COVER Book Cover

IBM SSA RAID Cluster Adapter
for PC Servers

Technical Reference

Document Number SA33-3275-01

ABSTRACT Abstract

This book gives technical reference information about the SSA RAID Cluster Adapter. This adapter can be included in PC Servers.

The book describes:

- The major components of the adapte
- The interface between the adapter and the PCI bus in the syste
- The interface between the adapters and the SSA devices that are attached to the
- The transactions used to communicate between the elements of the whole SSA subsystem, including the device-driver programs.

NOTICES Notices

Note!

Before using this information and the product it supports, be sure to read the general information under Appendix B, "Notices" in topic B.0.

EDITION Edition Notice

Second Edition (October 1997)

The following paragraph does not apply to any country where such provisions are inconsistent with local law: THIS PUBLICATION IS PRINTED "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. Some states do not allow disclaimer of express or implied warranties in certain transactions; therefore, this statement may not apply to you.

This publication could contain technical inaccuracies or typographical errors. Changes are periodically made to the information herein; these changes will be incorporated in new editions of the publication.

It is possible that this publication may contain reference to, or information about, products (machines and programs), programming, or services that are not announced in your country. Such references or information must not be construed to mean that such products, programming, or services will be offered in your country.

Note to U.S. Government Users -- Documentation related to restricted rights -- Use, duplication or disclosure is subject to restrictions set forth in GSA ADP Schedule Contract with IBM Corp.

CONTENTS Table of Contents

FIGURES Figures

FRONT_1 Conventions

Subtopics

FRONT_1.1 Bits

The SSA RAID Cluster Adapter uses 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

FRONT_1.2 Bytes

Except as noted below, the SSA RAID Cluster Adapter 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.

Because the processors in personal computers also use the Little-endian convention, device drivers do not need to take any special action.

When using a SSA RAID Cluster 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.

FRONT_1.3 Words

In a SSA RAID Cluster Adapter, a **word** is 4 bytes.

FRONT_1.4 Registers

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

FRONT_1.5 Serial links

When a SSA RAID Cluster 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.)

1.0 Chapter 1. Description

Subtopics

1.1 Introduction to the Adapter

The IBM SSA RAID Cluster Adapter for PC Servers is a PCI bus-master adapter that serves as the interface between systems using the Peripheral Component Interconnect (PCI) architecture and devices using the Serial Storage Architecture (SSA). The adapter provides high-performance implementation of RAID-1 arrays. The adapter supports Windows NT two-way clustering using a pair of SSA RAID Cluster Adapters, one in each of two PC Servers. Chapter 6, "Clustering" in topic 6.0 describes the clustering function in more detail. The adapter includes a DRAM data buffer with a storage capacity of 32 MB, and a nonvolatile parity store of 8 KB with checking implemented by firmware.

The adapter provides 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. 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 SSA dual-port disk drives and another SSA RAID Cluster Adapter in a closed loop. The disk drives can be configured as non-RAID disks (which can be accessed individually by the using systems), as members of RAID-1 arrays, or as hot spares.

If an SSA loop is broken by a fault, the adapter 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 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.

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

The device driver for the adapter and its subsystem handle SCSI commands, status, and sense. The adapter also deals with the SSA protocols and it recovers link errors and some disk errors internally. The device driver supports Microsoft Windows NT 4.0 Server or Microsoft Windows NT 4.0 Server/Enterprise Edition which includes Microsoft Cluster Server. Microsoft Cluster Server handles failover between clustered adapters.

The maximum configurations are:

- Under Microsoft Cluster Server, for a pair of servers, the total number of non-RAID disks and RAID-1 arrays can be up to 22. If no disk drives are configured as non-RAID disks, up to 44 disk drives can be configured as members of RAID-1 arrays, in addition to any disks that are configured as hot spares.
- Without Microsoft Cluster Server, for a pair of servers, up to 96 RAID-1 arrays can be configured; that is, up to 32 arrays on each of three pairs of adapters.

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 RAID-1 array functions are implemented as an IPN filter 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 can control disk drives individually or when configured in arrays. The system can boot from a disk attached to an adapter.

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 adapter provides a Compatibility Basic Input/Output Subsystem (CBIOS) that can be used to boot a system.

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. 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 is 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 40 MB/s when transferring data to an array. See 6.3 for more details of performance in a cluster.

1.2 Device Driver and Utility Programs

Two diskettes are provided with the adapter:

PC-DOS Configurator and Utilities diskett

This contains:

- Bootable DOS program
- SSA Configuration and Service Aids utility (not all service-aid functions are available under DOS)
- Adapter microcode download utility
- Disk drive microcode download utility

Windows NT Device Driver and Utilities diskett

This contains:

- Windows NT Device Driver
- SSA Configuration and Service Aids utility
- Adapter microcode download utility
- Disk drive microcode download utility

The device driver includes:

Installation progra
README for installatio
Error/event logger

The SSA Configuration and Service Aids utility has character-based menu screens and can be used concurrently with other functions. It provides the following configuration functions:

Create and delete array
List and identify members of array
Exchange members of array
View attributes of arrays and disk drive
Change attributes of array
Change use of disk drive
Attach arrays and disk drives to the syste
View and change NVRA

The service aid functions are:

Analyze and view error/event log entrie
Set disk drives into service mod
Run concurrent diagnostic
Run nonconcurrent diagnostic
Display Vital Product Data (VPD) for the adapter and disk drive
Format disk drive
Certify disk drives

1.3 Adapter Functions

The principal functions of the SSA RAID Cluster 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 RAID-1 read operations, the data is transferred from disk to host memory directly; for write operations, the data is transferred via the data buffer. 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.

When accessing RAID-1 arrays in a clustered environment, only one of the adapters (the one with the higher SSA Unique ID) controls the arrays. I/O requests for an array from the PC server containing the other adapter are passed for processing from its adapter to the adapter that controls the arrays. For such operations, the originating adapter is used to fetch and store data from its PC server and to return the results of the I/O request but not to control the array.

1.4 Disk Arrays

The adapter provides RAID-1 functions by means of a filter between the device driver and the disk drives. The filter is implemented by microcode that runs in the adapter. It presents the image of a single logical disk drive to the device driver and uses two components to implement this image. Up to 32 arrays can be configured on a single adapter. Optional hot spares can be configured that are shared between all the arrays.

Chapter 5, Array Filters defines these functions in more detail.

1.5 Supported Standards

The SSA RAID Cluster Adapter implements the standards described in the following documents:

PCI Local Bus Specification, production version, revision 2.1

Serial Storage Architecture, 1995 Physical (SSA-IA/95PH), October 1995

Serial Storage Architecture, 1995 SCSI-2 Protocol (SSA-IA/95SP), October 1995

Small Computer System Interface - 2 (SCSI-2), X3.131.199X, Revision 10m.

1.6 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 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.

```
  HOST OPERATING SYSTEM
          ¦
+---------¦----------------------------------------+
¦ +-----------------+ +-------------+ +----------+ ¦
¦ ¦ INTERFACE       ¦ ¦ERROR LOGGER ¦ ¦REGISTRY  ¦ ¦
¦ ¦ IPN master      ¦ ¦IPN slave    ¦ ¦IPN slave ¦ ¦
¦ +-----------------+ +-------------+ +----------+ ¦
¦      MCB¦                                        ¦
¦ +-------¦--------------------------------------+ ¦
¦ ¦       ¦         IPN KERNEL                   ¦ ¦  +-------------+
¦ +-------¦--------------------------------------+ ¦  ¦ DISK DRIVE  ¦
¦      MCB¦                                        ¦  ¦             ¦
¦ +-----------------+                              ¦  +-------------+
¦ ¦ PCI             ¦                              ¦         ¦
¦ ¦ GATEWAY         ¦                              ¦         ¦
¦ ¦                 ¦               DEVICE DRIVER  ¦         ¦SSA
+-¦                 ¦------------------------------+         ¦
  ¦ Delivery pipe   ¦                                        ¦
+-¦                 ¦----------------------------------------¦----------+
¦ ¦                 ¦                               +-----------------+ ¦
¦ +-----------------+                               ¦ SSA DRIVER      ¦ ¦
¦         ¦                                         ¦                 ¦ ¦
¦      TCB¦                                         +-----------------+ ¦
¦         ¦                               +------------+     ¦          ¦
¦         ¦  +--------+   +-----+     +-------+     +-----------------+ ¦
¦         ¦  ¦REGISTRY¦   ¦ CFG ¦     ¦ PEER  ¦     ¦ DISK SERVICE    ¦ ¦
¦         ¦  ¦        ¦   ¦AGENT¦     ¦GATEWAY¦     ¦ IPN slave       ¦ ¦
¦         ¦  +--------+   +-----+     +-------+     +-----------------+ ¦
¦         ¦      ¦           ¦            ¦                  ¦          ¦
¦ +------------------------------------------------------------------+  ¦
¦ ¦                         IPN KERNEL                               ¦  ¦
¦ +------------------------------------------------------------------+  ¦
¦                      +------+                                         ¦
¦                      ¦RAID-1¦                                         ¦
¦                      +------+                                         ¦
¦                                                                       ¦
¦ ADAPTER                                                               ¦
+-----------------------------------------------------------------------+
```

Figure 1-1. IPN Components

The SSA RAID Cluster Adapter contains a PCI gateway, a disk service, a registry service, an SSA driver, the IPN kernel, and a service for the RAID-1 function, as shown in Figure 1-1. A typical transaction to read data from a RAID-1 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-1 service for the addressed resource with a pointer to the TCB.

6. The RAID-1 service generates an IPN transaction for the disk drive, of the pair, that it decides to use and sends this transaction 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 command to the disk drives using the SSA protocol.

9. When the disk drive offers the requested data, the SSA driver transfers the data to host memory through the PCI gateway.

10. When the drive returns good-completion status, the disk service calls IPN with the result of the transaction generated by the RAID-1 service.

11. The RAID-1 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.

If the transaction is sent to the adapter that does not control the RAID-1 array (that is, the one with the lower SSA Unique ID), that adapter sends the transaction to the other adapter for processing.

2.0 Chapter 2. System-to-Adapter Interface

Subtopics

2.1 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.

Subtopics

2.1.1 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 2-1. Format of a GTCB

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

Not Yet Valid (N) Byte 3 bit 7 is the not-yet-valid 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 number for the disk service is dynamically allocated and can be obtained from the registry.

Major_function This byte is coded as follows:

02h **Application**. These transactions are defined separately by each service. See "Disk Service" in topic 7.4 for the transactions supported by the adapter disk service, "Registry Service" in topic 7.3 for the transactions supported by the registry service, and "Array-Configuration Service" in topic 7.5 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)" in topic 2.1.2 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" in topic 2.1.4.)

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. Byte 96 bit 1 is reserved. (On another SSA RAID adapter it is defined as the not-yet-valid flag.)

2.1.2 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 2-2. 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 before the offset is applied.

2.1.3 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 2-3. 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.

2.1.4 Result Word

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

Table 2-4. 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" in topic 7.6.

2.1.5 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.

2.1.6 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" in topic 2.1.7). The length fields in each DDR indicate if any data needs to be transferred.

```
Byte
     +---------------------------+
  0  ¦                           ¦
     ¦     Undefined (16 bytes)  ¦
 12  ¦                           ¦
     +---------------------------¦
 16  ¦                           ¦
     ¦                           ¦
     ¦                           ¦
     ¦                           ¦
     ¦           GTCB            ¦
     ¦     (First 80 bytes)      ¦
     ¦                           ¦
     ¦                           ¦
     ¦                           ¦
 92  ¦                           ¦
     +---------------------------+
      --------- 4 bytes ---------_
```

Figure 2-1. 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 2-2.

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

```
Contents:                                    Pointers:
            +-------------------+
Free        ¦      -      ¦ 001 ¦  -- Q_start   (First Element)
            +-------------+-----¦
Reply       ¦GTCB pointer ¦ 011 ¦  -- Deq_ptr   (Next Element to dequeue,
            +-------------+-----¦                internal to destination)
Request     ¦ TCB token   ¦ 001 ¦
            +-------------+-----¦
Request     ¦ TCB token   ¦ 001 ¦
            +-------------+-----¦
Free        ¦      -      ¦ 000 ¦  -- Enq_ptr   (Next Element to enqueue,
            +-------------+-----¦                internal to source)
Free        ¦      -      ¦ 010 ¦
            +-------------+-----¦
Free        ¦      -      ¦ 000 ¦
            +-------------+-----¦
Free        ¦      -      ¦ 000 ¦  -- 4xQ_length+  (Last Element)
            +-------------------+     Q_start-4
             ---- 4 bytes ------_
```

Figure 2-2. 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 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

2.1.7 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 2-5. Format of a Host-Slave-Operation Control Block

Byte	**3**	**2**	**1**	**0**
0	Operation Type			
4	TCB Token			
8	Offset			
12 . . 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)" in topic 2.1.2.

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.

2.1.8 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.

2.2 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.

Subtopics

2.2.1 Initialize

The Initialize command configures the delivery pipes used by IPN transactions and allocates an IPN node number to the adapter. "Shared Runtime Registers" in topic 3.4 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 2-6. Parameter block for Initialize

Byte	**3**	**2**	**1**	**0**
0	Adapter_Q_start			00
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.

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.

2.2.2 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 2-7. Parameter Block for Download

Byte	**3**	**2**	**1**	**0**
0	G ¦ Reserved =0000000b	Reserved = 00h	SG_length	
4	Address			
8	Length			
12	LRC			
16 ... 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.

2.2.3 Execute I/O

The Execute I/O command provides a simple synchronous I/O interface to support system IPL and software installation.

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 adapter could not execute the command successfully because of an error in the I/O device or an attachment error.
The adapter has detected a catastrophic error, for example, invalid parameter. The code in the error register defines the reason for the error.

Table 2-8. 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.

Resource ID The 4-byte Resource ID of 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, array 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.

2.2.4 Independent Execute I/O

This command is identical to Execute I/O except:

Completion of the command (successfully or unsuccessfully) does not cause a PCI interrupt

The completion of the command is indicated by IEXECIO register bit 31 = 1b (see"IEXECIO Register" in topic 3.4.6). The host must set IEXECIO register bit 31 to 0b before issuing the command.

Unsuccessful completion of the command is indicated by IEXECIO register bit 30 = 1b when bit 31 = 1b. The adapter error code is then in bits 7 through 0. If the adapter encountered a catastrophic error, it sets Adapter Catastrophic Error in the PCI Interrupt Doorbell register.

The command register value for Independent Execute I/O is 33h.

2.3 Resets

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

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

Table 2-9. SSA RAID Cluster Adapter Reset Actions

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.

2.4 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 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. The value 10 is returned for this 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 SSA RAID Cluster 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.

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) 1 0
* 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
```

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 31 30
¦  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
```

3.0 Chapter 3. PCI Interface

Subtopics

3.1 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 only use 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.

The adapter provides a Compatibility BIOS (called Local Expansion ROM). This is sufficient to install Windows NT and boot a PC system from a hard disk attached to the adapter.

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.

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" in topic 3.2.

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" in topic 3.3.

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" in topic 3.4.

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.

3.2 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.

Subtopics

3.2.1 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

```
 31                                15                            0
+--------------------------------------------------------------+
¦          Device ID              ¦        Vendor ID           ¦
+--------------------------------------------------------------+
```

Figure 3-1. PCI Configuration ID Register

Table 3-1. PCI Configuration ID Register

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

3.2.2 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

```
 15        10 9 8 7 6 5 4 3 2 1 0
+--------------------------------+
¦ Reserved   ¦ ¦ ¦ ¦ ¦ ¦ ¦ ¦ ¦ ¦ ¦
+--------------------------------+
              ¦ ¦ ¦ ¦ ¦ ¦ ¦ ¦ ¦ +-- I/O Space
              ¦ ¦ ¦ ¦ ¦ ¦ ¦ ¦ +---- Memory Space
              ¦ ¦ ¦ ¦ ¦ ¦ ¦ +------ Master Enable
              ¦ ¦ ¦ ¦ ¦ ¦ +-------- Special Cycle
              ¦ ¦ ¦ ¦ ¦ +---------- Memory Write/Invalidate
              ¦ ¦ ¦ ¦ +------------ VGA Palette Snoop
              ¦ ¦ ¦ +-------------- Parity Error Response
              ¦ ¦ +---------------- Wait Cycle Control
              ¦ +------------------ SERR# Enable
              +-------------------- Fast Back-to-Back Enable
```

Figure 3-2. PCI Command Register

Table 3-2. 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.	Yes	0

	This bit (and bit 6) must be on to report address parity errors.		
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 ... 10	Reserved	No	0

3.2.3 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

```
 15 14 13 12 11 10 9  8  7  6       0
+-----------------------------------+
¦  ¦  ¦  ¦  ¦  ¦     ¦  ¦  ¦ Reserved¦
+-----------------------------------+
  ¦  ¦  ¦  ¦  ¦   ¦    ¦  +------------ Fast Back-to-Back Capable
  ¦  ¦  ¦  ¦  ¦   ¦    +--------------- Data Parity Detected
  ¦  ¦  ¦  ¦  ¦   +-------------------- DEVSEL Timing
  ¦  ¦  ¦  ¦  +------------------------ Target Abort
  ¦  ¦  ¦  +--------------------------- Received Target Abort
  ¦  ¦  +------------------------------ Received Master Abort
  ¦  +--------------------------------- Signaled System Error
  +------------------------------------ Detected Parity Error
```

Figure 3-3. PCI Status Register

Table 3-3. PCI Status Register

Bits	**Description**	**Writeable**	**Initial value**
6 ... 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 ... 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	Yes	0

	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.		
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
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

3.2.4 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

```
 7              0
+----------------+
¦ Revision ID    ¦
+----------------+
```

Figure 3-4. PCI Revision ID Register

3.2.5 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

```
 23                15               7            0
+-------------------------------------------------+
¦  Base Class    ¦  Sub Class    ¦ Prog I/face  ¦
+-------------------------------------------------+
```

Figure 3-5. PCI Class Code Register

Table 3-4. PCI Class Code Register

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

3.2.6 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

```
 7              0
+----------------+
¦ Cache Line Size¦
+----------------+
```

Figure 3-6. PCI Cache Line Size Register

3.2.7 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

```
 7              0
+----------------+
¦ Latency Timer  ¦
+----------------+
```

Figure 3-7. 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.

3.2.8 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

```
 7              0
+----------------+
¦ Header Type    ¦
+----------------+
```

Figure 3-8. PCI Header Type Register

3.2.9 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.

```
 7 6 5 4 3      0
+----------------+
¦ ¦ ¦0¦0¦        ¦
+----------------+
 ¦ ¦        +---- Completion Code
 ¦ +------------- Start BIST
 +--------------- BIST Capable
```

Figure 3-9. PCI Built-In Self Test Register

Table 3-5. PCI Built-In Self Test (BIST) Register

Bits	**Description**	**Writeable**	**Initial value**
3 ... 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 ... 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

3.2.10 PCI Base Address Register for Memory Access to Runtime Registers

PCI address: Offset 10h
CFE Address: Offset 10h
Initialized value: 00000000h

```
 31                                                     3 2 1 0
+------------------------------------------------------------+
¦            Memory Base Address                        ¦ ¦   ¦ ¦
+------------------------------------------------------------+
                                                         ¦  ¦   ¦
                                   Prefetchable  --------+  ¦   ¦
                                       Location  -----------+   ¦
                                   Memory Space  ---------------+
```

Figure 3-10. Base Address Register for Memory Access to Runtime Registers

Table 3-6. 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 ... 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 ... 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 ... 7	**Memory Base Address.** This is the memory base address that is used to access the adapter runtime registers.	Yes	0

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

PCI address: Offset 14h
CFE Address: Offset 14h
Initialized value: 00000001h

```
 31                                                          2 1 0
+----------------------------------------------------------------+
¦                  I/O Base Address                        ¦0¦  ¦
+----------------------------------------------------------------+
```

Figure 3-11. Base Address Register for I/O Access to Runtime Registers

Table 3-7. 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 ... 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
31 ... 7	**I/O Base Address.** This is base address for I/O access to the adapter runtime registers.	Yes	0

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

PCI address: Offset 18h
CFE Address: Offset 18h
Initialized value: 00000000h

```
 31                                                     3 2 1 0
+----------------------------------------------------------------+
¦            Memory Base Address                        ¦ ¦   ¦ ¦
+----------------------------------------------------------------+
                                                          ¦  ¦  ¦
                                  Prefetchable  --------+  ¦  ¦
                                      Location  -----------+  ¦
                                  Memory Space  --------------+
```

Figure 3-12. Base Address Register for Memory Access to I/O Local Address Space

Table 3-8. 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 ... 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 ... 4	**Memory Base Address.** This is the memory base address that is used to access the adapter local address space.	Yes	0

3.2.13 PCI Expansion ROM Base Register

PCI address: Offset 30h
CFE Address: Offset 30h
Initialized value: 00000001h

```
 31                                     11 10                1 0
+----------------------------------------------------------------+
¦     Expansion ROM Base Address          ¦     Reserved      ¦ ¦
+----------------------------------------------------------------+
                                                               ¦
                                   Address Decode Enable ------+
```

Figure 3-13. PCI Expansion ROM Base Register

Table 3-9. 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 ... 1	Reserved	No	0
31 ... 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

3.2.14 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

```
 7              0
+----------------+
¦ Interrupt Line ¦
+----------------+
```

Figure 3-14. PCI Interrupt Line Register

3.2.15 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

```
 7              0
+----------------+
¦ Interrupt Pin  ¦
+----------------+
```

Figure 3-15. PCI Interrupt Pin Register

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

3.2.16 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

```
 7               0
+----------------+
¦    Min_Gnt     ¦
+----------------+
```

Figure 3-16. 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.

3.2.17 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

```
 7              0
+----------------+
¦    Max_Lat     ¦
+----------------+
```

Figure 3-17. PCI Max_Lat Register

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

3.3 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.

Subtopics

3.3.1 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

```
 31                                                       3 2 1 0
+----------------------------------------------------------------+
¦            Memory Base Address                        ¦ ¦   ¦ ¦
+----------------------------------------------------------------+
                                                         ¦  ¦  ¦
                                   Prefetchable  --------+  ¦  ¦
                                       Location  -----------+  ¦
                                   Memory Space  --------------+
```

Figure 3-18. Local Address Space 0 Range Register for PCI to Local Bus

Table 3-10. 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 ... 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
3	**Prefetchable.** This bit is 1b to indicate that there are no side effects on reads to memory space.	Yes	1
31 ... 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

3.3.2 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.

PCI address: Offset 04h from runtime base address
CFE Address: Offset 84h
Initialized value: 3E000001h

```
 31                                                    4 3   1 0
+-----------------------------------------------------------------+
¦                Remap Address Bits                   ¦     ¦ ¦
+-----------------------------------------------------------------+
                                                         ¦   ¦
                                         Not Used -----+   ¦
                                   Space 0 Enable ---------+
```

Figure 3-19. Local Address Space 0 Base Address (Remap) Register for PCI to Local Bus

Table 3-11. 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 ... 2	Not used when local space is mapped into memory space	Yes	00b
31 ... 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.

3.3.3 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

```
 31                                     11 10                 0
+----------------------------------------------------------------+
¦            Range Address Bits            ¦     Not Used        ¦
+----------------------------------------------------------------+
```

Figure 3-20. Local Expansion ROM Range Register for PCI to Local Bus

Table 3-12. Local Expansion ROM Range Register for PCI to Local Bus

Bits	**Description**	**Writeable**	**Initial value**
10 ... 0	Not used	Yes	0
31 ... 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

3.3.4 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

```
 31                                    11 10        5 4 3     0
+----------------------------------------------------------------+
¦            Range Address Bits           ¦  Not Used ¦ ¦        ¦
+----------------------------------------------------------------+
                                                       ¦    ¦
                         Local Bus BREQo Enable  ------+    ¦
                Direct Slave BREQo Delay Clocks  -----------+
```

Figure 3-21. Local Expansion ROM Local Base Address (Remap)

Table 3-13. Local Expansion ROM Local Base Address (Remap)

Bits	**Description**	**Writeable**	**Initial value**
3 ... 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 ... 5	Not used	No	0
31 ... 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

3.3.5 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

```
 31                                                              0
+----------------------------------------------------------------+
¦                Local Bus Region Descriptor                      ¦
+----------------------------------------------------------------+
```

Figure 3-22. Local Bus Region Descriptor for PCI to Local Accesses Register

Table 3-14. Local Bus Region Descriptor for PCI to Local Accesses Register

Bits	**Description**	**Writeable**	**Initial value**
1 ... 0	**Memory Space 0 Local Bus Width** This field is 11b to indicate a 32 bit width.	Yes	11b
5 ... 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
15 ... 8	Not used	Yes	0
17 ... 16	**Expansion ROM Space Local Bus Width.** When this field is 11b, it indicates a 32-bit width.	Yes	11
21 ... 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 ... 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

3.3.6 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

```
 31                              15                            0
+---------------------------------------------------------------+
¦   Range Address Bits          ¦      Not Used                 ¦
+---------------------------------------------------------------+
```

Figure 3-23. Local Range Register for Direct Master to PCI

Table 3-15. Local Range Register for Direct Master to PCI

Bits	**Description**	**Writeable**	**Initial value**
15 ... 0	Not used (64KB increments)	No	0
31 ... 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

3.3.7 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

```
 31                              15                             0
+-------------------------------------------------------------+
¦    Local Bus Base Address     ¦      Not Used                ¦
+-------------------------------------------------------------+
```

Figure 3-24. Local Bus Base Address Register for Direct Master to PCI Memory

Table 3-16. Local Bus Base Address Register for Direct Master to PCI Memory

Bits	**Description**	**Writeable**	**Initial value**
15 ... 0	Not used	No	0
31 ... 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

3.3.8 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

```
 31                              15              7 6 5 4 3 2 1 0
+----------------------------------------------------------------+
¦    Local Bus Base Address     ¦   Not Used    ¦     ¦ ¦ ¦ ¦ ¦ ¦
+----------------------------------------------------------------+
                                                    ¦  ¦ ¦ ¦ ¦ ¦
                Programmable Almost Full Flag ------+  ¦ ¦ ¦ ¦ ¦
                  Direct Master PCI Read Mode ---------+ ¦ ¦ ¦ ¦
     Direct Master Read Prefetch Size Control -----------+ ¦ ¦ ¦
                            Lock Input Enable -------------+ ¦ ¦
              Direct Master I/O Access Enable ---------------+ ¦
           Direct Master Memory Access Enable -----------------+
```

Figure 3-25. PCI Base Address (Remap) Register for Direct Master to PCI

Table 3-17. 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 ... 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 ...	Not used	Yes	0

8			
31 ... 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

3.4 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.

<table>
<tr><th>PCI Offset Addr</th><th>31</th><th>23</th><th>15</th><th>7 0</th></tr>
<tr><td>40h</td><td colspan="4">Parameter Register</td></tr>
<tr><td>44h</td><td colspan="2">Adapter Flags</td><td>Adapter Error</td><td>Adapter Command</td></tr>
<tr><td>48h</td><td colspan="4">Adapter Error Code Register</td></tr>
<tr><td>4Ch</td><td colspan="4">IEXECIO Register</td></tr>
<tr><td>50h</td><td colspan="4">Interrupt Control Register</td></tr>
<tr><td>54h</td><td colspan="4">Mailbox Register 5 (not used)</td></tr>
<tr><td>58h</td><td colspan="4">Mailbox Register 6 (not used)</td></tr>
<tr><td>5Ch</td><td colspan="4">Mailbox Register 7 (not used)</td></tr>
<tr><td>60h</td><td colspan="4">Adapter Interrupt Doorbell Register</td></tr>
<tr><td>64h</td><td colspan="4">PCI Interrupt Doorbell Register</td></tr>
<tr><td>68h</td><td colspan="4">Interrupt Control / Status</td></tr>
<tr><td>6Ch</td><td colspan="4">EEPROM Control, PCI Command Codes, User I/O Control, INIT Control</td></tr>
</table>

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

Subtopics

3.4.1 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

```
 31                                                             0
+---------------------------------------------------------------+
¦                     Parameter Register                         ¦
+---------------------------------------------------------------+
```

Figure 3-26. 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.

3.4.2 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

```
 7              0
+----------------+
¦ Adapter Command¦
+----------------+
```

Figure 3-27. 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.

3.4.3 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

```
           7 6             0
          +-----------------+
          ¦ ¦  Error Code   ¦
          +-----------------+
           ¦
Source ----+
```

Figure 3-28. 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)
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	

3.4.4 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

```
 15 14 13                    0
+----------------------------+
¦  ¦  ¦  Reserved=0          ¦
+----------------------------+
 ¦  ¦
 ¦  +--- Disable Execute I/O
 +------ Reset Complete
```

Figure 3-29. Adapter Flags Register

Table 3-18. Adapter Flags Register

Bits	**Description**	**Writeable**	**Initial value**
13 ... 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
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

3.4.5 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

```
 31                 23                                          0
+-----------------------------------------------------------------+
¦    Template      ¦lsb         Adapter Error Code             msb¦
+-----------------------------------------------------------------+
```

Figure 3-30. Adapter Error Code Register

Table 3-19. Adapter Error Code Register

Bits	**Description**	**Writeable**	**Initial value**
23 ... 0	**Adapter Error Code** This field provides details of the reason for the POST2A failure. The codes are described in "Adapter Error Logging Data" in topic 8.5.	Yes	0
31 ... 24	**Template** This byte is 0Ah (SSA_HDW_ERROR) for failures detected by the POST2A tests.	Yes	0

3.4.6 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

```
 31 30                                              7           0
+-----------------------------------------------------------------+
¦  ¦  ¦              Not Used                      ¦Adapter Error¦
+-----------------------------------------------------------------+
  ¦  ¦
  ¦  +--------- IExecute I/O executed unsuccessfully
  +------------ IExecute I/O command complete
```

Figure 3-31. IEXECIO Register

Table 3-20. IEXECIO Register

Bits	**Description**	**Writeable**	**Initial value**
7 ... 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" in topic 3.4.3). The contents are valid only when bit 30 = 1b, indicating an unsuccessful completion of an IExecute I/O command.	Yes	0
29 ... 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
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

3.4.7 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" in topic 3.4.8).

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

```
 31                                                             0
+----------------------------------------------------------------+
¦                     Interrupt Control                          ¦
+----------------------------------------------------------------+
```

Figure 3-32. Interrupt Control Register

3.4.8 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

```
 31                                                     4 3 2 1 0
+-----------------------------------------------------------------+
¦                     Not Used                          ¦ ¦ ¦ ¦ ¦ ¦
+-----------------------------------------------------------------+
                                                          ¦ ¦ ¦ ¦
                                 Host Heartbeat  ---------+ ¦ ¦ ¦
                                  Host Slave Op  -----------+ ¦ ¦
                        Command Register Loaded  -------------+ ¦
                                  New Host GTCB  ---------------+
```

Figure 3-33. Adapter Interrupt Doorbell Register

Table 3-21. 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**. The host sets this bit to 1b to set the refuse flag. When the refuse flag is set, the adapter does not generate any more adapter-to-host transactions. The refuse flag is reset the next time the host sets this bit to 1b.	Yes	0
31 ... 5	Not used	Yes	0

3.4.9 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

```
 31                                              7 6 5 4 3 2 1 0
+---------------------------------------------------------------+
¦                     Not Used                    ¦ ¦ ¦ ¦ ¦ ¦ ¦ ¦
+---------------------------------------------------------------+
                                                   ¦ ¦ ¦ ¦ ¦ ¦ ¦
                           Host Heartbeat Reply  --+ ¦ ¦ ¦ ¦ ¦ ¦
          Host Slave Op Executed unsuccessfully  ----+ ¦ ¦ ¦ ¦ ¦
            Host Slave Op Executed Successfully  ------+ ¦ ¦ ¦ ¦
                     Adapter Catastrophic Error  --------+ ¦ ¦ ¦
        Adapter Executed Command Unsuccessfully  ----------+ ¦ ¦
          Adapter Executed Command Successfully  ------------+ ¦
                        Host Inbound Pipe Entry  --------------+
```

Figure 3-34. PCI Interrupt Doorbell Register

Table 3-22. 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.**	Yes	0

	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.		
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
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 ... 7	Not used	Yes	0

3.4.10 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

```
 31                                                              0
+----------------------------------------------------------------+
¦                  Interrupt Control/Status                      ¦
+----------------------------------------------------------------+
```

Figure 3-35. Interrupt Control/Status

Table 3-23. Interrupt Control/Status

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 ... 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
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
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	No	0

	PCI-configuration-BIST register. Clearing bit 6 clears the interrupt.		
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 ... 28	Not used	No	0h

3.4.11 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

```
 31                                                            0
+---------------------------------------------------------------+
¦            PCI Command Codes, EEPROM Control, Init Control    ¦
+---------------------------------------------------------------+
```

Figure 3-36. PCI Command Codes, EEPROM Control, User I/O Control, INIT Control

Table 3-24. PCI Command Codes, EEPROM Control, User I/O Control, INIT Control

Bits	Description	Writeable	Initial value
3 ... 0	**PCI Read Command Code for DMA.**	Yes	1100
7 ... 4	**PCI Write Command Code for DMA.**	Yes	0111
11 ... 8	**PCI Memory Read Command Code for Direct Master.**	Yes	1100
15 ... 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 ... 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 retries until this bit is set. This bit is forced to 1b while NB low (not used on this adapter).	Yes	1

3.5 IPL

The Compatibility Basic Input/Output Subsystem (CBIOS) is used by personal systems for Initial Program Load (IPL) and installation of the operating system. It is a simple synchronous interface for executing one I/O operation at a time. CBIOS supports up to 8 disk drives only. The adapter provides a subset of the CBIOS for personal systems.

In normal operation the device driver accesses the adapter directly without using any of the CBIOS functions.

There are two software components involved:

> The majority of the function is implemented by internal adapter firmware that maps the request to IPN Transactions. This firmware is invoked using the Execute I/O command (described in "Execute I/O" in topic 2.2.3). It is not a direct mapping of the standard CBIOS register interface. In particular, the disk drive is addressed by a logical block address (LBA) rather than a cylinder, head, and sector (CHS).
>
> There is a small interface routine in x86 code to implement the standard INT 13h and INT 19h calls. This routine copies the x86 registers to and from the Execute I/O parameter block and translates the CHS address to and from an LBA. The interface routine also manages the disk drive data area at address 40:74h through 40:77h in host memory.
>
> The personal-system interface routine is stored on the adapter card and is fetched by the system by PCI accesses using the PCI Expansion ROM Base configuration register to identify the starting address of the BIOS image. BIOS is included as part of the firmware load that is downloaded using the Download command. The size of the BIOS image held on the card can be obtained by the host by writing all ones to the PCI-expansion-ROM base register and reading the value back. The adapter returns zeroes in all undefined bits. Only a single BIOS image is supported and its size is 4 KB.

In a cluster environment, the two PC servers must IPL from different disks. Normally, all disks are available to both PC servers; in which case, both PC servers would use the same disk as the IPL device. To avoid this, a procedure is provided, within the configuration utility, that identifies one or more disks as private to one PC server; that is, these disks are not available to the other PC server. This permits the PC servers each to have their own IPL device.

Subtopics

3.5.1 INT 13h functions

The table below shows which INT 13h functions are supported by the x86-interface routine:

Table 3-25. INT 13h functions

AH register	**Function and restrictions**
00h	Reset Disk System
01h	Read Status Of Last Operation
02h	Read Desired Sectors Into Memory
03h	Write Desired Sectors From Memory
04h	Verify Desired Sectors (No action - Not implemented)
05h	Format Desired Cylinder (Invalid request - Not implemented)
06h	Format Desired Cylinder And Set Bad Sector Flags (Invalid request - Not implemented)
07h	Format Drive Starting At Desired Cylinder (Invalid request - Not implemented)
08h	Read Drive Parameters
09h	Initialize Drive Pair Characteristics (No action - Not implemented)
0Ah ... 0Bh	Reserved
0Ch	Seek (No action - Not implemented)
0Dh	Alternate Disk Reset (Executed as Reset Disk System)
0Eh ... 0Fh	Reserved
10h	Test Drive Ready
11h	Recalibrate (No action - Not implemented)
12h ... 14h	Reserved
15h	Read DASD Type (Invalid request - Not implemented)
16h ... 18h	Reserved
19h	Park Heads (Invalid request - Not implemented)
1Ah	Format Unit (The defect table and modifiers are not supported)
1Bh ... FFh	Reserved

3.5.2 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 and also 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 3-26. Expansion ROM Header Format

Offset	**Length**	**Value**	**Description**
0 ... 1	2	55AAh	ROM signature
2	1		Initialization size
3 ... 5	3		Entry point for INIT function
6 ... 17	12h	xx	Reserved
18 ... 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 aligned. The offset field is a hexadecimal number of bytes from the beginning of the structure. The length field is in bytes.

Table 3-27. PCI Data Structure (expansion ROM) Format

Offset	**Length**	**Value**	**Description**
0 ... 3	4	PCIR	Signature
4 ... 5	2	1014h	Vendor identification
6 ... 7	2	0044h	Device identification
8 ...	2		Pointer to Vital Product Data

9			
A ... B	2		PCI data-structure length
C	1	00h	PCI data-structure revision
D ... F	3	00020Ch	Class code
10 ... 11	2	0008h	Image length
12 ... 13	2	0	Revision level of code/data
14	1	0	Code type
15	1	80h	Indicator
16 ... 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" in topic 2.4.

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. The value 00h identifies code for a x86.

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

4.0 Chapter 4. Adapter-to-Device Interface

Except where noted below the SSA ports conform fully with the SSA architectures described in "Supported Standards" in topic 1.5.

The SSA RAID Cluster Adapter card has 4 SSA ports which always operate as 2 dual-port nodes. The adapter does not operate as an SSA switch 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.

The upper level protocol returned in the query-node-reply SMS is FCh (Vendor Unique).

Subtopics

4.1 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

4.2 Master Functions

The adapter card can function as an SSA master. If it is an SSA master (that is, it has the higher SSA Unique ID), it also controls all the RAID-1 arrays on the attached disk drives.

Subtopics

4.2.1 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 guide-lines 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.

4.2.2 Asynchronous Alerts

When it is operating as a master, the adapter is responsible for returning the affected ports to normal mode after a transient unrecoverable link error, and performing third-party quiesce on behalf of a missing adapter.

4.3 Configurations

The SSA RAID Cluster 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 should be connected in the same loop

A single loop can contain up to two adapters

If two SSA adapters are in a loop and there is a second loop containing disk drives attached to one adapter, it must be attached to the other adapter also.

¦ Neither of the two servers in an SSA cluster can share disk drives with a third server

Disks that are members of an array can be on different loops

Up to 3 SSA adapters (depending on the system) can be plugged into the same PCI bus

¦ "SSA Cluster Configurations and Rules" in topic 6.2 provides more details for clustered configurations.

4.4 Adapter Card

The SSA RAID Cluster Adapter is a PCI 5V standard card size. The card measures 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 serial number of the adapter.

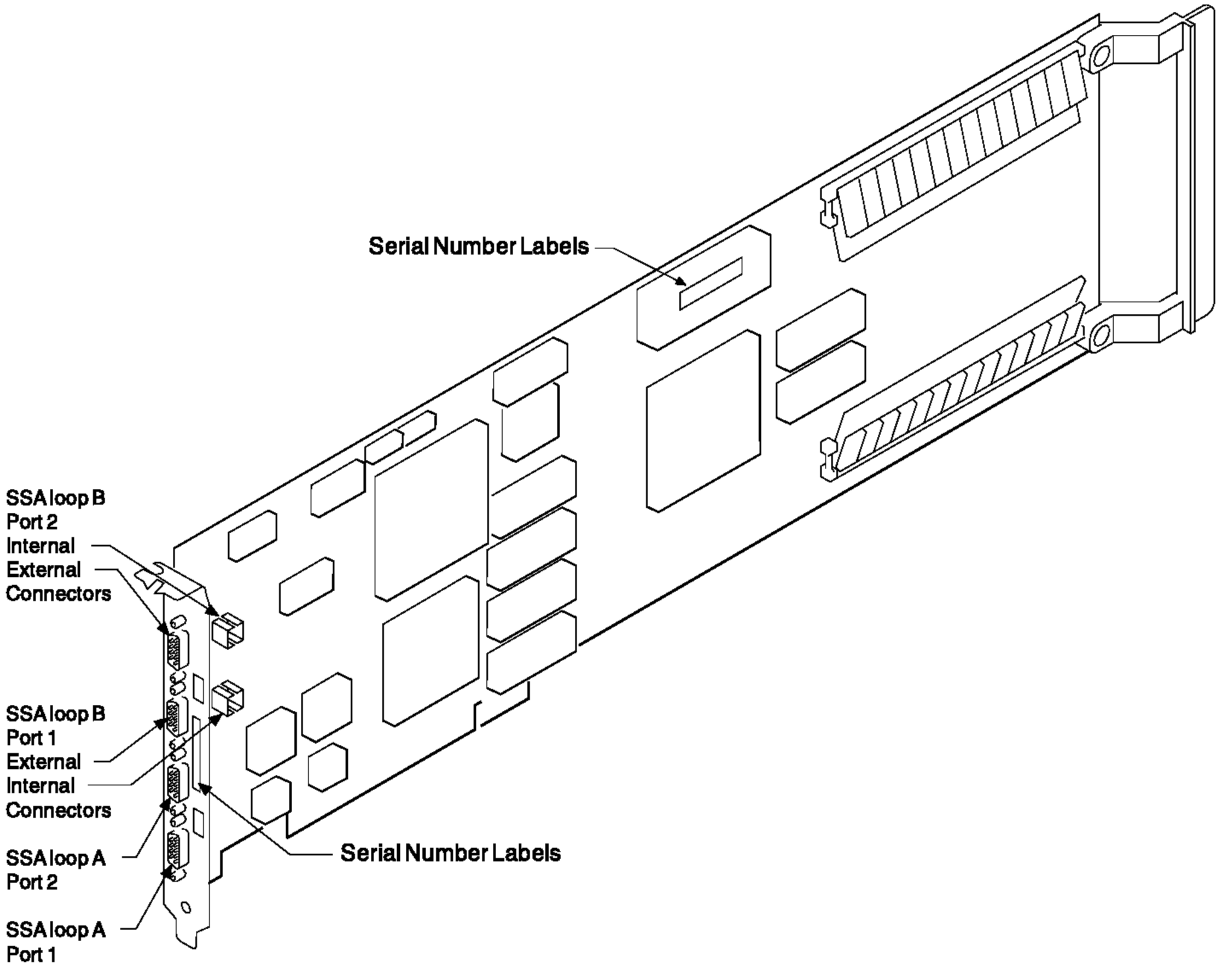


Figure 4-1. SSA RAID Cluster Adapter Card Layout

Subtopics
4.4.1 SSA Connectors
4.4.2 Indicators
4.4.3 DRAM Buffer
4.4.4 Power Requirements
4.4.5 Environment

4.4.1 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.

4.4.2 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.

4.4.3 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 16 MB are supplied on the adapter, giving a total DRAM capacity of 32 MB.

4.4.4 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

4.4.5 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

4.5 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.

4.6 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.

5.0 Chapter 5. Array Filters

Subtopics

5.1 Introduction

The SSA RAID Cluster Adapter provides 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 two disk drives to implement this image. IPN transactions are provided to configure the image. The members of the array 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 members of an array.

The adapter can support up to 32 arrays.

5.2 Disk Drives not in Arrays

SSA RAID Cluster Adapter supports 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 _s and the microprocessor utilization is less than 330 _s. The adapter can execute more than 3000 short read/write operations per second, depending on the attached devices and the capability of the system.

5.3 RAID-1

RAID-1 is also known as mirroring or dual copy. It provides redundancy with better performance than a single disk, but requires real disk capacity of two times the data size.

The RAID-1 filter uses two disk drive members. If the members have different capacities, the resulting RAID-1 array has the capacity of the smaller member. Each block N of the array appears on block N of each of the two members.

A write operation typically results in data being transferred once over the host bus into the adapter data buffer followed by two concurrent write operations to the member disk drives. The write operation is reported to be complete when the data has been written to both members.

A read operation results in data being read from one of the two members. A heuristic algorithm is used to determine which member to use based on the current known position of the actuator and the outstanding operations for that member. Data read from the member is not staged in the adapter data buffer.

If a write operation completes successfully to one of the members, but does not complete to the other member, the data contained in one member might be different from the data contained in the other member. (This condition can occur if, for example, power to one member is turned off before the write operation is complete.) While the condition exists, the adapter always returns consistent data for all following read operations to the affected block.

If one member fails, operations continue, mapped to the good member only. Performance is degraded for read operations (because all read operations are executed on a single disk), but improved for write operations (because they involve only a single disk), so a mixed workload might not experience substantial change. When either the original member returns and a write operation has been issued to the array while it was missing, or it is replaced (either manually through member exchange or automatically by the hot spare mechanism), a rebuild occurs. When the rebuild is complete the array reenters the nondegraded state. This rebuild is performed in parallel with any activity to the array. If a missing member disk becomes available again and no write operations have been received while the member was missing, and the disk has not been replaced by a hot spare or through member exchange, it rejoins the array and is not rebuilt.

When a RAID-1 array is created, it starts in the rebuilding state as the two member disks almost certainly do not contain identical data.

If a RAID-1 member disk appears on an adapter, and that disk has not previously been used in a non-degraded (or rebuilding) array on that adapter, an error log is produced. The array remains offline until either the other member disk appears, or the operator indicates that the one member should be used. To indicate that the one member is to be used, the operator changes the SplitResolution Resource Dependent Value for that member disk.

5.4 Hot spares

A hot-spare disk drive is a disk drive that is defined for automatic use if a disk drive within an array fails. The hot-spare must have a storage capacity greater than or equal to that of the smallest member of an array. You can define as many hot spares as you want. Any RAID-1 array on an adapter can use the hot-spare disk drives on that adapter. For full clustering operations to continue, the hot-spare used by a RAID-1 array must be in an SSA loop between the same two servers as the array.

If a disk drive within an array fails, after an interval, the adapter automatically uses a hot spare instead of the failed disk drive, and rebuilds the data that was on the failed disk on to the hot spare.

5.5 Array States

An array can be in one of the following states:

Good The array is online and it can be read and written. Both member disk drives of the array are present. No data rebuilding is outstanding. The array is fully protected against the loss of one member.

Exposed State One member is missing from the array. The first write operation causes the array to enter the degraded state.

After an interval, if a hot spare is available, it replaces the missing member and the array enters the rebuilding state.

Degraded State One member is missing and a write operation has been received for the array. Read and write operations to the array are supported. If a hot spare disk drive is available, after an interval, the hot spare replaces the missing member and the array enters the rebuilding state.

The missing member is permanently excluded from the array.

Rebuilding State The array is online and it can be read and written. Both members of the array are present but data is being rebuilt on one of the members.

Offline State An array is in this state when one of the following conditions exists:

Both members of the array are missing or have failed.

One member of the array is missing when a failover occurs.

The rebuilding of a replacement member is not complete when the remaining original member fails.

Read and write operations to the array are not supported.

The movement between states is illustrated in the following diagram:

```
                                                           Components present
               +----------------------------------+
               ¦              Offline             ¦                  0
               +----------------------------------+
                  ¦
              Add ¦                ¦ Loss      ¦ Loss
                  ¦      +---------+           ¦
                         ¦                     ¦
       +------------------------+  Write  +-------------------+
       ¦       Exposed          +--------_¦      Degraded     ¦      1
       +------------------------+         +-------------------+
                  ¦                            ¦
              Add ¦    ¦ Loss  ¦               ¦Add
                       ¦       ¦               ¦
     +-------------------+     ¦               ¦
     ¦     Rebuilding    ¦ ----+---------------+                     2
     +-------------------+     ¦
                  ¦            ¦
         Complete ¦            ¦ Loss
                               ¦
               +-------------------------+
               ¦          Good           ¦
               +-------------------------+
```

Figure 5-1. Array State Transitions

5.6 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 member disks of each array. This includes:

Array serial numbe

Disk drive serial numbe

Resource dependent value

Out-of-sync fla

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

6.0 Chapter 6. Clustering

Subtopics

6.1 Introduction

The SSA RAID Cluster Adapter can be used with Windows NT to support two-way "shared-nothing" clustering. A cluster consists of two systems. Each system in the cluster contains an SSA RAID Cluster Adapter card. These adapter cards and the disk drives are connected on one or two SSA loops that are between the systems.

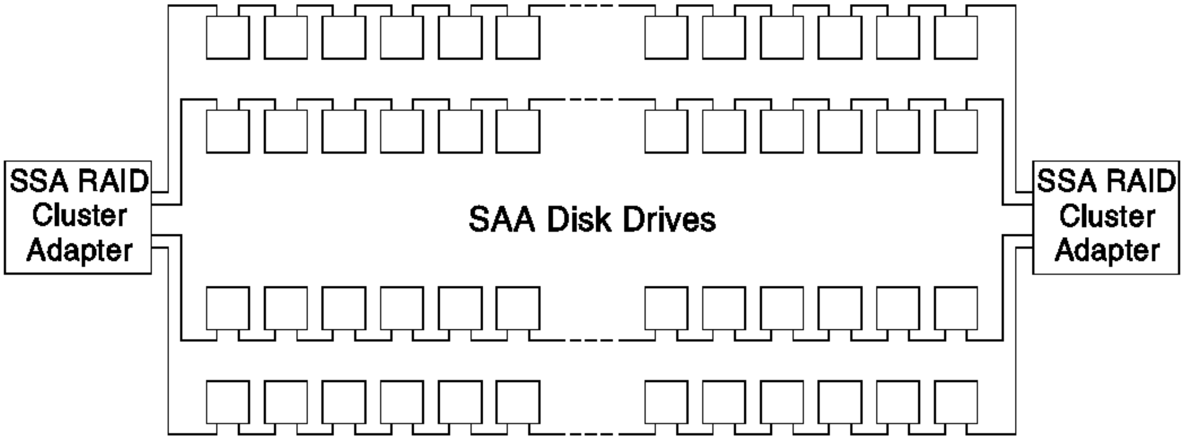
The SSA RAID Cluster Adapter provides support for RAID-1 and non-RAID disks. Each system in the cluster accesses non-RAID disks directly from its SSA RAID Cluster Adapter. For RAID-1 arrays, by default, the adapter with the higher SSA Unique ID is defined to be the *primary adapter* for all the arrays that are configured on the two adapters. There is an option, within the configuration utility, to select the other adapter in the cluster to be the primary adapter. The primary adapter controls all the arrays that are configured on the adapters. Transactions for an array from the adapter that is not the primary adapter are routed to the primary adapter to be implemented. Transactions for a disk that is not a member of an array are implemented by the adapter in the server that issued the transaction.

If the non-primary adapter detects that the primary adapter has failed., the non-primary adapter becomes the primary adapter and takes control of all the arrays.

6.2 SSA Cluster Configurations and Rules

The two adapters of a cluster can be connected by one or two SSA loops. They can provide support for up to 96 SSA disks (48 maximum per loop). The disk drives can be configured as non-RAID disks (which can be accessed individually by the using systems), as members of RAID-1 arrays, or as hot spares.

The following diagram shows an example configuration that has two SSA loops.



For SSA loops that contain the SSA RAID Cluster Adapter, the following rules apply:

Each SSA loop must be connected to valid pair of connectors on the adapter (that is, either connectors A1 and A2, or connectors B1 and B2).

A maximum of two SSA loops can be connected to an SSA RAID Cluster Adapter

A maximum of two SSA RAID Cluster Adapters can be connected in a particular SSA loop

If two adapters are present in a particular SSA loop, both adapters must be SSA RAID Cluster Adapters

In a configuration that contains two adapters and two SSA loops, both adapters must be SSA RAID Cluster Adapters, and both adapters must be connected to both SSA loops.

One system must be designated system A; the other must be designated system B

The Nth SSA adapter in system A must be wired to the Nth SSA adapter in system B. (The adapters in a particular system are usually in slot sequence.)

A maximum of 48 devices can be connected in a particular SSA loop

Under Microsoft Cluster Server, for a pair of servers, the total number of non-RAID disks and RAID-1 arrays can be up to 22. If no disk drives are configured as non-RAID disks, up to 44 disk drives can be configured as members of RAID-1 arrays, in addition to any disks that are configured as hot spares.

Without Microsoft Cluster Server, for a pair of servers, up to 96 RAID-1 arrays can be configured; that is, up to 32 arrays on each of three pairs of adapters.

Member disks of an array can be on different loops

Configurations that are not valid produce error logs. For descriptions of error codes, see Chapter 8, "Error Recovery and Error Logging" in topic 8.0.

6.3 RAID-1 Arrays

The primary adapter (that is, the one with the highest SSA Unique ID, unless selected using the configuration utility) controls the RAID-1 arrays. The non-primary adapter is a router. It sends transactions from its system to the primary adapter in the other system. In a cluster, the arrays are made available to each system through the primary adapter only.

The same arrays are reported to each system. The NT system determines which server to use to communicate to each array. Each system can reserve each array.

If a non-primary adapter detects that it has lost access to the other adapter on the SSA loop, it becomes the primary adapter, and reports the presence of the arrays. This action is called *failover*. After failover has occurred, the device driver in the system containing the new primary adapter reissues, to the new primary adapter, all the transactions that had been addressed to the previous primary adapter but were incomplete at the time of failover.

Each adapter checks that it can still communicate with its system by periodically sending SF_Ping to its host device driver. If one adapter does not receive a reply, the firmware on that adapter stops; this error is detected as a failure of that adapter by the other adapter. Failover is initiated.

If an array has one of its members missing (that is, the array is in the exposed or degraded state) when a failover occurs, the status of the array becomes offline and an error is logged. Manual intervention is needed to resolve this error.

A RAID-1 array is defined as synchronized if both members are present and no write operations have been received in the last 20 seconds. If the array is not synchronized when a failover occurs, the new primary adapter rebuilds the array.

If a failed adapter is replaced by an adapter whose SSA Unique ID is higher than that of the present primary adapter, a failover occurs. The replacement adapter becomes the primary adapter, and rebuilds all unsynchronized arrays.

Failover from a failed primary adapter to a non-primary adapter does not cause I/O operations to fail if those operations originate from the server that contains the previously non-primary adapter. I/O operations that come from the system that contains the failed adapter, however, might fail.

Failover from a non-primary adapter to a primary adapter generates an error log. For descriptions of error codes, see Chapter 8, "Error Recovery and Error Logging" in topic 8.0.

The performance available for I/O operations to RAID-1 arrays is:

Highest when all the I/O operations originate from the system that contains the primary adapte

Lowest when they all originate from the system containing the non-primary adapter

6.4 Non-RAID Disks

Non-RAID disks provide support for IPL. Because these disks are normally available to each PC server, both servers would attempt to IPL from the same disk unless they are prevented from doing so. A preventive procedure is available that configures selected non-array disks as private to a particular adapter. The private disks are available only to that particular adapter. They can, therefore, be used for IPL purposes.

Each system accesses its non-RAID disks directly through its local SSA RAID Cluster Adapter, because the device driver uses only the local adapter. Each disk may be reserved to either system.

7.0 Chapter 7. IPN Transactions

Subtopics

7.1 Introduction

The SSA RAID Cluster Adapter provides a registry service, a disk service, and an array-configurator 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)" in topic 2.1.1.

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.

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 drive or other resource that acts like one (see "Disk Service" in topic 7.4)

TP_Registry A local information server (see "Registry Service" in topic 7.3)

TP_IACL An array configurator (see "Array-Configuration Service" in topic 7.5).

7.2 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.

Subtopics

7.2.1 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    2     1     0
      +-----------------------+
      ¦ OMT ¦     Number      ¦
      +-----------------------+
```

```
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.  The following values are used:

  OMT = 1 - Not Owned by anyone
        2 - Device Driver Physical Adapters
        3 - Device Driver Physical Targets
        4 - Device Driver manually configured logical disks
        5 - Device Driver automatically configured logical disks
       'G'- RAID-1
       '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. For all but one owning-module type 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. The exception is OM_DriverManualDisk, which is used like a SCSI target number in the system and is permanently assigned to a disk; this information is kept in the device label record.

7.2.2 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

```
byte    3     2     1     0
      +-----------------------+
  0   ¦                       ¦
      ¦      Signature        ¦
      ¦                       ¦
      +-----------------------¦
  8   ¦    Revision number    ¦
      +-----------------------¦
  12  ¦    Reserved = 0       ¦
      +-----------------------¦
  16  ¦                       ¦
      ¦   496 bytes of data   ¦
      .                       .
      .                       .
 508  ¦                       ¦
      +-----------------------+
```

Figure 7-1. ISAL Reserved Area Sector Format

```
    The signature is a unique 8-byte field that is used to identify the
    sector as containing 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 caller 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 RAID Cluster 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.

7.2.3 Label Record

The label record is where the Owning Module Type (OMT) is recorded. If the OMT is OM_DriverManualDisk, another number is also stored, this is the disk number. For this owning-module type, the resource ID consists of the OMT in byte 3 (msb), zero in byte 2, the disk number in bytes 1 and 0 (lsb). 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. It has the following format:

```
byte    3     2     1     0
      +-----------------------+
  0   ¦                       ¦  LBL_Signature
      ¦      Signature        ¦
      ¦                       ¦
      +-----------------------¦
  8   ¦       Revision        ¦  LBL_Revision
      +-----------------------¦
 12   ¦     Reserved = 0      ¦
      +-----------------------¦
 16   ¦                 ¦ OMT ¦  LBL_OwningModuleType
      +-----------------------¦
 20   ¦  Unused   ¦Disk Number¦  LBL_DiskNumbers
      +-----------------------¦
 24   ¦                       ¦
      ¦                       ¦
      ¦    488 bytes unused   ¦
      ¦                       ¦
508   ¦                       ¦
      +-----------------------+
```

Figure 7-2. Label Record Format

7.3 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 7-1. 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
FN_REGY_SyncHCheckFromRegy	37

```
+-------------------------------------------------------------------------+
```

Subtopics

7.3.1 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

7.3.2 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)

7.3.3 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:

<table>
<tr><th>Byte</th><th>3</th><th>2</th><th>1</th><th>0</th></tr>
<tr><td>0</td><td colspan="3">Reserved = 0</td><td>Type</td></tr>
</table>

Type This identifies the type of services that should be reported. The types can be:

TP_ISAL	Disk service (or something that acts like one)
TP_Registry	Local information server
TP_CfgAgent	Array-configuration service

Transmit_DDR Null

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

<table>
<tr><th>Byte</th><th>3</th><th>2</th><th>1</th><th>0</th></tr>
<tr><td>0</td><td colspan="4">Service</td></tr>
<tr><td>4</td><td colspan="4">Service</td></tr>
<tr><td>.</td><td colspan="4"></td></tr>
<tr><td>n</td><td colspan="4">Service</td></tr>
</table>

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:

<table>
<tr><th>Byte</th><th>3</th><th>2</th><th>1</th><th>0</th></tr>
<tr><td>0</td><td colspan="4">Count</td></tr>
</table>

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)

7.3.4 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

7.3.5 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

7.3.6 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)

7.3.7 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)

7.3.8 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

7.3.9 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

7.3.10 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 ... n	Error Data			

Error Data See "FN_REGY_LogErrorFromRegistry" in topic 7.3.11 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)

7.3.11 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 ... 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 ... 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. Member serial number in bytes 48 through 62

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" in topic 7.4.2.

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 member 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)

7.3.12 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

7.3.13 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

7.3.14 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.

ResourceID This identifies the 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

7.3.15 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.

Change Code This code identifies the reason for the resource change. The states of the resource are defined in "FN_REGY_ResrcChangeToRegistry" in topic 7.3.14.

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

7.3.16 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_DriverManualDisk This indicates a disk that has been assigned a permanent resource ID with a configuration tool. This is the OMT type that must be set for all logical resources used by the device driver on PC servers.

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_Raid0Filter The RAID-0 filter owns the disk

OM_Raid1Filter The RAID-1 filter owns the disk

OM_Raid5Filter The RAID-5 filter owns the disk

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)

7.3.17 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)

7.3.18 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

7.3.19 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

7.3.20 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)

7.3.21 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)

7.3.22 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 ... 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" in topic 7.3.14):

RS_Online
RS_Offline

Result The following result fields can be returned:

AS_Success
AE_Failure
Illegal Request (range)
AE_NotInTable

7.3.23 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)

7.3.24 FN_REGY_SetClusterNumber

This transaction identifies the cluster number of the system to the registry service. The cluster number can be in the range 1 through 127.

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

7.3.25 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 ... 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 members 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.

7.3.26 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 ... 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

7.3.27 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 and returns the sense data that would usually be logged to the registry service, in the Status_DDR and returns AE_Failure. 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 ... 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

7.4 Disk Service

The SSA RAID Cluster Adapter disk service uses the IPN Storage Access Language (ISAL) to provide access to the disk drives 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.

The ISAL interface is the same for both disk drives and arrays. Therefore, the same host operating system device drivers can be used for disk drives connected to either the IBM SSA Adapter for PCI or the SSA RAID Cluster Adapter. One or more array filters can be configured into the logical data flow to manipulate the data as required on its way to or from the physical disk drives. The transactions that configure the filters are directed to the configuration-agent service; a description of these transactions begins on page 7.5.

Subtopics

7.4.1 ISAL Transactions

The ISAL transactions that the disk service handles are listed in Table 7-2.

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 7-2. 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
FN_ISALMgr_SetOwningModuleType	42	Yes	No
FN_ISALMgr_AssignManualResrcID	43	Yes	No
FN_ISALMgr_GetPhysicalResrcIDs	44	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_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	62	Yes	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.

4. Array managers do not support the FN_ISALMgr_VPDInquiry transaction

7.4.2 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 ... 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 ... 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 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 device.

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 ... 11	Array Name			
12	Reserved = 0	'A'	Array Name	

+--+

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

Array Letter This ASCII character identifies the type of filter of the array resource. The letters for each filter type are defined in "Configuration / Array Identifiers" in topic A.4.

Result The following Result fields can be returned:

AS_Success
Illegal Request (range)
AE_InvalidRID

7.4.3 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_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 field provides more details if the status is ST_Failed:

HF_MotorFail The motor is stopped

HF_Unknown No more details are available

Result The following result fields can be returned:

AS_Success

Illegal Request (range)
AE_InvalidRID
AE_ReservationConflict
AE_Offline

7.4.4 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

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.

7.4.5 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.

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

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.

7.4.6 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

7.4.7 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.

7.4.8 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 ... 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 members 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.

7.4.9 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

7.4.10 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_Offline
AE_FormatInProgress

7.4.11 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

7.4.12 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

7.4.13 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_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 result, if it is sent to any resource ID that is not of the owning module type OM_DriverPhysicalDisk
AE_LogOpen result, if the associated logical resource is currently open
AE_SSAString if the SSA network is a string rather than a loop and the resource is not the last node of the string.

When a resource is in service mode, the SSA links on the adjacent resources 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_LogOpen
AE_SSAString
AE_Offline
AE_InServiceMode

7.4.14 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.

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)

7.4.15 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	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. The values of undefined bits 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 members, and might be reconstructed from the other members, but it is not transferred to the client.

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.

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

7.4.16 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	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. The values of undefined bits 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_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.

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.

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

7.4.17 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.

7.4.18 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 illegal-request result if the owning module type of the resource is other than OM_DriverPhysicalDisk or if the corresponding logical resource ID 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

7.4.19 FN_ISAL_Lock

This transaction is sent to the disk service to reserve exclusively to this client the resource identified by the handle field.

Minor_function 56

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 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

7.4.20 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.

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

7.4.21 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

7.4.22 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

7.4.23 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.

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

ISAL Manager transactions FN_ISALMgr_SetOwningModuleType, and FN_ISALMgr_AssignManualResourceID

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

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.

7.4.24 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.

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 ... 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

7.5 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 7-3. Array-Configuration Transactions

Transaction	**Minor_function**
FN_IACL_Register	102
FN_IACL_Unregister	103
FN_IACL_Command	101

Subtopics

7.5.1 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)

7.5.2 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)

7.5.3 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 7.5.4 through 7.5.32.

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 ... n	Command Parameters			

Function This specifies the one of the following functions:

Table 7-4. IACL Functions

Function	**Code**
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_VPDHardwareInquiry	16
FC_CompExchCandCount	17
FC_CompExchCandList	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

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_Raid1Filter
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 types FT_DriverAutomaticDisk and FT_DriverManualDisk support 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 ... n	Command Result			

The length field is the byte count of Status_DDR data that follows this field.

7.5.4 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)

7.5.5 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)

7.5.6 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 ... 19	Serial Number (n)			
20	Reserved = 0	Serial Number (n)		
24	Reserved = 0		Percent	Status
28 ... 39	Serial Number (n+1)			
40	Reserved = 0	Serial Number (n+1)		
44	Reserved = 0		Percent	Status
.				
20m-12 ... 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" in topic 7.3.14. If it is a real filter, this status indicates that the array does not have enough members to function or it contains inconsistent members.

FS_ResrcOnline This is only returned for a pseudofilter. It indicates the resource is in the RS_Online state defined in "FN_REGY_ResrcChangeToRegistry" in topic 7.3.14.

FS_ResrcOnlineNonDeg The array is not degraded and is fully operational.

FS_ResrcOnlineDeg The array is degraded.

FS_ResrcOnlineRebuild The missing member has been returned to a degraded array which is in the process of rebuilding.

FS_ResrcOnlineExposed A member is missing from an array and no write operations have yet been required to that member.

FS_ResrcUnknown This is the state that an array is in until N-1 members 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)

7.5.7 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 ... 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 ... 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 members 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" in topic 7.5.15). 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" in topic 7.3.14. If it is a real filter, this status indicates that the array does not have enough members to function or it contains inconsistent members.

FS_ResrcOnline
This is only returned for a pseudofilter. It indicates the resource is in the RS_Online state defined on "FN_REGY_ResrcChangeToRegistry" in topic 7.3.14.

FS_ResrcOnlineNonDeg
The array is not degraded and is fully operational.

FS_ResrcOnlineDeg
The array is degraded.

FS_ResrcOnlineRebuild
The missing member has been returned to a degraded array which is in the process of rebuilding.

FS_ResrcOnlineExposed
A member is missing from a RAID-5 array and no writes have yet been required to that member.

FS_ResrcUnknown
This is the state that an array is in until N-1 members 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)

7.5.8 FC_CandidateCount

This function reports the total number of candidate members that are available for use in creating an array. Only those currently unused candidates that match exactly the specified type (resource dependant 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 ... 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" in topic 7.5.7 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 members, 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)

7.5.9 FC_CandidateList

This function reports the serial numbers of candidate members that are available for use in creating an array. (The total number of available members is returned by the FC_CandidateCount function.) The function specifies the ordinal number of the first member 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 ... 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" in topic 7.5.7 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 ... 19	Serial Number (n)			
20	Reserved = 0	Serial Number (n)		
24	Reserved = 0		Percent	Status
28 ... 39	Serial Number (n+1)			
40	Reserved = 0	Serial Number (n+1)		
44	Reserved = 0		Percent	Status
.				
20m-12 ... 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 member.

Status This can be one of the following:

FS_CandOnline The member is in the RS_Online state (see "FN_REGY_ResrcChangeToRegistry" in topic 7.3.14).

FS_CandOffline The member is in the RS_Offline state (see "FN_REGY_ResrcChangeToRegistry" in topic 7.3.14).

Percent This field is zero.

Result The following result fields can be returned:

AS_Success
AE_Failure
AE_BadParameterValues
AE_InvalidCandidateRequest
Illegal Request (range)

7.5.10 FC_ResrcCreate

This function is used to create a new resource, composed from a group of members (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 member 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 ... 15	Resource Serial Number			
16	Reserved = 0	Resource Serial Number		
20 ... 51	Resource Dependent Values			
52	Component Count (n)			
56 ... 67	Serial Number (1)			
68	Reserved = 0	Serial Number (1)		
72 ... 83	Serial Number (2)			
84	Reserved = 0	Serial Number (2)		
.				
40+16n ... 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" in topic 7.5.7 for details)

Component Count This is the number of members to be used to create the new resource.

Serial number These 15-character ASCII strings are the serial numbers of the members 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)

7.5.11 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 ... 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
Illegal Request (range)

7.5.12 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 ... 15	Old Serial Number			
16	Reserved = 0	Old Serial Number		
20 ... 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)

7.5.13 FC_ComponentView

This function is used to return the serial numbers of all the members of a resource. The number of the members is returned by the FC_ResrcView function. The request includes the ordinal number of the first member and the number of members to be reported (maximum 16). The returned length field describes the number of members 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 ... 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 member (starting at zero) to be reported.

Requested Count This is the maximum number of members that should be reported starting from the identified first member

Status_DDR This is a pointer to the following data:

Byte	**3**	**2**	**1**	**0**
0	Unused			
4	Length			
8 ... 19	Serial Number (n)			
20	Reserved = 0	Serial Number (n)		
24	Reserved = 0		Percent	Status
28 ... 39	Serial Number (n+1)			
40	Reserved = 0	Serial Number (n+1)		
44	Reserved = 0		Percent	Status
.				
20m-12 ... 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 member.

Status This can be one of the following:

FS_CompPresent This is returned if the resource is made out of other members that are all present.

FS_CompNotPresent This is returned if the array is made out of other arrays and this member is missing.

FS_CompNotPresentDeconf This can be returned in a RAID-5 array for a member that is deconfigured.

FS_CompNotPresentBlank This member 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)

7.5.14 FC_ComponentExchange

This function is used to replace a member of a resource with a new member, 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 member 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 ... 15	Resource Serial Number			
16	Reserved = 0	Resource Serial Number		
20 ... 31	Old Component Serial Number			
32	Reserved = 0	Old Component Serial Number		
36 ... 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 member.

New Component Serial Number This 15_character ASCII string is the name of the new member.

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)

7.5.15 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 ... 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 members for this filter type.

Max Component This is the maximum number of members 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)

7.5.16 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 ... 15	Serial Number			
16	Reserved = 0	Serial Number		
20 ... 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" in topic 7.5.15 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)

7.5.17 FC_FlashIndicator

This function is used to cause the light on all the members 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 ... 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 member 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)

7.5.18 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 ... 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 ... 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)

7.5.19 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 ... 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 a 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)

7.5.20 FC_CompExchCount

This function returns the number of members 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 ... 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 member 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 members that could be exchanged.

FT_HotSpare The count field in the status data refers to the number of hot-spare members 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 members			

Result The following result fields can be returned:

AS_Success
AE_BadResrcSerialNumber
Illegal Request (range)
AE_Failure
AE_FiltersOnly
AE_NotInTable

7.5.21 FC_CompExchCandList

This function reports the serial numbers of all the exchange members 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 ... 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 member exchange is required.

First Candidate This is the ordinal number of the first member (starting with zero) to be reported.

Requested Count This is the maximum number of members to be reported.

Exchange type This can be one of the following:

FT_NotOwned The identification of members that can be exchanged is returned in the status data

FT_HotSpare The identification of members 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 ... 19	Serial Number (n)			
20	Reserved = 0	Serial Number (n)		
24	Reserved = 0		Percent	Status
28 ... 39	Serial Number (n+1)			
40	Reserved = 0	Serial Number (n+1)		
44	Reserved = 0		Percent	Status
.				
20m-12 ... 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 member is in the RS_Online state (see "FN_REGY_ResrcChangeToRegistry" in topic 7.3.14).

FS_CandOffline The member is in the RS_Offline state (see "FN_REGY_ResrcChangeToRegistry" in topic 7.3.14).

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)

7.5.22 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 ... n	VPD data			

Length This is the number of bytes that follow in this field.

VPD data This is the adapter VPD data (see "Vital Product Data" in topic 2.4 for the definition of VPD)

Result The following result fields can be returned:

AS_Success
AE_NotInTable
Illegal Request (range)

7.5.23 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 the following data:

Byte	**3**	**2**	**1**	**0**
0	Reserved = 0			
4	Length			
8 ... n	Sense data			

Length This is the number of bytes that follow in this field (maximum 36).

Sense data The sense data is the most significant error log data from attached services. "FN_REGY_LogErrorFromRegistry" in topic 7.3.11 defines the sense data.

Result The following result fields can be returned:

AS_Success
AE_NotInTable
Illegal Request (range)
AE_Failure

7.5.24 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" in topic 7.4.13.

Parameter_DDR This is a pointer to the following data:

Byte	**3**	**2**	**1**	**0**
0	Reserved = 0	Filter Type	Function = 21	
4 ... 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	

7.5.25 FC_Unwrap

This function closes the identified physical resource that had previously been in service mode. If the resource had 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 ... 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	

7.5.26 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
Illegal Request (range)
AE_Failure

7.5.27 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 a 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 ... 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 internal checkout is 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)

7.5.28 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 ... 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 tested.

Blocksize This is the number of bytes in each block. This must be a value 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)

7.5.29 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 which 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 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 ... 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 tested.

Blocksize This is the number of bytes in each block. This must be a value 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)

7.5.30 FC_Read

This function reads a single sector from the disk identified by the serial number field.

The array-configuration service issues a 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.

As this IACL function does not involve sending a handle to identify the resource, certain applications, for example, service, may 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 ... 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
The logical block address of the block to be read

Priority This field is reserved for future use

Flags This field controls the type of read to be executed. These are defined in detail in FN_ISAL_Read (see "FN_ISAL_Read" in topic 7.4.15 for details).

Status_DDR This is a pointer to the following data:

Byte	**3**	**2**	**1**	**0**
0	Reserved = 0			
4	Length = 0			
8 ... N	Data			

Length Number of bytes of data following in the status data. Only 512-bytes 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)

7.5.31 FC_Write

This function writes a single sector to the disk identified by the serial number field.

The array-configuration service issues a 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.

As this IACL function does not involve sending a handle to identify the resource, certain applications, for example, service, may 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 ... 15	Serial Number			
16	Reserved = 0	Serial Number		
20	Logical Block Address			
24	Reserved = 0		Flags	Priority
28 ... 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
The logical block address of the block to be written

Priority This field is reserved for future use

Flags This field controls the type of write to be executed. These are defined in detail in FN_ISAL_Write (see "FN_ISAL_Write" in topic 7.4.16 for details).

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 present.

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)

7.5.32 FC_AdapterSN

This function returns the serial number of the adapter. This is required by the PC configurator when private disks are used.

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 ... 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 information.

Result The following result fields can be returned:

AS_Success
AE_Failure
AE_UnknownFunction

7.6 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.

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

8.0 Chapter 8. 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.

Subtopics

8.1 Strategy

The following strategy is implemented for error recovery and error reporting:

Subtopics

8.1.1 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.

SSA Link errors are recovered in accordance with the SSA link ERP specification included in the SSA architecture document. If the ERP fails, the error data is logged against the resource ID of the adapter.

8.1.2 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 devic
- Adapter-detected failures (includes errors in the adapter, arrays, SSA links, and the SSA configuration
- Device-driver-detected failure
- Device driver healthcheck-detected error

The error data that is logged is defined in "FN_REGY_LogErrorFromRegistry" in topic 7.3.11. This includes the following data:

- 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. For array errors, this might include the serial number of the array and that of the failing component.

8.1.3 Error Record Templates

The error record template is a code that is included in the error log data. The code associates the error with one of the following error classes:

Template Code	**Error Class**
01h	Permanent physical volume hardware error
02h	Recovered physical volume hardware error
03h	DASD statistical data
04h	Permanent physical volume media error
05h	Recovered physical volume media error
06h	Recovered SSA link error
07h	SSA detected error
08h	Permanent SSA device error
09h	SSA degraded condition
0Ah	Permanent SSA hardware error
0Bh	Recovered SSA hardware error
0Ch	Software or microcode error
0Dh	SSA loop open
0Eh	DASD-detected software or microcode error
0Fh	Device driver logging error
10h	Array error

8.2 Device Error Recovery

Any device attached to the adapter reports failures by means of asynchronous-alert conditions, SCSI status codes, and SCSI sense data. The error recovery performed by the adapter is identified by reference to a table. The contents of the sense-key, additional-sense-code, and additional-sense-code-qualifier (K/C/Q) fields of the sense data returned point to the appropriate entry in the table. The table contains the following information:

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

"SSA Disk Drive Error Recovery Table" in topic 8.6 contains the table for SSA disk drives.

8.3 Bad Sector Management

When a sector cannot be read from a disk, but that data can be reconstructed from the other member of the array, the bad sector is reassigned by a facility named *IDISK*.

All sectors that are on disks owned by RAID-1 are checked at defined intervals to verify that they can be read successfully. Sectors that cannot be read successfully are reconstructed from the other member of the array. The block is reassigned.

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.

RAID-1 arrays rewrite data whenever they have to reconstruct it because a disk has reported an unrecoverable data error. This rewriting causes IDISK to reassign the bad sector before writing the data.

8.4 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 06h, no error recovery is applied. However, if the asynchronous alert type is 06h 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.

8.5 Adapter Error Logging Data

The following table defines the error codes used when adapter error recovery procedures are invoked. These are hexadecimal characters. The first digit is the recommended number of occurrences of the error before a service action is required. Each of the errors in this table is logged against the adapter resource ID.

Table 8-1. Adapter Error Logging Data

Condition	**Error Code**
No error (only returned for AdapterHealthCheck)	00 00 00
Async type = 00 - 01 (no log)	-
Async type = 02 Unknown message	32 A0 02
Async type = 03 Invalid message	32 A0 03
Async type = 04 Protocol error	32 A0 04
Async type = 05 Environmental error (not reported)	02 A0 05
Async type = 06, Subtype = 01. Where ERP finds only one async of this type and subtype	02 A0 06
Async type = 06, Subtype = 01. Where ERP finds more than one device reporting this async type and subtype	02 A1 06
Async type = 06 Subttype = 03. Port not operational and POSTs failed	02 A2 06
Async type = 10 Permanent line fault P=port(0-3) HH=hop(00-99)	22 0P HH
Async type = 11 No characters received P=port(0-3) HH=hop(00-99)	A2 1P HH
Async type = 12 Remote port disabled P=port(0-3) HH=hop(00-99)	A2 2P HH
Async type = 13 Link reset failed P=port(0-3) HH=hop(00-99)	A2 3P HH
Async type = 14 Retry limit exceeded P=port(0-3) HH=hop(00-99)	02 4P HH
Async type = 15 Hardware error P=port(0-3) HH=hop(00-99)	A2 5P HH
Async type = 16 Frame reject P=port(0-3) HH=hop(00-99)	A2 6P HH
Async type = 17 Invalid retry status P=port(0-3) HH=hop(00-99)	A2 7P HH
Async type = 18 Time-out waiting for Disabled state P=port(0-3) HH=hop(00-99)	A2 8P HH
Async type = 19 Time-out waiting for Ready state P=port(0-3) HH=hop(00-99)	A2 9P HH
Invalid async. code	32 FF FF
Single device reports loss of redundant power or cooling	03 00 C0
Multiple devices report loss of redundant power or cooling	03 01 C0
Invalid SCSI status received	03 03 FF
Reallocations exceeded	03 10 00
Adapter issued Device Reset	23 10 00
Adapter Hardware Failure	04 00 00

Adapter 1 MB DRAM module zero failure	04 00 01
Adapter 2 MB DRAM module zero failure	04 00 02
Adapter 4 MB DRAM module zero failure	04 00 04
Adapter 8 MB DRAM module zero failure	04 00 08
Adapter 16 MB DRAM module zero failure	04 00 16
Adapter 32 MB DRAM module zero failure	04 00 32
Adapter 1 MB DRAM module one failure	04 10 01
Adapter 2 MB DRAM module one failure	04 10 02
Adapter 4 MB DRAM module one failure	04 10 04
Adapter 8 MB DRAM module one failure	04 10 08
Adapter 16 MB DRAM module one failure	04 10 16
Adapter 32 MB DRAM module one failure	04 10 32
Adapter both DRAM failure	04 20 00
SSA device is preventing the completion of link configuration, where P=port (0-3) and HH=decimal hop (00-99) of the failing device.	04 3P HH
Device with 'failed' status where P=port (0-3) and HH=decimal hop (00-99) of the failed device	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	24 5P HH
Array offline - more than one disk not available	04 60 00
Minimum disk resources not available for array filter	04 70 00
Incorrect data area of array	04 75 00
Illegal Link Configuration (SIC-SIC/>48 drives/>1 adapter)	04 80 00
Array filter degraded - one disk not available	04 90 00
Array filter exposed - one disk not available	04 91 00
No spares available for an array that is configures for spares	04 95 00
Incorrect parity in array	04 97 00
Different adapter on each loop	04 98 00
Unsupported configuration because some disks do not have failover protection	04 C0 00
Corresponding adapters are not wired together	04 C2 00
One host must be configured as A, the other as B	04 C4 00
Failover has occurred; this adapter is now controlling the arrays	04 C6 00
Only one member of a RAID-1 array present, and owning adapter serial number does not match this adapter	04 C8 00
Data scrubbing reconstructed a data block, and reassigned it	04 E0 00
Duplicated resource IDs of disks on the adapter	04 F0 00

Both members of a RAID-1 array have the Signal flag set (array serial number reported)	04 F1 00
Both members of a RAID-1 array have the Signal flag set (ignored component serial number reported)	04 F1 01
Member of RAID-1 array has been disowned (array serial number reported)	04 F2 00
Member of RAID-1 array has been disowned (component serial number reported)	04 F2 01

8.6 SSA Disk Drive Error Recovery Table

The following are the error recovery procedures for SSA disk drives when connected to a SSA RAID adapter. In the sense data returned from the SSA disk drive, the key (K) is in byte 2, the additional sense code (C) is in byte 12, and the additional sense code qualifier (Q) is in byte 13. Each of the errors in this table is logged against the disk drive resource ID.

Table 8-2. SCSI Sense Key/Code/Qualifier recovery procedures

Description	**SCSI K/C/Q**	**ERP#**	**Log**
No Additional Sense Information	0 00 00	1	0
No Index/Sector Signal	1 01 00	1	1
No Seek Complete	1 02 00	1	1
Peripheral Device Write Fault	1 03 00	1	1
Track Following Error	1 09 00	1	1
Write Error Recovered With Auto Reallocation	1 0C 01	1	1
Write Error - Recommend Reassignment	1 0C 03	2	1
Record Not Found	1 14 01	1	1
Record Not Found - Recommend Reassignment	1 14 05	2	1
Record Not Found - Data Auto Reallocated	1 14 06	1	1
Random Positioning Error	1 15 00	1	1
Positioning Error Detected by Read of Medium	1 15 02	1	1
Data Synchronization Mark Error	1 16 00	1	1
Data Synchronization Mark Error - Data Rewritten	1 16 01	1	1
Data Synchronization Mark Error - Recommend Rewrite	1 16 02	3	1
Data Synchronization Mark Error - Data Auto-Reallocated	1 16 03	1	1
Data Synchronization Mark Error - Recommend Reassignment	1 16 04	2	1
Recovered Data With Retries	1 17 01	1	1
Recovered Data with Positive Head Offset	1 17 02	1	1
Recovered Data with Negative Head Offset	1 17 03	1	1
Recovered Data using previous sector ID	1 17 05	2	1
Recovered Data Without ECC - Data Auto-Reallocated	1 17 06	1	1
Recovered Data Without ECC - Recommend Reassignment	1 17 07	2	1
Recovered Data Without ECC - Recommend Rewrite	1 17 08	3	1
Recovered Data Without ECC - Data	1 17 09	1	1

Rewritten			
Recovered Data with Error Correction and Retries Applied	1 18 01	1	1
Recovered Data - Data Auto-Reallocated	1 18 02	1	1
Recovered Data - Recommend-Reassignment	1 18 05	2	1
Recovered Data With ECC - Recommend Rewrite	1 18 06	3	1
Recovered Data With ECC - Data Rewritten	1 18 07	1	1
Primary Defect List Not Found	1 1C 01	12	1
Grown Defect List Not Found	1 1C 02	12	1
Partial Defect List Transferred	1 1F 00	12	1
Internal Target Failure	1 44 00	1	1
Spindles Not Synchronized	1 5C 02	15	2
Predictive Failure Analysis Threshold Reached on Recovered Error	1 5D 00	1	1
Logical Unit Not Ready Cause Not Reportable	2 04 00	6	2
Logical unit is in the process of becoming ready	2 04 01	7	2
Logical Unit Not Ready, initialization command required	2 04 02	6	2
Logical Unit Not Ready, Format in Progress	2 04 04	4	2
Medium Format Corrupted Reassign Failed	2 31 00	8	1
Format Command Failed	2 31 01	4	1
Diagnostic Failure	2 40 80	8	2
Diagnostic Failure	2 40 85	14	1
Diagnostic Failure	2 40 B0	9	1
Logical Unit Failed Self-Configuration	2 4C 00	8	2
Write Error - Auto-Reallocation Failed	3 0C 02	5	1
Write Error - Recommend Reassignment	3 0C 03	5	1
Unrecovered Read Error	3 11 00	5	1
Unrecovered Read Error - Auto Reallocation Failed	3 11 04	5	1
Unrecovered Read Error - Recommend Reassignment	3 11 0B	5	1
Recorded Entity Not Found	3 14 00	5	1
Record Not Found	3 14 01	5	1
Record Not Found - Recommend Reassignment	3 14 05	5	1
Data Synchronization Mark Error	3 16 00	5	1

Data Synchronization Mark Error - Recommend Reassignment	3 16 04	5	1
Defect List Error in Primary List	3 19 02	4	1
Defect List Error in Grown List	3 19 03	4	1
Medium Format Corrupted Reassign Failed	3 31 00	4	1
Format Failed	3 31 01	4	1
Internal Target Failure	3 44 00	8	1
No Index/Sector Signal	4 01 00	13	2
No Seek Complete	4 02 00	13	2
Peripheral Device Fault	4 03 00	13	2
Track Following Error	4 09 00	13	2
Unrecovered Read Error in Reserved Area	4 11 00	4	1
Recorded Entity Not Found	4 14 00	13	2
Record Not Found - Reserved Area	4 14 01	4	1
Random Positioning Error	4 15 00	13	2
Positioning Error Detected by Read of Medium	4 15 02	13	2
Data Synchronization Mark Error in Reserved Area	4 16 00	4	1
Defect List Error in Primary List	4 19 02	4	1
Defect List Error in Grown List	4 19 03	4	1
Medium Format Corrupted Reassign Failed	4 31 00	5	1
No Defect Spare Location Available	4 32 00	4	1
Defect list update failure	4 32 01	4	1
Diagnostic Failure	4 40 80	8	2
Diagnostic Failure	4 40 85	14	1
Diagnostic Failure	4 40 90	8	2
Diagnostic Failure	4 40 A0	8	2
Diagnostic Failure	4 40 B0	9	1
Diagnostic Failure	4 40 C0	8	2
Diagnostic Failure	4 40 D0	8	2
Internal Target Failure	4 44 00	13	2
Spindles Not Synchronized	4 5C 02	15	2
Parameter List Length Error	5 1A 00	10	2
Invalid Command Operation Code	5 20 00	10	2
Logical Block Address out of Range	5 21 00	10	2
Invalid Field in CDB	5 24 00	10	2

Logical Unit Not Supported	5 25 00	10	2
Invalid Field in Parameter List	5 26 00	10	2
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
Write Protected	7 27 00	4	2
Internal Target Failure	B 44 00	13	2
Invalid Message Error	B 49 00	10	2
Overlapped Commands Attempted	B 4E 00	16	2
Miscompare During Verify Operation	E 1D 00	4	1
Invalid KCQ	x xx xx	4	2

A.0 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.

Subtopics

A.1 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

A.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
MD_ISAL	0
MD_SCSI	1
MD_Service	2
AT_All	0
AT_ReadOnly	1
AT_WriteOnly	2
SM_DenyNone	0
SM_DenyAll	1
SM_DenyWrite	2
SM_DenyRead	3
FF_Verify	01h
FF_NoCache	02h
FF_PreFetch	04h
FF_Split	08h
FF_ReadDisk	10h
FF_FastWrite	20h
FF_ISALReservedArea	40h
FF_ReassignWrite	80h
RF_ReassignWarn	01h
RF_DriveWarn	02h
RF_RewriteWarn	04h
RF_BlockWarn	08h
UL_Normal	0
UL_Forced	1
ST_Good	0
ST_Failed	1
VP_NoEVPD	0
VP_EVPD	1
TT_Test	0
TT_Diag	1
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
OM_NotOwned	1
OM_DriverAdapters	2
OM_DriverPhysical	3
OM_DriverManualDisk	4
OM_DriverAutomaticDisk	5
OM_FirstFilter	E
OM_Raid0Filter	F
OM_Raid1Filter	G
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

A.3 Service / Transaction Directives

Name	**Value**
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
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
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_Debug	17
TP_Nvram	18

A.4 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_Raid0Filter	F
FT_Raid1Filter	G
FT_Raid5Filter	K
FT_HotSpareDisk	Y
FT_BlankReserved	Z
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
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
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_CompInUse	68
FS_ResrcInUse	70
FS_ResrcDormant	71
FS_ResrcWrapped	72
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_BAD_PTY_STRIDE	25
SDS_BAD_COMPONENT_STRIDE	26
SDS_BAD_STRIPE	27
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

B.0 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.

Subtopics

B.1 Trademarks

The following term is a trademark of IBM Corporation in the United States or other countries or both:

IBM

Microsoft and Windows are trademarks or registered trademarks of Microsoft Corporation.

Other company, product, or service names may be trademarks or service marks of others.