Inbound Post List Head Pointer. The processor shall continual to fetch commands until the Inbound Post List Tail Pointer equals the Inbound Post List Head Pointer.

Before a command response is sent the controller must write any error information to the command's error info block in host memory. The controller reads the tag field from its local copy of the command. If an error has occurred the controller sets bit one of the tag. The controller writes the tag field to the location in the Outbound Post List FIFO referred to by the Outbound Post List Tail Pointer. The controller must increment the Outbound Post List Tail Pointer. When the Outbound Post List FIFO becomes non-empty, an interrupt is driven to the host. The host reacts to the interrupt by reading the Outbound Post Queue Register. This read retrieves the oldest element in the Outbound Post List FIFO and causes the Outbound Post List Head pointer to increment. The tag is used to reference the command in host memory. The host must then check bit one in the tag, this bit will be set if an error has been reported. If the error bit is set, the host may look in the error info block for further error detail and react to the error. Finally the host will consider the command complete and free the memory the command resides in. The host must then read the Outbound Post Queue again to get the next completion. The host must continue to retrieve completions from the Outbound Post Queue until a value of 0xFFFFFFFF is read. This signals that the Outbound Post Queue is empty. On the queue becoming empty the interrupt is de-asserted.

Note that only the first 4 bytes of the Tag field are used in Simple Mode. There is not a decent method of posting an 8-byte reply.

9.2.2. Register Definitions

This section defines the registers used by the host to implement this Transport Layer Method. The interface registers used by the host for the Simple Method of Transport Layer are Inbound Post Queue and Outbound Post Queue. These two registers are defined below. Head and tail pointer registers are defined for further clarity.

9.2.2.1. PCI Configuration Base Register

32 bits wide, config address 0x10
The configuration register that defines the base address of the memory mapped Transport Layer registers. Note that the Transport Layer registers used are in fact I2O registers.

9.2.2.2. PCI Configuration Vendor Id

16 bits wide, config address 0x00
Value is 0x0E11.

9.2.2.3. PCI Configuration Device Id

16 bits wide, config address 0x02
Value defined in Section 6.

9.2.2.4. Interrupt Status Register

32 bits wide, address 0x30, read only
Host may read this register for interrupt status. Bit 3 set indicates that the Outbound Post List FIFO is not empty. Emptying the FIFO by reading automatically clears this interrupt.

9.2.2.5. Interrupt Mask Register

32 bits wide, address 0x34, read/write
Host manipulates to set/clear interrupt. A bit set masks the interrupt from driving the PCI interrupt line. Bit 3 is used for the Outbound Post List FIFO not empty interrupt.

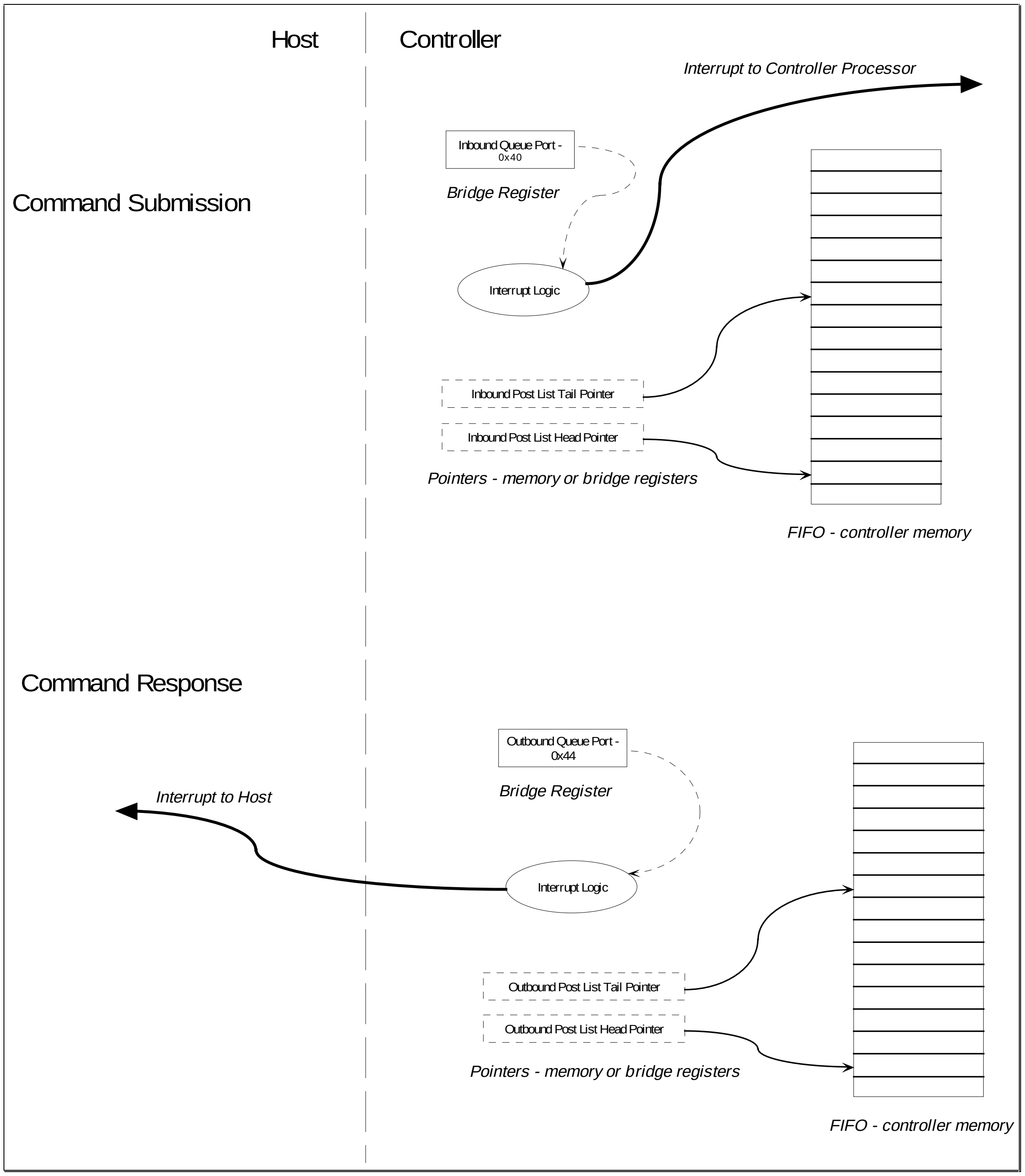
Host
Controller
Command Submission
Interrupt to Controller Processor
Inbound Queue Port - 0x40
Bridge Register
Interrupt Logic
Inbound Post List Tail Pointer
Inbound Post List Head Pointer
Pointers - memory or bridge registers
FIFO - controller memory
Command Response
Outbound Queue Port - 0x44
Bridge Register
Interrupt to Host
Interrupt Logic
Outbound Post List Tail Pointer
Outbound Post List Head Pointer
Pointers - memory or bridge registers
FIFO - controller memory

Figure 4 Simple Method - Registers

9.2.2.6. Inbound Post Queue

32 bits wide, address 0x40, treat as write-only

The host uses this register to access the Inbound Post List FIFO. When this register is written, the data written is placed at the tail of the Inbound Post List FIFO and the Inbound Post List Tail Pointer is incremented by four bytes.

9.2.2.7. Outbound Post Queue

32 bits wide, address 0x44, treat as read-only

The host uses this register to access the Outbound Post List FIFO. When this register is read, data is retrieved from the head of the Outbound Post List FIFO and the Outbound Post List Head Pointer is incremented by four bytes. A value of 0xFFFFFFFF is retrieved if the Outbound Post List FIFO is empty when this register is read.

9.2.2.8. Inbound Post List Head Pointer

16 bits wide, address 0x64, not accessible by host
Register indicates the current position of the head pointer within the Inbound Post List FIFO. The position indicated is a byte offset from the base address for the FIFO. Controller reads from the FIFO at the address indicated by the head pointer and must manually update the pointer.

9.2.2.9. Inbound Post List Tail Pointer

16 bits wide, address 0x66, not accessible by host
Register indicates the current position of the tail pointer within the Inbound Post List FIFO. The position indicated is a byte offset from the base address of the FIFO. Data written to the Inbound Post Queue is passed to this location and this register is incremented automatically.

9.2.2.10. Outbound Post List Head Pointer

16 bits wide, address 0x6C, not accessible by host
Register indicates the current position of the head pointer within the Outbound Post List FIFO. The position indicated is a byte offset from the base address for the FIFO. A host read of the Outbound Post Queue retrieves data referred to by this pointer at the head of the Outbound Post List FIFO. This register is incremented automatically with the host read.

9.2.2.11. Outbound Post List Tail Pointer

16 bits wide, address 0x6E, not accessible by host
Register indicates the current position of the tail pointer within the Outbound Post List FIFO. The position indicated is a byte offset from the base address for the FIFO. The controller writes data to the position in the FIFO indicated by this register and must increment the pointer manually.

9.2.3. Algorithm

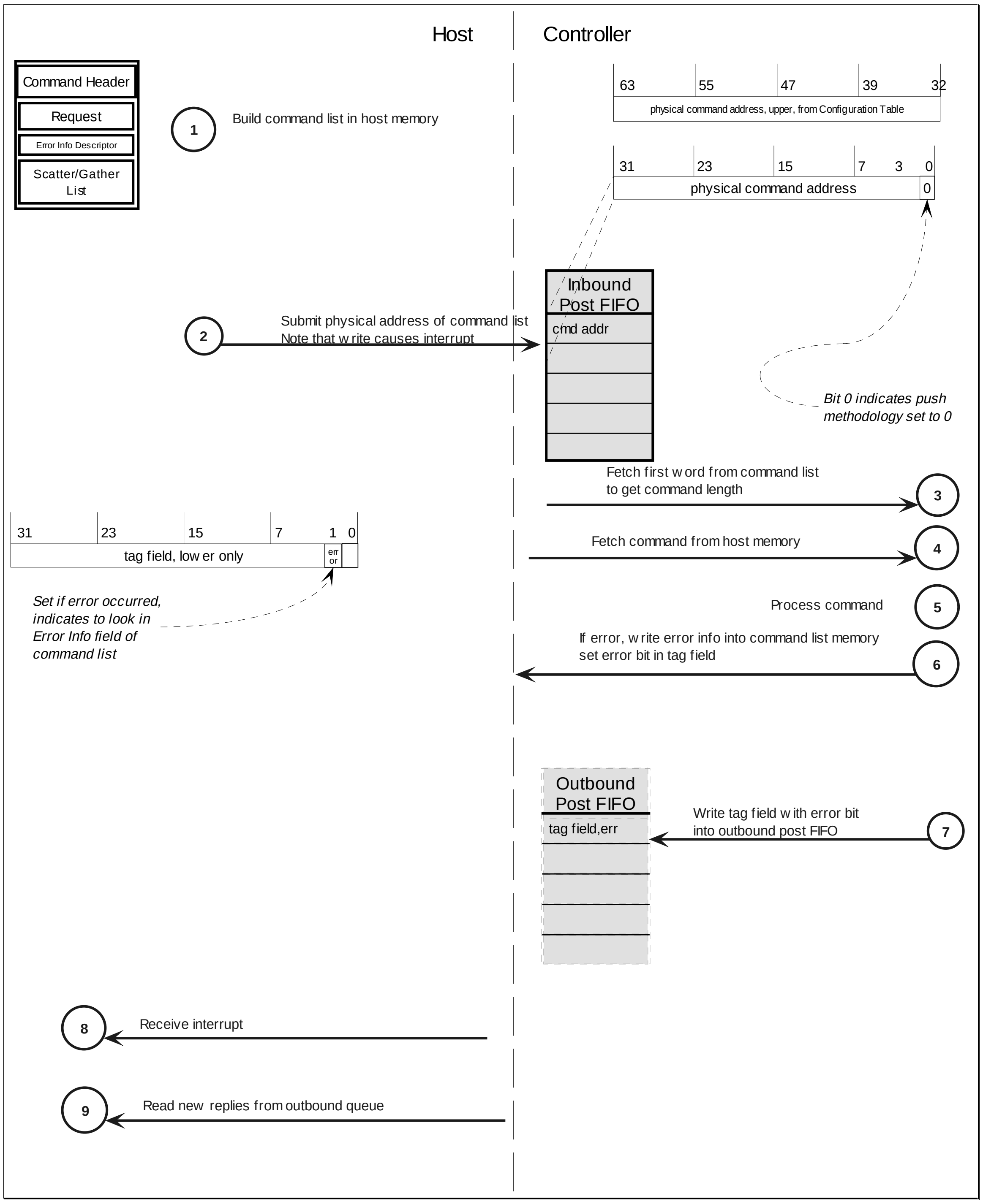
Host
Controller
Command Header
Request
Error Info Descriptor
Scatter/Gather List
1
Build command list in host memory
63 55 47 39 32
physical command address, upper, from Configuration Table
31 23 15 7 3 0
physical command address
0
2
Submit physical address of command list
Note that write causes interrupt
Inbound Post FIFO
cmd addr
Bit 0 indicates push methodology set to 0
Fetch first word from command list to get command length
3
31 23 15 7 1 0
tag field, lower only
error
Set if error occurred, indicates to look in Error Info field of command list
Fetch command from host memory
4
Process command
5
If error, write error info into command list memory set error bit in tag field
6
Outbound Post FIFO
tag field,err
Write tag field with error bit into outbound post FIFO
7
8
Receive interrupt
9
Read new replies from outbound queue

Figure 5 Simple Method - Algorithm

1. The host builds a 4-byte aligned command in a predetermined 4Gig region of memory, writing the tag field with a value to be used to reference the command upon completion. The upper 32-bit address of the 4Gig region must be initialized in the Configuration Table.

2. The host writes the physical address of the command to the Inbound Post Queue on the controller. The lower bit of the physical address is set to zero, not indicating a pull model.
3. The Inbound Post Queue becoming non-empty asserts an interrupt to the controller. The controller reads the oldest entry from the Inbound Post Queue, which is a command physical address. The controller fetches the first 4-byte value pointed to by this physical address. This 4-byte value contains the scatter/gather count from which the command length can be calculated.
4. The controller fetches the entire length from this physical address.
5. Controller processes the command.
6. If an error occurs during processing of a request, the controller builds an error info structure. Controller copies the error info structure for the command into the host location designated by the error info descriptor. Controller sets bit 1 of tag field to indicate the error.
7. Controller writes the modified tag field to the Outbound Post Queue on the controller.
8. The Outbound Post Queue becoming non-empty asserts an interrupt to the host.
9. Host gets interrupt from controller.
10. Host reads tag field from Outbound Post Queue, retrieving oldest entry. Host tests for error by checking tag field error bit, gets error info by de-referencing the error info descriptor. Process error if needed, otherwise processes completion. Host ultimately frees command memory. Host must continue to read Outbound Post Queue until a value of 0xFFFFFFFF is read, indicating the queue is empty.

9.2.4. Restrictions

- Command lists and queues must reside in a single 4Gig region in host memory.
- Command list physical addresses must be 4-byte aligned.

A. Appendix – Additional Error Info Codes

Command Status	**Info**	**Error Code**
Target Status	Target Status	0x0004_0000
	Complete With Error	0x04
Data Underrun	Frame Underrun	0x0002
	Complete With Underrun	0x22
Data Overrun	IOP Frame Overrun	0x0004
	Overrun	0x12
Invalid Command	Byte 0 – byte # of offense	
	Byte 1 – size of offending entry	
	Byte 4-7 – offending value	
Protocol Error	Class Connect Open	0x0080
	Programming Error	0x0100
	Retries Exceeded	0x0200
	Frame Rejected	0x0400
	Frame Time Out	0x0800
	Acknowledge Time Out	0x1000
	Abort Requested	0x2000
	Class 1 Error	0x8000
	Message Reject	0x0D
	Parity	0x0F
	Auto	0x10
	Bus Free	0x13
	Sequence	0x14
	Blocked	0x23
	Message	0x24
	Unsolicited Reset	0x2E
	SBMC	0x2F
	SMGR	0x31
Hardware Error	Hardware Error	0x07
	Fatal Error	0x06
	PCI Hard Error	0x26
	Hardware	0x30
Connection Lost	IOP Target Gone	0x20

	Link Down	0x4000
	Reset Logins	
	Warning Out of Sync	
	Warning Laser Fault	
	Loop Time Out	
	Loop Failure	
	Warning Offline	
	Logged Out	
	Loop Problem	
	Selection Time Out	0x0A

B. Appendix - Event Codes

This section describes event codes that may be expressed by the Notify on Event command for event notification.

Definitions:

• BYTE –	8 bits
• WORD -	2 bytes
• DWORD -	4 bytes
• QWORD -	8 bytes

Class	Sub-Class	Detail	Explanation	Severity*
0			Event Notifier Protocol.	
	0		Event Notifier Status Information	
		0	No Event. (This is returned in synchronous mode when no events have occurred.)	4
		1	Event Notifier Disabled. This event occurs when a Cancel Event Notify command is issued to disable the Event Notifier.	4
		2	Asynchronous event timeout reached; no event occurred in the specified number of seconds.	4
	1		Protocol Errors.	
		0	Event Queue Overflow. At least one event notification has been lost. The controller will only queue up to 100 events.	4
1			Hot-Plug.	

	0		Physical Drive Change. The Event Specific data Structure is as follows:
			
			struct PhysicalDriveChange {
			Word PhysicalDriveNumber;
			Byte ConfiguredDriveFlag;
			Byte SpareDriveFlag;
			Byte BigPhysicalDriveNumber
			Byte EnclosureBayNumber
			};
			
			On controllers with firmware that does not support more than 7 drives per bus, PhysicalDriveNumber 0-6 represents drives on the first SCSI bus and 7-13 represents drives on the second SCSI bus. On controllers with firmware that DOES support more than 7 drives per bus, bit 7 will be set if the drive's SCSI ID is greater than 7. If bit 7 is set, this value may be used as an argument to the ID Physical Drive command to determine which bus and drive ID this PhysicalDriveNumber actually corresponds to.
			
			The ConfiguredDriveFlag field will be 1 (configured) or 0 (not configured). A spare drive is considered configured whether or not it is active.
			
			The SpareDriveFlag field will be 1 (is a spare) or 0 (is not a spare). This field is only valid if the ConfiguredDriveFlag field is 1.
			
			On controllers with firmware that support more than 7 drives per bus, BigPhysicalDriveNumber will be 0x80 and higher. This value may be used as an argument to the ID Physical Drive command to determine which bus and drive ID this PhysicalDriveNumber actually corresponds to.	

		0	Physical Drive Removed.	4
		1	Physical Drive Inserted.	4

	1		Redundant Power Supply
			Event specific data returned:
			
			Struct power_supply_detail {
			        Word Port;
			        Word PSupID;
			        Word Box
			} ;
			
			where Port contains the port number of the external storage device and PsupID the ID of the affected power supply. Box indicates the drive enclosure number on the SCSI bus.	

		0	Power Supply Removed	
		1	Power Supply Inserted	

	2		External Fan.
			Event specific data returned:
			
			struct fan_data_detail {
			        Wrd Port;
			        Word FanID;
			        Word Box
			} ;
			
			where Port contains the port number of the external storage device reporting the condition, and FanID contains the ID of the affected fan module. Box indicates the drive enclosure number on the SCSI bus.	

		0	Fan Module Removed	
		1	Fan Module Inserted	
	3		AC Power (UPS Status) Event specific data returned: struct UPS_detail { Word Port; Word PSup_ID; } ; where Port contains the port number of the external storage device, and PSup_ID contains the ID of the power supply to which the affected UPS is connected.	
		0	UPS Removed	
		1	UPS Connected	
	4		Redundant Controller Event specific data returned: Struct red_ctrlr_detail { Word Slot; } ; where Slot contains the slot number of the affected controller	
		0	Controller Removed	
		1	Controller Inserted	
2			Hardware Errors and Failures.	
	0		Cables. The controllers that have > 1 SCSI connector and 1 SCSI controller. For > 1 attached cable w/ drives, event generated.	
	1		Memory. If there are too many runtime ECC errors in the byte lanes of the attached SIMM module, set an event and restart the controller w/o SIMM module.	
	2		External Fan. Event specific data returned: struct fan_data_detail {}; See Hot-Plug section of table for structure description.	

		0	Fan Fault. One of the fan modules in the external storage device is reporting a failure. The controller may attempt to shut down power to the storage device and/or spin down the installed disk drives.	
		1	Fan Degraded. One of the fan modules in the external storage device is reporting a degraded condition.	
		2	Fan OK Fans in the external storage device are now operational.	
	3		Voltage Regulator Module (VRM)	
		0	VRM Removed. One of the VRMs has been removed.	
		1	VRM Inserted. One of the VRMs has been inserted.	
		2	VRM Failed. VRM in the Cage box has been failed.	
		3	VRM OK. VRMs are now operational.	
3			Environment	

	0		Temperature.
			Event specific data returned:
			
			struct temperature_data_detail {
			        Word Port;
			        Word SensorID;
			        Word Box
			} ;
			
			where Port contains the port number of the external storage device and SensorID contains the ID of the temperature sensor reporting the condition, and Box indicates the drive enclosure number on the SCSI bus.	

		0	Temperature Limit Exceeded At least one of the temperature sensors in the external storage device has detected that the internal temperature has exceeded the preset limit. The controller may attempt to shut down power to the storage device and/or spin down the installed disk drives.	
		1	Temperature Warning Level Reached At least one of the temperature sensors in the external storage device has reported that the internal temperature is nearing the preset temperature limit. The controller may enact precautionary measures to prevent data loss should the temperature reach the operating limit.	
		2	Temperature Warning Condition Removed A previously existing temperature warning condition has been corrected. All of the temperature sensors in the external storage device are now reporting acceptable temperature levels.	
	1		Redundant Power Supply Event specific data returned: struct power_supply_detail {}; See Hot-Plug section of table for structure description.	
		0	Power Supply Fault	
		2	Power Supply OK. A redundant power-supply is now operational, or has been removed from the chassis.	

	2		Chassis Integrity
			Event specific data returned:
			
			struct chassis_detail {
			        Word Port;
			        Word Reserved;
			        Word Box;
			} ;
			
			where Port contains the port number of the external storage device reporting the chassis information. Box indicates the drive enclosure number on the SCSI bus.	

		0	Chassis Door Open. The controller may attempt to shut down power to the storage device and/or spin down the installed disk drives.	
		2	Chassis Door Closed	
	3		AC Power (UPS Status) Event specific data returned: struct UPS_detail {}; See Hot-Plug section of table for structure description.	
		0	AC Failure	
		1	Battery Low	
4			Physical Drive. The number of the physical drive is the first WORD in the Event Specific data field. On controllers with firmware that does not support more than 7 drives per bus, PhysicalDriveNumber 0-6 represents drives on the first SCSI bus and 7-13 represents drives on the second SCSI bus. On controllers with firmware that DOES support more than 7 drives per bus, bit 7 may be set. If bit 7 is set, this value may be used as an argument to the ID Physical Drive command to determine which bus and drive ID this PhysicalDriveNumber actually corresponds to.	
	0		Physical Drive State	

		0	Physical Drive Failure.
			
			The Event Specific data structure is as follows:
			
			struct PhyStatChange {
			        Word PhysicalDriveNumber;
			        Byte FailureReason;
			        Byte ConfiguredDriveFlag;
			        Byte SpareDriveFlag;
			        Byte BigPhysicalDriveNumber;
			        Byte EnclosureBayNumber
			};
			
			All drives that fail pertain to this event. This includes configured and non-configured drives. For example, a drive that is hot-plug inserted for capacity expansion that fails during its start-up will trigger this event.
			
			See Identify Physical Drive command for definitions of failure reason codes.
			
			On controllers with firmware that support more than 7 drives per bus, BigPhysicalDriveNumber will be 0x80 and higher. This value may be used as an argument to the ID Physical Drive command to determine which bus and drive ID this PhysicalDriveNumber actually corresponds to.
				2

5			Logical Drive. The number of the logical drive is the first WORD in the Event Specific data field.	
	0		Logical Drive Status.	

		0	The logical drive status has changed. The Event Specific data structure is as follows:
			
			struct LogStatChange {
			    Word LogicalDriveNumber;
			    Byte PreviousLogicalDriveState;
			    Byte NewLogicalDriveState;
			    Byte CurrentSpareStatus;
			};
			
			The PreviousLogicalDriveState, NewLogicalDriveState and CurrentSpareStatus fields are representative of the logical drive status and spare status fields in the "Identify Logical Drive Status" command. The PreviousLogicalDriveState is the logical drive state before the event and the NewLogicalDriveState is the state after the event.
				1, 2, 3 or 4.

		1	Exchanged Media. The logical drive is in a Failed state but has had one or more drive replacements and is ready to go to OK. However this will not happen until an Accept Media Exchange command is issued to the logical drive. (For example, a drive failed in a no Fault Tolerance configuration and the drive was just replaced via hot-plug while the server and storage sub-system were up and running.)	4

		2	Rebuild Aborted Due to a Read Error. The rebuild of a replacement drive was aborted because data could not be regenerated due to a drive read error. The nature of the error is not specified other than it is unrecoverable. The Event Specific data structure is as follows:
			
			struct RebuildAborted {
			    Word LogicalDriveNumber;
			    Byte ReplacementDrive;
			    Byte ErrorDrive;
			    Byte BigReplacementDrive;
			    Byte BigErrorDrive;
			};
			
			The ReplacementDrive field indicates the number of the physical drive that was rebuilding before the procedure aborted. The ErrorDrive field indicates the number of the physical drive that returned an error that resulted in the abortion.
			
			On controllers with firmware that does not support more than 7 drives per bus, drive numbers 0-6 represents drives on the first SCSI bus and 7-13 represents drives on the second SCSI bus. On controllers with firmware that DOES support more than 7 drives per bus, bit 7 may be set. If bit 7 is set, this value may be used as an argument to the ID Physical Drive command to determine which bus and drive ID this drive number actually corresponds to. Bit 7 will always be set in the new BigReplacementDrive and BigErrorDrive fields which always use the new drive numbering scheme.
				2

		3	Rebuild Aborted Due to a Write Error. The rebuild of a replacement drive was aborted because a write command to the replacement drive did not successfully complete. The nature of the error is not specified other than it was unsuccessful. The Event Specific data structure is the same the previous event (Rebuild Aborted Due to a Read Error).	2
	1		Logical Drive Error	

		0	Logical I/O Request Fatal Error A fatal error has been returned to the host on a read/write request on this volume. The Event Specific data structure is as follows:
			
			struct LogFatalErrorIO {
			    Word LogicalDriveNumber;
			    Dword LogicalBlockAddress;
			    Word LogicalBlockCount;
			    Byte LogicalCommand;
			    Byte FatalDriveBus;
			    Byte FatalDriveID;
			    Qword BigLogicalBlockAddress;
			};
			
			The LogicalBlockAddress, Count, and Command are taken from the failed logical I/O request. The FatalDriveBus and FatalDriveID fields indicate the *last* physical drive associated with this logical request to report a fatal error condition. If the block number is above 2TB, the LogicalBlockAddress field will contain 0xFFFFFFFF and the BigLogicalBlockAddress field will be set to the actual LBA.
				1,2

	2		Surface Analysis (Auto Reliability Monitoring)	

		0	Consistency Initialization Pass Completed
			The Event Specific data structure is as follows:
			
			struct SurfaceData {
			    Word LogicalDriveNumber;
			};	

6			Redundant Controller Operation	
	0		Redundant Operation Status.	

		0	The redundant operation status has changed. The Event Specific data structure is as follows:
			
			Struct RedStatChange
			{
			    Byte PreferredSlotControl;
			    Byte CurrentRedundancyMode;
			    Byte RedundantControllerStatus;
			    Byte RedundantFailureReason;
			    Byte PreviousSlotControl;
			    Byte PreviousRedundancyMode;
			    Byte PreviousRedundantCtrlrStatus;
			    Byte PreviousRedundantFailureReason;
			};
			
			The PreferredSlotControl, CurrentRedundancyMode, RedundantControllerStatus and RedundantFailureReason fields contain the same data returned in those fields by the Sense Redundant Controller Parameters command (0x82). Refer to the command section for descriptions of these fields.
			
			The respective ‘Previous…’ fields contain the values of those fields prior to the redundant operation status change.	

08			CISS specific event notifiers	
	00		Redundant status of the device is changing. If the most significant bit is set, this indicates that a “preferred” path has transitioned to this state.	
		00/80	Access path to device has become active.	
		01/81	Access path to device has become standby.	
	01		Path status to device is changing.	
		00	Path has returned to OK state.	
		01	Path has failed.	
		02	Path has degraded.	

	02		Hardware Error is detected. The event specific data structure is defined as follows:
			Struct loop_hw_info{
			Word bus;
			};
			The bus field indicates the bus number represents this FC loop at upper level software's view.	

		00	Parity Error. This indicates parity error detected by FC protocol chip. No action needs to be taken by upper level software in this case.	
	03		Logical unit detection	
		00	New logical device has been detected.	
		01	Existing logical device has gone away. This implies the firmware has some knowledge a previously available logical unit is no longer available, like being removed by a configuration command or meta drive failover caused it to disappear.	
9	All	All	Reserved.	

*Note: The severity levels are defined as follows: 1=Failure or Critical Error, 2=Warning or Degraded, 3= Repaired or Recovered, 4=Informational Note.