IBM FlashSystem™

# SCSI Interface Guide

IBM FlashSystem™

# SCSI Interface Guide

**Note**

Before using this information and the product it supports, read the information in “Notices” on page 47.

This edition applies to IBM FlashSystem™ and to all subsequent releases and modifications until otherwise indicated in new editions.

# Contents

# IBM FlashSystem™ SCSI Interface Guide

Learn about the Small Computer System Interface (SCSI) commands that are supported by your IBM® FlashSystem™.

## SCSI command set

Learn about the interface specifications and requirements as implemented by IBM FlashSystem™ SCSI controllers.

Use this information as a basic reference for supported commands, required parameters, and common command responses.

### Conventions of multibyte values

Unless otherwise explicitly stated, all multibyte values are stored with the most significant byte first (that is, big-endian byte ordering).

For example, in a 4-byte field, byte 0 contains the most significant bit (MSB) and byte 3 contains the least significant bit (LSB). Within a single byte, the most significant bit is listed first, and the least significant bit is listed last. For example, the value 5h is listed as 0101b, with the first 0 as the MSB and the last 0 as the LSB.

All commands that contain reserved fields are expected to be padded with zeros. Reserved fields are ignored. All parameter data and responses that contain reserved fields are padded with zeros by the device.

### Supported versions

Learn which SCSI specifications are supported on your system.

The supported SCSI specifications of the SCSI standard follow:

- SAM-3 (ISO/IEC 14776-413, ANSI INCITS 402-2005)
- SPC-3 (ISO/IEC 14776-453,ANSI INCITS 408-2005)
- SBC-2 (ISO/IEC 14776-322,ANSI INCITS 405-2005)

Note that in some instances, some obsolete commands (for example, the **Reserve/Release6/10** command) are supported for compatibility with older host bus adapters (HBAs) and operating systems.

### Supported SCSI commands

Learn about the supported SCSI commands.

The following table lists the supported SCSI commands and any supported optional features.

If an unsupported command is received, the command is ended with a status of CHECK CONDITION. The status also includes a sense key of ILLEGAL REQUEST and an extra sense code of INVALID FIELD IN CDB.

*Table 1. Supported SCSI commands*

| Primary command | Optional features |
|---|---|
| 00h TEST UNIT READY | |
| 03h REQUEST SENSE | |
| 04h FORMAT UNIT | |
| 08h READ (6) | |
| 0Ah WRITE (6) | |
| 0Bh SEEK (6) | |
| 12h INQUIRY | Vital product data page (00h) |
| | Unit serial number page (80h) |
| | Device identification page (83h) |
| | Management network addresses page (85h) |
| | Block Limits (B0h) |
| | Block Device Characteristics (B1h) |
| 15h MODE SELECT (6) | |
| 16h RESERVE (6) | |
| 17h RELEASE (6) | |
| 1Ah MODE SENSE (6) | Read/write error recovery page (01h) |
| | Disconnect/reconnect page (02h) |
| | Format device page (03h) |
| | Rigid disk geometry page (04h) |
| | Verify error recovery page (07h) |
| | Caching page (08h) |
| | Control page (0Ah) |
| | Medium types supported page (0Bh) |
| | Fibre Channel interface LUN page (18h) |
| | Fibre Channel interface control page (19h) |
| | Information exceptions control page (1Ch) |
| 1Bh START UNIT/STOP UNIT | |
| 1Dh SEND DIAGNOSTIC | |
| 1Eh PREVENT/ALLOW MEDIUM REMOVAL | |
| 25h READ CAPACITY (10) | |
| 28h READ (10) | |
| 2Ah WRITE (10) | |
| 2Bh SEEK (10) | |
| 2Eh WRITE AND VERIFY (10) | |
| 2Fh VERIFY (10) | |
| 35h SYNCHRONIZE CACHE (10) | |
| 3Bh WRITE BUFFER | Data mode (02h) |
| | Echo buffer mode (0Ah) |
| | Enable expander communications protocol and echo buffer mode (1Ah) |

*Table 1. Supported SCSI commands (continued)*

| Primary command | Optional features |
|---|---|
| 3Ch READ BUFFER | Data mode (02h) |
| | Descriptor mode (03h) |
| | Echo buffer mode (0Ah) |
| | Echo buffer descriptor mode (0Bh) |
| | Enable expander communications protocol and echo buffer mode (1Ah) |
| 4Dh LOG SENSE | Supported log page (00h) |
| | Read error counter page (03h) |
| 55h MODE SELECT (10) | |
| 56h RESERVE (10) | |
| 57h RELEASE (10) | |
| 5Ah MODE SENSE (10) | See MODE SENSE (6) |
| 5Eh PERSISTENT RESERVE IN | Read Keys (00h) |
| | Read Reservation (01h) |
| | Report Capabilities (02h) |
| | Read Full Status (03h) |
| 5Fh PERSISTENT RESERVE OUT | Register (00h) |
| | Reserve (01h) |
| | Release (02h) |
| | Clear (03h) |
| | Preempt (04h) |
| | Preempt and Abort (05h) |
| | Register and Ignore Existing Key (06h) |
| 88h READ (16) | |
| 8Ah WRITE (16) | |
| 8Eh WRITE AND VERIFY (16) | |
| 8Fh VERIFY (16) | |
| 91h SYNCHRONIZE CACHE (16) | |
| 9Eh SERVICE ACTION IN (16) | Read Capacity (16) (10h) |
| A0h REPORT LUNS | |
| A3h MAINTENANCE IN | Report Device Identifier (05h) |
| A4h MAINTENANCE OUT | Set Device Identifier (06h) |
| A8h READ (12) | |
| AAh WRITE (12) | |
| AEh WRITE AND VERIFY | |
| AFh VERIFY (12) | |

## Format Unit

Learn about the **FORMAT UNIT** command for your system.

Use the following table to review the **FORMAT UNIT** command for your system.

*Table 2. FORMAT UNIT (04h)*

| Byte | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|
| 0 | OPERATION CODE (04H) | | | | | | | |
| 1 | FMTPI | RTOREQ | LONGL | FMTD | CMPLST | DEFECT LIST FORMAT | | |
| 2 | Vendor specific | | | | | | | |
| 3 | Obsolete | | | | | | | |
| 4 | | | | | | | | |
| 5 | CONTROL | | | | | | | |

The **FORMAT UNIT** command requests that the device server format the medium into logical blocks that are accessible to the application client. This function can be specified in the number of blocks and block length values that are received in the last mode parameter block descriptor in a **MODE SELECT** command.

If the **FMTPI** bit is set to 0 and the **RTOREQ** bit is set to 1, the command is ended with a status of CHECK CONDITION. The status also includes a sense key of ILLEGAL REQUEST and an extra sense code of INVALID FIELD IN CDB.

All other fields are ignored and a status of GOOD is returned for all other conditions. The medium is not altered as a result of this command.

## Inquiry

Learn about the **INQUIRY** command.

Use the following table to review the parameters for the **INQUIRY** command.

*Table 3. Inquiry (12h)*

| Byte | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|
| 0 | OPERATION CODE (12H) | | | | | | | |
| 1 | Reserved | | | | | | Obsolete | EVPD |
| 2 | PAGE CODE | | | | | | | |
| 3 | ALLOCATION LENGTH | | | | | | | |
| 4 | | | | | | | | |
| 5 | CONTROL | | | | | | | |

The **Obsolete** field was defined in earlier standards as the **CmdDt** (command support data) bit. If this bit is set to 1, the command is ended with a status of CHECK CONDITION. The status also includes a sense key of ILLEGAL REQUEST and an extra sense code of INVALID FIELD IN CDB.

An enable vital product data (**EVPD**) bit that is set to 0 returns standard inquiry data. If the **EVPD** bit is set to 1, the vital product data that is specified by the **PAGE CODE** field is returned.

The **PAGE CODE** field specifies which vital product data page or pages to return in the data-in buffer. If the **PAGE CODE** field is not set to 0 when the **EVPD** bit is set to 0, the command is ended with a status of CHECK CONDITION. The status also includes a sense key of ILLEGAL REQUEST and an extra sense code of INVALID FIELD IN CDB. If the **INQUIRY** command is received with a **PAGE CODE** value that is unsupported, the command is ended with a status of CHECK CONDITION. The status also includes a sense key of ILLEGAL REQUEST and an extra sense code of INVALID FIELD IN CDB.

The **ALLOCATION LENGTH** field specifies the amount of inquiry data to transfer to the data-in buffer. When the inquiry response returns a status of GOOD, the number of bytes of inquiry data that is returned is the minimum of the amount available. Alternatively, the parameter data that is returned is the amount that is specified in the **ALLOCATION LENGTH** field in the command descriptor block (CDB).

The standard inquiry data never changes after the initial power-on sequence.

## Log Sense

The **LOG SENSE** command provides a means for the application client to retrieve statistical or other operational information that is maintained by the SCSI target device about the SCSI target device or its logical units.

Use the following table to review the parameters for the **Log Sense** command.

*Table 4. Log Sense (4Dh)*

| Byte | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|
| 0 | OPERATION CODE (4Dh) | | | | | | | |
| 1 | Reserved | | | | | | PPC | SP |
| 2 | PC | | PAGE CODE | | | | | |
| 3 | Reserved | | | | | | | |
| 4 | | | | | | | | |
| 5 | PARAMETER POINTER | | | | | | | |
| 6 | | | | | | | | |
| 7 | ALLOCATION LENGTH | | | | | | | |
| 8 | | | | | | | | |
| 9 | CONTROL | | | | | | | |

The **LOG SENSE** command is used with the **LOG SELECT** command.

If the **PPC** (parameter pointer control) bit is set to 1, the log page is returned only with parameter code values that changed since the last **LOG SELECT** or **LOG SENSE** command was run. If the **PPC** bit is set to 0, all parameter code values are returned regardless of whether they changed since the last **LOG SELECT** or **LOG SENSE** command was run.

An **SP** (save parameters) bit that is set to 0 specifies that the device server will run the specified **LOG SENSE** command and will not save any log parameters. If the **SP** bit is set to 1, the command is ended with a status of CHECK CONDITION. The status also includes a sense key of ILLEGAL REQUEST and an extra sense code of INVALID FIELD IN CDB.

The **PC** (page control) field is ignored.

The **PAGE CODE** field specifies which log page of data is being requested. If the log page code is reserved or not implemented, the command is ended with a status of CHECK CONDITION. The status also includes a sense key of ILLEGAL REQUEST and an extra sense code of INVALID FIELD IN CDB.

The **PARAMETER POINTER** field is ignored.

If the **LOG SENSE** command returns a status of GOOD, the amount of log page parameter data that is returned to the data-in buffer is the minimum of the amount of data available. Alternatively, the parameter data that is returned is the amount that is specified in the **ALLOCATION LENGTH** field in the command descriptor block (CDB).

## Mode Select (6)

The **MODE SELECT (6)** command provides a means for the application client to specify medium, logical unit, or peripheral device parameters to the device server.

Application clients need to issue the **MODE SENSE (6)** command before each **MODE SELECT (6)** command to determine supported mode pages, page lengths, and other parameters.

Use the following table to review the parameters for the **Mode Select (6)** command.

*Table 5. Mode Select (6) (15h)*

| Byte | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|
| 0 | OPERATION CODE (15h) | | | | | | | |
| 1 | Reserved | | | PF | Reserved | | | SP |
| 2 | Reserved | | | | | | | |
| 3 | | | | | | | | |
| 4 | PARAMETER LIST LENGTH | | | | | | | |
| 5 | CONTROL | | | | | | | |

The mode page policy that is used is the Shared mode page policy. This means one copy of the mode page exists that is shared by all IT nexuses. The mode page policy is reported in the Mode Page Policy VPD page.

Logical units share the mode parameter header values and block descriptor values across all I T nexuses. An I T nexus loss does not affect mode parameter header values, block descriptor values, and mode page values.

A **PF** (page format) bit that is set to 0 specifies that all parameters after the block descriptors are vendor-specific. No vendor-specific parameters are supported. A **PF** bit that is set to 1 specifies that the **MODE SELECT** parameters that follow the header and block descriptors are structured as pages of related parameters and are as defined in SPC-3.

An **SP** (save pages) bit that is set to 0 specifies that the device server will perform the specified MODE SELECT operation, and will not save any mode pages. If the **SP** bit is set to 1, the command is ended with a status of CHECK CONDITION. The status also includes a sense key of ILLEGAL REQUEST and an extra sense code of INVALID FIELD IN CDB.

The **PARAMETER LIST LENGTH** field specifies the length in bytes of the mode parameter list that is contained in the data-out buffer. A parameter list length of 0 specifies that the data-out buffer is empty.

If the parameter list length results in the truncation of any mode parameter header, mode parameter block descriptor, or mode page, then the command is ended with a status of CHECK CONDITION. The status also includes a sense key of ILLEGAL REQUEST, and an extra sense code of PARAMETER LIST LENGTH ERROR.

## Mode Select (10)

The **MODE SELECT (10)** command provides a means for the application client to specify medium, logical unit, or peripheral device parameters to the device server.

Application clients must issue the **MODE SENSE (10)** command before each **MODE SELECT (10)** command to determine supported mode pages, page lengths, and other parameters.

See “Mode Select (6)” for a description of the fields and operation of this command.

Use the following table to review the parameters for the **Mode Select** command.

*Table 6. Mode Select (10) (55h)*

| Byte | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|
| 0 | OPERATION CODE (55h) | | | | | | | |
| 1 | Reserved | | | PF | Reserved | | | SP |
| 2 | Reserved | | | | | | | |
| 6 | | | | | | | | |
| 7 | PARAMETER LIST LENGTH | | | | | | | |
| 8 | | | | | | | | |
| 9 | CONTROL | | | | | | | |

## Mode Sense (6)

The **MODE SENSE (6)** command provides a means for a device server to report parameters to an application client.

The **MODE SENSE (6)** command is used with the **MODE SELECT (6)** command.

Use the following table to review the parameters for the **MODE SENSE (6)** command.

*Table 7. Mode Sense (6) (1Ah)*

| Byte | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|
| 0 | OPERATION CODE (1Ah) | | | | | | | |
| 1 | Reserved | | | | DBD | Reserved | | |
| 2 | PC | | | | | | | |
| 3 | SUBPAGE CODE | | | | | | | |
| 4 | PARAMETER LIST LENGTH | | | | | | | |
| 5 | CONTROL | | | | | | | |

If the **DBD** (disable block descriptors) bit is set to 0, the device server returns a single block descriptor in the returned mode sense data. If the **DBD** bit is set to 1, no block descriptors are returned in the mode sense data.

The **PC** (page control) field specifies the type of mode parameter values to be returned in the mode pages. Current values (00b), default values (10b), and saved values (11b) are treated equally by the device server. Identical responses are returned for **MODE SENSE** commands that request those types of mode parameter values.

The **PAGE CODE** and **SUBPAGE CODE** fields specify which mode pages and subpages to return. If the page code or subpage code is not supported, the command is ended with a status of CHECK CONDITION. The status also includes a sense key of ILLEGAL REQUEST and an extra sense code of INVALID FIELD IN CDB.

If the application client requests all supported mode pages (for example, **PAGE CODE** equal to 3Fh), the supported pages are returned in ascending page code order beginning with mode page 01h if mode page 01h is implemented. If mode page 01h is not implemented, the supported pages are returned beginning with the first mode page in ascending order after 01h.

If the **PC** field and the **PAGE CODE** field are both set to 0, only a mode parameter header and block descriptor, if applicable, are returned to the data-in buffer.

If the **MODE SENSE** command returns a status of GOOD, the amount of **MODE SENSE** parameter data that is returned to the data-in buffer is the minimum of the amount of data available. Alternatively, the parameter data that is returned is the amount that is specified in the **ALLOCATION LENGTH** field in the command descriptor block (CDB).

## Mode Sense (10)

The **MODE SENSE (10)** command provides a means for a device server to report parameters to an application client.

The **MODE SENSE (10)** command is used with the **MODE SELECT (10)** command.

Use the following table to review the parameters for the **MODE SENSE (10)** command.

*Table 8. Mode Sense (10) (5Ah)*

| Byte | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|
| 0 | OPERATION CODE (5Ah) | | | | | | | |
| 1 | Reserved | | | LLBAA | DBD | Reserved | | |
| 2 | PC | | PAGE CODE | | | | | |
| 3 | SUBPAGE CODE | | | | | | | |
| 4 | Reserved | | | | | | | |
| 6 | | | | | | | | |
| 7 | ALLOCATION LENGTH | | | | | | | |
| 8 | | | | | | | | |
| 9 | CONTROL | | | | | | | |

If the **LLBAA** (Long LBA Accepted) bit is set to 1, parameter data with the **LONGLBA** bit equal to 1 is returned. If the **LLBAA** bit is set to 0, the **LONGLBA** bit is set to 0 in the returned parameter data. See the "Mode Sense (6)" on page 7 command for a description of all other fields and operation of this command.

## Persistent Reserve In

The **PERSISTENT RESERVE IN** command is used to obtain information about persistent reservations and reservation keys (that is, registrations) that are active within the device server.

This command is used with the **PERSISTENT RESERVE OUT** command.

Use the following table to review the parameters for the **MODE SENSE (6)** command.

*Table 9. Persistent Reserve In (5Eh)*

| Byte | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|
| 0 | OPERATION CODE (5Eh) | | | | | | | |
| 1 | Reserved | | | SERVICE ACTION | | | | |
| 2 | Reserved | | | | | | | |
| 6 | | | | | | | | |
| 7 | ALLOCATION LENGTH | | | | | | | |
| 8 | | | | | | | | |
| 9 | CONTROL | | | | | | | |

The **SERVICE ACTION** field specifies which **PERSISTENT RESERVE IN** parameter data to return to the data-in buffer. If an unsupported service action is requested, the command is ended with a status of CHECK CONDITION. The status also includes a sense key of ILLEGAL REQUEST and an extra sense code of INVALID FIELD IN CDB.

*Table 10. Supported PERSISTENT RESERVE IN service action codes*

| Code | Name | Description |
|---|---|---|
| 00h | READ KEYS | Reads all registered reservation keys (that is, registrations) |
| 01h | READ RESERVATION | Reads the current persistent reservations |
| 02h | REPORT CAPABILITIES | Returns capability information |

## Persistent Reserve Out

The **PERSISTENT RESERVE OUT** command is used to request service actions that reserve a logical unit for the exclusive or shared use of a particular IT nexus.

Use the following table to review the parameters for the **Persistent Reserve Out** command.

*Table 11. Persistent Reserve Out (5Fh)*

| Byte | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|
| 0 | OPERATION CODE (5Fh) | | | | | | | |
| 1 | Reserved | | | SERVICE ACTION | | | | |
| 2 | SCOPE | | | TYPE | | | | |
| 3 | Reserved | | | | | | | |
| 4 | | | | | | | | |
| 5 | ALLOCATION LENGTH | | | | | | | |
| 8 | | | | | | | | |
| 9 | CONTROL | | | | | | | |

If a **PERSISTENT RESERVE OUT** command is attempted, but there are insufficient resources to complete the operation, the command is ended with a status of CHECK CONDITION. The status also includes a sense key of ILLEGAL REQUEST, and an extra sense code of INSUFFICIENT REGISTRATION RESOURCES.

The **SCOPE** field specifies the scope to which the persistent reservation applies. If the scope contains any value other than 0h (LU SCOPE), the command is ended with a status of CHECK CONDITION. The status also includes a sense key of ILLEGAL REQUEST and an extra sense code of INVALID FIELD IN CDB.

The **TYPE** field specifies the characteristics of the persistent reservation that is established for all logical blocks within the logical unit. If the requested type is not implemented, the command is ended with a status of CHECK CONDITION. The status also includes a sense key of ILLEGAL REQUEST and an extra sense code of INVALID FIELD IN CDB.

## Prevent Allow Medium Removal

The **PREVENT ALLOW MEDIUM REMOVAL** command requests that the logical unit enable or disable the removal of the medium.

Use the following table to review the parameters for the **PREVENT ALLOW MEDIUM REMOVAL** command.

*Table 12. Prevent Allow Medium Removal (1Eh)*

| Byte | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|
| 0 | | | | | | | | |

*Table 12. Prevent Allow Medium Removal (1Eh) (continued)*

| Byte | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|
| 1 | OPERATION CODE (1Eh) | | | | | | | |
| 3 | | | | | | | | |
| 4 | Reserved | | | | | | PREVENT | |
| 5 | CONTROL | | | | | | | |

All fields are ignored. The **`PREVENT ALLOW MEDIUM REMOVAL`** command always returns a status of `GOOD` upon completion.

## Read (6)

Use the **LOGICAL BLOCK ADDRESS** field to specify the first logical block that is accessed by the **`READ (6)`** command.

If the logical block address exceeds the capacity of the medium, the command is ended with a status of `CHECK CONDITION`. The status also includes a sense key of `ILLEGAL REQUEST`, and an extra sense code of `LOGICAL BLOCK ADDRESS OUT OF RANGE`.

Use the following table to review the parameters for the **`READ (6)`** command.

*Table 13. Read (6) (08h)*

| Byte | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|
| 0 | OPERATION CODE (08H) | | | | | | | |
| 1 | Reserved | | | | | | | |
| 2 | LOGICAL BLOCK ADDRESS | | | | | | | |
| 3 | | | | | | | | |
| 4 | TRANSFER LENGTH | | | | | | | |
| 5 | CONTROL | | | | | | | |

The **TRANSFER LENGTH** field specifies the number of contiguous logical blocks of data to read and transfer to the data-in buffer. The **LOGICAL BLOCK ADDRESS** field specifies the first logical block that is accessed by this command. A **TRANSFER LENGTH** field that is set to 0 specifies that 256 logical blocks are read. If the logical block address plus the transfer length exceeds the capacity of the medium, the command is ended with a status of `CHECK CONDITION`. The status also includes a sense key of `ILLEGAL REQUEST` and an extra sense code of `LOGICAL BLOCK ADDRESS OUT OF RANGE`. The **TRANSFER LENGTH** field is constrained by the **MAXIMUM TRANSFER LENGTH** field in the Block Limits VPD page.

## Read (10)

The **`READ (10)`** command requests that the device server read the specified logical block or blocks and transfer them to the data-in buffer.

Use the following table to review the parameters for the **`READ (10)`** command.

*Table 14. Read (10) (28h)*

| Byte | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|
| 0 | OPERATION CODE (28h) | | | | | | | |
| 1 | RDPROTECT | | | DPO | FUA | Reserved | F. NV | Obsolete |
| 2 | LOGICAL BLOCK ADDRESS | | | | | | | |
| 5 | | | | | | | | |

*Table 14. Read (10) (28h) (continued)*

| Byte | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|
| 6 | Reserved | | | GROUP NUMBER | | | | |
| 7 | TRANSFER LENGTH | | | | | | | |
| 8 | | | | | | | | |
| 9 | CONTROL | | | | | | | |

If a **`READ (10)`** command is received with the **RDPROTECT** field set to any value other than 000b, the command is ended with a status of `CHECK CONDITION`. The status also includes a sense key of `ILLEGAL REQUEST` and an extra sense code of `INVALID FIELD IN CDB`.

The **`DPO`** (disable page-out), **`FUA`** (force unit access), and **`FUA NV`** (force unit access nonvolatile cache) bits are ignored.

The **LOGICAL BLOCK ADDRESS** field specifies the first logical block that is accessed by this command. If the logical block address exceeds the capacity of the medium, the command is ended with a status of `CHECK CONDITION`. The status also includes a sense key of `ILLEGAL REQUEST`, and an extra sense code of `LOGICAL BLOCK ADDRESS OUT OF RANGE`.

The **GROUP NUMBER** field is ignored.

The **TRANSFER LENGTH** field specifies the number of contiguous logical blocks of data to be read and transferred to the data-in buffer, starting with the logical block specified by the **LOGICAL BLOCK ADDRESS** field. A **TRANSFER LENGTH** field that is set to 0 specifies that no logical blocks is read. If the logical block address plus the transfer length exceeds the capacity of the medium, the command is ended with a status of `CHECK CONDITION`. The status also includes a sense key of `ILLEGAL REQUEST`, and an extra sense code of `LOGICAL BLOCK ADDRESS OUT OF RANGE`. The **TRANSFER LENGTH** field is constrained by the **MAXIMUM TRANSFER LENGTH** field in the Block Limits VPD page.

## Read (12)

The **`READ (12)(A8h)`** command requests that the device server read the specified logical block or blocks and transfer them to the data-in buffer.

Use the following table to review the parameters for the **`READ (12)`** command.

*Table 15. READ (12) (A8h)*

| Byte | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|
| 0 | OPERATION CODE (A8h) | | | | | | | |
| 1 | RDPROTECT | | | DPO | FUA | Reserved | F. NV | Obsolete |
| 2 | LOGICAL BLOCK ADDRESS | | | | | | | |
| 5 | | | | | | | | |
| 6 | TRANSFER LENGTH | | | | | | | |
| 9 | | | | | | | | |
| 10 | Restrict. | Reserved | | GROUP NUMBER | | | | |
| 11 | CONTROL | | | | | | | |

See the “Read (10)” on page 10 command for a description of the fields and operation of this command.

## Read (16)

The **READ (16) (88h)** command requests that the device server read the specified logical block or blocks and transfer them to the data-in buffer.

Use the following table to review the parameters for the **READ (16)** command.

*Table 16. Read (16) (88h)*

| Byte | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|
| 0 | OPERATION CODE (88h) | | | | | | | |
| 1 | RDPROTECT | | | DPO | FUA | Reserved | F. NV | Obsolete |
| 2 | LOGICAL BLOCK ADDRESS | | | | | | | |
| 9 | | | | | | | | |
| 10 | TRANSFER LENGTH | | | | | | | |
| 13 | | | | | | | | |
| 14 | Rest. | Reserved | | GROUP NUMBER | | | | |
| 15 | CONTROL | | | | | | | |

See the "Read (10)" on page 10 command for a description of the fields and operation of this command.

## Read Buffer

The **READ BUFFER** command is used with the **WRITE BUFFER** command as a diagnostic function for testing memory in the SCSI device and the integrity of the service delivery subsystem.

The **READ BUFFER** command does not alter the medium.

Use the following table to review the parameters for the **READ BUFFER** command.

*Table 17. Read Buffer*

| Byte | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|
| 0 | OPERATION CODE (3Ch) | | | | | | | |
| 1 | Reserved | | | MODE | | | | |
| 2 | OPERATION CODE | | | | | | | |
| 3 | BUFFER ID | | | | | | | |
| 4 | BUFFER OFFSET | | | | | | | |
| 5 | | | | | | | | |
| 6 | ALLOCATION LENGTH | | | | | | | |
| 7 | | | | | | | | |
| 8 | CONTROL | | | | | | | |

Use the **MODE** field to define the function of this command and the meaning of the fields within the command descriptor block (CDB). If the **MODE** field contains an unsupported mode, the command is ended with a status of CHECK CONDITION. The status also includes a sense key of ILLEGAL REQUEST and an extra sense code of INVALID FIELD IN CDB.

*Table 18. Supported READ BUFFER MODE field values*

| Mode | Description |
|---|---|
| 02h | Data |
| 03h | Descriptor |

*Table 18. Supported READ BUFFER MODE field values (continued)*

| Mode | Description |
|---|---|
| 0Ah | Echo buffer |
| 0Bh | Echo buffer descriptor |
| 1Ah | Enable expander communications protocol and Echo buffer |

If the **READ BUFFER** command returns a status of GOOD, the amount of read buffer data returned to the data-in buffer is the minimum of the amount of data available. Alternatively, the read buffer data returned is the amount that is specified in the **ALLOCATION LENGTH** field in the CDB.

## Read Capacity (10)

Use the **READ CAPACITY (10)** command to request that the device server transfer 8 bytes of parameter data.

The **READ CAPACITY (10)** command can also be used to describe the capacity and medium format of the direct-access block device to the data-in buffer.

Use the following table to review the parameters for the **READ CAPACITY (10)** command.

*Table 19. Read Capacity (10) (25h)*

| Byte | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|
| 0 | OPERATION CODE (25h) | | | | | | | |
| 1 | Reserved | | | | | | | Obsolete |
| 2 | LOGICAL BLOCK ADDRESS | | | | | | | |
| 3 | | | | | | | | |
| 4 | Reserved | | | | | | | |
| 5 | | | | | | | | |
| 6 | Reserved | | | | | | | PMI |
| 7 | CONTROL | | | | | | | |

The obsolete byte 1, bit 0 was defined in a previous standard as the relative address (RELADR) and is ignored.

A **PMI** (partial medium indicator) bit that is set to 0 specifies that the device server return information about the last logical block on the direct-access block device.

A **PMI** bit that is set to 1 specifies that the device server return information about the last logical block after that specified in the **LOGICAL BLOCK ADDRESS** field before a substantial vendor-specific delay in transfer is encountered. There is no logical block address (LBA) after which a substantial delay occurs in this device; therefore, the value that is returned when the **PMI** bit is set to 1 will be equal to the value returned when the **PMI** bit is set to 0.

If the **PMI** bit is set to 0 and the **LOGICAL BLOCK ADDRESS** field is not set to 0, the command is ended with a status of CHECK CONDITION. The status also includes a sense key of ILLEGAL REQUEST and an extra sense code of INVALID FIELD IN CDB.

*Table 20. Read Capacity (10) parameter data*

| Byte | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|
| 0 | RETURNED LOGICAL BLOCK ADDRESS | | | | | | | |
| 3 | | | | | | | | |
| 4 | BLOCK LENGTH | | | | | | | |
| 7 | | | | | | | | |

If the **PMI** bit is set to 0, the **RETURNED LOGICAL BLOCK ADDRESS** field is set to the lower of the following options:

- The logical block address (LBA) of the last logical block on the device
- FFFFFFFFh

If the **PMI** bit is set to 1, the **RETURNED LOGICAL BLOCK ADDRESS** field is set to the lower of the following options:

- The last LBA after that specified in the **LOGICAL BLOCK ADDRESS** field of the CDB before a substantial vendor-specific delay in data transfer can be encountered
- FFFFFFFFh

The **BLOCK LENGTH** field contains the number of bytes of user data in the logical block that is indicated by the **RETURNED LOGICAL BLOCK ADDRESS** field. This value does not include protection information or extra information (for example, ECC bytes) recorded on the medium. This value is either 512 or 4096.

## Read Capacity (16)

Use the **READ CAPACITY (16)** command to request that the device server sends parameter data to the data-in buffer. The parameter data includes information on the capacity and medium format of the direct-access block device.

This command is implemented as a service action of the **SERVICE ACTION IN** operation code.

Use the following table to review the parameters for the **READ CAPACITY (16)** command.

*Table 21. Read Capacity (16) (9Eh/10h)*

| Byte | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|
| 0 | OPERATION CODE (9eh) | | | | | | | |
| 1 | Reserved | | | SERVICE ACTION (10h) | | | | |
| 2 | LOGICAL BLOCK ADDRESS | | | | | | | |
| 3 | | | | | | | | |
| 4 | ALLOCATION LENGTH | | | | | | | |
| 5 | | | | | | | | |
| 6 | Reserved | | | | | | | PMI |
| 7 | CONTROL | | | | | | | |

A **PMI** (partial medium indicator) bit that is set to 0 specifies that the device server return information on the last logical block on the direct- access block device.

A **PMI** bit that is set to 1 specifies that the device server return information on the last logical block after that specified in the **LOGICAL BLOCK ADDRESS** field before a substantial vendor-specific delay in transfer might be encountered. There is no logical block address (LBA) after which a substantial delay will occur in this device; therefore, the value returned when the **PMI** bit is set to 1 will be equal to the value returned when the **PMI** bit is set to 0.

If the PMI bit is set to 0 and the **LOGICAL BLOCK ADDRESS** field is not set to 0, the command is ended with a status of CHECK CONDITION. The status also includes a sense key of ILLEGAL REQUEST and an extra sense code of INVALID FIELD IN CDB. The **ALLOCATION LENGTH** field specifies the amount of **READ CAPACITY** parameter data to transfer to the data-in buffer. When the read capacity response returns a status of GOOD, the number of bytes of parameter data that is returned is the minimum of 32 or the allocation length that is specified in the command descriptor block (CDB).

*Table 22. Read Capacity (16) parameter data*

| Byte | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|
| 0 | RETURNED LOGICAL BLOCK ADDRESS | | | | | | | |
| 7 | | | | | | | | |
| 8 | BLOCK LENGTH | | | | | | | |
| 11 | | | | | | | | |
| 12 | Reserved | | | | | | RTO EN | PROT EN |
| 13 | Reserved | | | | LOGICAL BLOCKS PER PHYSICAL BLOCK EXPONENT | | | |
| 14 | Reserved | | | | | | | |
| 31 | | | | | | | | |

The **RETURNED LOGICAL BLOCK ADDRESS** field and **BLOCK LENGTH** field of the **READ CAPACITY (16)** parameter data are the same as in the **READ CAPACITY (10)** parameter data. The maximum value that is returned in the **RETURNED LOGICAL BLOCK ADDRESS** field is FFFFFFFF FFFFFFFEh. The **RTO EN** (reference tag own enable) and **PROT EN** (protection enabled) bits are both set to 0 in the returned **READ CAPACITY** parameter data. The **LOGICAL BLOCKS PER PHYSICAL BLOCK EXPONENT** field is 3 if the device is configured with 512-byte logical blocks, and the field is 0 if you have a device that uses 4 kb blocks.

## Release (6)

The **RELEASE (6)** command is used to release a previously reserved logical unit.

Use the following table to review the parameters for the **RELEASE (6)** command.

*Table 23. Release (6) (17h)*

| Byte | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|
| 0 | OPERATION CODE (17h) | | | | | | | |
| 1 | Reserved | | | Obsolete | | | | |
| 2 | Obsolete | | | | | | | |
| 3 | Reserved | | | | | | | |
| 4 | | | | | | | | |
| 5 | CONTROL | | | | | | | |

The **RELEASE (6)** command does not release third-party reservations. A reservation is only released by a **RELEASE** command from the initiator that made the reservation. If an application client attempts to release a reservation that is not currently valid or held by another initiator, the command returns a status of GOOD without altering any other reservation.

Obsolete byte 1 bit 0 and bytes 2 - 4 provide an obsolete way to release previously reserved extents within a logical unit. If byte 1, bit 0 is set to 1, the command is ended with a status of CHECK CONDITION. The status also includes a sense key of ILLEGAL REQUEST and an extra sense code of INVALID FIELD IN CDB.

## Release (10)

The **RELEASE (10)** command is used to release a previously reserved logical unit.

Use the following table to review the parameters for the **RELEASE (10)** command.

*Table 24. Release (10) (57h)*

| Byte | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|
| 0 | OPERATION CODE (57h) | | | | | | | |
| 1 | Reserved | | | 3RDPTY | Reserved | | LONGID | Obsolete |
| 2 | Obsolete | | | | | | | |
| 3 | THIRD-PARTY DEVICE ID | | | | | | | |
| 4 | Reserved | | | | | | | |
| 6 | | | | | | | | |
| 7 | PARAMETER LIST LENGTH | | | | | | | |
| 8 | | | | | | | | |
| 9 | CONTROL | | | | | | | |

A reservation is only released by a **RELEASE** command from the initiator that made the reservation. If an application client attempts to release a reservation that is not currently valid or held by another initiator, the command returns a status of GOOD without altering any other reservation.

Obsolete byte 1 bit 0 and bytes 2 - 4 provide an obsolete way to release previously reserved extents within a logical unit. If byte 1, bit 0 is set to 1, the command is ended with a status of CHECK CONDITION. The status also includes a sense key of ILLEGAL REQUEST and an extra sense code of INVALID FIELD IN CDB.

If the **3RDPTY** (third-party) bit is set to 0, then a third-party release is not requested. The release is attempted for the initiator sending the **RELEASE (10)** command. In addition, the **LONGID** and **PARAMETER LIST LENGTH** fields are ignored.

If the **3RDPTY** bit is set to 1, then the reservation is released, but only if the initiator ID, **3RDPTY** bit, and **THIRD-PARTY DEVICE ID** field are identical when compared to the **RESERVE** command that established the reservation. Device ID formats are protocol specific. If the **LONGID** field is set to 0, the device ID is contained within the **THIRD-PARTY DEVICE ID** field. If the **LONGID** field is set to 1, the device ID is contained in the parameters that are located in the data-out buffer. In addition, the **THIRD-PARTY DEVICE ID** field in the command descriptor block (CDB) is ignored.

The **PARAMETER LIST LENGTH** field specifies the length in bytes of the release parameter list that is contained in the data-out buffer. If the **LONGID** bit is set to 1 and the parameter list length is not 8, the command is ended with a status of CHECK CONDITION. The status also includes a sense key of ILLEGAL REQUEST and an extra sense code of INVALID FIELD IN CDB.

## Report Device Identifier

The **REPORT DEVICE IDENTIFIER** command requests that the device server send device identification information to the application client.

As defined in the SCC-2 standard, the **REPORT DEVICE IDENTIFIER** command is the **REPORT PERIPHERAL DEVICE/COMPONENT DEVICE IDENTIFIER** service action of the **MAINTENANCE IN** command.

Use the following table to review the parameters for the **REPORT DEVICE IDENTIFIER** command.

*Table 25. Report Device Identifier (A3h/05h)*

| Byte | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|
| 0 | OPERATION CODE (A3h) | | | | | | | |
| 1 | Reserved | | | SERVICE ACTION (05h) | | | | |
| 2 | Reserved | | | | | | | |
| 3 | | | | | | | | |
| 4 | Restricted | | | | | | | |
| 5 | | | | | | | | |
| 6 | ALLOCATION LENGTH | | | | | | | |
| 9 | | | | | | | | |
| 10 | Reserved | | | | | | Restr. | Reserved |
| 11 | CONTROL | | | | | | | |

When the **REPORT DEVICE IDENTIFIER** command response returns a status of GOOD, the number of bytes of parameter data that is returned is the minimum of the amount available. Alternatively, the parameter data that is returned is the amount that is specified in the **ALLOCATION LENGTH** field in the command descriptor block (CDB).

*Table 26. Report Device Identifier parameter data*

| Byte | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|
| 0 | IDENTIFIER LENGTH (n-3) | | | | | | | |
| 3 | | | | | | | | |
| 4 | IDENTIFIER | | | | | | | |
| *n* | | | | | | | | |

The **IDENTIFIER LENGTH** field indicates the length in bytes of the **IDENTIFIER** field. The identifier length is initially equal to 0, and is only changed by a successful **SET DEVICE IDENTIFIER** command. The **IDENTIFIER** field is the last value written by a successful **SET DEVICE IDENTIFIER** command. The identifier persists through logical unit reset operations, I T nexus losses, media format operations, and media replacement. The same identifier is returned to all application clients.

## Report LUNS

The **REPORT LUNS** command is used to request that the logical unit inventory of the peripheral device that is accessible to the I T nexus is sent to the application client.

This device includes only logical unit numbers for logical units that have a **PERIPHERAL QUALIFIER** value of 000b.

Use the following table to review the parameters for the **REPORT LUNS** command.

*Table 27. Report LUNs (A0h)*

| Byte | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|
| 0 | OPERATION CODE (xxh) | | | | | | | |
| 1 | Reserved | | | | | | | |

Table 27. Report LUNs (A0h) (continued)

| Byte | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|
| 2 | SELECT REPORT | | | | | | | |
| 3 | Reserved | | | | | | | |
| 4 | | | | | | | | |
| 5 | ALLOCATION LENGTH | | | | | | | |
| 6 | | | | | | | | |
| 7 | Reserved | | | | | | | |
| 8 | CONTROL | | | | | | | |

The list contains the logical units that are accessible to the I T nexus with the following addressing methods:

- Logical unit addressing method
- Peripheral device addressing method
- Flat space addressing method

If there are no logical units, the **LUN LIST LENGTH** field is 0. The list contains only known logical units, if any. If there are no known logical units, the **LUN LIST LENGTH** field is 0. The list contains all logical units that are accessible to the I T nexus.

Table 28. Supported **SELECT REPORT** field values

| Mode | Description |
|---|---|
| 00h | The list contains the logical units that are accessible to the I T nexus with the following addressing methods: logical unit addressing method, peripheral device addressing method; and flat space addressing method. If there are no logical units, the **LUN LIST LENGTH** field is 0. |
| 01h | The list contains only known logical units, if any. If there are no known logical units, the **LUN LIST LENGTH** field is 0. |
| 02h | The list contains all logical units that are accessible to the I T nexus. |

When the **REPORT LUNS** command returns a status of GOOD, the number of bytes of parameter data that is returned is the minimum of the amount available. Alternatively, the parameter data that is returned is the amount that is specified in the **ALLOCATION LENGTH** field in the command descriptor block (CDB).

If the **ALLOCATION LENGTH** field is less than 16 bytes, the command is ended with a status of CHECK CONDITION. The status also includes a sense key of ILLEGAL REQUEST and an extra sense code of INVALID FIELD IN CDB.

The **LUN LIST LENGTH** field contains the length in bytes of the LUN list that is available to be transferred. The LUN list length is the number of logical unit numbers in the logical unit inventory that is multiplied by eight.

Table 29. REPORT LUNS parameter data

| Byte | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|
| 1 | LUN LIST LENGTH (n-3) | | | | | | | |
| 2 | | | | | | | | |
| 3 | Reserved | | | | | | | |
| 4 | | | | | | | | |

*Table 29. REPORT LUNS parameter data (continued)*

| Byte | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|
| 5 | First LUN | | | | | | | |
| 6 | | | | | | | | |
| n-7 | Last LUN | | | | | | | |
| n | | | | | | | | |

## Request Sense

The **REQUEST SENSE** command requests that the device server transfer sense data to the application client.

Use the following table to review the parameters for the **REQUEST SENSE** command.

*Table 30. Request Sense (03h)*

| Byte | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|
| 0 | OPERATION CODE (03h) | | | | | | | |
| 1 | Reserved | | | | | | | DESC |
| 2 | Reserved | | | | | | | |
| 3 | | | | | | | | |
| 4 | ALLOCATION LENGTH | | | | | | | |
| 5 | CONTROL | | | | | | | |

The **DESC** (descriptor format) bit specifies which sense data format is returned. If the **DESC** bit is set to 0, fixed-format sense data is returned. If the **DESC** bit is set to 1, the command is ended with a status of CHECK CONDITION. The status also includes a sense key of ILLEGAL REQUEST and an extra sense code of INVALID FIELD IN CDB.

The **ALLOCATION LENGTH** field specifies the amount of sense data to transfer to the data-in buffer. When the request sense response returns a status of GOOD, the number of bytes of parameter data that is returned is the minimum of 18 or the allocation length that is specified in the command descriptor block (CDB).

*Table 31. Fixed-format sense data*

| Byte | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|
| 0 | VALID | RESPONSE CODE (70h) | | | | | | |
| 1 | Obsolete | | | | | | | |
| 2 | FILMRK | EOM | ILI | Reserved | SENSE KEY | | | |
| 3 | INFORMATION | | | | | | | |
| 6 | | | | | | | | |
| 7 | ADDITIONAL SENSE LENGTH (n-7) | | | | | | | |
| 8 | COMMAND-SPECIFIC INFORMATION | | | | | | | |
| 11 | | | | | | | | |
| 12 | ADDITIONAL SENSE CODE | | | | | | | |
| 13 | ADDITIONAL SENSE CODE QUALIFIER | | | | | | | |
| 14 | FIELD REPLACEABLE UNIT CODE | | | | | | | |
| 15 | SKSV | | | | | | | |
| 16 | SENSE KEY SPECIFIC | | | | | | | |

*Table 31. Fixed-format sense data (continued)*

| Byte | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|
| 18 - n | Additional sense bytes | | | | | | | |

The **ILI** bit specifies the amount of sense data to transfer to the data-in buffer. When the request sense response returns a status of GOOD, the number of bytes of parameter data that is returned is the minimum of 18 or the allocation length that is specified in the command descriptor block (CDB).

## Reserve (6)

The **RESERVE (6)** command is used to reserve a logical unit.

Use the following table to review the parameters for the **RESERVE (6)** command.

*Table 32. Reserve (6) (16h)*

| Byte | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|
| 0 | OPERATION CODE (16h) | | | | | | | |
| 1 | Reserved | | | Obsolete | | | | |
| 2 | Obsolete | | | | | | | |
| 4 | | | | | | | | |
| 5 | CONTROL | | | | | | | |

If the logical unit is reserved for another initiator, the command is ended with a reservation conflict status.

Obsolete byte 1 bit 0 and bytes 2 - 4 provide an obsolete way to reserve extents within a logical unit. If byte 1, bit 0 is set to 1, the command is ended with a status of CHECK CONDITION. The status also includes a sense key of ILLEGAL REQUEST and an extra sense code of INVALID FIELD IN CDB.

## Reserve (10)

The **RESERVE (10)** command is used to reserve a logical unit.

Use the following table to review the parameters for the **RESERVE (10)** command.

*Table 33. Reserve (10) (56h)*

| Byte | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|
| 0 | OPERATION CODE (56h) | | | | | | | |
| 1 | Reserved | | | 3RDPTY | Reserved | | LONGID | Obsolete |
| 2 | Obsolete | | | | | | | |
| 3 | THIRD-PARTY DEVICE ID | | | | | | | |
| 4 | Reserved | | | | | | | |
| 5 | | | | | | | | |
| 6 | PARAMETER LIST LENGTH | | | | | | | |
| 7 | | | | | | | | |
| 8 | CONTROL | | | | | | | |

If the logical unit is reserved for another initiator, the command is ended with a reservation conflict status.

Obsolete byte 1 bit 0 and bytes 2 - 4 provide an obsolete way to reserve extents within a logical unit. If byte 1, bit 0 is set to 1, the command is ended with a status of CHECK CONDITION. The status also includes a sense key of ILLEGAL REQUEST and an extra sense code of INVALID FIELD IN CDB.

If the **3RDPTY** (third-party) bit is set to 0, then a third-party reservation is not requested. The reservation is attempted for the initiator that sends the **RESERVE (10)** command. In addition, the **LONGID** and **PARAMETER LIST LENGTH** fields are ignored.

If the **3RDPTY** bit is set to 1, then the reservation is attempted on the specified logical unit for the SCSI device that is specified in the **THIRD-PARTY DEVICE ID** field. Device ID formats are protocol specific. If a third-party reservation is requested and the **LONGID** field is set to 0, the device ID is contained within the **THIRD-PARTY DEVICE ID** field. If the **LONGID** field is set to 1, the device ID is contained in the parameters that are located in the data-out buffer. In addition, the **THIRD-PARTY DEVICE ID** field in the command descriptor block (CDB) is ignored.

The **PARAMETER LIST LENGTH** field specifies the length in bytes of the reserve parameter list that is contained in the data-out buffer. If the **LONGID** bit is set to 1 and the parameter list length is not 8, the command is ended with a status of CHECK CONDITION. The status also includes a sense key of ILLEGAL REQUEST and an extra sense code of INVALID FIELD IN CDB. The extra sense code might be INVALID PARAMETER LIST LENGTH.

## Seek (6)

The **SEEK (6)** command requests that the logical drive perform a seek to the specified logical block address.

Use the following table to review the parameters for the **SEEK (6)** command.

*Table 34. Seek (6) (0Bh)*

| Byte | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|
| 0 | OPERATION CODE (0bh) | | | | | | | |
| 1 | Reserved | | | | | | | |
| 2 | LOGICAL BLOCK ADDRESS | | | | | | | |
| 3 | | | | | | | | |
| 4 | Reserved | | | | | | | |
| 5 | CONTROL | | | | | | | |

All fields are ignored. The **SEEK 6** command always returns a status of GOOD upon completion.

## Seek (10)

The **SEEK (10)** command requests that the logical drive perform a seek to the specified logical block address.

Use the following table to review the parameters for the **SEEK (10)** command.

*Table 35. Seek (10) (2bh)*

| Byte | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|
| 0 | OPERATION CODE (2bh) | | | | | | | |
| 1 | Reserved | | | | | | | |
| 2 | LOGICAL BLOCK ADDRESS | | | | | | | |
| 5 | | | | | | | | |

*Table 35. Seek (10) (2bh) (continued)*

| Byte | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|
| 6 | Reserved | | | | | | | |
| 8 | | | | | | | | |
| 9 | CONTROL | | | | | | | |

All fields are ignored. The **SEEK (10)** command always returns a status of GOOD upon completion.

## Send Diagnostic

The **SEND DIAGNOSTIC** command is used to request the device server to run diagnostic operations on the SCSI target device, on the logical unit, or on both.

Use the following table to review the parameters for the **SEND DIAGNOSTIC** command.

*Table 36. Send Diagnostic (1Dh)*

| Byte | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|
| 0 | OPERATION CODE (1Dh) | | | | | | | |
| 1 | SELF-TEST CODE | | | PF | Reserved | ST | DOL | UOL |
| 2 | Reserved | | | | | | | |
| 3 | PARAMETER LIST LENGTH | | | | | | | |
| 4 | | | | | | | | |
| 5 | CONTROL | | | | | | | |

All fields are ignored. The **SEND DIAGNOSTIC** command always returns a status of GOOD upon completion.

## Start Stop Unit

The **START STOP UNIT** command requests that the device server change the power condition of the logical unit, or requests that the device server load or eject the medium.

Use the following table to review the parameters for the **START STOP UNIT** command.

*Table 37. Start Stop Unit (1Bh)*

| Byte | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|
| 0 | OPERATION CODE (1Bh) | | | | | | | |
| 1 | Reserved | | | | | | | IMMED |
| 2 | Reserved | | | | | | | |
| 3 | | | | | | | | |
| 4 | POWER CONDITION | | | | Reserved | | LOEJ | START |
| 5 | CONTROL | | | | | | | |

Regardless of whether the **IMMED** (immediate) bit is set to 0 or 1, the command always immediately returns status after the operation is complete because there is no rotating medium to start or stop.

If the **START STOP UNIT** command is received with the **POWER CONDITION** field set to **FORCE IDLE 0 (Ah)** or **FORCE STANDBY 0 (Bh)**, the command is ended with a status of CHECK CONDITION. The status also includes a sense key of ILLEGAL REQUEST and an extra sense code of INVALID FIELD IN CDB. All other values for the **POWER CONDITION** field other than 0 are ignored. The **LOEJ** (load eject) and **START** bits are ignored.

## Synchronize Cache (10)

The **SYNCHRONIZE CACHE (10)** command requests that the device server ensures that the specified logical blocks have their most recent data values that are recorded in nonvolatile cache or recorded on the medium, or both, based on the **SYNC NV** bit.

Use the following table to review the parameters for the **SYNCHRONIZE CACHE (10)** command.

*Table 38. Synchronize Cache (10) (35h)*

| Byte | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|
| 0 | OPERATION CODE (35h) | | | | | | | |
| 1 | Reserved | | | | | SYNC NV | IMMED | Obsolete |
| 2 | LOGICAL BLOCK ADDRESS | | | | | | | |
| 5 | | | | | | | | |
| 6 | Reserved | | | | GROUP NUMBER | | | |
| 7 | NUMBER OF BLOCKS | | | | | | | |
| 8 | | | | | | | | |
| 9 | CONTROL | | | | | | | |

The **SYNC NV** bit, **IMMED** (immediate) bit, **GROUP NUMBER** field, and **NUMBER OF BLOCKS** field is ignored.

The **LOGICAL BLOCK ADDRESS** field specifies the first logical block that is accessed by this command. If the logical block address exceeds the capacity of the medium, the command is ended with a status of CHECK CONDITION. The status also includes a sense key of ILLEGAL REQUEST, and an extra sense code of LOGICAL BLOCK ADDRESS OUT OF RANGE. Otherwise, the command returns a status of GOOD upon completion.

## Synchronize Cache (16)

The **SYNCHRONIZE CACHE (16)** command requests that the device server ensures that the specified logical blocks have their most recent data values that are recorded in nonvolatile cache or recorded on the medium, or both based on the **SYNC NV** bit.

Use the following table to review the parameters for the **SYNCHRONIZE CACHE (16)** command.

*Table 39. Synchronize Cache (16) (91h)*

| Byte | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|
| 0 | OPERATION CODE (91h) | | | | | | | |
| 1 | Reserved | | | | | SYNC NV | IMMED | Reserved |
| 2 | LOGICAL BLOCK ADDRESS | | | | | | | |
| 9 | | | | | | | | |
| 10 | NUMBER OF BLOCKS | | | | | | | |
| 13 | | | | | | | | |
| 14 | Reserved | | | GROUP NUMBER | | | | |
| 15 | CONTROL | | | | | | | |

See the “Synchronize Cache (10)” command for definitions of fields and operation of this command.

## Test Unit Ready

The **TEST UNIT READY** command provides a means to check if the logical unit is ready.

Use the following table to review the parameters for the **TEST UNIT READY** command.

*Table 40. Test Unit Ready (00h)*

| Byte | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|
| 0 | OPERATION CODE (00h) | | | | | | | |
| 1 | Reserved | | | | | | | |
| 4 | | | | | | | | |
| 5 | CONTROL | | | | | | | |

The **TEST UNIT READY** command always completes with a status of GOOD.

## Verify (10)

Use the **VERIFY (10)** command to request that the device server verify the specified logical block or blocks on the medium.

Use the following table to review the parameters for the **Verify (10)** command.

*Table 41. Verify (10) (2Fh)*

| Byte | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|
| 0 | OPERATION CODE (2Fh) | | | | | | | |
| 1 | WRPROTECT | | | DPO | Reserved | | BYTCHK | Obsolete |
| 2 | LOGICAL BLOCK ADDRESS | | | | | | | |
| 5 | | | | | | | | |
| 6 | Restr. | Reserved | | GROUP NUMBER | | | | |
| 7 | VERIFICATION LENGTH | | | | | | | |
| 8 | | | | | | | | |
| 9 | CONTROL | | | | | | | |

If a **VERIFY (10)** command is received with the **VRPROTECT** field set to any value other than 000b, the command is ended with a status of CHECK CONDITION. The status also includes a sense key of ILLEGAL REQUEST and an extra sense code of INVALID FIELD IN CDB.

The **LOGICAL BLOCK ADDRESS** field specifies the first logical block that is accessed by this command. If the logical block address exceeds the capacity of the medium, the command is ended with a status of CHECK CONDITION. The status also includes a sense key of ILLEGAL REQUEST, and an extra sense code of LOGICAL BLOCK ADDRESS OUT OF RANGE. All other fields are ignored, and the command returns a status of GOOD in all other cases.

## Verify (12)

The **VERIFY (12)** command requests that the device server verify the specified logical block or blocks on the medium.

Use the following table to review the parameters for the **VERIFY (12)** command.

*Table 42. Verify (12) (2Fh)*

| Byte | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|
| 0 | OPERATION CODE (2Fh) | | | | | | | |

*Table 42. Verify (12) (2Fh) (continued)*

| Byte | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|
| 1 | VRPROTECT | | | DPO | Reserved | | BYTCHK | Obsolete |
| 2 | LOGICAL BLOCK ADDRESS | | | | | | | |
| 5 | | | | | | | | |
| 6 | VERIFICATION LENGTH | | | | | | | |
| 9 | | | | | | | | |
| 10 | Restricted | Reserved | | GROUP NUMBER | | | | |
| 11 | CONTROL | | | | | | | |

See the "Verify (10)" on page 24 command for a description of the fields and operation of this command.

## Verify (16)

The **VERIFY (16)** command requests that the device server verify the specified logical block or blocks on the medium.

Use the following table to review the parameters for the **VERIFY (16)** command.

*Table 43. Verify (16) (8Fh)*

| Byte | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|
| 0 | OPERATION CODE (8Fh) | | | | | | | |
| 1 | VRPROTECT | | | DPO | Reserved | | BYTCHK | Reserved |
| 2 | LOGICAL BLOCK ADDRESS | | | | | | | |
| 9 | | | | | | | | |
| 10 | VERIFICATION LENGTH | | | | | | | |
| 13 | | | | | | | | |
| 14 | Restr. | Reserved | | GROUP NUMBER | | | | |
| 15 | CONTROL | | | | | | | |

See the "Verify (10)" on page 24 command for a description of the fields and operation of this command.

## Write (6)

The **WRITE (6)** command requests that the device server transfer the specified logical block or blocks from the data-out buffer and write them.

Use the following table to review the parameters for the **WRITE (6)** command.

*Table 44. Write (6) (0Ah)*

| Byte | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|
| 0 | OPERATION CODE (0Ah) | | | | | | | |
| 1 | Reserved | | | | | | | |
| 2 | LOGICAL BLOCK ADDRESS | | | | | | | |
| 3 | | | | | | | | |
| 4 | TRANSFER LENGTH | | | | | | | |
| 5 | CONTROL | | | | | | | |

The **LOGICAL BLOCK ADDRESS** field specifies the first logical block that is accessed by this command. If the logical block address exceeds the capacity of the medium, the command is ended with a status of CHECK CONDITION. The status also includes a sense key of ILLEGAL REQUEST, and an extra sense code of LOGICAL BLOCK ADDRESS OUT OF RANGE.

The **TRANSFER LENGTH** field specifies the number of contiguous logical blocks of data to be transferred from the data-out buffer and written, starting with the logical block specified by the **LOGICAL BLOCK ADDRESS** field. A **TRANSFER LENGTH** field that is set to 0 specifies that 256 logical blocks are written. If the logical block address plus the transfer length exceeds the capacity of the medium, the command is ended with a status of CHECK CONDITION. The status also includes a sense key of ILLEGAL REQUEST, and an extra sense code of LOGICAL BLOCK ADDRESS OUT OF RANGE. The **TRANSFER LENGTH** field is constrained by the **MAXIMUM TRANSFER LENGTH** field in the Block Limits VPD page.

## Write (10)

The **WRITE (10)** command requests that the device server transfer the specified logical block or blocks from the data-out buffer and write them.

Use the following table to review the parameters for the **WRITE (10)** command.

*Table 45. Write (10) (2Ah)*

| Byte | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|
| 0 | OPERATION CODE (2h) | | | | | | | |
| 1 | WRPROTECT | | | DPO | FUA | Reserved | FUA NV | Obsolete |
| 2 | LOGICAL BLOCK ADDRESS | | | | | | | |
| 3 | | | | | | | | |
| 4 | Reserved | | | GROUP NUMBER | | | | |
| 5 | TRANSFER LENGTH | | | | | | | |
| 6 | | | | | | | | |
| 7 | CONTROL | | | | | | | |

If a **WRITE (10)** command is received with the **WRPROTECT** field set to any value other than 000b, the command is ended with a status of CHECK CONDITION. The status also includes a sense key of ILLEGAL REQUEST and an extra sense code of INVALID FIELD IN CDB.

The DPO (disable page-out), **FUA** (force unit access), and **FUA NV** (force unit access nonvolatile cache) bits are ignored.

The **LOGICAL BLOCK ADDRESS** field specifies the first logical block that is accessed by this command. If the logical block address exceeds the capacity of the medium, the command is ended with a status of CHECK CONDITION. The status also includes a sense key of ILLEGAL REQUEST, and an extra sense code of LOGICAL BLOCK ADDRESS OUT OF RANGE.

The **GROUP NUMBER** field is ignored. The **TRANSFER LENGTH** field specifies the number of contiguous logical blocks of data to be transferred from the data-out buffer and written, starting with the logical block specified by the **LOGICAL BLOCK ADDRESS** field. A **TRANSFER LENGTH** field that is set to 0 specifies that no logical blocks are written. If the logical block address plus the transfer length exceeds the capacity of the medium, the command is ended with a status of CHECK CONDITION. The status also includes a sense key that is set to ILLEGAL REQUEST, and an extra sense code of LOGICAL BLOCK ADDRESS OUT OF RANGE. The **TRANSFER LENGTH** field is constrained by the **MAXIMUM TRANSFER LENGTH** field in the Block Limits VPD page.

## Write (12)

The **WRITE (12)** command requests that the device server transfer the specified logical block or blocks from the data-out buffer and write them.

Use the following table to review the parameters for the **WRITE (12)** command.

*Table 46. Write (12) (AAh)*

| Byte | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|
| 0 | OPERATION CODE (AAh) | | | | | | | |
| 1 | WRPROTECT | | | DPO | FUA | Reserved | FUA NV | Obsolete |
| 2 | LOGICAL BLOCK ADDRESS | | | | | | | |
| 5 | | | | | | | | |
| 6 | TRANSFER LENGTH | | | | | | | |
| 9 | | | | | | | | |
| 10 | Restr. | Reserved | | GROUP NUMBER | | | | |
| 11 | CONTROL | | | | | | | |

If a **WRITE (12)** command is received with the **WRPROTECT** field set to any value other than 000b, the command is ended with a status of CHECK CONDITION. The status also includes a sense key of ILLEGAL REQUEST and an extra sense code of INVALID FIELD IN CDB.

The **DPO** (disable page out), **FUA** (force unit access), and **FUA NV** (force unit access nonvolatile cache) bits are ignored.

The **LOGICAL BLOCK ADDRESS** field specifies the first logical block that is accessed by this command. If the logical block address exceeds the capacity of the medium, the command is ended with a status of CHECK CONDITION. The status also includes a sense key of ILLEGAL REQUEST, and an extra sense code of LOGICAL BLOCK ADDRESS OUT OF RANGE.

The **TRANSFER LENGTH** field specifies the number of contiguous logical blocks of data to be transferred from the data-out buffer and written, starting with the logical block specified by the **LOGICAL BLOCK ADDRESS** field. A **TRANSFER LENGTH** field that is set to 0 specifies that no logical blocks are written. If the logical block address plus the transfer length exceeds the capacity of the medium, the command is ended with a status of CHECK CONDITION. The status also includes a sense key of ILLEGAL REQUEST, and an extra sense code of LOGICAL BLOCK ADDRESS OUT OF RANGE. The **TRANSFER LENGTH** field is constrained by the **MAXIMUM TRANSFER LENGTH** field in the Block Limits VPD page.

The **GROUP NUMBER** field is ignored.

## Write (16)

The **WRITE (16)** command requests that the device server transfer the specified logical block or blocks from the data-out buffer and write them.

Use the following table to review the parameters for the **WRITE (16)** command.

*Table 47. Write (16) (8Ah)*

| Byte | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|
| 0 | OPERATION CODE (8Ah) | | | | | | | |
| 1 | WRPROTECT | | | DPO | FUA | Reserved | FUA NV | Reserved |

Table 47. Write (16) (8Ah) (continued)

| Byte | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|
| 2<br>9 | LOGICAL BLOCK ADDRESS | | | | | | | |
| 10<br>13 | TRANSFER LENGTH | | | | | | | |
| 14 | Restr. | Reserved | | GROUP NUMBER | | | | |
| 15 | CONTROL | | | | | | | |

If a **WRITE (16)** command is received with the **WRPROTECT** field set to any value other than 000b, the command is ended with a status of CHECK CONDITION. The status also includes a sense key of ILLEGAL REQUEST and an extra sense code of INVALID FIELD IN CDB.

The **DPO** (disable page out), **FUA** (force unit access), and **FUA NV** (force unit access nonvolatile cache) bits are ignored.

The **LOGICAL BLOCK ADDRESS** field specifies the first logical block that is accessed by this command. If the logical block address exceeds the capacity of the medium, the command is ended with a status of CHECK CONDITION. The status also includes a sense key of ILLEGAL REQUEST, and an extra sense code of LOGICAL BLOCK ADDRESS OUT OF RANGE.

The **TRANSFER LENGTH** field specifies the number of contiguous logical blocks of data to be transferred from the data-out buffer and written, starting with the logical block specified by the **LOGICAL BLOCK ADDRESS** field. A **TRANSFER LENGTH** field that is set to 0 specifies that no logical blocks are written. If the logical block address plus the transfer length exceeds the capacity of the medium, the command is ended with a status of CHECK CONDITION. The status also includes a sense key of ILLEGAL REQUEST, and an extra sense code of LOGICAL BLOCK ADDRESS OUT OF RANGE. The **TRANSFER LENGTH** field is constrained by the **MAXIMUM TRANSFER LENGTH** field in the Block Limits VPD page.

The **GROUP NUMBER** field is ignored.

## Write and Verify (10)

The **WRITE AND VERIFY (10)** command requests that the device server transfer the specified logical block or blocks from the data-out buffer, write them to the medium, and then verify that they are correctly written.

Use the following table to review the parameters for the **WRITE AND VERIFY (10)** command.

Table 48. Write and Verify (10) (2Eh)

| Byte | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|
| 0 | OPERATION CODE (2Eh) | | | | | | | |
| 1 | WRPROTECT | | | DPO | Reserved | | BYTCHK | Obsolete |
| 2<br>5 | LOGICAL BLOCK ADDRESS | | | | | | | |
| 6 | Reserved | | | GROUP NUMBER | | | | |
| 7<br>8 | TRANSFER LENGTH | | | | | | | |
| 9 | CONTROL | | | | | | | |

If a **WRITE AND VERIFY (10)** command is received with the **WRPROTECT** field set to any value other than 000b, the command is ended with a status of CHECK CONDITION. The status also includes a sense key of ILLEGAL REQUEST and an extra sense code of INVALID FIELD IN CDB.

The **DPO** (disable page-out) and **BYTCHK** (byte check) bits are ignored. The **LOGICAL BLOCK ADDRESS** field specifies the first logical block that is accessed by this command. If the logical block address exceeds the capacity of the medium, the command is ended with a status of CHECK CONDITION. The status also includes a sense key of ILLEGAL REQUEST, and an extra sense code of LOGICAL BLOCK ADDRESS OUT OF RANGE.

The **GROUP NUMBER** field is ignored.

The **TRANSFER LENGTH** field specifies the number of contiguous logical blocks of data to be transferred from the data-out buffer and written, starting with the logical block specified by the **LOGICAL BLOCK ADDRESS** field. A **TRANSFER LENGTH** field that is set to 0 specifies that no logical blocks are written. If the logical block address plus the transfer length exceeds the capacity of the medium, the command is ended with a status of CHECK CONDITION. The status also includes a sense key of ILLEGAL REQUEST, and an extra sense code of LOGICAL BLOCK ADDRESS OUT OF RANGE. The **TRANSFER LENGTH** field is constrained by the **MAXIMUM TRANSFER LENGTH** field in the Block Limits VPD page.

## Write and Verify (12)

The **WRITE AND VERIFY (12)** command requests that the device server transfer the specified logical block or blocks from the data-out buffer, write them to the medium, and then verify that they are correctly written.

Use the following table to review the parameters for the **WRITE AND VERIFY (12)** command.

*Table 49. Write and Verify (12) (AEh)*

| Byte | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|
| 0 | OPERATION CODE (AEh) | | | | | | | |
| 1 | WRPROTECT | | | DPO | Reserved | | BYTCHK | Obsolete |
| 2 | LOGICAL BLOCK ADDRESS | | | | | | | |
| 5 | | | | | | | | |
| 6 | TRANSFER LENGTH | | | | | | | |
| 9 | | | | | | | | |
| 10 | Restr. | Reserved | | GROUP NUMBER | | | | |
| 11 | CONTROL | | | | | | | |

See the **WRITE AND VERIFY (10)** command for a description of the fields and operation of this command.

## Write and Verify (16)

The **WRITE AND VERIFY (16)** command requests that the device server transfer the specified logical block or blocks from the data-out buffer, write them to the medium, and then verify that they are correctly written.

Use the following table to review the parameters for the **WRITE AND VERIFY (16)** command.

*Table 50. Write and Verify (16) (8Eh)*

| Byte | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|
| 0 | OPERATION CODE (8Eh) | | | | | | | |
| 1 | WRPROTECT | | | DPO | Reserved | | BYTCHK | Obsolete |

*Table 50. Write and Verify (16) (8Eh) (continued)*

| Byte | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|
| 2 | LOGICAL BLOCK ADDRESS | | | | | | | |
| 9 | | | | | | | | |
| 10 | TRANSFER LENGTH | | | | | | | |
| 13 | | | | | | | | |
| 14 | Restr. | Reserved | | GROUP NUMBER | | | | |
| 15 | CONTROL | | | | | | | |

See the **WRITE AND VERIFY (10)** command for a description of the fields and operation of this command.

## Write Buffer

The **WRITE BUFFER** command is used with the **READ BUFFER** command as a diagnostic function for testing logical unit memory in the SCSI target device and for testing the integrity of the service delivery subsystem.

The **WRITE BUFFER** does not alter the medium.

Use the following table to review the parameters for the **WRITE BUFFER** command.

*Table 51. Write Buffer (3Bh)*

| Byte | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|
| 0 | OPERATION CODE (3Bh) | | | | | | | |
| 1 | Reserved | | | MODE | | | | |
| 2 | BUFFER ID | | | | | | | |
| 3 | BUFFER OFFSET | | | | | | | |
| 5 | | | | | | | | |
| 6 | PARAMETER LIST LENGTH | | | | | | | |
| 8 | | | | | | | | |
| 9 | CONTROL | | | | | | | |

The **MODE** determines the function of this command and the meaning of the fields within the command descriptor block (CDB). If the **MODE** field contains an unsupported mode, the command is ended with a status of CHECK CONDITION. The status also includes a sense key of ILLEGAL REQUEST and an extra sense code of INVALID FIELD IN CDB.

*Table 52. Supported* **WRITE BUFFER MODE** *field values*

| Mode | Mode Description |
|---|---|
| 02h | Data |
| 0Ah | Echo buffer |
| 1Ah | Enable expander communications protocol and the echo buffer |

# Task management functions

Learn about the SCSI task management functions available for your system.

The SCSI task management supported functions follow:

- Terminate task

- Target reset
- Logical unit reset
- Clear task set
- Abort task set
- Clear the auto contingent allegiance (ACA) condition, to recover from an error and resume a queue of SCSi commands for the device.

If an unsupported task management function is received, the command is ended with an appropriate no support response that depends on the transport protocol in use.

## Inquiry responses

Learn about the various inquiry responses.

### Standard inquiry data

When an **INQUIRY** command is received with the **EVPD** bit that is set to 0, the device returns standard inquiry data.

All **Obsolete** and **Reserved** fields are set to 0.

The following table lists the response to a standard inquiry request.

*Table 53. Standard Inquiry data*

| Byte | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|
| 0 | PERIPHERAL QUALIFIER | | | PERIPHERAL DEVICE TYPE | | | | |
| 1 | RMB | Reserved | | | | | | |
| 2 | VERSION=05h | | | | | | | |
| 3 | Obsolete | Obsolete | NACA=0 | HS=1h | RESPONSE DATA FORMAT=2h | | | |
| 4 | ADDITIONAL LENGTH=5Bh | | | | | | | |
| 5 | SCCS=0 | ACC=0 | TPGS=0 | | 3PC=0 | Reserved | | PRT=0 |
| 6 | BQ=0 | ES=0 | VS=0 | MP=1h | MC=0 | Obsolete | | A16=0 |
| 7 | Obsolete | | WB16=0 | SYNC=0 | LINK=0 | Obsolete | CQ=1h | VS=0 |
| 8 | T10 VENDOR IDENTIFICATION | | | | | | | |
| 15 | | | | | | | | |
| 16 | PRODUCT IDENTIFICATION | | | | | | | |
| 31 | | | | | | | | |
| 32 | PRODUCT REVISION LEVEL | | | | | | | |
| 35 | | | | | | | | |
| 36 | Vendor specific | | | | | | | |
| 55 | | | | | | | | |
| 56 | Reserved | | | | CLOCKING=0 | | QAS=0 | IUS=0 |
| 57 | Reserved | | | | | | | |
| 58 | VERSION DESCRIPTOR 1 | | | | | | | |
| 59 | | | | | | | | |
| 72 | VERSION DESCRIPTOR 8 | | | | | | | |
| 73 | | | | | | | | |

Table 53. Standard Inquiry data (continued)

| Byte | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|
| 74 | Reserved | | | | | | | |
| 95 | | | | | | | | |

Table 54. Reported **PERIPHERAL QUALIFIER** field values

| PQ | Description |
|---|---|
| 000b | The logical unit is a valid logical unit, and the host that sends the command has access to the logical unit. That is, any further commands that are sent by this host to this logical unit are accepted into the task queue. |
| 001b | The logical unit on which the command is received is LUN 0, but that logical unit is not configured or the host that sent the command does not have access to the logical unit. That is, any commands that are sent by this host to this logical unit are not accepted into the task queue. |
| 011b | The logical unit on which the command is received is not LUN 0 and the logical unit is not configured. That is, any commands that are sent by this host to this logical unit are not accepted into the task queue. |

Table 55. Reported **PERIPHERAL DEVICE TYPE** field values

| PQ | Description |
|---|---|
| 00h | The logical unit is a direct-access logical unit. This value is returned if the **PERIPHERAL QUALIFIER** field contains a 000b or 001b. |
| 1Fh | The logical unit type is unknown. This value is returned if the **PERIPHERAL QUALIFIER** field contains a 011b. |

The **PRODUCT REVISION** field contains 4 ASCII digits that represent the product revision. The product revision changes with each firmware release. Bytes 36 - 45 (the first 10 bytes of the Vendor-specific field) contain the logical-unit serial number (that is, the same number reported in the **INQUIRY VPD** field). The format of the serial number is 10 ASCII digits, with the first 6 digits representing the IEEE-registered Company ID followed by 4 digits that represent the logical unit.

Table 56. Fibre Channel **VERSION DESCRIPTOR** field values

| Number | Value | Description |
|---|---|---|
| 1 | 0060h | SAM-3 (no version) |
| 2 | 0d80h | FC-PH-3 (no version) |
| 3 | 0d60h | FC-AL-2 (no version) |
| 4 | 0a00h | FCP-3 (no version) |
| 5 | 0300h | SPC-3 (no version) |
| 6 | 0320h | SBC-2 (no version) |
| 7 | 0000h | No standard identified |
| 8 | 0000h | No standard identified |

Table 57. InfiniBand **VERSION DESCRIPTOR** field values

| Number | Value | Description |
|---|---|---|
| 1 | 0060h | SAM-3 (no version) |
| 2 | 0940h | SRP (no version) |
| 3 | 0300h | SPC-3 (no version) |

*Table 57. InfiniBand* **VERSION DESCRIPTOR** *field values (continued)*

| Number | Value | Description |
|---|---|---|
| 4 | 0320h | SBC-2 (no version) |
| 5 | 0000h | No standard identified |
| 6 | 0000h | No standard identified |
| 7 | 0000h | No standard identified |
| 8 | 0000h | No standard identified |

## Vital product data

When an **INQUIRY** command is received with the **EVPD** bit that is set to 1, the device returns the vital product data (VPD) page that was requested.

Use the following tables to review the parameters for the VPD pages.

*Table 58. Supported VPD Pages (00h)*

| Byte | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|
| 0 | PERIPHERAL QUALIFIER | | | PERIPHERAL DEVICE TYPE | | | | |
| 1 | PAGE CODE=00h | | | | | | | |
| 2 | Reserved | | | | | | | |
| 3 | PAGE LENGTH=06h | | | | | | | |
| 4 | 00h | | | | | | | |
| 5 | 80h | | | | | | | |
| 6 | 83h | | | | | | | |
| 7 | 85h | | | | | | | |
| 8 | B0h | | | | | | | |
| 9 | B1h | | | | | | | |

For a description of the **PERIPHERAL QUALIFIER** and **PERIPHERAL DEVICE TYPE** fields, see "Standard inquiry data" on page 31.

*Table 59. Unit Serial Number Page (80h)*

| Byte | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|
| 0 | PERIPHERAL QUALIFIER | | | PERIPHERAL DEVICE TYPE | | | | |
| 1 | PAGE CODE=80h | | | | | | | |
| 2 | Reserved | | | | | | | |
| 3 | PAGE LENGTH=0Ah | | | | | | | |
| 4 | PRODUCT SERIAL NUMBER | | | | | | | |
| 13 | | | | | | | | |

For a description of the **PERIPHERAL QUALIFIER** and **PERIPHERAL DEVICE TYPE** fields, see "Standard inquiry data" on page 31.

*Table 60. PRODUCT SERIAL NUMBER field*

| Byte | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|
| 0 | IEEE Company ID | | | | | | | |
| 5 | | | | | | | | |
| 6 | Logical Unit | | | | | | | |
| 9 | | | | | | | | |

*Table 61. Device Identification Page (83h)*

| Byte | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|
| 0 | PERIPHERAL QUALIFIER | | | PERIPHERAL DEVICE TYPE | | | | |
| 1 | PAGE CODE=83h | | | | | | | |
| 2 | Reserved | | | | | | | |
| 3 | PAGE LENGTH=46h | | | | | | | |
| 4 | PROTOCOL IDENTIFIER | | | | CODE SET=2h | | | |
| 5 | PIV=0 I | Reserved | ASSOC.=00b | | IDENTIFIER TYPE=1h | | | |
| 6 | Reserved | | | | | | | |
| 7 | IDENTIFIER LENGTH=22h | | | | | | | |
| 8 | T10 VENDOR IDENTIFICATION | | | | | | | |
| 15 | | | | | | | | |
| 16 | PRODUCT IDENTIFICATION | | | | | | | |
| 31 | | | | | | | | |
| 32 | SERIAL NUMBER | | | | | | | |
| 41 | | | | | | | | |
| 42 | | | | | | | | |
| 43 | PIV=0 I | Reserved | ASSOC.=00b | | IDENTIFIER TYPE=2h | | | |
| 44 | Reserved | | | | | | | |
| 45 | IDENTIFIER LENGTH=08h | | | | | | | |
| 46 | IEEE COMPANY ID | | | | | | | |
| 48 | | | | | | | | |
| 49 | VENDOR SPECIFIC EXTENSION IDENTIFIER (LU) | | | | | | | |
| 50 | | | | | | | | |
| 51 | VENDOR SPECIFIC EXTENSION IDENTIFIER (SNb) | | | | | | | |
| 53 | | | | | | | | |
| 54 | PROTOCOL IDENTIFIER | | | | CODE SET=1h | | | |
| 55 | PIV=1 | Reserved | ASSOC.=01b | | IDENTIFIER TYPE=4h | | | |
| 56 | Reserved | | | | | | | |
| 57 | IDENTIFIER LENGTH=04h | | | | | | | |
| 58 | Obsolete | | | | | | | |
| 59 | | | | | | | | |
| 60 | RELATIVE TARGET PORT IDENTIFIER | | | | | | | |
| 61 | | | | | | | | |
| 62 | PROTOCOL IDENTIFIER | | | | CODE SET=1h | | | |

*Table 61. Device Identification Page (83h) (continued)*

| Byte | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|
| 63 | PIV=1 | Reserved | ASSOC.=00b | | IDENTIFIER TYPE=2h | | | |
| 64 | Reserved | | | | | | | |
| 65 | IDENTIFIER LENGTH=08h | | | | | | | |
| 66 | IEEE COMPANY ID | | | | | | | |
| 68 | | | | | | | | |
| 69 | VENDOR SPECIFIC EXTENSION IDENTIFIER | | | | | | | |
| 70 | | | | | | | | |
| 71 | VENDOR SPECIFIC EXTENSION IDENTIFIER | | | | | | | |
| 73 | | | | | | | | |

For a description of the **PERIPHERAL QUALIFIER** and **PERIPHERAL DEVICE TYPE** fields, see "Standard inquiry data" on page 31.

If the **ASSOCIATION** field contains a 01b (that is, a target port) or 10b (that is, a SCSI target device) and the **PIV** bit is set to 1, the **PROTOCOL IDENTIFIER** field contains 0h if the interface is Fibre Channel. Alternatively, if the interface uses InfiniBand technology, the **PROTOCOL IDENTIFIER** field contains 4h. Otherwise, the **PROTOCOL IDENTIFIER** field can be ignored.

The **SERIAL NUMBER** field contains ASCII data for the product's vendor-assigned serial number.

*Table 62. Management Network Addresses Page (85h)*

| Byte | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|
| 0 | PERIPHERAL QUALIFIER | | | PERIPHERAL DEVICE TYPE | | | | |
| 1 | PAGE CODE=85h | | | | | | | |
| 2 | Reserved | | | | | | | |
| 3 | PAGE LENGTH (n-3) Network services descriptor list (4-n) | | | | | | | |
| 4 | Reserved | ASSOC.=10b | | SERVICE TYPE=03h | | | | |
| 5 | Reserved | | | | | | | |
| 6 | NETWORK ADDRESS LENGTH (n-7) | | | | | | | |
| 7 | | | | | | | | |
| 8 | NETWORK ADDRESS | | | | | | | |
| n | | | | | | | | |

For a description of the **PERIPHERAL QUALIFIER** and **PERIPHERAL DEVICE TYPE** fields, see "Standard inquiry data" on page 31.

The **NETWORK ADDRESS** field contains a null-padded and null-ended ASCII digit that represents the system management IP address in the form `scheme://xxx.xxx.xxx.xxx`. One or more IP addresses can be specified, depending on the system configuration.

*Table 63. Block Limits Page (B0h)*

| Byte | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|
| 0 | PERIPHERAL QUALIFIER | | | PERIPHERAL DEVICE TYPE | | | | |
| 1 | PAGE CODE=B0h | | | | | | | |

Table 63. Block Limits Page (B0h) (continued)

| Byte | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|
| 2 | Reserved | | | | | | | |
| 3 | PAGE LENGTH (0Ch) | | | | | | | |
| 4 | Reserve | | | | | | | |
| 5 | | | | | | | | |
| 6 | OPTIMAL TRANSFER LENGTH GRANULARITY | | | | | | | |
| 7 | | | | | | | | |
| 8 | MAXIMUM TRANSFER LENGTH | | | | | | | |
| 11 | | | | | | | | |
| 12 | OPTIMAL TRANSFER LENGTH | | | | | | | |
| 15 | | | | | | | | |

Table 64. Block Device Characteristics Addresses Page (B1h)

| Byte | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|
| 0 | PERIPHERAL QUALIFIER | | | PERIPHERAL DEVICE TYPE | | | | |
| 1 | PAGE CODE=B1h | | | | | | | |
| 2 | PAGE LENGTH (003Ch) | | | | | | | |
| 3 | | | | | | | | |
| 4 | MEDIUM ROTATION RATE | | | | | | | |
| 5 | | | | | | | | |
| 6 | Reserved | | | | | | | |
| 7 | Reserved | | | | NOMINAL FORM FACTOR | | | |
| 8 | Reserved | | | | | | | |
| 63 | | | | | | | | |

For a description of the **PERIPHERAL QUALIFIER** and **PERIPHERAL DEVICE TYPE** fields, see "Standard inquiry data" on page 31.

## Mode page parameters

Learn about the supported mode parameters.

Use the following table to review the supported mode page parameters.

Table 65. Supported Mode Pages

| Page code | Page name |
|---|---|
| 01h | Read/write error recovery page |
| 02h | Disconnect/reconnect page |
| 03h | Format device page |
| 04h | Rigid disk device geometry page |
| 07h | Verify error recovery page |
| 08h | Caching page |
| 0Ah | Control page |

*Table 65. Supported Mode Pages (continued)*

| Page code | Page name |
|---|---|
| 0Bh | Medium types supported page |
| 18h | Fibre Channel specific LUN page |
| 19h | Fibre Channel specific port page |
| 1Ch | Information exceptions control page |

The following subtopics describe the supported mode parameter payloads. The parameters that are listed are returned to the data-out buffer when a **MODE SENSE** command is run, and are returned when some support is being changed to the parameters that are specified in the data-out buffer when a MODE SELECT command is run.

## Read/write error recovery

The read/write error recovery mode page specifies the error recovery parameters that the device server will use for any command that runs a read or write operation to the medium (for example, **READ** commands, **WRITE** commands, and **WRITE AND VERIFY** commands).

Use the following table to review the parameters for the read/write recovery mode page.

*Table 66. Read/write error recovery page*

| Byte | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|
| 0 | PS=0 | SPF=0 | PAGE CODE=01h 1 3 | | | | | |
| 1 | PAGE LENGTH=0Ah | | | | | | | |
| 2 | AWRE=0 | ARRE=0 | TB=0 | RC=0 | EER=0 | 2PER=0 | DTE=0 | DCR=0 |
| 3 | READ RETRY COUNT=0 | | | | | | | |
| 4 | Obsolete | | | | | | | |
| 6 | | | | | | | | |
| 7 | Reserved | | | | | | | Rest. |
| 8 | WRITE RETRY COUNT=0 | | | | | | | |
| 9 | Reserved | | | | | | | |
| 10 | RECOVERY TIME LIMIT=0 | | | | | | | |
| 11 | | | | | | | | |

This mode page is only readable by the **MODE SENSE** command. The mode page cannot be changed by the **MODE SELECT** command.

## Disconnect/reconnect

The disconnect/reconnect mode page provides the application client the means to tune the performance of the service delivery subsystem.

Use the following table to learn about the disconnect/reconnect page parameters.

*Table 67. Disconnect/reconnect page*

| Byte | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|
| 0 | PS=0 | SPF=0 | PAGE CODE=02h | | | | | |
| 1 | PAGE LENGTH=0Eh | | | | | | | |
| 2 | BUFFER FULL RATIO=0 | | | | | | | |

*Table 67. Disconnect/reconnect page (continued)*

| Byte | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|
| 3 | BUFFER EMPTY RATIO=0 | | | | | | | |
| 4 | BUS INACTIVITY LIMIT=0 | | | | | | | |
| 5 | | | | | | | | |
| 6 | DISCONNECT TIME LIMIT=03 | | | | | | | |
| 7 | | | | | | | | |
| 8 | CONNECT TIME LIMIT=0 | | | | | | | |
| 9 | | | | | | | | |
| 10 | MAXIMUM BURST SIZE=0 | | | | | | | |
| 11 | | | | | | | | |
| 12 | EMDP=0 | FAIR ARB.=0 | | | DIMM=0 | DTDC=0 | | |
| 13 | Reserved | | | | | | | |
| 14 | FIRST BURST SIZE=0 | | | | | | | |
| 15 | | | | | | | | |

This mode page is only readable by the **MODE SENSE** command. The mode page cannot be changed by the **MODE SELECT** command.

## Format device

The format device page contains parameters that specify the medium format.

*Table 68. Format device page*

| Byte | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|
| 0 | PS=0 | SPF=0 | PAGE CODE=03h | | | | | |
| 1 | PAGE LENGTH=16h | | | | | | | |
| 2 | TRACKS PER ZONE=0 | | | | | | | |
| 3 | | | | | | | | |
| 4 | ALTERNATE SECTORS PER ZONE=0 | | | | | | | |
| 5 | | | | | | | | |
| 6 | ALTERNATE TRACKS PER ZONE=0 | | | | | | | |
| 7 | | | | | | | | |
| 8 | ALTERNATE TRACKS PER LOGICAL UNIT=0 | | | | | | | |
| 9 | | | | | | | | |
| 10 | SECTORS PER TRACK=0020h | | | | | | | |
| 11 | | | | | | | | |
| 12 | DATA BYTES PER PHYSICAL SECTOR=0200h | | | | | | | |
| 13 | | | | | | | | |
| 14 | INTERLEAVE=0 | | | | | | | |
| 15 | | | | | | | | |
| 16 | TRACK SKEW FACTOR=0 | | | | | | | |
| 17 | | | | | | | | |

*Table 68. Format device page (continued)*

| Byte | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|
| 18 | CYLINDER SKEW FACTOR=0 | | | | | | | |
| 19 | | | | | | | | |
| 20 | SSEC=1 | HSEC=1 | RMB=0 | SURF=0 | Reserved | | | |
| 21 | Reserved | | | | | | | |
| 23 | | | | | | | | |

This mode page is only readable by the **MODE SENSE** command. The mode page cannot be changed by the **MODE SELECT** command.

## Verify error recovery

The verify error recovery mode page specifies the error recovery parameters that the device server uses when the **VERIFY** command is run and specifies the verify operation of the **WRITE AND VERIFY** command.

Use the following table to review the parameters for the verify error recovery mode page.

*Table 69. Verify error recovery page*

| Byte | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|
| 0 | PS=0 | SPF=0 | PAGE CODE=07h | | | | | |
| 1 | PAGE LENGTH=0Ah | | | | | | | |
| 2 | Reserved | | | | EER=0 | PER=0 | DTE=0 | DCR=0 |
| 3 | VERIFY RETRY COUNT=0 | | | | | | | |
| 4 | Obsolete | | | | | | | |
| 5 | Reserved | | | | | | | |
| 9 | | | | | | | | |
| 10 | VERIFY RECOVERY TIME LIMIT=0 | | | | | | | |
| 11 | | | | | | | | |

This mode page is only readable by the **MODE SENSE** command. The mode page cannot be changed by the **MODE SELECT** command.

## Caching

The caching mode page defines the parameters that affect the use of the cache.

*Table 70. Caching page*

| Byte | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|
| 0 | | | PS=0 SPF=0 PAGE CODE=08h | | | | | |
| 1 | PAGE LENGTH=12h | | | | | | | |
| 2 | IC=0 | ABPF=0 | CAP=0 | DISC=0 | SIZE=0 | WCE | MF=0 | RCD=0 |
| 3 | DEMAND READ RETENTION PRIORITY=0 | | | | WRITE RETENTION PRIORITY=0 | | | |
| 4 | DISABLE PRE-FETCH TRANSFER LENGTH=0 | | | | | | | |
| 5 | | | | | | | | |
| 6 | MINIMUM PRE-FETCH=0 | | | | | | | |
| 7 | | | | | | | | |

Table 70. Caching page (continued)

| Byte | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|
| 8 | MAXIMUM PRE-FETCH=0 | | | | | | | |
| 9 | | | | | | | | |
| 10 | MAXIMUM PRE-FETCH CEILING=0 | | | | | | | |
| 11 | | | | | | | | |
| 12 | FSW=0 | LBCSS=0 | DRA=0 | Vendor specific=0 | | Reserved | | NVDIS=0 |
| 13 | NUMBER OF CACHE SEGMENTS=0 | | | | | | | |
| 14 | CACHE SEGMENT SIZE=0 | | | | | | | |
| 15 | | | | | | | | |
| 16 | Reserved | | | | | | | |
| 17 | Obsolete | | | | | | | |
| 19 | | | | | | | | |

This mode page is readable by the **MODE SENSE** command, and the **WCE** bit can be changed by the **MODE SELECT** command. No other fields can be changed.

The write cache-enabled **WCE** bit specifies whether write caching is enabled for this logical unit. By default, this bit is set to 1. If the bit was changed by a **MODE SELECT** command, the bit can return 0.

## Control

The control mode page provides control over SCSI features that are applicable to all device types (for example, task set management and error logging).

Table 71. Control page

| Byte | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|
| 0 | | | PS=0 SPF=0 PAGE CODE=0Ah | | | | | |
| 1 | PAGE LENGTH=0Ah | | | | | | | |
| 2 | TST=0 | | | TMF=0 | Reserved | DSENSE | GLTSD | RLEC=0 |
| 3 | QUEUE ALGORITHM MODIFIER=1h | | | | Reserved | QERR=0 | Obsolete | |
| 4 | VS=0 | RAC=0 | UA INTLCK CTRL=0 | | SWP=0 | Obsolete | | |
| 5 | ATO=0 | TAS=0 | Reserved | | | AUTOLOAD MODE=0 | | |
| 6 | Obsolete | | | | | | | |
| 7 | | | | | | | | |
| 8 | BUSY TIMEOUT PERIOD=FFFFh | | | | | | | |
| 9 | | | | | | | | |
| 10 | EXTENDED SELF-TEST COMPLETION TIME=0010h | | | | | | | |
| 11 | | | | | | | | |

This mode page is only readable by the **MODE SENSE** command. The mode page cannot be changed by the **MODE SELECT** command.

## Medium types supported

The medium types supported mode page contains a list of the medium types that are implemented by the device server for logical units.

*Table 72. Medium types supported page*

| Byte | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|
| 0 | PS=0 | SPF=0 | PAGE CODE=0Bh | | | | | |
| 1 | PAGE LENGTH=06h | | | | | | | |
| 2 | Reserved | | | | | | | |
| 3 | | | | | | | | |
| 4 | MEDIUM TYPE ONE SUPPORTED=0 | | | | | | | |
| 5 | MEDIUM TYPE TWO SUPPORTED=0 | | | | | | | |
| 6 | MEDIUM TYPE THREE SUPPORTED=0 | | | | | | | |
| 7 | MEDIUM TYPE FOUR SUPPORTED=0 | | | | | | | |

This mode page is only readable by the **MODE SENSE** command. The mode page cannot be changed by the **MODE SELECT** command.

## Fibre Channel logical-unit control

The Fibre Channel logical-unit control mode page contains those parameters that select Fibre Channel Protocol (FCP) logical unit operations.

This mode page is only readable by the **MODE SENSE** command. The mode page cannot be changed by the **MODE SELECT** command.

*Table 73. Fibre Channel logical unit control page*

| Byte | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|
| 0 | PS=0 | SPF=0 | PAGE CODE=18h | | | | | |
| 1 | PAGE LENGTH=06h | | | | | | | |
| 2 | Reserved | | | | PROTOCOL IDENTIFIER=0h | | | |
| 3 | Reserved | | | | | | | EPDC=0 |
| 4 | Reserved | | | | | | | |
| 5 | | | | | | | | |

This mode page is only readable by the **MODE SENSE** command. The mode page cannot be changed by the **MODE SELECT** command.

## Fibre Channel port control

The Fibre Channel port-control mode page contains those parameters that select the Fibre Channel protocol (FCP) operation options.

*Table 74. Fibre Channel port-control page*

| Byte | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|
| 0 | PS=0 | SPF=0 | PAGE CODE=19h | | | | | |
| 1 | PAGE LENGTH=06h | | | | | | | |
| 2 | Reserved | | | | PROTOCOL IDENTIFIER=0h | | | |
| 3 | DTFD=0 | PLPB=0 | DDIS=0 | DLM=0 | RHA=0 | ALWI=0 | DTIPE=0 | DTOLI=0 |
| 4 | Reserved | | | | | | | |
| 5 | Reserved | | | | | | | |
| 6 | Reserved | | | | | RR TOV UNITS=0 | | |

Table 74. Fibre Channel port-control page (continued)

| Byte | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|
| 7 | SEQ. INITIATIVE RESOURCE RECOVERY TIMEOUT VALUE (RR TOV SEQ INIT)=0 | | | | | | | |

This mode page is only readable by the **MODE SENSE** command. The mode page cannot be changed by the **MODE SELECT** command.

### Informational exceptions control

The informational-exceptions-control mode page defines the methods that are used by the device server to control the reporting and the operations of specific informational exception conditions.

This information only applies to informational exceptions that report an extra sense code of FAILURE PREDICTION THRESHOLD EXCEEDED or an extra sense code of WARNING to the application client.

Table 75. Informational-exceptions-control page

| Byte | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|
| 0 | PS=0 | SPF=0 | PAGE CODE=1Ch | | | | | |
| 1 | PAGE LENGTH=0Ah | | | | | | | |
| 2 | PERF=0 | Reserved | EBF=0 | EW=0 | DE=0 | TEST=0 | Reserved | LE=0 |
| 3 | | | | | | | | |
| 4 | INTERVAL TIMER=0 | | | | | | | |
| 7 | | | | | | | | |
| 8 | REPORT COUNT=0 | | | | | | | |
| 11 | | | | | | | | |

This mode page is only readable by the **MODE SENSE** command. this mode cannot be changed by the **MODE SELECT** command.

## Fibre Channel command set

Learn about the **Fibre Channel** command set including supported versions, supported Fibre Channel link services, and Fibre Channel login options.

## Supported versions

Learn about the Fibre Channel specification versions supported by the storage system.

The Fibre Channel specifications supported by the storage system include:

- Fibre Channel Protocol - 3 (FCP-3)
- Fibre Channel 3rd Generation Physical Interface (FC-PH-3)
- Fibre Channel Arbitrated Loop (FC-AL-2)

## Supported Fibre Channel link services

Learn about the supported Fibre Channel link services for each type of frame. The frames include basic link service, basic link service reply, extended link service, and extended link service reply.

Table 76. Link services supported

| Type of frame | Link service |
|---|---|
| Basic link service frames | Abort Sequence (ABTS), Remove Connection (RMC), and No Operation (NOP) |

*Table 76. Link services supported (continued)*

| Type of frame | Link service |
|---|---|
| Basic link service reply frames | Basic Accept (BA ACC), and Basic Reject (BA RJT) |
| Extended link service frames | N Port Login (PLOGI), Fabric Login (FLOGI), Logout (LOGO), Read Timeout Value (RTV), Read Link Error Status Block (RLS), Echo (ECHO), Test (TEST), Reinstate Recovery Qualifier (RRQ), Read Exchange Concise (REC), Process Login (PRLI), Process Logout (PRLO), Address Discovery (ADISC), Port Discovery (PDISC), Fabric Address Notification (FAN), Registered State Change Notification (RSCN), and State Change Registration (SCR) |
| Extended link service reply frames | Accept (ACC) and Link Service Reject (LS RJT) |

## Fibre Channel port login

Learn about the Fibre Channel port login content layout.

*Table 77. Port login (PLOGI) payload*

| Bytes | Byte contents | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 - 15 | 03 | XX | XX | XX | XX | XX | XX | CR | XX | FS | FS | XX | XX | XX | XX | XX |
| 16 - 31 | XX | XX | XX | XX | XX | PN | PN | PN | PN | PN | PN | PN | NN | NN | NN | NN |
| 32 - 47 | NN | NN | NN | NN | XX | XX | XX | XX | XX | XX | XX | XX | XX | XX | XX | XX |
| 48 - 115 | XX | XX | XX | XX | XX | XX | XX | XX | XX | XX | XX | XX | XX | XX | XX | XX |

**Legend**

**CR** Credit Limit

**FS** Receive buffer field size

**NN** Node name

**PN** Port name

**XX** Ignored

## Fibre Channel process login

Learn about the Fibre Channel process login.

*Table 78. Fibre Channel process login (PRLI) payload*

| Bytes | Byte contents | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 - 15 | 20 | XX | XX | XX | XX | XX | XX | XX | XX | XX | XX | XX | XX | XX | XX | XX |
| 16 - 19 | XX | XX | XX | XX | | | | | | | | | | | | |

- XX - Ignored

## Fibre Channel fabric login accept page

Learn about the Fibre Channel fabric login accept (ACC) page

*Table 79. Fibre Channel fabric login accept (ACC)*

| Bytes | Byte contents | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 - 15 | 02 | XX | XX | XX | XX | XX | XX | CR | 1X | FS | FS | XX | XX | XX | XX | XX |

*Table 79. Fibre Channel fabric login accept (ACC) (continued)*

| Bytes | Byte contents | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 16 - 31 | XX | XX | XX | XX | PN | PN | PN | PN | PN | PN | PN | PN | NN | NN | NN | NN |
| 32 - 47 | NN | NN | NN | NN | XX | XX | XX | XX | XX | XX | XX | XX | XX | XX | XX | XX |
| 48 - 115 | XX | XX | XX | XX | XX | XX | XX | XX | XX | XX | XX | XX | XX | XX | XX | XX |

**Legend**

**CR** Credit Limit

**FS** Receive buffer field size

**NN** Node name

**PN** Port name

**XX** Ignored

## Task management functions

Only one task management function can be specified in any frame.

*Table 80. Fibre Channel SCSI FCP task management functions*

| Task name | Bit location |
|---|---|
| Terminate task | 1000 0000b |
| Clear ACA | 0100 0000b |
| Target reset | 0010 0000b |
| Logical unit reset | 0001 0000b |
| Clear task set | 0000 0100b |
| Abort task set | 0000 0010b |

## InfiniBand information units

Learn about the InfiniBand information units.

## Supported versions

Learn about the supported versions of the InfiniBand specification.

The InfiniBand specifications supported by the storage system follow:

- SRP
- IBTA-1.2

## Port login payload

Learn about the port login payload.

The login content includes the following information:

*Table 81. Login request (SRP LOGIN REQ) payload*

| Bytes | Byte contents | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 - 15 | 00 | XX | XX | XX | XX | XX | XX | XX | TG | TG | TG | TG | TG | TG | TG | TG |

*Table 81. Login request (SRP LOGIN REQ) payload (continued)*

| Bytes | Byte contents | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 16 - 31 | ML | ML | ML | ML | XX | XX | XX | XX | BF | BF | X0 | PN | NN | NN | NN | NN |
| 32 - 47 | IP | IP | IP | IP | IP | IP | IP | IP | IP | IP | IP | IP | IP | IP | IP | IP |
| 48 - 63 | TP | TP | TP | TP | TP | TP | TP | TP | TP | TP | TP | TP | TP | TP | TP | TP |

**Legend**

**BF** Required buffer

**IP** Initiator port identifier

**ML** Requested maximum initiator to target IU length

**NN** Node name

**TG** Tag

**TP** Target port identifier

**XX** Ignored

## Task management functions

Only one task management function can be specified in any frame.

*Table 82. InfiniBand SCSI SRP task management functions*

| Task Name | Bit location |
|---|---|
| Terminate task | 1000 0001b |
| Clear ACA | 0100 0000b |
| Logical unit reset | 0000 1000b |
| Clear task set | 0000 0100b |
| Abort task set | 0000 0010b |

## Sense data

Learn about the SCSI sense keys including the key or code and name.

The following tables list the sense keys.

*Table 83. SCSI sense keys*

| Key | Name |
|---|---|
| 00h | No sense |
| 01h | Recovered error |
| 02h | Not ready |
| 03h | Medium error |
| 05h | Illegal request |
| 06h | Unit attention |
| 0Bh | Aborted command |

*Table 84. Additional sense codes*

| Code | Name |
|---|---|
| 0000h | No additional sense information |
| 1100h | Unrecoverable read error |
| 1800h | Recovered data with error-correcting code (ECC) |
| 1a00h | Parameter list length error |
| 2000h | Invalid command operation |
| 2100h | Logical block address out of range |
| 2400h | Invalid field In command descriptor block (CDB) |
| 2500h | LUN not supported |
| 2900h | Device reset occurred |
| 2c00h | Command sequence error reported |
| 3f0eh | LUN data has changed |
| 4900h | Invalid message error |
| 4e00h | Overlapped commands attempted |
| 5504h | Insufficient registration resources |

# Notices

This information was developed for products and services offered in the U.S.A.

The manufacturer may not offer the products, services, or features discussed in this document in other countries. Consult the manufacturer's representative for information on the products and services currently available in your area. Any reference to the manufacturer's product, program, or service is not intended to state or imply that only that product, program, or service may be used. Any functionally equivalent product, program, or service that does not infringe any intellectual property right of the manufacturer may be used instead. However, it is the user's responsibility to evaluate and verify the operation of any product, program, or service.

The manufacturer may have patents or pending patent applications covering subject matter described in this document. The furnishing of this document does not grant you any license to these patents. You can send license inquiries, in writing, to the manufacturer.

For license inquiries regarding double-byte character set (DBCS) information, contact the Intellectual Property Department in your country or send inquiries, in writing, to the manufacturer.

**The following paragraph does not apply to the United Kingdom or any other country where such provisions are inconsistent with local law:** THIS PUBLICATION IS PROVIDED “AS IS ” WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF NON-INFRINGEMENT, 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 information could include technical inaccuracies or typographical errors. Changes are periodically made to the information herein; these changes will be incorporated in new editions of the publication. The manufacturer may make improvements and/or changes in the product(s) and/or the program(s) described in this publication at any time without notice.

Any references in this information to websites not owned by the manufacturer are provided for convenience only and do not in any manner serve as an endorsement of those websites. The materials at those websites are not part of the materials for this product and use of those websites is at your own risk.

The manufacturer may use or distribute any of the information you supply in any way it believes appropriate without incurring any obligation to you.

Licensees of this program who wish to have information about it for the purpose of enabling: (i) the exchange of information between independently created programs and other programs (including this one) and (ii) the mutual use of the information which has been exchanged, should contact the manufacturer.

Such information may be available, subject to appropriate terms and conditions, including in some cases, payment of a fee.

The licensed program described in this document and all licensed material available for it are provided by IBM under terms of the IBM Customer Agreement, IBM International Program License Agreement, IBM License Agreement for Machine Code, or any equivalent agreement between us.

Any performance data contained herein was determined in a controlled environment. Therefore, the results obtained in other operating environments may vary significantly. Some measurements may have been made on development-level systems and there is no guarantee that these measurements will be the same on generally available systems. Furthermore, some measurements may have been estimated through extrapolation. Actual results may vary. Users of this document should verify the applicable data for their specific environment.

Information concerning products not produced by this manufacturer was obtained from the suppliers of those products, their published announcements or other publicly available sources. This manufacturer has not tested those products and cannot confirm the accuracy of performance, compatibility or any other claims related to products not produced by this manufacturer. Questions on the capabilities of products not produced by this manufacturer should be addressed to the suppliers of those products.

All statements regarding the manufacturer's future direction or intent are subject to change or withdrawal without notice, and represent goals and objectives only.

The manufacturer's prices shown are the manufacturer's suggested retail prices, are current and are subject to change without notice. Dealer prices may vary.

This information is for planning purposes only. The information herein is subject to change before the products described become available.

This information contains examples of data and reports used in daily business operations. To illustrate them as completely as possible, the examples include the names of individuals, companies, brands, and products. All of these names are fictitious and any similarity to the names and addresses used by an actual business enterprise is entirely coincidental.

COPYRIGHT LICENSE:

This information contains sample application programs in source language, which illustrate programming techniques on various operating platforms. You may copy, modify, and distribute these sample programs in any form without payment to the manufacturer, for the purposes of developing, using, marketing or distributing application programs conforming to the application programming interface for the operating platform for which the sample programs are written. These examples have not been thoroughly tested under all conditions. The manufacturer, therefore, cannot guarantee or imply reliability, serviceability, or function of these programs. The sample programs are provided "AS IS", without warranty of any kind. The manufacturer shall not be liable for any damages arising out of your use of the sample programs.

Each copy or any portion of these sample programs or any derivative work, must include a copyright notice as follows:

© (your company name) (year). Portions of this code are derived from IBM Corp. Sample Programs. © Copyright IBM Corp. _enter the year or years_.

If you are viewing this information in softcopy, the photographs and color illustrations may not appear.

## Trademarks

IBM, the IBM logo, and ibm.com are trademarks or registered trademarks of International Business Machines Corp., registered in many jurisdictions worldwide. Other product and service names might be trademarks of IBM or other companies. A current list of IBM trademarks is available on the web at Copyright and trademark information at www.ibm.com/legal/copytrade.shtml.

INFINIBAND, InfiniBand Trade Association, and the INFINIBAND design marks are trademarks and/or service marks of the INFINIBAND Trade Association.

## Terms and conditions

Permissions for the use of these publications are granted subject to the following terms and conditions.

**Applicability:** These terms and conditions are in addition to any terms of use for the IBM website.

**Personal Use:** You may reproduce these publications for your personal, noncommercial use provided that all proprietary notices are preserved. You may not distribute, display or make derivative works of these publications, or any portion thereof, without the express consent of IBM.

**Commercial Use:** You may reproduce, distribute and display these publications solely within your enterprise provided that all proprietary notices are preserved. You may not make derivative works of these publications, or reproduce, distribute or display these publications or any portion thereof outside your enterprise, without the express consent of IBM.

**Rights:** Except as expressly granted in this permission, no other permissions, licenses or rights are granted, either express or implied, to the Publications or any information, data, software or other intellectual property contained therein.

IBM reserves the right to withdraw the permissions granted herein whenever, in its discretion, the use of the publications is detrimental to its interest or, as determined by IBM, the above instructions are not being properly followed.

You may not download, export or re-export this information except in full compliance with all applicable laws and regulations, including all United States export laws and regulations.

IBM MAKES NO GUARANTEE ABOUT THE CONTENT OF THESE PUBLICATIONS. THE PUBLICATIONS ARE PROVIDED "AS-IS" AND WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, NON-INFRINGEMENT, AND FITNESS FOR A PARTICULAR PURPOSE.

Printed in USA