PCI SSA RAID Adapters

Technical Reference

SA33-3225-02

PCI SSA RAID Adapters

Technical Reference

SA33-3225-02

Note!: Before using this information and the product it supports, be sure to read the general information under “Appendix B. Notices” on page 245.

Third Edition (June 1998)

Contents

Figures

Conventions

Bits

The PCI SSA 4-Port RAID Adapter and PCI SSA Multi-Initiator/RAID EL Adapter use the standard convention for numbering the bits within bytes and words. Bit 0 is the least-significant bit; the number of the most-significant bit is 1 less than the width of the data.

Bit values are represented like this: 010b

Hexadecimal values are represented like this: 7Ah

Bytes

Except as noted below, the adapters' host interface uses the Little-endian convention, that is, it assumes that the least-significant byte of a number or an address is stored at the lowest byte address.

The Power processors in RS/6000 use the Big-endian convention. Therefore, AIX device drivers must take specific action to reverse the byte order that is naturally generated by the processor. PowerPC processors can operate in either Big-endian or Little-endian mode.

When using an SSA adapter in SCSI pass-through mode, it is important to note that parallel SCSI, and hence SSA-SCSI, sends the most-significant byte of a number first. This means that numbers in command-descriptor blocks, sense data and mode parameters appear in Big-endian format in memory.

Words

In a PCI SSA 4-Port RAID Adapter, a **word** is 4 bytes.

Registers

All register bits are read/write unless explicitly noted in the description of a bit.

Serial links

When an SSA adapter transfers information over a serial link, the bytes are normally sent and received in strict order of ascending storage addresses. This guarantees that customer data can be retrieved correctly when an SSA disk drive is interchanged between different host systems. (However an attached device may explicitly request out-of-order data transfers, for example, for a split read/write.)

Chapter 1. Description

Introduction to the Adapter

The IBM PCI SSA 4-Port RAID Adapter and PCI SSA Multi-Initiator/RAID EL Adapter are PCI bus-master adapters that serve as the interface between systems using the Peripheral Component Interconnect (PCI) architecture and devices using the Serial Storage Architecture (SSA). The adapters provide high-performance implementation of RAID-5 arrays. Each array can have from 2 through 15 member disk drives plus one for parity. Up to 32 arrays can be controlled by one PCI SSA 4-Port RAID Adapter or PCI SSA Multi-Initiator/RAID EL Adapter.

The adapters include:

- A DRAM data buffer with a storage capacity of 8MB on a PCI SSA 4-Port RAID Adapter, and 32MB on a PCI SSA Multi-Initiator/RAID EL Adapter.
- Hardware XOR capability
- A nonvolatile parity store of 8 KB with checking implemented by firmware. This is required to implement RAID-5, and is used to record details of parity updates in progress on the disks.

The adapters each provide 4 SSA ports for the attachment of storage devices such as hard disk drives. Each port operates at 20 MB/s full-duplex using point-to-point copper cables up to 25 meters long. As an alternative to copper cables, fiber optic cables can be used to link SSA nodes. Nodes linked by fiber optic cables can be up to 2.4 km (7874 ft) apart. Fibre-Optic Extenders, which are features of SSA units, connect the fiber optic cables to the SSA nodes.

SSA retains the SCSI-2 commands, queuing model, status, and sense bytes; it is an industry-standard interface.

Each of the 2 pairs of SSA ports can attach up to 48 dual-port devices in a closed loop. If the loop is broken by a fault, the two ports continue to access the devices using the remaining connections as a string; however, it is not intended that the devices should be configured as a string initially. These SSA features support fault-tolerant applications. One pair of SSA ports can be accessed from inside or outside of the system unit for use by internal or external devices. The other pair can only be accessed by external devices.

Only a single PCI SSA 4-Port RAID Adapter is permitted in an SSA loop. The number of PCI SSA Multi-Initiator/RAID EL Adapters permitted in an SSA loop depends on how the disks are configured and the level of the adapter microcode. Adapter microcode at, or later than, level 50 supports:

- Only one adapter in a loop if the fast-write facility is configured active for any disk drive or RAID-5 array.
- Up to 2 adapters in a loop if any disk drive is configured as a member of a RAID-5 array and no array is configured for the fast-write facility.
- Up to 8 adapters in a loop if no disks are configured for the fast-write facility or as a member of a RAID-5 array.

Adapter microcode before level 50 supports:

- Only one adapter in a loop if the fast-write facility is configured active for any disk drive or if any disk drive is configured as a member of a RAID-5 array.
- Up to 2 adapters in a loop if the fast-write facility is not configured active for any disk drive and no disk drive is configured as a member of a RAID-5 array.

Note: The adapter microcode level is contained in the Vital Product Data (VPD) record as the first byte in the ROS Level and ID field.

The PCI SSA Multi-Initiator/RAID EL Adapter has an optional SSA Fast-Write Cache Option Card. When this feature is installed, data that is to be written to a disk drive is first written to a nonvolatile cache. This cache has a capacity of 4MB (plus ECC) and is on a removable daughter card. A battery on the daughter card enables it to retain data for up to 10 years. The daughter card can be moved to another PCI SSA Multi-Initiator/RAID EL Adapter if the original adapter fails.

The PCI SSA Multi-Initiator/RAID EL Adapter supports SSA target-mode operations. In these, 512 bytes of data are transmitted from one host to another. The device driver supports transfers of larger amounts of data, but only by breaking it into 512–byte blocks. Target-mode operations are optimized for small data transfers; they are intended to allow systems to enquire about each other's health or for synchronizing purposes, rather than for large data transfers or high performance.

The adapters each have a PCI bus interface able to operate with PCI clocks up to 33 MHz (132 MB/s). The adapters operate as PCI bus masters to transfer data. I/O transactions are transferred between the host system and the adapter using control blocks in host memory.

The device drivers for the adapters and their subsystems handle SCSI commands, status, and sense. The adapters also deal with the SSA protocols and it recovers link errors and some disk errors internally.

The device drivers and adapter communicate with each other by means of a logical client-server network called the Independent Packet Network (IPN). IPN provides a consistent application programming interface (API) independent of the environment. This produces a software environment that allows new functions to be added easily as additional servers or filters.

The array functions are implemented as IPN filters in the adapter card. This means that the interface to the device driver for I/O operations to an array is the same as the interface to a normal SSA disk drive. The device driver for the PCI SSA Adapters can control disk drives individually or when configured in arrays. The system cannot boot from a disk or from an array.

Additional IPN transactions are provided to configure arrays. Configuration manager software using these transactions is required to provide a user interface to array configuration.

The performance of the adapter takes full advantage of the SSA links. A single adapter can support at least 200 overlapped physical I/O requests at a time; these might result from, for example, 105 RAID-5 I/O requests. Operations to array filters are processed by the adapter and result in operations being sent to disk drives attached to the filter. Subject to the capabilities of the system bus, the maximum bandwidth for data transfers can be 40 MB/s for PCI fetches and 65 MB/s for PCI store operations when transferring data to a disk drive that is not in an array. The bandwidth for data transfers is approximately 25 MB/s when transferring data from an array. Chapter 5. Array and Fast Write Filters includes performance details for the different RAID configurations.

Using an SSA Fast-Write Cache Option Card enhances performance in two ways:

- By reducing the adapter service time to less than 1 millisecond for write operations that use the cache (that is, for those operations that are not too long and for which there is space in the cache).
- By joining split sequential write operations into larger data transfers when destaging to the disk drives.

Both of these are particularly helpful for RAID-5 arrays because write operations to RAID-5 arrays involve first reading the old data and then merging the new data with it before writing to the disk drives (“RAID-5” on page 77 describes this in more detail). Even more advantage can be gained if there is sufficient new data to write a complete stripe of data to a disk drive (as described on 78); this removes the need to read the old data first.

Using an SSA Fast-Write Cache Option Card might reduce performance because:

- Additional firmware overhead for moving data from the disk drive to the cache and back reduces the number of operations per second possible.
- Longer internal bus cycles are needed when moving data from the disk drive to the cache; this reduces the available internal bandwidth.

Configurations

An example of an SSA subsystem that can be attached to a PCI SSA 4-Port RAID Adapter is the **7133 SSA Subsystem**. Each 7133 unit contains up to 16 SSA disk drives with fault-tolerant power and cooling. It is available either as a 19-inch rack-mounted unit (4 EIA units) or as a deskside unit.

Adapter Functions

The principal functions of the PCI SSA 4-Port RAID Adapter and PCI SSA Multi-Initiator/RAID EL Adapter are:

- The adapter performs a power-on self-test (POST) to verify correct operation of the hardware.
- The adapter configures the SSA network. It can act as the master node if required.
- When interrupted by the host processor, the adapter fetches IPN transactions by Direct Memory Access (DMA) from host memory.
- The adapter translates each transaction into SCSI commands and issues them to the addressed device over a serial link. A pass-through mode is also provided to allow any SCSI command to be issued.
- When requested by a device, the adapter fetches write data from host memory by DMA and transmits it to the device. Similarly the adapter receives read data from the device and stores it in host memory by DMA.
- For disk drives that are not in an array, data is transferred between the host memory and the devices without using the data buffer. For RAID-5, data is transferred via the data buffer; subsequent reads might be satisfied from the data buffer without a further operation to the device. The adapter can scatter or gather the data to or from noncontiguous regions of host memory.
- The adapter receives SCSI status from the device. If there is an error the adapter issues a SCSI Request Sense command to the device and may then attempt to recover the error. In all cases the adapter interrupts the host processor to present the result of the transaction and to log errors if appropriate.

Disk Arrays

The adapter provides RAID functions by means of filters between the device driver and the disk drives. The filters are implemented by microcode that runs in the adapter. They present the image of a single logical disk drive to the device driver and use one or more components to implement this image. Up to 32 arrays can be configured on a single adapter.

Chapter 5. Array and Fast Write Filters defines these functions in more detail.

Supported Standards

The PCI SSA 4-Port RAID Adapter implements the standards described in the following documents:

- *PCI Local Bus Specification*, production version, revision 2.0
- *Serial Storage Architecture, 1995 Physical (SSA-IA/95PH)*, October 1995
- *Serial Storage Architecture, 1995 SCSI-2 Protocol 9SSA-IA/95SP)*, October 1995
- *Small Computer System Interface - 2 (SCSI-2)*, X3.131.199X, Revision 10m.

Introduction to the Independent Packet Network (IPN)

The device drivers and adapter communicate with each other by means of a logical client-server network called an Independent Packet Network (IPN).

IPN is a logical network of **services**. A client can access a service by specifying its address in the IPN network, without being concerned where the service is physically located. In IPN terminology, the client is a **master** and the Service is a **slave**.

The unit of work in IPN is a **transaction**. The routing layer of IPN establishes a connection between the master and slave for the duration of each transaction. A master may queue multiple transactions in the same slave. However, the slave can execute the transactions in any order it chooses and even execute several transactions concurrently.

An IPN **node** is a hardware unit that runs the IPN kernel, a host system or the adapter are examples of nodes. In addition to network routing, the IPN kernel also performs such tasks as scheduling, memory management, and timer functions.

The adapter provides a **disk service** to give basic read/write access to each attached disk drive. Additional services can be added, such as a RAID service.

The host device driver is an IPN master and also provides an **error logger**, which is a service for logging subsystem errors.

Every IPN node also contains a **registry** service. The registry keeps a list of all services running on its node and all other nodes that are directly accessible through a gateway on that node. The registry also forwards errors detected by the services running on its node to the error logger.

IPN spans the device driver and the adapter. IPN uses a **gateway** to cross a physical interface such as the PCI interface. The gateway is transparent to the master and slave and it incorporates the specific features of the physical interface.

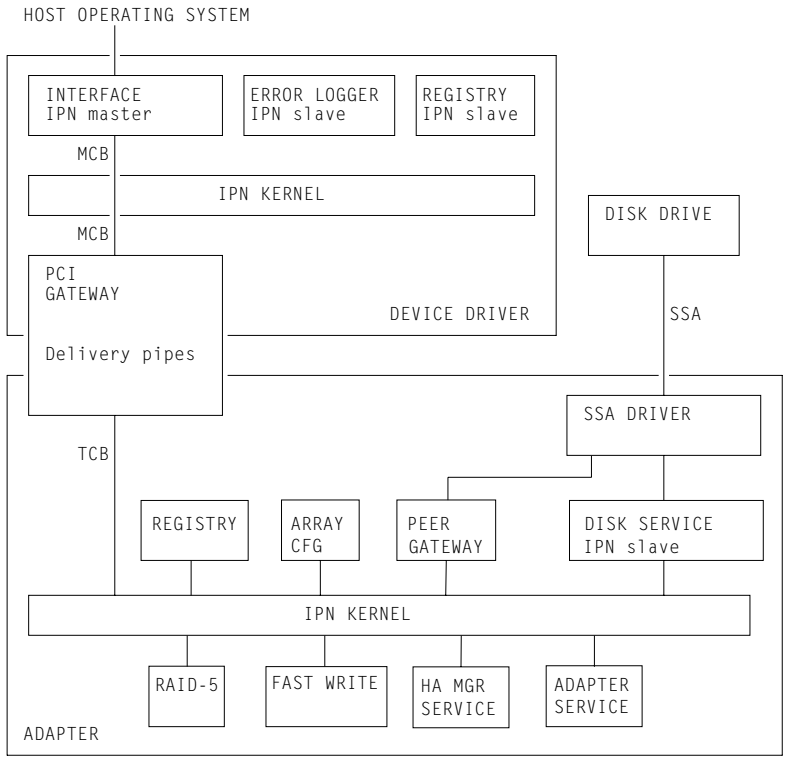


Figure 1. IPN Components

The PCI SSA 4-Port RAID Adapter contains a PCI gateway, a disk service, a registry service, an SSA driver, the IPN kernel, and a service for each RAID function, as shown in Figure 1. A typical transaction to read data from a RAID-5 array would be processed as follows:

1. The device driver contains a master process that generates IPN transactions. The master calls the host IPN kernel with a pointer to a **master control block (MCB)** for the transaction. The MCB is addressed to the disk service.
2. The host IPN kernel calls the PCI gateway with a pointer to the MCB.
3. The host side of the PCI gateway creates a **gateway transaction control block (GTCB)** in host memory. This is a form of the TCB that is optimized for the gateway function. Each GTCB has a pointer to the next GTCB to form a chain to GTCBs.

   The PCI gateway interrupts the adapter when a new GTCB has been set up.
4. The adapter side of the PCI gateway fetches the GTCB by DMA. The gateway then creates a **transaction control block (TCB)** in the adapter address space. A TCB is a subset of an MCB. Finally the gateway calls the adapter IPN kernel to submit the TCB.
5. IPN calls the RAID-5 service for the addressed resource with a pointer to the TCB.

6. The RAID-5 service generates IPN transactions for each of the disk drives and sends these transactions to the disk service using the IPN kernel.
7. The disk service generates the appropriate SSA-95SP read commands and passes them to the SSA driver.
8. The SSA driver issues the SCSI commands to the disk drives using the SSA protocol.
9. When the disk drives offer the requested data, the SSA driver transfers the data to DRAM. The data is transferred to host memory through the PCI gateway.
10. When the drive returns good-completion status, the disk service calls IPN with the result of each transaction generated by the RAID-5 service.
11. When all the transactions have completed, the RAID-5 service informs the host that the original read transaction has completed. It does this by using the PCI gateway to put a pointer to the GTCB into the adapter outgoing delivery pipe in host memory.

Chapter 2. System-to-Adapter Interface

IPN Transactions

The device driver issues IPN transactions to the adapter to access the attached devices. Occasionally, the adapter issues a transaction to the device driver to log an error. Each transaction is created as an MCB, transferred over the PCI interface in a GTCB and finally delivered to the destination service as a TCB. The GTCB format is described here.

To avoid deadlock at the bridge between the PCI and the adapter local bus, the system does not access adapter memory during normal operation when a data transfer to host memory could be in progress.

Gateway Transaction Control Block (GTCB)

Each transaction passed over the PCI interface is described by a GTCB. The GTCB is located by a pointer in the previous GTCB for transactions to the adapter. It is located by a pointer in a delivery pipe for transactions from the adapter. The GTCB has a fixed length of 100 bytes and it is aligned on an 8-byte boundary. The GTCB is built by the master side of the gateway in its local memory. It remains allocated from the time the transaction request is issued to the gateway until the gateway returns a reply for the transaction.

Table 1. Format of a GTCB

Byte	3	2	1	0
0	N	Destination_node		
4	Destination_service			
8	Reserved = 00h	Major_function	Minor_function	
12 through 27	Parameter_DDR			
28 through 43	Transmit_DDR			
44 through 59	Receive_DDR			
60 through 75	Status_DDR			
76	Result_pointer			
80 through 95	Parameters			
96	Next_GTCB_Address			

Destination_node

This field contains a 31-bit unsigned integer to identify the destination adapter card. The device driver assigns each card a unique number based on the host system and physical bus slot occupied by that card.

NotYetValid (N)

Byte 3 bit 7 is the NotYetValid flag. This bit is set to 1b for any GTCB in the chain that has not yet been set up. It is in the first word that is fetched by the adapter when obtaining a new GTCB.

Destination_service

This field contains a 32-bit unsigned integer to identify the destination service.

The registry has a fixed service number of 0000 0001h. The service numbers for the disk service and filters are dynamically allocated and can be obtained from the registry. The error logger connects itself to the registry during initialization.

Major_function

This byte is coded as follows:

02h **Application**. These transactions are defined separately by each service. See “Disk Service” on page 117 for the transactions supported by the adapter disk service, “Registry Service” on page 89 for the transactions supported by the registry service, <!Á %comment; Á:hdref refid=adapser. for the transactions supported by the adapter

service,. ||||> and “Array-Configuration Service” on page 176 for the transactions supported by the configuration-agent service.

All other values are reserved.

Minor_function
These 2 bytes select a particular transaction.

Parameter_DDR
This 16-byte field contains the data descriptor for the transaction parameters. See “Data Descriptor (DDR)” for the format of a data descriptor. If the length of the parameter is less than, or equal to, 16 bytes, the parameter must be in bytes 80 through 95 of the GTCB and the type, address, and offset fields of the DDR are ignored.

Transmit_DDR
This 16-byte field contains the data descriptor for data to be transmitted from the master to the slave.

Receive_DDR
This 16-byte field contains the data descriptor for data to be received by the master from the slave.

Status_DDR
This 16-byte field contains the data descriptor for the transaction status. The status, if any, is defined by the particular transaction.

Result_pointer
This field points to the result word for the transaction. (See “Result Word” on page 13 .)

For transactions from the host this field contains a PCI memory address.

Parameters
This field can contain up to 16 bytes of parameters for the transaction. This space is sufficient for all performance-critical transactions.

If a transaction requires more than 16 bytes of parameters then the additional bytes are appended following the GTCB. Alternatively, all of the parameters can be stored separately from the GTCB. Both alternatives require the adapter to perform an additional DMA operation after fetching the GTCB.

The parameters are defined separately for each particular transaction.

Next_GTCB_Address
This field contains a pointer to the next GTCB for transactions from the host. The two low order bits of the address are not used as part of the address. The low order bit (byte 96 bit 0) is a Last_GTCB flag. This is set in the last GTCB of the chain of GTCBs that have been generated and is used by the adapter to stop the fetching of further GTCBs.

Data Descriptor (DDR)

A DDR is a component of the GTCB or Slave operation that provides the parameters, the receive data area, the transmit data, or the status area for a transaction.

Table 2. Format of a Data Descriptor

Byte	3	2	1	0
0	Type	SG_length	Reserved = 0000h	
4	Address			
8	Offset			
12	Data_length			

Type This field is coded to select one of the following types:

0Bh DT_Null. No data is present.

0Ch DT_PCI The address field points to the data.

0Dh DT_PCIScatGat. The address field points to a scatter/gather list whose entries point to the data.

Note: This function is provided by the device driver.

All other values are reserved.

SG_length
This 1-byte field is only used when the type field is 0Dh. It contains an unsigned integer that specifies the number of entries in the scatter/gather list.

Address
This field points to the data or a scatter/gather list. For transactions from the host the Address field contains a PCI memory address.

Offset This unsigned integer allows a *logical* offset to be added to locate the first byte of data. It is used mainly when the data is located by a scatter/gather list.

Data_length
This unsigned integer specifies the length of the buffer available for data.

Scatter/Gather List

The scatter/gather list is a variable-length list which allows data to be relocated in a virtual-memory environment. The list entries describe the data fragments in turn. Each entry specifies the physical address and length of a fragment.

Table 3. Format of a scatter/gather list

Byte	3	2	1	0
0	Address 1			
4	Length 1			
8	Address 2			
...	...			
8N - 4	Length N			

Address
This field contains the physical address of a fragment of data. The address may be on any byte boundary.

Length This field contains an unsigned integer that is the length of the fragment in bytes. The length may be any number of bytes.

Result Word

The result word is used to return the results of a transaction. It is aligned on a 4-byte boundary.

Table 4. Format of a Result word

Byte	3	2	1	0
0	Reserved = 00h	Network_result	Application_result	

Network_result
This field is reserved for reporting errors in IPN networks. The master process preformats this field with 00h, indicating no error. This avoids the need for the slave to update it when there is no error.

Application_result
This field contains errors reported by the destination service. The master process preformats this field with 0000h, indicating successful completion.

The specific errors reported are defined in “Application Results” on page 218.

Delivery Mechanism from the Host

Transactions from the host are passed to the adapter in GTCBs. The GTCBs are linked in a chain, the address of the next GTCB being defined in the Next_GTCB_Address field of each GTCB. The low order bit of this field is used for control:

Byte 96 bit 0
Last_GTCB

When the Last_GTCB bit is set to 1b, this GTCB is the last valid GTCB in the chain. Initially, the Last_GTCB bit is zero in all GTCBs except the first one of the chain. The NotYetValid bit (byte 3 bit 7) is set to 1b in all GTCBs that have not yet been set up.

The host informs the adapter that one or more GTCBs have been set up by setting the new-host-GTCB bit in the Adapter Interrupt Doorbell register. When interrupted, the adapter fetches the next GTCB that was pointed to in the last GTCB it had fetched using DMA accesses. After processing the GTCB, it may fetch another GTCB if the Last_GTCB bit was zero (and the adapter interrupt was zero). To minimize overshooting the end of the chain of GTCBs, the adapter and host must set flags, pointers and interrupt controls in the order shown below.

The host specifies the location of the first GTCB in the chain initially by the value of the Host_GTCB_start_pointer field in the Initialize command.

Host actions to set up a GTCB:

1. Set up the new GTCB in the next GTCB of the chain.
2. Set up the pointer in this new GTCB to point to the following GTCB of the chain. The NotYetValid bit should be on in this following GTCB.

3. Set the Last_GTCB bit to 1b in the new GTCB.
4. Clear the NotYetValid bit in the new GTCB.
5. Clear the Last_GTCB bit in the previous GTCB.
6. Set the new-host-GTCB bit in the Adapter Interrupt Doorbell register to interrupt the adapter.

Adapter actions when interrupted by the doorbell register:

1. Clear the new-host-GTCB bit in the Adapter Interrupt Doorbell register.
2. Fetch the new GTCB from PCI memory using the Next_GTCB_Address field as the pointer from the last GTCB fetched.
3. If the NotYetValid bit is 0b in the new GTCB, process the GTCB. If it is 1b, the GTCB has not yet been set up and the adapter should not process this GTCB.
4. If the Last_GTCB bit is 0b in the GTCB fetched, and the new-host-GTCB bit in the Adapter Interrupt Doorbell register is 0b, loop back to step 1 and fetch the next GTCB.

Delivery Mechanism from the Adapter

Transactions originating from the adapter are defined by a Transaction Control Block (TCB) held in adapter memory. This has the same format as the first 80 bytes of a GTCB offset by 16 bytes. The DDR fields do not have to be used by the host because it executes the transactions by sending host slave operations to the adapter (see “Host Slave Operations” on page 16). The length fields in each DDR indicate if any data needs to be transferred.

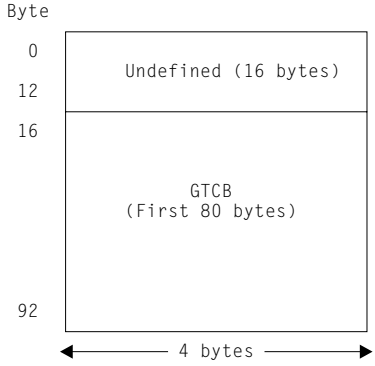


Figure 2. Format of a TCB

The PCI Gateway uses a delivery pipe for communication between the adapter and the host. The delivery pipe delivers **control elements**. The pipe is a circular queue in which each element is a 4-byte token that identifies a TCB in adapter memory, as shown in Figure 3 on page 15.

The adapter outgoing pipe is located in host memory and the adapter accesses it by DMA.

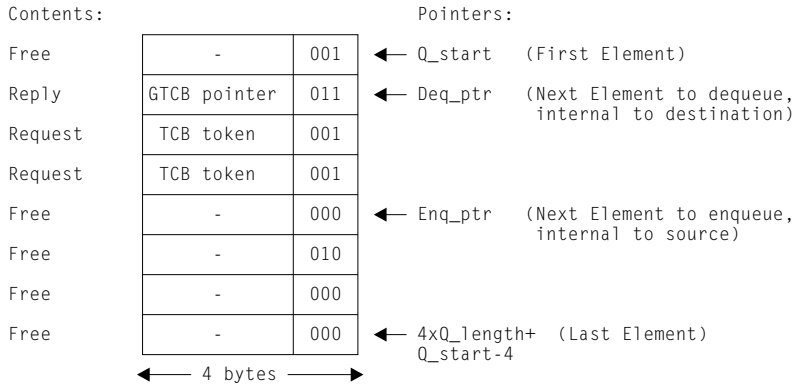


Figure 3. Delivery Pipe

Because a TCB must be aligned on a 8-byte boundary, the 3 low-order bits of each element can be used for identification, as follows:

000b Adapter Transaction Request (Phase 0)

001b Adapter Transaction Request (Phase 1)

010b Host Transaction Reply (Phase 0)

011b Host Transaction Reply (Phase 1)

The low-order bit is a phase flag which prevents the dequeue agent from fetching elements not yet stored by the enqueue agent.

The host issues the Initialize command to the adapter to specify the start address of the adapter outgoing pipe, *Adapter_Q_start*, the total number of words in the pipe, *Q_length*, and the maximum number of outstanding requests that the adapter is allowed to originate, *Adapter_max_requests*.

The adapter must always leave at least one free element in its outgoing pipe plus sufficient elements for replies to the maximum number of outstanding requests from the host. Therefore, the maximum number of outstanding request elements that the adapter is allowed to originate is:

Adapter_max_requests = Q_length - (1 + Host_max_requests)

Initially, the host fills its *incoming* pipe with dummy phase 1 request elements and sets its local variables as follows:

Adapter_enq_ptr = _Q_start
Set enqueue pointer to start of outgoing pipe

Host_deq_ptr = _Q_start'
Set dequeue pointer to start of incoming pipe

Adapter_enq_phase = 0
Phase of enqueue elements

Host_deq_phase = 0
Expected phase of dequeue elements

Av_requests = Host_max_requests
Number of available outstanding requests

To enqueue an element, the adapter:

1. If enqueuing a request, checks that the number of available outstanding requests is greater than zero.
2. If enqueuing a transaction request, builds the TCB in local address space.
3. If enqueuing a transaction reply with a non-zero result, stores the result word.
4. Stores the TCB pointer at the address in the enqueue pointer, setting the three low-order bits as previously defined.
5. Interrupts the host by setting the host-inbound-pipe-entry bit in the PCI Interrupt Doorbell register.
6. Advances the enqueue pointer to the next free element. If the enqueue pointer wraps around to the beginning of the pipe, the adapter toggles the enqueue-phase bit.
7. If enqueuing a request, decrements the number of available outstanding requests.

When the host is interrupted, it repeats the following procedure until the pipe is empty.

1. The host fetches the element addressed by the dequeue pointer. If the phase flag does not match the dequeue-phase bit, the pipe is empty.
2. Otherwise, if the element is a reply, the host increments the number of available outstanding requests.
3. The host advances the dequeue pointer to the next element. If the dequeue pointer wraps around to the beginning of the pipe, the host toggles the dequeue-phase bit.
4. The host processes the element dequeued.

This protocol ensures that:

- Elements will not be over-written by the enqueue agent before they have been dequeued.
- The adapter will always have space in its outbound pipe to reply to all outstanding transactions from the host. Therefore, the pipes cannot dead-lock.
- There are no dynamic shared variables to control the pipes.

Host Slave Operations

IPN transactions originating from the adapter are implemented using host slave operations from the host. These operations request the adapter to transfer the transaction control block (TCB) or data referenced by the TCB.

The adapter informs the host that a TCB has been created by updating the host incoming delivery pipe in host PCI memory and by setting an interrupt bit in the PCI Interrupt Doorbell register. To process the transaction, the host sets the Parameter register to point to a host-slave-operation control block in host memory, and then sets the host-slave-op bit in the Adapter Doorbell Interrupt register. When interrupted, the

adapter fetches the host-slave-operation control block, executes the operation specified, and informs the host when it is completed by setting one of the host-slave-operation-completion interrupt bits in the PCI Interrupt Doorbell register. The format of the host-slave-operation control block is:

Table 5. Format of a Host-Slave-Operation Control Block

Byte	3	2	1	0
0	Operation Type			
4	TCB Token			
8	Offset			
12 through 24	Data Descriptor			
28	Length			
32	Result			

Operation type

This field defines the operation to be performed for the adapter transaction identified in the TCB address field.

The following codes are supported:

OT_XferTCB
Transfer the transaction control block to host memory

OT_Parms
Transfer the parameter data to host memory

OT_Fetch
Transfer the transmit data to host memory

OT_Store
Transfer the receive data to adapter memory

OT_Status
Transfer the status data to adapter memory

OT_Done
Transfer the result data to adapter memory in the result field and, optionally if the data-descriptor field is not null, also transfer status data.

TCB Token

This field is a pointer to the TCB in adapter memory; this identifies the adapter TCB for this operation.

Offset This field is an unsigned integer that is an offset to be added to the start of the buffer address in adapter memory that is defined by the TCB's DDR selected by the operation type. This allows the transfer of data to be started at any point in the adapter data buffer.

Data Descriptor
This is a 16-byte field that defines host-memory addresses and data lengths for the data to be transferred by the operation. The format is defined in “Data Descriptor (DDR)” on page 11.

Length This field is a pointer to a 4-byte location in host memory in which the adapter stores the length of data transferred during the operation.

Result This field contains the result data of the transaction. This data is only returned for an OT_Done operation.

Timeouts

The adapter times a GTCB from arrival to reply. If the reply is not sent within two minutes and the transaction is not FN_REGY_TestResrcReady, the adapter sets the error register with the error code SS_TIMEOUT, interrupts the host, and waits to be reset. While waiting to be reset, the adapter does not respond to heartbeats or accept any transactions from the host.

Commands

The adapter command set provides a low-level interface to the adapter, for initialization for example. Only one command can be in progress at a time.

The command interface uses the following protocol:

1. If the host has previously sent a command to the adapter, it should not send another command until it has received an interrupt from the adapter to indicate completion of the current command.
2. If necessary, the host should write the Parameter register.
3. The host should write the command code to the Command register.
4. The host sets the command-register-loaded bit in the Adapter Interrupt Doorbell register to cause an interrupt to the adapter.
5. The adapter reads the Command register and (if necessary) the Parameter register.
6. When the adapter completes a command, it writes to the PCI Interrupt Doorbell register to interrupt the host. The host reads the PCI Interrupt Doorbell register to determine the reason for the interrupt and then clears that bit in the register.

The adapter commands Initialize, Download, and Execute I/O use the Parameter register for the address in host memory of a parameter block. A parameter block must be on a 4-byte boundary.

After a Download command has been issued, no further commands can be issued unless the adapter is reset. A Download command must be immediately preceded by a reset.

Initialize

The Initialize command configures the delivery pipes used by IPN transactions and allocates an IPN node number to the adapter. “Shared Runtime Registers” on page 51 describes the registers used by the adapter commands.

Command register
30h

Parameter register
Physical address of the parameter block.

Exceptions
The following exceptions can be indicated in the PCI Interrupt register:

- Adapter has detected a catastrophic error.

  Further details of the error are provided in the error code in the Error register, for example, Invalid Parameter.

Table 6. Parameter block for Initialize

Byte	3	2	1	0		
0	Adapter_Q_start				R	W
4	Q_length		Host_max_requests			
8	Host_GTCB_start_pointer					
12	Node					

Adapter_Q_start
This field contains the physical address of the first word in the adapter outgoing delivery pipe in PCI memory space. This must be aligned on a 4-byte boundary. The pipe must be allocated in contiguous host memory without using scatter/gather.

R When the ’R’ bit = 1b, the following functions are executed:

- All reads of data from non-RAID disks are staged through DRAM. This has the effect of fetching up to 4Kbytes consecutively from the same disk rather than 128 byte fetches from different disks.
- PCI data bursts for fetches and stores are maximised for 64 bytes in length and 64 byte alignment.
- Data fetches from adapter DRAM use 1Kbyte scatter gather lists to limit non-64 byte alignment to a 1K segment.
- The delivery pipe in host memory is updated at the end of each transaction using DMA rather than direct master transfers.
- When the ’W’ bit (byte 0 bit 0) is also 1b, the adapter is PCI 2.1 compliant. However, throughput in operations per second and adapter bandwidth is reduced when in this mode.

W When the ’W’ bit = 1b, all writes of data to non-RAID disks are staged through DRAM.

Q_Length
This field contains a 2-byte unsigned integer specifying the number of elements allocated to each pipe.

The only valid setting is 256 elements.

Host_max_requests
This field contains an unsigned integer specifying the maximum number of outstanding requests that the host is allowed to originate. *Host_max_requests < Q_length - 1.*

The host-max-requests field must be set to 200, otherwise the adapter reports an exception for Invalid Parameter.

Host_GTCB_start_pointer
This field contains the physical address of the first word of the first host GTCB in the chain of GTCBs. This must be aligned on a 8-byte boundary.

Node This field contains the IPN node number assigned to the adapter by the device driver.

Download

The Download command allows updated microcode to be down-loaded into the adapter. The microcode load includes the BIOS code. A single BIOS image is held in the adapter. The Download command must be immediately preceded by a reset.

Command register
31h

Parameter register
Physical address of the parameter block.

Exceptions
The following exceptions may be indicated in the PCI Interrupt register:

- Adapter has detected a catastrophic error.

  Further details of the error are provided in the error code in the error register.

Table 7. Parameter block for Download

Byte	3		2	1	0
0	G	Reserved = 0000000b	Reserved = 00h	SG_Length	
4	Address				
8	Length				
12	LRC				
16 through 23	ROS level				

Gather (G)
If byte 3 bit 7 is set to 1b then the Address parameter points to a scatter/gather list. Otherwise Address points to the microcode itself.

SG_length
This 2-byte field is only used when the gather bit is set to 1b. It contains an unsigned integer which specifies the number of entries in the scatter/gather list.

Address
This field contains the PCI address of the microcode or a scatter/gather list which locates the microcode. Addresses are aligned on a 4-byte boundary.

Length This field contains a 32-bit unsigned integer which specifies the length of the microcode in bytes. Length is a multiple of 4.

LRC This word contains a Longitudinal Redundancy Check (LRC) to ensure integrity of the microcode. The LRC is formed by adding each word of the microcode to the constant AAAA AAAAh, using 32-bit arithmetic.

ROS level
This 8-digit ASCII-coded field contains the level of the flash EPROM after the download. This is the value that is reported in the 8 most significant bytes of the RL field of the VPD; the least significant 4 bytes of that field are zero.

The updated microcode is downloaded as follows:

1. The host sets the PCI-adapter-software-reset bit in the shared runtime register at PCI offset 6Ch.
2. The adapter disables the SSA ports and boots from the protected sectors of the flash EPROM only.
3. The host issues the Download command.
4. The adapter fetches the microcode to RAM.
5. If the LRC is good, the adapter writes the new microcode to flash EPROM.
6. The adapter generates a host interrupt by setting the adapter-executed-command-successfully bit in the PCI Interrupt Doorbell register to inform the host that the command has completed.
7. The host must then issue an adapter reset followed by a command other than Download before the adapter boots from the flash EPROM, including the new microcode, and enables the SSA ports.

Execute I/O

The Execute I/O command provides a simple synchronous I/O interface to support system IPL and software installation. In a system running AIX, it is not permitted to send an IPL operation to an array or a device that is a member of an array; the adapter implements this restriction by reporting only those disk drives that are not members of RAID-5 arrays in response to a Ready Test operation, when the mode bit is 1b.

Execute I/O can perform only one I/O operation at a time.

Command register
32h

Parameter register
Physical address of the parameter block in host memory.

Exceptions
The following exceptions may be indicated in the PCI Interrupt register:

- The command could not be successfully completed because of an I/O error or an attachment error. The system may be able to recover by using an alternative device.
- The command could not be successfully completed because of a catastrophic error. The code in the Error register defines the reason for the error.

Table 8. Parameter Block for Execute I/O

Byte	3	2			1	0
0	Operation	M	P	Reserved = 000000b	Reserved = 0000h	
4	Disk					
8	LBA					
12	Length					
16	Buffer_address					

Operation
This byte is coded as follows to specify the function to be performed:

01h Inquiry. This operation checks that the disk is ready. If it is, the following 24-byte descriptor is stored in host memory at the address in the buffer-address field.

Block_size
A 4-byte unsigned integer specifying the block size in bytes.

Capacity
A 4-byte unsigned integer specifying the disk capacity in blocks.

Serial_number
16 bytes containing the ASCII serial number of the resource.

ResourceID
A 4-byte unsigned integer identifying the resource.

02h Ready Test. The command completes successfully when all the attached resources are ready or when the time period in seconds defined in the length field has expired, whichever is the shortest time. The value of the physical (P) bit determines if these are logical or physical resources. If the mode bit (M) is 0b, the logical resources are of owning-module type DriverManualDisk; if the mode bit is 1b, they are of type DriverAutomaticDisk. A 4-byte unsigned integer that specifies the number of attached resources that are ready is stored in

host memory at the address provided in the buffer-address field. A list of resources is kept in the adapter. When the mode bit is 1b, RAID-5 resources are not included in the list of resources. A Ready Test operation must be issued before any other Execute I/O operations that have the mode bit set to 1b are issued.

03h Execute DC_StartTransaction IPN directive. The type of DDR must be either DT_Microchannel, DT_MicrochannelScatGat or DT_Null.

04h Diagnostic. If the adapter detects a degraded condition, this operation completes successfully; early models also stored an SRN for degraded conditions.

10h Read.

11h Write.

Mode (M)

The mode bit controls the definition of the disk field and the type of resources reported to a Ready Test operation.

If the mode bit is 0b, the disk field contains a resource ID.

If the mode bit is 1b then the disk field contains an index into a list of configured disks starting at zero, created at the last Ready Test Execute I/O operation.

If the operation is Ready Test and the mode bit is 0b, the logical resources listed are all of owning module type DriverManualDisk and can be RAID-5 or non-RAID resources.

If the operation is Ready Test and the mode bit is 1b, the logical resources listed are all of owning module type DriverAutomaticDisk and are all non-RAID resources.

Physical (P)

The physical bit value is only used during the Ready Test operation. If the physical bit is 0b, the disk field identifies a logical resource ID and the Ready Test operation refers to logical resources. If the physical bit is 1b, the disk field identifies a physical resource ID and the Ready Test operation refers to physical resources. If the physical resources attached to the adapter are configured into arrays, the number of logical resources may not be the same as the number of physical resources.

The physical bit is 0b for normal IPL operations to ensure that logical resources are used to find the required resource from which to read IPL data. The physical bit is set to 1b to obtain the serial number of each physical resource using Ready Test and Inquiry operations. This is executed after an unsuccessful completion of the Diagnostic operation to compare the serial numbers of good physical resources with those reported after a successful IPL process.

Disk An unsigned integer to select a particular resource according to the specified mode field (see the definition of the mode field for more details).

LBA An unsigned integer specifying the starting logical block address for a read or write request.

Length An unsigned integer specifying the number of blocks to be accessed in a read or write operation. It is assumed that the host memory buffer is large enough for the read data.

When the Ready Test operation is specified, the length field defines the number of seconds allowed for all the resources to become ready.

Buffer_address
The Micro Channel address of a buffer in host memory for read/write data or IPN directive or System Reference Number.

Resets

The actions taken for the various resets of the adapter are defined in this section.

Table 9. PCI SSA 4-Port RAID Adapters Reset Actions

	PCI Reset or Power-on Reset	**Command Reset**	**TotalReset (note 1)**	**Absolute Reset (note 5)**	**Link Reset**
Wrap/unwrap links during reset	Both SSA loops	Both SSA loops	No	Both SSA loops	No
POSTs	Yes	No	No	No	No
Reset configuration table	Both SSA loops	Both SSA loops	One SSA loop	Both SSA loops	No
Internally purge SSA commands (note 2)	Both SSA loops	Both SSA loops	One SSA loop	Both SSA loops	No
Async Alerts sent (note 3)	Yes	Yes	Yes	Yes	No
Reconfigure SSA network (note 4)	Yes	Yes	Yes	Yes	No

Notes:

1. The Device_reset SSA message is not supported.
2. SSA commands purged internally are reissued after the links have been reconfigured.
3. An Async_alert type code Remote Port Disabled is sent by the adjacent node when the link is wrapped. The master initiator should send a Master_alert to all other initiators to unconfigure this node from its configuration table.

   An Async_alert type code port now operational is sent by the adjacent node when the link is unwrapped and ready. The master initiator should send a Master_alert to all other initiators to reconfigure this node into its configuration table.
4. Reconfiguration

   After unwrapping each port, the initiator:

   - Issues a Query_node to all nodes from that port to walk the network and build the configuration table.

- Issues Quiesce to all nodes that support SSA-SCSI upper level protocol to purge all commands from this initiator and remove old return_paths in the target's initiator table.
- Issues Query_node again to each node to add the return_path to the target's initiator table.
- If the Query_node_reply responses indicate that this initiator should be the master, issues Configure_port specifying 'set normal mode' to all ports that are operational and a Master_alert specifying 'Port now operational' to each other primary initiator.

If, after reconfiguration for a PCI Reset operation, this is the only initiator in the network for an SSA loop, a Clear_queue message is issued to all the nodes attached to that SSA loop before any commands are issued, to ensure that all commands issued previously from any initiator are purged.

While the port is wrapped, another initiator may have detected this condition and elected to be a master initiator and issued Configure_port to nodes informing them it is the master. If the initiator completing its reset determines that it should be the master, it issues Configure_ports and Master_alerts as described above and becomes the master again.

5. The initial action taken by the firmware on receipt of an Absolute reset is to stop execution, set a showstop error code, and interrupt the host. If no command reset is received from the device driver after a period of time, the actions taken in the table (which are equivalent to a command reset) are taken. This reset applies only to the PCI SSA Multi-Initiator/RAID EL Adapter.

Vital Product Data

Vital Product Data (VPD) is information that uniquely defines the adapter card. This can be fetched from the location specified in the PCI Expansion ROM header; it can also be fetched from local bus address 0x3E070010.

The VPD fields supported are:

Part Number
This is the 8-digit ASCII-coded part number of the adapter card. If fewer than eight digits are used the leading digits are padded with zeros.

FRU Part Number
This is the 8-digit ASCII-coded part number of the field-replaceable card unit. If fewer eight digits are used the leading digits are padded with zeros.

Serial Number
This is an 8-digit ASCII-coded FRU serial number. This serial number is unique for the FRU part number and is part of the manufacturing serial number printed on the card. The serial number is in the range 00000000 through ZZZZZZZZ.

Engineering Change Level
This is a 10-digit ASCII-coded Engineering Change (EC) level number. This

number is updated whenever a hardware or microcode change is made on the card. If fewer than ten digits are used, the leading digits are padded with zeros.

Manufacturing Location
This 6-digit ASCII-coded field indicates the plant of manufacture.

ROS Level
This 8-digit ASCII-coded field indicates the ROS level of the card. A value of 00000000 in this field indicates that the POST code has detected a check-sum error in the code and a new version of code must be downloaded before the adapter can become fully operational. The SSA Adapter Microcode diskette, which is shipped with each adapter card, contains a version of adapter microcode that recovers this error in the event of the host system being unable to IPL because of this failure.

Loadable Microcode Level
This 2-digit ASCII-coded field indicates the version of loadable microcode required for satisfactory operation of this card.

02 PCI SSA 4-Port RAID Adapter

04 PCI SSA Multi-Initiator/RAID EL Adapter

Device Driver Level
This 2-digit ASCII-coded field indicates the minimum level of device-driver program required for this level of card.

Description of Function
This ASCII-coded field describes the function of this adapter card. For a PCI SSA 4-Port RAID Adapter this is ‘SSA-ADAPTER’.

DRAM Size (Z0)
This ASCII-coded field contains the characters ‘DRAM=’ followed by three characters indicating the size of the installed DRAM in megabytes.

Fast-Write Cache Size (Z1)
This ASCII-coded field contains the characters ‘CACHE=’ followed by a character indicating the size of the installed Fast-Write cache card in megabytes. If no cache is installed, the size character is ‘0’. This item appears in the VPD only for the PCI SSA Multi-Initiator/RAID EL Adapter.

Adapter ID (Z2)
This field contains the 8–byte AdapterID reported as 16 ASCII characters. This is the same as the SSA-UID of the modules that control the SSA links, with the least-significant bit = 0.

An example of the layout of the adapter card VPD is:

```
V P D (00) L X X
* P N (06) 1 2 3 4 5 6 7 8
* F N (06) 1 2 3 4 5 6 7 8
* S N (06) 1 2 3 4 5 6 7 8
* E C (07) 1 2 3 4 5 6 7 8 9 A
* M F (05) I B M 9 0 2
* R L (06) 0 0 0 0 0 0 0 1
```

```
* L L (03) 0 4
* D D (03) 0 0
* D S (08) S S A - A D A P T E R
* Z 0 (06) D R A M = 0 0 8
* Z 1 (06) C A C H E = 1
* Z 2 (10) 1 2 3 4 5 6 7 8 9 A B C D E F 0
```

The decimal number in () is the inclusive descriptor length divided by 2. Each descriptor field including the first 4 identification characters must be an even length. Some fields, for example, the *DS field, may have to be padded with a null character to make it an even length.

L is the inclusive VPD field length divided by 2, starting at the eighth byte, that is the first *.

XX is the CRC value. Starting from address X'00 08' to the end of the data field and calculated using the polynomial of 1 + X(exp 5) + X(exp 12) + X(exp 16) where CRC is initialized to 1s (this is the same as the CRC polynomial used for most diskette records).

An example of the VPD is:

```
Hex Address      Data
(Offset)

0000             56 50 44 00 2B (CRC)
0007             2A 50 4E 06 31 32 33 34 35 36 37 38
0013             2A 46 4E 06 31 32 33 34 35 36 37 38
001F             2A 53 4E 06 31 32 33 34 35 36 37 38
002B             2A 45 43 07 31 32 33 34 35 36 37 38 39 41
0039             2A 4D 46 05 49 42 4D 39 30 32
0043             2A 52 4C 08 30 30 30 30 30 30 30 30 30 30 30 31
0053             2A 4C 4C 03 30 34
0059             2A 44 44 03 30 30
005F             2A 44 53 08 53 53 41 2D 41 44 41 50 54 45 52 20
006F             2A 5A 30 06 44 52 41 4D 3D 30 30 38
007B             2A 5A 31 06 43 41 43 48 45 3D 31 20
0087             2A 5A 32 10 31 32 33 34 35 36 37 38 39 41 42 43 44 45 46 30
```

Chapter 3. PCI Interface

Characteristics

The adapter can operate as a PCI bus master with instantaneous data transfers up to 132 megabytes/second. The hardware interface uses the PCI 9060 module and provides the following facilities:

- 4 bytes of address and 4 bytes of data, with parity (single bit) on both.
- The adapter is a PCI bus target for access to I/O registers, Local Expansion ROM memory, and local bus memory. To avoid any possibility of a deadlock in the bridge between the PCI and local buses, the host system should not attempt accesses to the adapter local bus memory while data transfers could be taking place from the adapter. The host system should use only single-word fetches and stores when communicating with the adapter when it is a PCI target.

  The adapter is a PCI bus master for data transfer. Data is transferred over the PCI bus in the order that it is requested by the attached devices. Normally, the adapter does not buffer data internally. For optimum performance it is recommended that data transfers should start on a 4-byte boundary in host memory.
- PCI Configuration registers control I/O and memory base addresses, PCI Expansion ROM address, and PCI arbitration.

The following protocols are implemented by the adapter microcode:

- A register-based command protocol is provided for initializing the adapter and downloading updated microcode.
- A transaction protocol is provided for normal read/write access to the attached disk drives. Multiple transactions can be queued in the adapter and the attached devices. Each transaction is controlled by a GTCB. GTCBs are chained together in host memory and the adapter is informed that a GTCB has been set up by means of an adapter interrupt.

Data stored to non-RAID disks and data fetched from non-RAID disks is normally transferred from the PCI to the SSA disks using the Direct Master controls and is not staged in DRAM. In the following situations, data is staged through DRAM for transfers to non-RAID disks.

1. **Stage data on writes:**
   - PCI address odd byte aligned.
   - PCI address bit 31 does not equal bit 31 of the Host_GTCB_start_pointer field in the parameter data of the Initialise command. Direct Master transfers are limited to a maximum of 2 Gbyte address range; data needs to be staged for part of host memory for systems that have more than 2 Gbyte of memory.
   - Parameter data byte 0 bit 0 = 1 of the Initialise command. The device driver may need to force staging on writes by this means on certain systems that require compliance to PCI 2.1.
2. **Stage data on reads:**
   a. PCI address odd byte aligned.

b. PCI address bit 31 does not equal bit 31 of the Host_GTCB_start_pointer field in the parameter data of the Initialise command. Direct Master transfers are limited to a maximum of 2 Gbyte address range; data needs to be staged for part of host memory for systems that have more than 2 Gbyte of memory.

c. Parameter data byte 0 bit 0 = 1 of the Initialise command. The device driver may need to force staging on reads by this means on certain systems that suffer performance penalties if consecutive PCI 128 byte cycles are for different disks. When staging is forced, up to 4 Kbytes is requested for the same disk in consecutive bus requests.

The adapter is PCI 2.0 compliant. It can be forced to be PCI 2.1 compliant if byte 0 bit 0 = 1 and byte 0 bit 1 = 1b of the parameter block in the Initialise command. When these bits are set, all data to and from non-RAID disks is staged through DRAM. When data is staged, if a PCI fetch or store is stopped, that request will be re-submitted again independent of any other actions on the PCI bus. Also when bit 1 = 1, all data transfers are maximised to be 64 bytes long and aligned on 64–byte boundaries providing the starting address is 64–byte aligned. In this mode of operation, the number of operations per second possible and the adapter data rate are reduced.

The adapter has the following registers that are used for PCI bus communications:

- **PCI Configuration Registers**

  PCI Configuration registers are accessible in configuration space. Only registers in the predefined 64-byte header region are supported. These registers uniquely identify the adapter and allow it to be controlled generically. These registers are defined in “PCI Configuration Registers” on page 32.

- **Local Configuration Registers**

  The Local Configuration registers are part of the runtime-registers address space for which a PCI base address is included in a PCI Configuration register. They define address ranges and PCI and local-bus base addresses for local address space, expansion-ROM space, and local-direct-master-to-PCI space. These registers are defined in “Local Configuration Registers” on page 42.

- **Shared Runtime Registers**

  The Shared Runtime registers are part of the runtime-registers address space for which a PCI base address is included in a PCI Configuration register. They are used for communication between the host and the adapter during normal operation. The main registers are:

  – **Parameter register** which is used by the command protocol to transfer a command parameter or a pointer to a parameter block.

  – **Command register** which is used by the command protocol for the host to initiate a command.

  – **Error register** which is used by the adapter to identify to the host errors detected by the adapter.

  – **PCI Interrupt Doorbell register** which is used by the adapter to interrupt the host.

  – **Adapter Interrupt Doorbell register** which is used by the host to interrupt the adapter.

These registers are defined in “Shared Runtime Registers” on page 51.

- **Local DMA Registers**

  The Local DMA registers are used to set up the local DMA channels. They are not accessible to the PCI bus and are defined in PCI 9060 specification and not in this document.

PCI Configuration Registers

The fields of the PCI configuration registers are mapped into PCI configuration space as follows:

PCI Config Addr	31	23	15	7 0
00h	Device ID		Vendor ID	
04h	Status		Command	
08h	Class Code			Revision ID
0Ch	BIST	Header Type	Latency Timer	Cache Line Size
10h	PCI Base Address for Memory Mapped Runtime Registers			
14h	PCI Base Address for I/O Mapped Runtime Registers			
18h	PCI Base Address for Local Address Space 0			
1Ch	Reserved			
20h	Reserved			
24h	Reserved			
28h	Reserved			
2Ch	Reserved			
30h	PCI Base Address for Local Expansion ROM			
34h	Reserved			
38h	Reserved			
3Ch	Max_Lat	Min_Gnt	Interrupt Pin	Interrupt Line

These registers are initialized to the values shown after a power on and a PCI reset. They are not changed by the adapter during an adapter software reset.

PCI Configuration ID Register

The PCI Configuration ID register uniquely identifies the adapter card.

PCI address: Offset 00h

CFE Address: Offset 00h

Initialized value: 0045 1014h

The register cannot be written from the PCI bus

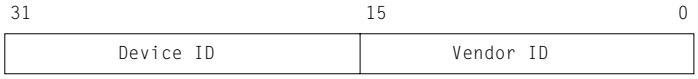


Figure 4. PCI Configuration ID Register

Table 10. PCI Configuration ID Register

Bits	Description	Writeable	Initial value
15 through 0	**Vendor ID.** Identifies the manufacturer of the adapter as issued by the PCI SIG.	Local only	1014h
31 through 16	**Device ID.** Identifies the adapter.	Local only	0045h

After a reset, the vendor-ID and device-ID fields are loaded from the flash EEPROM by the local processor as part of initialization. When the loading of the configuration registers has completed, the local-INIT-status bit in the EEPROM-control, PCI-command-codes, user-I/O-control, and init-control register is set to one.

PCI Command Register

The PCI command register provides coarse control over the adapter's ability to generate and respond to PCI cycles. When the register is 0000h, the adapter is logically disconnected from the PCI bus for all accesses except configuration accesses. The host system controls how the adapter can use the PCI bus by setting bits in this register.

PCI address: Offset 04h

CFE Address: Offset 04h

Initialized value: 0000h

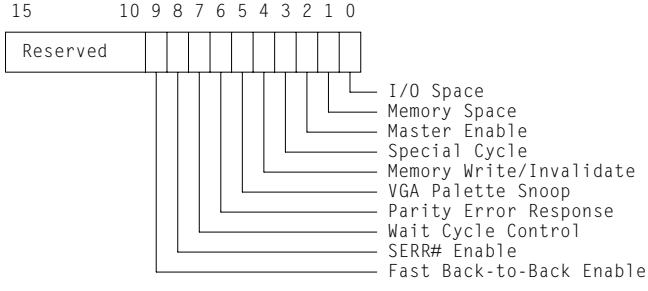


Figure 5. PCI Command Register

Table 11. PCI Command Register

Bits	Description	Writeable	Initial value
0	**I/O Space.** When set to 1b, the adapter is allowed to respond to I/O space accesses. When set to 0b, the adapter is disabled from responding to I/O space accesses.	Yes	0
1	**Memory Space.** When set to 1b, the adapter is allowed to respond to memory space accesses. When set to 0b, the adapter is disabled from responding to memory space accesses.	Yes	0
2	**Master Enable.** When set to 1b, the adapter is allowed to behave as a bus master. When set to 0b, the adapter is disabled from generating bus master accesses.	Yes	0
3	**Special Cycle.** This bit is not supported	No	0
4	**Memory Write/Invalidate.** This bit is not supported	No	0
5	**VGA Palette Snoop.** This bit is not supported	No	0
6	**Parity Error Response.** When set to 1b, parity checking is enabled. When set to 0b, parity errors are ignored and operations continue.	Yes	0
7	**Wait Cycle Control.** When set to 0b, the adapter does not perform address/data stepping. This bit is hard-wired to 0b.	No	0
8	**SERR# Enable.** When set to 1b, the SERR# driver is enabled. When set to 0b, the SERR# driver is disabled. This bit (and bit 6) must be on to report address parity errors.	Yes	0
9	**Fast Back-to-Back Enable.** When set to 1b, fast back-to-back transfers can occur to different agents on the bus. When set to 0b, fast back-to-back transfers can only occur to the same agent as the previous cycle.	Yes	0
15 through 10	Reserved	No	0

PCI Status Register

The PCI status register is used to record status information for PCI bus related events. Reads to this register behave normally. Writes to bits however can only cause bits to be reset. A bit is reset whenever the register is written and the data in the corresponding data position is a 1b.

PCI address: Offset 06h

CFE Address: Offset 06h

Initialized value: 0280h

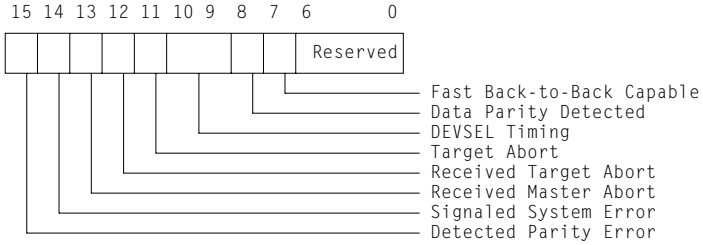


Figure 6. PCI Status Register

Table 12. PCI Status Register

Bits	Description	Writeable	Initial value
6 through 0	Reserved	No	0
7	**Fast Back-to-Back Capable.** When this bit is set to 1b, the adapter can accept fast back-to-back transactions.	No	1
8	**Data Parity Detected.** When this bit is set to 1b, the adapter has detected that the PERR# signal has gone active and the parity-error-response bit in the command register is set. Writing a 1b to this bit clears the bit to a 0b.	Yes	0
10 through 9	**DEVSEL Timing.** These bits encode the timing of DEVSEL#. 00b is fast, 01b is medium, 10b is slow and 11b is reserved. These bits indicate the slowest time that the adapter asserts DEVSEL# for any bus command except Configuration Read and Configuration Write.	No	01b
11	**Signaled Target Abort.** When this bit is set to 1b, the adapter as a target has terminated a PCI transaction with target abort. Writing a 1b to this bit resets the bit to 0.	Yes	0
12	**Received Target Abort.** When this bit is set to 1b, the adapter as a master has received a target-abort signal that has terminated a PCI transaction. Writing a 1b to this bit resets the bit to 0.	Yes	0
13	**Received Master Abort.** When this bit is set to 1b. the adapter has terminated a PCI transaction with master abort. Writing a 1b to this bit resets the bit to a 0.	Yes	0
14	**Signaled System Error.** When this bit is set to 1b, the adapter has asserted SERR# to indicate that it is reporting a system error. Writing a 1b to this bit resets the bit to a 0.	Yes	0

Table 12. PCI Status Register (continued)

Bits	Description	Writeable	Initial value
15	**Detected Parity Error.** When this bit is set to 1b, the adapter has detected a PCI bus parity error even when parity error handling is disabled. Writing a 1b to this bit resets the bit to a 0.	Yes	0

PCI Revision ID Register

The PCI Revision ID register identifies the revision level of the adapter.

PCI address: Offset 08h

CFE Address: Offset 08h

Initialized value: 03h

The register cannot be written from the PCI bus

Figure 7. PCI Revision ID Register

PCI Class Code Register

The PCI Class Code register identifies the generic function of the adapter.

PCI address: Offset 09h

CFE Address: Offset 09h

Initialized value: 0C0200h

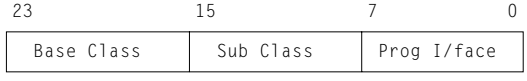


Figure 8. PCI Class Code Register

Table 13. PCI Class Code Register

Bits	Description	Writeable	Initial value
7 through 0	**Programming Interface.** This field is 00h because no programming interface has been defined.	Local only	00h
15 through 8	**Subclass Encoding.** This field is set to 02h to indicate that the adapter is a SSA adapter subclass.	Local only	02h
23 through 16	**Base Class Encoding.** This field is set to 0Ch to identify that the adapter is a serial-bus controller.	Local only	0Ch

PCI Cache Line Size Register

The PCI Cache Line Size register specifies the system cache line size. This is not supported on the adapter.

PCI address: Offset 0Ch

CFE Address: Offset 0Ch

Initialized value: 00h

The register cannot be written from PCI or Local Bus

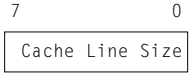


Figure 9. PCI Cache Line Size Register

PCI Latency Timer Register

The PCI Latency Timer register specifies, in units of PCI bus clocks, the value of the latency timer for the adapter when a bus master. The host system sets this register. A value of 40h is recommended for the latency timer to allow the adapter to burst for 256-byte transfers on the PCI bus.

PCI address: Offset 0Dh

CFE Address: Offset 0Dh

Initialized value: 00h

Figure 10. PCI Latency Timer Register

The latency timer, which is specified in units of PCI bus clocks, is the amount of time the adapter when a bus master can burst data on the PCI bus.

PCI Header Type Register

The PCI Header register identifies that the adapter does not support multiple functions because bit 7 is 0b.

PCI address: Offset 0Eh

CFE Address: Offset 0Eh

Initialized value: 00h

The register cannot be written from the PCI bus

Figure 11. PCI Header Type Register

PCI Built-In Self Test Register

The PCI Built-In Self Test register controls the execution of built-in self tests (BIST).

PCI address: Offset 0Fh

CFE Address: Offset 0Fh

Initialized value: 80h

Only bit 6 can be written from the PCI bus. Bits 7 and 3 through 0 can be written from the local bus.

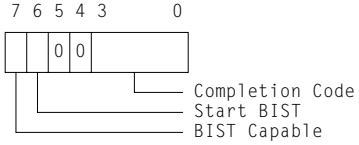


Figure 12. PCI Built-In Self Test Register

Table 14. PCI Built-In Self Test (BIST) Register

Bits	Description	Writeable	Initial value
3 through 0	**BIST Completion Code.** A value of 0h in this field means that the adapter has passed its BIST. A nonzero value indicates that BIST was unsuccessful and the value indicates a failure code.	Local only	0h
5 through 4	Reserved	No	00b
6	**Start BIST.** This bit is written with 1b from the PCI bus to invoke BIST. An interrupt is generated to the local processor. The local processor resets this bit when BIST is complete. Software should fail the adapter if BIST is not complete within 2 seconds.	Yes	0
7	**BIST Capable.** This bit is set to 1b to indicate that the adapter is capable of BIST.	Local only	1

PCI Base Address Register for Memory Access to Runtime Registers

PCI address: Offset 10h

CFE Address: Offset 10h

Initialized value: 00000000h

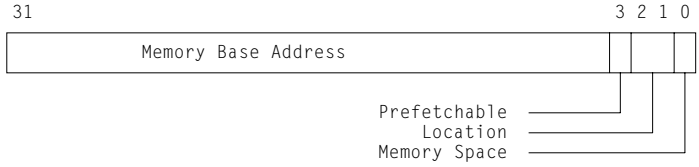


Figure 13. Base Address Register for Memory Access to Runtime Registers

Table 15. PCI Base Address Register for Memory Access to Runtime Registers

Bits	Description	Writeable	Initial value
0	**Memory Space Indicator.** This bit is set to 0b to indicate register maps into memory space.	No	0h
2 through 1	**Register Location.** This field has the value 00b to indicate that the runtime registers can be mapped anywhere in 32 bit memory address space.	No	00b
3	**Prefetchable.** This bit is set to 0b. A value of 1b indicates that there are no side effects on reads.	No	0
6 through 4	**Memory Base Address.** These are bits 6 through 4 of the memory base address that is used to access the adapter runtime registers. The value is 00b (addresses are on 128 byte boundaries).	No	0
31 through 7	**Memory Base Address.** This is the memory base address that is used to access the adapter runtime registers.	Yes	0

PCI Base Address Register for I/O Access to Runtime Registers

PCI address: Offset 14h

CFE Address: Offset 14h

Initialized value: 00000001h

Figure 14. Base Address Register for I/O Access to Runtime Registers

Table 16. PCI Base Address Register for I/O Access to Runtime Registers

Bits	Description	Writeable	Initial value
0	**Memory Space Indicator.** This bit is set to 1b to indicate that the register maps into I/O space.	No	1
1	Reserved	No	0
6 through 2	**I/O Base Address.** These are bits 6 through 2 of the base address for I/O access to the adapter runtime registers. They have the value 00000b (addresses are on 128 byte boundaries).	No	00000b

Table 16. PCI Base Address Register for I/O Access to Runtime Registers (continued)

Bits	Description	Writeable	Initial value
31 through 7	**I/O Base Address.** This is base address for I/O access to the adapter runtime registers.	Yes	0

PCI Base Address Register for Memory Access to Local Address Space 0

PCI address: Offset 18h

CFE Address: Offset 18h

Initialized value: 00000000h

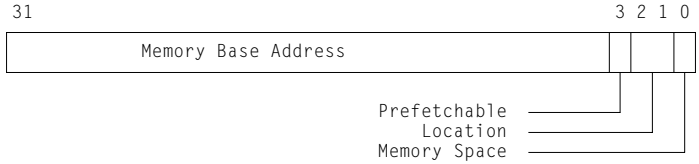


Figure 15. Base Address Register for Memory Access to I/O Local Address Space

Table 17. PCI Base Address Register for Memory Access to I/O Local Address Space

Bits	Description	Writeable	Initial value
0	**Memory Space Indicator.** This bit is set to 0b to indicate that the register maps into memory space.	No	0
2 through 1	**Register Location.** This field has the value 00b to indicate that Local Address Space 0 can be mapped anywhere in 32-bit memory address space.	No	0
3	**Prefetchable.** This bit is set to 0b. A value of 1b indicates that there are no side effects on reads.	No	0
31 through 4	**Memory Base Address.** This is the memory base address that is used to access the adapter local address space.	Yes	0

PCI Expansion ROM Base Register

PCI address: Offset 30h

CFE Address: Offset 30h

Initialized value: 00000001h

Figure 16. PCI Expansion ROM Base Register

Table 18. PCI Expansion ROM Base Register

Bits	Description	Writeable	Initial value
0	**Address Decode Enable.** When this bit is set to 1b, it indicates that the adapter accepts accesses to the expansion ROM address. A value of 0b indicates the adapter does not accept accesses to expansion ROM space.	Yes	1
10 through 1	Reserved	No	0
31 through 11	**Expansion ROM Base Address.** This field is the upper 21 address bits. Software can determine what alignment is supported by writing a value of all 1s to the address portion of this register and reading the value back. The adapter returns zeros in all undefined bits, effectively specifying what alignment is supported.	Yes	0

PCI Interrupt Line Register

The value of the PCI-interrupt-line register indicates to which input of the system interrupt controller the adapter's interrupt line is connected.

PCI address: Offset 3Ch

CFE Address: Offset 3Ch

Initialized value: 00h

Figure 17. PCI Interrupt Line Register

PCI Interrupt Pin Register

The value of the PCI interrupt-pin register indicates which interrupt pin in used by the adapter.

PCI address: Offset 3Dh

CFE Address: Offset 3Dh

Initialized value: 01h

The register cannot be written from the PCI bus

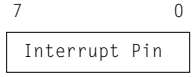


Figure 18. PCI Interrupt Pin Register

The value of this register is 01h to indicate that the INTA# pin is used.

PCI Min_Gnt Register

The PCI Min_Gnt register specifies the burst period required by the adapter.

PCI address: Offset 3Eh

CFE Address: Offset 3Eh

Initialized value: 0Ah

The register cannot be written from the PCI bus

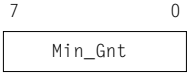


Figure 19. PCI Min_Gnt Register

The value of this register is the length of burst period required by the adapter assuming a clock rate of 33 MHz. The value specifies a period of time in units of ¼ microsecond.

PCI Max_Lat Register

The PCI Max_Lat register specifies how often the adapter needs to gain access to the PCI bus.

PCI address: Offset 3Fh

CFE Address: Offset 3Fh

Initialized value: 00h

The register cannot be written from the PCI bus

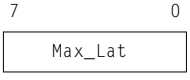


Figure 20. PCI Max_Lat Register

The value of maximum hold-off is specified as a time in units of ¼ microsecond.

Local Configuration Registers

The following registers define address ranges and PCI and local-bus base addresses for local-address space, expansion-ROM space, and local-direct-master-to-PCI space. They are part of the runtime-register space for which a PCI base address is included in a PCI configuration register.

PCI Offset Addr	31	23	15	7 0
00h	Local Address Space 0 Range for PCI to Local Bus			
04h	Local Address Space 0 Base Address (Remap) for PCI to Local Bus			
08h	Reserved			
0Ch	Reserved			
10h	Local Expansion ROM Range for PCI to Local Bus			
14h	Local Expansion ROM Local Base Address (Remap)			
18h	Local Bus Region Descriptors for PCI to Local Accesses			
1Ch	Local Bus Range for Direct Master to PCI			
20h	Local Bus Base Address for Direct Master to PCI Memory			
24h	Reserved			
28h	PCI Base Address (Remap) for Direct Master to PCI			
2Ch	PCI Configuration Address Register for Direct Master to PCI IO/CFG (not used)			

These registers are initialized to the values shown after a power on, a PCI reset, and an adapter software reset.

Local Address Space 0 Range Register for PCI to Local Bus

The local-address-space-0-range register identifies the range of addresses for local address space 0.

PCI address: Offset 00h from runtime base address

CFE address: Offset 80h

Initialized value: FFFFFFF8h

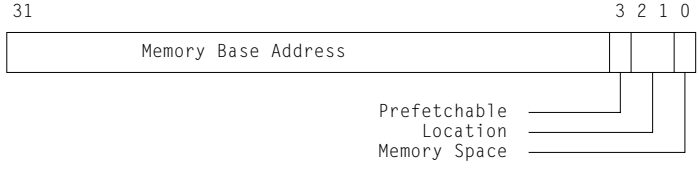


Figure 21. Local Address Space 0 Range Register for PCI to Local Bus

Table 19. Local Address Space 0 Range Register for PCI to Local Bus

Bits	Description	Writeable	Initial value
0	**Memory Space.** This bit is set to 0b to indicate that the register maps into memory space.	Yes	0
2 through 1	**Register Location.** This field has the value 00b to indicate that local address space 0 can be mapped anywhere in 32-bit memory address space.	Yes	00b

Table 19. Local Address Space 0 Range Register for PCI to Local Bus (continued)

Bits	Description	Writeable	Initial value
3	**Prefetchable.** This bit is 1b to indicate that there are no side effects on reads to memory space.	Yes	1
31 through 4	**Local Address Space Range.** This field specifies which PCI address bits are used to decode a PCI access to local bus space 0. Each of the bits corresponds to an address bit (bit 31 corresponds to address bit 31). A value of 1 is written to all bits that should be included in decode and a 0 to others.	Yes	FFFFFFFh

Local Address Space 0 Base Address (Remap) Register for PCI to Local Bus

The local-address-space-0-base-address (remap) register, for PCI to local bus, register identifies the remapping of PCI address bits to local-bus address bits for local address space.

Offset 04h from runtime base address

CFE Address: Offset 84h

Initialized value: 3E000001h

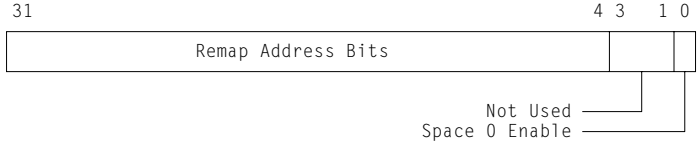


Figure 22. Local Address Space 0 Base Address (Remap) Register for PCI to Local Bus

Table 20. Local Address Space 0 Base Address (Remap) Register for PCI to Local Bus

Bits	Description	Writeable	Initial value
0	**Space 0 Enable.** When this bit is set to 1b, decodes are enabled for PCI addresses for direct slave accesses to local space 0. A value of 0b disables that decode.	Yes	1
1	Not used	Yes	0
3 through 2	Not used when local space is mapped into memory space	Yes	00b
31 through 4	**Remap Address.** This field remaps the PCI address to local address space 0 into a local address space. These bits are used to remap (replace) the PCI address bits used in decode as the local address bits.	Yes	3E00000h

The default remap address 3E0000 is the start address of SRAM on the local bus (bits 31 through 4). The range of local bus addresses is only 16 bytes, so the remap address in this register should be changed to access other SRAM locations.

Certain locations in SRAM have fixed definitions. These are:

Local Bus Address (hex)	31	23	15	7 0
3E070010 through 3E07009B	Vital Product Data			
3E07009C through 3E0700DB	Not Defined			
3E0700DC	Reserved = 0			Template
3E0700E0	Reserved = 0	Adapter Error Code		
3E0700E4	Reserved = 0	Dump Status	Trace Address	
3E0700E8 through 3E0700EF	Disk Unique ID			
3E0700F0	Dump Start LBA			
3E0700F4 through 3E070FF	Not Defined			

Template
This field reports the template to be used for logging the error. The template is SSA_HDW_ERROR (0Ah) for all failures detected during POST2A that report SS_POST2A_FAIL in the adapter-error register.

Adapter Error Code
This field provides details of the error detected in POST2A. The adapter error codes are defined in “Adapter Error Logging Data” on page 226.

Trace Address
This field contains trace information when the error register contains SS_SENSE.

Dump Status
This field indicates the presence of a dump on disk.

Disk Unique ID
This field contains the unique ID (UID) of the disk that holds the adapter dump following a showstop.

Dump Start LBA
The field contains the location of the dump on disk.

Local Expansion ROM Range Register for PCI to Local Bus

The local-expansion-ROM-range register, for PCI to local bus, identifies the range of addresses for local bus expansion ROM.

PCI address: Offset 10h from runtime base address

CFE Address: Offset 90h

Initialized value: FFFF0000h

Figure 23. Local Expansion ROM Range Register for PCI to Local Bus

Table 21. Local Expansion ROM Range Register for PCI to Local Bus

Bits	Description	Writeable	Initial value
10 through 0	Not used	Yes	0
31 through 11	**PCI Address Range.** This field specifies which PCI address bits are used to decode a PCI to local bus expansion ROM. Each of the bits corresponds to an address bit (bit 31 corresponds to address bit 31). A value of 1b is written to all bits that should be included in the decode and 0b to all other bits.	Yes	Bits 31 through 12 = FFFFFh, Bit 11 = 0b

Local Expansion ROM Local Base Address (Remap) Register

The local-expansion-ROM-local-base-address (remap) register identifies the remapping of PCI address bits to local bus address bits for local expansion ROM. It is also used to control the BREQo signal.

PCI address: Offset 14h from runtime base address

CFE Address: Offset 94h

Initialized value: 3E00200Fh

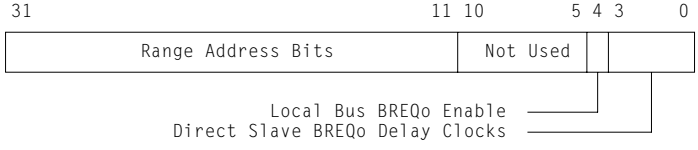


Figure 24. Local Expansion ROM Local Base Address (Remap)

Table 22. Local Expansion ROM Local Base Address (Remap)

Bits	Description	Writeable	Initial value
3 through 0	**BREQo Delay.** This field defines the number of local bus clocks for which a direct-slave-HOLD request is pending and a local-direct-master access is in progress and not being granted the bus (HOLDA) before asserting BREQo (LSB = 8 clocks). Asserting BREQo causes an interrupt to the microprocessor that is used to avoid a deadlock situation with the PCI bus.	Yes	Fh
4	**BREQo Enable.** When this bit is set to 1b, the BREQo signal can be asserted.	Yes	0
10 through 5	Not used	No	0
31 through 11	**Remap Address.** This field remaps the PCI address of PCI expansion ROM space into a local address space. These bits are used to remap (replace) the PCI address bits used in decode as the local address bits.	Yes	Bits 31 through 12 = 3E002h, Bit 11 = 0b

Local Bus Region Descriptor for PCI to Local Accesses Register

The local-bus-region-descriptor register, for PCI to local accesses, is used to initialize characteristics of the local bus.

PCI address: Offset 18h from runtime base address

CFE Address: Offset 98h

Initialized value: 40430043h

Figure 25. Local Bus Region Descriptor for PCI to Local Accesses Register

Table 23. Local Bus Region Descriptor for PCI to Local Accesses Register

Bits	Description	Writeable	Initial value
1 through 0	**Memory Space 0 Local Bus Width**. This field is 11b to indicate a 32 bit width.	Yes	11b
5 through 2	**Memory Space 0 Internal Wait States.**	Yes	0
6	**Memory Space 0 Ready Input Enable.** A 1b value enables Ready input and a 0b value disables Ready input.	Yes	1
7	**Memory Space 0 Bterm Input Enable.** A 1b value enables Bterm input and a 0b value disables Bterm input.	Yes	0

Table 23. Local Bus Region Descriptor for PCI to Local Accesses Register (continued)

Bits	Description	Writeable	Initial value
15 through 8	Not used	Yes	0
17 through 16	**Expansion ROM Space Local Bus Width.** When this field is 11b, it indicates a 32-bit width.	Yes	11
21 through 18	**Expansion ROM Space Internal Wait States.**	Yes	0
22	**Expansion ROM Space Ready Input Enable.** When this bit is set to 1b, ready input is enabled; when set to 0b, ready input is disabled.	Yes	1
23	**Expansion ROM Space Bterm Input Enable.** When this bit is set to 1b, Bterm input is enabled; when set to 0b, Bterm input is disabled.	Yes	0
24	**Memory Space 0 Burst Enable.** When this bit is set to 1b, bursting is enabled; when set to 0b, bursting is disabled.	Yes	0
25	Not used	Yes	0
26	**Expansion ROM Space Burst Enable.** When this bit is set to 1b, bursting is enabled; when set to 0b, bursting is disabled.	Yes	0
27	**Direct Slave PCI Write Mode.** When this bit is set to 0b, the adapter disconnects when the direct-slave-write FIFO is full; when set to 1b, the adapter deasserts TRDY when the write FIFO is full.	Yes	0
31 through 28	**PCI Target Retry Delay Clocks.** When bit 27 is 1b, this field is the value (multiplied by 8) of the number of bus clocks after a PCI-local read or write access and not successfully completing a transfer.	Yes	4h

Local Range Register for Direct Master to PCI

The local-range register, for direct master to PCI, is used to identify the range of local-bus addresses that can be used to decode a local to PCI bus access.

PCI address: Offset 1Ch from runtime base address

CFE Address: Offset 9Ch

Initialized value: 80000000h

Read/Write

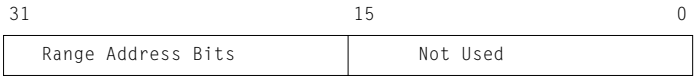


Figure 26. Local Range Register for Direct Master to PCI

Table 24. Local Range Register for Direct Master to PCI

Bits	Description	Writeable	Initial value
15 through 0	Not used (64KB increments)	No	0
31 through 16	**Local Address Range Bits.** This field specifies which local-bus address bits are used to decode a local to PCI bus access. Each of the bits correspond to an address bit (bit 31 corresponds to address bit 31). A value of 1b is written to all bits that should be included in the decode and a 0b to all others.	Yes	8000h

Local Bus Base Address Register for Direct Master to PCI Memory

The local-bus-base-address register, for direct master to PCI memory, is used to identify the local-bus address bits used to decode a local to PCI memory access.

PCI address: Offset 20h from runtime base address

CFE Address: Offset A0h

Initialized value: 80000000h

Read/Write

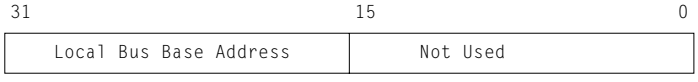


Figure 27. Local Bus Base Address Register for Direct Master to PCI Memory

Table 25. Local Bus Base Address Register for Direct Master to PCI Memory

Bits	Description	Writeable	Initial value
15 through 0	Not used	No	0
31 through 16	**Local Bus Base Address.** This field specifies which local-bus address bits are used to decode a local to PCI bus access. Each of the bits corresponds to an address bit (bit 31 corresponds to address bit 31). A value of 1b is written to all bits that should be included in the decode and a 0b to all others.	Yes	8000h

PCI Base Address (Remap) Register for Direct Master to PCI

The PCI-base-address (remap) register, for direct master to PCI, is used to identify the remapping of local-bus address bits to PCI addresses.

PCI address: Offset 28h from runtime base address

CFE Address: Offset A8h

Initialized value: 00000001h

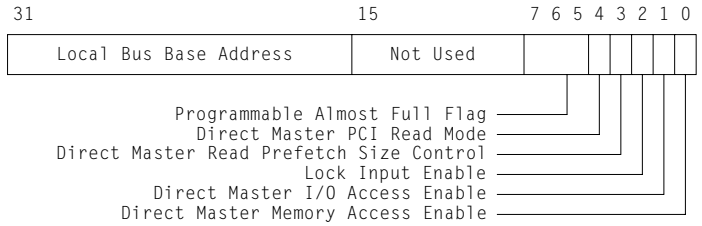


Figure 28. PCI Base Address (Remap) Register for Direct Master to PCI

Table 26. PCI Base Address (Remap) Register for Direct Master to PCI

Bits	Description	Writeable	Initial value
0	**Direct Master Memory Access Enable.** When this bit is set to 1b, decode of direct-master-memory accesses is enabled; when set to 0b, decode of direct-master-memory accesses is disabled.	Yes	1
1	**Direct Master I/O Access Enable.** This bit is set to 0b as direct-master-I/O accesses are disabled.	Yes	0
2	**LOCK Input Enable.** When this bit is set to 1b, LOCK input is enabled, allowing PCI locked sequences; when set to 0b, LOCK input is disabled.	Yes	0
3	**Direct Master Read Prefetch Size Control.** When this bit is set to 1b, the PCI9060 reads up to 4 Lwords from the PCI bus for each direct-master-burst-read access; when set to 0b, the PCI9060 continues to prefetch read data until the direct-master access is finished.	Yes	0
4	**Direct Master PCI Read Mode.** When this bit is set to 1b, the PCI9060 keeps the PCI bus and deasserts IRDY when the read FIFO becomes full; when set to 0b, the PCI9060 releases the PCI bus when the read FIFO becomes full.	Yes	0
7 through 5	**Programmable Almost Full Flag.** When the number of entries in the 8-deep direct-master-write FIFO exceeds this value, the output pin DMAF# is asserted low.	Yes	0
15 through 8	Not used	Yes	0
31 through 16	**Remap Address.** This field is a remap of local to PCI space into PCI memory space. The bits in this register remap (replace) the local-address bits used in decode as the PCI address bits.	Yes	0

Shared Runtime Registers

It is recommended that these registers are mapped to PCI I/O space; they can also be mapped to PCI memory space. The base address of the runtime registers is programmable using the configuration register. The PCI offset from this base address is defined for each register.

PCI Offset Address	31	23	15	7 0
40h	Parameter Register			
44h	Adapter Flags		Adapter Error	Adapter Command
48h	Adapter Error Code Register			
4Ch	IEXECIO Register			
50h	Interrupt Control Register			
54h	Mailbox Register 5 (not used)			
58h	Mailbox Register 6 (not used)			
5Ch	Mailbox Register 7 (not used)			
60h	Adapter Interrupt Doorbell Register			
64h	PCI Interrupt Doorbell Register			
68h	Interrupt Control / Status			
6Ch	EEPROM Control, PCI Command Codes, User I/O Control, INIT Control			

These registers are initialized to the values shown after a power on, a PCI reset and an adapter software reset.

Parameter Register

The parameter register is used by the command and host-slave-operation protocols. It contains either the command parameters or a pointer to a parameter block. The parameter block must be aligned on a 4-byte boundary.

PCI address: Offset 40h from runtime base address

CFE Address: Offset C0h

Initialized value: 0000 0000h

Figure 29. Parameter Register

The host should not change the parameter register while the adapter is executing a command. There is no interlock in the adapter hardware that prevents the parameter register being changed at this time.

Adapter Command Register

The adapter-command register is used by the command protocol. It is written by the host to initiate a command. The host should also write to the adapter-interrupt register after setting the adapter-command register to generate an interrupt to the adapter.

PCI address: Offset 44h from runtime base address

CFE Address: Offset C4h

Initialized value: 00h

Figure 30. Adapter Command Register

The host should not change the adapter-command register while the adapter is executing a command. There is no interlock in the adapter hardware that prevents the adapter-command register being written while executing a command.

Adapter Error Register

The adapter-error register identifies an error detected by the adapter, examples are: host programming errors, adapter microcode errors, and adapter hardware errors.

PCI address: Offset 45h from runtime base address

CFE Address: Offset C5h

Initialized value: 00h

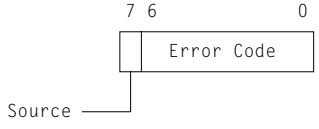


Figure 31. Adapter Error Register

If the source bit is 1b, the error code originated from the IPN kernel; if the source bit is 0h, the error originated from microcode other than the kernel.

The error codes, in hexadecimal, are:

01 SS_INSANE
Adapter error: out-of-control error trap (should be preceded by trace point)

02 SS_WRONG_INT
Host error: an unexpected adapter interrupt occurred

03 SS_WRONG_INI_PARMS
Host error in Initialize command parameters

04 SS_NOT_INIT_CMD
Host error: command was not Initialize

05 SS_PARMS_NOT_INLINE
Host error: all parameters should be in-line

06 SS_TOO_MANY_REQS
Host error: too many simultaneous requests

07 SS_MIAMI_DMA_FAILED
Host error: host disabled DMA in process

08 SS_NOT_IMPLEMENTED
Microcode not implemented

09 SS_KNL_TRAP
Kernel detected error

0A SS_KNL_INSANE
Kernel detected out-of-control error

0B SS_PARMDDRTYPE_INVALID
Parm DDR should be DT_PCI or DT_Null

0C SS_NOT_DNLD_CMD,
Not Download command (during download reset)

0D SS_DNLD_TOO_BIG,
Code Download length too big

0E SS_DNLD_TOO_MANY_SG_ELS
Download has too many scatter/gather elements

0F SS_DNLD_SGLN_MISMATCH
Download scatter/gather fragments don't add up

10 SS_DNLD_LRC_FAILURE

11 SS_SIC_CLASS1
An SSA loop interface indicated a class 1 problem

12 SS_WRONG_XIO_OPCODE
Invalid operation requested in Execute I/O

13 SS_ASSERT
An assert has been hit

14 SS_DBG_STOP
Debug service stop

15 SS_XIL_ERROR
Adapter detected error

16 SS_XIL_INSANE
Error in adapter error handler

17 SS_SIC_DMA_FAILED

18 SS_SLVOP_BUSY

19 SS_INVALID_HOST_SLAVE_OP

1A SS_GTCB_BEFORE_INITIALISE

1B SS_GTCB_PROTOCOL_ERROR

1C SS_INVALID_DOORBELL

1D SS_DNLD_FLASH_FAILURE

1E SS_INVALID_OT_DONE

1F SS_STORAGE
Watchdog failed to get storage before timeout

20 SS_VSC
TransferToHost transaction timeout

21 SS_POST2A_FAIL
POST2A error checking first 1 MB of DRAM

22 SS_TIMEOUT
Timeout on transaction from host (> 2 minutes)

23 SS_SENSE
Additional trace information is available in SRAM addresses 3E0700E4h through 3E0700E5h and a dump might have been written to a disk whose address is defined in SRAM address 3E0700E8h through 3E0700EFh, starting at an LBA defined in SRAM addresses 3E0700F0h through 3E0700F3h.

24 SS_THIRD_PARTY_RESET
SSA absolute reset received from another adapter

25 SS_LINK_CONFIG_FAILED
SSA link configuration failed

40 XER_NoPrecedingReadyTest
All Execute I/O operations must be preceded by a Ready test after a reset or power on

41 XER_M1DiskGTResourceCount .
Disk field is too large when Mode field is 1b

42 XER_IpnBadResult
IPN transaction failed

43 XER_M0ResourceNotInList
Resource not available when Mode field is 0b

44 XER_ResourceNotRecognised

45 XER_DevNoLongerAccessible
Resource no longer accessible

46 XER_ReadWriteFailed

47 XER_OpenFailed

Adapter Flags Register

The adapter-flags register is used to communicate states of the adapter to the host.

PCI address: Offset 46h from runtime base address

CFE Address: Offset C6h

Initialized value: 8000h

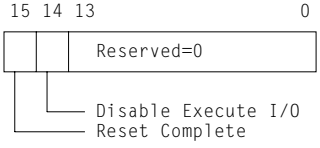


Figure 32. Adapter Flags Register

Table 27. Adapter Flags Register

Bits	Description	Writeable	Initial value
13 through 0	Not used	Yes	0
14	**Disable Execute I/O.** This bit allows host systems to avoid sending an Execute I/O command while another command is being executed. The host system can set this bit to prevent another component of the host from issuing an Execute I/O command. The adapter ignores this bit.	Yes	0

Table 27. Adapter Flags Register (continued)

Bits	Description	Writeable	Initial value
15	**Reset Complete.** This bit is set to 1b by the adapter at the end of a reset operation. The host should reset this bit to 0b before initiating a reset operation. The host can then use the bit as an indicator: when the adapter sets it to 1b again, the reset operation is complete. During the period from the receipt of a software reset to the setting of the reset-complete bit, the local-configuration registers might contain values initialized by the hardware that are not correct values set by the microcode. This can mean that the range of addresses used for Expansion ROM accesses is 64K rather than 4K; as a result, during this period, the adapter might respond to PCI bus addresses outside its correct range. To avoid this problem, before issuing a software reset, the host should disable PCI Expansion ROM address decodes by: • Resetting bit 0 of the PCI Expansion ROM Base Address register, and • Resetting bit 1 of the PCI Command register. These bits can be set again when the reset is complete.	Yes	1

Adapter Error Code Register

The adapter-error-code register can be used to communicate the adapter error code to the host when the POST2A tests fail and the firmware in the adapter cannot run correctly. The contents of this register are valid only after a command sent to the adapter is rejected with a SS_POST2A_FAIL (20h) error reported in the error register.

PCI address: Offset 48h from runtime base address

CFE address: Offset C8h

Initialized value: 0000 0000h

Figure 33. Adapter Error Code Register

Table 28. Adapter Error Code Register

Bits	Description	Writeable	Initial value
23 through 0	**Adapter Error Code**. This field provides details of the reason for the POST2A failure. The codes are described in “Adapter Error Logging Data” on page 226.	Yes	0
31 through 24	**Template**. This byte is 0Ah (SSA_HDW_ERROR) for failures detected by the POST2A tests.	Yes	0

IEXECIO Register

This register is used by the adapter to indicate the completion of an IExecute I/O command. This command does not cause a PCI interrupt on completion and the host should poll bit 31 of this register; this bit is set to 1b when the command has completed.

PCI address: Offset 4Ch from runtime base address

CFE Address: Offset CCh

Initialized value: 0000 0000h

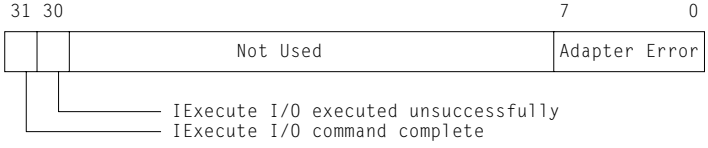


Figure 34. IEXECIO Register

Table 29. IEXECIO Register

Bits	Description	Writeable	Initial value
7 through 0	**Adapter Error.** This field identifies an error detected by the adapter when executing an IExecute I/O command. The contents are as defined for the Adapter Error register (see “Adapter Error Register” on page 52). The contents are valid only when bit 30 = 1b, indicating an unsuccessful completion of an IExecute I/O command.	Yes	0
29 through 8	Not used	Yes	0
30	**Unsuccessful IExecute I/O**. When an IExecute I/O command completes (bit 31 = 1b), this bit is set to 1b when the command completed unsuccessfully and bits 7 though 0 contain the error code. This bit is set to 0b if the IExecute I/O command completes successfully.	Yes	0

Table 29. IEXECIO Register (continued)

Bits	Description	Writeable	Initial value
31	**IExecute I/O command completion** When an IExecute I/O command completes, this bit is set to 1b by the adapter. The host should set this bit to 0b before issuing the IExecute I/O command and poll on this bit becoming 1b to indicate the completion of the commend.	Yes	0

Interrupt Control Register

This register is used to report to the host the state of the refuse interrupt flag. The register contains 00000001h when the refuse flag is active. The register cannot be used to set the refuse flag. The flag is set and reset by successive setting of the refuse bit in the Adapter Interrupt Doorbell register (see “Adapter Interrupt Doorbell Register”).

The contents of the register are valid only when the refuse bit in the Adapter Doorbell register has been cleared.

PCI address: Offset 50h from runtime base address

CFE Address: Offset D0h

Initialized value: 0000 0000h

Figure 35. Interrupt Control Register

Adapter Interrupt Doorbell Register

The adapter-interrupt-doorbell register can be used by the host to cause interrupts to the adapter.

PCI address: Offset 60h from runtime base address

CFE Address: Offset E0h

Initialized value: 0000 0000h

Figure 36. Adapter Interrupt Doorbell Register

Table 30. Adapter Interrupt Doorbell Register

Bits	Description	Writeable	Initial value
0	**New Host GTCB.** This bit is set to 1b from the PCI bus to cause an interrupt to the adapter to inform it that the host has set up a new host GTCB. The adapter can clear the register bit by writing a 1b to this bit from the local bus.	Yes	0
1	**Command Register Loaded.** This bit is set to 1b from the PCI bus to cause an interrupt to the adapter to inform it that the host has loaded a command into the adapter-command register. The adapter can clear the register bit by writing a 1b to this bit from the local bus.	Yes	0
2	**Host Slave Operation.** This bit is set to 1b from the PCI bus to cause an interrupt to the adapter to request a host-slave operation. The parameter register must have been set before this interrupt to identify where in host memory the host-slave-operation control block is located.	Yes	0
3	**Host Heartbeat.** This bit is set to 1b by the host to request that the adapter replies by setting the host-heartbeat-reply bit in the PCI-Interrupt-Doorbell register.	Yes	0
4	**Refuse.** This bit is set to 1b by the host to set the refuse flag. When refuse is set, the adapter does not generate any more adapter-to-host transactions. The refuse flag is reset the next time that the host sets this bit to 1b.	Yes	0
31 through 5	Not used	Yes	0

PCI Interrupt Doorbell Register

The PCI-interrupt-doorbell register can be used by the adapter to causes interrupts to the host.

PCI address: Offset 64h from runtime base address

CFE Address: Offset E4h

Initialized value: 0000 0000h

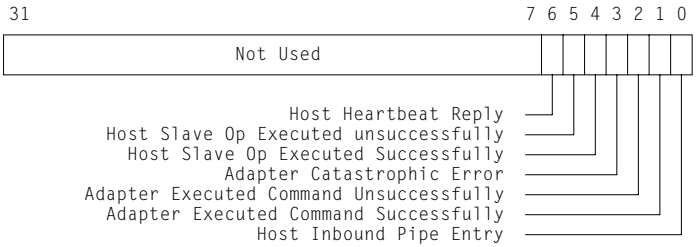


Figure 37. PCI Interrupt Doorbell Register

Table 31. PCI Interrupt Doorbell Register

Bits	Description	Writeable	Initial value
0	**Host Inbound Pipe Entry.** This bit is set to 1b from the local bus to cause an interrupt to the host to inform it that the adapter has added an entry to the host incoming pipe. The host can clear the register bit by writing a 1b to this bit.	Yes	0
1	**Adapter Executed Command Successfully.** This bit is set to 1b from the local bus to cause an interrupt to the host to inform it that the adapter has successfully executed a command. The host can clear the register bit by writing a 1b to this bit.	Yes	0
2	**Adapter Executed Command Unsuccessfully.** This bit is set to 1b from the local bus to cause an interrupt to the host to inform it that the adapter has executed a command unsuccessfully because of an error in the I/O device or an attachment error. The error code in the error register defines the error. The host can clear the register bit by writing a 1b to this bit.	Yes	0
3	**Adapter Catastrophic Error.** This bit is set to 1b from the local bus to cause an interrupt to the host to inform it that the adapter has detected a catastrophic error. The error code in the error register defines the error. The adapter presents this interrupt continuously until it is reset by the device driver. The host can clear the register bit by writing a 1b to this bit.	Yes	0
4	**Host Slave Op Executed Successfully.** This bit is set to 1b from the local bus to cause an interrupt to the host to inform it that the adapter has executed a host-slave operation successfully. The host can clear the register bit by writing a 1b to this bit.	Yes	0
5	**Host Slave Op Executed Unsuccessfully.** This bit is set to 1b from the local bus to cause an interrupt to the host to inform it that the adapter has executed a host-slave operation unsuccessfully. The error code in the error register defines the error. The host can clear the register bit by writing a 1b to this bit.	Yes	0

Table 31. PCI Interrupt Doorbell Register (continued)

Bits	Description	Writeable	Initial value
6	**Host Heartbeat Reply.** This bit is set to 1b from the local bus to cause an interrupt to the host to acknowledge that the adapter has received a host-heartbeat adapter interrupt.	Yes	0
31 through 7	Not used	Yes	0

Interrupt Control/Status Register

The interrupt control/status register controls interrupts and provides some status information.

PCI address: Offset 68h from runtime base address

CFE Address: Offset E8h

Initialized value: 000F 1F03h

Figure 38. Interrupt Control/Status Register

Table 32. Interrupt Control/Status Register

Bits	Description	Writeable	Initial value
0	**Enable Local Bus LSERR# for Aborts.** When this bit is set to 1b, the PCI 9060 is able to assert LSERR# interrupt output when the PCI-bus target-abort or master-abort status bit is set in the PCI Status-Configuration register.	Yes	1
1	**Enable Local Bus LSERR# for Parity Errors.** When this bit is set to 1b, the PCI 9060 is able to assert LSERR# interrupt output when a PCI parity error occurs during a local-master transfer or a local-slave access.	Yes	1
2	**Generate PCI Bus SERR#.** When this bit is 0b, setting it to 1b generates a PCI bus SERR#.	Yes	0
7 through 3	Not used	No	0
8	**PCI Interrupt Enable.** When this bit is set to 1b, PCI interrupts are enabled.	Yes	1
9	**PCI Doorbell Interrupt Enable.** When this bit is set to 1b, interrupts caused by the PCI-interrupt-doorbell register are enabled. This is used in conjunction with bit 8 (PCI interrupt enable). Clearing the doorbell interrupt bits causing the interrupt clears the interrupt.	Yes	1

Table 32. Interrupt Control/Status Register (continued)

Bits	Description	Writeable	Initial value
10	**PCI Abort Interrupt Enable.** When this bit is set to 1b, a master abort or a master detect of a target abort is enabled to generate a PCI interrupt. Clearing the abort status bits clears the interrupt.	Yes	1
11	**PCI Local Interrupt Enable.** When this bit is set to 1b, a local interrupt is able to generate a PCI interrupt. This is used in conjunction with PCI interrupt enable. Clearing the local bus cause of the interrupt clears the PCI interrupt.	Yes	1
12	**Retry Abort Enable.** When this bit is set to 1b, the PCI 9060 is enabled to treat 256 consecutive master retries to target as a target abort; when set to 0b, the PCI 9060 is allowed to attempt master retries indefinitely.	Yes	1
13	**PCI Doorbell Interrupt.** When this bit is set to 1b, it indicates that a PCI interrupt is active	No	0
14	**PCI Abort Interrupt.** When this bit is set to 1b, it indicates that a PCI abort interrupt is active.	No	0
15	**Adapter Doorbell Interrupt.** When this bit is set to 1b, it indicates that an adapter interrupt is active.	No	0
16	**Local Interrupt Enable.** When this bit is set to 1b, local adapter interrupts are enabled.	Yes	1
17	**Adapter Doorbell Interrupt Enable.** When this bit is set to 1b, interrupts caused by the adapter doorbell interrupt register are enabled. This is used in conjunction with the local-interrupt-enable bit. Clearing the adapter-doorbell-interrupt register bits that caused the interrupt clears the interrupt.	Yes	1
18	**Local DMA Channel 0 Interrupt Enable.** When this bit is set to 1b, DMA-channel-0 interrupts are enabled. This is used in conjunction with the local-interrupt-enable bit. Clearing the DMA status bits clears the interrupt.	Yes	1
19	**Local DMA Channel 1 Interrupt Enable.** When this bit is set to 1b, DMA-channel-1 interrupts are enabled. This is used in conjunction with the local-interrupt-enable bit. Clearing the DMA status bits clears the interrupt.	Yes	1
20	**Adapter Doorbell Interrupt.** When this bit is set to 1b, it indicates that an interrupt caused by the adapter doorbell interrupt register is active.	No	0
21	**DMA Channel 0 Interrupt.** When this bit is set to 1b, it indicates that a DMA-channel-0 interrupt is active.	No	0
22	**DMA Channel 1 Interrupt.** When this bit is set to 1b, it indicates that a DMA-channel-1 interrupt is active.	No	0

Table 32. Interrupt Control/Status Register (continued)

Bits	Description	Writeable	Initial value
23	**BIST Interrupt.** When this bit is set to 1b, it indicates that a BIST interrupt is active. The BIST interrupt is generated when a 1b is written to bit 6 of the PCI-configuration-BIST register. Clearing bit 6 clears the interrupt.	No	0
24	**Abort when Direct Master.** When this bit is set to 1b, it indicates that a direct master was the bus master during a master or target abort.	No	0
25	**Abort when DMA Channel 0 Master.** When this bit is set to 1b, it indicates that a DMA channel 0 was the bus master during a master or target abort.	No	0
26	**Abort when DMA Channel 1 Master.** When this bit is set to 1b, it indicates that a DMA channel 1 was the bus master during a master or target abort.	No	0
27	**Target Abort.** When this bit is set to 1b, it indicates that a target abort was generated by the PCI 9060 after 256 consecutive master retries to a target.	No	0
31 through 28	Not used	No	0h

PCI Command Codes, EEPROM Control, User I/O Control, INIT Control

This register identifies PCI commands used, and controls EEPROM and local INIT status. EEPROM for the PC19060 module is not used on the adapter as the configuration registers are loaded by the local processor from flash EPROM.

PCI address: Offset 6Ch from runtime base address

CFE Address: Offset ECh

Initialized value: 88037C7Ch

Figure 39. PCI Command Codes, EEPROM Control, User I/O Control, INIT Control

Table 33. PCI Command Codes, EEPROM Control, User I/O Control, INIT Control

Bits	Description	Writeable	Initial value
3 through 0	**PCI Read Command Code for DMA.** (See note 1.)	Yes	1100
7 through 4	**PCI Write Command Code for DMA.**	Yes	0111
11 through 8	**PCI Memory Read Command Code for Direct Master.** (See note 2.)	Yes	1100

Table 33. PCI Command Codes, EEPROM Control, User I/O Control, INIT Control (continued)

Bits	Description	Writeable	Initial value
15 through 12	**PCI Memory Write Command Code for Direct Master.**	Yes	0111
16	**General Purpose Output.** When this bit is set to 1b, the USERO output of the PCI 9060 goes high; when set to 0b, the USERO output goes low. USERO output is not used.	Yes	1
17	**General Purpose Input.** When this bit is set to 1b, the USERI input pin is high; when set to 0b, the USERI input pin is low. USERI pin is not used.	Yes	1
23 through 18	Not used	No	0
24	**EEPROM Clock**. Toggling this bit generates an EEPROM clock. EEPROM is not currently used on the adapter.	Yes	0
25	**EEPROM Chip Select.** Setting this bit to 1b provides the EEPROM chip select for local or PCI reads or write to EEPROM. EEPROM is not currently used on the adapter.	Yes	0
26	**Write EEPROM.** For writes, this output bit is the input to the EEPROM clocked by EEPROM Clock. EEPROM is not currently used on the adapter.	Yes	0
27	**Read EEPROM.** For reads, this input bit is the output from the EEPROM clocked by EEPROM Clock.	Yes	1
28	**EEPROM Present.** When this bit is set to 1b, it indicates that an EEPROM is present.	No	0
29	**Reload Configuration Registers.** When this bit is set to 0b, writing a 1b causes the PCI configuration registers are to be loaded from the EEPROM. EEPROM is not currently used on the adapter.	No	0
30	**PCI Adapter Software Reset.** When this bit is set to 1b, the PCI 9060 is reset and a reset is issued to the local bus causing a reset of the adapter. The contents of the PCI configuration registers are not changed by this reset. This bit can only be cleared from the PCI bus. The host must clear this bit before the local bus can be used for accesses.	Yes	0
31	**Local INIT Status.** This bit should be set to 1b when local initialization is complete. The adapter responds to PCI accesses with RETRYs until this bit is set. This bit is forced to 1b while NB low (not used on this adapter).	Yes	1

Notes:

1. The initial value is 1110 for the PCI SSA 4-Port RAID Adapter.
2. The initial value is 0110 for the PCI SSA 4-Port RAID Adapter.

Expansion ROM

PCI expansion ROM is loaded into SRAM from flash ROM every power on or reset. The PCI-expansion-ROM-base register identifies the PCI address the system uses to access the first word of expansion ROM. The size of expansion ROM is 4 KB and consists of a single ROM image.

The information in the expansion ROM image is split into two areas. One area, the ROM header, is located at the beginning of the ROM image. The second area, the PCI data structure, is located at an offset from the beginning identified by a pointer field in the ROM header. This data structure contains a code area (not used) and an area for vital product data (VPD).

Expansion ROM Header

The offset field is a hexadecimal number of bytes from the beginning of the image. The length field is in bytes.

Table 34. Expansion ROM Header Format

Offset	Length	Value	Description
0 through 1	2	55AAh	ROM signature
2	1		Initialization size
3 through 5	3		Entry point for INIT function
6 through 17	12h	xx	Reserved
18 through 19	2	xxxx	Pointer to PCI Data Structure

ROM signature
This two-byte field contains 55h in the first byte and AAh in the second byte.

Initialization size
This one-byte field identifies the size of the code in units of 512 bytes.

Entry point for INIT
POST does a FAR CALL to this location

Pointer to PCI data structure
This is a two-byte pointer in little-endian format that points to the PCI data structure. The reference point for this pointer is the beginning of the ROM image.

PCI Data Structure

This is 32-bit-word aligned. The offset field is a hexadecimal number of bytes from the beginning of the structure. The length field is in bytes.

Table 35. PCI Data Structure (expansion ROM) Format

Offset	Length	Value	Description
0 through 3	4	PCIR	Signature
4 through 5	2	1014h	Vendor identification
6 through 7	2	0044h	Device identification
8 through 9	2		Pointer to Vital Product Data
A through B	2		PCI data-structure length
C	1	00h	PCI data-structure revision
D through F	3	00020Ch	Class code
10 through 11	2	0008h	Image length
12 through 13	2	0	Revision level of code/data
14	1	0	Code type
15	1	80h	Indicator
16 through 17	2	0000h	Reserved

Signature

This four-byte field contains the string 'PCIR' with 'P' at offset 0 and 'R' at offset 3.

Vendor identification

This 16-bit field has the same definition as the vendor identification field in the configuration space. The value assigned is 1014h.

Device Identification

This 16-bit field has the same definition as the device identification field in the configuration space. The value assigned is 0044h.

Pointer to Vital Product Data

This 16-bit field is the offset in little-endian format from the start of the ROM image and points to the Vital Product Data (VPD). It consists of 135 bytes and is located after the last byte of executable code in the ROM image. The VPD fields supported are defined in "Vital Product Data" on page 25.

PCI Data Structure Length

This 16-bit field defines the length of the data structure from the start of the structure (the first byte of the signature field). The field is in little-endian format and is in units of bytes.

PCI data-structure revision

This 8-bit field identifies the data-structure revision level. The revision level is 0.

Class code

This 24-bit field has the same definition as the class code field in the configuration space for this adapter.

Image length

This 16-bit field defines the length of the image. This field is in little-endian format and the value is in units of 512 bytes.

Revision level
This 16-bit field defines the revision level of the code in the ROM image.

Code type
This 8-bit field identifies the type of code contained in this section of the ROM.

Indicator
Bit 7 of this field is 1b to indicate that this is the last image. Bits 6 through 0 are reserved.

Chapter 4. Adapter-to-Device Interface

Except where noted below the SSA ports conform fully with the SSA architectures described in “Supported Standards” on page 4.

The PCI SSA 4-Port RAID Adapter and PCI SSA Multi-Initiator/RAID EL Adapter cards each have 4 SSA ports which always operate as 2 dual-port nodes. The adapters do not operate as SSA switches or as single-port nodes.

Each dual-port node is an initiator with its own SSA unique ID. These differ only in the low order bit.

The initiators always use the shortest available path to the addressed device.

Upper Level Protocol

The upper level protocol returned by a PCI SSA Multi-Initiator/RAID EL Adapter in the query-node-reply message is FCh (Vendor Unique).

The protocol—list field in query_protocol_reply contains the following information:

Byte	Contents	
0	FCh	
1 through 3	000629h	(Bytes 1 through 3 are part of the unique ID number identifying IBM)
4	01h	Adapters with code before level 50 that support up to 2 adapters in a loop
	02h	Adapters with code at or later than level 50 that support up to 8 adapters in a loop

Data Transfers

Each SSA port can sustain 10 concurrent data transfers.

The adapter hardware can only perform SSA data transfers that contain an even number of data bytes. This imposes the following requirements on the target:

- If the target needs to return an odd number of data bytes (for example, for an Inquiry command), it must append a pad byte and the byte count in the previous data-ready message must be an even number.
- The byte count in data-request messages must be an even number.

Master Functions

Either or both of the initiators on the adapter card can function as an SSA master. When an SSA loop contains more than one adapter, the adapter with the highest unique ID, of the highest master-priority field returned in a reply to a query-node message, is the master. (All of the unique IDs are already known to each initiator in the SSA configuration table, which is built by walking the network with query-node messages.)

Port Configuration

When it is operating as a master the adapter issues a configure-port message to each port in the network:

- The port is allocated a tag, port, and return path for use by subsequent async-alert messages.
- The A and B SAT quotas are configured according to the guidelines in the SSA standard.

  Note: The adapter does not support multiple SAT regions. All ports are configured to propagate SAT tokens.
- The routing of user-defined characters through the master node in a loop is blocked to avoid continuous circulation.

  Note: The adapter does not originate or use spindle-sync characters.

Asynchronous Alerts

When it is operating as a master, the adapter is responsible for:

- Propagating an asynchronous alert by sending a Master_alert message to each initiator
- Performing a third-party quiesce on behalf of a missing initiator
- Returning the affected ports to normal mode after a transient unrecoverable error.

Configurations

The SSA adapter supports string and loop networks only. The following restrictions are imposed to ensure good performance:

- To be fault-tolerant and allow concurrent maintenance, all networks should be installed as loops rather than strings.
- No more than 48 devices can be connected in the same loop.
- Both the initiators of an adapter must not be connected in the same SSA loop.
- No more than 3 SSA adapters should be plugged into the same PCI bus. Generally, if the using system has multiple PCI buses, maximum performance of multiple adapters is achieved if the adapters are plugged into separate PCI buses.
- All member disks of an array configured on a PCI SSA Multi-Initiator/RAID EL Adapter must be in the same SSA loop. The member disks of an array configured on a PCI SSA 4-Port RAID Adapter can be in any loop.
- A single SSA loop can contain only one adapter if:
  1. The adapter is a PCI SSA 4-Port RAID Adapter, or
  2. The adapter is a PCI SSA Multi-Initiator/RAID EL Adapter and any disk or RAID array is configured for Fast-write active.
  3. The adapter is a PCI SSA Multi-Initiator/RAID EL Adapter with code before level 50 and any disk is configured as a member of a RAID/5 array.
- A single SSA loop can contain up to two adapters if:
  1. Either adapter is a PCI or MicroChannel SSA Multi-Initiator/RAID EL adapter with code before level 50 and no links are configured as members of a RAID array or for fast-write active, or
  2. Both adapters are PCI or MicroChannel SSA Multi-Initiator/RAID EL adapters with code at or later than level 50 and whose disks are configured as disks or members of RAID-5 arrays with no disks or RAID-5 arrays configured for fast-write active.
- A single SSA loop can contain up to eight adapters if the adapters are PCI or MicroChannel SSA Multi-Initiator/RAID EL adapters with code at or later than level 50 and no disks are configured as members of RAID-5 arrays or with fast-write active.

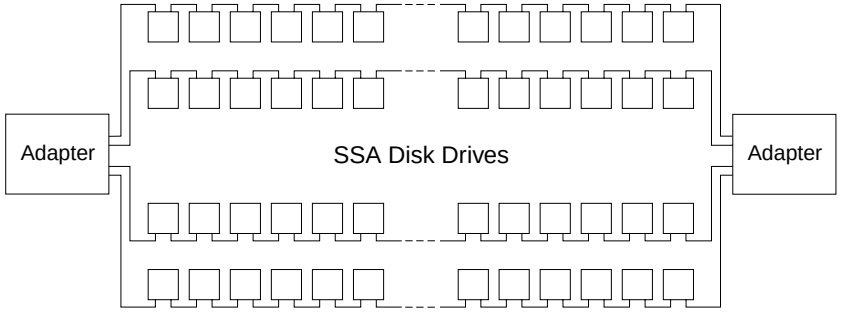


Figure 40. An Example of Two Adapters Sharing Two SSA Loops

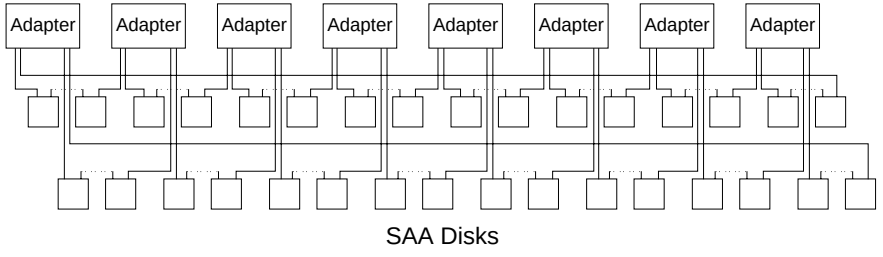


Figure 41. An Example of Eight Adapters Sharing Two SSA Loops

Adapter Card

The PCI SSA 4-Port RAID Adapter and PCI SSA Multi-Initiator/RAID EL Adapter are PCI 5V standard size cards. The cards measure 312 mm long by 106 mm high, excluding the PCI connector. On the front panel of the adapter and on one of its modules are labels on which is printed the 15-character SSA unique ID of the adapter.

The SSA Fast-Write Cache Option Card is a separate card with a PCMCIA connector that can be plugged in to a PCI SSA Multi-Initiator/RAID EL Adapter. The SSA Fast-Write Cache Option Card has 4MB of nonvolatile memory and supporting logic. A SSA Fast-Write Cache Option Card can be removed from a failed adapter and installed on a replacement adapter.

Figure 42 on page 73 shows a PCI SSA 4-Port RAID Adapter card. Figure 43 on page 74 shows a PCI SSA Multi-Initiator/RAID EL Adapter card with an SSA Fast-Write Cache Option Card installed on it.

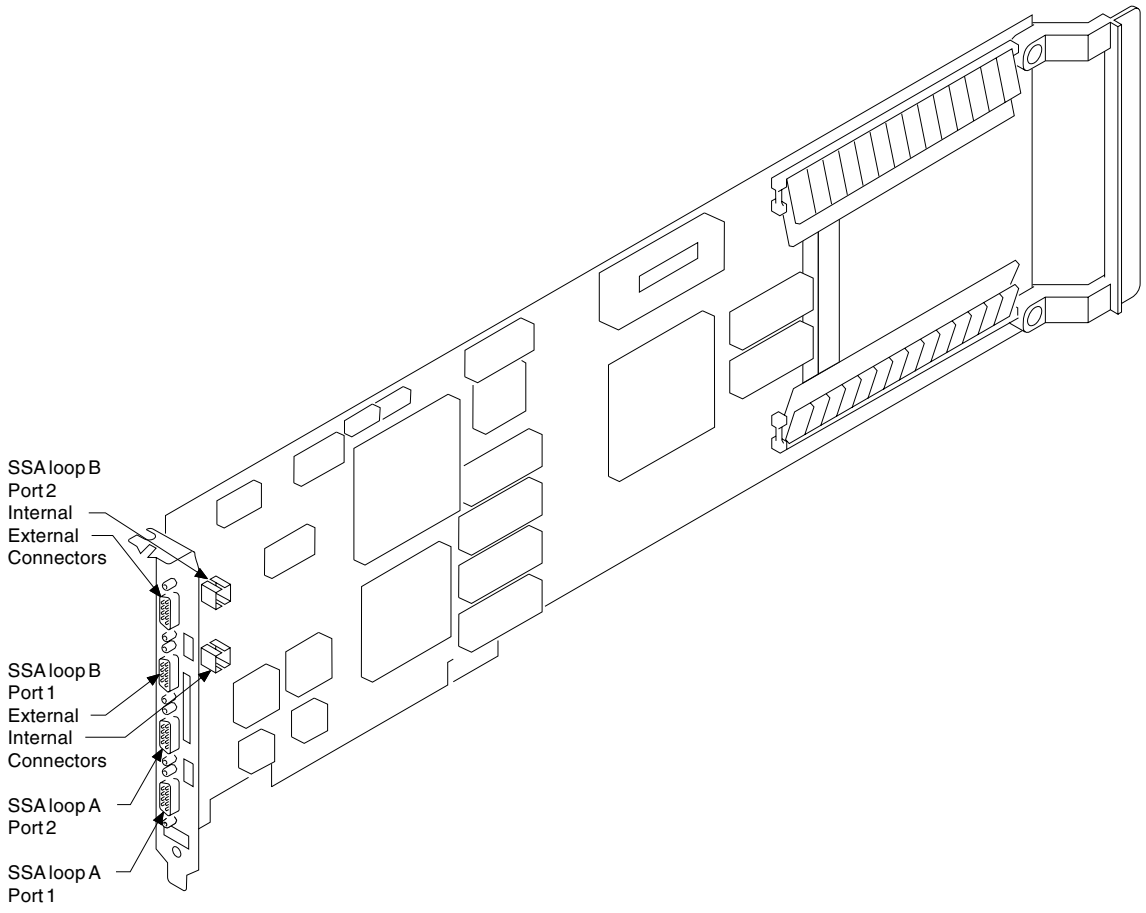


Figure 42. PCI SSA 4-Port RAID Adapter Card Layout

SSA Connectors

The adapter card has 2 internal SSA connectors and 4 external connectors. This allows one of the 2 dual-port SSA nodes to be connected either internally or externally to the system unit. The internal connectors are 2 x 3 pin SSA connectors.

The ports are clearly numbered ‘A1’, ‘A2’, ‘B1’, and ‘B2’ at the connectors. Ports B1 and B2 have both internal and external connectors. The marking also indicates that ports A1 and A2 are paired, that is, they are connected to the same SSA loop interface chip. Similarly, ports B1 and B2 are paired.

+5 V power is available on the connector to power an external optical extender.

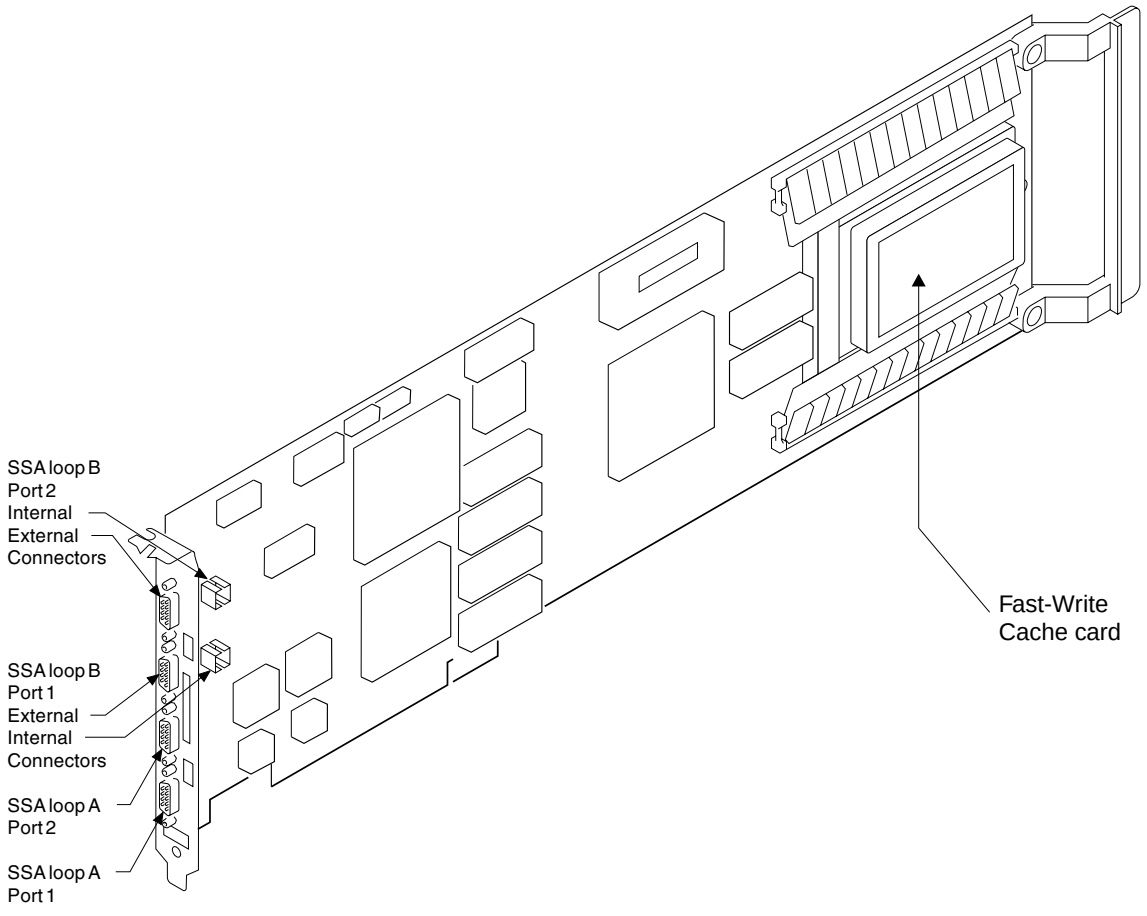


Figure 43. PCI SSA Multi-Initiator/RAID EL Adapter Card Layout

Indicators

A light is provided for each SSA loop interface to assist in service for the SSA network. When one port of a SSA loop interface is not operational, the light for that SSA loop interface flashes continuously at approximately once every 2 seconds.

In addition there is a light mounted internally on the card rather than at the side with the connectors that flickers when there is activity on the adapter, that is, when it is processing I/O requests. If it is not flickering it means that either the system is not requesting any I/O operations or that the adapter has failed. This light may not be clearly visible on all systems.

DRAM Buffer

Two equal-capacity SIMMs must be installed. Each must be a 36-bit 72-pin 70-ns fast-page SIMM. SIMMs with a storage capacity of 4MB are supplied on the PCI SSA

4-Port RAID Adapter, giving a total DRAM capacity of 8MB. SIMMs with a storage capacity of 16MB are supplied on the PCI SSA Multi-Initiator/RAID EL Adapter, giving a total DRAM capacity of 32MB.

Power Requirements

Voltage
5.0 +5%, −4.5%

Current
3.09 A maximum (2.0 A typical)

Power 15.4 W maximum

Ripple 100 mV peak-to-peak maximum, dc to 50 KHz

Environment

Operating

Temperature
10 to 40°C

Humidity
8 to 80 %, noncondensing

Altitude
0 to 7,000 feet

Cooling
Natural convection

EMC FCC class A and CISPR 22 class A when packaged in a system unit

Nonoperating

Temperature
−40 to 60°C

Humidity
5 to 80 %, noncondensing

Altitude
−1,000 to 40,000 feet

SSA Cables

The external cable comply with the electrical characteristics defined in the *Serial Storage Architecture - 1995 Physical* document. Cable of 28 AWG gauge can be used for lengths up to 20 meters. For a length of 25 meters, a 26 AWG gauge cable should be used.

Fibre-Optic Extender

A fibre-optic extender is available to connect an industry-standard fiber optic cable within an SSA loop when that loop is used with a 7133 SSA Disk Subsystem. With a fibre-optic extender attached at each node, the distance between them can be up to 2.4 km. The maximum sustainable data-transfer rate in a single direction on an SSA link is approximately 18 MB/s. Extending the link with a fiber optic cable results in no degradation of this data-transfer rate for lengths up to 200 meters, and a gradual reduction in the achievable data-transfer rate beyond this distance. This data-transfer rate reduction generally has no effect on SSA subsystem performance for most applications.

Chapter 5. Array and Fast Write Filters

The PCI SSA 4-Port RAID Adapter and PCI SSA Multi-Initiator/RAID EL Adapter provide RAID functions by means of a filter between the device driver and the disk drive. The filter is implemented in microcode that runs in the adapter. The filter presents the image of a single disk drive to the device driver, and uses one or more components to implement this image. IPN transactions are provided to configure the image. The components are disk drives that are attached to this adapter (on either SSA loop). All, some, or none of the disk drives attached to an adapter can be components of the filter. There can be up to 16 components in an image. On a PCI SSA 4-Port RAID Adapter, the member disk drives of an array can be in either loop; on a PCI SSA Multi-Initiator/RAID EL Adapter, all the member disk drives of an array must be in the same loop.This chapter describes the number of components allowed, and the mapping from the image to the components, for the filter that is supplied in the adapter.

If the components of an array have different sizes, only the capacity of the smallest component is used for each component.

The adapter can support up to 32 arrays.

Disk Drives not in Arrays

The adapters support the use of attached disk drives without any filter. Data is transferred directly from the system interface to the SSA link without passing through a buffer.

The adapter overhead for nonarray operations is typically less than 125 µs and the microprocessor utilization is less than 330 µs. The adapter can execute over 3000 short read/write operations per second , depending on the attached devices and the capability of the system.

RAID-5

The RAID-5 filter combines N+1 components of equal capacity, C. A few kilobytes K are reserved for use by the filter; the rest are used for data and parity storage. In general, the first S blocks (a strip) of the image are mapped to the first S blocks of one component, and the second S blocks to the first S blocks of the next component, and

so on across N components. The exclusive-OR of these strips (parity) is written on the first S blocks of another component of the N+1 array. In the SSA 4-Port RAID Adapter, the parity of the first strips is held on component 1, the first data strip is on component 2, the second data strip is on component 3, and so on to the Nth data strip which is on component N+1. The parity of the next N strips is on component 1, the (N+1)th data strip is on component 2, the (N+2)th data strip is on component 3, and so on.

This pattern is repeated for the first 4xN data strips. For the next 4xN data strips, the parity is on the second component and the data strips start on the third component. This rotation of the parity component continues every 4xN data strips.

Read operations from the host are mapped into read operations to the components holding the required data. For short operations, this is usually one component; for longer transfers, all components might be used.

Write operations are handled in one of two ways:

- If aligned NxS blocks are to be written, data is written to N components, and the parity calculated and written to the remaining component.
- For other write operations, the old data is read from one component and the corresponding old parity from another. Then the new parity is calculated by exclusive-OR of old data, new data, and old parity, and the new data and new parity are written to the appropriate components. That is, a single host write operation becomes a read operation and a write operation to each of two components.

Writing using the first of these methods gives better performance because it requires fewer disk accesses.

A write operation is reported complete when the new data is written to the data disk. The new parity may be written to disk later. To protect against power failure before this is done, a note of the outstanding parity update is made in the nonvolatile memory.

Data transfer for RAID-5 operations takes place through the data buffer; data in the data buffer is used as a read cache to satisfy later read operations. This is particularly useful for RAID-5, because even write operations to a RAID-5 array result in read operations to physical disk drive components.

The microprocessor utilization for RAID-5 operations is less than 1 ms for a mixture of 30% write and 70% read, so that the adapter can perform up to 1000 RAID-5 operations per second.

The number of components N+1 must be in the range 3 to 16. The size (S) of each strip is 64KB. If the component capacity C-K is not a multiple of S, then the excess space is not used; the capacity of the image is `((C-K) mod S)xSxN`.

A resource-dependent-value attribute (PageAlignedSplits) is defined for RAID-5 that permits data to be written out of order even when the FF_Split bit is off. This can be set at configuration time.

If a write operation, whose data lies wholly within an aligned 4K page, is interrupted and does not complete, the resulting data recorded (as retrieved by a subsequent read operation) is one of:

- All the old data
- All the new data
- A single transition from new data to old data at some sector boundary within the data.

(An *aligned 4K page* is the first 4KB section of the logical array address space (starting at address 0) and each subsequent contiguous 4KB section of its address space.)

An operation that straddles more than one aligned 4K page is regarded as having been broken into aligned-4K-page operations and each is treated independently by the above rules. There is no guarantee of the order of the 4K pages to which the data is written.

When the PageAlignedSplits attribute is on, write operations take place in any order. The FF_Split value of the writes to each component matches that on the write transaction to the array; so, if split is turned off, data is guaranteed to be written in ascending LBA order on each component. When the PageAlignedSplits attribute is off and FF_Split is off, writes take place in ascending LBA order for the array.

Hot spares

Arrays can be configured with hot-spare disk drives. If an array has a hot spare disk drive available when a component fails, the hot spare is automatically used to replace the failed component. On a PCI SSA 4-Port RAID Adapter, a single hot spare can be used to replace any failed disk drive that is a member of any array; on a PCI SSA Multi-Initiator/RAID EL Adapter, a hot spare is required on each SSA loop in which there are array members.

When a disk drive fails and is missing from an SSA loop, it is replaced by a hot spare when a write transaction is received or, if the array has no incorrect data, when a read transaction is received.

It is recommended that hot spares are available. A write operation to an array that has a member missing causes that array to enter the degraded state. Unless the array is operating in the read-only-while-exposed mode, if an array is in the degraded state and if a write operation to the parity disk has not been completed and if there is a loss of power, data for the unwritten blocks cannot be recreated when the missing disk is replaced.

Fast Write

The fast write filter adds fast-write caching capability for individual disk drives and arrays. A write operation to a fast write filter results initially in the data being written to both the non-volatile fast write cache and the volatile data buffer on the adapter; after which, status is sent to indicate that the operation is complete. At some later time, the data is written to the underlying disk drive or array; after which, the data in the

non-volatile fast write cache is discarded. If a second write operation is addressed to the same location, the later one might replace the earlier in the buffer before it has been written to the underlying component. If multiple writes are done to adjacent locations, they may be combined into a single write to the disk. The service time is much shorter because completion is signalled as soon as the data is in the buffer.

If power fails before the data is written to disk, the data is preserved in the fast write cache. When power is restored the adapter, it writes the data to disk. If the adapter fails, any data not yet written to disk is preserved in the fast write cache, which can be removed and fitted to the replacement adapter card. When this new adapter is powered up, it writes the data to disk.

The fast write cache size is 4 MB. An LRC is generated for each page in the cache for integrity checking after a loss of power.

Any array or individaully-accessed disk can be configured for fast write. For these arrays or disks all transactions from the host are sent to the fast write filter. The execution of the transactions is as follows:

Write Operations

- If the length of data is less than a defined value (which can be set by the user), the data is saved in the cache and completion is signaled before data is written to disk. Also, the user can specify the range of logical block addresses that are to be candidates for saving in the cache.
- If the length of data is greater than a defined value (which can be set by the user), the transaction is passed to the array or individually-accessed disk and data is written to disk before completion is signaled.

Read Operations

- If all the requested data is held in the cache, the fast write filter sends it from the volatile data buffer to the host without involving the RAID or disk services.
- If none of the requested data is held in the cache, the read transaction is forwarded to the RAID filter or disk service for execution.
- If part of the requested data is held in the cache, this data is destaged to the disks before the read transaction is forwarded to the RAID filter or disk service for execution.

Data is destaged from the non-volatile data buffer to the disks at the following times:

- When the resource is closed.
- When a FN_ISAL_Flush transaction is executed for the resource.
- When the cache contains a set percentage of its maximum capacity. Data is not destaged immediately to benefit from the possibility of merging writes to disks.
- When data has been in the cache for a set duration (a few minutes).
- When a contiguous block of data has been saved in the cache. The amount of contiguous data for the RAID filter is a stripe (Nxstrip).

Array States

An array can be in one of the following states:

Online-Good
The array is online and it can be read and written. All the array components are present. All parity data (except that affected by recently completed write operations) is synchronized. No data or parity rebuilding is outstanding. The array is fully protected against the loss of one component.

Online-Exposed
One component is missing from the array. When the array is read, data can be reconstructed for the missing component. The first write operation causes the array to enter the Online-Degraded state, unless there is no hot spare available that can be used to replace the missing component.

In the Online-Exposed state, the missing component can be reintroduced or replaced. Then, after any necessary rebuilding, the array is returned to the Online-Good state.

Online-Degraded
One component is missing and a write operation has been received for the array. Read and write operations to the array are supported. However, if power is lost before all the parity data has been written, it might not be possible to recreate all the data for the missing component.

The missing component is permanently excluded from the array.

Online-Rebuilding
The array is online and it can be read and written. The full complement of array components are present but data and parity are being rebuilt on one of the components.

Offline
More than one component of the array is missing or has failed. Read and write operations are not supported.

Unknown
The array is initially in this state until N-1 components are visible to the filter for the first time.

The movement between states is illustrated in Figure 44 on page 82.

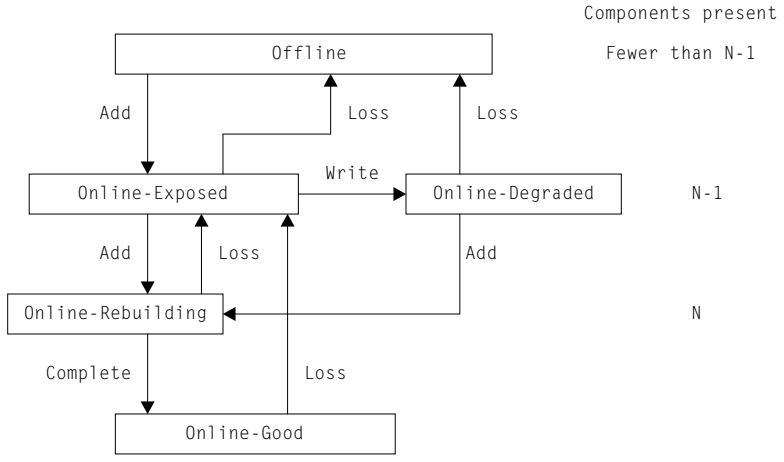


Figure 44. Array State Transitions

Array Configuration

An array-configuration utility is provided to create, delete, and change an array. Essential array information is maintained in the reserved area of the component disks of each array. This includes:

- Array serial number
- Component serial number
- Resource dependent values
- Bit maps (for bad parity)
- Out-of-sync flag

All the essential information is on the disks. The bit maps may be supplemented by information in the NVRAM. The information is stored in such a way that:

- If any disk is removed, it is still possible to operate the array
- If more than one disk is removed, it is possible to identify the serial number of the array but, possibly, no more.
- If any update to the information is interrupted or fails for any reason, it is still possible to determine the state of the array.

Multi-way RAID

On PCI SSA 4-Port RAID adapters and PCI SSA Multi-Initiator RAID/EL adapters with code prior to level 50, arrays are only available to a single host system. The PCI SSA Multi-Initiator RAID/EL adapter with code from level 50 allows 2 host systems to share the arrays. These hosts may also shared devices that are not configured in arrays.

Each system can execute to shared arrays all the functions currently provided for unshared arrays plus additional locking and fencing functions to allow systems to control access to the arrays.

No single failure of any component in the system should cause an array to be inaccessible to all healthy systems.

Member disks of an array must all be on the same SSA loop. All adapters on the same loop as a PCI SSA Multi-Initiator/EL adapter with code from level 50 where any disk is configured for RAID, must have a SSA upper level protocol = FCh and a Query_protocol list = 02h. This identifies either a PCI or Micro Channel Multi-Initiator/EL adapter that supports RAID from two adapters. Up to two adapters can control the arrays that exist on two SSA loops between both adapters.

The two adapters act as peers for operations to arrays that have been configured from disks on the SSA loops. Each can read and write directly to the disks, but need to request locks to the other adapter to ensure their operations do not conflict. Some operations do not require a lock from the partner adapter before execution, for example a read if the partner has not locked the area of LBAs. All write operations require a lock with the partner adapter. A synchronising agent controls the transfer and acknowledgement of locks. A write operation involves requesting a lock for the required strips from the partner adapter. The partner grants the lock and remembers the strip that may now contain out of date parity information in its NVRAM. If the first adapter fails before the write completes, the partner has the information in its NVRAM of strips with out of date parity and is able to regenerate the parity after the adapter failure.

By exchanging locks during operations and by keeping synchronising information on each array, whenever one of the partner adapter fails, the other can continue to operate and there is no impact to any of the arrays

If there is only one adapter in the network, the array filters continue to request locks through the synchronising agent which immediately grants all lock requests on behalf of the absent adapter.

Fast Write Cache is supported provided there is only one adapter in the SSA loop.

Chapter 6. IPN Transactions

Introduction

The SSA 4-Port RAID Adapter provides a registry service, a disk service, and an array-configuration service and an adapter service. Transactions are transmitted across the PCI interface to these services in a Gateway Transaction Control Block (GTCB). The format of the GTCB is defined in “Gateway Transaction Control Block (GTCB)” on page 9.

Services are at the heart of the IPN architecture. They form the server side of the client-server model. All communication to and from a service uses IPN transactions. Each server can be said to exist on a node and have its own unique service number. The combination of the node and service number form the network address of the service.

Generally services are used to gain access to a resource, whose size and importance can vary greatly.

Every service has a service language that describes the way that the communication to that service must be performed. IPN Storage Access Language (ISAL) is the language used by the disk service. IPN Array Configuration Language (IACL) is the language used by the array-configuration service.

When a service is installed into an IPN kernel the type of the service must be declared. This effectively declares what type of language the service understands. The service type is a one-byte code and can be one of the following:

TP_ISAL
A disk or other resource that acts like one (see “Disk Service” on page 117)

TP_Registry
A local information server (see “Registry Service” on page 89)

TP_CfgAgent
An array configurator (see “Array-Configuration Service” on page 176).

TP_AdapterService
An adapter service (see “Adapter Service” on page 156).

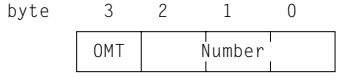
TP_ErrorLogger
A service in the device driver that receives error logs.

Device Addressing

Logical disks are identified by a resource ID. The host uses this resource ID to open the resource. During the process of opening the resource, a handle is returned for the resource. The host uses this handle when sending transactions to the resource.

Resource ID

The resource ID is an identifier that is passed to the resource manager to identify which logical disk the caller is referring to. The resource ID has the following structure:
Byte 3 is the owning-module-type field. This is a number that identifies the logical owner



of the resource. An SSA disk might be logically owned by the host disk driver; or, if it is part of a disk array, it might be owned by the RAID-5 manager.

The following values are used:

```
OMT = 1 - Not Owned by anyone
      2 - Device Driver Physical Adapters
      3 - Device Driver Physical Targets                        (pdisks)
      4 - Device Driver manually configured logical disks       (hdisks)
      5 - Device Driver automatically configured logical disks (hdisks)
     'K'- RAID-5
     'W'- Disowned
     'X'- Nvram entry
     'Y'- Hot spare disk
```

The lower 24 bits of the resource ID is a number that is used to identify which resource is being used. This number is set automatically by the resource manager; asking the registry for a temporary resource ID (using the FN_REGY_GetTempResrcID transaction) provides a unique 24-bit number for this field.

ISAL Reserved Area

ISAL disk resources maintain a reserved area of 512 byte blocks. The number of blocks available is reported in the FN_ISALMgr_Characteristics transaction. The SSA Disk ISAL manager internally has 32 blocks mirrored of which 30 are available to the user. The blocks are normally mirrored on a disk so that 64 sectors are required. The normal

ISAL interface (FN_ISAL_Read/Write) is used to read and write this area. A flag specifies that the I/O should be directed to the reserved area. There are a number of restrictions that apply to data in this area which are:

- I/O operations can only be one block in length.
- The first 16 bytes of all blocks are reserved, so each block must have the following format:
  The signature is a unique 8-byte field that is used to identify the sector as containing

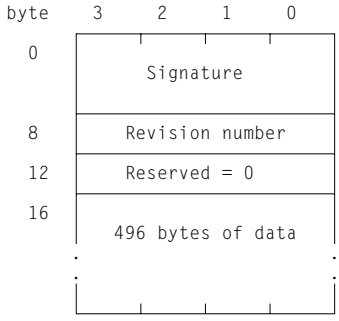


Figure 45. ISAL Reserved Area Sector Format

valid reserved area data (all the 32 sectors share the same signature). The signature field is an ASCII string 'ISALSIGN'. The revision number is used when reading the reserved data. The resource manager should read both mirrored copies and return the sector that contains the highest revision number (normally both are the same).

When writing a sector in the reserved area, the call sets up the first 16 bytes according to the rules here, and it is recommended that the new revision number is higher than the old value.

For the SSA 4-Port RAID Adapter, the reserved area starts 128 sectors from the end of the disk. The first sector of the reserved data (sector 0) is reserved as the device label record. Sectors 2 to 31 appear as ISAL reserved blocks 0 to 29.

Label Record

The label record is where the owning-module type (OMT) is recorded. The label record contains an array of disk numbers, one for each cluster. A cluster is a number assigned to each group of IPN nodes: this number allows a particular disk to be known by different disk numbers in different systems. The default cluster number is zero. The label record is kept in the ISAL reserved area but is not accessible by a Read or Write operation. It is only written when an OMT other than OM_DriverAutomaticDisk is set.

Registry Service

The function of the registry service is to maintain a database of IPN information. Each node runs a copy of the registry service. The registry service has a fixed service number (0000 0001h).

The registry service keeps a list of all of the services running on its node, and also a list of all the other nodes that can be accessed through a gateway from its node. Using these two lists, it is possible to walk the whole IPN network and discover what services are available.

In addition, the registry service performs a number of asynchronous notification services, such as error logging. The error logging process registers itself with all the registries. When a module detects an error, it reports this to its local registry service. The registry service sees that the error is sent to the error logger. This approach avoids the error logger having to register itself with every module that is capable of logging an error.

The registry service supports the following application transactions:

Table 36. Registry Transactions

Transaction	Minor_function
FN_REGY_SystemVersionInfo	10
FN_REGY_GatewayNodeList	11
FN_REGY_Servicelist	13
FN_REGY_ConnectForNodeChange	14
FN_REGY_DiscForNodeChange	15
FN_REGY_NodeChangeToRegistry	16
FN_REGY_NodeChangeFromRegistry	17
FN_REGY_ConnectForErrorLogging	18
FN_REGY_DiscForErrorLogging	19
FN_REGY_LogErrorTo Registry	20
FN_REGY_LogErrorFromRegistry	21
FN_REGY_ConnectForResrcChange	22
FN_REGY_DiscForResrcChange	23
FN_REGY_ResrcChangeToRegistry	24
FN_REGY_ResrcChangeFromRegistry	25
FN_REGY_ResrcList	26
FN_REGY_GetTempResrcID	27
FN_REGY_ConnectForHealthCheck	28
FN_REGY_DiscForHealthCheck	29
FN_REGY_HealthCheckToRegistry	30
FN_REGY_HealthCheckFromRegistry	31
FN_REGY_SerialNumberSearch	32
FN_REGY_TestResrcsReady	33
FN_REGY_SetClusterNumber	34
FN_REGY_TestOneResrcReady	35
FN_REGY_SyncHCheckToRegy	36

Table 36. Registry Transactions (continued)

Transaction	Minor_function
FN_REGY_SyncHCheckFromRegy	37

FN_REGY_SystemVersionInfo

This transaction can be sent to a registry service to obtain its code level.

Minor_function
10

Parameter_DDR
Null

Transmit_DDR
Null

Receive_DDR
Null

Status_DDR
This is a pointer to the buffer allocated to receive the following data:

Byte	3	2	1	0
0	Version			

Version
This field contains a 32-bit unsigned integer that identifies the current level of the registry code.

Result The following result fields can be returned:
AS_Success

FN_REGY_GatewayNodeList

This transaction returns the numbers of all the IPN nodes that might be known to the system. Further investigation is required to determine if a node is currently attached. The adapter registry services return a list of all nodes that could be connected for this configuration.

Minor_function
11

Parameter_DDR
Null

Transmit_DDR
Null

Receive_DDR
This is a pointer to a buffer which will receive the following data:

Byte	3	2	1	0
0	Node			
4	Node			
.				
n	Node			

Node This field contains the number of an IPN node that might be attached.

Status_DDR
This is a pointer to a buffer that receives the following data:

Byte	3	2	1	0
0	Count			

Count This field contains the number of entries in the received data DDR.

Result The following result fields can be returned:
AS_Success
Illegal Request (range)

FN_REGY_ServiceList

This transaction returns the numbers of the Services that are running on the same node as the registry service.

Minor_function
13

Parameter_DDR
This is a pointer to the following data:

Byte	3	2	1	0
0	Reserved = 0			Type

Type This identifies the type of services that should be reported. “Service / Transaction Directives” on page 238 gives the full list of types supported; these include:
TP_ISAL
Disk service (or something that acts like one)
TP_Registry
Local information server
TP_AdapterService
Adapter service
TP_CfgAgent
Array-configuration service

Transmit_DDR
Null

Receive_DDR
This is a pointer to a buffer that receives the following data:

Byte	3	2	1	0
0	Service			
4	Service			
.				
n	Service			

Service
This identifies the services of the requested type on this node.

Status_DDR
This is a pointer to a buffer that receives the following data:

Byte	3	2	1	0
0	Count			

Count This field contains the number of entries in the received data DDR.

Result The following result fields can be returned:
AS_Success
Illegal Request (range)

FN_REGY_ConnectForNodeChange

This transaction registers the caller as being interested in node-change asynchronous alerts.

Minor_function
14

Parameter_DDR
This is a pointer to the following data:

Byte	3	2	1	0
0	Node			
4	Service			
8	Reserved= 0			Synchro

Node This identifies the IPN node to which node-change asynchronous alerts should be reported.

Service
This identifies the service of the IPN node to which node-change asynchronous alerts should be reported.

Synchro
When the synchro field is SR_Synchro, the registry service sends node-change asynchronous alerts for all nodes known to the registry service before this transaction completes. When the synchro field is SR_NoSynchro, node-change asynchronous alerts are only sent for nodes that register after the transaction.

Transmit_DDR
Null

Receive_DDR
Null

Status_DDR
Null

Result The following result fields can be returned:
AS_Success
Illegal Request (range)
AE_TableFull

FN_REGY_DiscForNodeChange

This transaction registers the caller as being no longer interested in node-change asynchronous alerts.

Minor_function
15

Parameter_DDR
This is a pointer to the following data:

Byte	3	2	1	0
0	Node			
4	Service			
8	Reserved = 0			Synchro

Node This identifies the IPN node to which node-change asynchronous alerts had been reported.

Service
This identifies the service of the IPN node to which node-change asynchronous alerts had been reported.

Synchro
When the synchro field is SR_Synchro, a node-change async with event type EV_NodeDead is reported for each known node. When the synchro field is SR_NoSynchro, no node-change asynchronous alerts are sent as a result of this transaction.

Transmit_DDR
Null

Receive_DDR
Null

Status_DDR
Null

Result The following result fields can be returned:
AS_Success
Illegal Request (range)

AE_NotInTable

FN_REGY_NodeChangeToRegistry

This transaction tells the registry service that the status of a node has changed. This is an internal transaction within the adapter.

Minor_function
16

Parameter_DDR
This is a pointer to the following data:

Byte	3	2	1	0
0	Node			
4	Event			
8	Reserved = 0			

Node This identifies the IPN node whose status has changed.

Event This identifies the event, which can be:
EV_NodeDead
Node has stopped working
EV_Rebooted
Node has completed its IPL

Transmit_DDR
Null

Receive_DDR
Null

Status_DDR
Null

Result The following result fields can be returned:
AS_Success
Illegal Request (range)

FN_REGY_NodeChangeFromRegistry

This async transaction is passed on to all the modules that have connected for node-change asynchronous alerts.

To ensure that deadlock does not occur in the registry service, the receiver of this transaction should complete this transaction before issuing another transaction to the registry service.

Minor_function
17

Parameter_DDR
This is a pointer to the following data:

Byte	3	2	1	0
0	Node			
4	Event			
8	Reserved = 0			Synchro

Node This identifies the IPN node whose status has changed.

Event This identifies the event, which can be:
EV_NodeDead
Node has stopped working
EV_Rebooted
Node has completed its IPL

Synchro
The synchro field is SR_Synchro if the transaction is sent as a result of a FN_REGY_ConnectForNodeChange or FN_REGY_DiscForNodeChange transaction in which the synchro field was SR_Synchro. Otherwise, the synchro field is SR_NoSynchro.

Transmit_DDR
Null

Receive_DDR
Null

Status_DDR
Null

Result The following result fields can be returned:
AS_Success
Illegal Request (range)

FN_REGY_ConnectForErrorLogging

This transaction tells the registry service the node and service number of the error logger.

Minor_function
18

Parameter_DDR
This is a pointer to the following data:

Byte	3	2	1	0
0	Node			
4	Service			

Node This identifies the IPN node to which error logs should be sent.

Service
This identifies the service of the IPN node to which error logs should be sent.

Transmit_DDR
Null

Receive_DDR
Null

Status_DDR
Null

Result The following result fields can be returned:
AS_Success
Illegal Request (range)
AE_TableFull

FN_REGY_DiscForErrorLogging

This transaction tells the registry service that the error logger is no longer interested in receiving error logging records.

Minor_function
19

Parameter_DDR
This is a pointer to the following data:

Byte	3	2	1	0
0	Node			
4	Service			

Node This identifies the IPN node to which error logs had previously been sent.

Service
This identifies the service of the IPN node to which error logs had previously been sent.

Transmit_DDR
Null

Receive_DDR
Null

Status_DDR
Null

Result The following result fields can be returned:
AS_Success
Illegal Request (range)
AE_NotInTable

FN_REGY_LogErrorToRegistry

This transaction requests the registry service to send an error logging record to the error logger.

Minor_function
20

Parameter_DDR
This is a pointer to the following data:

Byte	3	2	1	0
0 through n	Error Data			

Error Data
See “FN_REGY_LogErrorFromRegistry” for the definition of error data.

Transmit_DDR
Null

Receive_DDR
Null

Status_DDR
Null

Result The following result fields can be returned:
AS_Success
Illegal Request (range)

FN_REGY_LogErrorFromRegistry

This transaction requests the error logger to log the error data supplied.

To ensure that deadlock does not occur in the registry service, the receiver of this transaction should complete this transaction before issuing another transaction to the registry service.

Minor_function
21

Parameter_DDR
This is a pointer to the following data:

Byte	3	2	1	0
0	Reserved = 0	Sense Format	Template	Type
4 through 15	Serial Number			
16	Reserved = 0	Serial Number		
20	Port 1 SSA loop A	Port 2 SSA loop A	Port 1 SSA loop B	Port 2 SSA loop B
24	Count			
28 through n	Sense Data			

Type This defines the type of the sender of the error data:

TY_Disk
Disk

TY_Adapter
Adapter

Template
This defines the error template that should be used for logging the error data.

Sense Format
This defines the format of the sense-data filed when the type is TY_Adapter, as follows:

0 SD_Code:

Error code in bytes 28 through 30

1 SD_CodeAsn:

Error code in bytes 28 through 30, reserved in byte 31, array serial number in bytes 32 through 46

2 SD_CodeAsnCsn:

Error code in bytes 28 through 30, reserved in byte 31, array serial number in bytes 32 through 46, component serial number in bytes 48 through 62

3

Byte	Content
28 through 30	Error code = 000000h
31	Reserved
32	Network
	0 NI_NetworkA
	1 NI_NetworkB
33	Loop
	0 LP_Unknown
	1 LP_Loop
	2 LP_String
34	Legal
	0 LG_Unknown
	1 LG_Legal
	2 LG_Illegal
35	Master

0 MN_Unknown

1 MN_Master

2 MN_NonMaster

36 through 39
Node count (that is, the number of nodes, including this one, on this SSA network or 0xFFFFFFFF, if unknown)

40 through 43
Initiator count (that is, the number of initiators, including this one, on this SSA network or 0xFFFFFFFF, if unknown)

4

28 through 30
Error code

31 Reserved

32 through 46
Resource serial number

48 through 62
Logical Block Address

Serial Number
This 15-byte ASCII character field contains the serial number of the sender.

When the type field is TY_Adapter, the format of the serial number is the ASCII card serial number (as reported in the POS register) in bytes 4 through 11 and ASCII blanks in bytes 12 through 18.

When the type field is TY_Disk, the format of the serial number is as defined in “FN_ISALMgr_Inquiry” on page 119.

Port n This is the SSA address of the node in error on this port of the adapter card, or FFh if the disk in error is not connected to this port. If the type is TY_Adapter, this field is FFh.

Count This is the number of sense data bytes that follow this field.

Sense Data
If the type is TY_Disk, this is the SCSI sense data from the disk.

Note: The sense data received from the SSA-SCSI attachment to the disk is in big-endian format and this is returned in the parameter_DDR data without any byte swapping.

If the type is TY_Adapter, this is adapter status data. This includes the adapter error code in bytes 30 through 28 (byte 28 is the most

significant byte). The rest of the sense data (up to byte 59) may include the serial number of the failing array and that of a component disk drive (where appropriate).

Transmit_DDR
Null

Receive_DDR
Null

Status_DDR
Null

Result The following result fields can be returned:
AS_Success
Illegal Request (range)

FN_REGY_ConnectForResrcChange

This transaction informs the registry service that the client is interested in resource-change asynchronous alerts for resources of the specified owning module type (OMT).

Minor_function
22

Parameter_DDR
This is a pointer to the following data:

Byte	3	2	1	0
0	Node			
4	Service			
8	Reserved = 0		Synchro	Owning Module Type

Node This identifies the IPN node to which resource-change asynchronous alerts should be sent.

Service
This identifies the service of the IPN node to which resource-change asynchronous alerts should be sent.

Owning Module Type
This identifies the type of resource for which resource-change asynchronous alerts should be sent.

Synchro
When the synchro field is SR_Synchro, the registry service sends, before this transaction completes, a FN_REGY_ResrcChangeFromRegistry transaction for all resources of the specified owning module type currently registered.

When the synchro field is SR_NoSynchro, only resource state changes registered after this transaction has completed are reported by a FN_REGY_ResrcChangeFromRegistry transaction.

Transmit_DDR
Null

Receive_DDR
Null

Status_DDR
Null

Result The following result fields can be returned:
AS_Success
Illegal Request (range)
AE_TableFull

FN_REGY_DiscForResrcChange

This transaction informs the registry service that the client is no longer interested in resource-change asynchronous alerts for resources of the specified owning module type.

Minor_function
23

Parameter_DDR
This is a pointer to the following data:

Byte	3	2	1	0
0	Node			
4	Service			
8	Reserved = 0		Synchro	Owning Module Type

Node This identifies the IPN node to which resource-change asynchronous alerts had previously been sent.

Service
This identifies the service of the IPN node to which resource-change asynchronous alerts had previously been sent.

Owning Module Type
This identifies the type of resource for which resource-change asynchronous alerts had previously been sent.

Synchro
When the synchro field is SR_Synchro, a FN_REGY_ResrcChangeFromRegistry transaction is sent before the completion of this transaction for each resource of the specified owning module type known by the registry service.

When the synchro field is SR_NoSynchro, no transactions are sent as a result of this transaction.

Transmit_DDR
Null

Receive_DDR
Null

Status_DDR
Null

Result The following result fields can be returned:
AS_Success
Illegal Request (range)
AE_NotInTable

FN_REGY_ResrcChangeToRegistry

This transaction informs the registry service about a resource change.

Minor_function
24

Parameter_DDR
This is a pointer to the following data:

Byte	3	2	1	0
0	Undefined			
4	Service			
8	ResourceID			
12	Reserved = 0			Change Code

Service
This identifies the service of the resource change.

ResouceID
This identifies the resource that has changed. <!Á %comment; Á:dt.Owning Module Type :dd.This identifies the owning module type of resource that has changed. ‖‖>

Change Code
The resource can be in one of the following states:

Unknown:
It is not possible to communicate with this resource and, if its presence had previously been known and it had been opened, the handle has been closed.

RS_Offline:
The presence of the resource has been detected and a handle is still assigned, but communication to the resource is not now possible. When in this state, the only valid transactions that can be sent to this handle are

FN_ISAL_Close, FN_ISALMgrCharacteristics, and FN_ISALMgrStatistics. A result field AE_Offline is returned to all other transactions.

RS_Online:
The presence of the resource is known and is operational. It may or may not have been opened and a handle assigned. Even though it is operational it may not be fully functional, and some transactions may not be fully executed due to the degraded condition of the resource.

The change-code field identifies the reason for the resource change:

CC_Add:
The resource, which was previously unknown, is now in the RS_Offline state.

CC_SetOnline:
The resource, which was previously in the RS_Offline state, is now in the RS_Online state. Communication with this resource, which had a handle assigned, is now possible again.

CC_Add+CC_SetOnline:
The resource, which was previously unknown, is now in the RS_Online state. Communication is now possible.

CC_SetOffline:
The resource, which was previously in the RS_Online state, is now in the RS_Offline state. Communication to the resource is no longer possible but the handle is still assigned.

CC_Remove:
The resource, which was previously in the RS_Offline state, is now unknown. Communication to the resource is not possible and the handle has been closed.

CC_SetOffline+CC_Remove:
The resource, which was previously in the RS_Online state, is now unknown. Communication to the resource is not possible and a handle is not assigned.

Transmit_DDR
Null

Receive_DDR
Null

Status_DDR
Null

Result The following result fields can be returned:
AS_Success
Illegal Request (range)

AE_InvalidRID

FN_REGY_ResrcChangeFromRegistry

This transaction informs the previously-identified service of a resource change.

To ensure that deadlock does not occur in the registry service, the receiver of this transaction should complete this transaction before issuing another transaction to the registry service.

Minor_function
25

Parameter_DDR
This is a pointer to the following data:

Byte	3	2	1	0
0	Node			
4	Service			
8	ResourceID			
12	Reserved = 0		Synchro	Change Code

Node This identifies the IPN node of the resource change.

Service
This identifies the service of the resource change.

ResourceID
This identifies the resource that has changed. <!Á %comment; Á:dt.Owning Module Type :dd.This identifies the owning module type of the resource that has changed. ‖‖>

Change Code
This code identifies the reason for the resource change. The states of the resource are defined in “FN_REGY_ResrcChangeToRegistry” on page 103.

CC_Add:
The resource, which was previously unknown, is now in the RS_Offline state.

CC_SetOnline:
The resource, which was previously in the RS_Offline state, is now in the RS_Online state. Communication to this resource, which had a handle assigned, is now possible again.

CC_Add+CC_SetOnline:
The resource, which was previously unknown, is now in the RS_Online state. Communication is now possible.

CC_SetOffline -
The resource, which was previously in the RS_Online state,

is now in the RS_Offline state. Communication to the resource is no longer possible but the handle is still assigned.

CC_Remove:
The resource, which was previously in the RS_Offline state, is now unknown. Communication to the resource is not possible and the handle has been closed.

CC_SetOffline+CC_Remove:
The resource, which was previously in the RS_Online state, is now unknown. Communication to the resource is not possible and a handle is not assigned.

Synchro
The synchro field is SR_Synchro when the transaction is sent as a result of the synchro field in a FN_REGY_ConnectForResrcChange or FN_REGY_DiscForResrcChange transaction being SR_Synchro.

If the transaction is sent as a result of the synchro field being SR_Synchro in a FN_REGY_ConnectForResrcChange transaction, the change-code field is:
- CC_Add, if the resource is in the RS_Offline state
- A combination of CC_Add and CC_SetOnline, if the resource is in the RS_Online state.

If the transaction is sent as a result of the synchro field in a FN_REGY_DiscForResrcChange transaction being SR_Synchro, the change-code field is:
- CC_Remove, if the resource is in the RS_Offline state
- A combination of CC_Remove and CC_SetOffline, if the resource is in the RS_Online state.

The synchro field is SR_NoSynchro when the resource change transaction is not a result of the synchro field in a FN_REGY_ConnectForResrcChange or FN_REGY_DiscForResrcChange transaction being RS_Synchro.

Transmit_DDR
Null

Receive_DDR
Null

Status_DDR
Null

Result The following result fields can be returned:
AS_Success
Illegal Request (range)
AE_RetryWhenMemory

FN_REGY_ResrcList

This transaction returns a list of resource IDs that have been added to the registry service for a particular owning module type (OMT).

Minor_function
26

Parameter_DDR
This is a pointer to the following data:

Byte	3	2	1	0
0	Skip		Reserved = 0	Owning Module Type

Skip This defines the number of resource-list entries that should be skipped before the first resource returned in the Receive_DDR data.

Owning Module Type

This identifies the owning module type of resources that should be reported. The owning module types are:

OM_DriverPhysicalDisk
This is for a physical SSA-SCSI disk. It is used by the host to identify disks that can perform commands, such as HardwareInquiry and Open in Service Mode, that cannot be sent to a logical disk. All physical disks have one of these entries in the registry as well as having one of the following logical disk entries. Errors are logged against resource IDs of this owning module type.

OM_NotOwned
This indicates that the disk is not owned by a resource manager or by a driver. This type of disk cannot be used by a driver or resource manager and is therefore a spare disk until the owning module type is changed.

OM_DriverAdapter
Resource IDs with this OMT refer to other adapter cards. This is used to implement adapter to adapter communications needed for HACMP.

OM_DriverManualDisk
This indicates a disk that has been assigned a permanent resource ID with a configuration tool. Only personal systems require this.

OM_DriverAutomaticDisk
This is the other type of driver-owned logical disk. This indicates that the adapter, rather than an operator, has automatically assigned a number to a disk. All new disks are initialized with this value.

OM_FastWriteFilter
The fast write caching filter owns the disk.

OM_Raid5Filter
The RAID-5 filter owns the disk

OM_ListAll
Report resource for all owning module types

Transmit_DDR
Null

Receive_DDR
This is a pointer to a buffer which will receive the following data:

Byte	3	2	1	0
0	ResourceID			
4	Service Number			
8	Reserved = 0			State
12	ResourceID			
16	Service Number			
20	Reserved = 0			State
24	ResourceID			
28	Service Number			
32	Reserved = 0			State
.				
.	.			
n-8	ResourceID			
n-4	Service Number			
n	Reserved = 0			State

ResourceID
These are the resource IDs of the resources with the requested owning-module type. They are sorted in ascending order.

Service Number
This identifies the service for each resource ID of the requested type on this node.

State This can be:

RS_Offline:
The presence of the resource has been detected and a handle is still assigned, but communication to the resource is not now possible. When in this state, the only valid transaction that can be sent to this handle are FN_ISAL_Close, FN_ISALMgrCharacteristics, and FN_ISALMgrStatistics. A result field of AE_Offline is returned to all other transactions.

RS_Online:

The presence of the resource is known and is operational. It may or may not have been opened and a handle assigned. Even though it is operational it may not be fully functional, and some transactions may not be fully executed due to the degraded condition of the resource.

Status_DDR

This is a pointer to a buffer that receives the following data:

Byte	3	2	1	0
0	Count			

Count The number of entries in the received data DDR.

Result The following result fields can be returned:

AS_Success
Illegal Request (range)

FN_REGY_GetTempResrcID

This transaction returns a temporary resource ID that can be used by a resource manager that needs to invent a resource ID name. The resulting 32-bit field has a 24-bit number unique among all resource IDs except those of type OM_DriverManualDisk. The upper 8 bits (the owning module type) is set to zero and the caller must fill in his owning module type before the resource ID can be used.

Minor_function

27

Parameter_DDR

None

Transmit_DDR

Null

Receive_DDR

Null

Status_DDR

This is a pointer to a buffer which will receive the following data:

Byte	3	2	1	0
0	Resource ID			

Resource ID

The resulting prototype resource ID.

Result The following result fields can be returned:

AS_Success
Illegal Request (range)

FN_REGY_ConnectForHealthCheck

This transaction sent to the local registry service by any client that needs to be informed when a health check should be performed.

Minor_function
28

Parameter_DDR
This is a pointer to the following data:

Byte	3	2	1	0
0	Node			
4	Service			

Node This identifies the IPN node.

Service
This identifies the service of the IPN node that is able to perform health checks.

Transmit_DDR
Null

Receive_DDR
Null

Status_DDR
Null

Result The following result fields can be returned:
AS_Success
Illegal Request (range)
AE_TableFull

FN_REGY_DiscForHealthCheck

This transaction is sent to the local registry service by any client that no longer needs to be informed of when health checks should occur.

Minor_function
29

Parameter_DDR
This is a pointer to the following data:

Byte	3	2	1	0
0	Node			
4	Service			

Node This identifies the IPN node.

Service
This identifies the service of the IPN node that is no longer able to perform health checks.

Transmit_DDR
Null

Receive_DDR
Null

Status_DDR
Null

Result The following result fields can be returned:
AS_Success
Illegal Request (range)
AE_NotInTable

FN_REGY_HealthCheckToRegistry

This transaction is sent to the registry service by a client when a health check needs to be performed.

Minor_function
30

Parameter_DDR
Null

Transmit_DDR
Null

Receive_DDR
Null

Status_DDR
Null

Result The following result fields can be returned:
AS_Success
Illegal Request (range)

FN_REGY_HealthCheckFromRegistry

This transaction is sent by the registry service to all the local services that are registered as being able to perform health checks. It indicates these tests should occur now. The service sends error log data to the registry service which, for detected error conditions that cause a degraded operation or require a service action, forwards it to the error logger.

To ensure that deadlock does not occur in the registry service, the receiver of this transaction should complete this transaction before issuing another transaction to the registry service.

Minor_function
31

Parameter_DDR
Null

Transmit_DDR
Null

Receive_DDR
Null

Status_DDR
Null

Result The following result fields can be returned:
AS_Success
Illegal Request (range)

FN_REGY_SerialNumberSearch

This transaction returns the resource ID and service number of the resource identified by the serial number supplied.

Minor_function
32

Parameter_DDR
This is a pointer to the following data:

Byte	3	2	1	0
0 through 11	Serial Number			
12	Owning Module Type	Serial Number		

Serial Number
This identifies the resource for which the resource ID is requested.

Owning Module Type
This identifies the type of resource, with the requested serial number, that should be reported. If this field is zero, the resourceID of the resource of any owning module type (OMT), except OM_DriverPhysicalDisk, is reported.

Transmit_DDR
Null

Receive_DDR
Null

Status_DDR
This is a pointer to a buffer that receives the following data:

Byte	3	2	1	0
0	ResourceID			
4	Service Number			
8	Reserved = 0			State

ResourceID
This is the resource ID of the resource identified by the serial number and owning module type.

Service Number
This is the service number of the manager that controls the resource.

State The current state of the resource can be one of the following (described in “FN_REGY_ResrcChangeToRegistry” on page 103):
RS_Online

RS_Offline

Result The following result fields can be returned:
AS_Success
AE_Failure
Illegal Request (range)
AE_NotInTable

FN_REGY_TestResrcsReady

This transaction returns an AS_Success result when all the known resources are ready to receive transactions. This may involve a delay while, for example, the spindle motor of a disk drive is started. If all the resources are not ready within the time period defined in the parameter_DDR, the AS_Failure result field is returned. The registry service sends FN_ISALMgr_TestResrcsReady transactions to all services that are registered to inquire if all their resources are ready.

Minor_function
33

Parameter_DDR
This is a pointer to a the following data:

Byte	3	2	1	0
0	Time			

Time This defines the maximum duration in seconds before a result field must be returned.

Transmit_DDR
Null

Receive_DDR
Null

Status_DDR
None

Result The following result fields can be returned:
AS_Success
AE_Failure
Illegal Request (range)

FN_REGY_SetClusterNumber

This transaction identifies the cluster number of the system to the registry service. The cluster number can be in the range 0 through 2048. The adapter assumes cluster number 0 from power on until it has been set by this transaction.

Minor_function
34

Parameter_DDR
This is a pointer to a the following data:

Byte	3	2	1	0
0	Cluster Number			

Transmit_DDR
Null

Receive_DDR
Null

Status_DDR
None

Result The following result field can be returned:
AS_Success

FN_REGY_TestOneResrcReady

This transaction enquires of the registry the state of a resource identified by a serial number. The resource might not be in a state that permits it to be declared to the registry service.

Minor_function
35

Parameter_DDR
This is a pointer to a the following data:

Byte	3	2	1	0
0 through 11	Serial Number			
12	Reserved = 0	Serial Number		

Transmit_DDR
Null

Receive_DDR
Null

Status_DDR
None

Result The following result field can be returned:

AS_Success
The resource is known by the registry service and by a resource manager and can be used. If the resource is an array, it is not exposed, but it might be degraded or rebuilding.

AE_NotReady
The resource is known by a resource manager but it is not ready for use and has not been declared to the registry service. The resource might be a disk drive that is starting. This result is only returned if the resource is expected to become usable later.

AE_Offline
The resource is known by a resource manager but cannot be used because the array is in the offline state. An array is in this state when more than one of its components is not available.

AE_AvoidWrite
The resource is known by the registry service and a resource manager and can be used. However, write operations to the resource should be delayed because a write operation would cause an array to change from the exposed to the degraded state.

AE_NotInTable
The resource is not known by any resource manager.

Illegal Request (range)

Note: If the transaction is rejected with result AE_UnknownFunction, this should be treated as AE_NotInTable.

FN_REGY_SyncHCheckToRegy

In response to a FN_REGY_SyncHCheckToRegy transaction, the registry service issues a FN_REGY_SyncHCheckFromRegy transaction to all the connected services. If they all return AS_Success or AE_UnknownFunction, the registry service returns AS_Success. Otherwise, the registry service returns the most serious sense data it has received by means of the Status_DDR with an AE_Failure result field.

Minor_function
36

Parameter_DDR
Null

Transmit_DDR
Null

Receive_DDR
Null

Status_DDR
This is a pointer to the following data:

Byte	3	2	1	0
0	Length			
4 through n	Sense Data			

The sense DDR consists of a 4-byte length field followed by sense information of variable length. The length of the sense data is a multiple of 4 and is less than, or equal to, 36.

Result The following result field can be returned:
AS_Success
Illegal request (range)
AE_Failure

FN_REGY_SyncHCheckFromRegy

In response to a FN_REGY_HealthCheckFromRegy transaction, a service generates a FN_REGY_LogErrorToRegistry transaction shortly afterwards. However, in response to a FN_REGY_SyncHCheckFromRegy transaction, a service determines the most serious health-check complaint. The sense data that would usually be logged to the registry is returned in the Status_DDR data, and AE_Failure is returned in the result data. If there are no health-check complaints, the service returns AS_Success.

Any service that connects for health checks receives the new FN_REGY_SyncHCheckFromRegy transaction as well as the FN_REGY_HealthCheckFromRegy transaction.

Minor_function
37

Parameter_DDR
Null

Transmit_DDR
Null

Receive_DDR
Null

Status_DDR
This is a pointer to the following data:

Byte	3	2	1	0
0	Length			
4 through n	Sense Data			

The sense DDR consists of a 4-byte length field followed by sense information of variable length. The length of the sense data is a multiple of 4 and is less than, or equal to, 36.

Only adapter errors can be returned by this means. The service receiving the FN_REGY_SyncHCheckFromRegy transaction is not permitted to perform lengthy processing (for example, that involving other transactions) before completing the transaction; such delay might cause deadlock within the adapter microcode.

Result The following result field can be returned:
AS_Success
Illegal request (range)
AE_Failure
AE_UnkownFunction

Disk Service

The SSA 4-Port RAID Adapter disk service uses the IPN Storage Access Language (ISAL) to provide access to the disks in SSA subsystems. The language is similar to SCSI; however, only the functions required by clients are included.

ISAL has a single access mode that is set when the resource is opened. The ISAL transaction that opens a resource establishes a logical connection between the master and slave for that resource. This transaction is sent to the ISAL manager service which returns a handle for that manager that is used, in subsequent transactions, to access the resource just opened. All requests that are sent to the disk service are attempted. Error recovery is performed by the ISAL server and, if this fails, the sender is not required to retry the failed request. There is no contingent allegiance mode. Error logs are reported to the error logger without the sender having to request error data. If a request fails, commands that are waiting are not rejected; they are attempted in turn.

ISAL Transactions

The ISAL transactions that the disk service handles are listed in the following table.

In addressing resources, the handle number acts as a disk number (like a SCSI LUN). The transmit and receive parameters are used to point to I/O data buffers. The function parameter is sent in the minor function code field of the transaction function word, and any other parameters are sent in the parameter field of the transaction.

A physical resource is one with owning module type OM_DriverPhysicalDisk. A logical resource is one with any other owning module type.

Table 37. ISAL Transactions

Transaction	Minor_function	Valid to Logical Resource	Valid to Physical Resource
FN_ISALMgr_Inquiry	40	Yes	Yes
FN_ISALMgr_HardwareInquiry	41	No	Yes

Table 37. ISAL Transactions (continued)

Transaction	Minor_function	Valid to Logical Resource	Valid to Physical Resource
FN_ISALMgr_SetOwningModuleType	42	Yes	No
FN_ISALMgr_AssignManualResrcID	43	Yes	No
FN_ISALMgr_GetPhysicalResrcIDs	44	Yes	No
FN_ISALMgr_GetPhysSvcAndRIDs	64	Yes	No
FN_ISALMgr_TestResrcsReady	45	Yes	Yes
FN_ISALMgr_TestOneResrcReady	63	Yes	Yes
FN_ISALMgr_VPDInquiry	46	Yes (note 4)	Yes
FN_ISALMgr_Characteristics	47	Yes	Yes
FN_ISALMgr_Statistics	48	Yes	Yes
FN_ISALMgr_FlashIndicator	49	Yes	Yes
FN_ISALMgr_NetworkInquiry (note 5)	66	Yes	No
FN_ISALMgr_Preferences (note 5)	67	Yes	No
FN_ISALMgr_LockQuery	69	Yes	Yes
FN_ISALMgr_Open	50	Yes (note 1)	Yes (note 3)
FN_ISAL_Close	51	Yes	Yes
FN_ISAL_Read	52	Yes	Yes
FN_ISAL_Write	53	Yes	Yes (Note 2)
FN_ISAL_Format	54	No	Yes (Note 2)
FN_ISAL_Progress	55	No	Yes (Note 2)
FN_ISAL_Lock	56	Yes	Yes
FN_ISAL_Unlock	57	Yes	Yes
FN_ISAL_Test	58	Yes	Yes
FN_ISAL_SCSI	59	No	Yes
FN_ISAL_Download	60	No	Yes
FN_ISAL_Fence (note 6)	62	Yes	Yes
FN_ISALMgr_Flush (note 5)	68	No	Yes

Notes:

1. A logical resource cannot be opened in MD_Service or MD_SCSI mode.
2. Format, Progress and Write transactions are not allowed to a physical resource if the corresponding logical resource for that device is also open.
3. A physical resource cannot be opened in MD_Service mode if the corresponding logical resource for that device is currently open. A physical resource cannot be opened in MD_ISAL_HA mode.
4. The array manager does not support the FN_ISALMgr_VPDInquiry.
5. These transactions are not supported on the PCI SSA 4-Port RAID Adapter.
6. Array manager does not support this transaction on the PCI SSA 4-Port RAID Adapter.

FN_ISALMgr_Inquiry

This transaction is sent to the disk service requesting the serial number of the specified resource.

Minor_function
40

Parameter_DDR
This is a pointer to the following data:

Byte	3	2	1	0
0	Resource ID			

Resource ID
This identifies the resource

Transmit_DDR
Null

Receive_DDR
Null

Status_DDR
This is a pointer to a buffer that receives the following data:

Byte	3	2	1	0
0 through 11	Serial Number			
12	Reserved = 0	Serial Number		

Serial Number
This 15-byte ASCII field contains the serial number of the specified resource. It has the following format:

- **Non-RAID Disk:**

Byte	3	2	1	0
0 through 11	Product Identifier			
12	Reserved = 0	'D'	SSA-SCSI LUN	

Note: The ASCII character 'D' is reported in byte 14 of the Status_DDR data if the resource is an SSA disk drive. If the SSA device is of any other type, byte 14 is the hexadecimal digit in bits 3 through 0 of byte 0 of the SSA-SCSI Inquiry data for that device, reported as an ASCII character. For example, the Character '5' is reported for a CD-ROM drive.

Product Identifier
This ASCII field identifies the device attached to the SSA bus. This is the 6-byte IEEE SSA unique ID translated to a 12-character ASCII string.

SSA-SCSI LUN
This ASCII field identifies the SSA-SCSI logical unit number of the resource.

- **Array resource:**

Byte	3	2	1	0
0 through 11	Array Name			
12	Reserved = 0	Array Letter	Array Name	

Array Name
This 14-ASCII-character field identifies the array.

Array Letter
This ASCII character identifies the type of filter of the array resource. The letter 'K' is used for a RAID-5 array.

Result The following Result fields can be returned:
AS_Success
Illegal Request (range)
AE_InvalidRID

FN_ISALMgr_HardwareInquiry

This transaction is sent to the disk service to return details about the specified resource. It returns hardware specific information. Only SSA resource managers that control physical SSA devices support this transaction. The transaction is rejected with illegal-request result if the owning module type of the resource is other than OM_DriverPhysicalDisk.

Minor_function
41

Parameter_DDR
This is a pointer to the following data:

Byte	3	2	1	0
0	Resource ID			
4	Reserved = 0			Immed

Resource ID
This identifies the resource

Immed This field controls whether the result field is returned immediately or after error recovery. (If the disk drive motor is stopped, error recovery can take over a minute.) The field can have the following values:

HI_Immediate
If the motor is stopped, AS_Success is returned immediately with status of ST_Failed and fail code of HF_MotorFail. The adapter attempts error recovery to restart the motor after the

result field is returned. HI_Immediate is assumed if the parameter data length is less than 8 bytes.

HI_NotImmediate
If the motor is stopped, full error recovery is performed before the result field is returned.

Transmit_DDR
Null

Receive_DDR
Null

Status_DDR
This is a pointer to a buffer that receives the following data:

Byte	3	2	1	0
0	Port 1 SSA loop A	Port 2 SSA loop A	Port 1 SSA loop B	Port 2 SSA loop B
4	Reserved = 0		Fail Code	Status

Port n This is the SSA address of the node on this port of the adapter card. If the resource is not connected to this port then a value of FFh is returned. These fields are valid if the result field is AS_Success or AE_ReservationConflict.

Status This reports the state of the resource and is valid if the result field is AS_Success. It has the following definition:

ST_Good
Good.

ST_Failed
Failed. In this state, if the resource is a target on a SSA link, a Test Unit Ready SSA command is rejected with check-condition status. This could be due to a failure of POST2, a stopped motor, or any degraded mode condition.

ST_LossRedundancy
In this state, the resource has lost some redundancy, for example, loss of redundant power or cooling. The ISAL manager determines this by sending a SSA-SCSI Inquiry command to the resource.

Fail Code
This provides more details if the status is ST_Failed:

HF_MotorFail
The motor is stopped

HF_Unknown
No more details of the failure are available.

Result The following result fields can be returned:
AS_Success
Illegal Request (range)
AE_InvalidRID
AE_ReservationConflict

AE_Offline
AE_OfflineTimeout
AE_Failure

FN_ISALMgr_SetOwningModuleType

This transaction is sent to the disk service to set the owning module type (OMT) for the specified resource. This causes the ID for the resource to change and the new OMT to be written in the label record of the ISAL reserved area. This transaction is not used to change the OMT to OM_DriverManualDisk. FN_ISAL_AssignManualResrcID is used for that purpose.

If the resource is in the open state when this transaction is received, AS_success is returned in the result field, the new resourceID is created, and the old resource goes to the RS_Offline state. The transaction is rejected with illegal-request result if the owning module type of the resource is OM_DriverPhysicalDisk.

Minor_function
42

Parameter_DDR
The data descriptor is a pointer to the following data:

Byte	3	2	1	0
0	Old Resource ID			
4	Reserved = 0			Owning Module Type

Old Resource ID
This specifies the current resource for which the owning module type should be set.

Owning Module Type
This defines the type of disk service that controls the resource.

Transmit_DDR
Null

Receive_DDR
Null

Status_DDR
The data descriptor is a pointer to the following data:

Byte	3	2	1	0
0	New Resource ID			

New Resource ID
This specifies the resource's new ID

Result The following result fields can be returned:
AS_Success
Illegal Request (range)

AE_InvalidRID
AE_MediumError
AE_HardwareError
AE_ReservationConflict
AE_FencedOut
AE_Offline
AE_TableFull
AE_FormatDegraded
AE_FormatInProgress
AE_Failure
AE_NonIsal

The resource manager responds to this transaction by removing the old resource ID from the registry, getting a new temporary resource ID (by using a FM_REGY_GetTempResrcID command), setting the new OMT into it, and adding this to the registry.

FN_ISALMgr_AssignManualResrcID

This transaction is sent to the disk service to change a resource ID and owning module type. The owning module type is changed to type OM_DriverManualDisk and this is written in the label record of the ISAL reserved area.

If the resource is in the open state when this transaction is received, AS_Success is returned in the result field, the new resourceID is created, and the old resource goes to the RS_Offline state. The transaction is rejected with illegal-request result if the owning module type of the resource is OM_DriverPhysicalDisk.

Minor_function
43

Parameter_DDR
The data descriptor is a pointer to the following data:

Byte	3	2	1	0
0	Old Resource ID			
4	New Resource ID			

Old Resource ID
This specifies the current resource's ID

New Resource ID
This specifies the resource's new ID. This must have an OMT of OM_DriverManualDisk in the format Ox04HHNNNN where:

Field	Value
HH	00
NNNN	disk number (a value of 0000 is valid but should be avoided)

Transmit_DDR
Null

Receive_DDR
Null

Status_DDR
Null

Result The following result fields can be returned:
AS_Success
AS_InvalidRID
Illegal Request (range)
AE_InvalidRID
AE_MediumError
AE_HardwareError
AE_ReservationConflict
AE_FencedOut
AE_Offline
AE_FormatDegraded
AE_FormatInProgress
AE_Failure
AE_NonIsal

The resource manager responds to this transaction by removing the old resource ID from the registry and adding the new one (using the FN_REGY_ResrcChangeToRegistry transaction for both actions). If the act of adding the new resource ID results in a return of AE_InvalidRID, this means that the new resource ID is already in use and an error is reported to the user.

FN_ISALMgr_GetPhysicalResrcIDs

This transaction is used to translate a logical resource ID into its physical components. This function returns a list of resource IDs that are of type OM_DriverPhysicalDisk. The transaction is rejected with illegal-request result if the owning module type of the resource is OM_DriverPhysicalDisk.

Minor_function
44

Parameter_DDR
This is a pointer to the following data:

Byte	3	2	1	0
0	Logical resource ID			

Logical resource ID
This identifies the logical resource ID that is to be translated.

Transmit_DDR
Null

Receive_DDR
This is a pointer to a buffer that receives the following data:

Byte	3	2	1	0
0	Physical resource ID			
4	Physical resource ID			
.				
n	Physical resource ID			

Physical resource ID
This is a list of physical resource IDs that make up the logical resource ID

Status_DDR
This is a pointer to a buffer that receives the following data:

Byte	3	2	1	0
0	Count			

Count The number of entries in the received data DDR.

Result The following result fields can be returned:
AS_Success
Illegal Request (range)
AE_InvalidRID
AE_Offline

FN_ISALMgr_GetPhysSvcAndRIDs

This transaction is used to translate the ID of a logical resource into the IDs of its physical components. This function returns a list of resource IDs that are of the type OM_DriverPhysicalDisk and the service number that owns this resource.

The transaction is rejected with Illegal Request if the owning module type of the resource is OM_DriverPhysicalDisk.

Minor_function
64

Parameter_DDR
This is a pointer to the following data:

Byte	3	2	1	0
0	Logical Resource ID			

Logical Resource ID
This identifies the logical resource ID that is to be translated.

Transmit_DDR
Null

Receive_DDR
This is a pointer to a buffer that receives the following data:

Byte	3	2	1	0
0	Service number			
4	Physical resource ID			
8	Service number			
12	Physical resource ID			
.	.			
n-4	Service number			
n	Physical resource ID			

Service number
This identifies the service that owns each resource in the list.

Physical resource ID
This is a list of physical resource IDs that make up the logical resource ID.

Status_DDR
This is a pointer to a buffer that receives the following data:

Byte	3	2	1	0
0	Count			

Count The number of entries in the received data DDR.

Result The following result fields can be returned:
AS_Success

Illegal Request (range)

AE_InvalidRID

AE_Offline

AE_UnknownFunction
(not supported on the PCI SSA 4-Port RAID Adapter)

FN_ISALMgr_TestResrcsReady

This transaction is used to test that all the resources that are known to and controlled by the resource manager have started and are operational.

Minor_function
45

Parameter_DDR
Null

Transmit_DDR
Null

Receive_DDR
Null

Status_DDR
Null

Result The following result fields can be returned:

AS_Success
All known resources are operational

AE_Failure
One or more resources controlled by this manager is not yet operational. This might be a disk drive that has not reached its operating speed.

FN_ISALMgr_TestOneResrcReady

The registry service sends this transaction to each resource manager to enquire about the state of a resource identified by a serial number. The resource might not be in a state that permits it to be declared to the registry service.

Minor_function
63

Parameter_DDR
This is a pointer to a the following data:

Byte	3	2	1	0
0 through 11	Serial Number			
12	Reserved = 0	Serial Number		

Transmit_DDR
Null

Receive_DDR
Null

Status_DDR
None

Result The following result field can be returned:

AS_Success
The resource is known by the registry service and by a resource manager and can be used. If the resource is an array, it is not exposed, but it might be degraded or rebuilding.

AE_NotReady
The resource is known by a resource manager but it is not ready for use and has not been declared to the registry service. The resource might be a disk drive that is starting. This result is only returned if the resource is expected to become usable later.

AE_Offline
The resource is known by a resource manager but cannot be used

because the array is in the offline state. An array is in this state when more than one of its components is not available.

AE_AvoidWrite
The resource is known by the registry service and a resource manager and can be used. However, write operations to the resource should be delayed because a write operation would cause an array to change from the exposed to the degraded state.

AE_NotInTable
The resource is not known by any resource manager.

Illegal Request (range)

Note: If the transaction is rejected with result AE_UnknownFunction, this should be treated as AE_NotInTable.

FN_ISALMgr_VPDInquiry

This transaction is sent to the disk service to obtain Vital Product Data of the resource identified by the resource ID field.

Minor_function
46

Parameter_DDR
This is a pointer to the following data:

Byte	3	2	1	0
0	ResourceID			
4	Reserved = 0		Page Code	EVPD

ResourceID
This identifies the resource for this transaction.

EVPD The Enable Vital Product Data (EVPD) field controls whether the data returned is standard inquiry data or individual VPD pages. EVPD can be:

VP_NoEVPD
Standard VPD inquiry data is returned.

VP_EVPD
The VPD inquiry data of the page identified by the page-code field is returned.

Page Code
This identifies the page of vital VPD inquiry data to be returned. Page 00h identifies the pages that can be returned.

Transmit_DDR
Null

Receive_DDR
This is a pointer to a buffer that receives the Vital Product Data. This is the

same data as that returned to the SSA-SCSI Inquiry command which is defined in the functional specification of the resource.

Status_DDR
This is a pointer to a buffer that receives the following data:

Byte	3	2	1	0
0	Count			

Count The number of bytes in the received data DDR.

Result The following result fields can be returned:
AS_Success
Illegal Request (range)
AE_InvalidRID
AE_HardwareError
AE_Offline
AE_FormatInProgress
AE_FormatDegraded

FN_ISALMgr_Characteristics

This transaction is sent to the disk service to obtain the blocksize and capacity of the resource identified by the resource ID field.

The size returned does not include the area of the disk that is reserved for use by the adapter.

Minor_function
47

Parameter_DDR
This is a pointer to the following data:

Byte	3	2	1	0
0	ResourceID			

ResourceID
This identifies the resource for this transaction.

Transmit_DDR
Null

Receive_DDR
Null

Status_DDR
This is a pointer to a buffer that receives the following data:

Byte	3	2	1	0
0	Number of Blocks			
4	Bytes per Block			
8	Number of Reserved Blocks			

Number of blocks
This field identifies the number of blocks available for user data.

Bytes per Block
This field identifies the blocksize of the user data.

Number of Reserved Blocks
This field identifies the number of blocks in the ISAL reserved area that are available. This does not include the blocks that the manager may be using for its own use, for example, for a label record.

Result The following result fields can be returned:
AS_Success
Illegal Request (range)
AE_InvalidRID
AE_HardwareError
AE_ReservationConflict
AE_FormatInProgress
AE_FormatDegraded

FN_ISALMgr_Statistics

This transaction is sent to the disk service to obtain statistics on the transactions executed for this adapter by the resource identified by the resource ID field. The statistics are cumulative from power-on, or adapter reset, and wrap on an overflow.

Minor_function
48

Parameter_DDR
This is a pointer to the following data:

Byte	3	2	1	0
0	ResourceID			

ResourceID
This field identifies the resource for this transaction.

Transmit_DDR
Null

Receive_DDR
Null

Status_DDR
This field is a pointer to a buffer that receives the following data:

Byte	3	2	1	0
0	Number of Reads			
4	Number of Writes			
8	Number of Blocks Read			
12	Number of Blocks Written			

Result The following result fields can be returned:
AS_Success
Illegal Request (range)
AE_InvalidRID

FN_ISALMgr_FlashIndicator

This transaction is sent to the disk service to flash a light on the resource identified by the resource ID field.

Minor_function
49

Parameter_DDR
This is a pointer to the following data:

Byte	3	2	1	0
0	ResourceID			
4	Flash			

ResourceID
This field identifies the resource for this transaction.

Flash When the flash field is 0h, the light does not flash. When the flash field is nonzero, the light flashes continuously: one second on, one second off.

Transmit_DDR
Null

Receive_DDR
Null

Status_DDR
Null

Result The following result fields can be returned:
AS_Success
Illegal Request (range)
AE_Offline
AE_InvalidRID
AE_FencedOut
AE_HardwareError
AE_ReservationConflict
AE_FormatInProgress
AE_OfflineTimeout

AE_NotSupported

FN_ISALMgr_NetworkInquiry

This transaction is sent to the disk service to determine on which network the resource is attached. This is required for HA adapters that require the member disks to be attached on the same loop.

Minor_function
66

Parameter_DDR
This is a pointer to the following data:

Byte	3	2	1	0
0	Resource ID			

Resource ID
This identifies the resource for this transaction.

Transmit_DDR
Null

Receive_DDR
Null

Status_DDR
This is a pointer to a buffer that receives the following data:

Byte	3	2	1	0
0	Network ID			

Network ID
This identifies the SSA loop or string on which the resource is attached. it can be one of the following:

NI_NetworkA
In an SSA loop or string attached to adapter SSA interface A

NI_NetworkB
In an SSA loop or string attached to adapter SSA interface B

NI_NullNetwork
No network applicable. This can be returned by a resource exported by RDSK.

Result The following result fields can be returned:
AS_Success

Illegal Request (range)

AE_UnkownFunction
(Returned by the PCI SSA 4-Port RAID Adapter)
AE_InvalidID

AE_Offline

FN_ISALMgr_Preferences

This transaction is sent to the disk service to enquire about the referred mode of operation. This is particularly useful to the fast-write filter to optimize its destaging of data.

Minor_function
67

Parameter_DDR
This is a pointer to the following data:

Byte	3	2	1	0
0	Resource ID			

Resource ID
This identifies the resource for this transaction.

Transmit_DDR
Null

Receive_DDR
Null

Status_DDR
This is a pointer to a buffer that receives the following data:

Byte	3	2	1	0
0	Valid mask			
4	Word 0			
8	Word 1			
.	.			
116	Word 28			
120	Word 29			

Valid mask
Each bit set in this field indicates that the corresponding word contains valid information. If the resource manager has no preferences, it can return all the bits in this field to zero. Bit 31 is reserved.

Words 0 through 29
Each word has an assigned meaning:

0 Destage Quantization

Recommended amount of data that should be destaged for optimum performance. Useful for the fast-write filter.

1 Destage offset

Recommended offset of the start of destaged data for optimum performance. Useful for the fast-write filter.

2 Queue depth

Recommended depth of queue of transactions to keep the resource busy.

3 Geometry sector

This word can be used to recommend as different value for OS/2 for the Cylinder/Head/Sector geometry.

4 Write queue depth

Recommended depth of queue of write transactions for this resource. This is used by the fast-write filter to restrict the number of queued write transactions to one; this increases the possibility of transactions being coalesced into full-stride writes. Other filters do not set the valid bit for word 4; therefore, the device driver includes both write and read transactions queued either using the value of word 2 or its own algorithm.

5 through 29
Not currently assigned.

Result The following result fields can be returned:
AS_Success

Illegal Request (range)

AE_UnkownFunction
This function might be returned by resource managers on adapters that do not support the function or by managers that do not have any preferences.
AE_InvalidRID

FN_ISALMgr_LockQuery

This transaction is sent to the disk service to determine to which adapter or system a resource is locked.

This transaction is supported only on adapters that perform locking without issuing a Reserve transaction to the disk drive. This transaction is not supported on the PCI SSA 4-Port RAID Adapter.

Minor_function
69

Parameter_DDR
The data descriptor is a pointer to the following data:

Byte	3	2	1	0
0	Resource ID			

ResourceID
This identifies the resource for this transaction.

Transmit_DDR
Null

Receive_DDR
Null

Status_DDR
This is a pointer to a buffer that receives the following data:

Byte	3	2	1	0
0	Reserved = 0		Cluster Number	
4 through 11	AdapterID			

If the resource is not locked, the entire status data is zero. If it is locked, it is either locked to a cluster number or an adapter ID and that field is returned.

Result The following result fields can be returned:
AS_Success

AE_UnkownFunction
(returned by the SSA 4-Port RAID Adapter)
AE_Offline

Illegal Request (range)

AE_FormatInProgress

AE_FormatDegraded

AE_NonIsal

FN_ISALMgr_Open

This transaction is sent to the disk service to request that a resource is opened. It returns a handle to be used to address the requested resource.

Minor_function
50

Parameter_DDR
The data descriptor is a pointer to the following data:

Byte	3	2	1	0
0	Resource ID			
4	Reserved = 00h	Sharing Mode	Access Type	Operation Mode

Resource ID

ID number of the resource requested to be opened.

Operation Mode

MD_ISAL

IPN Storage Access Language (ISAL)

MD_ISAL_HA

The IPN Storage Access Language (ISAL) is to be used as for the MD_ISAL_HA operation mode, but whenever access to the resources is lost (which may only be temporary) a FN_REGY_ResrcChangeToRegistry transaction is sent with change code CC_SetOffline and transactions to the resource are terminated with AE_Offline. In the HA manager, the client filter needs to be aware of any loss of a resource even if it is temporary.

In operation mode MD_ISAL, temporary access to a resource is not immediately reported as access may be restored after an interval following, for instance, an SSA network transient or network reset. The device driver should use MD_ISAL operation mode and array filters in an HA environment will use MD_ISAL_HA.

MD_ISAL_HA mode is not supported on the PCI SSA 4-Port RAID Adapter. It is also not supported for physical resource IDs.

If the SSA network is illegal when an attempt is made to open a resource in MD_ISAL_HA mode, the transaction is rejected with an AE_Failure result.

MD_SCSI

SCSI pass-through. When a resource is in SCSI pass-through mode transactions other than SCSI sent to the returned handle are rejected with illegal-request result. If this mode is requested, the Open transaction is rejected with illegal-request result if it is sent to any resource ID that is not of the owning module type OM_DriverPhysicalDisk.

MD_Service

Service Mode. If this mode is requested, the Open transaction is rejected with Illegal Request, if it is sent to any resource ID that is not of the owning-module type OM_Driver_PhysicalDisk.

Certain conditions do not allow the resource to be open in MD_Service mode:

AE_Logopen
The associated logical resource is currently open.

AE_SSAString
The resource is in an SSA string rather than an SSA loop, but it is not the last node in the string

AE_ReservationConflict
The associated logical resource is currently locked. (Not supported on the PCI SSA 4-Port RAID Adapter.)

AE_FencedOut
The associated logical resource is currently fenced out. (Not supported on the PCI SSA 4-Port RAID Adapter.)

When a resource is in service mode, the adjacent SSA ports to this node are wrapped and the check light on the selected resource is turned on.

Access Type

AT_All Read and Write transactions allowed

Sharing mode

SM_DenyNone
Multiple clients are allowed to open this disk service for this resource.

SM_DenyAll
Deny read and write access.

If another client issues an Open transaction to this disk service for this resource, it is rejected with an illegal-request result. This controls whether other clients can open the resource through this disk service; it does not imply that the resource is reserved to this client only; another client can access the resource through another disk service.

Transmit_DDR
Null

Receive_DDR
Null

Status_DDR
The data descriptor is a pointer to the following data:

Byte	3	2	1	0
0	Handle			

Handle This is the number that the client should use to address the resource.

Result The following result fields can be returned:

AS_Success
AE_AccessDenied
Illegal Request (range)
AE_InvalidRID
AE_SSAString
AE_LogOpen
AE_SSAString
AE_InService
AE_ReservationConflict
AE_FencedOut
AE_Offline
AE_InServiceMode
AE_ReservationConflict (only if opening in MD_Service mode)
AE_FencedOut (only if opening in MD_Service mode)
AE_OtherAdapterInServiceMode

FN_ISAL_Close

This transaction is sent to the disk service to close the resource identified by the handle field. If any transactions are active for the resource with this handle, the resource is not closed and the transaction terminates with an illegal-request result field.

If the resource being closed was in service mode, it is returned to normal mode before the close is completed. This may involve unwrapping SSA links of adjacent nodes.

If the resource being closed was locked before the ISAL_Close transaction, it remains locked at the end of the transaction.

If the resource can be closed and it is not open to other clients, any data or metadata that is held for any of its members is flushed to those members so that they are synchronized. That is, the service behaves as if it had received a FN_ISAL_Flush transaction immediately before the FN_ISAL_Close transaction. Any errors encountered during this flushing are ignored and the result field returned to the FN_ISAL_Close transaction is that which would have been returned if the flushing had not been attempted.

Minor_function
51

Parameter_DDR
The data descriptor is a pointer to the following data:

Byte	3	2	1	0
0	Handle			

The handle field identifies the resource for this transaction.

Transmit_DDR
Null

Receive_DDR
Null

Status_DDR
Null

Result The following result fields can be returned:
AS_Success
Illegal Request (range)

FN_ISAL_Read

This transaction is sent to the disk service to read the specified blocks from the resource identified by the handle field.

Minor_function
52

Parameter_DDR
This is a pointer to the following data:

Byte	3	2	1	0
0	Handle			
4	Address			
8	Count			
12	Extended Flags	Reserved = 0	Flags	Priority

Handle This identifies the resource that is to be read

Address
This is the logical block address of the data to be read

Count This is the number of logical blocks to be read

Priority
Reserved

Flags This field has bit-significant values. Multiple flags can be set. Bit values not specified are ignored.

FF_Verify
Verify data. No data is transferred to the client. The manager validates that the data could be read if requested. For an array, data is read from the components, and might be reconstructed from the other components, but it is not transferred to the client.

This option is available primarily for service functions to physical disks or to verify that data in an array can be read. If it is issued to a fast-write resource, an attempt might be made to destage cached data first. It is not recommended that FF_Verify is used for fast-write resources because this might incur write operations to the disk which might not have been intended.

FF_ExtendedFlags
The extended-flags field is not zero.

FF_Split
Data is allowed to be received out of order

FF_ReadDisk
Data must be read from the device and not from a cache

FF_ISALReservedArea

This flag causes the data to be read from the area of the disk drive reserved for ISAL. This is a separately addressed area of the resource starting at address zero. It follows the access type and sharing modes defined when the resource is opened.

The blocks that can be read are from address zero to the number of reserved blocks reported in FN_ISALMgr_Characteristics, minus one. The client may use these blocks as needed. The label record and fence sector are not visible through this interface.

Extended Flags
This field has bit-significant values.

EF_Override
Read operations are allowed to the resource even when it is reserved to another host or fenced out from this host.

Transmit_DDR
Null

Receive_DDR
This is a pointer to the buffer that receives the read data. The length of this buffer must be equal to or greater than the total number of bytes in the logical blocks requested.

Status_DDR
This is a pointer to the following data that is returned if the result field is AE_Warning or AE_MediumError:

Byte	3	2	1	0
0	Reserved = 0			Hint Flags
4	Address			

Hint Flags

RF_ReassignWarn
This flag, when set, indicates that the logical block identified by the address field should be reassigned. The logical block

address must be within the range of the blocks requested in the Read transaction. All blocks up to this address must have been sent to the client.

Address
This is the address of the logical block that should be reassigned when the result field contains either AE_Medium Error or AS_Warning and the hint-flag field is RF_ReassignWarn.

Result The following result fields can be returned:
AS_Success
AE_ReservationConflict
AS_Warning
AE_HardwareError
AE_NotReady
AE_MediumError
AE_AccessDenied
AE_InvalidSignature
Illegal Request (range)
AE_Offline
AE_FencedOut
AE_FormatInProgress
AE_FormatDegraded
AE_OfflineTimeout

FN_ISAL_Write

This transaction is sent to the disk service to write the specified blocks to the resource identified by the handle field. The transaction is rejected with illegal-request result if the owning module type of the resource is OM_DriverPhysicalDisk and the corresponding logical resource is currently open.

Minor_function
53

Parameter_DDR
This is a pointer to the following data:

Byte	3	2	1	0
0	Handle			
4	Address			
8	Count			
12	Extended Flags	Reserved = 0	Flags	Priority

Handle The identifies the resource that is to be written

Address
This is the logical block address where the data is to be written

Count This is the number of logical blocks to be written

Priority
Reserved

Flags This field has bit-significant values. Multiple flags can be set. Bit values not specified are ignored.

FF_Verify
Verify that, after writing the data, it can be read back (with reconstruction if necessary, in the case of an array).

FF_ExtendedFlags
The extended-flags field is not zero.

FF_FastWrite
The transaction can be completed before the data is written to the disk.

FF_ReassignWrite
The logical block is reassigned to another physical sector before being written with the data supplied. The count must be 1 when this option is requested.

FF_Split
Data is allowed to be written on the disk out of order. For RAID-5 arrays, a resource-dependent-value attribute can be set, from the configurator, to allow splits on an aligned 4K page even when the FF_Split bit is off. (see “RAID-5” on page 77.)

FF_ISALReservedArea

The data is written to the area of the disk reserved for ISAL. This is a separately addressed area of the resource starting at address zero. It follows the access type and sharing modes defined in the open of the resource.

The blocks that can be written are from address zero to the number of reserved blocks reported in FN_ISALMgr_Characteristics, minus one. The client can use these blocks as needed. The label record and fence sector are not visible through this interface.

Extended Flags
This field has bit-significant values. No flags are specified.

Transmit_DDR
This is a pointer to the transmit buffer. The length of this buffer must be equal to or greater than the total number of bytes of the logical blocks requested.

Receive_DDR
Null

Status_DDR
This is a pointer to the following data, which is returned when the result field is AE_Warning or AE_MediumError:

Byte	3	2	1	0
0	Reserved = 0			Hint Flags
4	Address			

Hint Flags

RF_ReassignWarn
This flag, when set, indicates that the logical block identified by the address field should be reassigned. The logical block address must be within the range of the blocks specified in the Write transaction.

Address
This is the address of the logical block that should be reassigned when the result field contains either AE_Medium Error or AS_Warning and the hint flag field is RF_BlockWarn.

Result The following result fields can be returned:
AS_Success
AE_ReservationConflict
AS_Warning
AE_HardwareError
AE_NotReady
AE_MediumError
AE_AccessDenied
AE_InvalidSignature
Illegal Request (range)
AE_Offline
AE_FencedOut
AE_WriteProtect
AE_LogOpen
AE_FormatInProgress
AE_FormatDegraded

FN_ISAL_Format

This transaction is sent to the disk service to start formatting the entire disk in the resource identified by the handle field. AS_Success is returned if formatting can be started. The FN_ISAL_Progress transaction can be used to track the progress and completion of the format operation. The transaction is rejected with illegal-request result if the owning module type of the resource is other than OM_DriverPhysicalDisk or corresponding logical resource ID for this device is currently open.

Minor_function
54

Parameter_DDR
This is a pointer to the following data:

Byte	3	2	1	0
0	Handle			
4	Blocksize			

Handle This identifies the resource for this transaction

Blocksize
This is the number of bytes in each logical block. This must be a value that is supported by the disk drive.

Transmit_DDR
Null

Receive_DDR
Null

Status_DDR
Null

Result The following result fields can be returned:
AS_Success
AE_ReservationConflict
AE_HardwareError
AE_NotReady
Illegal Request (range)
AE_Offline
AE_FencedOut
AE_LogOpen
AE_FormatInProgress

The progress of the format operation can be obtained by issuing a Progress transaction. If the format operation is aborted or cannot be completed (for example, if the disk drive is powered off before the operation completes) the disk drive enters degraded mode. A Format transaction must then be reissued and completed before the disk drive will allow reads and writes.

FN_ISAL_Progress

This transaction is sent to the disk service to determine the progress of a format operation to the resource identified by the handle field. The transaction is rejected with AE_IllegalRequest result if the owning module type of the resource is other than OM_DriverPhysicalDisk. On the PCI SSA 4-Port RAID Adapter, it is rejected with AE_IllegalRequest if the corresponding logical resourceID for this device is currently open.

Minor_function
55

Parameter_DDR
This is a pointer to the following data:

Byte	3	2	1	0
0	Handle			

Handle This identifies the resource for this transaction

Transmit_DDR
Null

Receive_DDR
Null

Status_DDR
This is a pointer to the following data:

Byte	3	2	1	0
0	Percent			

The percent field contains the percentage of a format operation that has been completed, as an unsigned integer from 0 to 99. A value of −1 is returned if a format operation is not in progress.

Result The following result fields can be returned:
AS_Success
Illegal Request (range)
AE_Offline
AE_FencedOut
AE_HardwareError
AE_LogOpen

FN_ISAL_Lock

This transaction is sent to the disk service to reserve exclusively to this client the resource identified by the handle field.

When a resource is locked, the lock is maintained for the following conditions:

- A lock comes into effect when an FN_ISAL_Lock is received for a resource that is not currently locked and a path exists from the client (for example, the device driver) to the resource.
- A lock is removed when:
  - An FN_ISAL_Unlock is executed successfully from the locking client and the path to the resource exists
  - An FN_ISAL_Unlock with the UL_Forced flag on is executed successfully from any client and the path to the resource exists
  - The client either unlocks or closes the resource and the path to the resource is lost. If the resource is held open and locked, the lock is not removed when the path to the resource is lost, but the lock might not be effective.
- The firmware does not attempt to retain the lock if a resource is closed with a lock in place and the path intact and the path is subsequently lost.

- A lock is not effective if the path from the locking client to the resource is broken (regardless of what the client does following loss of the path). Another client can access the resource or place its own lock.
- If a path is reestablished and the lock has not been lost according to the rules above, it is the responsibility of the software immediately above the break in the path to reestablish the lock (assuming that the lock has not been broken by another client).
- The disk is formatted.

A path is lost, that is, communication to the resource is not possible, if the firmware issues a CC_Remove change code if it can. If the resource is held open, the firmware cannot issue a CC_Remove, but the path is still lost. A transient CC_Offline that is followed by a CC_SetOnline (caused, for example, by an SSA network reset) is not a loss of path.

On the PCI SSA 4-Port RAID Adapter, locking of a disk is implemented by sending a Reserve command to the disk drive. This has the effect of locking both the physical resource and the associated logical resource. On the PCI SSA Multi-Initiator/RAID EL Adapter, locking is implemented within the adapter and information is held on the disk for the adapter to which it is currently locked. The following transactions, which are permitted on the PCI SSA 4-Port RAID Adapter, are not permitted on the PCI SSA Multi-Initiator/RAID EL Adapter if they are addressed to a resource of type OM_DriverPhysicalDisk:

- FN_ISAL_Lock
- FN_ISAL_Unlock
- FN_ISAL_Fence
- FN_ISAL_Flush

When the associated logical resource is locked to another initiator, the following transactions, which are rejected on the PCI SSA 4-Port RAID Adapter, are executed on a PCI SSA Multi-Initiator/RAID EL Adapter when addressed to a resource of type OM_DriverPhysicalDisk:

- FN_ISALMgr_HardwareInquiry
- FN_ISALMgr_Characteristics
- FN_ISALMgr_FlashIndicator
- FN_ISAL_Progress

The resource identified by the handle field must be a disk drive, an array, or a Fast-write resource. Otherwise the transaction is rejected with an AE_NonIsal result field.

Minor_function
56

Parameter_DDR
This is a pointer to the following data:

Byte	3	2	1	0
0	Handle			
4	Reserved = 0			Type

Handle This identifies the resource for this transaction

Type This defines the type of lock to be performed:

LT_Normal
Lock resource to the cluster number of the client, if available, or to the adapter ID if it is not.

LT_AdapID
Lock resource to the adapter ID of the client.

Note: On the PCI SSA 4-Port RAID Adapter, the type field is not inspected and a FN_ISAL_Lock transaction causes the resource to be locked to the adapter.

Transmit_DDR
Null

Receive_DDR
Null

Status_DDR
Null

Result The following result fields can be returned:
AS_Success
AE_ReservationConflict
Illegal Request (range)
AE_Offline
AE_FencedOut
AE_HardwareError
AE_Offline
AE_FormatInProgress
AE_FormatDegraded
AE_OfflineTimeout

FN_ISAL_Unlock

This transaction is sent to the disk service to terminate the previous reservation to this client of the resource identified by the handle field. This transaction has no effect on the list of systems fenced out from the resource, even if the flag field is UL_Forced.

This transaction is rejected with AE_IllegalRequest if it is addressed to a resource of type OM_DriverPhysicalDisk. This transaction has no effect on the list of systems fenced out for the resource even when the flag is UL_Forced.

Minor_function
57

Parameter_DDR
This is a pointer to the following data:

Byte	3	2	1	0
0	Handle			
4	Reserved = 0			Flag

Handle This identifies the resource for this transaction.

Flag The following flags control whether the unlock should be unconditional or not:

UL_Normal
The unlock operation is unsuccessful if the resource is already locked to another client.

UL_Forced
The resource is unlocked even if it is locked to another client. This can be implemented by resetting the resource.

Transmit_DDR
Null

Receive_DDR
Null

Status_DDR
Null

Result The following result fields can be returned:
AS_Success
AE_ReservationConflict
Illegal Request (range)
AE_Offline
AE_FencedOut
AE_HardwareError
AE_FormatInProgress
AE_FormatDegraded
AE_OfflineTimeout

FN_ISAL_Test

This transaction is sent to the disk service to test the ability of the resource identified by the handle field to execute transactions. This might involve internal tests being performed by the resource.

Minor_function
58

Parameter_DDR
This is a pointer to the following data:

Byte	3	2	1	0
0	Handle			
4	Reserved = 0			Diagnostic

Handle This identifies the resource for this transaction.

Diagnostic

TT_Test
No internal test is performed in the resource

TT_Diag
Internal tests are performed in the resource

Transmit_DDR
Null

Receive_DDR
Null

Status_DDR
Null

Result The following result fields can be returned:
AS_Success
AE_NotReady
AE_ReservationConflict
AE_HardwareError
Illegal Request (range)
AE_Offline
AE_FencedOut
AE_Offline
AE_FormatDegraded
AE_FormatInProgress
AE_OfflineTimeout

FN_ISAL_Download

This transaction is sent to the disk service to download code to the resource. If the resource ID is not for a resource of owning module type OM_DriverPhysicalDisk, the transaction is rejected with an illegal-request result.

Execution of transactions sent to this physical disk after this transaction are delayed until after the Download transaction has completed.

Minor_function
60

Parameter_DDR
This is a pointer to the following data:

Byte	3	2	1	0
0	Handle			
4	Count			
8	Reserved = 0			Flag

Handle This identifies the resource for this transaction

Count This is the number of bytes of the download

Flag This controls if the downloaded code is saved in nonvolatile storage:

DL_Save
Downloaded code is saved in nonvolatile storage

DL_NoSave
Downloaded code is not saved and will be lost when power is removed from the resource.

Transmit_DDR
This is a pointer to the transmit buffer

Receive_DDR
Null

Status_DDR
Null

Result The following result fields can be returned:
AS_Success
AE_NotReady
AE_ReservationConflict
AE_HardwareError
Illegal Request (range)
AE_Offline
AE_FencedOut
AE_FormatInProgress

FN_ISAL_Fence

This transaction removes or adds initiators to the list of those fenced for the resource identified by the handle field. The list of systems fenced for that resource at the end of the transaction is returned. Only a single RAID adapter can be attached to disk drives in an SSA loop, so this transaction has no use.

Fencing provides a means of preventing access by one or more hosts that are suspected of malfunctioning or should be excluded from access to the resource for other reasons. In a two-initiator network, one processor can exclude the other by using the Lock transaction. With more than two initiators, the Lock transaction cannot be used for this purpose, because it excludes all hosts but one.

When an initiator is fenced out for the resource, the following transactions are rejected with result field AE_FencedOut:

- All transactions that require the resource to be opened before execution, except FN_ISAL_Fence with FF_Force or FN_ISAL_Close.
- ISAL Manager transactions FN_ISALMgr_SetOwningModuleType, and FN_ISALMgr_AssignManualResourceID.

If the host attempts to fence itself out from the resource, the transaction is failed with an illegal-request result field.

When the resource is locked to another initiator, the FN_ISAL_Fence transaction is still executed.

The transaction is rejected with AE_IllegalRequest if it is addressed to a resource of type OM_Driver_PhysicalDisk.

Hosts are identified for fencing by their cluster number which is set in the adapter by a FN_REGY_SetClusterNumber transaction. The cluster number can be in the range 1 through 2048. The adapter defaults to cluster number 0 from when power is turned on to it until it receives the FN_REGY_SetClusterNumber transaction. The maximum number of cluster numbers that can be fenced in or fenced out is 96.

Unless the resource identified by the handle field is a disk drive, an array, or a fast-write resource, it is rejected by an AE_NonIsal result field.

Minor_function
62

Parameter_DDR
This is a pointer to the following data:

Byte	3	2	1	0
0	Handle			
4	Reserved = 0	Force	Count	

Handle This identifies the resource for this transaction.

Force

FF_Normal
If the resource is fenced out from this initiator, the transaction is not executed and is terminated with AE_FencedOut result field.

FF_Force
The transaction is executed even when the resource is fenced out from this initiator. FF_Force can be used to forcibly change the list of initiators fenced out that have been set by an initiator that has failed. It will also cause any reservation for that resource to be released.

Count This is the number of bytes of data that are pointed to by the Transmit_DDR parameter. If the count field is zero, the list of initiators fenced for the resource is returned without any change.

Transmit_DDR

This is a pointer to the following data:

Byte	3	2	1	0
0	Mask Count		Modifier	ListFormat
4	Reserved = 0	Change	Cluster Number (1)	
8	Reserved = 0	Change	Cluster Number (2)	
.				
4n	Reserved = 0	Change	Cluster Number (n)	
4n+4	Mask Cluster Number (1)		Reserved = 0	
4n+8	Mask Cluster Number (3)		Mask Cluster Number (2)	
4n+4+2m	Mask Cluster Number (m)		Mask Cluster Number (m-1)	

ListFormat

This defines whether the list of systems is to be interpreted as systems fenced out or systems fenced in from the resource. If the list-format parameter of the current list of fenced systems is not in the same format as required by this transaction, the list format is changed to the new format and the previous list is deleted.

FL_FenceOut

The system identified by the following cluster-number field is to be added or removed to the list of initiators fenced out for this resource.

FL_FenceIn

The system identified by the following cluster-number field is to be added or removed from the list of fenced initiators **not** fenced out for this resource.

Modifier

FM_Change

The systems identified by the cluster numbers supplied are to be added or removed from the list of fenced clusters.

FM_CompareAndSwap

A mask of cluster numbers is provided in the Transmit_DDR data. Clusters are only removed from or added to the list of fenced clusters when the list of mask cluster numbers matches the list of fenced cluster numbers at the start of the transaction. The cluster numbers in the list of mask cluster numbers must be in ascending order.

Mask Count

This is a count of the number of bytes of Transmit_DDR data used for mask cluster numbers.

Change

This controls if the cluster number is added or removed from the list:

FC_Add
The system identified by the following cluster number field is added to the list of fenced initiators for this resource.

FC_Remove
The system identified by the following cluster number field is removed from the list of fenced initiators for this resource.

Cluster Number
The cluster number identifies the system that is to be added or removed from the list of those fenced out. The cluster number can be in the range 1 through 128.

Receive_DDR
This is a pointer to the following data:

Byte	3	2	1	0
0	Cluster Number (1)		ListFormat	
4	Cluster Number (3)		Cluster Number (2)	
8	Cluster Number (5)		Cluster Number (4)	
.				
2n-2	Cluster Number (n)		Cluster Number (n-1)	

Status_DDR
This is a pointer to a buffer that receives the following data:

Byte	3	2	1	0
0	Count			

Count This identifies the number of bytes in the Receive_DDR area.

Result The following result fields can be returned:
AS_Success
AE_ReservationConflict
AE_FencedOut
AE_Offline
AE_ClusterNumberNotKnown
AE_HardwareError
AE_NotReady
Illegal Request (range)
AE_FormatDegraded
AE_FormatInProgress
AE_OfflineTimeout
AE_NonIsal

If a resource is fenced out and also reserved to another initiator, transactions to that resource are rejected with AE_ReservationConflict result field.

The list of systems fenced for the resource is held in a fence sector in the ISAL reserved area which is mirrored for availability.

Initialization During the initialization process, the fence sector is read and the list analyzed to determine if that adapter is fenced out from the resource. If it is not fenced out, all transactions can be executed to the resource. If it is fenced out, all transactions sent to the handle are rejected with result field AE_FencedOut. However, after initialization, the system could have been fenced in by another system, without being informed by means of a unit-attention condition report because it was not sending commands to that resource. Therefore, when initialization has found this system is fenced out, from the resource, any future transactions to the handle cause first a Test Unit Ready command to be sent to the resource. If this receives check status for a mode-parameters-changed unit-attention condition, the resource is reinitialized and the transaction can be executed if the system is no longer fenced out. If it is still fenced out, or the Test Unit Ready transaction executed without error, the transaction is terminated with AE_FencedOut result field.

FN_ISAL_SCSI

This transaction is sent to the disk service to issue a raw SCSI command to the resource identified by the handle field. Unlike all other transactions, the only recovery performed by the adapter is for SSA link errors.

Only one SCSI transaction can be accepted at a time for a given resource. The resource must have been opened in SCSI pass-through mode for this transaction to be executed. If the resource ID is not for a resource of owning module type OM_DriverPhysicalDisk, the transaction is rejected with an illegal-request result.

SSA-SCSI linked commands should not be used. The link bit in the command descripter block must be 0.

Minor_function
59

Parameter_DDR
This is a pointer to the following data:

Byte	3	2	1	0
0	Handle			
4	Reserved = 0			Identifier
8 through n+3	Command Descriptor Block			

Handle This identifies the resource for this transaction. The physical resource identified can contain several logical units (LUNs).

Identifier
This field identifies the SSA_SCSI logical unit number to which the resource manager should send the SCSI command. The format of this field is as defined for SSA_SCSI:
- If bit 7 is 1, the field identifies the target routine
- Bits 6 through 0 identify the logical-unit routine.

Command Descriptor Block
This is as defined for SCSI and can be 6, 10, or 12 bytes.

Transmit_DDR
This is null or a pointer to any data or parameters to be sent to the device.

Receive_DDR
This is null or a pointer to a buffer for any data received from the device.

The Transmit_DDR and Receive_DDR fields cannot both be non-zero. These are used by the resource manager to determine the direction of data transfer.

Status_DDR
This is a pointer to a buffer that receives the following data when the result field is AE_SCSIError:

Byte	3	2	1	0
0	Reserved = 0			SCSI Status

SCSI Status
This is the status byte as defined in SSA_SCSI that is returned by the resource. It is always non-zero. If zero status (good) is returned in the SCSI_status SSA message, the result field is AS_Success and no data is sent to the buffer pointed to by the Status_DDR field.

Result The following result fields can be returned:
AS_Success
Illegal Request (range)
AE_SCSIError
AE_Offline
AE_FencedOut
AE_ReservationConflict
AE_NotSupported′

FN_ISAL_Flush

This transaction requests the service to flush any data or metadata for the resource so that the components of the resource are synchronized. This allows the adapter to be changed or adapters and disks to be powered off without the risks of having to rebuild arrays or losing data because of the removal of power. When the components are synchronized, any NVRAM contents are not critical.

The transaction causes the service to write any data and metadata required to its components to get them synchronized. The service then issues an FN_ISAL_Flush to each of its components. Services that do not have anything to flush, for example, the base disk service, return AS_Success. When all the components have returned AS_Success to each FN_ISAL_Flush, AS_Success is returned to this transaction.

After the array or filter is synchronized, it might remain synchronized for an indeterminate period, for example, until the next FN_ISAL_Write is received, or while an FN_ISAL_Write is not yet complete.

On a Micro Channel SSA Multi-Initiator/RAID EL Adapter, this transaction is rejected with AE_IllegalRequest if it is addressed to a resource of type OM_DriverPhysicalDisk.

Minor_function
68

Parameter_DDR
This is a pointer to a the following data:

Byte	3	2	1	0
0	Handle			

Handle This identifies the resource for this transaction.

Transmit_DDR
Null

Receive_DDR
Null

Status_DDR
None

Result The following result fields can be returned:
AS_Success

AE_Failure
(Flush of resource failed)
AE_FlushComponentFailure
(Flush of component failed)
AE_Offline

Illegal Request (range)

AE_OfflineTimeout

AE_UnknownFunction

Adapter Service

The adapter service can be used for adapter-only transactions and for transactions to be issued to other adapters. The service number is fixed for the adapter service. The adapter service supports the following transactions:

Table 38. Adapter Transactions

Transaction	Minor_function
FN_ADAP_TransferFromHost	80
FN_ADAP_TargetTransfer	83
FN_ADAP_TransferToHost	81

Table 38. Adapter Transactions (continued)

Transaction	Minor_function
FN_ADAP_ConnectForHostTransfer	84
FN_ADAP_DisconnectForHostTransfer	85
FN_ADAP_GetClusterNumber	86
FN_ADAP_AdapterHealthCheck	90
FN_ADAP_ListSSANodes	91
FN_ADAP_QueryNodes	93
FN_ADAP_GetAdapterUID	94
FN_ADAP_SetTime	95
FN_ADAP_SetMasterPriority	96
FN_ADAP_GetMasterPriority	97
FN_ADAP_GetSupportLevel	98

FN_ADAP_TransferFromHost

This transaction is sent to the adapter service to request that data is sent to one or more systems. When the cluster-number field is FFFFh, the data is sent to the adapter service on all other nodes connected using FN_ADAP_TargetTransfer transactions. When the cluster number is any other value, a single FN_ADAP_TargetTransfer transaction is sent to the adapter service on that cluster.

If the type is TT_VSC, the service returns successful completion to the FN_ADAP_TransferFromHost transaction when it has received a completion (successful or unsuccessful) to all the FN_ADAP_TargetTransfer transactions it has sent or timed out waiting for a completion. If the completion is unsuccessful, the device driver of the other system will also have been monitoring its adapter and will have detected an error. If the type is TT_DataTransfer and the Cluster Number is not FFFFh, the application result either indicates successful completion, or why no FN_ADAP_TargetTransfer transaction could be sent or the application result returned by that transaction. If the cluster number is FFFFh for a TT_DataTransfer transaction type, the service returns successful completion to the FN_ADAP_TransferFromHost transaction when it has received a completion (successful or unsuccessful) to all the FN_ADAP_TargetTransfer transactions it has sent or timed out waiting for a completion.

Minor_function
80

Parameter_DDR
This is a pointer to the following data:

Byte	3	2	1	0
0	Cluster Number		Reserved = 0	Type
4	Count			
8 through n	Parameter Data			

Cluster Number
This identifies the cluster to which data should be sent. If the value is FFFFh, data is sent to all clusters attached. The cluster number must be FFFFh if the type is TT_VSC.

If the cluster number specified has multiple adapters, the adapter service chooses to which adapter it sends the FN_ADAP_TargetTransfer transaction.

Type This can be the following:

TT_VSC
Volume-status-change (VSC) type requires that the data is transmitted to all attached clusters. This can be used to inform other systems of a change in status of a particular resource.

TT_DataTransfer
Data is to be transmitted to the specified cluster (or all clusters if the cluster number is FFFFh). This can be used to send data between systems (target mode).

TT_CNUM
This indicates that the data transmitted consists of the cluster number. This is transmitted to all other clusters when the cluster-number field is FFFFh.

Count This defines the number of bytes of data to be sent. The maximum value is 512. The location of this data is pointed to by the transmit_DDR. The count field must be an even number.

Parameter Data
If the type is TT_VSC, parameter data is a 16–byte field that includes the 15–byte ASCII serial number of the resource to which the broadcast data refers. If the type is TT_DataTransfer, this field contains miscellaneous fields including a definition of the originating cluster.

Transmit_DDR
This is a pointer to the data in the buffer that is to be transmitted. The maximum Data_length is 512 bytes.

Receive_DDR
Null

Status_DDR
Null

Result The following result fields can be returned:
AS_Success

Illegal Request (range)

AE_MissingCluster

AE_RoutingError

AE_RemoteTimeout

AE_TargetNot Available

AE_TargetReceiverFull

AE_TargetTransferTooLarge

AE_Failure

FN_ADAP_TargetTransfer

This transaction is sent to the adapter service on the node identified in the cluster-number field. If the cluster number is FFFFh, the transaction is sent to the adapter service on all nodes that are attached. The data transmitted is that identified in the earlier FN_ADAP_TransferFromHost transaction. The receiving service sends an FN_ADAP_TransferToHost transaction to the host service identified by a previous FN_ADAP_ConnectForHostTransfer transaction.

If the type is TT_VSC, the receiving service returns a successful-completion result to the FN_ADAP_TarhgetTransfer transaction when it has received a completion to the FN_ADAP_TransferToHost transactions it has sent (successful or unsuccessful) or if no host node has connected for host transfers. If this receiving service is not able to complete the FN_ADAP_TransferToHost transaction successfully, the adapter continually presents an error status of SS_VSC. The host device driver must reset the adapter card to recover from this situation.

If the type is TT_DataTransfer, the result returned indicates the success or otherwise of executing the transaction.

Minor_function
83

Parameter_DDR
This is a pointer to the following data:

Byte	3	2	1	0
0	Cluster Number		Reserved = 0	Type
4	Count			
8 through n	Parameter Data			

Cluster Number
This identifies the cluster to which data should be sent. If the value is FFFFh, data is sent to all clusters attached. The cluster number must be FFFFh if the type is TT_VSC.

If the cluster number specified has multiple adapters, the adapter service chooses to which adapter it sends the FN_ADAP_TargetTransfer transaction.

Type This can be the following:

TT_VSC
Volume-status-change (VSC) type requires that the data is transmitted to all attached clusters

TT_DataTransfer
Data is to be transmitted to the specified cluster (or all clusters if the cluster number is FFFFh).

TT_CNUM
This indicates that the data transmitted consists of the cluster number. This is transmitted to all other clusters when the cluster-number field is FFFFh.

Count This defines the number of bytes of data to be sent. The maximum value is 512. The location of this data is pointed to by the transmit_DDR. The count field must be an even number.

Parameter Data
If the type is TT_VSC, parameter data is a 16–byte field that includes the 15–byte ASCII serial number of the resource to which the broadcast data refers. If the type is TT_DataTransfer, this field contains miscellaneous fields including a definition of the originating cluster.

Transmit_DDR
This is a pointer to the data that is to be transmitted. The maximum Data_length is 512 bytes.

Receive_DDR
Null

Status_DDR
Null

Result The following result fields can be returned:
AS_Success

Illegal Request (range)

AE_Failure

AE_TargetNot Available

AE_TargetReceiverFull

AE_TargetTransferTooLarge

FN_ADAP_TransferToHost

This transaction is sent to the host service identified by a previous FN_ADAP_ConnectForHostTransfer transaction. The data transmitted is that identified in the earlier FN_ADAP_TargetTransfer transaction. If the type field is TT_VSC and, after a timeout period, no completion is received from the host service, good-completion result is returned to the FN_ADAP_TargetTransfer transaction and the adapter continually presents an error status of SS_VSC. The host device driver must reset the adapter card to recover from this situation.

Minor_function
81

Parameter_DDR
This is a pointer to the following data:

Byte	3	2	1	0
0	Cluster Number		Reserved = 0	Type
4	Count			
8 through n	Parameter Data			

Cluster Number
This identifies the cluster to which data should be sent. If the value is FFFFh, data is sent to all clusters attached. The cluster number must be FFFFh if the type is TT_VSC.

If the cluster number specified has multiple adapters, the adapter service chooses to which adapter it sends the FN_ADAP_TargetTransfer transaction.

Type This can be the following:

TT_VSC
Volume-status-change (VSC) type requires that the data is transmitted to all attached clusters

TT_DataTransfer
Data is to be transmitted to the specified cluster (or all clusters if the cluster number is FFFFh).

TT_CNUM
This indicates that the data transmitted consists of the cluster number. This is transmitted to all other clusters when the cluster-number field is FFFFh.

Count This defines the number of bytes of data to be sent. The maximum value is 512. The location of this data is pointed to by the transmit_DDR. The count field must be an even number.

Parameter Data
If the type is TT_VSC, parameter data is a 16–byte field that includes the 15–byte ASCII serial number of the resource to which the

broadcast data refers. If the type is TT_DataTransfer, this field contains miscellaneous fields including a definition of the originating cluster.

Transmit_DDR
This is a pointer to the data that is to be transmitted.

Receive_DDR
Null

Status_DDR
Null

Result The following result fields can be returned:
AS_Success

Illegal Request (range)

AE_Failure

AE_TargetNot Available

AE_TargetReceiverFull

AE_TargetTransferTooLarge

FN_ADAP_ConnectForHostTransfer

This transaction informs the adapter service of the service number in the host node that is able to receive FN_ADAP_TransferToHost transactions. Only a single host node can be connected at a time for each type of target transfer data.

Minor_function
84

Parameter_DDR
This is a pointer to the following data:

Byte	3	2	1	0
0	Node			
4	Service			
8	Timeout		Cluster Number	
12	Reserved = 0			Type

Node This is the IPN node that can receive the data.

Service
This is the service on that node that can receive the data for the specified type of target transfer.

Timeout
This is the time in seconds that the adapter service should use to timeout FN_ADAP_TransferToHost transactions that are sent to the host.

Cluster Number
This is the cluster number of the node.

Type This defines the type of target transfer transactions for which the host node is connecting. It can be:
TT_VSC
TT_DataTransfer
TT_CNUM

Note: If the type field is not included in the Parameter_DDR data (parameter length 12 bytes), the adapter assumes that the host node is connecting for TT_VSC type of transfers.

Transmit_DDR
Null

Receive_DDR
Null

Status_DDR
Null

Result The following result fields can be returned:
AS_Success

Illegal Request (range)

FN_ADAP_DiscForHostTransfer

This transaction informs the adapter service of the service number in the host node that is no longer able to receive FN_ADAP_TransferToHost transactions of the type specified.

Minor_function
85

Parameter_DDR
This is a pointer to the following data:

Byte	3	2	1	0
0	Node			
4	Service			
8	Reserved = 0		Cluster Number	
12	Reserved = 0			Type

Node This is the IPN node that can no longer receive the data.

Service
This is the service on that node that can no longer receive the data for the specified type of target transfer.

Cluster Number
This is the cluster number of the node.

Type This defines the type of target transfer transactions for which the host node is connecting. It can be:
TT_VSC
TT_DataTransfer
TT_CNUM

Note: If the type field is not included in the Parameter_DDR data (parameter length 12 bytes), the adapter assumes that the host node is connecting for TT_VSC type of transfers.

Transmit_DDR
Null

Receive_DDR
Null

Status_DDR
Null

Result The following result fields can be returned:
AS_Success

Illegal Request (range)

FN_ADAP_GetClusterNumber

This transaction is sent from an adapter service on one node to another node to obtain the cluster number of that node.

Minor_function
86

Parameter_DDR
Null

Transmit_DDR
Null

Receive_DDR
Null

Status_DDR
This is a pointer to a buffer that receives the following data:

Byte	3	2	1	0
12	Reserved = 0		Cluster Number	

Cluster Number
This field contains the cluster number of the node that has been set by the FN_REGY_SetClusterNumber transaction. If this has not yet been set, a cluster number of 0 is returned (0 means undefined). The cluster number can be in the range 1 through 127.

Result The following result fields can be returned:
AS_Success

Illegal Request (range)

FN_ADAP_AdapterHealthCheck

This transaction is sent to the adapter to report any adapter errors. Only degraded type errors can be reported in the Status_DDR data.

If the adapter has knowledge of multiple degraded errors that could be reported, only the lowest adapter error code is reported.

Minor_function
90

Parameter_DDR
Null

Transmit_DDR
Null

Receive_DDR
Null

Status_DDR
This is a pointer to a buffer that receives the following data:

Byte	3	2	1	0
12	Reserved = 0	Adapter Error Code		

Adapter Error Code
This field contains the adapter error code of 6 hexadecimal characters. These codes are the same as are reported in adapter sense data when logging an error to the error logger. Byte 0 is the most-significant byte of the error code.

Result The following result fields can be returned:
AS_Success

Illegal Request (range)

FN_ADAP_ListSSANodes

This transaction is sent to the adapter to report details of all nodes on the SSA network specified that have been configured. If a network change is in progress, the transaction is terminated with a result field AE_InConfig and no other information is returned.

The list of nodes is ordered according to network topology. The list starts with the node nearest port 1 and ends if it is a loop with the node closest to port 2. If the SSA network is a string, the list is ordered by the list of nodes starting from port 1 to the end of that part of the string followed by the list of nodes starting from port 2 to the end of that part of the string.

The nodes reported are those configured at the time of the last valid SSA configuration. This may not be exactly consistent with the adapters known to all services, for example, adapter service for target mode, at the time of the transaction. If an illegal SSA network is detected during a SSA configuration by the adapter, the configuration held in the adapter that is reported by this transaction is not updated from the valid configuration.

Minor_function
91

Parameter_DDR
This is a pointer to a buffer that contains the following data:

Byte	3	2	1	0
0	Reserved = 0	NetworkId		

NetworkID
This identifies the SSA interface on the adapter to which the SSA loop or string is attached. It can be one of the following:
NI_NetworkA
NI_NetworkB

Transmit_DDR
Null

Receive_DDR
This is a pointer to a buffer that receives the following data:

Byte	3	2	1	0
0 through 4	Adapter ID or device SSA UID			
8	SSA Path 1			
12	SSA Path 2			
16	Total Other Ports	Port on Path 1	Port on Path 2	Remote NetworkID
20 through n	Repeat for each adapter			

AdapterID or Device SSA UID
This 8-byte binary number uniquely identifies the node. If the node is an adapter (identified by a remote networkID of NI_NetworkA or

NI_NetworkB), it is the adapterID. This is the same as the SSA UID of the modules that control the SSA links with the least significant bit (bit 0) = 0; bytes 6 and 7 are 00h.

If the node is a device (identified by a Remote NetworkID of NI_Device), the 8-byte number is the device SSA UID of the node; bytes 6 and 7 are 00h.

SSA Path 1

This 4-byte field is the path component of the SSA address field from port 1 to the node. As switches are not supported, byte 11 is the only byte used for the SSA address field; bytes 8 through 10 are zero. Bit 7 of byte 11 is zero to indicate that there are no other bytes in the path component.

If this remote node is not attached to port 1 of the adapter, FFFFFFFFh is returned.

SSA Path 2

This 4-byte field is the path component of the SSA address field from port 2 to the node. As switches are not supported, byte 15 is the only byte used for the SSA address field; bytes 12 through 14 are zero. Bit 7 of byte 15 is zero to indicate that there are no other bytes in the path component.

If this remote node is not attached to port 2 of the adapter, FFFFFFFFh is returned.

Total Other Ports

This contains a value that is one less than the total number of ports implemented on the node. This is normally 1b as all nodes supported have two ports.

Port on Path 1

This identifies the number of the port on the remote node that is linked to port 1 of this adapter that is either:

PI_Port1
Port 1 of the SSA chip is being used

PI_Port2
Port 2 of the SSA chip is being used

If this remote node is not attached to port 1 of the adapter, FFh is returned.

Port on Path 2

This identifies the number of the port on the remote node that is linked to port 2 of this adapter that is either:

PI_Port1
Port 1 of the SSA chip is being used

PI_Port2
Port 2 of the SSA chip is being used

If this remote node is not attached to port 2 of the adapter, FFh is returned.

Remote NetworkID
This identifies the SSA chip on the remote node to which this SSA loop or string is attached when the remote node is an adapter or that the remote node contains a single SSA chip when the node is a device. It can be one of the following:

NI_NetworkA
Connected to SSA chip A on node

NI_NetworkB
Connected to SSA chip B on node

NI_Device
Single SSA chip on node

Status_DDR
This is a pointer to a buffer that receives the following data:

Byte	3	2	1	0
0	Number of Entries			
4	Reserved = 0		LoopFlag	Reserved = 0
8	ChangeCount			

Number of Entries
This is the number of nodes configured on the SSA loop or string; identification and port information of each of these is returned in the Receive_DDR data. It does not include the current adapter, so a value of zero is returned if the adapter is the only node on the SSA network.

LoopFlag
This indicates whether the SSA network is a loop, a string, or an invalid configuration:

SC_Loop
SSA loop configuration

SC_String
SSA string configuration

SC_IllegalString
Invalid SSA configuration (either a loop or a string)

ChangeCount
This indicates the number of times the configuration information has changed from power on. It can be used to determine if there have been any changes since the last transaction was used.

Result The following result fields can be returned:
AS_Success

Illegal Request (range)

AE_InConfig

FN_ADAP_QueryNodes

This transaction can be used to determine what devices and adapters are on an SSA network. This might be useful when the adapter determines that the network is illegal, and does not configure new devices or adapters. Other transactions report only details of a legal network or parts of an illegal network that existed before it became illegal.

If a network change is in progress, the transaction is terminated with a result field AE_InConfig and no other information is returned.

Minor_function
93

Parameter_DDR
This is a pointer to a buffer that contains the following data:

Byte	3	2	1	0
0	Reserved = 0	Query Adapter	Port	NetworkID
4	Path			

NetworkID
This identifies the SSA interface on the adapter to which the SSA loop or string is attached. It can be one of the following:

NI_NetworkA

NI_NetworkB

Port This identifies the port on the SSA interface from which the Query_node SSA message should be sent. It can be:

PI_Port1
Port 1 of the SSA interface to be used

PI_Port2
Port 2 of the SSA interface to be used

Query Adapter
When this field is 1, the status data refers to this adapter and not to the node identified by the path field. The path field is only used to identify the node when the query-adapter field is 0.

Path This 4-byte field is the path component of the SSA address field from the adapter to the device. Because there can be no switches, byte 7 is the only byte that is used for the SSA address field; bytes 4-6 are zero. Bit 7 of byte 7 is zero to indicate that there are no other bytes in the path component. If switches are supported in the future, the path component could contain up to four byte fields.

Transmit_DDR
Null

Receive_DDR
Null

Status_DDR
This is a pointer to a buffer that receives the following data that is part of the Query_node_reply data that is returned by the target SSA node:

Byte	3	2	1	0
0	Valid Mask			
4	ChangeCode			
8	Version	ULP	Total Other Ports	Current Port
12	Reserved = 0			
16 through 20	Unique_ID			
24	Reserved = 0			Port mask
28	Reserved = 0			

Valid Mask
This identifies which bytes of the Status field are valid. Bit 0 of byte 0 = 1b shows that byte 0 is valid; bit 7 of byte 3 = 1b shows that byte 31 is valid.

ChangeCode
This indicates how many times the configuration information has changed since power-on time. ChangeCode can be used to determine whether any changes have occurred since this transaction was previously used.

Current Port
This identifies the number of the port on the destination node that is now being used. Current Port can have one of the following values:

Pl_Port1
Port 1 of the SSA chip is being used

Pl_Port2
Port 2 of the SSA chip is being used

Total Other Ports
This contains a value that is one less than the total number of ports implemented. This will normally be 1b as devices supported all have two ports.

ULP The Upper Level protocol identifies the SSA upper-level protocol that the node supports. This can be:

80 SSA-IA/95SP (level supported by current disk drives)

FC The PCI SSA Multi-Initiator/RAID EL Adapter

Version

This identifies the version of the SSA transport layer that is supported by the adapter that is sending the message to the node. It can be:

01h SSA-1A/95PH

The value to be used for extensions for 40 MB/s SSA support is to be determined.

Master Priority

Bits 6-4 define the priority of the node for becoming the Master of the SSA web. A value of zero indicates that this node is not capable of becoming a Master (that is, it is not an adapter).

Bit 7 is used in SSA Query_node_reply to indicate that the node has no space to add an entry for the adapter that sent the Query_node. This is 0b for this transaction because the Query_node sent has the Disable Registration bit = 0; the node does not, therefore, add the sender of the Query_node to its adapter or configuration table.

Bits 0-3 are reserved = 0.

Unique_ID

This 8-byte binary number uniquely identifies the node. It consists of two reserved bytes in bytes 22-23 (containing zeroes) followed by a 6-byte IEEE Universal Address.

Port Mask

This field is a bit vector to indicate the port number of those ports operational on the target node. Bit 7 = 1b indicates that port 1 is operational; Bit 6 = 1 indicates that port 2 is operational. It can therefore be determined whether the other port of a 2-port node is operational and has another node attached to it. This can be done by inspecting the other port number in the Port Mask field from the port number on which the message was received as identified in the Current Port field.

Result The following result fields can be returned:

AS_Success

Illegal Request (range)

AE_UnknownFunction

AE_InConfig

AE_QNTimedOut

FN_ADAP_GetAdapterUID

This transaction returns the SSA UID of the specified SSA interface on the adapter. The SSA UID of both SSA interfaces on the adapter are identical except for the least significant bit (bit 0).

Minor_function
94

Parameter_DDR
This is a pointer to a buffer that contains the following data:

Byte	3	2	1	0
0	Reserved = 0	NetworkId		

NetworkID
This identifies the SSA interface on the adapter for which the SSA UID is requested. It can be one of the following:

NI_NetworkA

NI_NetworkB

Transmit_DDR
Null

Receive_DDR
Null

Status_DDR
This is a pointer to a buffer that receives the following data:

Byte	3	2	1	0
0 through 4	SSA UID			

SSA UID
This 8-byte binary number uniquely identifies the adapter that is using this SSA interface. Bytes 6 and 7 are 00h.

Result The following result fields can be returned:
AS_Success

Illegal Request (range)

FN_ADAP_SetTime

This transaction is used to set a time value that is held in the adapter. It is used in the adapter to identify the value of the adapter internal timer at that time. The time value cannot be read by the host, but can be part of data that is saved during a dump to identify the times of the traces.

Minor_function
95

Parameter_DDR
This is a pointer to a buffer that contains the following data:

Byte	3	2	1	0
0	TimeInSeconds			

TimeInSeconds
This is the time, in seconds, since EPOCH.

TimeInMilliseconds
This is the time, in milliseconds, within the second that was identified in the TimeInSeconds field.

Transmit_DDR
Null

Receive_DDR
Null

Status_DDR
Null

Result The following result fields can be returned:
AS_Success

Illegal Request (range)

FN_ADAP_SetMasterPriority

This transaction is used to set the SSA adapter master priority. It is used during SSA configuration to determine which adapter is to become the SSA master. The SSA master is the adapter that has the highest UID for the adapter with the highest master priority.

Minor_function
96

Parameter_DDR
This is a pointer to a buffer that contains the following data:

Byte	3	2	1	0
0	Reserved = 0		Master Priority	NetworkID

NetworkID
This identifies the SSA interface on the adapter whose Master Priority is to be set. It can be one of the following:

NI_NetworkA

NI_NetworkB

Master Priority
This is used to set the Master Priority of the SSA interface specified. It can have values 0 through 7 decimal. The default value of Master

Priority is 4. It is recommended that the Master Priority be set to a value of 5 for an adapter that is required to be the SSA master.

Transmit_DDR
Null

Receive_DDR
Null

Status_DDR
Null

Result The following result fields can be returned:
AS_Success

AE_UnknownFunction

Illegal Request (range)

FN_ADAP_GetMasterPriority

This transaction is used to fetch the SSA master priority. It is used during SSA configuration to determine which adapter is to become the SSA master. The SSA master is the adapter that has the highest UID for the adapter with the highest master priority

Minor_function
97

Parameter_DDR
This is a pointer to a buffer that contains the following data:

Byte	3	2	1	0
0	Reserved = 0	NetworkId		

NetworkID
This identifies the SSA interface on the adapter whose Master Priority is to be fetched. It can be one of the following:

NI_NetworkA

NI_NetworkB

Transmit_DDR
Null

Receive_DDR
Null

Status_DDR
This is a pointer to a buffer that receives the following data:

Byte	3	2	1	0
0	Reserved = 0	Master Priority		

Master Priority
This is used to set the Master Priority of the SSA interface specified. It can have values 0 through 7 decimal. The default value of Master Priority is 4.

Result The following result fields can be returned:
AS_Success

Illegal Request (range)

FN_ADAP_GetSupportLevel

This transaction is sent from an adapter service on one node to obtain the support level of that adapter. This can be used in conjunction with the LL field in the VPD data and the device ID field to precisely identify the card and its firmware function.

Minor_function

Parameter_DDR
Null

Transmit_DDR
Null

Receive_DDR
Null

Status_DDR
This is a pointer to a buffer which will receive the following data:

Byte	3	2	1	0
0	Reserved = 0			Support Level

Support Level
This field contains the support level of the adapter and identifies the firmware on the card. the support levels are:

02 SSA Multi-Initiator/EL adapter

03 Reserved

Result The following result fields can be returned:
AS_Success

AE_UnknownFunction
not supported on adapters prior to SSA Multi-Initiator/EL adapter with code at level 50 or later

Array-Configuration Service

The array-configuration service uses the IPN array configuration language (IACL) to define the configuration of array filters to be used in the adapter. In these transactions, Parameter_DDR and Status_DDR are used, but Transmit_DDR and Receive_DDR are not.

The array-configuration service handles the following transactions:

Table 39. Array-Configuration Transactions

Transaction	Minor_function
FN_IACL_Register	102
FN_IACL_Unregister	103
FN_IACL_Command	101

FN_IACL_Register

This transaction is issued by a filter service to declare to the array-configuration service that the filter exists. This must be sent before any configuration transactions can be issued to the array-configuration service.

Minor Function
102

Parameter_DDR
This is a pointer to the following data:

Byte	3	2	1	0
0	Service			
4	Reserved = 0			Filter Type

Service
This is the service number of the registering filter

Filter Type
This is the filter type of the registering filter

Status_DDR
Null

Result The following result fields can be returned:
AS_Success
Illegal Request (range)

FN_IACL_Unregister

This transaction is issued by a filter service to declare to the array-configuration service that no more transactions should be sent to this filter.

Minor Function
103

Parameter_DDR
This is a pointer to the following data:

Byte	3	2	1	0
0	Reserved = 0			Filter Type

Filter Type
This is the filter type of the registered filter

Status_DDR
Null

Result The following result fields can be returned:
AS_Success
Illegal Request (range)

FN_IACL_Command

In this transaction, the real function is defined in the first word of the parameter DDR. The functions are defined on pages 179 through 218.

Minor Function
101

Parameter_DDR
This is a pointer to data that has the following format:

Byte	3	2	1	0
0	Reserved = 0	Filter Type	Function	
4 through n	Command Parameters			

Function
This specifies the one of the following functions:

Code	Function
1	FC_IACLVersion
2	FC_ResrcCount
3	FC_ResrcList
4	FC_ResrcView
5	FC_CandidateCount
6	FC_CandidateList
7	FC_ResrcCreate
8	FC_ResrcDelete
9	FC_ResrcRename
10	FC_ComponentView
11	FC_ComponentExchange

12 FC_QueryMetaResrcParams

13 FC_ModifyResrcParams

14 FC_FlashIndicator

15 FC_VPDInquiry

16 FC_VPDHardwareInquiry

17 FC_CompExchCount

18 FC_CompExchCandList

19 FC_AdapterVPD

20 FC_SyncHealth

21 FC_Wrap

22 FC_Unwrap

23 FC_UnwrapAll

24 FC_Test

25 FC_Format

26 FC_Certify

27 FC_Read

28 FC_Write

29 FC_AdapterSN

30 FC_CacheFormat

Filter Type

This identifies the filter that is being configured, for both arrays and disks that are not in arrays. The valid filter types are:

- FT_DriverAutomaticDisk *
- FT_DriverManualDisk *
- FT_Raid5Filter
- FT_FastWriteFilter
- FT_PhysicalDisk *
- FT_NotOwned *
- FT_HotSpare *
- FT_BlankReserved *
- FT_Disowned *

The filter types marked with a * are not filters but represent resources that are either unowned by or are logically attached to the system device driver. These filter types are referred to as *pseudofilters*. The other filter types are referred to as *real filters*.

The filter type FT_DriverAutomaticDisk supports only the following commands:

- FC_IACLVersion
- FC_ResrcCount

- FC_ResrcList
- FC_ResrcView
- FC_ResrcCreate
- FC_ResrcDelete
- FC_CandidateCount
- FC_CandidateList

Status_DDR

All FN_IACL_Command transactions return Status_DDR data. The format of this data is:

Byte	3	2	1	0
0	Unused			
4	Length			
8 through n	Command Result			

The length field is the byte count of Status_DDR data that follows this field.

FC_IACLVersion

This function returns the version number of the IACL language. This allows the array-configuration service to validate that the IACL level supported by the adapter card (array-configuration service and the RAID Filters) is correct. It also allows the array-configuration service to determine which filter types are present on the adapter. The array-configuration service returns AE_NotInTable for filters not present.

Parameter_DDR

This is a pointer to the following data:

Byte	3	2	1	0
0	Reserved = 0	Filter Type	Function = 1	

Function

This is the function code, 1, for FC_IACLVersion

Filter Type

This is the filter type to which the function is directed.

Status_DDR

This is a pointer to the following data:

Byte	3	2	1	0
0	Unused			
4	Length = 4			
8	Version			

Length This is 4, showing that 4 bytes of data follow this field

Version
This is a 32-bit unsigned integer that identifies the code level of the filter. A value of zero is returned if the filter is not present.

Result The following result fields can be returned:
AS_Success
AE_Failure
Illegal Request (range)

FC_ResrcCount

This function returns the number of resources that a particular filter has created by earlier FC_ResrcCreate functions.

Parameter_DDR
This is a pointer to the following data:

Byte	3	2	1	0
0	Reserved = 0	Filter Type	Function = 2	

Function
This is the function code, 2, for FC_ResrcCount

Filter Type
This is the filter type to which the function is directed

Status_DDR
This is a pointer to the following data:

Byte	3	2	1	0
0	Unused			
4	Length = 4			
8	Resource Count			

Length This is 4, showing that 4 bytes of data follow this field

Resource Count
This is a 32 bit unsigned integer that identifies the number of resources created for this filter.

Result The following result fields can be returned:
AS_Success
AE_Failure
Illegal Request (range)

FC_ResrcList

This function requests a list of resources for the specified filter and their status. The selection of resource names (serial numbers) that are required is identified in the parameter data.

Parameter_DDR
This is a pointer to the following data:

Byte	3	2	1	0
0	Reserved = 0	Filter Type	Function = 3	
4	First Resource Number (n)			
8	Requested Count (m)			

Function
This is the function code, 3, for FC_RescrList

Filter Type
This is the filter type to which the function is directed

First Resource Number (n)
This is the ordinal number of the first resource (starting with zero) that is reported in the Status_DDR data.

Requested Count (m)
This is the number of resources from the first resource number that are to be reported.

Status_DDR
This is a pointer to the following data:

Byte	3	2	1	0
0	Unused			
4	Length			
8 through 19	Serial Number (n)			
20	Reserved = 0	Serial Number (n)		
24	Reserved = 0		Percent	Status
28 through 39	Serial Number (n+1)			
40	Reserved = 0	Serial Number (n+1)		
44	Reserved = 0		Percent	Status
.				
20m-12 through 20m-1	Serial Number (n+m)			
20m	Reserved = 0	Serial Number (n+m)		
20m+4	Reserved = 0		Percent	Status

Length The identifies the number of bytes that follow this field (320 maximum)

Serial Number
This 15-character ASCII string is the name of the array.

Status This can be one of the following:

FS_ResrcOffline
If this is a pseudofilter, this status indicates the resource is in

the RS_Offline state defined on “FN_REGY_ResrcChangeToRegistry” on page 103. If it is a real filter, this status indicates that the array does not have enough components to function or it contains inconsistent components.

FS_ResrcOnline
This is only returned for a pseudofilter. It indicates the resource is in the RS_Online state defined in “FN_REGY_ResrcChangeToRegistry” on page 103.

FS_ResrcOnlineNonDeg
The array is not degraded and is fully operational.

FS_ResrcOnlineDeg
The array is degraded.

FS_ResrcOnlineRebuild
The missing component has been returned to a degraded array which is in the process of rebuilding.

FS_ResrcOnlineExposed
A component is missing from an array and no write operations have yet been required to that component.

FS_ResrcUnknown
This is the state that an array is in until N-1 components are visible for the first time.

FS_ResrcWrapped
The physical resource is wrapped

FS_RescrFormatting
The physical resource is being formatted: the percent field reports the amount currently formatted.

FS_ResrcCertifying
The physical resource is being certified: the percent field reports the amount currently certified.

FS_ResrcFormatFailed
Formatting the disk has failed; the percent field reports how much of the disk was formatted before the failure.

FS_ResrcCertifyFailed
Certifying the disk has failed; the percent field reports how much of the disk was certified before the failure.

FS_ResrcInUse
This is only reported for an NVRAM resource. It indicates that the defined resource is associated with a known array.

FS_ResrcDormant
This is only reported for an NVRAM resource. It indicates that the defined resource is not associated with any known array.

Percent

This is an integer in the range 0 through 99 indicating the percentage completion of an operation for the following fields:

FS_ResrcRebuild
FS_ResrcFormatting
FS_ResrcCertifying

Result The following result fields can be returned:

AS_Success
AE_Failure
Illegal Request (range)

FC_ResrcView

This function is used to examine one resource of a filter in more detail. The resource name is sent in the parameter_DDR data. Details of the resource characteristics and status are returned in the status_DDR data.

Parameter_DDR

This is a pointer to the following data:

Byte	3	2	1	0
0	Reserved = 0	Filter Type	Function = 4	
4 through 15	Serial Number			
16	Reserved = 0			

Function

This is the function code, 4, for FC_ResrcView

Filter Type

This is the filter type to which the function is directed

Serial Number

This 15-character ASCII string is the name of the resource.

Status_DDR

This is a pointer to the following data:

Byte	3	2	1	0
0	Unused			
4	Length = 44			
8	Component Count			
12	Resource Size			
16 through 47	Resource Dependent Values			
48	Reserved = 0		Percent	Status

Length This is the number of bytes of data, 44, that follow this field

Component Count
This is the number of components that are incorporated into the resource

Resource Size
This is the number of blocks available for user data

Resource Dependent Values
These resource parameters differ for each filter type. The structure for each filter type is reported by the FC_QueryMetaResrcParams function (see “FC_QueryMetaResrcParams” on page 194). All filters report the block size in bytes 19 through 16. Zeroes are returned in fields not defined.

Status This can be one of the following:

FS_ResrcOffline
If this is a pseudofilter, this status indicates the resource is in the RS_Offline state defined on “FN_REGY_ResrcChangeToRegistry” on page 103. If it is a real filter, this status indicates that the array does not have enough components to function or it contains inconsistent components.

FS_ResrcOnline
This is only returned for a pseudofilter. It indicates the resource is in the RS_Online state defined on “FN_REGY_ResrcChangeToRegistry” on page 103.

FS_ResrcOnlineNonDeg
The array is not degraded and is fully operational.

FS_ResrcOnlineDeg
The array is degraded.

FS_ResrcOnlineRebuild
The missing component has been returned to a degraded array which is in the process of rebuilding.

FS_ResrcOnlineExposed
A component is missing from a RAID-5 array and no writes have yet been required to that component.

FS_ResrcUnknown
This is the state that an array is in until N-1 components are visible for the first time.

FS_ResrcWrapped
The physical resource is wrapped

FS_RescrFormatting
The physical resource is being formatted: the percent field reports the amount currently formatted.

FS_ResrcCertifying
The physical resource is being certified: the percent field reports the amount currently certified.

FS_ResrcFormatFailed
Formatting the disk has failed; the percent field reports how much of the disk was formatted before the failure.

FS_ResrcCertifyFailed
Certifying the disk has failed; the percent field reports how much of the disk was certified before the failure.

FS_ResrcInUse
This is only reported for an NVRAM resource. It indicates that the defined resource is associated with a known array.

FS_ResrcDormant
This is only reported for an NVRAM resource. It indicates that the defined resource is not associated with any known array.

Percent
This is an integer in the range 0 through 99 indicating the percentage completion of an operation for the following fields:
FS_ResrcRebuild
FS_ResrcFormatting
FS_ResrcCertifying

Result The following result fields can be returned:
AS_Success
AE_Failure
AE_BadResrcSerialNumber
Illegal Request (range)

FC_CandidateCount

This function reports the total number of candidate components that are available for use in creating an array. Only those currently unused candidates that match exactly the specified type (resource dependent values) are included in the count.

Parameter_DDR
This is a pointer to the following data:

Byte	3	2	1	0
0	Reserved = 0	Filter Type	Function = 5	
4 through 35	Resource Dependent Values			

Function
This is the function code, 5, for the FC_CandidateCount function

Filter Type
This is the filter type to which the function is directed

Resource Dependent Values
This field differs for each filter type (see “FC_ResrcView” on page 183 for details)

Status_DDR
This is a pointer to the following data:

Byte	3	2	1	0
0	Unused			
4	Length = 4			
8	Candidate count			

Length This is the number of bytes of data that follow this field (4)

Candidate Count
This is the number of currently unused components, with characteristics matching the resource-dependent-value field, that could be used to create the array.

Result The following result fields can be returned:
AS_Success
AE_Failure
AE_BadParameterValues
AE_InvalidCandidateRequest
Illegal Request (range)

FC_CandidateList

This function reports the serial numbers of candidate components that are available for use in creating an array. (The total number of available components is returned by the FC_CandidateCount function.) The function specifies the ordinal number of the first component and number of candidates (maximum 16) for which data is to be reported. The length field reports the number of candidates for which data is returned.

Parameter_DDR
This is a pointer to the following data:

Byte	3	2	1	0
0	Reserved = 0	Filter Type	Function = 6	
4 through 35	Resource Dependent Values			
36	First Candidate (n)			
40	Requested Count (m)			

Function
This is the function code, 6, for FC_CandidateList

Filter Type
This is the filter type to which the function is directed

Resource Dependent Values
This field differs for each filter type (see “FC_ResrcView” on page 183 for details)

First Candidate (n)
This is the ordinal number of the first candidate (starting with zero) that could be used to create the array specified.

Requested count
This is the number of candidates (maximum 16), starting with the candidate specified in the first-candidate field, for which data is requested.

Status_DDR
This is a pointer to the following data:

Byte	3	2	1	0
0	Unused			
4	Length			
8 through 19	Serial Number (n)			
20	Reserved = 0	Serial Number (n)		
24	Reserved = 0		Percent	Status
28 through 39	Serial Number (n+1)			
40	Reserved = 0	Serial Number (n+1)		
44	Reserved = 0		Percent	Status
.				
20m-12 through 20m-1	Serial Number (n+m)			
20m	Reserved = 0	Serial Number (n+m)		
20m+4	Reserved = 0		Percent	Status

Length This is the number of data bytes that follow this field (320 maximum)

Serial number
This 15-character ASCII string is the serial number of each component.

Status This can be one of the following:

FS_CandOnline
The component is in the RS_Online state (see “FN_REGY_ResrcChangeToRegistry” on page 103).

FS_CandOffline
The component is in the RS_Offline state (see “FN_REGY_ResrcChangeToRegistry” on page 103).

Percent
This field is zero.

Result The following result fields can be returned:
AS_Success

AE_Failure
AE_BadParameterValues
AE_InvalidCandidateRequest
Illegal Request (range)

FC_ResrcCreate

This function is used to create a new resource, composed from a group of components (maximum 16). The new resource will have the name or serial number provided in the resource-serial-number field. If the filter type is FT_DriverManualDisk or FT_DriverAutomaticDisk, the ComponentCount must be set to zero and there are no associated component serial numbers.

Parameter_DDR

This is a pointer to the following data:

Byte	3	2	1	0
0	Reserved = 0	Filter Type	Function = 7	
4 through 15	Resource Serial Number			
16	Reserved = 0	Resource Serial Number		
20 through 51	Resource Dependent Values			
52	Component Count (n)			
56 through 67	Serial Number (1)			
68	Reserved = 0	Serial Number (1)		
72 through 83	Serial Number (2)			
84	Reserved = 0	Serial Number (2)		
.				
40+16n through 51+16n	Serial Number (n)			
52+16n	Reserved = 0	Serial Number (n)		

Function

This is the function code, 7, for FC_ResrcCreate

Filter Type

This is the filter type to which the function is directed

Resource Serial Number

This 15-character ASCII string is the name or serial number of the resource that is created by this function.

Resource Dependent Values

This field differs for each filter type (see “FC_ResrcView” on page 183 for details)

Component Count
This is the number of components to be used to create the new resource.

Serial number
These 15-character ASCII strings are the serial numbers of the components to be used to create the new resource.

Status_DDR
No Status data is required for this function, but a Status_DDR that points to the following data is returned:

Byte	3	2	1	0
0	Unused			
4	Length = 0			

Result The following result fields can be returned:
AS_Success
AE_BadResrcSerialNumber
AE_BadComponentCount
AE_BadComponentSerialNumber
AE_BadParameterValues
AE_Failure
AE_SetOMTFailed
AE_NvramError
AE_InvalidCreateRequest
Illegal Request (range)

FC_ResrcDelete

This function is used to delete an existing resource.

Parameter_DDR
This is a pointer to the following data:

Byte	3	2	1	0
0	Reserved = 0	Filter Type	Function = 8	
4 through 15	Serial Number			
16	Reserved = 0	Serial Number		

Function
This is the function code, 8, for FC_ResrcDelete

Filter Type
This is the filter type to which the function is directed

Serial Number
This 15-character ASCII string is the name of the resource that is to be deleted.

Status_DDR

No status data is required for this function, but a Status_DDR that points to the following data is returned:

Byte	3	2	1	0
0	Unused			
4	Length = 0			

Result The following result fields can be returned:

AS_Success
AE_BadResrcSerialNumber
AE_Failure
AE_ConfirmRequired
Illegal Request (range)

FC_ResrcRename

This function is used to rename an existing resource.

Parameter_DDR

This is a pointer to the following data:

Byte	3	2	1	0
0	Reserved = 0	Filter Type	Function = 9	
4 through 15	Old Serial Number			
16	Reserved = 0	Old Serial Number		
20 through 31	New Serial Number			
32	Reserved = 0	New Serial Number		

Function

This is the function code, 9, for FC_ResrcRename

Filter Type

This is the filter type to which the function is directed

Old Serial Number

This 15-character ASCII string is the old name or serial number of the resource.

New Serial Number

This 15-character ASCII string is the new name or serial number of the resource.

Status_DDR

No status data is required for this function, but a Status_DDR that points to the following data is returned:

Byte	3	2	1	0
0	Unused			
4	Length = 0			

Result The following result fields can be returned:
AS_Success
AE_BadOldSerialNumber
AE_BadNewSerialNumber
AE_Failure
Illegal Request (range)

FC_ComponentView

This function is used to return the serial numbers of all the components of a resource. The number of the components is returned by the FC_ResrcView function. The request includes the ordinal number of the first component and the number of components to be reported (maximum 16). The returned length field describes the number of components reported.

It is not valid to address the FC_ComponentView function to a pseudofilter.

Parameter_DDR
This is a pointer to the following data:

Byte	3	2	1	0
0	Reserved = 0	Filter Type	Function = 10	
4 through 15	Resource Serial Number			
16	Reserved = 0	Resource Serial Number		
20	First Component (n)			
24	Requested Count (m)			

Function
This is the function code, 10, for FC_ComponentView

Filter Type
This is the filter type to which the function is directed

Resource Serial Number
This 15-character ASCII string is the name of the resource.

First Component
This is the ordinal number of the first component (starting at zero) to be reported.

Requested Count
This is the maximum number of components that should be reported starting from the identified first component

Status_DDR
This is a pointer to the following data:

Byte	3	2	1	0
0	Unused			
4	Length			
8 through 19	Serial Number (n)			
20	Reserved = 0	Serial Number (n)		
24	Reserved = 0		Percent	Status
28 through 39	Serial Number (n+1)			
40	Reserved = 0	Serial Number (n+1)		
44	Reserved = 0		Percent	Status
.				
20m-12 through 20m-1	Serial Number (n+m)			
20m	Reserved = 0	Serial Number (n+m)		
20m+4	Reserved = 0		Percent	Status

Length This is the number of data bytes that follow this field (320 maximum)

Serial number
These 15-character ASCII strings are the serial numbers of each component.

Status This can be one of the following:

FS_CompPresent
This is returned if the resource is made out of other components that are all present.

FS_CompNotPresent
This is returned if the array is made out of other arrays and this component is missing.

FS_CompNotPresentDeconf
This can be returned in a RAID-5 array for a component that is deconfigured.

FS_CompNotPresentBlank
This component is a blank slot (that is, type FT_BlankReserved)

FS_CompPresentRebuild
This is a destination of a current rebuild operation.

FS_CompPresentRebuildme
This is a destination of a future rebuild operation.

Percent
These fields are zero.

Result The following result fields can be returned:

AS_Success
AE_Failure
AE_BadResrcSerialNumber
AE_FiltersOnly
Illegal Request (range)

FC_ComponentExchange

This function is used to replace a component of a resource with a new component, for example, to replace a faulty disk drive in a RAID-5 array. It is acceptable not to define a new replacement if one is not available; provide a null serial number instead (this should be unique if more than one null replacement is to be undertaken). Attempting to exchange a component of a degraded array is not permitted because it would cause deletion of the array; FC_ResrcDelete should be used instead.

Parameter_DDR

This is a pointer to the following data:

Byte	3	2	1	0
0	Reserved = 0	Filter Type	Function = 11	
4 through 15	Resource Serial Number			
16	Reserved = 0	Resource Serial Number		
20 through 31	Old Component Serial Number			
32	Reserved = 0	Old Component Serial Number		
36 through 47	New Component Serial Number			
48	Reserved = 0	New Component Serial Number		

Function

This is the function code, 11, for FC_ComponentExchange

Filter Type

Filter Type to which the function is directed

Resource Serial Number

This 15-character ASCII string is the name of the array.

Old Component Serial number

This 15-character ASCII string is the name of the old component.

New Component Serial Number

This 15-character ASCII string is the name of the new component.

Status_DDR

No status data is required for this function, but a Status_DDR that points to the following data is returned:

Byte	3	2	1	0
0	Unused			
4	Length = 0			

Result The following result fields can be returned:
AS_Success
AE_BadResrcSerialNumber
AE_BadOldComponentSerialNumber
AE_BadNewComponentSerialNumber
AE_DegradedArray
AE_Failure
AE_ArrayIsBroken
AE_BadExchangeCandidate
AE_FiltersOnly
Illegal Request (range)

FC_QueryMetaResrcParams

This function returns a description of the resource parameters for the specified filter type. These are used in other functions, for example, FC_ResrcView.

Parameter_DDR
This is a pointer to the following data:

Byte	3	2	1	0
0	Reserved = 0	Filter Type	Function = 12	

Function
This is the function code, 12, for FC_QueryMetaResrcParams

Filter Type
This is the filter type to which the function is directed

Status_DDR
This is a pointer to the following data:

Byte	3	2	1	0
0	Unused			
4	Length			
8	Fields	Reserved = 0	Max Component	Min Component
12	Offset		Size	Type
16	MinValue			
20	MaxValue			
24	Default Value			
28	StepValue			
32	Control			
36 through n	Further definitions (copies of bytes 12 through 35 for each parameter)			

Length The number of data bytes that follow this field

Min Component
This is the minimum number of components for this filter type.

Max Component
This is the maximum number of components for this filter type.

Fields This is the number of parameters defined in the function. Each parameter is described in the same format as bytes 12 through 35 of the status data.

Type This is the type of parameter, which can include the following:
SDS_BLOCKSIZE
SDS_DISK_NUMBER
SDS_STRIPE_SIZE
SDS_STRETCH_SIZE
SDS_MODE_FLAGS
SDS_STRIDE_SIZE
SDS_HOT_SPARE_ENABLED
SDS_HOT_SPARE_EXACT_SIZE
SDS_REBUILD_PRIORITY
SDS_SPEC_READ
SDS_BAD_PTY_STRIDE
SDS_BAD_COMPONENT_STRIDE
SDS_BAD_STRIPE

Size This is the size in bits of the contents of the parameter

Offset This is the offset of this parameter from the start of the resource parameters. It must be byte aligned even though the parameter might not contain an integer number of bytes.

MinValue
This is the minimum value allowed for this parameter

MaxValue
This is the maximum value allowed for this parameter

Default value
This is the default value used for this parameter

StepValue
This is the increment allowed for this parameter

Control
This contains the following control information for the parameter:
SDSF_MB
Units are MB
SDSF_KB
Units are KB
SDSF_PERCENT
Units are %
SDSF_ON_OFF
Display On/Off rather than 0/1

SDSF_BYTES
Units are bytes
SDSF_READONLY
Cannot be changed
SDSF_UNIQUE
Entries must be different

Result The following result fields can be returned:
AS_Success
AE_Failure
Illegal Request (range)

FC_ModifyResrcParams

This function is used to modify resource parameters of a specified array.

Parameter_DDR
This is a pointer to the following data:

Byte	3	2	1	0
0	Reserved = 0	Filter Type	Function = 13	
4 through 15	Serial Number			
16	Reserved = 0	Serial Number		
20 through 51	New Resource Dependent Values			

Function
This is the function code, 13, for FC_ModifyResrcParams

Filter Type
This is the filter type to which the function is directed

Serial number
This is the serial number of the resource.

New Resource Dependent Values
This contains new data for the resource parameter for this filter. The data differs for each filter type (see “FC_QueryMetaResrcParams” on page 194 for details).

Status_DDR
No status data is required for this function, but a Status_DDR that points to the following data is returned:

Byte	3	2	1	0
0	Unused			
4	Length = 0			

Result The following result fields can be returned:
AS_Success
AE_BadParameterValues

AE_Failure
AE_NotInTable
AE_FiltersOnly
Illegal Request (range)

FC_FlashIndicator

This function is used to cause the light on all the components of the resource to flash or to stop it flashing.

Parameter_DDR

This is a pointer to the following data:

Byte	3	2	1	0
0	Reserved = 0	Filter Type	Function = 14	
4 through 15	Serial Number			
16	Reserved = 0	Serial Number		
20	Flash			

Function

This is the function code, 14, for FC_FlashIndicator

Filter Type

This is the filter type to which the function is directed

Serial Number

This is the serial number of the component whose light is to be set flashing or turned off.

Flash When this is zero, the light does not flash. When this is nonzero the light flashes on and off continuously. The duration of each flash is approximately one second.

Status_DDR

No status data is required for this function, but a Status_DDR that points to the following data is returned:

Byte	3	2	1	0
0	Unused			
4	Length = 0			

Result The following result fields can be returned:

AS_Success
AE_BadParameterValues
AE_Failure
AE_NotInTable
Illegal Request (range)

FC_VPDInquiry

This function returns the Vital Product Data (VPD) information from the resource. It is only valid for it to be sent to a resource type FT_DriverAutomaticDisk, FT_DriverManualDisk, FT_PhysicalDisk, or FT_NotOwned. It is not valid to sent it to an array filter.

Parameter_DDR

This is a pointer to the following data:

Byte	3	2	1	0
0	Reserved = 0	Filter Type	Function = 15	
4 through 15	Serial Number			
16	Reserved = 0	Serial Number		
20	Reserved = 0		Page Code	EVPD

Function

This is the function code, 15, for FC_VPDInquiry

Filter Type

Filter Type to which the function is directed

Serial Number

This is the serial number of the resource whose VPD is requested.

EVPD This field, Enable Vital Product Data (EVPD), controls whether the data returned is standard inquiry data or individual VPD pages. EVPD can be:

VP_NoEVPD

Standard VPD inquiry data is returned.

VP_EVPD

The VPD inquiry data of the page identified by the page-code field is returned.

Page Code

This identifies the page of vital VPD inquiry data to be returned. Page 00h identifies the pages that can be returned.

Status_DDR

This is a pointer to the following data:

Byte	3	2	1	0
0	Unused			
4	Length			
8 through n	VPD Data			

Length This is the number of bytes that follow this field.

VPD Data
This is the same data as that returned to a SSA-SCSI Inquiry command to the resource. This data is defined in the *Technical Reference* for the resource.

Result The following result fields can be returned:
AS_Success
AE_BadParameterValues
AE_Failure
AE_NotInTable
Illegal Request (range)

FC_HardwareInquiry

This function returns hardware-specific information about the specified resource. It is only valid for resource types FT_DriverAutomaticDisk, FT_DriverManualDisk, FT_PhysicalDisk, and FT_NotOwned. It is not valid to send it to an array filter.

Parameter_DDR
This is a pointer to the following data:

Byte	3	2	1	0
0	Reserved = 0	Filter Type	Function = 16	
4 through 15	Serial Number			
16	Reserved = 0	Serial Number		

Function
This is the function code, 16, for FC_HardwareInquiry

Filter Type
Filter type to which the function is directed

Serial Number
This is the serial number of the resource for which information is requested.

Status_DDR
This is pointer to the buffer that receives the following data:

Byte	3	2	1	0
0	Port 1 SSA loop A	Port 2 SSA loop A	Port 1 SSA loop B	Port 2 SSA loop B
4	Reserved = 0			Status

Port n This is the SSA address of the node on this port of the adapter card. If the resource is not connected to this port then a value of FFh is returned. This field is valid when the result field is AS_Success or AE_ReservationConflict.

Status This reports the state of the resource and is valid when the result field is AS_Success. It has the following definition:

ST_Good
Good

ST_Failed
Failed. In this state, if the resource is a target on an SSA link, a Test Unit Ready SSA command is rejected with check-condition status. This might be caused by a failure of power-on self-tests, a stopped motor, or any degraded mode condition.

ST_LossRedundancy
In this state, the resource has lost some redundancy, for example, loss of redundant power or cooling. The disk service determines this by sending a SSA-SCSI Inquiry command to the resource.

Result The following result fields can be returned:
AS_Success
AE_NotFilters
AE_Failure
AE_NotInTable
Illegal Request (range)

FC_CompExchCount

This function returns the number of components that are available to be exchanged into a given array.

Parameter_DDR
This is a pointer to the following data:

Byte	3	2	1	0
0	Reserved = 0	Filter Type	Function = 17	
4 through 15	Serial Number			
16	Reserved = 0	Serial Number		
20	Reserved = 0			Exchange Type

Function
This is the function code, 17, for FC_CompExchCount

Filter Type
This is the filter type to which the function is directed

Serial Number
This 15-character ASCII string is the name of the array for which a component exchange is required.

Exchange Type
This can be one of the following:

FT_NotOwned
The count field in the status data refers to the number of components that could be exchanged.

FT_HotSpare
The count field in the status data refers to the number of hot-spare components available to this filter.

Status_DDR
This is a pointer to the following data:

Byte	3	2	1	0
0	Unused			
4	Length of following data = 4			
8	Count of available exchange components			

Result The following result fields can be returned:
AS_Success
AE_BadResrcSerialNumber
Illegal Request (range)
AE_Failure
AE_FiltersOnly
AE_NotInTable

FC_CompExchCandList

This function reports the serial numbers of all the exchange components that are available to be exchanged into a specified array. The function specifies the ordinal number of the first candidate and the number of candidates for which serial numbers are requested.

Parameter_DDR
This is a pointer to the following data:

Byte	3	2	1	0
0	Reserved = 0	Filter Type	Function = 18	
4 through 15	Serial Number			
16	Reserved = 0	Serial Number		
20	First Candidate			
24	Requested Count			
28	Reserved = 0			Exchange Type

Function
This is the function code, 18, for FC_CompExchCandList

Filter Type
This is the filter type to which the function is directed

Serial number
This 15-character ASCII string is the name of the array for which a component exchange is required.

First Candidate
This is the ordinal number of the first component (starting with zero) to be reported.

Requested Count
This is the maximum number of components to be reported.

Exchange type
This can be one of the following:

FT_NotOwned
The identification of components that can be exchanged is returned in the status data

FT_HotSpare
The identification of components that can be hot spares for this filter is returned in the status data.

Status_DDR
This is a pointer to the following data:

Byte	3	2	1	0
0	Unused			
4	Length			
8 through 19	Serial Number (n)			
20	Reserved = 0	Serial Number (n)		
24	Reserved = 0		Percent	Status
28 through 39	Serial Number (n+1)			
40	Reserved = 0	Serial Number (n+1)		
44	Reserved = 0		Percent	Status
.				
20m-12 through 20m-1	Serial Number (n+m)			
20m	Reserved = 0	Serial Number (n+m)		
20m+4	Reserved = 0		Percent	Status

Length This is the number of data bytes that follow this field.

Serial number
This 15-character ASCII string is the serial number of each candidate.

Status This can be one of the following:

FS_CandOnline
The component is in the RS_Online state (see “FN_REGY_ResrcChangeToRegistry” on page 103).

FS_CandOffline
The component is in the RS_Offline state (see “FN_REGY_ResrcChangeToRegistry” on page 103).

Percent
This field is zero.

Result The following result fields can be returned:
AS_Success
AE_Failure
AE_BadResrcSerialNumber
AE_FiltersOnly
AE_NotInTable
Illegal Request (range)

FC_AdapterVPD

This function returns the adapter VPD information.

Parameter_DDR
This is a pointer to the following data:

Byte	3	2	1	0
0	Reserved = 0	Filter Type	Function = 19	

Function
This is the function code, 19, for FC_AdapterVPD

Filter Type
This is the filter type to which the function is directed. This must be FT_Adapter, which is not really a filter.

Status_DDR
This is a pointer to the buffer that receives the following data:

Byte	3	2	1	0
0	Reserved = 0			
4	Length			
8 through n	VPD data			

Length This is the number of data bytes that follow this in field.

VPD data
This is the adapter VPD data

Result The following result fields can be returned:
AS_Success

AE_NotInTable

Illegal Request (range)

FC_SyncHealth

This function returns the most significant health-check sense data of all services attached to the registry. In response to the FC_SyncHealth transaction, the array-configuration service issues a FN_REGY_SyncHealthCheckToRegy transaction to the registry service. If no sense data is to be returned, the result field is AS_Success. The result field is AE_Failure, if sense data is returned in the status data.

Parameter_DDR
This is a pointer to the following data:

Byte	3	2	1	0
0	Reserved = 0	Filter Type	Function = 20	

Function
This is the function code, 20, for FC_SyncHealth

Filter Type
This is the filter type to which the function is directed. This must be FT_Adapter, which is not really a filter.

Status_DDR
This is a pointer to the buffer that receives following data:

Byte	3	2	1	0
0	Reserved = 0			
4	Length			
8 through n	Sense data			

Length This is the number of data bytes that follow this in field.

Sense data
The sense data is the most significant error log data from attached services. “FN_REGY_LogErrorFromRegistry” on page 98 defines the sense data.

Result The following result fields can be returned:
AS_Success

AE_NotInTable

Illegal Request (range)

AE_Failure

FC_Wrap

This function opens the identified physical resource in service mode. This causes the SSA ports on the adjacent nodes to be wrapped. The handle returned to the array-configuration service, when the resource is opened in service mode, is not returned to the client but held by the array-configuration service pending a future FC_Unwrap or FC_UnwrapAll or until the adapter is rebooted.

The rules for resources that can be opened in service mode causing adjacent SSA ports to be wrapped are defined in “FN_ISALMgr_Open” on page 135.

Parameter_DDR
This is a pointer to the following data:

Byte	3	2	1	0
0	Reserved = 0	Filter Type	Function = 21	
4 through 15	Serial Number			
16	Reserved = 0	Serial Number		

Function
This is the function code, 21, for FC_Wrap

Filter Type
This is the filter type to which the function is directed. This must be FT_PhysicalDisk, which is not really a filter.

Serial Number
This is the serial number of the resource to be wrapped.

Status_DDR
No status data is required for this function, but a Status_DDR that points to the following data is returned:

Byte	3	2	1	0
0	Reserved = 0			
4	Length = 0			

Result The following result fields can be returned:
AS_Success

AE_NotInTable
Filter not FT_PhysicalDisk or serial number not found
AE_PhysWrapped
Device is currently wrapped
AE_PhysFormatting
Device is currently formatting
AE_PhysCertifying
Device is currently certifying
AE_AccessDenied

AE_Failure

AE_InvalidRID

AE_LogOpen

AE_SSAString

Illegal Request (range)

AE_InServiceMode

FC_Unwrap

This function closes the identified physical resource that had previously been in service mode. If the resource has not previously been opened in MD_Service mode (via the FC_Wrap IACL transaction), AE_NotOpen result field is returned.

Parameter_DDR
This is a pointer to the following data:

Byte	3	2	1	0
0	Reserved = 0	Filter Type	Function = 22	
4 through 15	Serial Number			
16	Reserved = 0	Serial Number		

Function
This is the function code, 22, for FC_Unwrap

Filter Type
This is the filter type to which the function is directed. This must be FT_PhysicalDisk, which is not really a filter.

Serial Number
This is the serial number of the resource to be unwrapped.

Status_DDR
No status data is required for this function, but a Status_DDR that points to the following data is returned:

Byte	3	2	1	0
0	Reserved = 0			
4	Length = 0			

Result The following result fields can be returned:
AS_Success

AE_NotInTable
Filter not FT_PhysicalDisk or serial number not found

AE_NotOpen
Resource not opened via FC_Wrap
Illegal Request (range)

AE_Failure

FC_UnwrapAll

This function closes any physical resource that had previously been in service mode.

Parameter_DDR
This is a pointer to the following data:

Byte	3	2	1	0
0	Reserved = 0	Filter Type	Function = 23	

Function
This is the function code, 23, for FC_UnwrapAll

Filter Type
This is the filter type to which the function is directed. This must be FT_PhysicalDisk, which is not really a filter.

Status_DDR
No status data is required for this function, but a Status_DDR that points to the following data is returned:

Byte	3	2	1	0
0	Reserved = 0			
4	Length = 0			

Result The following result fields can be returned:
AS_Success

AE_NotInTable
Filter not FT_PhysicalDisk or serial number not found
Illegal Request (range)

AE_Failure

FC_Test

This function causes internal checkouts to be executed in a physical resource. It is implemented by the array-configuration service issuing a FN_ISALMgr_Open and FN_ISAL_Test followed by an FN_ISAL_Close to the identified physical resource.

Parameter_DDR
This is a pointer to the following data:

Byte	3	2	1	0
0	Reserved = 0	Filter Type	Function = 24	
4 through 15	Serial Number			
16	Reserved = 0	Serial Number		
20	Reserved = 0			Type

Function
This is the function code, 24, for FC_Test

Filter Type
This is the filter type to which the function is directed. This must be FT_PhysicalDisk, which is not really a filter.

Serial Number
This is the serial number of the resource to be tested.

Type This must be TT_Test to request that internal checkout be performed in the resource.

Status_DDR
No status data is required for this function, but a Status_DDR that points to the following data is returned:

Byte	3	2	1	0
0	Reserved = 0			
4	Length = 0			

Result The following result fields can be returned:
AS_Success

AE_NotInTable
Filter not FT_PhysicalDisk or serial number not found
AE_PhysWrapped
Device is currently wrapped
AE_PhysFormatting
Device is currently formatting
AE_PhysCertifying
Device is currently certifying
AE_AccessDenied
(Copied from FN_ISALMgr_Open)
AE_InvalidRID
(Copied from FN_ISALMgr_Open)
AE_LogOpen
(Copied from FN_ISALMgr_Open or FN_ISAL_Test)
Illegal Request (range)

AE_ReservationConflict
(Copied from FN_ISAL_Test)
AE_HardwareError
(Copied from FN_ISAL_Test)

AE_NotReady
(Copied from FN_ISAL_Test)
AE_Offline
(Copied from FN_ISAL_Test)
AE_FencedOut
(Copied from FN_ISAL_Test)
AE_FormatDegraded
(Copied from FN_ISAL_Test)
AE_FormatinProgress
(Copied from FN_ISAL_Test)

FC_Format

This function causes formatting of the physical disk to start. AS_Success is returned if formatting does start. The array-configuration service issues a FN_ISALMgr_Open and FN_ISAL_Format to the physical disk. If formatting starts successfully, the array-configuration service constructs a record that tracks the serial number of the disk.

The array-configuration service periodically issues FN_ISAL_Progress to this disk to determine the progress of the formatting. If a FC_ResrcList or FC_ResrcView transaction is issued to a disk being formatted, the progress of the format from the last FN_ISAL_Progress transaction issued is returned in the SNS_Percent field.

When formatting completes successfully, the handle is closed and the record of the disk serial number is removed. If formatting fails, the handle is closed and a record kept for that disk so that FS_ResrcFormatFailed can be returned to a subsequent FC_ResrcList or FC_ResrcView transaction. This failure record persists until the adapter is re-booted or a wrap, format or certify is issued.

Parameter_DDR
This is a pointer to the following data:

Byte	3	2	1	0
0	Reserved = 0	Filter Type	Function = 25	
4 through 15	Serial Number			
16	Reserved = 0	Serial Number		
20	Blocksize			

Function
This is the function code, 25, for FC_Format

Filter Type
This is the filter type to which the function is directed. This must be FT_PhysicalDisk, which is not really a filter.

Serial Number
This is the serial number of the resource to be formatted.

Blocksize
This is the number of bytes in each block. This must be a value that is supported by the disk drive.

Status_DDR
No status data is required for this function, but a Status_DDR that points to the following data is returned:

Byte	3	2	1	0
0	Reserved = 0			
4	Length = 0			

Result The following result fields can be returned:

AS_Success

AE_NotInTable
Filter not FT_PhysicalDisk or serial number not found

Illegal Request (range)

AE_PhysWrapped
Device is currently wrapped

AE_PhysFormatting
Device is currently formatting

AE_PhysCertifying
Device is currently certifying

AE_Failure

AE_AccessDenied
(Copied from FN_ISALMgr_Open)

AE_InvalidRID
(Copied from FN_ISALMgr_Open)

AE_LogOpen
(Copied from FN_ISALMgr_Open or FN_ISAL_Format)

AE_ReservationConflict
(Copied from FN_ISAL_Format)

AE_HardwareError
(Copied from FN_ISAL_Format)

AE_NotReady
(Copied from FN_ISAL_Format)

AE_Offline
(Copied from FN_ISAL_Format)

AE_FencedOut
(Copied from FN_ISAL_Format)

AE_LogOpen
(Copied from FN_ISAL_Format)

AE_FormatinProgress
(Copied from FN_ISAL_Format)

FC_Certify

This function starts the verification of every block on the physical disk. AS_Success is returned if this verification can be started successfully.

The array-configuration service issues a FN_ISALMgr_Open to the physical disk before returning the result field to the FC_Certify transaction. If the disk opens successfully, the array-configuration service constructs a record that tracks the serial number of the disk being certified and the progress of the certification.

The array-configuration service sends FN_ISAL_Reads with FF_Verify flag on to verify that all blocks on the disk can be read. If a FC_ResrcList or FC_ResrcView transaction is issued to a disk that is being certified, the progress of the certify is returned in the SNS_Percent field.

When certifying completes successfully, the handle is closed and the record of the disk serial number is removed. If certifying fails, the handle is closed and a record kept for that disk so that FS_ResrcCertifyFailed can be returned to a subsequent FC_ResrcList or FC_ResrcView transaction. This failure record persists until the adapter is re-booted or a wrap, format, or certify is issued.

Parameter_DDR

This is a pointer to the following data:

Byte	3	2	1	0
0	Reserved = 0	Filter Type	Function = 26	
4 through 15	Serial Number			
16	Reserved = 0	Serial Number		
20	Blocksize			

Function

This is the function code, 26, for FC_Certify

Filter Type

This is the filter type to which the function is directed. This must be FT_PhysicalDisk, which is not really a filter.

Serial Number

This is the serial number of the resource to be certified.

Blocksize

This is the number of bytes in each block. This must be a value that is supported by the disk drive.

Status_DDR

No status data is required for this function, but a Status_DDR that points to the following data is returned:

Byte	3	2	1	0
0	Reserved = 0			
4	Length = 0			

Result The following result fields can be returned:

AS_Success

AE_NotInTable
Filter not FT_PhysicalDisk or serial number not found

Illegal Request (range)

AE_PhysWrapped
Device is currently wrapped

AE_PhysFormatting
Device is currently formatting

AE_PhysCertifying
Device is currently certifying

AE_Failure
Unable to begin certifying

AE_AccessDenied
(Copied from FN_ISALMgr_Open)

AE_InvalidRID
(Copied from FN_ISALMgr_Open)

AE_LogOpen
(Copied from FN_ISALMgr_Open)

FC_Read

This function reads a single sector from the disk identified by the serial number field.

The array-configuration service issues an FN_ISALMgr_Open, FN_ISALMgrCharacteristics, FN_ISAL_Read, and FN_ISAL_Close to the resource before returning the result field for the FC_Read function.

Because this IACL function does not involve sending a handle to identify the resource, some applications, for example, service, might prefer to use this rather than FN_ISAL_Read when reading a sector from a disk.

The function can be addressed to a resource of any filter type, but that resource must not already be open by the device driver or by another filter.

Parameter_DDR
This is a pointer to the following data:

Byte	3	2	1	0
0	Reserved = 0	Filter Type	Function = 27	
4 through 15	Serial Number			
16	Reserved = 0	Serial Number		
20	Logical Block Address			
24	Reserved = 0		Flags	Priority

Function
This is the function code, 27, for FC_Read

Filter Type
This is the filter type to which the function is directed.

Serial Number
This is the serial number of the resource to be read.

Logical Block Address
This is the logical block address of the block to be read.

Flags This field controls the type of read to be executed. These are defined in detail in “FN_ISAL_Read” on page 139.

Priority
This field is reserved for future use.

Status_DDR
This is a pointer to the following data:

Byte	3	2	1	0
0	Reserved = 0			
4	Length = 0			
8 through n	Data			

Length Number of bytes of data following in the status data. Only 512-byte blocksize is supported.

Data The data on the LBA requested.

Result The following result fields can be returned:
AS_Success

AE_NotInTable
Filter not FT_PhysicalDisk or serial number not found
Illegal Request (range)

AE_PhysWrapped
Device is currently wrapped
AE_PhysFormatting
Device is currently formatting

AE_PhysCertifying
Device is currently certifying
AE_Failure
Unable to begin reading
AE_AccessDenied
(Copied from FN_ISALMgr_Open)
AE_InvalidRID
(Copied from FN_ISALMgr_Open)
AE_LogOpen
(Copied from FN_ISALMgr_Open)
AE_ReservationConflict
(Copied from FN_ISAL_Read)
AE_HardwareError
(Copied from FN_ISAL_Read)
AE_NotReady
(Copied from FN_ISAL_Read)
AE_MediumError
(Copied from FN_ISAL_Read)
AE_InvalidSignature
(Copied from FN_ISAL_Read)
AE_Offline
(Copied from FN_ISAL_Read)
AE_FencedOut
(Copied from FN_ISAL_Read)
AE_FormatDegraded
(Copied from FN_ISAL_Read)
AE_FormatinProgress
(Copied from FN_ISAL_Read)
AS_Warning
(Copied from FN_ISAL_Read)

FC_Write

This function writes a single sector to the disk identified by the serial number field.

The array-configuration service issues an FN_ISALMgr_Open, FN_ISALMgrCharacteristics, FN_ISAL_Write, and FN_ISAL_Close to the resource before returning the result field for the FC_Write function.

Because this IACL function does not involve sending a handle to identify the resource, some applications, for example, service, might prefer to use this rather than FN_ISAL_Write when writing a sector to a disk.

The function can be addressed to a resource of any filter type, but that resource must not already be open by the device driver or by another filter.

Parameter_DDR
This is a pointer to the following data:

Byte	3	2	1	0
0	Reserved = 0	Filter Type	Function = 28	
4 through 15	Serial Number			
16	Reserved = 0	Serial Number		
20	Logical Block Address			
24	Reserved = 0		Flags	Priority
28 through n	Data			

Function
This is the function code, 28, for FC_Write

Filter Type
This is the filter type to which the function is directed.

Serial Number
This is the serial number of the resource to be written.

Logical Block Address
This is the logical block address of the block to be written.

Flags This field controls the type of write to be executed. These are defined in detail in “FN_ISAL_Write” on page 141.

Priority
This field is reserved for future use.

Data The data to be written. The number of bytes should be that of the blocksize of the disk. Only 512-byte blocksize is supported at this time.

Status_DDR
No status data is returned for this function, but a Status_DDR that points to the following data is returned:

Byte	3	2	1	0
0	Reserved = 0			
4	Length = 0			

Result The following result fields can be returned:
AS_Success

AE_NotInTable
Filter not FT_PhysicalDisk or serial number not found
Illegal Request (range)

AE_PhysWrapped
Device is currently wrapped
AE_PhysFormatting
Device is currently formatting

AE_PhysCertifying
Device is currently certifying
AE_Failure
Unable to begin writing
AE_AccessDenied
(Copied from FN_ISALMgr_Open)
AE_InvalidRID
(Copied from FN_ISALMgr_Open)
AE_LogOpen
(Copied from FN_ISALMgr_Open)
AE_ReservationConflict
(Copied from FN_ISAL_Write)
AE_HardwareError
(Copied from FN_ISAL_Write)
AE_NotReady
(Copied from FN_ISAL_Write)
AE_MediumError
(Copied from FN_ISAL_Write)
AE_InvalidSignature
(Copied from FN_ISAL_Write)
AE_Offline
(Copied from FN_ISAL_Write)
AE_FencedOut
(Copied from FN_ISAL_Write)
AE_FormatDegraded
(Copied from FN_ISAL_Write)
AE_FormatinProgress
(Copied from FN_ISAL_Write)
AE_WriteProtect
(Copied from FN_ISAL_Write)
AS_Warning
(Copied from FN_ISAL_Write)

FC_AdapterSN

This function returns the serial number of the adapter.

Parameter_DDR
This is a pointer to the following data:

Byte	3	2	1	0
0	Reserved = 0	Filter Type	Function = 29	

Function
This is the function code, 29, for FC_AdapterSN

Filter Type
This is the filter type to which the function is directed. This must be FT_Adapter, which is not really a filter.

Status_DDR
This is a pointer to the buffer that receives the following data:

Byte	3	2	1	0
0	Reserved = 0			
4	Length = 8			
8 through 12	AdapterID			

Length Number of bytes that follow in this field (8)

AdapterID
This 8–byte binary number uniquely identifies the adapter. It is the serial number of the adapter as reported in the VPD.

Result The following result fields can be returned:

AS_Success

AE_Failure

AE_UnknownFunction

FC_CacheFormat

This function zeroes all the data in the cache if the data has all been destaged to the disk drive. The function is provided for security purposes.

If the cache contains any data that has not been destaged to the disk drive, the transaction is rejected with AE_Failure. The function should only be sent to the fast write filter; it is rejected with AE_UnknownFunction by all other filters.

Parameter_DDR
This is a pointer to the following data:

Byte	3	2	1	0
0	Reserved = 0	Filter Type	Function = 30	

Function
This is the function code, 30, for FC_CacheFormat

Filter Type
This is the filter type to which the function is directed. This must be FT_FastWriteFilter.

Status_DDR
No status data is required for this function, but a Status_DDR that points to the following data is returned:

Byte	3	2	1	0
0	Reserved = 0			
4	Length = 0			

Result The following result fields can be returned:

AS_Success

AE_Failure
Cache contains data not yet destaged to disk

AE_UnknownFunction
Filter not FT_FastWriteFilter but a known filter. This is always returned by the SSA 4-Port RAID Adapter

AE_NotInTable
Unknown filter

Application Results

The following Application-result fields might be returned at the end of a transaction:

AS_Success
The transaction has been successfully completed.

AS_Warning
The transaction has been successfully completed, but the Status DDR contains warning information that either the unit might fail soon or a logical block should be reassigned.

AE_NotReady
The service is not ready to execute this transaction.

AE_MediumError
The transaction has terminated with a nonrecoverable error condition caused by a flaw in the disk surface or an error in the recovered data. The Status DDR contains the address of the logical block in error.

AE_HardwareError
A nonrecoverable hardware error was detected during this transaction.

AE_ReservationConflict
The transaction was not executed because the resource was reserved to another client.

AE_WriteProtect
The transaction was not executed because write operations are not permitted to the resource.

AE_Failure
The transaction could not be completed for a reason other than an error.

AE_AccessDenied
Access is denied because of the mode in the open operation for that resource.

Illegal Request (range)
There was an illegal field in the transaction. A range of result field codes are reserved for Illegal Request to provide more information on the field in error.

AE_Offline
The resource was in the RS_Offline state and the transaction to this handle could not be executed. The only valid transactions that can be addressed to a handle for a resource in the RS_Offline state are FN_ISAL_Close, FN_ISALMgr_Characteristics, and FM_ISALMgr_Statistics.

AE_SCSIError
Nonzero SSA-SCSI status was returned from the resource while opened in MD_SCSI mode.

AE_UnknownFunction
The function requested is not supported.

AE_LogOpen
This function cannot be executed while the corresponding logical resource is open. This could be an attempt to open the physical resource in MD_Service mode or to send certain transactions to the physical resource.

AE_SSAString
An attempt was made to open a physical resource in MD_Service mode while that resource was in an SSA string network rather than in a loop.

AE_FencedOut
The resource is currently fenced out from executing this transaction from this client.

AE_TableFull
Resource table is full.

AE_InvalidRID
Resource is currently not known by the recipient.

AE_NotInTable
Only returned to transactions originating from the registry.

AE_NotYetImplemented
Function is not yet implemented.

AE_RetryWhenMemory
Only returned to transactions originating from the registry.

AE_ClusterNumberNotKnown
Only returned in the FN_ISAL_Fence transaction.

AE_FormatDegraded
A format operation to the disk drive has unsuccessfully completed and the user data area is not accessible.

AE_FormatInProgress
The disk drive is currently executing a formatting operation.

AE_MissingCluster
The cluster number is not known to the system.

AE_RoutingError
Error in executing a TargetTransfer transaction to another node.

AE_RemoteTimeout
The remote host did not respond to a TransferToHost transaction within the timeout period.

AE_TargetNotAvailable
The remote node is not available to receive data from the sending cluster.

AE_TargetReceiverFull
The buffer in the remote host is not available to receive data.

AE_TargetTransferTooLarge
The buffer in the remote host is too small to receive the data specified.

AE_MediaReadOnly
Write operations are not permitted to this resource, for example, to a degraded array.

AE_ParityNotValid
Parity is not valid for an array.

AE_QNTimedOut
Returned only to FN_ADAP_QueryNodes to indicate that the node did not respond to the Query Node SMS that was sent.

AE_InConfig
The adapter was in the process of SSA reconfiguration when the transaction was received.

AE_InServiceMode
The transaction cannot be executed because the resource is in service mode.

AE_OfflineTimeout

AE_NotFound

AE_PhysWrapped

AE_PhysCertifying

AE_PhysFormatting

AE_NotOpen

AE_TransferFailed

AE_TabAborted

AE_NonIsal

AE_NotSupported

AE_OtherAdapterInServiceMode

The following Application_result fields might be returned during configuration or array transactions:

AE_BadSerialNumber

AE_BadOldSerialNumber

AE_BadNewSerialNumber

AE_BadComponentCount

AE_BadComponentSerialNumber

AE_BadResrcSerialNumber

AE_BadOldComponentSerialNumber

AE_BadNewComponentSerialNumber

AE_BadParameterValues

AE_ArrayIsBroken

AE_SetOMTFailed

AE_BadExchangeCandidate

AE_FiltersOnly

AE_NotFilters

AE_NvramError

AE_InvalidCandidateRequest

AE_InvalidCreateRequest

AE_ReadOnlyParameterValue

AE_ArrayIsBrokenOrDegraded

AE_AvoidWrite

AE_AvoidReadWrite

AE_NotLocal

AE_Flush CompFailure

AE_ConfirmRequired

Chapter 7. Error Recovery and Error Logging

This section defines the error recovery and error reporting that is performed by the adapter when adapter, SSA-link, or attached-device errors are detected.

Strategy

The following strategy is implemented for error recovery and error reporting:

Error Recovery

- All possible error recovery for the attached devices is performed by the adapter. The error recovery is based on async-alert conditions, SCSI status, or a decode of the SCSI Key/Code/Qualifier sense data. If an error log entry is to be made as a result of the error recovery procedure (ERP), the adapter sends the error data to the error logger with the ID of the failing physical resource and the ID of the error log template.
- SSA Link errors are recovered in accordance with the SSA link ERP specification included in the SSA-PH functional specification. If the ERP fails, the error data is logged against the resource ID of the adapter.

Error Logging

The adapter logs disk errors only against the failing physical device.

Errors are logged as a result of:

- Device errors reported by an I/O device
- Adapter-detected failures (These include errors in the adapter, arrays, SSA links, and SSA configuration.)
- Device-driver-detected failures
- Device driver healthcheck-detected errors

The following error data is logged:

- Error data logged against disk drives consists of the 32 bytes of SCSI sense data returned by the disk drive.
- Error data logged as a result of adapter-detected failures consists of up to 36 bytes of data. The first three bytes are an adapter error code. The remainder of the data depends on the error type.

- It is possible for a user to unconfigure a pdisk without losing access to the hdisk that is assigned to that pdisk. Under these circumstances, an attempt to log errors against the pdisk fails. If the error logger receives an error log for a pdisk that is unconfigured, it should log an error against the adapter.

Error Record Templates

Each error code is assigned to an error log template.

The following is a list of error templates with their ID numbers. The ID numbers are used in byte 1 of the FN_REGY_LogErrorFromRegisty transaction to identify the template. The device driver translates the error-log-template ID to the system error ID.

01h SSA_DISK_ERR4

02h DISK_ERR4

03h SSA_DISK_ERR2

04h DISK_ERR1

05h SSA_DISK_ERR3

06h SSA_LINK_ERROR

07h SSA_DETECTED_ERROR

08h SSA_DEVICE_ERROR

09h SSA_DEGRADED_ERROR

0Ah SSA_HDW_ERROR

0Bh SSA_HDW_RECOVERED

0Ch SSA_SOFTWARE_ERROR

0Dh SSA_LINK_OPEN

0Eh SSA_DISK_ERR1

0Fh SSA_LOGGING_ERROR

10h SSA_ARRAY_ERROR

11h SSA_SETUP_ERROR

12h SSA_REMOTE_ERROR

13h SSA_CACHE_ERROR

14h SSA_CONFIG_COMPLETE

Device Error Recovery

Any device attached to the adapter reports failures by means of asynchronous-alert conditions, SCSI status codes, and SCSI sense data. For each device type attached to the adapter, a table is maintained in the adapter that defines the error recovery procedure (ERP) to be used and the data to log for all failure conditions reported by means of SCSI sense data. The default ERP table is for the SSA disk drives supplied with the 7133 SSA Subsystem. These ERP tables are built from data provided by the attaching devices. The data provided by the devices is in the following format:

Description
A text description of the error condition. This is here for information only and is not included in the ERP tables in the adapter microcode.

SCSI K/C/Q
The SCSI sense data key, code, and qualifier fields.

ERP# The error recovery procedure to be used.

Log The error-logging strategy that is used by the error recovery procedure as follows:
0 = No log entry
1 = Log the sense data
2 = Log first sense data if the ERP fails
3 = Log last sense data if the ERP fails

Template
The error logging template that should be used to log this error.

Bad Sector Management

When a sector cannot be read from a disk, but that data can be reconstructed from the other components of an array, the bad sector is reassigned by a facility called *IDISK*.

IDISK notes all sectors that return an unrecoverable medium error. When a write operation is next directed to such a sector, IDISK reassigns that block before writing the new data. If the sector cannot be reassigned by the drive, IDISK performs a software reassignment to an area of the disk outside the customer data area. Future read and write operations to that logical block address use the new sector.

When a disk that is a component of a RAID-5 array reports that it has an unrecoverable data error on one of its sectors, the array rewrites that sector when it has to reconstruct the data for that sector. The rewriting action causes IDISK to reassign the bad sector before the data is rewritten.

SSA Link Error Recovery

The SSA link error recovery procedures are defined in the SSA functional specification. This section defines the error logging strategy that is applied when link errors are reported to the adapter by means of asynchronous alert codes.

When an asynchronous alert is received, the adapter that is the SSA master logs the error code in accordance with the adapter error logging data table below.

If the error recovery fails, all adapters on the network are left with an open loop. Under these circumstances, the adapters log an error code indicating that the serial link is in degraded mode.

For alerts of type 6, no error recovery is applied. However, if the asynchronous alert type is 06 and the subtype is 01 (redundant power failure), the adapter waits for a period to see if asynchronous alerts with the same type and subtype fields are reported from more than one device and then logs the appropriate error code as defined in the table below.

Adapter Error Logging Data

The following table defines the error codes and error-log templates used when adapter error recovery procedures are invoked. These are hexadecimal characters. Each of the errors in this table are logged against the adapter resource ID.

Table 40. Adapter Error Logging Data

Condition	Template	Error Code
No error (only returned for AdapterHealthCheck)	-	00 00 00
Async type = 00 - 01 (no log)	-	-
Async type = 02 Unknown message	SSA_SOFTWARE_ERROR	32 A0 02
Async type = 03 Invalid message	SSA_SOFTWARE_ERROR	32 A0 03
Async type = 04 Protocol error	SSA_SOFTWARE_ERROR	32 A0 04
Async type = 05 Environmental error (not reported)	SSA_DETECTED_ERROR	02 A0 05
Async type = 06, Subtype = 01. Where ERP finds only one async of this type and subtype	SSA_DETECTED_ERROR	02 A0 06
Async type = 06, Subtype = 01. Where ERP finds more than one device reporting this async type and subtype	SSA_DETECTED_ERROR	02 A1 06
Async type = 06 Subttype = 03. Port not operational and POSTs failed	SSA_DEVICE_ERROR	02 A2 06
Async type = 10 Permanent line fault P=port(0-3) HH=hop(00-99)	SSA_LINK_OPEN	22 0P HH
Async type = 11 No characters received P=port(0-3) HH=hop(00-99)	SSA_LINK_ERROR	A2 1P HH
Async type = 12 Remote port disabled P=port(0-3) HH=hop(00-99)	SSA_LINK_ERROR	A2 2P HH
Async type = 13 Link reset failed P=port(0-3) HH=hop(00-99)	SSA_LINK_ERROR	A2 3P HH
Async type = 14 Retry limit exceeded P=port(0-3) HH=hop(00-99)	SSA_LINK_ERROR	02 4P HH
Async type = 15 Hardware error P=port(0-3) HH=hop(00-99)	SSA_LINK_ERROR	A2 5P HH
Async type = 16 Frame reject P=port(0-3) HH=hop(00-99)	SSA_LINK_ERROR	A2 6P HH

Table 40. Adapter Error Logging Data (continued)

Condition	Template	Error Code
Async type = 17 Invalid retry status P=port(0-3) HH=hop(00-99)	SSA_LINK_ERROR	A2 7P HH
Async type = 18 Time-out waiting for Disabled state P=port(0-3) HH=hop(00-99)	SSA_LINK_ERROR	A2 8P HH
Async type = 19 Time-out waiting for Ready state P=port(0-3) HH=hop(00-99)	SSA_LINK_ERROR	A2 9P HH
Invalid async. code	SSA_LINK_ERROR	32 FF FF
Single device reports loss of redundant power or cooling	SSA_DETECTED_ERROR	03 00 C0
Multiple devices report loss of redundant power or cooling	SSA_DETECTED_ERROR	03 01 C0
Invalid SCSI status received	SSA_DEVICE_ERROR	03 03 FF
Adapter Hardware Failure	SSA_HDW_ERROR	04 00 00
Adapter 1 MB DRAM module zero failure	SSA_DEGRADED_ERROR	04 00 01
Adapter 2 MB DRAM module zero failure	SSA_DEGRADED_ERROR	04 00 02
Adapter 4 MB DRAM module zero failure	SSA_DEGRADED_ERROR	04 00 04
Adapter 8 MB DRAM module zero failure	SSA_DEGRADED_ERROR	04 00 08
Adapter 16 MB DRAM module zero failure	SSA_DEGRADED_ERROR	04 00 16
Adapter 32 MB DRAM module zero failure	SSA_DEGRADED_ERROR	04 00 32
Adapter 1 MB DRAM module one failure	SSA_DEGRADED_ERROR	04 10 01
Adapter 2 MB DRAM module one failure	SSA_DEGRADED_ERROR	04 10 02
Adapter 4 MB DRAM module one failure	SSA_DEGRADED_ERROR	04 10 04
Adapter 8 MB DRAM module one failure	SSA_DEGRADED_ERROR	04 10 08
Adapter 16 MB DRAM module one failure	SSA_DEGRADED_ERROR	04 10 16
Adapter 32 MB DRAM module one failure	SSA_DEGRADED_ERROR	04 10 32
Adapter both DRAM failure	SSA_HDW_ERROR	04 20 00
Adapter Fast Write Cache card failure	SSA_DEGRADED_ERROR	04 25 00
Not enough DRAM for Fast Write Cache	SSA_DEGRADED_ERROR	04 25 10
Fast Write resource detected, but no Fast Write Cache card on adapter	SSA_DEGRADED_ERROR	04 25 15
Incorrect data on identified resource (LBAs not known)	SSA_CACHE_ERROR	04 25 20
Incorrect data on unidentified resource	SSA_CACHE_ERROR	04 25 21
Incorrect data on identified resource (LBAs known)	SSA_CACHE_ERROR	04 25 22
Bad version number of Fast Write Cache	SSA_CACHE_ERROR	04 25 23
Fast Write Cache not accessible for the resource	SSA_CACHE_ERROR	04 25 24
Fast Write Cache is not correct for the resource	SSA_CACHE_ERROR	04 25 25
The adapter does not provide support for a Fast Write Cache card	SSA_DEGRADED_ERROR	04 25 26
Dormant Fast Write cache entry exists	SSA_DEGRADED_ERROR	04 25 27
Duplicate fast Write disk serial number detected	SSA_DEGRADED_ERROR	04 25 28

Table 40. Adapter Error Logging Data (continued)

Condition	Template	Error Code
SSA device is preventing the completion of link configuration, where P=port (0-3) and HH=decimal hop (00-99) of the failing device	SSA_DEVICE_ERROR	04 3P HH
Device with 'failed' status where P=port (0-3) and HH=decimal hop (00-99) of the failed device	SSA_DEVICE_ERROR	04 4P HH
Open SSA Link where P=port (0-3) and HH=decimal hop (00-99) of the first device that is not accessible on the shortest link	SSA_LINK_ERROR	24 5P HH
Array offline - more than one disk not available	SSA_DEGRADED_ERROR	04 60 00
Minimum disk resources not available for array filter	SSA_ARRAY_ERROR	04 70 00
Incorrect data area of array	SSA_ARRAY_ERROR	04 75 00
Illegal Link Configuration (SIC-SIC/>48 drives/>1 adapter)	SSA_DEGRADED_ERROR	04 80 00
Illegal Link Configuration detected by array filter	SSA_DEGRADED_ERROR	04 85 00
One array member disk is on a different loop	SSA_DEGRADED_ERROR	04 86 00
Multiple array member disks are on a different loop	SSA_DEGRADED_ERROR	04 87 00
Array not available - invalid strip table full	SSA_DEGRADED_ERROR	04 88 00
Array not available - multiple device failures	SSA_DEGRADED_ERROR	04 89 00
Disk failure during array build	SSA_DEGRADED_ERROR	04 89 50
Array filter degraded - one disk not available	SSA_ARRAY_ERROR	04 90 00
Array filter exposed - one disk not available	SSA_ARRAY_ERROR	04 91 00
No spares available for an array that is configures for spares	SSA_ARRAY_ERROR	04 95 00
Incorrect parity in array	SSA_ARRAY_ERROR	04 97 00
Different adapter on each loop	SSA_DEGRADED_ERROR	04 98 00
Error accessing a remote device	SSA_REMOTE_ERROR	04 A0 00
Unable to configure device	SSA_DEVICE_ERROR	04 BP HH
Data scrubbing reconstructed a data block and reassigned it	SSA_ARRAY_ERROR	04 E0 00

SSA Disk Drive Error Recovery Table

The following are the error recovery procedures implemented in the adapter for SSA disk drives when connected to a SSA adapter. Each of the errors in this table is logged against the disk drive resource ID.

Table 41. SCSI Sense Key/Code/Qualifier recovery procedures

Description	SCSI K/C/Q	ERP#	Log	Template
No Additional Sense Information	0 00 00	1	0	-
No Index/Sector Signal	1 01 00	1	1	SSA_DISK_ERR2
No Seek Complete	1 02 00	1	1	SSA_DISK_ERR2
Peripheral Device Write Fault	1 03 00	1	1	SSA_DISK_ERR2

Table 41. SCSI Sense Key/Code/Qualifier recovery procedures (continued)

Description	SCSI K/C/Q	ERP#	Log	Template
Track Following Error	1 09 00	1	1	SSA_DISK_ERR2
Write Error Recovered With Auto Reallocation	1 0C 01	1	1	SSA_DISK_ERR3
Write Error - Recommend Reassignment	1 0C 03	2	1	SSA_DISK_ERR3
Record Not Found	1 14 01	1	1	SSA_DISK_ERR3
Record Not Found - Recommend Reassignment	1 14 05	2	1	SSA_DISK_ERR3
Record Not Found - Data Auto Reallocated	1 14 06	1	1	SSA_DISK_ERR3
Random Positioning Error	1 15 00	1	1	SSA_DISK_ERR2
Positioning Error Detected by Read of Medium	1 15 02	1	1	SSA_DISK_ERR2
Data Synchronization Mark Error	1 16 00	1	1	SSA_DISK_ERR3
Data Synchronization Mark Error - Data Rewritten	1 16 01	1	1	SSA_DISK_ERR3
Data Synchronization Mark Error - Recommend Rewrite	1 16 02	3	1	SSA_DISK_ERR3
Data Synchronization Mark Error - Data Auto-Reallocated	1 16 03	1	1	SSA_DISK_ERR3
Data Synchronization Mark Error - Recommend Reassignment	1 16 04	2	1	SSA_DISK_ERR3
Recovered Data With Retries	1 17 01	1	1	SSA_DISK_ERR3
Recovered Data with Positive Head Offset	1 17 02	1	1	SSA_DISK_ERR3
Recovered Data with Negative Head Offset	1 17 03	1	1	SSA_DISK_ERR3
Recovered Data using previous sector ID	1 17 05	2	1	SSA_DISK_ERR3
Recovered Data Without ECC - Data Auto-Reallocated	1 17 06	1	1	SSA_DISK_ERR3
Recovered Data Without ECC - Recommend Reassignment	1 17 07	2	1	SSA_DISK_ERR3
Recovered Data Without ECC - Recommend Rewrite	1 17 08	3	1	SSA_DISK_ERR3
Recovered Data Without ECC - Data Rewritten	1 17 09	1	1	SSA_DISK_ERR3
Recovered Data with Error Correction and Retries Applied	1 18 01	1	1	SSA_DISK_ERR3
Recovered Data - Data Auto-Reallocated	1 18 02	1	1	SSA_DISK_ERR3
Recovered Data - Recommend-Reassignment	1 18 05	2	1	SSA_DISK_ERR3
Recovered Data With ECC - Recommend Rewrite	1 18 06	3	1	SSA_DISK_ERR3
Recovered Data With ECC - Data Rewritten	1 18 07	1	1	SSA_DISK_ERR3
Primary Defect List Not Found	1 1C 01	12	1	SSA_DISK_ERR2
Grown Defect List Not Found	1 1C 02	12	1	SSA_DISK_ERR2
Partial Defect List Transferred	1 1F 00	12	1	SSA_DISK_ERR2
Internal Target Failure	1 44 00	1	1	SSA_DISK_ERR2
Spindles Not Synchronized	1 5C 02	15	2	SSA_DISK_ERR4

Table 41. SCSI Sense Key/Code/Qualifier recovery procedures (continued)

Description	SCSI K/C/Q	ERP#	Log	Template
Predictive Failure Analysis Threshold Reached on Recovered Error	1 5D 00	1	1	SSA_DISK_ERR4
Logical Unit Not Ready Cause Not Reportable	2 04 00	6	2	SSA_DISK_ERR4
Logical unit is in the process of becoming ready	2 04 01	7	2	SSA_DISK_ERR4
Logical Unit Not Ready, initialization command required	2 04 02	6	2	SSA_DISK_ERR4
Logical Unit Not Ready, Format in Progress	2 04 04	4	2	SSA_DISK_ERR4
Medium Format Corrupted Reassign Failed	2 31 00	8	1	SSA_DISK_ERR4
Format Command Failed	2 31 01	4	1	SSA_DISK_ERR4
Diagnostic Failure	2 40 80	8	2	SSA_DISK_ERR4
Diagnostic Failure	2 40 85	14	1	SSA_DISK_ERR4
Diagnostic Failure	2 40 B0	9	1	DISK_ERR4
Logical Unit Failed Self-Configuration	2 4C 00	8	2	SSA_DISK_ERR4
Write Error - Auto-Reallocation Failed	3 0C 02	5	1	DISK_ERR1
Write Error - Recommend Reassignment	3 0C 03	5	1	DISK_ERR1
Unrecovered Read Error	3 11 00	5	1	SSA_DISK_ERR2
Unrecovered Read Error - Auto Reallocation Failed	3 11 04	5	1	DISK_ERR1
Unrecovered Read Error - Recommend Reassignment	3 11 0B	5	1	DISK_ERR1
Recorded Entity Not Found	3 14 00	5	1	DISK_ERR1
Record Not Found	3 14 01	5	1	SSA_DISK_ERR2
Record Not Found - Recommend Reassignment	3 14 05	5	1	DISK_ERR1
Data Synchronization Mark Error	3 16 00	5	1	SSA_DISK_ERR2
Data Synchronization Mark Error - Recommend Reassignment	3 16 04	5	1	DISK_ERR1
Defect List Error in Primary List	3 19 02	4	1	SSA_DISK_ERR4
Defect List Error in Grown List	3 19 03	4	1	SSA_DISK_ERR4
Medium Format Corrupted Reassign Failed	3 31 00	4	1	SSA_DISK_ERR4
Format Failed	3 31 01	4	1	SSA_DISK_ERR4
Internal Target Failure	3 44 00	8	1	DISK_ERR1
No Index/Sector Signal	4 01 00	13	2	SSA_DISK_ERR4
No Seek Complete	4 02 00	13	2	SSA_DISK_ERR4
Peripheral Device Fault	4 03 00	13	2	SSA_DISK_ERR4
Track Following Error	4 09 00	13	2	SSA_DISK_ERR4
Unrecovered Read Error in Reserved Area	4 11 00	4	1	SSA_DISK_ERR4
Recorded Entity Not Found	4 14 00	13	2	SSA_DISK_ERR4
Record Not Found - Reserved Area	4 14 01	4	1	SSA_DISK_ERR4
Random Positioning Error	4 15 00	13	2	SSA_DISK_ERR4

Table 41. SCSI Sense Key/Code/Qualifier recovery procedures (continued)

Description	SCSI K/C/Q	ERP#	Log	Template
Positioning Error Detected by Read of Medium	4 15 02	13	2	SSA_DISK_ERR4
Data Synchronization Mark Error in Reserved Area	4 16 00	4	1	SSA_DISK_ERR4
Defect List Error in Primary List	4 19 02	4	1	SSA_DISK_ERR4
Defect List Error in Grown List	4 19 03	4	1	SSA_DISK_ERR4
Medium Format Corrupted Reassign Failed	4 31 00	5	1	SSA_DISK_ERR4
No Defect Spare Location Available	4 32 00	4	1	SSA_DISK_ERR4
Defect list update failure	4 32 01	4	1	SSA_DISK_ERR4
Diagnostic Failure	4 40 80	8	2	SSA_DISK_ERR4
Diagnostic Failure	4 40 85	14	1	SSA_DISK_ERR4
Diagnostic Failure	4 40 90	8	2	SSA_DISK_ERR4
Diagnostic Failure	4 40 A0	8	2	SSA_DISK_ERR4
Diagnostic Failure	4 40 B0	9	1	DISK_ERR4
Diagnostic Failure	4 40 C0	8	2	SSA_DISK_ERR4
Diagnostic Failure	4 40 D0	8	2	SSA_DISK_ERR4
Internal Target Failure	4 44 00	13	2	SSA_DISK_ERR4
Spindles Not Synchronized	4 5C 02	15	2	SSA_DISK_ERR4
Parameter List Length Error	5 1A 00	10	2	SSA_DISK_ERR1
Invalid Command Operation Code	5 20 00	10	2	SSA_DISK_ERR1
Logical Block Address out of Range	5 21 00	10	2	SSA_DISK_ERR1
Invalid Field in CDB	5 24 00	10	2	SSA_DISK_ERR1
Logical Unit Not Supported	5 25 00	10	2	SSA_DISK_ERR1
Invalid Field in Parameter List	5 26 00	10	2	SSA_DISK_ERR1
Not Ready To Ready Transition, (Medium may have changed)	6 28 00	9	0	-
Power On Reset, or Reset Message occurred	6 29 00	9	0	
Mode Parameters Changed	6 2A 01	9	0	-
Log Parameter Changed	6 2A 02	9	0	-
Commands Cleared by Another Initiator	6 2F 00	9	0	-
Microcode has been changed	6 3F 01	9	0	-
Spindles Synchronized	6 5C 01	13	0	-
Spindles Not Synchronized	6 5C 02	15	2	SSA_DISK_ERR4
Write Protected	7 27 00	4	2	SSA_DISK_ERR4
Internal Target Failure	B 44 00	13	2	SSA_DISK_ERR4
Invalid Message Error	B 49 00	10	2	SSA_DISK_ERR1
Overlapped Commands Attempted	B 4E 00	16	2	SSA_DISK_ERR1
Miscompare During Verify Operation	E 1D 00	4	1	SSA_DISK_ERR4
Invalid KCQ	x xx xx	4	2	SSA_DISK_ERR4

Appendix A. Identifier Values

This section supplies the values of the identifiers referenced elsewhere in this document. All values are decimal, except where shown as hexadecimal.

Registry Transactions

Name	Value
FN_REGY_SystemVersionInfo	10
FN_REGY_GatewayNodeList	11
FN_REGY_DriverGatewayNodeList	12
FN_REGY_ServiceList	13
FN_REGY_ConnectForNodeChange	14
FN_REGY_DiscForNodeChange	15
FN_REGY_NodeChangeToRegistry	16
FN_REGY_NodeChangeFromRegistry	17
FN_REGY_ConnectForErrorLogging	18
FN_REGY_DiscForErrorLogging	19
FN_REGY_LogErrorToRegistry	20
FN_REGY_LogErrorFromRegistry	21
FN_REGY_ConnectForResrcChange	22
FN_REGY_DiscForResrcChange	23
FN_REGY_ResrcChangeToRegistry	24
FN_REGY_ResrcChangeFromRegistry	25
FN_REGY_ResrcList	26
FN_REGY_GetTempResrcID	27
FN_REGY_ConnectForHealthCheck	28
FN_REGY_DiscForHealthCheck	29
FN_REGY_HealthCheckToRegistry	30
FN_REGY_HealthCheckFromRegistry	31
FN_REGY_SerialNumberSearch	32
FN_REGY_TestResrcsReady	33
FN_REGY_SetClusterNumber	34
FN_REGY_TestOneResrcReady	35
FN_REGY_SyncHCheckToRegy	36
FN_REGY_SyncHCheckFromRegy	37
EV_Rebooting	1
EV_NodeDead	2
EV_Rebooted	3
EV_NodeUnreliable	4
RS_Offline	1
RS_Online	2
CC_Add	01h
CC_SetOnline	02h
CC_SetOffline	04h
CC_Remove	08h
TY_Disk	0
TY_Adapter	1
SR_NoSynchro	0
SR_Synchro	1

Name	Value
LP_Unknown	0
LP_Loop	1
LP_String	2
LG_Unknown	0
LG_Legal	1
LG_Illegal	2
MN_Unknown	0
MN_Master	1
MN_NonMaster	2
SD_Code	0
SD_CodeAsn	1
SD_CodeAsnCsn	2

ISAL Transactions

Name	Value
FN_ISALMgr_Inquiry	40
FN_ISALMgr_HardwareInquiry	41
FN_ISALMgr_SetOwningModuleType	42
FN_ISALMgr_AssignManualResrcID	43
FN_ISALMgr_GetPhysicalResrcIDs	44
FN_ISALMgr_TestResrcsReady	45
FN_ISALMgr_VPDInquiry	46
FN_ISALMgr_Characteristics	47
FN_ISALMgr_Statistics	48
FN_ISALMgr_FlashIndicator	49
FN_ISALMgr_Open	50
FN_ISAL_Close	51
FN_ISAL_Read	52
FN_ISAL_Write	53
FN_ISAL_Format	54
FN_ISAL_Progress	55
FN_ISAL_Lock	56
FN_ISAL_Unlock	57
FN_ISAL_Test	58
FN_ISAL_SCSI	59
FN_ISAL_Download	60
FN_ISALMgr_QueryFilterType	61
FN_ISAL_Fence	62
FN_ISALMgr_TestOneResrcReady	63
FN_ISALMgr_Get PhysSvcAndRIDs	64
FN_ISALMgr_ServiceMode	65
FN_ISALMgr_NetworkInquiry	66
FN_ISALMgr_Preferences	67
FN_ISAL_Flush	68
FN_ISALMgr_LockQuery	69
MD_ISAL	0
MD_SCSI	1
MD_Service	2
MD_ISAL_HA	3

Name	Value
AT_All	0
AT_ReadOnly	1
AT_WriteOnly	2
SM_DenyNone	0
SM_DenyAll	1
SM_DenyWrite	2
SM_DenyRead	3
SM_DenyNothing	4
FF_Verify	01h
FF_NoCache	02h
FF_PreFetch	04h
FF_Split	08h
FF_ReadDisk	10h
FF_FastWrite	20h
FF_ISALReservedArea	40h
FF_ReassignWrite	80h
EF_NoCache	01h
EF_RelocationArea	02h
RF_ReassignWarn	01h
RF_DriveWarn	02h
RF_RewriteWarn	04h
RF_BlockWarn	08h
UL_Normal	0
UL_Forced	1
LT_Normal	0
LT_AdapID	1
ST_Good	0
ST_Failed	1
ST_LossRedundancy	2
ST_FormatInProgress	3
VP_NoEVPD	0
VP_EVPD	1
TT_Test	0
TT_Diag	1

Name	Value
AE_HardwareError	−4
AE_ParityNotValid	−6
AE_MediaReadOnly	−7
AE_IllegalRequest	−100 to −150
AE_TableFull	−9
AE_InvalidRID	−10
AE_InvalidSignature	−11
AE_AccessDenied	−12
AE_NotReady	−13
AE_ReservationConflict	−14
AE_WriteProtect	−15
AE_NotInTable	−16
AE_Offline	−17
AE_MediumError	−18
AE_SCSIError	−20
AE_LogOpen	−21
AE_FencedOut	−22
AE_FormatDegraded	−23
AE_FormatInProgress	−24
AE_ClusterNumberUnknown	−25
AE_SSAString	−26
AE_AvoidReadMe	−27
AE_AvoidWrite	−28
AE_NotLocal	−29
AE_NotYetImplemented	−30
AE_RetryWhenMemory	−31
AE_NotFound	−32
AE_FlushCompFailure	−33
AE_InServiceMode	−34
AE_OfflineTimeout	−35
AE_BadSequenceNumber	−36
AE_BadTransferLRC	−37
AE_BadblockLRC	−38
AE_NotSupported	−40
AE_NonIsal	−41
AE_OtherAdapterInServiceMode	−42

Name	Value
AE_BadSerialNumber	−500+0
AE_BadOldSerialNumber	−500+1
AE_BadNewSerialNumber	−500+2
AE_BadComponentCount	−500+3
AE_BadComponentSerialNumber	−500+4
AE_BadResrcSerialNumber	−500+5
AE_BadOldComponentSerialNumber	−500+6
AE_BadNewComponentSerialNumber	−500+7
AE_BadParameterValues	−500+8
AE_ArrayIsBroken	−500+9
AE_SetOMTFailed	−500+10
AE_BadExchangeCandidate	−500+11
AE_FiltersOnly	−500+12
AE_NotFilters	−500+13
AE_NvramError	−500+14
AE_InvalidCandidateRequest	−500+15
AE_InvalidCreateRequest	−500+16
AE_ReadOnlyParameterValue	−500+17
AE_ArrayIsBrokenOrDegraded	−500+18
AE_Physwrapped	−500+19
AE_PhysCertifying	−500+20
AE_PhysFormatting	−500+21
AE_NotOpen	−500+22
AE_BadComponent	−500+23
AE_ConfirmRequired	−500+24
OM_NotOwned	1
OM_DriverAdapters	2
OM_DriverPhysical	3
OM_DriverManualDisk	4
OM_DriverAutomaticDisk	5
OM_FirstFilter	E
OM_Raid5Filter	K
OM_Disowned	W
OM_NvramEntry	X
OM_HotSpareDisk	Y
HF_Unknown	0
HF_MotorFail	1
HI_NotImmediate	0
HI_Immediate	1
DL_NoSave	0
DL_Save	1
FF_Normal	0
FF_Force	1
FL_FenceOut	1
FL_FenceIn	2
FC_Add	1
FC_Remove	2
FM_Change	0
FM_CompareAndSwap	1

Adapter Services

Name	Value
FN_ADAP_TransferFromHost	80
FN_ADAP_TargetTransfer	83
FN_ADAP_TransferToHost	81
FN_ADAP_ConnectForTransfer	84
FN_ADAP_DisconnectForTransfer	85
FN_ADAP_AdapterHealthCheck	90

Service / Transaction Directives

Name	Value
DC_StartTransaction	19
DT_Illegal	0
DT_VAddress	1
DT_VChain	2
DT_VScatGatV	3
DT_VScatGatP	4
DT_VTcb	5
DT_VMonitor	6
DT_PAddress	7
DT_PScatGatP	8
DT_Indirect	9
DT_Invalid	10
DT_Null	11
DT_Microchannel	12
DT_MicrochannelScatGat	13
DT_Cfe	14
DT_ADDScatGatP	15
DT_Xmd	16
DT_WindowOffset	17
DT_ADDBaseVP	18
DT_Buf	19
DT_VAddressScatGat	20
DT_VMicrochannel	21
MF_System	1
MF_Application	2
MF_Gateway	3
AS_Warning	1
AS_Success	0
AE_UnknownFunction	-1
AE_Busy	-2
AE_Failure	-3
AE_NotYetImplemented	-30
AE_RetryWhenMemory	-31
AE_NotFound	-32

Name	Value
OT_XferTCB	0
OT_Parms	1
OT_Fetch	2
OT_Store	3
OT_Status	4
OT_Done	5
OT_FastDone	6
FF_Exclusive	1
DC_SlaveInstallService	23
SN_Router	0
SN_Registry	1
SN_TimeServer	2
SN_ErrorLogger	3
SN_SSAGS	4
SN_SSADS	5
SN_CfgAgent	6
SN_AdapterService	7
SN_WAManager	12
TP_Unused	0
TP_Unknown	1
TP_Router	2
TP_ISAL	3
TP_FileSystem	4
TP_Database	5
TP_Resource	6
TP_Registry	7
TP_TimeServer	9
TP_ErrorLogger	10
TP_SSAGS	11
TP_SSADS	12
TP_Window	13
TP_BlowTorch	14
TP_CfgAgent	15
TP_AdapterService	16
TP_Debug	17
TP_Nvram	18

Node Numbers

Name	Value	
NN_Local	0	The local node
NN_Host	1	The host
NN_AdapterA	11	First Adapter on bus 1
NN_AdapterAEnd	18	Last Adapter on bus 1
NN_AdapterB	21	First Adapter on bus 2
NN_AdapterBEnd	28	Last Adapter on bus 2
NN_DaughterA	31	First Daughter on bus 1
NN_DaughterAEnd	38	Last Daughter on bus 1
NN_DaughterB	41	First Daughter on bus 2
NN_DaughterBEnd	48	Last Daughter on bus 2
NN_UserStart	49	First User mode node
NN_UserEnd	63	Last User mode node
NN_PeerStart	64	
NN_PeerEnd	95	

Configuration / Array Identifiers

Name	Value
FN_IACL_Command	101
FN_IACL_Register	102
FN_IACL_Unregister	103
FT_NotOwned	A
FT_DriverAutomaticDisk	B
FT_DriverManualDisk	C
FT_PhysicalDisk	D
FT_FastWriteFilter	E
FT_Raid5Filter	K
FT_PartitioningFilter	L
FT_Disowned	W
FT_HotSpareDisk	Y
FT_BlankReserved	Z
FT_Adapter	Z+1

Name	Value
FC_IACLVersion	1
FC_ResrcCount	2
FC_ResrcList	3
FC_ResrcView	4
FC_CandidateCount	5
FC_CandidateList	6
FC_ResrcCreate	7
FC_ResrcDelete	8
FC_ResrcRename	9
FC_ComponentView	10
FC_ComponentExchange	11
FC_QueryMetaResrcParams	12
FC_ModifyResrcParams	13
FC_FlashIndicator	14
FC_VPDInquiry	15
FC_HardwareInquiry	16
FC_ComponentExchCandCount	17
FC_ComponentExchCandList	18
FC_AdapterVPD	19
FC_SyncHealth	20
FC_Wrap	21
FC_Unwrap	22
FC_UnwrapAll	23
FC_Test	24
FC_Format	25
FC_Certify	26
FC_Read	27
FC_Write	28
FC_AdapterSN	29
FS_CandOnline	20
FS_CandOffline	21
FS_ResrcOffline	40
FS_ResrcOnline	41
FS_ResrcOnlineNonDeg	42
FS_ResrcOnlineDeg	43
FS_ResrcOnlineRebuild	44
FS_ResrcOnlineExposed	45
FS_ResrcFormatting	50
FS_ResrcFormatFailed	51
FS_ResrcCertifying	52
FS_ResrcCertifyFailed	53
FS_CompNotPresent	60
FS_CompNotPresentDeconf	61
FS_CompNotPresentBlank	62
FS_CompPresent	65
FS_CompPresentRebuild	66
FS_CompPresentRebuildMe	67
FS_ResrcInUse	68
FS_ResrcDormant	69
FS_ResrcInUse	70 InUse (X%used)
FS_ResrcDormant	71 Dormant (X%used)

Name	Value
FS_ResrcWrapped	80 Wrapped, in service mode
SDS_BLOCKSIZE	1
SDS_DISK_NUMBER	2
SDS_STRIPE_SIZE	3
SDS_STRETCH_SIZE	4
SDS_MODE_FLAGS	5
SDS_STRIDE_SIZE	6
SDS_HOT_SPARE_ENABLED	7
SDS_HOT_SPARE_EXACT_SIZE	8
SDS_REBUILD_PRIORITY	9
SDS_SPEC_READ	10
SDS_DATA_SCRUB_ENABLED	11
SDS_DATA_SCRUB_RATE	12
SDS_SIZE	13
SDS_CACHE_SIZE	14
SDS_INITIALIZE	15
SDS_DELAY	16
SDS_SPLIT_RESOLUTION	17
SDS_WRITE_IN_DEG	18
SDS_LAZY_PARITY_WRITE	19
SDS_PAGE_ALIGN_SPLIT	20
SDS_NETWORK_ID	21 Network ID
SDS_GEOM_SECT	22 Geometry hint
SDS_READ_CACHE_DISABLE	23 Read cache disabled
SDS_READ_AHEAD_ENABLE	24 Read ahead enabled
SDS_BAD_PTY_STRIDE	25
SDS_BAD_COMPONENT_STRIDE	26
SDS_BAD_STRIPE	27
SDS_MIN_LBA	28 Fastwrite min LBA of cached LBA range
SDS_MAX_LBA	29 Fastwrite max LBA of cached LBA range
SDS_MAX_WRITE_LENGTH	30 Fastwrite max length of writes cached
SDS_ALLOW_DELETE	31 Fastwrite allow data in cache to be deleted
SDS_MIRROR_ENABLE	32 Fastwrite snoop data into battery- backed SRAM
SDS_SKIP_WR_REBUILD	33 Skip Write Rebuild
SDS_SPLIT_CONFIRM	34 Split Confirm (RAID-1 only)
SDS_FAST_DECONFIGURE	35 Fast Deconfigure
SDS_CACHE_FSW	36 Set if full stride writes are to be cached
SDS_BLOCK_LRC	37 Block LRC protection

Name	Value
SDSF_MB	00000001h
SDSF_KB	00000002h
SDSF_PERCENT	00000004h
SDSF_ON_OFF	00000008h
SDSF_BYTES	00000010h
SDSF_SECONDS	00000020h
SDSF_MINUTES	00000040h
SDSF_HOURS	00000080h
SDSF_READONLY	80000000h
SDSF_UNIQUE	40000000h

Appendix B. Notices

References in this book to IBM products, programs, or services do not imply that IBM intends to make these available in all countries in which IBM operates. Any reference to an IBM product, program, or service is not intended to state or imply that only that IBM product, program, or service may be used. Subject to IBM's valid intellectual property or other legally protectable rights, any functionally equivalent product, program, or service may be used instead of the IBM product, program, or service. The evaluation and verification of operation in conjunction with other products, except those expressly designated by IBM, are the responsibility of the user.

Any examples of parameters or definitions are for guidance only. Some details may differ from the requirements in your environment. Contact your IBM representative if you need further assistance.

IBM may have patents or pending patent applications covering subject matter in this document. The furnishing of this document does not give you any license to these patents. You can send license enquiries, in writing, to the IBM Director of Licensing, IBM Corporation, 500 Columbus Avenue, Thornwood, New York 10594, U.S.A.

Trademarks

The following terms are trademarks of International Business Machines Corporation in the United States, or other countries, or both:

IBM
RS/6000
Micro Channel

Other company, product, and service names may be trademarks or service marks of others.

Index

Numerics

A

B

C

D

E

F

G

H

I

L

M

N

O

P

Q

R

S

T

U

V

W

Printed in the U.S.A.

SA33-3225-02