INTERNATIONAL STANDARD

ISO/IEC 14776-452

First edition
2005-08

Information technology –
Small computer system interface (SCSI) –

Part 452:
Primary Commands-2 (SPC-2)

Reference number
ISO/IEC 14776-452:2005(E)

INTERNATIONAL STANDARD

ISO/IEC 14776-452

First edition
2005-08

Information technology – Small computer system interface (SCSI) –

Part 452: Primary Commands-2 (SPC-2)

PRICE CODE XF

For price, see current catalogue

Contents

Tables

Figures

INFORMATION TECHNOLOGY – SMALL COMPUTER SYSTEM INTERFACE (SCSI) – Part 452: Primary Commands-2 (SPC-2)

FOREWORD

1) ISO (International Organization for Standardization) and IEC (International Electrotechnical Commission) form the specialized system for worldwide standardization. National bodies that are members of ISO or IEC participate in the development of International Standards through technical committees established by the respective organization to deal with particular fields of technical activity. ISO and IEC technical committees collaborate in fields of mutual interest. Other international organizations, governmental and non-governmental, in liaison with ISO and IEC, also take part in the work.

2) In the field of information technology, ISO and IEC have established a joint technical committee, ISO/IEC JTC 1. Draft International Standards adopted by the joint technical committee are circulated to national bodies for voting. Publication as an International Standard requires approval by at least 75 % of the national bodies casting a vote.

3) All users should ensure that they have the latest edition of this publication.

4) No liability shall attach to IEC or ISO or its directors, employees, servants or agents including individual experts and members of their technical committees and IEC or ISO National Committees for any personal injury, property damage or other damage of any nature whatsoever, whether direct or indirect, or for costs (including legal fees) and expenses arising out of the publication of, use of, or reliance upon this ISO/IEC publication or any other IEC or IEC/ISO publications.

5) Attention is drawn to the normative references cited in this publication. Use of the referenced publications is indispensable for the correct application of this publication.

6) Attention is drawn to the possibility that some of the elements of this ISO/IEC Publication may be the subject of patent rights. ISO/IEC shall not be held responsible for identifying any or all such patent rights.

International Standard ISO/IEC 14776-452 was prepared by subcommittee 25: Interconnection of information technology equipment, of ISO/IEC joint technical committee 1: Information technology.

Introduction

The SCSI family of standards provides for many different types of SCSI devices (disks, tapes, printers, scanners and many more). This standard defines a device model that is applicable to all SCSI devices. Other SCSI command standards (see 3.1.12) expand on the general SCSI device model in ways appropriate to specific types of SCSI devices.

The set of SCSI standards specifies the interfaces, functions and operations necessary to ensure interoperability between conforming SCSI implementations. This standard is a functional description. Conforming implementations may employ any design technique that does not violate interoperability.

Figure 1 shows the relationship of this standard to the other standards and related projects in the SCSI family of standards as of the publication of this standard.

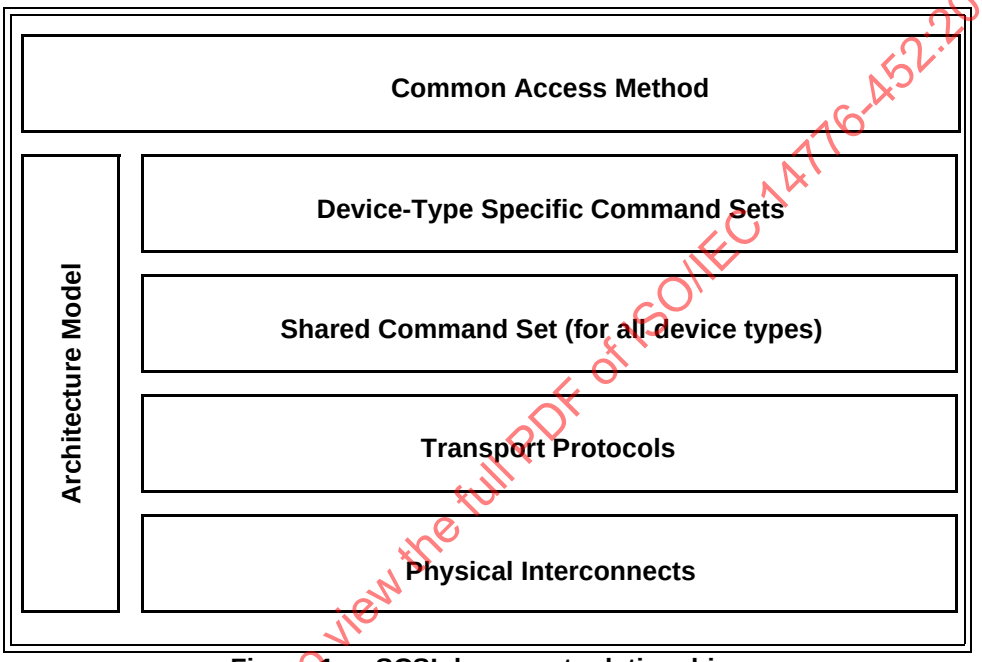

Figure 1 — SCSI document relationships

Figure 1 is intended to show the general relationship of the documents to one another. Figure 1 is not intended to imply a relationship such as a hierarchy, protocol stack or system architecture. It indicates the applicability of a standard to the implementation of a given transport.

For the general structure and references of the SCSI standards, refer to Annex E. The term SCSI is used to refer to the family of standards listed in this annex.

This International Standard is divided into ten clauses.

Clause 1	Scope
Clause 2	Normative references that apply to this standard
Clause 3	Definitions, symbols and abbreviations used in this standard
Clause 4	Conceptual relationship between this document and the SCSI-3 architecture model
Clause 5	Command model for all SCSI devices
Clause 6	Command model for processor type for all SCSI devices
Clause 7	Commands that may be implemented by any SCSI device
Clause 8	Parameter data formats that may be implemented by a processor type SCSI device
Clause 9	Commands that may be implemented by a processor type SCSI device
Clause 10	Parameter data formats that may be implemented by a processor type SCSI device
Annexes	Provide information to assist with the implementation of this standard. The information in the annexes applies to all the SCSI command standards.

INFORMATION TECHNOLOGY – SMALL COMPUTER SYSTEM INTERFACE (SCSI) – Part 452: Primary Commands-2 (SPC-2)

1 Scope

This part of ISO/IEC 14776 defines the SCSI commands that are mandatory and optional for all SCSI devices. It also defines the SCSI commands that may apply to any device model.

Since a host processor is a part of any SCSI domain, the processor device model is defined in this standard. The commands that may be implemented by a SCSI processor device are also defined in this standard. Some target SCSI devices may implement an initiator subset of the processor device model to support the Asynchronous Event Reporting capability defined in the SCSI-3 Architecture Model.

2 Normative references

2.1 General

The following referenced documents are indispensable for the application of this document. For dated references, only the edition cited applies. For undated references, the latest edition of the referenced document (including any amendments) applies.

2.2 Approved references

IEC 60027-2, *Letter symbols to be used in electrical technology - Part 2: Telecommunications and electronics*

ISO/IEC 9316:1995, *Information technology – Small computer system interface-2 (SCSI-2)*[1]

ISO/IEC 14776-412, *Information technology – Small computer system interface (SCSI) – Part 412: SCSI Architecture Model-2 (SAM-2)*

ISO/IEC 14776-222, *Information technology – Small computer system interface (SCSI) – Part 222: SCSI-3 Fibre Channel Protocol-2 for SCSI, Second Versin (FCP-2)*

2.3 References under development

At the time of publication, the following referenced standards were still under development. For information on the current status of the document, or regarding availability, contact the relevant standards body or other organization as indicated.

ISO/IEC 14776-351, *Information technology – Small computer system interface-3 (SCSI-3) – Part 351: SCSI-3 Medium Changer Commands (SMC)*[2]

1 [ANSI INCITS 270:1996]

2 [ANSI INCITS 314:1998]

3 Definitions, symbols, abbreviations and conventions

3.1 Definitions

3.1.1
active condition:
When a logical unit is capable of responding immediately to media access requests, and operations complete in the shortest practical time.

3.1.2
additional sense code:
A combination of the ADDITIONAL SENSE CODE and ADDITIONAL SENSE CODE QUALIFIER fields (see 7.20.2) in the sense data (see 3.1.47).

3.1.3
application client:
An object that is the source of SCSI commands. Further definition of an application client may be found in SAM-2.

3.1.4
attached medium changer:
A medium changer that is attached to and accessed through some other type of SCSI device. See 5.7.

3.1.5
asynchronous event reporting (AER):
A mechanism used by a logical unit to signal an initiator that an asynchronous event has occurred. The mechanism for asynchronous event reporting is protocol specific. A detailed definition of AER may be found in SAM-2.

3.1.6
auto contingent allegiance (ACA):
One of the conditions of a task set following the return of a CHECK CONDITION status. A detailed definition of ACA may be found in SAM-2.

3.1.7
autosense data:
The sense data that is automatically delivered to the application client by the device server in a protocol specific manner when a command completes with a CHECK CONDITION status (see 4.2 and SAM-2).

3.1.8
blocked task:
A blocked task that is in the blocked state as defined in SAM-2. Tasks become blocked when an ACA condition occurs. The blocked state ends when the ACA condition is cleared. A detailed definition of the blocked task state may be found in SAM-2.

3.1.9
byte:
Indicates an 8-bit construct.

3.1.10
command:
A request describing a unit of work to be performed by a device server. A detailed definition of a command may be found in SAM-2.

3.1.11
command descriptor block (CDB):
The structure used to communicate commands from an application client to a device server. A CDB may have a fixed length of up to 16 bytes or a variable length of between 12 and 260 bytes.

3.1.12
command standard:
A SCSI standard that defines the model, commands and parameter data for a device type (e.g., SBC, SCC, SGC, SMC, SSC, MMC, or SES).

3.1.13
contingent allegiance (CA):
One of the conditions of a task set following the return of a CHECK CONDITION status. A detailed definition of CA may be found in SCSI-2.

3.1.14
copy manager:
The device server that receives an EXTENDED COPY command and performs the operation requested.

3.1.15
data-in buffer:
The buffer identified by the application client to receive data from the device server during the execution of a command (see 4.2 and SAM-2).

3.1.16
data-out buffer:
The buffer identified by the application client to supply data that is sent from the application client to the device server during the execution of a command (see 4.2 and SAM-2).

3.1.17
data packet:
The data transferred in the Data-In Buffer associated with a processor device RECEIVE command, or in the Data-Out Buffer associated with a processor device SEND command.

3.1.18
device server:
An object within a logical unit that executes SCSI tasks according to the rules of task management. A detailed definition of a device server may be found in SAM-2.

3.1.19
device service request:
A request, submitted by an application client, conveying a SCSI command to a device server. A detailed definition of a device service request may be found in SAM-2.

3.1.20
device service response:
The response returned to an application client by a device server on completion of a SCSI command. A detailed definition of a device service response may be found in SAM-2.

3.1.21
device type:
The type of device (or device model) implemented by the device server.

3.1.22
element:
An addressable physical component of a medium changer SCSI device that may serve as the location of a removable unit of data storage medium. A detailed definition of an element may be found in SMC.

3.1.23
enabled task state:
The only task state in which a task may make progress towards completion. A detailed definition of the enabled task state may be found in SAM-2.

3.1.24
field:
A group of one or more contiguous bits, a part of a larger structure such as a CDB (see 3.1.11) or sense data (see 3.1.47).

3.1.25
hard reset:
A target response to a reset event or TARGET RESET task management function. A detailed definition of hard reset may be found in SAM-2.

3.1.26
host:
A SCSI device with the characteristics of a primary computing device, typically a personal computer, workstation, minicomputer, mainframe computer, or auxiliary computing device or server. A host typically functions as an initiator.

3.1.27
idle condition:
In idle condition a logical unit is capable of responding quickly to media access requests. However, a logical unit in idle condition may take longer to complete a command than a logical unit in active condition because it may have to activate some circuitry.

3.1.28
initiator:
A SCSI device containing application clients that originate device service requests to be processed in a device server. A detailed definition of an initiator may be found in SAM-2.

3.1.29
linked command:
One in a series of SCSI commands processed by a single task that collectively make up a discrete I/O operation. A detailed definition of a linked command may be found in SAM-2.

3.1.30
logical unit:
An externally addressable entity within a target that implements a SCSI device model and contains a device server. A detailed definition of a logical unit may be found in SAM-2.

3.1.31
logical unit identifier:
An object that is part of the SAM-2 definition of a logical unit. A logical unit identifier uniquely identifies a logical unit in a SCSI domain. Detailed definitions of SCSI domain and logical unit identifier may be found in SAM-2.

3.1.32
logical unit inventory:
The list of the logical unit numbers reported by a REPORT LUNS command (see 7.19). The list includes the logical unit numbers of all logical units with a PERIPHERAL QUALIFIER value of 000b (see 7.3.2) and may include in a vendor specific manner the logical unit numbers for those logical units with a PERIPHERAL QUALIFIER value of 100b, 101b, 110b, or 111b.

3.1.33
logical unit number (LUN):
An encoded 64-bit identifier for a logical unit. A detailed definition of a logical unit number may be found in SAM-2.

3.1.34
medium:
A physical entity that stores data in a nonvolatile manner (retained through a power cycle) in accordance with commands processed by the device server.

3.1.35
medium auxiliary memory (MAM):
An auxiliary memory residing on a medium that is accessible to the device server (e.g., a tape cartridge). Medium auxiliary memory may be nonvolatile and independent of the main function of the device server.

3.1.36
medium changer:
A device that mechanizes the movement of media to and from the SCSI device that records on or reads from the media. A detailed definition of a medium changer may be found in SMC.

3.1.37
one:
The logical true condition of a variable.

3.1.38
page:
A regular parameter structure (or format) used by several commands. These pages are identified with a value known as a page code.

3.1.39
persist through power loss:
An optional capability associated with some features that allows an application client to request that a device server maintain information regarding that feature across power failures.

3.1.40
protocol specific:
A requirement that is defined by a SCSI transport protocol standard. A detailed definition of protocol specific may be found in SAM-2.

3.1.41
protocol standard:
A SCSI standard that defines SCSI transport protocol (e.g., SPI-3, SBP-2, or FCP-2).

3.1.42
resource:
A part of a processor device required to operate on or store a data packet.

3.1.43
registered:
The condition that exists for an initiator (or application client) from successful completion of a PERSISTENT RESERVE OUT command with a REGISTER or REGISTER AND IGNORE EXISTING KEY service action (see 5.5.3.4) until the initiator registration is removed (see 5.5.3.6).

3.1.44
registrant:
An initiator (or application client) that is registered (see 3.1.43).

3.1.45
SCSI device:
A device that is connected to a service delivery subsystem and supports a SCSI application protocol. A detailed definition of a SCSI device may be found in SAM-2.

3.1.46
SCSI domain:
The interconnection of two or more SCSI devices and a service delivery subsystem. A detailed definition of a SCSI Domain may be found in SAM-2.

3.1.47
sense data:
Data describing an error or exceptional condition that a device server delivers to an application client as a result of an autosense operation (see 3.1.7), asynchronous event report (see 3.1.5), or REQUEST SENSE command (see 7.20). The format of sense data is the format defined for parameter data returned by the REQUEST SENSE command in 7.20.2.

3.1.48
sense key:
The contents of the SENSE KEY field (see 7.20.2) in the sense data (see 3.1.47).

3.1.49
service action:
A request describing a unit of work to be performed by a device server. A service action is an extension of a command. See SAM-2 for a detailed definition of a command.

3.1.50
service delivery subsystem:
That part of a SCSI I/O system that transmits service requests to a logical unit and returns logical unit responses to an initiator. A detailed definition of a service delivery subsystem may be found in SAM-2.

3.1.51
standby condition:
When a logical unit is capable of accepting commands, but media is not immediately accessible (e.g., spindle is stopped).

3.1.52
status:
One byte of response information sent from a device server to an application client upon completion of each command. A detailed definition of status may be found in SAM-2.

3.1.53
system:
One or more SCSI domains operating as a single configuration.

3.1.54
target:
A SCSI device that receives SCSI commands and directs such commands to one or more logical units. A detailed definition of a target may be found in SAM-2.

3.1.55
task:
An object within a logical unit that represents the work associated with a command or a group of linked commands. A detailed definition of a task may be found in SAM-2.

3.1.56
task set:
A group of tasks within a logical unit, whose interaction is dependent on the task management (queuing), CA, and ACA rules. See SAM-2 and the Control mode page (see 8.3.6).

3.1.57
third-party:
An EXTENDED COPY command issued to one SCSI device to perform a copy operation between two other SCSI devices; or a RESERVE or RELEASE command issued by one initiator to manage a reservation on behalf of another initiator within the same SCSI domain and using the same SCSI protocol (e.g., a processor device requests that a direct-access device reserve itself for use by a sequential-access device).

3.1.58
unit attention condition:
A state that a logical unit maintains while it has asynchronous status information to report to one or more initiators. A detailed definition of the unit attention condition may be found in SAM-2.

3.1.59
vendor specific (VS):
Something (e.g., a bit, field, code value, etc.) that is not defined by this standard and may be vendor defined.

3.1.60
zero:
The logical false condition of a variable.

3.2 Acronyms

ACA	Auto Contingent Allegiance (see 3.1.6)
AER	Asynchronous Event Reporting (see 3.1.5)
ASC	Additional Sense Code (see 7.20.2)
ASCQ	Additional Sense Code Qualifier (see 7.20.2)
CA	Contingent Allegiance (see 3.1.13 and SCSI-2)
CDB	Command Descriptor Block (see 3.1.11)
CRC	Cyclic Redundancy Check
D_ID	Destination Identifier (defined in FC-FS)
FC-FS	Fibre Channel Framing and Signaling Interface
FCP-2	SCSI-3 Fibre Channel Protocol - 2
LBA	Logical Block Address
LSB	Least significant bit
LUN	Logical Unit Number (see 3.1.33)
MAM	Medium Auxiliary Memory (see 3.1.35)
MMC-2	SCSI Multi-Media Commands -2
MSB	Most significant bit
NCITS	National Committee for Information Technology Standards
OCRW	SCSI Specification for Optical Card Reader/Writer
OSD	Object-based Storage Devices Commands
RAID	Redundant Array of Independent Disks
RBC	SCSI Reduced Block Commands
RMC	SCSI Reduced Multi-Media Commands
SAM	SCSI-3 Architecture Model
SAM-2	SCSI Architecture Model -2
SBC	SCSI-3 Block Commands
SCC-2	SCSI Controller Commands -2
SCSI	The architecture defined by the family of standards described in the introduction
SCSI-2	The architecture defined by the Small Computer System Interface - 2 standard (see 2.2)
SES	SCSI-3 Enclosure Services
SMC	SCSI-3 Medium Changer Commands
SPC	SCSI-3 Primary Commands (ISO/IEC 14776-311)
SPC-2	SCSI Primary Commands -2 (this standard)
SPI-3	SCSI Parallel Interface -3
SPI-4	SCSI Parallel Interface -4
SSC	SCSI-3 Stream Commands
VPD	Vital Product Data (see 8.4)
VS	Vendor Specific (see 3.1.59)

3.3 Keywords

3.3.1
expected:
A keyword used to describe the behavior of the hardware or software in the design models assumed by this standard. Other hardware and software design models may also be implemented.

3.3.2
ignored:
A keyword used to describe an unused bit, byte, word, field or code value. The contents or value of an ignored bit, byte, word, field or code value shall not be examined by the receiving SCSI device and may be set to any value by the transmitting SCSI device.

3.3.3
invalid:
A keyword used to describe an illegal or unsupported bit, byte, word, field or code value. Receipt of an invalid bit, byte, word, field or code value shall be reported as an error.

3.3.4
mandatory:
A keyword indicating an item that is required to be implemented as defined in this standard.

3.3.5
may:
A keyword that indicates flexibility of choice with no implied preference (equivalent to "may or may not").

3.3.6
may not:
A keyword that indicates flexibility of choice with no implied preference (equivalent to "may or may not").

3.3.7
obsolete:
A keyword indicating that an item was defined in prior SCSI standards but has been removed from this standard.

3.3.8
optional:
A keyword that describes features that are not required to be implemented by this standard. However, if any optional feature defined by this standard is implemented, then it shall be implemented as defined in this standard.

3.3.9
reserved:
A keyword referring to bits, bytes, words, fields and code values that are set aside for future standardization. A reserved bit, byte, word or field shall be set to zero, or in accordance with a future extension to this standard. Recipients are not required to check reserved bits, bytes, words or fields for zero values. Receipt of reserved code values in defined fields shall be reported as error.

3.3.10
restricted:
A keyword referring to bits, bytes, words, and fields that are set aside for use in other SCSI standards. A restricted bit, byte, word, or field shall be treated as a reserved bit, byte, word or field for the purposes of the requirements defined in this standard.

3.3.11
shall:
A keyword indicating a mandatory requirement. Designers are required to implement all such mandatory requirements to ensure interoperability with other products that conform to this standard.

3.3.12
should:
A keyword indicating flexibility of choice with a strongly preferred alternative; equivalent to the phrase "it is strongly recommended".

3.3.13
x or **xx:**
The value of the bit or field is not relevant.

3.4 Conventions

Certain words and terms used in this standard have a specific meaning beyond the normal English meaning. These words and terms are defined either in 3.1 or in the text where they first appear. Names of commands, statuses, sense keys, and additional sense codes are in all uppercase (e.g., REQUEST SENSE). Lowercase is used for words having the normal English meaning.

If there is more than one CDB length for a particular command (e.g., MODE SENSE(6) and MODE SENSE(10)) and the name of the command is used in a sentence without any CDB length descriptor (e.g., MODE SENSE), then the condition specified in the sentence applies to all CDB lengths for that command.

The names of fields are in small uppercase (e.g., ALLOCATION LENGTH). When a field name is a concatenation of acronyms, uppercase letter may be used for readability (e.g., NORMACA). Normal case is used when the contents of a field are being discussed. Fields containing only one bit are usually referred to as the name bit instead of the name field.

Numbers that are not immediately followed by lower-case b or h are decimal values.

Numbers immediately followed by lower-case b (xxb) are binary values.

Numbers or upper case letters immediately followed by lower-case h (xxh) are hexadecimal values.

When the value of the bit or field is not relevant, x or xx appears in place of a specific value.

Lists sequenced by letters (e.g., a-red, b-blue, c-green) show no priority relationship between the listed items. Numbered lists (e.g., 1-red, 2-blue, 3-green) show a priority ordering between the listed items.

If a conflict arises between text, tables or figures, the order of precedence to resolve the conflicts is text, then tables and finally figures. Not all tables or figures are fully described in the text. Tables show data format and values. Notes do not constitute any requirements for implementors.

3.5 Notation for procedures and functions

In this standard, the model for functional interfaces between objects is the callable procedure. Such interfaces are specified using the following notation:

[Result =] Procedure Name (IN ([input-1] [,input-2] …), OUT ([output-1] [,output-2] …))

Where:

Result:	A single value representing the outcome of the procedure or function.
Procedure Name:	A descriptive name for the function to be performed.
Input-1, Input-2, …:	A comma-separated list of names identifying caller-supplied input data objects.
Output-1, Output-2, …:	A comma-separated list of names identifying output data objects to be returned by the procedure.
"[…]":	Brackets enclosing optional or conditional parameters and arguments.

This notation allows data objects to be specified as inputs and outputs. The following is an example of a procedure specification:

Found = Search (IN (Pattern, Item List), OUT ([Item Found]))

Where:

Found = Flag
Flag, which, if set, indicates that a matching item was located.

Input Arguments:

Pattern = … /* Definition of Pattern object */
Object containing the search pattern.

Item List = Item<NN> /* Definition of Item List as an array of NN Item objects*/
Contains the items to be searched for a match.

Output Arguments:

Item Found = Item … /* Item located by the search procedure */
This object is only returned if the search succeeds.

4 General concepts

4.1 Introduction

This standard defines behaviors that are common to all SCSI device models (see clause 5). This standard defines the SCSI commands that are basic to more than one device model and the SCSI commands that may apply to any device model (see clause 7). This standard defines the parameters that are basic to more than one device model (see clause 8).

The processor device model (see clause 6), commands (see clause 9), and parameters (see clause 10) are defined in this standard.

4.2 The request-response model

The SCSI command set assumes an underlying request-response protocol. The fundamental properties of the request-response protocol are defined in SAM-2. Action on SCSI commands shall not be deemed completed until a response is received. The response shall include a status that indicates the final disposition of the command. As per SAM-2, the request-response protocol may be modeled as a procedure call, specifically:

Service response = Execute Command (IN(I_T_L_x Nexus, CDB, [Data-Out Buffer], Task Attributes),
OUT([Data-In Buffer], [Autosense Data], [Autosense Return Flag], Status))

SAM-2 defines all of the inputs and outputs in the procedure call above. As they may apply to any SCSI device, this standard defines the contents of the following procedure inputs and outputs; CDB, Data-Out Buffer, Data-In Buffer, and Autosense Data. This standard does not define all possible instances of these procedure inputs and outputs. This standard defines only those instances that may apply to any SCSI device or to processor type SCSI devices. Instances of the procedure inputs and outputs that apply to specific SCSI device models are defined in the applicable SCSI command standards (see 3.1.12).

This standard references values returned via the Status output parameter. Examples of such status values are CHECK CONDITION and RESERVATION CONFLICT. Status values are not defined by this standard. SAM-2 defines all Status values.

The entity that makes the procedure call from an initiator is an application client, as defined in SAM-2. The procedure call's representation arrives at the target in the form of a device service request. The entity that performs the work of the procedure call in a target is a device server, which is an object within a logical unit and is defined in SAM-2.

4.3 The Command Descriptor Block (CDB)

4.3.1 CDB usage and structure

A command is communicated by sending a command descriptor block (CDB) to the device server. For several commands, the CDB is accompanied by a list of parameters in the Data-Out Buffer. See the specific commands for detailed information.

The fixed length CDB formats are described in 4.3.2. The variable length CDB formats are described in 4.3.3. The CDB fields that are common to most commands are described in 4.3.4. The fields shown in 4.3.2 and 4.3.3 and described in 4.3.4 are used consistently by most commands. However, the actual usage of any field (except OPERATION CODE and CONTROL) is described in the subclause defining that command. If a device server receives a CDB containing an operation code that is invalid or not supported, it shall return CHECK CONDITION status with the sense key set to ILLEGAL REQUEST and an additional sense code of INVALID COMMAND OPERATION CODE.

For all commands, if there is an invalid parameter in the CDB, then the device server shall terminate the command without altering the medium.

4.3.2 The fixed length CDB formats

All fixed length CDBs shall have an OPERATION CODE field as their first byte and a CONTROL byte as their last byte. Table 1 shows the typical format of a 6-byte CDB. Table 2 shows the typical format of a 10-byte CDB. Table 3 shows the typical format of a 12-byte CDB. Table 4 shows the typical format of a 16-byte CDB. Table 5 shows the format of a 16-byte CDB for commands that provide for a long LBA.

Table 1 — Typical CDB for 6-byte commands

Bit Byte	7	6	5	4	3	2	1	0
0	OPERATION CODE							
1	Reserved			(MSB)				
2	LOGICAL BLOCK ADDRESS (if required)							
3								(LSB)
4	TRANSFER LENGTH (if required) PARAMETER LIST LENGTH (if required) ALLOCATION LENGTH (if required)							
5	CONTROL							

Table 2 — Typical CDB for 10-byte commands

Bit Byte	7	6	5	4	3	2	1	0
0	OPERATION CODE							
1	Reserved			SERVICE ACTION (if required)				
2	(MSB)	LOGICAL BLOCK ADDRESS (if required)						
3								
4								
5								(LSB)
6	Reserved							
7	(MSB)	TRANSFER LENGTH (if required)						
8	PARAMETER LIST LENGTH (if required) ALLOCATION LENGTH (if required)							(LSB)
9	CONTROL							

Table 3 — Typical CDB for 12-byte commands

Bit Byte	7	6	5	4	3	2	1	0
0	OPERATION CODE							
1	Reserved			SERVICE ACTION (if required)				
2	(MSB)							
3								
4				LOGICAL BLOCK ADDRESS (if required)				
5								(LSB)
6	(MSB)							
7				TRANSFER LENGTH (if required) PARAMETER LIST LENGTH (if required) ALLOCATION LENGTH (if required)				
8								
9								(LSB)
10	Reserved							
11	CONTROL							

Table 4 — Typical CDB for 16-byte commands

Bit Byte	7	6	5	4	3	2	1	0
0	OPERATION CODE							
1	Reserved			SERVICE ACTION (if required)				
2	(MSB)							
3								
4				LOGICAL BLOCK ADDRESS (if required)				
5								(LSB)
6	(MSB)							
7								
8				Additional CDB data (if required)				
9								(LSB)
10	(MSB)							
11				TRANSFER LENGTH (if required) PARAMETER LIST LENGTH (if required) ALLOCATION LENGTH (if required)				
12								
13								(LSB)
14	Reserved							
15	CONTROL							

Table 5 — Typical CDB for long LBA 16-byte commands

Bit Byte	7	6	5	4	3	2	1	0
0	OPERATION CODE							
1	Reserved			miscellaneous CDB information				
2	(MSB)							
3								
4								
5								
6				LOGICAL BLOCK ADDRESS				
7								
8								
9								(LSB)
10	(MSB)							
11				TRANSFER LENGTH (if required)				
12				PARAMETER LIST LENGTH (if required) ALLOCATION LENGTH (if required)				
13								(LSB)
14	Reserved							
15	CONTROL							

4.3.3 The variable length CDB formats

Operation code 7Fh heads a variable length CDB. The CONTROL byte is the second byte in the variable length CDB (see table 6).

Table 6 — Typical variable length CDB

Bit Byte	7	6	5	4	3	2	1	0
0	OPERATION CODE (7Fh)							
1	CONTROL							
2	Reserved							
3	Reserved							
4	Reserved							
5	Reserved							
6	Reserved							
7	ADDITIONAL CDB LENGTH (n-7)							
8	(MSB)							
9				SERVICE ACTION				(LSB)
10								
n	Service action specific fields							

The ADDITIONAL CDB LENGTH field indicates the number of additional CDB bytes. This value in the ADDITIONAL CDB LENGTH field shall be a multiple of 4. If the number of CDB bytes delivered by the service delivery subsystem is not sufficient to contain the number of bytes specified by the ADDITIONAL CDB LENGTH field, the command shall be termi-

nated with a CHECK CONDITION status. The sense key shall be set to ILLEGAL REQUEST and the additional sense code shall be set to INVALID FIELD IN CDB.

The SERVICE ACTION field indicates the action being requested by the application client. The SERVICE ACTION field is required in the variable length CDB format and is described in 4.3.4.2. Each service action code description defines a number of service action specific fields that are needed for that service action.

A 32-byte variable length CDB format is defined for long LBA operations (see table 7).

Table 7 — Typical variable length CDB for long LBA 32-byte commands

Bit / Byte	7	6	5	4	3	2	1	0
0	OPERATION CODE (7Fh)							
1	CONTROL							
2	Reserved							
4								
5	Reserved							
6	Reserved							
7	ADDITIONAL CDB LENGTH (18h)							
8	(MSB)							
9	SERVICE ACTION							(LSB)
10	Reserved			DPO	FUA	Reserved		
11	Reserved							
12	(MSB)							
19	LOGICAL BLOCK ADDRESS							(LSB)
20	Additional CDB data							
27								
28	(MSB)							
31	TRANSFER LENGTH (if required) PARAMETER LIST LENGTH (if required) ALLOCATION LENGTH (if required)							(LSB)

4.3.4 Common CDB fields

4.3.4.1 Operation code

The OPERATION CODE field contains the code value identifying the operation being requested by the CDB. SAM-2 defines the general structure of the operation code value. The OPERATION CODE field has a consistently defined meaning across all commands. This standard specifies the operation code values used by the commands defined herein.

4.3.4.2 Service action

All CDB formats except the 6-byte format provide for a SERVICE ACTION field containing a coded value identifying a function to be performed under the more general command function specified in the OPERATION CODE field. While the SERVICE ACTION field is defined for CDB formats, it is used as described in this subclause only in those CDB formats that contain a SERVICE ACTION field. When the specific field SERVICE ACTION is not defined in a CDB format, the bits identified as the SERVICE ACTION field in a CDB shall be used or reserved as specified by the particular CDB format.

4.3.4.3 Logical block address

The logical block addresses on a logical unit or within a volume partition shall begin with block zero and be contiguous up to the last logical block of that logical unit or within that partition.

A six-byte CDB contains a 21-bit LOGICAL BLOCK ADDRESS field. The ten-byte and the twelve-byte CDBs contain 32-bit LOGICAL BLOCK ADDRESS fields. The sixteen-byte CDB has two formats one with a 32-bit LOGICAL BLOCK ADDRESS field (see table 4) and one with a 64-bit LOGICAL BLOCK ADDRESS field (see table 5). LOGICAL BLOCK ADDRESS fields in additional parameter data have their length specified for each occurrence. See the specific command descriptions.

4.3.4.4 Transfer length

The TRANSFER LENGTH field specifies the amount of data to be transferred, usually the number of blocks. Some commands use transfer length to specify the requested number of bytes to be sent as defined in the command description. See the following descriptions and the individual command descriptions for further information.

Commands that use one byte for the TRANSFER LENGTH field allow up to 256 blocks of data to be transferred by one command. A TRANSFER LENGTH value of 1 to 255 indicates the number of blocks that shall be transferred. A value of zero specifies that 256 blocks shall be transferred.

In commands that use multiple bytes for the TRANSFER LENGTH field, a transfer length of zero indicates that no data transfer shall take place. A value of one or greater indicates the number of blocks that shall be transferred.

Refer to the specific command description for further information.

4.3.4.5 Parameter list length

The PARAMETER LIST LENGTH field is used to specify the number of bytes sent from the Data-Out Buffer. This field is typically used in CDBs for parameters that are sent to a device server (e.g., mode parameters, diagnostic parameters, log parameters). A parameter length of zero indicates that no data shall be transferred. This condition shall not be considered as an error.

4.3.4.6 Allocation length

The ALLOCATION LENGTH field specifies the maximum number of bytes that an application client has allocated for returned data. An allocation length of zero indicates that no data shall be transferred. This condition shall not be considered as an error. The device server shall terminate transfers to the Data-In Buffer when allocation length bytes have been transferred or when all available data have been transferred, whichever is less. The allocation length is used to limit the maximum amount of data (e.g., sense data, mode data, log data, diagnostic data) returned to an application client. If the information being transferred to the Data-In Buffer includes fields containing counts of the number of bytes in some or all of the data, the contents of these fields shall not be altered to reflect the truncation, if any, that results from an insufficient ALLOCATION LENGTH value, unless the standard that describes the Data-In Buffer format specifically states otherwise.

If the amount of information to be transferred exceeds the maximum value that may be specified in the ALLOCATION LENGTH field the device server shall transfer no data and return a CHECK CONDITION status; the sense key shall be set to ILLEGAL REQUEST and the additional sense code shall be set to INVALID FIELD IN CDB.

4.3.4.7 Control

The contents of the CONTROL field are defined in SAM-2. The CONTROL field has a consistently defined meaning across all commands.

5 Model common to all device types

5.1 Introduction to the model common to all device types

This model describes some of the general characteristics expected of most SCSI devices. It is not intended to alter any requirements defined elsewhere in SCSI. Devices conforming to this standard also shall conform to SAM-2.

5.2 Commands implemented by all SCSI device servers

5.2.1 Summary of commands implemented by all SCSI device servers

This standard defines three commands that all SCSI device servers shall implement - INQUIRY, REQUEST SENSE, and TEST UNIT READY. These commands are used to configure the system, to test devices, and to return important information concerning errors and exception conditions.

5.2.2 Using the INQUIRY command

The INQUIRY command may be used by an application client to determine the configuration of the logical unit. Device servers respond with information that includes their type and standard version and may include the vendor's identification, model number and other information. It is recommended that device servers be capable of returning this information (or whatever part of it that is available) upon completing power-on initialization. A device server may take longer to get certain portions of this information, especially if it retrieves the information from the medium.

5.2.3 Using the REQUEST SENSE command

Whenever a command completes with a CHECK CONDITION status and autosense data is not provided, the application client that received the error status should issue a REQUEST SENSE command to receive the sense data describing the cause of the condition. If the application client issues a command other than REQUEST SENSE, the sense data is lost.

5.2.4 Using the TEST UNIT READY command

The TEST UNIT READY command allows an application client to poll a logical unit until it is ready without the need to allocate space for returned data. The TEST UNIT READY command may be used to check the media status of logical units with removable media. Device servers should respond promptly to indicate the current status of the SCSI device, delays to achieve GOOD status may adversely affect initiator performance.

5.3 Parameter rounding

Certain parameters sent to a device server with various commands contain a range of values. Device servers may choose to implement only selected values from this range. When the device server receives a value that it does not support, it either rejects the command (CHECK CONDITION status with ILLEGAL REQUEST sense key) or it rounds the value received to a supported value.

When parameter rounding is implemented, a device server that receives a parameter value that is not an exact supported value shall adjust the value to one that it supports and shall return CHECK CONDITION status with a sense key of RECOVERED ERROR. The additional sense code shall be set to ROUNDED PARAMETER. The application client should issue an appropriate command to learn what value the device server has selected.

The device server shall reject unsupported values unless rounding is permitted in the description of the parameter. When the description of a parameter states that rounding is permitted, the device server should adjust maximum-value fields down to the next lower supported value than the one specified by the application client. Minimum-value fields should be rounded up to the next higher supported value than the one specified by the application client. In some cases, the type of rounding (up or down) is explicitly specified in the description of the parameter.

5.4 Self-test Operations

5.4.1 Default self-test

The SEND DIAGNOSTIC command provides a means to request that a SCSI device perform a self test. While the test is vendor specific, the means of requesting the test is standardized.

The default self-test is mandatory for all device types that support the SEND DIAGNOSTICS command. The response is GOOD status if the test detects no exceptions or a CHECK CONDITION status if the test detects exceptions.

5.4.2 The short and extended self-tests

There are two optional types of self-test aside from the mandatory default self-test that may be invoked using the SELF-TEST CODE field in the SEND DIAGNOSTICS command: a short self-test and an extended self-test. The goal of the short self-test is to quickly identify if the logical unit determines that it is faulty. A goal of the extended self-test routine is to simplify factory testing during integration by having logical units perform more comprehensive testing without application client intervention. A second goal of the extended self-test is to provide a more comprehensive test to validate the results of a short self-test, if its results are judged by the application client to be inconclusive.

The criteria for the short self-test are that it has one or more segments and completes in two minutes or less. The criteria for the extended self-test are that it is has one or more segments and that the completion time is vendor specific. Any tests performed in the segments are vendor specific.

The following are examples of segments:

a) An electrical segment wherein the logical unit tests its own electronics. The tests in this segment are vendor specific, but some examples of tests that may be included are: a buffer RAM test, a read/write circuitry test, and/or a test of the read/write head elements;
b) A seek/servo segment wherein a device tests its capability to find and servo on data tracks; and
c) A read/verify scan segment wherein a device performs read scanning of some or all of the medium surface.

The tests performed in the segments may be the same for the short and extended self-tests. The time required by a logical unit to complete its extended self-test is reported in the EXTENDED SELF-TEST COMPLETION TIME field in the Control mode page (see 8.3.6).

5.4.3 Self-test modes

There are two modes for short and extended self-tests: a foreground mode and a background mode. These modes are described in 5.4.3.1 and 5.4.3.2.

5.4.3.1 Foreground mode

When a device server receives a SEND DIAGNOSTICS command specifying a self-test to be performed in the foreground mode, the device server shall return status for that command after the self-test has been completed.

While performing a self-test in the foreground mode, the device server shall respond to all commands except INQUIRY, REPORT LUNS, and REQUEST SENSE with a CHECK CONDITION status, a sense key of NOT READY and an additional sense code of LOGICAL UNIT NOT READY, SELF-TEST IN PROGRESS.

If a device server is performing a self-test in the foreground mode and a test segment error occurs during the test, the device server shall update the self-test results log page (see 8.2.8) and report CHECK CONDITION status with a sense key of HARDWARE ERROR and an additional sense code of LOGICAL UNIT FAILED SELF-TEST. The application client may obtain additional information about the failure by reading the self-test results log page. If the device server is unable to update the self-test results log page it shall return a CHECK CONDITION status with a sense key of HARDWARE ERROR and an additional sense code of LOGICAL UNIT UNABLE TO UPDATE SELF-TEST LOG.

An application client should reserve the logical unit before initiating a self-test in the foreground mode. An application client may terminate a self-test that is being performed in the foreground mode using an ABORT TASK, ABORT TASK SET, or CLEAR TASK SET task management function. If a task manager receives an ABORT TASK, ABORT TASK SET, or CLEAR TASK SET task management function while performing a self-test in the foreground mode, then it shall abort the self-test and update the self-test results log page (see 8.2.8).

5.4.3.2 Background mode

When a device server receives a SEND DIAGNOSTICS command specifying a self-test to be performed in the background mode, the device server shall return status for that command as soon as the CDB has been validated.

After returning status for the SEND DIAGNOSTICS command specifying a self-test to be performed in the background mode, the device server shall initialize the self-test results log page (see 8.2.8) as follows. The self-test code from the SEND DIAGNOSTICS command shall be placed in the SELF-TEST CODE field in the log page. The SELF-TEST RESULTS field shall be set to Fh. After the self-test results log page is initialized, the device server shall begin the first self-test segment.

While the device server is performing a self-test in the background mode, it shall terminate with a CHECK CONDITION status any SEND DIAGNOSTICS command it receives that meets one of the following criteria:

a) The SELFTEST bit is one; or
b) The SELF-TEST CODE field contains a value other than 000b or 100b.

When terminating the SEND DIAGNOSTICS command, the sense key shall be set to NOT READY and the additional sense code shall be set to LOGICAL UNIT NOT READY, SELF-TEST IN PROGRESS.

While performing a self-test in the background mode, the device server shall suspend the self-test to service any other commands received with the exceptions listed in table 8. Suspension of the self-test to service the command shall occur as soon as practical and shall not take longer than two seconds.

Table 8 — Exception commands for background self-tests

Device type	Command		Reference
All device types	SEND DIAGNOSTICS (with SELF-TEST CODE field set to 100b) WRITE BUFFER (with the mode set to any download microcode option)		7.23 7.26
Direct access	FORMAT UNIT	START/STOP UNIT	SBC
Sequential access	ERASE FORMAT MEDIUM LOAD UNLOAD LOCATE READ READ POSITION READ REVERSE	REWIND SPACE VERIFY WRITE WRITE BUFFER WRITE FILEMARKS	SSC
Medium changer	EXCHANGE MEDIUM INITIALIZE ELEMENT STATUS MOVE MEDIUM POSITION TO ELEMENT READ ELEMENT STATUS (if CURDATA=0 and device motion is required) WRITE BUFFER		SMC
NOTE 1 Device types not listed in this table do not have commands that are exceptions for background self-tests, other than those listed above for all device types.			

If one of the exception commands listed in table 8 is received, the device server shall abort the self-test, update the self-test log, and service the command as soon as practical but not longer than two seconds after the CDB has been validated.

An application client may terminate a self-test that is being performed in the background mode by issuing a SEND DIAGNOSTICS command with the SELF-TEST CODE field set to 100b (Abort background self-test function).

5.4.3.3 Elements common to foreground and background self-test modes

The PROGRESS INDICATION field returned in response to a REQUEST SENSE command (see 7.20) may be used by the application client at any time during a self-test operation to poll the logical unit's progress. While a self-test operation is in progress unless an error has occurred, a device server shall respond to a REQUEST SENSE command by returning a sense key of NOT READY and an additional sense code of LOGICAL UNIT NOT READY, SELF-TEST IN PROGRESS with the sense key specific bytes set for progress indication.

The application client may obtain information about the twenty most recently completed self-tests by reading the self-test results log page (see 8.2.8). This is the only method for an application client to obtain information about self-tests performed in the background mode.

Table 9 summarizes when a logical unit returns status after receipt of a self-test command, how an application client may abort a self-test, how a logical unit handles new commands that are received while a self-test is in progress, and how a logical unit reports a self-test failure.

Table 9 — Self-test mode summary

Mode	When Status is Returned	How to abort the self-test	Processing of subsequent commands while self-test is executing	Self-test failure reporting
Fore-ground	After the self-test is complete	ABORT TASK task manage-ment function	If the command is INQUIRY, REPORT LUNS or REQUEST SENSE, process normally. Otherwise, termi-nate with CHECK CONDI-TION status, NOT READY sense key, and LOGICAL UNIT NOT READY, SELF-TEST IN PROGRESS additional sense code.	Terminate with CHECK CONDI-TION status, HARDWARE ERROR sense key, and LOGICAL UNIT FAILED SELF-TEST or LOGICAL UNIT UNABLE TO UPDATE SELF-TEST LOG addi-tional sense code.
Back-ground	After the CDB is validated	SEND DIAG-NOSTICS command with SELF-TEST CODE field set to 100b	Process the command, except as described in 5.4.3.2.	Application client checks Self-test results log page (see 8.2.8) after the PROGRESS INDICATION field returned from REQUEST SENSE indicates the self-test is complete

5.5 Reservations

5.5.1 Reservations overview

Reservations may be used to allow a device server to execute commands from a selected set of initiators. The device server shall reject commands from initiators outside the selected set of initiators by uniquely identifying initiators using protocol specific mechanisms.

Application clients may add or remove initiators from the selected set using reservation commands. If the application clients do not cooperate in the reservation protocol, data may be unexpectedly modified and deadlock conditions may occur.

The general description of reservations involves two groups of considerations;

a) the type of reservation established, and
b) the method used to establish, rescind, and manage the reservation.

There are limits on the combinations of reservation types available under some reservation management methods. See the reservations management commands descriptions for details.

The scope of a reservation shall be one of the following:

a) **logical unit reservations** - a logical unit reservation restricts access to the entire logical unit; and
b) **element reservations** - an element reservation restricts access to a specified element within a medium changer.

Reservations may be further qualified by restrictions on types of access (e.g., read, write, control). However, any restrictions based on the type of reservation are independent of the scope of the reservation. In addition, some methods of reservation management permit establishing reservations on behalf of another device in the same SCSI domain using the same SCSI protocol (third-party reservations).

The methods of managing reservations are identified by the commands associated with the methods. The methods of managing reservations are:

a) Reserve/Release (see 5.5.2) - associated with the RESERVE(10), RELEASE(10), RESERVE(6), and RELEASE(6) commands (see 7.21, 7.16, 7.22, and 7.17, respectively); and
b) Persistent Reservations (see 5.5.3) - associated with the PERSISTENT RESERVE OUT and PERSISTENT RESERVE IN commands (see 7.11 and 7.10, respectively).

The two methods are prevented from creating conflicting and undefined interactions using RESERVATION CONFLICT status in the following manner. If a logical unit has executed a PERSISTENT RESERVE OUT command with the REGISTER or the REGISTER AND IGNORE EXISTING KEY service action and is still registered by any initiator, all RESERVE commands and all RELEASE commands regardless of initiator shall conflict and shall terminate with a RESERVATION CONFLICT status. If a logical unit has been reserved by any RESERVE command and is still reserved by any initiator, all PERSISTENT RESERVE IN and all PERSISTENT RESERVE OUT commands shall conflict regardless of initiator or service action and shall terminate with a RESERVATION CONFLICT status.

Reservation restrictions are placed on commands as a result of access qualifiers associated with the type of reservation. The details of which commands are allowed under what types of reservations are described in table 10. For the reservation restrictions placed on commands for the reserve/release management method see table 10 column [A]. For the reservation restrictions placed on commands for the persistent reservations management method see the table 10 columns under [B].

If any element is reserved within a logical unit, that logical unit shall be considered reserved for the commands listed in table 10 and the accept/conflict information in table 10 shall apply.

In table 10 and table 11 the following key words are used:

allowed: Commands issued by initiators not holding the reservation or by initiators not registered when a registrants only persistent reservation is present should complete normally.

conflict: Commands issued by initiators not holding the reservation or by initiators not registered when a registrants only persistent reservation is present shall not be performed and the device server shall terminate the command with a RESERVATION CONFLICT status.

Commands from initiators holding a reservation should complete normally. The behavior of commands from registered initiators when a registrants only persistent reservation is present is specified in table 10 and table 11.

A command that does not explicitly write the medium shall be checked for reservation conflicts before the command enters the current task state for the first time. Once the command has entered the current task state, it shall not be terminated with a RESERVATION CONFLICT due to a subsequent reservation.

A command that explicitly writes the medium shall be checked for reservation conflicts before the device server modifies the medium or cache as a result of the command. Once the command has modified the medium, it shall not be terminated with a RESERVATION CONFLICT due to a subsequent reservation.

For each command, this standard or a related command standard (see 3.1.12) defines the conditions that result in RESERVATION CONFLICT. Command standards define the conditions either in the device model (preferred) or in the descriptions of each specific command. Annex B contains the RESERVATION CONFLICT information for some of the command sets.

Table 10 — SPC commands that are allowed in the presence of various reservations

Command	Addressed LU is reserved by another initiator [A]	Addressed LU has this type of persistent reservation held by another initiator [B]				
		From any initiator		From registered initiator (RO all types)	From initiator not registered	
		Write Excl	Excl Access		Write Excl RO	Excl Access – RO
EXTENDED COPY	Conflict	Conflict	Conflict	Allowed	Conflict	Conflict
INQUIRY	Allowed	Allowed	Allowed	Allowed	Allowed	Allowed
LOG SELECT	Conflict	Conflict	Conflict	Allowed	Conflict	Conflict
LOG SENSE	Allowed	Allowed	Allowed	Allowed	Allowed	Allowed
MODE SELECT(6)/ MODE SELECT(10)	Conflict	Conflict	Conflict	Allowed	Conflict	Conflict
MODE SENSE(6)/ MODE SENSE(10)	Conflict	Conflict	Conflict	Allowed	Conflict	Conflict
PERSISTENT RESERVE IN	Conflict	Allowed	Allowed	Allowed	Allowed	Allowed
PERSISTENT RESERVE OUT	see table 11					
PREVENT/ALLOW (Prevent=0)	Allowed	Allowed	Allowed	Allowed	Allowed	Allowed
PREVENT/ALLOW (Prevent<>0)	Conflict	Conflict	Conflict	Allowed	Conflict	Conflict
READ BUFFER	Conflict	Conflict	Conflict	Allowed	Conflict	Conflict
RECEIVE COPY RESULTS	Conflict	Conflict	Conflict	Allowed	Conflict	Conflict
RECEIVE DIAGNOSTICS	Conflict	Conflict	Conflict	Allowed	Conflict	Conflict
RELEASE(6)/RELEASE(10)	Allowed [a]	Conflict	Conflict	Conflict	Conflict	Conflict
REPORT DEVICE IDENTIFIER	Allowed	Allowed	Allowed	Allowed	Allowed	Allowed
REPORT LUNS	Allowed	Allowed	Allowed	Allowed	Allowed	Allowed
REQUEST SENSE	Allowed	Allowed	Allowed	Allowed	Allowed	Allowed
RESERVE(6)/RESERVE(10)	Conflict	Conflict	Conflict	Conflict	Conflict	Conflict
SEND DIAGNOSTICS	Conflict	Conflict	Conflict	Allowed	Conflict	Conflict
SET DEVICE IDENTIFIER	Conflict	Conflict	Conflict	Allowed	Conflict	Conflict
TEST UNIT READY	Conflict	Conflict	Conflict	Allowed	Conflict	Conflict
WRITE BUFFER	Conflict	Conflict	Conflict	Allowed	Conflict	Conflict

Key: LU=Logical Unit, Excl=Exclusive, RO=Registrants Only, <> Not Equal

[a] The reservation is not released (see 7.16 and 7.17).

Table 11 — PERSISTENT RESERVE OUT service actions that are allowed in the presence of various reservations

Command Service Action	Addressed LU is reserved by another initiator [A]	Addressed LU has this type of persistent reservation held by another initiator [B]	
		Command is from a registered initiator	Command is from a not registered initiator
CLEAR	Conflict	Allowed	Conflict
PREEMPT	Conflict	Allowed	Conflict
PREEMPT & ABORT	Conflict	Allowed	Conflict
REGISTER	Conflict	Allowed	Allowed
REGISTER AND IGNORE EXISTING KEY	Conflict	Allowed	Allowed
RELEASE	Conflict	Allowed [a]	Conflict
RESERVE	Conflict	Conflict	Conflict
Key: LU=Logical Unit			
[a] The reservation is not released (see 5.5.3.6.2).			

The time at which a reservation is established with respect to other tasks being managed by the device server is vendor specific. Successful completion of a reservation command indicates that the new reservation is established. A reservation may apply to some or all of the tasks in the task set before the completion of the reservation command. The reservation shall apply to all tasks received by the device server after successful completion of the reservation command. Any reserve/release command or persistent reserve service action shall be performed as a single indivisible event.

Multiple reserve/release commands or persistent reserve service actions may be present in the task set at the same time. The order of execution of such commands is defined by the tagged queueing restrictions, if any, but each is executed as a single indivisible command without any interleaving of actions that may be required by other reservation commands.

5.5.2 The Reserve/Release management method

The reserve/release management method commands, RESERVE(6), RESERVE(10), RELEASE(6), and RELEASE(10) are used among multiple initiators that do not require operations to be protected across initiator failures (and subsequent hard resets). The reserve/release reservations management method also allows an application client to provide restricted device access to one additional initiator within the same SCSI domain using the same SCSI protocol (a third-party initiator), usually a temporary initiator performing a service for the application client sending the reservation command.

Reservations managed using the reserve/release method do not persist across some recovery actions (e.g., hard resets). When a target performs one of these recovery actions, the application client(s) have to rediscover the configuration and re-establish the required reservations. Reserve/release managed reservations are retained by the device server until released or until reset by mechanisms specified in this standard.

The RESERVE(6) and RESERVE(10) commands allow superseding reservations.

5.5.3 The Persistent Reservations management method

5.5.3.1 Overview of the Persistent Reservations management method

The persistent reservations management method is the mechanism specified by this standard for use by multiple initiators that require operations to be protected across initiator failures, which usually involve hard resets. Persistent reservations persist across recovery actions, to provide initiators with more detailed control over reservations recovery. Persistent reservations are not reset by the TARGET RESET task management function or other global actions.

Persistent reservations for failing initiators may be preempted by another initiator as part of the recovery process. Persistent reservations shall be retained by the device server until released, preempted, or cleared by mechanisms specified in this standard. Even though different protocols that transport SCSI commands handle hard resets differently (e.g., parallel uses a reset signal, fibre channel loops use primitive signals) the persistent reservation shall be preserved. Optionally, persistent reservations may be retained when power to the target is removed.

The PERSISTENT RESERVE OUT and PERSISTENT RESERVE IN commands provide the basic mechanism for dynamic contention resolution in multiple initiator systems using multiple port targets. Before a persistent reservation may be established, an initiator shall register with a device server using a reservation key. Reservation keys are necessary to allow:

a) authentication of subsequent PERSISTENT RESERVE OUT commands;
b) identification of other initiators that are registered;
c) identification of the reservation key(s) that have an associated reservation;
d) preemption of a persistent reservation from a failing or uncooperative initiator; and
e) multiple initiators to participate in a reservation.

The reservation key provides a method for the application client to associate a protocol-independent identifier with an initiator on a specific port of a device server. The reservation key is used in the PERSISTENT RESERVE IN command to identify which initiators are registered and which initiator, if any, holds the reservation. The reservation key is used in the PERSISTENT RESERVE OUT command: to register an initiator, to verify the initiator issuing the PERSISTENT RESERVATION OUT command is registered, and to specify which initiator's registration or persistent reservation to preempt.

Reservation key values may be used by application clients to identify initiators, using application specific methods that are outside the scope of this standard. This standard provides the ability to register no more than one key per initiator/logical unit pair. Multiple initiators may use the same key for a logical unit. An initiator may establish registrations for multiple logical units in a SCSI device using any combination of unique or duplicate keys. These rules provide the ability for an application client to preempt multiple initiators with a single PERSISTENT RESERVE OUT command, but they do not provide the ability for the application client to uniquely identify the initiators using the PERSISTENT RESERVE commands.

5.5.3.2 Preserving persistent reservations

The application client may request activation of the persist through power loss device server capability to preserve the persistent reservation and registrations across power cycles by setting the APTPL bit to one in PERSISTENT RESERVE OUT parameter data sent with a REGISTER, or a REGISTER AND IGNORE EXISTING KEY service action.

After the application client enables the persist through power loss capability the device server shall preserve all current and future registrations and persistent reservations associated with the logical unit to which the REGISTER or the REGISTER AND IGNORE EXISTING KEY service action was addressed until an application client disables the persist through power loss capability. The APTPL value from the most recent successfully completed REGISTER or REGISTER AND IGNORE EXISTING KEY service action from any application client shall determine the logical unit's behavior in the event of a power loss.

The device server shall preserve the following information for each registration across any reset, and if the persist through power loss capability is enabled, across any power cycle:

a) Initiator identifier;
b) reservation key; and
c) when supported by the protocol, the initiator port's world wide identification.

The device server shall preserve the following reservation information across any reset, and if the persist through power loss capability is enabled, across any power cycle:

a) Initiator identifier;
b) reservation key;
c) scope;
d) type; and
e) when supported by the protocol, the initiator port's world wide identification.

For those protocols for which the initiator port's world wide identification is available to the device server the initiator port's world wide identification shall be used to determine if the initiator identifier has changed. This determination shall be made at any time the target detects that the configuration of the system may have changed. If the initiator identifier changed, the device server shall assign the new initiator identifier to the existing registration and reservation of the initiator port having the same world wide identification.

The capability of preserving persistent reservations and registrations across power cycles requires the use of a nonvolatile memory within the SCSI device. Any SCSI device that supports the persist through power loss capability of persistent reservation and has nonvolatile memory that is not ready shall allow the following commands into the task set:

a) INQUIRY;
b) LOG SENSE;
c) READ BUFFER;
d) REPORT LUNS;
e) REQUEST SENSE;
f) START/STOP UNIT (with the START bit set to one and POWER CONDITIONS field value of 0h); and
g) WRITE BUFFER.

When nonvolatile memory has not become ready since a power cycle, other than those listed above shall return CHECK CONDITION status. The sense key shall be set to NOT READY and the additional sense data shall be set as described in table 117 (see 7.25).

5.5.3.3 Finding persistent reservations and reservation keys

5.5.3.3.1 Summary of commands for finding persistent reservations and reservation keys

The application client may obtain information about the persistent reservation and the reservation keys that are present within a device server by issuing PERSISTENT RESERVE IN commands with a READ RESERVATION service action or a READ KEYS service action.

5.5.3.3.2 Reporting reservation keys

An application client may issue a PERSISTENT RESERVE IN command with a service action of READ KEYS to determine if any initiators have registered with a logical unit.

In response to a PERSISTENT RESERVE IN with a READ KEYS service action the device server shall report the following:

a) the current generation value (see 7.10.3); and
b) the reservation key for every initiator that is currently registered.

The generation value allows the application client to verify that the configuration of the initiators registered with a logical unit has not been modified.

The application client may examine the reservation keys to identify relationships between initiators based on mechanisms that are outside the scope of this standard. Duplicate keys shall be reported if multiple initiators use the same reservation key.

5.5.3.3.3 Reporting persistent reservations

An application client may issue a PERSISTENT RESERVE IN command with a service action of READ RESERVATION to receive the persistent reservation information.

In response to a PERSISTENT RESERVE IN command with a READ RESERVATION service action the device server shall report the following as an uninterrupted series of actions:

a) the current generation value (see 7.10.3);
b) the registered reservation key, if any, associated with the initiator that holds the persistent reservation;
c) the scope and type of each persistent reservation, if any; and
d) the scope-specific address, if any (see 7.10.4.1).

If an application client uses a different reservation key for each initiator/logical unit pair the application client may use the reservation key to associate the persistent reservation with the initiator that holds the persistent reservation. This association is done using techniques that are outside the scope of this standard.

5.5.3.4 Registering

To establish a persistent reservation the initiator shall first register with a logical unit. An initiator registers with a logical unit by issuing a PERSISTENT RESERVE OUT command with service action of REGISTER or REGISTER AND IGNORE EXISTING KEY.

If the initiator has not yet established a reservation key or the reservation key has been removed, the registration is accomplished by issuing a PERSISTENT RESERVE OUT command with service action of REGISTER with the following parameters:

a) APTPL bit optionally set to one;
b) reservation key set to zero; and
c) service action reservation key set to a non-zero value.

If the initiator has an established registration it may change its reservation key. This is accomplished by issuing a PERSISTENT RESERVE OUT command with service action of REGISTER with the following parameters:

a) APTPL bit optionally set to one;
b) reservation key set to the value of the previously established reservation key; and
c) service action reservation key set to a non-zero value.

Alternatively, an initiator may establish a reservation key without regard for whether one has previously been established by issuing a PERSISTENT RESERVE OUT command with a service action of REGISTER AND IGNORE EXISTING KEY and the following parameters:

a) APTPL bit optionally set to one; and
b) service action reservation key set to a non-zero value.

If a PERSISTENT RESERVE OUT with a REGISTER AND IGNORE EXISTING KEY service action is sent when an established registration exists, that registration shall be superseded with the specified service action reservation key. If a PERSISTENT RESERVE OUT with a REGISTER AND IGNORE EXISTING KEY service action is sent when there is no established registration, a new registration shall be established.

If a PERSISTENT RESERVE OUT with a REGISTER or a REGISTER AND IGNORE EXISTING KEY service action is attempted, but there are insufficient device server resources to complete the operation, the device server shall return a CHECK CONDITION status. The sense key shall be set to ILLEGAL REQUEST and the additional sense data shall be set to INSUFFICIENT REGISTRATION RESOURCES.

In response to a PERSISTENT RESERVE OUT with a REGISTER or a REGISTER AND IGNORE EXISTING KEY service action the device server shall perform a registration by doing the following as an uninterrupted series of actions:

a) Process the registration request regardless of any persistent reservations;
b) process the APTPL bit;
c) ignore the contents of the SCOPE and TYPE fields;
d) map the reservation key to the registering initiator using the initiator identification and, if available, the initiator port's world wide identification;
e) register the reservation key without changing any persistent reservation that may exist; and
f) retain the reservation key and associated information.

After the registration request has been processed, the device server shall then allow other PERSISTENT RESERVE OUT commands from the registered initiator to execute. For each initiator that performs a PERSISTENT RESERVE OUT with a REGISTER or a REGISTER AND IGNORE EXISTING KEY service action, the device server shall retain the reservation key until the key is changed by a new PERSISTENT RESERVE OUT command with the REGISTER or the REGISTER AND IGNORE EXISTING KEY service action from the same initiator or until the initiator registration is removed (see 5.5.3.6).

Any PERSISTENT RESERVE OUT command service action received from an unregistered initiator, other than the REGISTER or the REGISTER AND IGNORE EXISTING KEY service action, shall be rejected with a RESERVATION CONFLICT status.

It is not an error for an initiator that is registered to register again with the same reservation key or a new reservation key. A registration shall have no effect on any other registrations (e.g., when more than one initiator is registered with the same reservation key and one of those initiators registers again it has no effect on the other initiator's registrations). A registration that contains a non-zero value in the service action reservation key field shall have no effect on any reservations (i.e., an initiator may change its reservation key without affecting any previously created reservation).

Multiple initiators may register with the same reservation key. An initiator may use the same reservation key for multiple logical units.

5.5.3.5 Creating a persistent reservation when there is no persistent reservation

An application client creates a persistent reservation by issuing a PERSISTENT RESERVE OUT command with a service action of RESERVE through a registered initiator with the following parameters:

a) RESERVATION KEY set to the value of the initiator/logical unit pair's established reservation key; and
b) TYPE and SCOPE set to the reservation being created.

Only one persistent reservation with a scope of logical unit is allowed at a time per logical unit. Multiple persistent reservations with a scope of element may be created in a logical unit that contains multiple elements. However, there shall only be one persistent reservation per element.

If the target receives a PERSISTENT RESERVE OUT command that attempts to create a persistent reservation when a persistent reservation already exists for the logical unit from an initiator other than the initiator that created the reservation, then the command shall be rejected with a RESERVATION CONFLICT status.

If the initiator that created the persistent reservation attempts to modify the TYPE or SCOPE of an existing reservation, then the command shall be rejected with a RESERVATION CONFLICT status.

If the device server receives a PERSISTENT RESERVE OUT command with a service action of RESERVE where the TYPE and SCOPE are the same as the existing TYPE and SCOPE from the initiator that created the persistent reservation, it shall not make any change to the existing reservation and shall return a GOOD status.

If the target receives a RESERVE(10) or RESERVE(6) command when a persistent reservation exists for the logical unit then the command shall be rejected with a RESERVATION CONFLICT.

See 5.5.1 for information on when a persistent reservation takes effect.

5.5.3.6 Removing registrations and persistent reservations

5.5.3.6.1 Overview of removing registrations and persistent reservations

A registered initiator using the value of the initiator/logical unit pair's established reservation key may release or preempt a persistent reservation by issuing one of the following commands:

a) a PERSISTENT RESERVE OUT command with a service action of RELEASE from the initiator that performed the reservation (see 5.5.3.6.2);
b) a PERSISTENT RESERVE OUT command with a PREEMPT service action specifying the reservation key of the initiator holding the reservation (see 5.5.3.6.3);
c) a PERSISTENT RESERVE OUT command with a PREEMPT AND ABORT service action specifying the reservation key of the initiator holding the reservation (see 5.5.3.6.4); or
d) a PERSISTENT RESERVE OUT command with a service action of CLEAR service action (see 5.5.3.6.5).

A registered initiator using the value of the initiator/logical unit pair's established reservation key may remove a registration by issuing one of the following commands:

a) a PERSISTENT RESERVE OUT command with a PREEMPT service action specifying that reservation key (see 5.5.3.6.3);
b) a PERSISTENT RESERVE OUT command with a PREEMPT AND ABORT service action specifying that reservation key (see 5.5.3.6.4);
c) a PERSISTENT RESERVE OUT command with a CLEAR service action (see 5.5.3.6.5); or
d) a PERSISTENT RESERVE OUT command with a REGISTER or a REGISTER AND IGNORE EXISTING KEY service action from the same initiator with the value of the service action reservation key field set to zero.

When a reservation key has been removed, no information shall be reported for that unregistered initiator in subsequent READ KEYS service action(s) until the initiator is registered again (see 5.5.3.4). Any persistent reservation associated with that unregistered initiator shall be released. If that released persistent reservation was of the type Write Exclusive – Registrants Only or Exclusive Access – Registrants Only the device server shall establish a unit attention condition for all registered initiators other than the initiator that issued the PERSISTENT RESERVE OUT command with PREEMPT or PREEMPT AND ABORT service action. The sense key shall be set to UNIT ATTENTION and the additional sense data shall be set to RESERVATIONS RELEASED.

A persistent reservation may also be released by a loss of power, if the persist through power loss capability is not enabled. When the most recent APTPL value received by the device server is zero (see 7.11.3), a power cycle:

a) performs a hard reset;
b) releases all persistent reservations; and
c) removes all registered reservation keys (see 5.5.3.4).

5.5.3.6.2 Releasing a persistent reservation

Only the initiator that creates the persistent reservation is allowed to release that persistent reservation.

An application client releases a persistent reservation it holds by issuing a PERSISTENT RESERVE OUT command with a service action of RELEASE through the registered initiator that holds the persistent reservation with the following parameters:

a) RESERVATION KEY set to the value of the initiator/logical unit pair's established reservation key; and
b) TYPE and SCOPE set to match the persistent reservation being released.

In response to a persistent reservation release request from an initiator that created the persistent reservation the device server shall perform a release by doing the following as an uninterrupted series of actions:

a) Releasing the persistent reservation;
b) Not removing any registration(s);
c) If the released persistent reservation has a type of Write Exclusive – Registrants Only or Exclusive Access – Registrants Only the device server shall establish a unit attention condition for all registered initiators

other than the initiator that issued the PERSISTENT RESERVE OUT command with RELEASE service action. The sense key shall be set to UNIT ATTENTION and the additional sense data shall be set to RESERVATIONS RELEASED; and

d) If the persistent reservation is of any other type the device server shall not establish a unit attention condition.

The established reservation shall not be altered and the device server shall return a CHECK CONDITION status for a PERSISTENT RESERVE OUT command that specifies the release of a persistent reservation held by the requesting initiator if the SCOPE and TYPE fields do not match the scope and type of the established persistent reservation. The sense key shall be set to ILLEGAL REQUEST and additional sense data shall be set to INVALID RELEASE OF PERSISTENT RESERVATION.

If there is no persistent reservation or in response to a persistent reservation release request from a registered initiator using the value of the initiator/logical unit pair's established reservation key that does not hold the persistent reservation, the device server shall do the following:

a) Not release the persistent reservation (if any);
b) Not remove or change any registration(s); and
c) Return a status of GOOD.

5.5.3.6.3 Preempting an existing persistent reservation with the PREEMPT service action

5.5.3.6.3.1 Overview of preempting an existing persistent reservation with the PREEMPT service action

A PERSISTENT RESERVE OUT command with a PREEMPT service action is used to preempt a persistent reservation and/or registration. The determination of whether the preempt relates to a persistent reservation or a registration is made by the device server by examining the value in the SERVICE ACTION RESERVATION KEY field of the PREEMPT service action. If the value in the SERVICE ACTION RESERVATION KEY field is associated with the reservation being preempted then the reservation is preempted and any matching registration(s) removed (see 5.5.3.6.3.3).

If the value in the SERVICE ACTION RESERVATION KEY field is not associated with the reservation, the reservation shall not be preempted but any matching registration(s) shall be removed (see 5.5.3.6.3.4).

See Figure 2 for a description of how a device server interprets a PREEMPT service action to determine its actions (e.g., preempt persistent reservation, remove registration or both preempt persistent reservation and remove reservation).

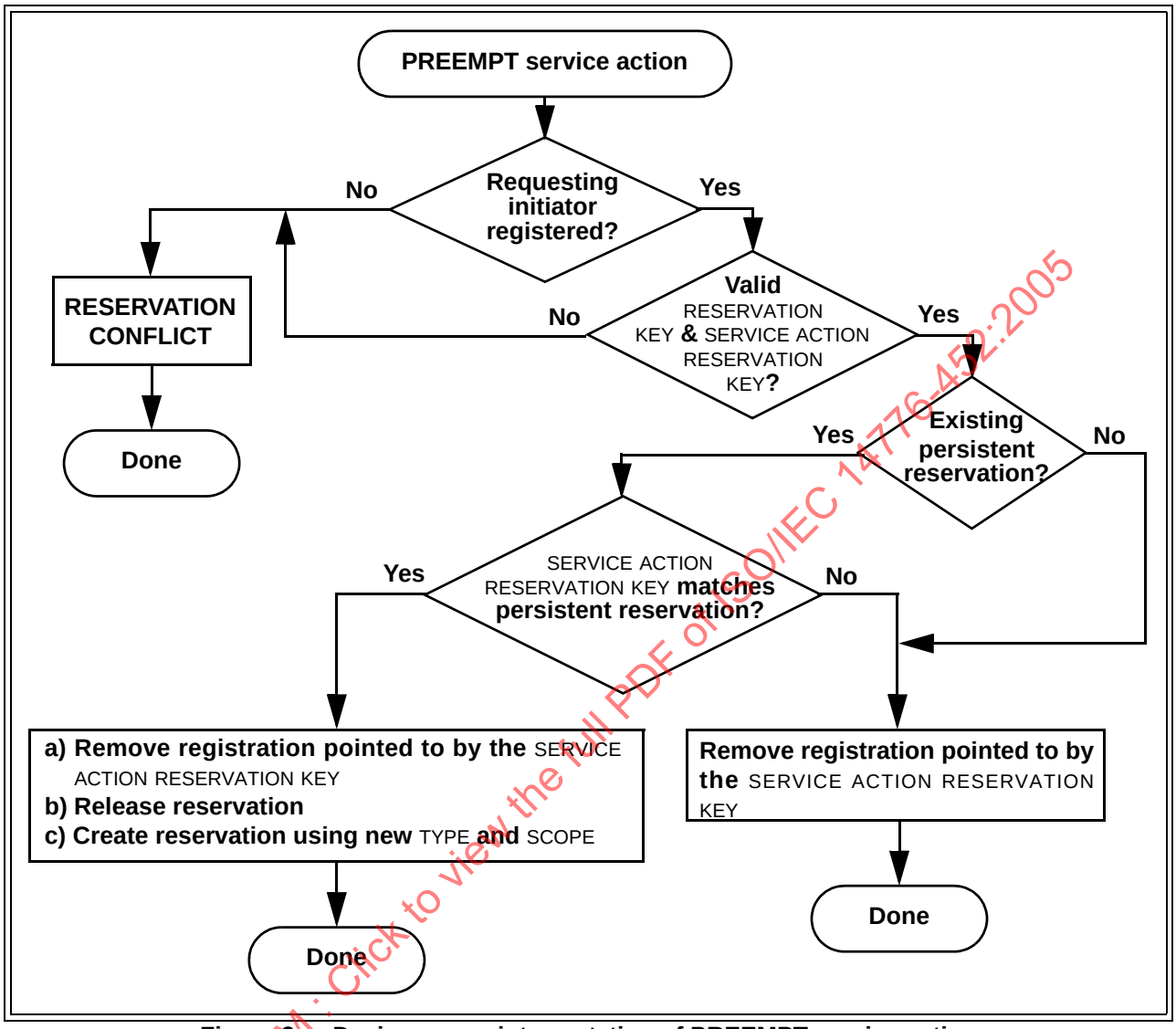

Figure 2 — Device server interpretation of PREEMPT service action

5.5.3.6.3.2 Failed persistent reservation preempt

If the preempting initiator's PREEMPT service action or PREEMPT AND ABORT service action fails (e.g., TASK SET FULL, BUSY, transport protocol time-out or time-out due to queue blocked due to failed initiator), the initiator may issue a LOGICAL UNIT RESET task management function to the failing logical unit to remove blocking tasks and then reissue the preempting service action.

5.5.3.6.3.3 Preempting reservations

Any registered initiator may preempt any persistent reservation with another persistent reservation by issuing a PERSISTENT RESERVE OUT command with a PREEMPT service action through a registered initiator with the following parameters:

a) RESERVATION KEY set to the value of the initiator/logical unit pair's established reservation key;
b) SERVICE ACTION RESERVATION KEY set to match the reservation key of the persistent reservation being preempted; and
c) TYPE and SCOPE set to define a new persistent reservation. The scope and TYPE of the persistent reservation created by the preempting initiator may be different from the persistent reservation being preempted.

If the SERVICE ACTION RESERVATION KEY is associated with a reservation, the device server shall perform a preempt by doing the following as an uninterrupted series of actions:

a) Release the persistent reservation for the initiator identified by the SERVICE ACTION RESERVATION KEY specified in the PERSISTENT RESERVE OUT parameter list;
b) Remove the registration for the initiator or initiators identified by the SERVICE ACTION RESERVATION KEY specified in the PERSISTENT RESERVE OUT parameter list (see 5.5.3.4);
c) establish a persistent reservation for the preempting initiator;
d) process tasks as defined in 5.5.1; and
e) establish a unit attention condition for any initiator that lost its reservation and/or registration. The sense key shall be set to UNIT ATTENTION and the additional sense data shall be set to REGISTRATIONS PREEMPTED.

After GOOD status has been returned for the PERSISTENT RESERVE OUT command, new tasks are subject to the persistent reservation restrictions established by the preempting initiator.

The following tasks shall be subjected in a vendor specific manner either to the restrictions established by the persistent reservation being preempted or to the restrictions established by the preempting initiator:

a) A task received after the arrival, but before the completion of the PERSISTENT RESERVE OUT command with the PREEMPT service action; or
b) A task in the dormant, blocked, or enable state at the time the PERSISTENT RESERVE OUT command with the PREEMPT service action is received.

Completion status shall be returned for each task.

A PERSISTENT RESERVE OUT specifying a PREEMPT service action with the SERVICE ACTION RESERVATION KEY value equal to the reservation key is not an error. In that case the device server shall establish the new reservation.

5.5.3.6.3.4 Removing registrations

When a registered reservation key is not associated with a persistent reservation, an application client may remove the registration(s) without affecting any persistent reservations by issuing a PERSISTENT RESERVE OUT command with a PREEMPT service action through a registered initiator with the following parameters:

a) RESERVATION KEY set to the value of the initiator/logical unit pair's established reservation key; and
b) SERVICE ACTION RESERVATION KEY set to match the reservation key of the registration being removed.

If the SERVICE ACTION RESERVATION KEY field is not associated with a reservation the device server shall perform a preempt by doing the following in an uninterrupted series of actions:

a) remove the registration for the initiator or initiators identified by the SERVICE ACTION RESERVATION KEY field specified in the PERSISTENT RESERVE OUT parameter list;
b) ignore the contents of the SCOPE and TYPE fields;
c) process tasks as defined in 5.5.1; and
d) establish a unit attention condition for any other initiator that lost its registration. The sense key shall be set to UNIT ATTENTION and the additional sense data shall be set to REGISTRATIONS PREEMPTED.

If a PERSISTENT RESERVE OUT specifying a PREEMPT service action sets the SERVICE ACTION RESERVATION KEY field to a value that does not match any registered reservation key the device server shall return a RESERVATION CONFLICT status.

5.5.3.6.4 Preempting an existing persistent reservation with the PREEMPT AND ABORT service action

The initiator's request for and the device server's responses to a PERSISTENT RESERVE OUT command PREEMPT AND ABORT service action are identical to the PREEMPT service action (see 5.5.3.6.3) except for the

following additions. If no reservation conflict occurred, the device server shall perform the following uninterrupted series of actions:

a) Perform the uninterrupted series of actions described for the PREEMPT service action (see 5.5.3.6.3);
b) All tasks from preempted initiators (called preempted tasks) shall be terminated and initiator notification shall be provided as specified by the TAS bit in the Control mode page (see 8.3.6) that applies to the preempted initiator as follows:
   A) If the TAS bit is set to zero then all preempted tasks shall be terminated as if an ABORT TASK SET task management function had been performed by each preempted initiator; or
   B) If the TAS bit is set to one then all preempted tasks from initiators other than the initiator that sent the PREEMPT AND ABORT service action shall be terminated with a TASK ABORTED status (see SAM-2). The preempted tasks from the initiator that sent the PREEMPT AND ABORT service action (if any) shall be terminated as if an ABORT TASK SET task management function had been performed by that initiator.

   If a terminated task is a command that causes the device server to generate additional commands and data transfers (e.g., EXTENDED COPY), all commands and data transfers generated by the command shall be terminated before the ABORT TASK SET task management function is considered completed. After the ABORT TASK SET function has completed, all new tasks are subject to the persistent reservation restrictions established by the preempting initiator;
c) The device server shall clear any ACA or CA condition associated with an initiator being preempted and shall clear any tasks with an ACA attribute from that initiator. If the TST field is 000b (see 8.3.6) and ACA or CA conditions exist for initiators other than the initiator being preempted, the PERSISTENT RESERVE OUT command shall be terminated prior to processing with a status of ACA ACTIVE if the NACA bit equals one (see SAM-2) or BUSY if the NACA equals zero. If the TST field contains 001b, then ACA or CA conditions for initiators other than the initiator being preempted shall not prevent the execution of the PERSISTENT RESERVE OUT command; and
d) For SCSI devices that implement the PREVENT ALLOW MEDIUM REMOVAL command, the device server shall perform an action equivalent to the execution of a PREVENT ALLOW MEDIUM REMOVAL command with the PREVENT field equal to zero for the initiator or initiators being preempted (see 7.12).

The actions described in the preceding list shall be performed for all initiators that are registered with the SERVICE ACTION RESERVATION KEY value, without regard for whether the preempted initiator(s) hold the reservation.

Any asynchronous event reporting operations in progress are not affected by the PREEMPT AND ABORT service action.

5.5.3.6.5 Clearing a persistent reservation

Any application client may release a persistent reservation and remove all registrations from a device server for a specific logical unit by issuing a PERSISTENT RESERVE OUT command with a service action of CLEAR service action through a registered initiator with the following parameter:

a) RESERVATION KEY set to the value of the initiator/logical unit pair's established reservation key.

In response to this request the device server shall perform a clear by doing the following as part of an uninterrupted series of actions:

a) Release any persistent reservation associated with the logical unit;
b) Remove all registration(s) (see 5.5.3.4);
c) Ignore the contents of the SCOPE and TYPE fields;
d) continue normal execution of any tasks from any initiator that have been accepted by the device server as nonconflicting; and
e) establish a unit attention condition for all other registered initiators, if any, for the cleared logical unit. The sense key shall be set to UNIT ATTENTION and the additional sense data shall be set to RESERVATIONS PREEMPTED.

Application clients should not use the CLEAR service action except during recoveries that are associated with initiator or system reconfiguration, since the effect of the CLEAR service action is to remove the persistent reservation management conventions that protect data integrity.

5.6 Multiple port and multiple initiator behavior

SAM-2 specifies the behavior of logical units being accessed by more than one initiator. Additional service delivery ports provide alternate service delivery paths through which the device server may be reached and may also provide connectivity for additional initiators. An alternate path may be used to improve the availability of devices in the presence of certain types of failures and to improve the performance of devices whose other paths may be busy.

If a SCSI device has more than one service delivery port, the arbitration and connection management among the service delivery ports is vendor specific. If one service delivery port is being used by an initiator, accesses attempted through other service delivery port(s) may:

a) receive a status of BUSY; or
b) be accepted as if the other service delivery port(s) were not in use.

The device server shall indicate the presence of multiple ports by setting the MULTIP bit to 1 in its standard INQUIRY data.

For the purposes of handling reservations, other initiators are defined as all initiators on the same service delivery port except the initiator holding the reservation and all initiators on all other service delivery ports. Only the following operations allow an initiator to interact with the tasks of another initiator, regardless of the service delivery port:

a) the PERSISTENT RESERVE OUT with PREEMPT service action preempts persistent reservations for other initiators (see 5.5.3.6.3);
b) the PERSISTENT RESERVE OUT with PREEMPT AND ABORT service action preempts persistent reservations and all tasks for other initiators (see 5.5.3.6.4);
c) the PERSISTENT RESERVE OUT with CLEAR service action releases persistent reservations and removes reservation keys for all initiators (see 5.5.3.6.5);
d) the TARGET RESET task management function releases reservations established by the reserve/release method and removes all tasks for all logical units in the target and for all initiators (see SAM-2). Persistent reservations remain unmodified;
e) the LOGICAL UNIT RESET task management function releases reservations established by the reserve/release method and removes all tasks for all initiators for the addressed logical unit and any logical units issuing from it in a hierarchical addressing structure (see SAM-2). Persistent reservations remain unmodified; and
f) the CLEAR TASK SET task management function removes all tasks for the selected logical unit for all initiators. Most other logical unit states remain unmodified, including MODE SELECT parameters, reservations, and ACA (see SAM-2).

5.7 Removable medium devices with an attached medium changer

When a logical unit is served by a medium changer, control over one medium transport element may be effected using medium changer commands sent to the device server within the logical unit. The level of control is not as complete as would be available if a fully functional medium-changer device server were implemented (see SMC). However, the amount of control is sufficient for paired device and medium changer configurations.

The device server shall indicate its ability to support medium changer commands by setting the MCHNGR bit to one in its standard INQUIRY data (see 7.3.2). An MCHNGR bit of one shall indicate that the MOVE MEDIUM ATTACHED and READ ELEMENT STATUS ATTACHED commands are supported by the device server. Definitions of the MOVE MEDIUM ATTACHED and READ ELEMENT STATUS ATTACHED commands may be found in SMC.

Only one medium transport element shall be permitted, element 0. Only one data transfer element shall be permitted. Media exchanges shall not be supported by attached medium changers. The RESERVE ELEMENT and RELEASE ELEMENT commands shall not be supported by attached medium changers.

6 Model for processor devices

A SCSI processor device is a primary computing device with the characteristics of a device server, such as a personal computer, minicomputer, mainframe computer, or auxiliary computing device or server. Such a primary computing device is often called a host. The processor device receives or provides packets of data as requested by an application client.

In the SCSI processor device, the device server accepts and provides data packets transferred according to commands from the application client. An application client and the processor device server are assumed to have a common set of rules by which information is to be exchanged between them, how the information is interpreted by the processor device server, and when it is allowable to exchange the information. These rules are not specified by this standard.

The application client requests that the processor device server accept a packet of data by transmitting a SEND command. The application client requests that the processor device server return a packet of data by transmitting a RECEIVE command. A COPY or EXTENDED COPY command may also be transmitted to the processor device server to request that it serve as a copy manager. The data flow may be between the processor device and another SCSI device or may be between two SCSI devices under control of the processor device acting as a third-party copy manager.

If a processor device server has no resource available to manage a data packet from the application client and has no data packet to provide to the application client, or has no resources assigned to perform the operation, the device server may choose one of the following responses:

a) Terminate the command with CHECK CONDITION status and the sense key NOT READY with the appropriate additional sense code for the condition. This is the appropriate response to a TEST UNIT READY command;
b) Delay data transmission until the necessary resource or data packet becomes available;
c) Terminate the command with BUSY status; or
d) Treat the logical unit as an incorrect logical unit (see SAM-2).

A single target may have more than one logical unit. Logical units may serve as additional paths to a single resource, and/or each logical unit may serve as a path to different resources within the device. A single logical unit may also serve as a path to multiple resources if the processor device server interprets information within the data packet and routes the packet to the appropriate resource.

If the processor device server determines that an exception condition has occurred while performing an operation specified by the contents of a data packet, the information describing the condition is returned as a part of a data packet. If the processor device server determines that an exception condition has occurred while executing the SCSI command from the application client, the command is terminated with a CHECK CONDITION and the failures are identified through the sense data.

Many types of devices may function as processor devices if no other suitable SCSI device type exists and if the packet exchange protocol specified by the processor device model meets their functional requirements.

Processor device types shall not implement element reservations.

Reservation restrictions are placed on commands as a result of access qualifiers associated with the type of reservation. The details of which commands are allowed under what types of reservations are described in table 12. For the reservation restrictions placed on commands for the reserve/release management method see table 12 column [A]. For the reservation restrictions placed on commands for the persistent reservations management method, see the columns under [B] in table 12.

In table 12 the following key words are used:

allowed: Commands issued by initiators not holding the reservation or by initiators not registered when a registrants only persistent reservation is present should complete normally.

conflict: Commands issued by initiators not holding the reservation or by initiators not registered when a registrant's only persistent reservation is present shall not be performed and the device server shall terminate the command with a RESERVATION CONFLICT status.

Commands from initiators holding a reservation should complete normally. The behavior of commands from registered initiators when a registrants only persistent reservation is present is specified in table 12.

A command that does not explicitly write the medium shall be checked for reservation conflicts before the command enters the current task state for the first time. Once the command has entered the current task state, it shall not be terminated with a RESERVATION CONFLICT due to a subsequent reservation.

A command that explicitly writes the medium shall be checked for reservation conflicts before the device server modifies the medium or cache as a result of the command. Once the command has modified the medium, it shall not be terminated with a RESERVATION CONFLICT due to a subsequent reservation.

Table 12 — Processor commands that are allowed in the presence of various reservations

Command	Addressed LU is reserved by another initiator [A]	Addressed LU has this type of persistent reservation held by another initiator [B]				
		From any initiator		From registered initiator (RO all types)	From initiator not registered	
		Write Excl	Excl Access		Write Excl – RO	Excl Access – RO
RECEIVE	Conflict	Allowed	Conflict	Allowed	Allowed	Conflict
SEND	Conflict	Conflict	Conflict	Allowed	Conflict	Conflict
Key: LU=Logical Unit, Excl=Exclusive, RO=Registrants Only						

7 Commands for all device types

7.1 Summary of commands for all device types

The operation codes for commands that apply to all device types are listed in table 13.

Table 13 — Commands for all device types

Command name	Operation code	Type	Reference
Obsolete	40h	OB	
Obsolete	39h	OB	
Obsolete	18h	OB	
Obsolete	3Ah	OB	
EXTENDED COPY	83h	O	7.2
INQUIRY	12h	M	7.3
LOG SELECT	4Ch	O	7.4
LOG SENSE	4Dh	O	7.5
MODE SELECT(6)	15h	Z	7.6
MODE SELECT(10)	55h	Z	7.7
MODE SENSE(6)	1Ah	Z	7.8
MODE SENSE(10)	5Ah	Z	7.9
MOVE MEDIUM ATTACHED [a]	A7h	Z	SMC
PERSISTENT RESERVE IN	5Eh	Z	7.10
PERSISTENT RESERVE OUT	5Fh	Z	7.11
PREVENT ALLOW MEDIUM REMOVAL	1Eh	Z	7.12
READ BUFFER	3Ch	O	7.13
READ ELEMENT STATUS ATTACHED [a]	B4h	Z	SMC
RECEIVE COPY RESULTS	84h	O	7.14
RECEIVE DIAGNOSTIC RESULTS	1Ch	O	7.15
RELEASE(10)	57h	Z	7.16
RELEASE(6)	17h	Z	7.17
REPORT DEVICE IDENTIFIER	A3h/05h [b]	O	7.18
REPORT LUNS	A0h	X	7.19
REQUEST SENSE	03h	Z	7.20
RESERVE(10)	56h	Z	7.21
RESERVE(6)	16h	Z	7.22
SEND DIAGNOSTIC	1Dh	Z	7.23
SET DEVICE IDENTIFIER	A4h/06h [b]	O	7.24
TEST UNIT READY	00h	M	7.25
WRITE BUFFER	3Bh	Z	7.26

Key: M = Command implementation is mandatory.
O = Command implementation is optional.
OB = Command implementation is defined in a previous standard
X = Command implementation requirements given in reference subclause of this standard.
Z = Command implementation is device type specific.

[a] The MOVE MEDIUM ATTACHED and READ ELEMENT STATUS ATTACHED operation codes shown here should be used by devices with attached medium changers.

[b] This command is defined by a combination of operation code and service action. The operation code value is shown preceding the slash and the service action value is shown after the slash.

7.2 EXTENDED COPY command

7.2.1 EXTENDED COPY command introduction

The EXTENDED COPY command (see table 14) provides a means to copy data from one set of logical units to another set or to the same set of logical units. The entity within a device server that receives and performs the EXTENDED COPY command is called the copy manager. The copy manager is responsible for copying data from the source device(s) to the destination device(s). The copy source and destination devices are logical units that may reside in different SCSI devices or the same SCSI device. It is possible that all the SCSI devices and the copy manager are the same logical unit.

Table 14 — EXTENDED COPY command

Bit / Byte	7	6	5	4	3	2	1	0
0	OPERATION CODE (83h)							
1	Reserved							
2	Reserved							
3	Reserved							
4	Reserved							
5	Reserved							
6	Reserved							
7	Reserved							
8	Reserved							
9	Reserved							
10	(MSB)							
11				PARAMETER LIST LENGTH				
12								
13								(LSB)
14	Reserved							
15	CONTROL							

Before the copy manager is instructed to move data, the application controlling the data movement shall independently take any necessary actions required to prepare the source and destination devices for the EXTENDED COPY command. These actions may include media changer commands, loading of tapes, MODE SELECT commands reservation commands, positioning of tape, etc. After all preparatory actions have been accomplished, the EXTENDED COPY command should be issued to the copy manager to start the data transfer.

The PARAMETER LIST LENGTH field specifies the length in bytes of the parameter data that shall be contained in the Data-Out Buffer. A parameter list length of zero indicates that copy manager shall not transfer any data or alter any internal state; this shall not be considered an error. If the parameter list length causes truncation of the parameter list in a target descriptor or segment descriptor, no data shall be transferred and the EXTENDED COPY command shall be terminated with a CHECK CONDITION status. The sense key shall be set to ILLEGAL REQUEST and the additional sense code shall be set to PARAMETER LIST LENGTH ERROR.

The EXTENDED COPY parameter list (see table 15) begins with a sixteen byte header that contains the LIST IDENTIFIER field, the STR, and NRCR bits, the command's priority, the length of the target descriptor list, the length of the segment descriptor list, and the length of the optional inline data. Immediately following the header is one or more target descriptors, followed by one or more segment descriptors, followed by any optional inline data.

Table 15 — EXTENDED COPY parameter list

Bit / Byte	7	6	5	4	3	2	1	0
0	LIST IDENTIFIER							
1	Reserved		STR	NRCR	Reserved	PRIORITY		
2	(MSB)							
3	TARGET DESCRIPTOR LIST LENGTH (n-15)							(LSB)
4								
7	Reserved							
8	(MSB)							
11	SEGMENT DESCRIPTOR LIST LENGTH (m-n)							(LSB)
12	(MSB)							
15	INLINE DATA LENGTH							(LSB)
	Target descriptor(s)							
16								
47	Target descriptor 0							
	.							
	.							
n-31								
n	Target descriptor x							
	Segment descriptor(s)							
n+1								
n+1+s	Segment descriptor 0 (See specific table for length.)							
	.							
	.							
m	Segment descriptor y (See specific table for length.)							
	Inline data							

NOTE 1 Unexpected results may occur when an initiator fails to zero the reserved bytes in this parameter list. Copy managers should insure that the reserved bytes 4 through 7 contain zeros.

The LIST IDENTIFIER field is a value selected by the application client to uniquely identify the extended copy operation to the copy manager. The list identifier also may be used in the RECEIVE COPY RESULTS command (see 7.14) to request status for a specific EXTENDED COPY command. The LIST IDENTIFIER value shall be unique for each concurrent EXTENDED COPY command sent by an initiator. If the copy manager detects a duplicate LIST IDENTIFIER value the command shall be terminated with a CHECK CONDITION, the sense key shall be set to ILLEGAL REQUEST and the additional sense code shall be set to OPERATION IN PROGRESS.

The PRIORITY field establishes the priority of data transfer operations resulting from this EXTENDED COPY command relative to data transfer operations resulting from other commands being executed by the same device server. All commands other than copy commands have a priority of 1h. Priority 0h is the highest priority, with increasing PRIORITY values indicating lower priorities.

A Sequential Striped bit (STR) value of one indicates to the copy manager that the majority of the disk references in the parameter list represent sequential access of several striped disks. This may be used by the copy manager to perform read operations from a source disk at any time and in any order during processing of an EXTENDED COPY command as described in 7.2.6.7. A STR value of zero indicates to the copy manager that disk references are not necessarily sequential.

If the No Receive Copy Results (NRCR) bit is zero, the copy manager shall hold data for retrieval by the application client using the RECEIVE COPY RESULTS command with the RECEIVE DATA service action (see 7.14.3) and specified by the segment descriptors. If NRCR is one, the copy manager may discard all data accessible to the application client via the RECEIVE COPY RESULTS command with the RECEIVE DATA service action. If the application client requests delivery of data that has been discarded as a result of NRCR being one, the copy manager shall respond as if the EXTENDED COPY command has not been processed.

The TARGET DESCRIPTOR LIST LENGTH contains the length in bytes of the target descriptor list that immediately follows the parameter list header. The number of target descriptors equals the length in bytes of the target descriptor list divided by 32.

An EXTENDED COPY command may reference one or more target devices (the name given by the EXTENDED COPY command description to source and/or the destination logical units). Each target device is described by a target descriptor. All target descriptors have their formats specified by an EXTENDED COPY descriptor code. A copy manager may not support all target descriptor formats and shall list all target descriptor formats supported in response to the RECEIVE COPY RESULTS command with OPERATING PARAMETERS service action (see 7.14.4). See 7.2.6 for a detailed description of the target descriptors.

Segment descriptors reference target descriptors by their position, or index, in the target descriptor list. The index for a target descriptor is computed by subtracting 16 from the starting byte number for the target descriptor in the parameter data and dividing the result by 32. The maximum number of target descriptors permitted within a parameter list is indicated by the MAXIMUM TARGET COUNT field in the copy manager's operating parameters (see 7.14.4). If the number of target descriptors exceeds the allowed number, the command shall be terminated with CHECK CONDITION status. The sense key shall be set to ILLEGAL REQUEST and the additional sense code shall be set to TOO MANY TARGET DESCRIPTORS.

The SEGMENT DESCRIPTOR LIST LENGTH contains the length in bytes of the segment descriptor list that follows the target descriptors. See 7.2.7 for a detailed description of the segment descriptors. The maximum number of segment descriptors permitted within a parameter list is indicated by the MAXIMUM SEGMENT COUNT field in the copy manager's operating parameters (see 7.14.4). If the number of segment descriptors exceeds the allowed number, the command shall be terminated with CHECK CONDITION status. The sense key shall be set to ILLEGAL REQUEST and the additional sense code shall be set to TOO MANY SEGMENT DESCRIPTORS.

The maximum length of the target and segment descriptors permitted within a parameter list is indicated by the MAXIMUM DESCRIPTOR LIST LENGTH field in the copy manager's operating parameters (see 7.14.4). If the combined length of the target and segment descriptors exceeds the allowed value, the command shall be terminated with CHECK CONDITION status. The sense key shall be set to ILLEGAL REQUEST and the additional sense code shall be set to PARAMETER LIST LENGTH ERROR.

The INLINE DATA LENGTH field contains the number of bytes of inline data, after the last segment descriptor. A value of zero indicates that no inline data is present.

The copy manager shall move data from the source devices to the destination devices as prescribed by the segment descriptors. The specific commands issued by the copy manager to the source and destination devices while processing the segment descriptors is vendor specific. Upon completion of an EXTENDED COPY command that returns GOOD status, the source and destination devices, particularly stream devices, shall be positioned at deterministic locations such that the device may be repositioned to the same location by the application client with appropriate commands.

7.2.2 Errors detected before starting processing of the segment descriptors

Errors may occur during processing of an EXTENDED COPY command before the first segment descriptor is processed. These conditions include CRC or parity errors while transferring the EXTENDED COPY command,

invalid parameters in the CDB or parameter data, invalid segment descriptors, and inability of the copy manager to continue operating. In the event of such an exception condition, the copy manager shall:

a) terminate the EXTENDED COPY command with CHECK CONDITION status; and
b) set the VALID bit in the sense data to zero. The sense key shall contain the sense key code describing the exception condition (i.e.: not COPY ABORTED).

7.2.3 Errors detected during processing of segment descriptors

Errors may occur after the copy manager has begun processing segment descriptors. These include invalid parameters in segment descriptors, invalid segment descriptors, unavailable targets referenced by target descriptors, inability of the copy manager to continue operating, and errors reported by source or destination target devices. If the copy manager receives a CHECK CONDITION status from one of the target devices, it shall recover the sense data associated with the exception condition and clear any ACA condition associated with the CHECK CONDITION status.

If processing of a segment cannot complete because the copy manager is unable to establish communications with a target device, or because the target device does not respond to INQUIRY, or because the data returned in response to INQUIRY indicates an unsupported logical unit, then the EXTENDED COPY command shall be terminated with a CHECK CONDITION status. The sense key shall be set to COPY ABORTED and the additional sense code shall be set to COPY TARGET DEVICE NOT REACHABLE.

If processing of a segment cannot complete because the data returned in response to an INQUIRY command indicates a device type that does not match the type in the target descriptor, then the EXTENDED COPY command shall be terminated with a CHECK CONDITION status. The sense key shall be set to COPY ABORTED and the additional sense code shall be set to INCORRECT COPY TARGET DEVICE TYPE.

If the copy manager has issued a command other than INQUIRY to a target device while processing an EXTENDED COPY command and the target device either fails to respond with status or responds with status other than BUSY, TASK SET FULL, ACA ACTIVE, or RESERVATION CONFLICT the condition shall be considered a target device command failure. In response to a target device command failure the EXTENDED COPY command shall be terminated with a CHECK CONDITION status. The sense key shall be set to COPY ABORTED and the additional sense code shall be set to THIRD PARTY DEVICE FAILURE.

If a target device responds to a command from the copy manager with a status of BUSY, TASK SET FULL, ACA ACTIVE, or RESERVATION CONFLICT the copy manager shall either retry the command or terminate the EXTENDED COPY command as a target device command failure.

NOTES
2 The copy manager is assumed to employ a vendor specific retry policy that minimizes time consuming and/or fruitless repetition of retries.
3 RESERVATION CONFLICT is listed only to give the copy manager leeway in multi-port cases. The copy manager may have multiple ports that are capable of reaching a target device, and there may be a third-party reservation for one of these ports. The copy manager may need to try access from multiple ports to find one with access.

If a target device responds to an input or output operation with a GOOD status but less data than expected is transferred, then the EXTENDED COPY command shall be terminated with a CHECK CONDITION status. The sense key shall be set to COPY ABORTED and additional sense code shall be set to COPY TARGET DEVICE DATA UNDERRUN. If an overrun is detected, then the EXTENDED COPY command shall be terminated with a CHECK CONDITION status. The sense key shall be set to COPY ABORTED and additional sense code shall be set to COPY TARGET DEVICE DATA OVERRUN.

Following an exception condition detected during segment descriptor processing, the copy manager shall:

a) terminate the EXTENDED COPY command with CHECK CONDITION status;
b) set the sense key code to COPY ABORTED;
c) indicate the segment that was being processed at the time of the exception by writing the segment number to third and forth bytes of the COMMAND-SPECIFIC INFORMATION field. The segment number is based on the relative position of the segment descriptor in the EXTENDED COPY parameter list. The first segment descriptor in the parameter list is assigned descriptor number zero, the second is assigned one, etc.;

d) If any data has been written to the destination for the segment being processed at the time the error occurred, the residual for the segment shall be placed in the INFORMATION field, and the VALID bit shall be set to one. The residual count shall be reported in bytes if the peripheral device type in the destination target descriptor is 03h, and in destination device blocks for all other device type codes. The residual count shall be computed by subtracting the number of bytes or blocks successfully written during the processing of the current segment from the number of bytes or blocks which would have been written if all commands had completed with GOOD status and all READ commands had returned the full data length requested. When computing the residual count, the copy manager shall include only the results of commands successfully completed by a destination device, specifically commands completed by a destination device with a GOOD status or with a CHECK CONDITION status and the EOM bit set to one in the sense data. If the copy manager has used out of order transfers the residual count shall be based solely on the contiguous successfully completed transfers starting at relative byte zero of the segment (i.e., any successfully completed transfers farther from relative byte zero than the first incomplete or unsuccessful transfer shall not contribute to the computation of the residual count). If no data has been written to the destination for the segment being processed at the time the error occurred, then the VALID bit shall be set to zero and the contents of the INFORMATION field are not defined. Segment descriptors that do not specify a transfer count shall not have a valid residual count returned;

e) If the exception condition is reported by the source logical unit, then the first byte of the COMMAND-SPECIFIC INFORMATION field shall specify the starting byte number, relative to the first byte of sense data, of an area that contains the status byte and sense data delivered to the copy manager by the source logical unit. The status byte and sense data shall not be modified by the copy manager or device server. A zero value indicates that no status byte and sense data is being returned for the source logical unit;

f) If the exception condition is reported by the destination logical unit, then the second byte of the COMMAND-SPECIFIC INFORMATION field shall specify the starting byte number, relative to the first byte of sense data, of an area that contains the status byte and sense data delivered to the copy manager by the destination logical unit. The status byte and sense data shall not be modified by the copy manager or device server. A zero value indicates that no status byte and sense data is being returned for the destination logical unit;

g) If segment processing is terminated because a target device is unreachable or as the result of a target command failure, then the SENSE-KEY SPECIFIC field shall be set as described in 7.20.3, with the FIELD POINTER field indicating the first byte of the target descriptor that identifies the target; and

h) If, during the processing of a segment descriptor, the copy manager detects an error in the segment descriptor, then the SENSE-KEY SPECIFIC field shall be set as described in 7.20.3, with the FIELD POINTER field indicating the byte in error. The FIELD POINTER field may be used to indicate an offset into either the parameter data or the segment descriptor. The SD bit is used to differentiate between these two cases. The SD bit shall be set to zero to indicate the FIELD POINTER field contains an offset from the start of the parameter data. The SD bit shall be set to one to indicate the FIELD POINTER field contains an offset from the start of the segment descriptor; and

i) The copy manager shall preserve information for the FAILED SEGMENT DETAILS service action of the RECEIVE COPY RESULTS command (see 7.14.5). The information shall be discarded as described in 7.14.5.

7.2.4 Abort task management functions

When a device server processes an ABORT TASK, ABORT TASK SET, or CLEAR TASK SET task management function that terminates an EXTENDED COPY command, the copy manager shall ensure that all commands and data transfers generated by the terminated EXENDED COPY command have been terminated and are no longer transferring data before allowing the task manager to complete the task management function. This requirement shall also apply to the processing the PREEMPT AND ABORT service action on the PERSISTENT RESERVE OUT command as described in 5.5.3.6.4.

7.2.5 Descriptor type codes

Target descriptors and segment descriptors share a single set of code values that identify the type of descriptor (see table 16). Segment descriptors use codes in the range 00h to BFh. The definitions of codes between C0h and DFh are vendor specific. Target descriptors use codes in the range E0h to FFh.

Table 16 — EXTENDED COPY descriptor type codes (part 1 of 2)

Descriptor type code	Reference	Description [a]	Shorthand [a]
00h	7.2.7.3	Copy from block device to stream device	block→stream
01h	7.2.7.4	Copy from stream device to block device	stream→block
02h	7.2.7.5	Copy from block device to block device	block→block
03h	7.2.7.6	Copy from stream device to stream device	stream→stream
04h	7.2.7.7	Copy inline data to stream device	inline→stream
05h	7.2.7.8	Copy embedded data to stream device	embedded→stream
06h	7.2.7.9	Read from stream device and discard	stream→discard
07h	7.2.7.10	Verify block or stream device operation	
08h	7.2.7.11	Copy block device with offset to stream device	block<o>→stream
09h	7.2.7.12	Copy stream device to block device with offset	stream→block<o>
0Ah	7.2.7.13	Copy block device with offset to block device with offset	block<o>→block<o>
0Bh	7.2.7.3	Copy from block device to stream device and hold a copy of processed data for the application client [b]	block→stream +application client
0Ch	7.2.7.4	Copy from stream device to block device and hold a copy of processed data for the application client [b]	stream→block +application client
0Dh	7.2.7.5	Copy from block device to block device and hold a copy of processed data for the application client [b]	block→block +application client
0Eh	7.2.7.6	Copy from stream device to stream device and hold a copy of processed data for the application client [b]	stream→stream +application client
0Fh	7.2.7.9	Read from stream device and hold a copy of processed data for the application client [b]	stream→discard +application client
10h	7.2.7.14	Write filemarks to sequential-access device	filemark→tape
11h	7.2.7.15	Space records or filemarks on sequential-access device	space→tape
12h	7.2.7.16	Locate on sequential-access device	locate→tape

[a] Block devices are those with peripheral device type codes 0h, 4h, 5h, 7h, and Eh. Stream devices are those devices with peripheral device type codes 1h and 3h. Sequential-access (indicated by "tape" in the shorthand column) devices are those with peripheral device type code 01h. See 7.3.2 for peripheral device type code definitions.

[b] The application client shall use the RECEIVE COPY RESULTS with a RECEIVE DATA service action to retrieve data held for it by the copy manager (see 7.14.3).

Table 16 — EXTENDED COPY descriptor type codes (part 2 of 2)

Descriptor type code	Reference	Description [a]	Shorthand [a]
13h	7.2.7.17	Image copy from sequential-access device to sequential-access device	<i>tape→<i>tape
14h	7.2.7.18	Register key	
15h - BFh		Reserved for segment descriptors	
C0h - DFh		Vendor unique descriptors	
E0h	7.2.6.2	Fibre Channel World Wide Name target descriptor	
E1h	7.2.6.3	Fibre Channel N_Port target descriptor	
E2h	7.2.6.4	Fibre Channel N_Port with World Wide Name checking target descriptor	
E3h	7.2.6.5	Parallel Interface T_L target descriptor	
E4h	7.2.6.6	Identification descriptor target descriptor	
E5h - FFh		Reserved for target descriptors	

[a] Block devices are those with peripheral device type codes 0h, 4h, 5h, 7h, and Eh. Stream devices are those devices with peripheral device type codes 1h and 3h. Sequential-access (indicated by "tape" in the shorthand column) devices are those with peripheral device type code 01h. See 7.3.2 for peripheral device type code definitions.

[b] The application client shall use the RECEIVE COPY RESULTS with a RECEIVE DATA service action to retrieve data held for it by the copy manager (see 7.14.3).

7.2.6 Target descriptors

7.2.6.1 Target descriptors introduction

All target descriptors are 32 bytes in length and begin with a four-byte header (see table 17) that contains the DESCRIPTOR TYPE CODE field, that identifies the format of the descriptor. The assigned values for target descriptors type codes are shown in table 16. Support for each target descriptor format is optional. If copy manager receives an unsupported descriptor type code in a target descriptor, the command shall be terminated with CHECK CONDITION status. The sense key shall be set to ILLEGAL REQUEST and the additional sense code shall be set to UNSUPPORTED TARGET DESCRIPTOR TYPE CODE.

Table 17 — Target descriptor format

Bit / Byte	7	6	5	4	3	2	1	0
0	DESCRIPTOR TYPE CODE (E0 - FFh)							
1	Reserved		NUL	PERIPHERAL DEVICE TYPE				
2	Reserved							
3	Reserved							
4	Target descriptor parameters							
27								
28	Device type specific parameters							
31								

The DESCRIPTOR TYPE CODE field is described in 7.2.5.

A null device (NUL) bit of zero indicates that the target descriptor identifies a SCSI device that is expected to respond to an INQUIRY command and to which data movement commands may be sent. A NUL bit of one

indicates that the descriptor identifies a null device that is not expected to be the recipient of any SCSI commands. If NUL is one, bytes 4-27 of the target descriptor shall be ignored. If the processing required by a segment descriptor necessitates sending a command to a target device whose target descriptor has the NUL bit set to one, then the EXTENDED COPY command shall be terminated as if an unreachable target had been encountered (see 7.2.3).

NOTE 4 Target descriptors with the NUL bit set to one are useful for processing the residual data from previous segment descriptors without affecting any media. For example, a segment descriptor of type 06h (stream device to discard) with a byte count of zero, CAT equal to zero, and a null source target descriptor with PAD equal to one may be used to discard all residual data.

The PERIPHERAL DEVICE TYPE field is described in 7.3.2. The value in the DESCRIPTOR TYPE CODE field determines the format of the target descriptor parameters that follow the four-byte header and precede the device type specific parameters. The values in the DESCRIPTOR TYPE CODE field are listed in table 16.

The value in the PERIPHERAL DEVICE TYPE field determines the format of the device type specific parameters that follow the target descriptor parameters. The device type specific parameters convey information specific to the type of device identified by the target descriptor.

Table 18 lists the peripheral device type code values having formats defined for the device type specific parameters in a target descriptor. Peripheral device types with code values not listed in table 18 are reserved in the EXTENDED COPY parameter list.

Table 18 — Device type specific parameters in target descriptors

Peripheral Device Type	Reference	Description	Shorthand
00h, 04h, 05h, 07h, and 0Eh	7.2.6.7	Block devices	Block
01h	7.2.6.8	Sequential-access devices	Stream or Tape
03h	7.2.6.9	Processor devices	Stream

The copy manager may, as part of processing a segment descriptor, verify the information in a target descriptor's device specific fields. However, when verifying the information, the copy manager shall not issue any commands that change the position of read/write media on the target without restoring it. Any errors encountered while verifying the information shall be handled as described in 7.2.3.

7.2.6.2 Fibre Channel World Wide Name target descriptor format

The target descriptor format shown in table 19 is used to identify a target using its Fibre Channel World Wide Name.

Table 19 — Fibre Channel World Wide Name target descriptor format

Bit / Byte	7	6	5	4	3	2	1	0
0	DESCRIPTOR TYPE CODE (E0h)							
1	Reserved		NUL	PERIPHERAL DEVICE TYPE				
2	Reserved							
3	Reserved							
4	(MSB)			LOGICAL UNIT NUMBER				
11								(LSB)
12	(MSB)			WORLD WIDE NAME				
19								(LSB)
20	Reserved							
27								
28	Device type specific parameters							
31								

The DESCRIPTOR TYPE CODE, PERIPHERAL DEVICE TYPE and NUL fields and the device type specific parameters are described in 7.2.6.1.

The LOGICAL UNIT NUMBER field specifies the logical unit within the SCSI device addressed by the data in the WORLD WIDE NAME field that shall be the source or destination for EXTENDED COPY operations.

The WORLD WIDE NAME field shall contain the port World Wide Name defined by the Physical Log In (PLOGI) extended link service, defined in FC-FS.

NOTE 5 The World Wide Name target descriptor format burdens the copy manager with translating the World Wide Name to an N_Port identifier (see 7.2.6.3).

7.2.6.3 Fibre Channel N_Port target descriptor format

The target descriptor format shown in table 20 is used to identify a target using its Fibre Channel N_Port.

Table 20 — Fibre Channel N_Port target descriptor format

Bit / Byte	7	6	5	4	3	2	1	0
0	DESCRIPTOR TYPE CODE (E1h)							
1	Reserved		NUL	PERIPHERAL DEVICE TYPE				
2	Reserved							
3	Reserved							
4	(MSB)			LOGICAL UNIT NUMBER				
11								(LSB)
12				Reserved				
20								
21	(MSB)			N_PORT				
22								
23								(LSB)
24				Reserved				
27								
28				Device type specific parameters				
31								

The DESCRIPTOR TYPE CODE, PERIPHERAL DEVICE TYPE and NUL fields and the device type specific parameters are described in 7.2.6.1.

The LOGICAL UNIT NUMBER field specifies the logical unit within the SCSI device addressed by the data in the N_PORT field that shall be the source or destination for EXTENDED COPY operations.

The N_PORT field shall contain the FC-FS port D_ID to be used to transport frames including PLOGI and FCP-2 related frames.

NOTE 6 Use of N_PORT addressing restricts this target descriptor format to a single fabric.

7.2.6.4 Fibre Channel N_Port with World Wide Name checking target descriptor format

Targets addressed using their Fibre Channel N_Port with World Wide Name checking are identified using the target descriptor format shown in table 21.

Table 21 — Fibre Channel N_Port with World Wide Name checking target descriptor format

Bit / Byte	7	6	5	4	3	2	1	0
0	DESCRIPTOR TYPE CODE (E2h)							
1	Reserved		NUL	PERIPHERAL DEVICE TYPE				
2	Reserved							
3	Reserved							
4	(MSB)			LOGICAL UNIT NUMBER				
11								(LSB)
12	(MSB)			WORLD WIDE NAME				
19								(LSB)
20	Reserved							
21	(MSB)			N_PORT				
22								
23								(LSB)
24	Reserved							
27								
28	Device type specific parameters							
31								

The DESCRIPTOR TYPE CODE, PERIPHERAL DEVICE TYPE and NUL fields and the device type specific parameters are described in 7.2.6.1.

The LOGICAL UNIT NUMBER field specifies the logical unit with in the SCSI device addressed by the data in the N_PORT and WORLD WIDE NAME and fields that shall be the source or destination for EXTENDED COPY operations.

The WORLD WIDE NAME field shall contain the port World Wide Name defined by the Physical Log In (PLOGI) extended link service, defined in FC-FS.

The N_PORT field shall contain the FC-FS port D_ID to be used to transport frames including PLOGI and FCP-2 related frames.

NOTE 7 Use of N_PORT addressing restricts this target descriptor format to a single fabric.

When the copy manager first processes a segment descriptor that references this target descriptor, it shall confirm that the D_ID in the N_PORT field is associated with the World Wide Name in the WORLD WIDE NAME field. If the association cannot be confirmed, the EXTENDED COPY command shall be terminated because the target is unavailable (see 7.2.3). The copy manager shall track configuration changes that affect the D_ID value for the duration of the EXTENDED COPY commands. An application client generating the EXTENDED COPY commands is responsible for tracking configuration changes between commands.

7.2.6.5 Parallel Interface T_L target descriptor format

Targets addressed using their parallel SCSI bus Target ID, and logical unit number are identified using the target descriptor format shown in table 22.

Table 22 — Parallel Interface T_L target descriptor format

Bit / Byte	7	6	5	4	3	2	1	0
0	DESCRIPTOR TYPE CODE (E3h)							
1	Reserved		NUL	PERIPHERAL DEVICE TYPE				
2	Reserved							
3	Reserved							
4	(MSB)			LOGICAL UNIT NUMBER				
11								(LSB)
12	Vendor unique							
13	TARGET IDENTIFIER							
14	Reserved							
27								
28	Device type specific parameters							
31								

The DESCRIPTOR TYPE CODE, PERIPHERAL DEVICE TYPE and NUL fields and the device type specific parameters are described in 7.2.6.1.

The LOGICAL UNIT NUMBER field specifies the logical unit with in the SCSI device addressed by the data in the TARGET IDENTIFIER field that shall be the target (source or destination) for EXTENDED COPY operations.

The TARGET IDENTIFIER field specifies the SCSI target identifier to be used when this target descriptor identifies the source or destination of an EXTENDED COPY operation.

7.2.6.6 Identification descriptor target descriptor format

The target descriptor format shown in table 23 instructs the copy manager to locate a target and logical unit that returns a device identification VPD page (see 8.4.4) containing an Identification descriptor having the specified CODE SET, ASSOCIATION, IDENTIFIER TYPE, IDENTIFIER LENGTH, and IDENTIFIER field values. The copy manager may use any N_Port, target identifier and logical unit number values that result in matching VPD field values to address the copy device. If multiple N_Port, target identifiers and logical unit number combinations access matching VPD field values, the copy manager may use any combination to address the copy device and shall try other combinations in the event that one combination becomes non-operational during the processing of an EXTENDED COPY command.

Table 23 — Identification descriptor target descriptor format

Bit / Byte	7	6	5	4	3	2	1	0
0	DESCRIPTOR TYPE CODE (E4h)							
1	Reserved			NUL	PERIPHERAL DEVICE TYPE			
2	Reserved							
3	Reserved							
4	Reserved				CODE SET			
5	Reserved		ASSOCIATION		IDENTIFIER TYPE			
6	Reserved							
7	IDENTIFIER LENGTH (n-7)							
8	(MSB) IDENTIFIER							
n								(LSB)
n+1	Reserved							
27								
28	Device type specific parameters							
31								

The DESCRIPTOR TYPE CODE, PERIPHERAL DEVICE TYPE and NUL fields and the device type specific parameters are described in 7.2.6.1.

The contents of the CODE SET, ASSOCIATION, IDENTIFIER TYPE, IDENTIFIER LENGTH, and IDENTIFIER fields are specified in 8.4.4.

The identifier length shall be 20 or less. If the identifier length is 20 there shall be no reserved bytes between the target descriptor parameters and the device type specific parameters.

Some combinations of code set, association, identifier type, identifier length and identifier do not uniquely identify a logical unit to serve as a copy target device. The application client shall not send such combinations to the copy manager.

7.2.6.7 Device type specific target descriptor parameters for block device types

The format for the device type specific target descriptor parameters for block device types (device type code values 00h, 04h, 05h, 07h, and 0Eh) is shown in table 24.

Table 24 — Device type specific target descriptor parameters for block device types

Bit / Byte	7	6	5	4	3	2	1	0
28	Reserved					PAD	Reserved	
29	(MSB)							
30				DISK BLOCK LENGTH				
31								(LSB)

The PAD bit is used in conjunction with the CAT bit (see 7.2.7.1) in the segment descriptor to determine what action should be taken when a segment of the copy does not fit exactly into an integer number of destination blocks.

The DISK BLOCK LENGTH field contains the number of bytes in a disk block for the logical device being addressed.

The copy manager may read ahead from sources of block device type. That is, the copy manager may perform read operations from a source disk at any time and in any order during processing of an EXTENDED COPY command, provided that the relative order of writes and reads on the same blocks within the same target descriptor does not differ from their order in the segment descriptor list.

7.2.6.8 Device type specific target descriptor parameters for sequential-access device types

The format for the device type specific target descriptor parameters for the sequential-access device type (device type code value 01h) is shown in table 25.

Table 25 — Device type specific target descriptor parameters for sequential-access device types

Bit / Byte	7	6	5	4	3	2	1	0
28	Reserved					PAD	Reserved	FIXED
29	(MSB)							
30				STREAM BLOCK LENGTH				
31								(LSB)

The contents of the FIXED bit and STREAM BLOCK LENGTH field are combined with the STREAM DEVICE TRANSFER LENGTH FIELD in the segment descriptor to determine the length of the stream read or write operation as specified in table 26.

Table 26 — Stream device transfer lengths

FIXED bit	STREAM BLOCK LENGTH field	Description
0	0	Use variable length reads or writes. The number of bytes for each read or write is specified by the STREAM DEVICE TRANSFER LENGTH field in the segment descriptor.
0	not 0	The command shall be terminated with a CHECK CONDITION status. The sense key shall be set to ILLEGAL REQUEST and the additional sense code shall be set to INVALID FIELD IN PARAMETER LIST.
1	0	The command shall be terminated with a CHECK CONDITION status. The sense key shall be set to ILLEGAL REQUEST and the additional sense code shall be set to INVALID FIELD IN PARAMETER LIST
1	not 0	Use fixed record length reads or writes. The number of bytes for each read or write shall be the product of the STREAM BLOCK LENGTH field and the STREAM DEVICE TRANSFER LENGTH field in the segment descriptor.

The PAD bit is used in conjunction with the CAT bit (see 7.2.7.1) in the segment descriptor to determine what action should be taken when a segment of the copy does not fit exactly into an integer number of destination blocks.

All read commands issued to sequential-access type devices shall have the SILI bit equal to zero.

NOTE 8 It is anticipated that future versions of this standard may use bit 1 of byte 28 in the device type specific target descriptor parameters for stream device types to indicate the value of the SILI bit for read commands, after T10 establishes how the copy manager is to process tape reads of unknown block length without error.

The copy manager shall not read ahead from sources of stream device type. That is, the read operations required by a segment descriptor for which the source is a stream device shall not be started until all write operations for previous segment descriptors have completed.

7.2.6.9 Device type specific target descriptor parameters for processor device types

The format for the device type specific target descriptor parameters for the processor device type (device type code value 03h) is shown in table 27.

Table 27 — Device type specific target descriptor parameters for processor device types

Bit / Byte	7	6	5	4	3	2	1	0
28	Reserved					PAD	Reserved	
29	Reserved							
31								

The PAD bit is used in conjunction with the CAT bit (see 7.2.7.1) in the segment descriptor to determine what action should be taken when a segment of the copy does not fit exactly into an integer number of SEND or RECEIVE commands.

When the processor device is a source, the number of bytes to be transferred by a SEND command shall be specified by STREAM DEVICE TRANSFER LENGTH field in the segment descriptor. When the processor device is a destination, the number of bytes to be transferred by a RECEIVE command shall be specified by STREAM DEVICE TRANSFER LENGTH field in the segment descriptor.

7.2.7 Segment Descriptors

7.2.7.1 Segment descriptors introduction

All segment descriptors begin with the eight byte header shown in table 28.

Table 28 — Segment descriptor header

Bit / Byte	7	6	5	4	3	2	1	0
0	DESCRIPTOR TYPE CODE (00h-3Fh)							
1	Reserved						DC	CAT
2	(MSB)			DESCRIPTOR LENGTH				
3								(LSB)
4	(MSB)			SOURCE TARGET DESCRIPTOR INDEX				
5								(LSB)
6	(MSB)			DESTINATION TARGET DESCRIPTOR INDEX				
7								(LSB)

The descriptor type code field is described in 7.2.5. Support for each segment descriptor format is optional. If copy manager receives an unsupported descriptor type code in a segment descriptor, the command shall be terminated with CHECK CONDITION status. The sense key shall be set to ILLEGAL REQUEST and the additional sense code shall be set to UNSUPPORTED SEGMENT DESCRIPTOR TYPE CODE.

The destination count (DC) bit is only applicable to segment descriptors with descriptor type code values of 02h and 0Dh. The DC bit is reserved for all other segment descriptors. Details of usage for the DC bit appear in the subclauses defining the segment descriptors that use it.

The CAT bit is described in 7.2.7.2.

The DESCRIPTOR LENGTH field contains the length in bytes of the fields that follow the DESCRIPTOR LENGTH field in the segment descriptor. In most cases, the length is constant.

The SOURCE TARGET DESCRIPTOR INDEX field contains an index into the target descriptor list (see 7.2.1) identifying the source target device. The DESTINATION TARGET DESCRIPTOR INDEX field contains an index into the target descriptor list (see 7.2.1) identifying the destination target device. Some segment descriptor formats do not require a SOURCE TARGET DESCRIPTOR INDEX field or a DESTINATION TARGET DESCRIPTOR INDEX field, in which case the field is reserved.

If the target identified by a SOURCE TARGET DESCRIPTOR INDEX field or a DESTINATION TARGET DESCRIPTOR INDEX field is not accessible to the copy manager, then the command shall be terminated with a CHECK CONDITION status, the sense key shall be set to COPY ABORTED and the additional sense code shall be set to UNREACHABLE COPY TARGET.

7.2.7.2 Segment descriptor processing

In processing a segment descriptor, the copy manager may be required:

a) To read source data by issuing data input commands to the source device;
b) To process data, an operation that generally designates data as destination data intended for transfer to the destination device; and
c) To write some or all of the destination data to the destination device.

The number of blocks to read and write, the number of bytes to process, and the nature of processing are determined by the segment descriptor type code, the parameters of the segment descriptor, and the amount of residual source or destination data retained from the previous segment, if any.

Except as otherwise specified by particular segment descriptor type codes:

a) Just enough whole-block read operations shall be performed to supply, together with residual source data from the previous segment or segments, the number of bytes to be processed;
b) Processing consists of removing bytes from the source data and designating them as destination data, without change.
c) As many whole-block write operations as possible shall be performed with the destination data, including any residual destination data from the previous segment or segments.

Any residual source data from the previous segment or segments shall be processed before any data read from the source device during processing of the current segment descriptor. Any residual destination data from the previous segment or segments shall be written before any data processed during processing of the current segment descriptor.

Exceptions and clarifications to these general rules are described in table 29 and the subclauses it references.

Table 29 — Descriptor Type Code Dependent Copy Manager Processing (part 1 of 2)

Segment Descriptor Type Code	Reference	Description
00h (block→stream) or 0Bh (block→stream+application client)	7.2.7.3	The number of bytes processed is determined by the BLOCK DEVICE NUMBER OF BLOCKS field for the source blocks (see applicable type code definition subclauses for details). [a]
02h (block→block) or 0Dh (block→block+application client) with DC=0	7.2.7.5	
02h (block→block) or 0Dh (block→block+application client) with DC=1	7.2.7.5	The number of blocks or byte range specified shall be output to the destination device. If residual destination data is sufficient to perform the output then no data shall be processed. Otherwise, just as much data as needed shall be processed (which may involve reading data from the source device) so that the destination data (which includes any residual destination data from the previous segment) is sufficient. [a]
01h (stream→block) or 0Ch (stream→block+application client)	7.2.7.3	
09h (stream→block<o>)	7.2.7.12	
03h (stream→stream) or 0Eh (stream→stream+application client)	7.2.7.6	The number of bytes specified in the segment descriptor shall be processed. [a]
04h (inline→stream)	7.2.7.7	The specified number of bytes of inline or embedded data shall be appended to the destination data, and no source data shall be processed.
05h (embedded→stream)	7.2.7.8	
06h (stream→discard)	7.2.7.9	The specified number of bytes shall be removed from the source data and discarded.

[a] For segment descriptor type codes 0Bh, 0Ch, 0Dh and 0Eh, a copy of the processed data shall also be held for retrieval by the application client.

Table 29 — Descriptor Type Code Dependent Copy Manager Processing (part 2 of 2)

Segment Descriptor Type Code	Reference	Description
07h (verify device operation)	7.2.7.10	No data shall be processed and no read or write operations shall be performed on target devices. Residual source or destination data, if any, shall be retained or discarded as if the CAT bit were equal to one.
10h (filemark→tape)	7.2.7.14	
11h (space→tape)	7.2.7.15	
12h (locate→tape)	7.2.7.16	
14h (register key)	7.2.7.18	
08h (block<o>→stream)	7.2.7.11	The required blocks shall be read from the source device, the designated byte range shall be extracted as source data, and the designated number of bytes (starting with residual source data, if any) shall be processed.
0Ah (block<o>→block<o>)	7.2.7.13	The source byte range specified shall be read into source data, the number of bytes specified shall be moved from source data to destination data, and the specified destination byte range shall be written using destination data.
0Fh (stream→discard+application client)	7.2.7.9	The specified number of bytes shall be removed from the source data and held for retrieval by the application client.
13h (<i>tape→<i>tape)	7.2.7.17	The data movement shall not involve "processing" as described in this subclause. Residual source or destination data, if any, shall not be used and shall be retained or discarded as if the CAT bit were equal to one.

[a] For segment descriptor type codes 0Bh, 0Ch, 0Dh and 0Eh, a copy of the processed data shall also be held for retrieval by the application client.

Reads and writes shall be performed using whole-block transfer lengths determined by the block size, transfer length, or both. Therefore some source data may remain unprocessed and some destination data may not have been transferred at the end of a segment. If so, the residue shall be handled according to the CAT bit in the segment descriptor and the PAD bits of the source and destination target descriptors, as defined in table 30.

Table 30 — PAD and CAT bit definitions

PAD bit in		CAT bit	Copy manager action
Source target descriptor	Destination target descriptor		
0 or 1	0 or 1	1	Any residual source data shall be retained as source data for a subsequent segment descriptor. Any residual destination data shall be retained as destination data for a subsequent segment descriptor. It shall not be an error if either the source or destination target index in the following segment descriptor does not match the corresponding target index with which residual data was originally associated. If the CAT bit is one on the last segment of an EXTENDED COPY command any residual data shall be discarded; this shall not be considered an error.
1	1	0	Any residual source data shall be discarded. Any residual destination data shall be padded with zeroes to make a whole block transfer. [a]
0	1	0	Any residual source data shall be handled as if the CAT bit is equal to one (i.e., discarded on the last segment and retained otherwise). Any residual destination data shall be padded with zeroes to make a whole block transfer. [a]
1	0	0	Any residual source or destination data shall be discarded.
0	0	0	If there is residual source or destination data the EXTENDED COPY command shall be terminated with a CHECK CONDITION status. The sense key shall be set to an COPY ABORTED and the additional sense code shall be set to UNEXPECTED INEXACT SEGMENT.

[a] When the CAT bit is set to zero and the destination target descriptor has the PAD bit set to one, the EXTENDED COPY command shall be terminated with a CHECK CONDITION status, the sense key shall be set to COPY ABORTED, and the additional sense code shall be set to UNEXPECTED INEXACT SEGMENT if any of the following conditions are met:

a) If any residual destination data is present after writing the designated byte range for a segment descriptor of type 09h (stream→block <o>) or 0Ah (block<o>→block<o>); or

b) If any residual destination data is present after the designated number of blocks have been written for a segment descriptor of type 02h (block→block) with DC set to one, 0Dh (block→block+application client) with DC set to one, 01h (stream→block) or 0Ch (stream→block+application client).

A few segment descriptors have either no source or no target and handling of the PAD bit for those descriptors shall be as follows. For segment descriptor types 04h (inline→stream, see 7.2.7.7) and 05h (embedded→stream, see 7.2.7.8), the handling shall be as if the PAD were equal to zero for the source target descriptor. For segment descriptor types 06h and 0Fh (stream→discard and stream→discard+application client, see 7.2.7.9), handling shall be as if the PAD were equal to zero for the destination target descriptor.

7.2.7.3 Block device to stream device operations

The segment descriptor format shown in table 31 is used by the copy operations that move data from a block device to a stream device or vice versa.

Table 31 — Block device to or from stream device segment descriptor

Bit / Byte	7	6	5	4	3	2	1	0
0	DESCRIPTOR TYPE CODE (00h, 01h, 0Bh, or 0Ch)							
1	Reserved							CAT
2	(MSB)			DESCRIPTOR LENGTH (0014h)				
3								(LSB)
4	(MSB)			SOURCE TARGET DESCRIPTOR INDEX				
5								(LSB)
6	(MSB)			DESTINATION TARGET DESCRIPTOR INDEX				
7								(LSB)
8	Reserved							
9	(MSB)			STREAM DEVICE TRANSFER LENGTH				
10								
11								(LSB)
12	Reserved							
13	Reserved							
14	(MSB)			BLOCK DEVICE NUMBER OF BLOCKS				
15								(LSB)
16	(MSB)			BLOCK DEVICE LOGICAL BLOCK ADDRESS				
23								(LSB)

The DESCRIPTOR TYPE CODE field is described in 7.2.5 and 7.2.7.1. Two DESCRIPTOR TYPE CODE values use the segment descriptor format shown in table 31 and described in this subclause.

For descriptor type code 00h (block→stream) or descriptor type code 0Bh (block→stream+application client), the copy manager shall copy the data from the source block device identified by the SOURCE TARGET DESCRIPTOR INDEX field to the destination stream device identified by the DESTINATION TARGET DESCRIPTOR INDEX field using the logical blocks starting at the location identified by the BLOCK DEVICE LOGICAL BLOCK ADDRESS field. As many blocks shall be read as necessary to process (see 7.2.7.2) a number of bytes equal to the contents of the DISK BLOCK LENGTH field in the target descriptor for the source device times the contents of the BLOCK DEVICE NUMBER OF BLOCKS field. The data shall be written to the stream device starting at the current position of the media.

For descriptor type code 0Bh (block→stream+application client), the copy manager also shall hold a copy of the processed data for delivery to the application client upon completion of the EXTENDED COPY command in response to a RECEIVE COPY RESULTS command with RECEIVE DATA service action as described in 7.14.3. The minimum amount of held data supported by the copy manager is returned in the response data for the RECEIVE COPY RESULTS command with OPERATING PARAMETERS service action (see 7.14.4). If the copy manager supports the 0Bh descriptor type code it also shall support the RECEIVE COPY RESULTS command with RECEIVE DATA service action.

The CAT bit is described in 7.2.7.2.

The DESCRIPTOR LENGTH field shall contain 20 (0014h). The SOURCE TARGET DESCRIPTOR INDEX and DESTINATION TARGET DESCRIPTOR INDEX fields are described in 7.2.7.1.

The STREAM DEVICE TRANSFER LENGTH field specifies the amount of data to be written on each write operation to the stream device. See 7.2.6.8 for a description of how data in the STREAM DEVICE TRANSFER LENGTH field in the segment descriptor interacts with data in the STREAM BLOCK LENGTH field in the device type specific target descriptor parameters for the sequential-access device type.

The BLOCK DEVICE NUMBER OF BLOCKS field specifies the length, in source logical blocks, of data to be processed (see 7.2.7.2) in the segment. A value of zero shall not be considered as an error. No data shall be processed, but any residual destination data retained from a previous segment shall be written if possible to the destination in whole-block transfers. A value of zero shall not modify the handling of residual data.

The BLOCK DEVICE LOGICAL BLOCK ADDRESS field specifies the starting logical block address on the block device for this segment.

7.2.7.4 Stream device to block device operations

The segment descriptor format shown in table 31 (see 7.2.7.3) also is used by the copy operations that move data from a stream device to a block device. Two DESCRIPTOR TYPE CODE values use the segment descriptor format shown in table 31 and described in this subclause.

For descriptor type code 01h (stream→block) or descriptor type code 0Ch (stream→block+application client), the copy manager shall copy the data from the source stream device identified by the SOURCE TARGET DESCRIPTOR INDEX field to the destination block device identified by the DESTINATION TARGET DESCRIPTOR INDEX field using the stream data starting at the current position of the stream device. The data shall be written to logical blocks starting at the location identified by the BLOCK DEVICE LOGICAL BLOCK ADDRESS field and continuing for the number of blocks specified in the BLOCK DEVICE NUMBER OF BLOCKS field.

For descriptor type code 0Ch (stream→block+application client), the copy manager also shall hold a copy of the processed data for delivery to the application client upon completion of the EXTENDED COPY command in response to a RECEIVE COPY RESULTS command with RECEIVE DATA service action as described in 7.14.3. The minimum amount of held data supported by the copy manager is returned in the response data for the RECEIVE COPY RESULTS command with OPERATING PARAMETERS service action (see 7.14.4). If the copy manager supports the 0Ch descriptor type code it also shall support the RECEIVE COPY RESULTS command with RECEIVE DATA service action.

The CAT bit is described in 7.2.7.2.

The DESCRIPTOR LENGTH field shall contain 20 (0014h). The SOURCE TARGET DESCRIPTOR INDEX and DESTINATION TARGET DESCRIPTOR INDEX fields are described in 7.2.7.1.

The STREAM DEVICE TRANSFER LENGTH field specifies the amount of data to be read from the source stream device on each read operation. See 7.2.6.8 for a description of how data in the STREAM DEVICE TRANSFER LENGTH field in the segment descriptor interacts with data in the STREAM BLOCK LENGTH field in the device type specific target descriptor parameters for the sequential-access device type.

The BLOCK DEVICE NUMBER OF BLOCKS field specifies the number blocks to be written by the segment. A value of zero indicates that no blocks shall be written in this segment. This shall not be considered as an error.

The BLOCK DEVICE LOGICAL BLOCK ADDRESS field specifies the starting logical block address on the block device for this segment.

7.2.7.5 Block device to block device operations

The segment descriptor format shown in table 32 is used by the copy operations that move data from a block device to a block device.

Table 32 — Block device to block device segment descriptor

Bit / Byte	7	6	5	4	3	2	1	0
0	DESCRIPTOR TYPE CODE (02h or 0Dh)							
1	Reserved						DC	CAT
2	(MSB)			DESCRIPTOR LENGTH (0018h)				
3								(LSB)
4	(MSB)			SOURCE TARGET DESCRIPTOR INDEX				
5								(LSB)
6	(MSB)			DESTINATION TARGET DESCRIPTOR INDEX				
7								(LSB)
8	Reserved							
9	Reserved							
10	(MSB)			BLOCK DEVICE NUMBER OF BLOCKS				
11								(LSB)
12	(MSB)			SOURCE BLOCK DEVICE LOGICAL BLOCK ADDRESS				
19								(LSB)
20	(MSB)			DESTINATION BLOCK DEVICE LOGICAL BLOCK ADDRESS				
27								(LSB)

The DESCRIPTOR TYPE CODE field is described in 7.2.5 and 7.2.7.1. Two DESCRIPTOR TYPE CODE values use the segment descriptor format shown in table 32 and described in this subclause.

For descriptor type code 02h (block→block) or descriptor type code 0Dh (block→block+application client), the copy manager shall copy the data from the source block device identified by the SOURCE TARGET DESCRIPTOR INDEX field to the destination block device identified by the DESTINATION TARGET DESCRIPTOR INDEX field using the logical blocks starting at the location identified by the SOURCE BLOCK DEVICE LOGICAL BLOCK ADDRESS field. The data shall be written to logical blocks starting at the location identified by the DESTINATION BLOCK DEVICE LOGICAL BLOCK ADDRESS field.

If the DC bit equals zero, as many blocks shall be read as necessary to process (see 7.2.7.2) a number of bytes equal to the contents of the DISK BLOCK LENGTH field in the target descriptor for the source device times the contents of the BLOCK DEVICE NUMBER OF BLOCKS field, and as many writes as possible shall be performed using any residual destination data from the previous segment and the data processed in this segment. If the DC bit equals one, the number of blocks specified by the BLOCK DEVICE NUMBER OF BLOCKS field shall be written to the destination block device, as many bytes shall be processed as necessary for these writes to be performed, and as many blocks shall be read as necessary to supply the data to be processed.

For descriptor type code 0Dh (block→block+application client), the copy manager also shall hold a copy of the processed data for delivery to the application client upon completion of the EXTENDED COPY command in response to a RECEIVE COPY RESULTS command with RECEIVE DATA service action as described in 7.14.3. The minimum amount of held data supported by the copy manager is returned in the response data for the RECEIVE COPY RESULTS command with OPERATING PARAMETERS service action (see 7.14.4). If the copy manager supports the 0Dh descriptor type code it also shall support the RECEIVE COPY RESULTS command with RECEIVE DATA service action.

The CAT bit is described in 7.2.7.2.

The destination count (DC) bit indicates whether the BLOCK DEVICE NUMBER OF BLOCKS field refers to the source or destination device. A DC bit of zero indicates that the BLOCK DEVICE NUMBER OF BLOCKS field refers to the source device. A DC bit of one indicates that the BLOCK DEVICE NUMBER OF BLOCKS field refers to the destination device.

The DESCRIPTOR LENGTH field shall contain 24 (0018h). The SOURCE TARGET DESCRIPTOR INDEX and DESTINATION TARGET DESCRIPTOR INDEX fields are described in 7.2.7.1.

The BLOCK DEVICE NUMBER OF BLOCKS field specifies the number of blocks to be processed (if DC is set to zero) or to be written to the destination device (if DC is set to one). A value of zero shall not be considered as an error. If the DC bit equals one, a value of zero indicates that no destination blocks shall be written and the only processing to be performed is that any residual source or destination data from the previous segment shall be handled as residual data as described in 7.2.7.2. If the DC bit equals zero, a value of zero indicates that no source blocks shall be read and no source data shall be processed, but any residual destination data from a previous segment shall be written if possible to the destination in whole-block transfers, and any residual data shall be handled as described in 7.2.7.2.

The SOURCE BLOCK DEVICE LOGICAL BLOCK ADDRESS field specifies the logical block address from which the reading of data shall start.

The DESTINATION BLOCK DEVICE LOGICAL BLOCK ADDRESS field specifies the logical block address to which the writing of data shall begin.

7.2.7.6 Stream device to stream device operations

The segment descriptor format shown in table 33 is used by the copy operations that move data from a stream device to a stream device.

Table 33 — Stream device to stream device segment descriptor

Bit / Byte	7	6	5	4	3	2	1	0
0	DESCRIPTOR TYPE CODE (03h or 0Eh)							
1	Reserved							CAT
2	(MSB)							
3	DESCRIPTOR LENGTH (0010h)							(LSB)
4	(MSB)							
5	SOURCE TARGET DESCRIPTOR INDEX							(LSB)
6	(MSB)							
7	DESTINATION TARGET DESCRIPTOR INDEX							(LSB)
8	Reserved							
9	(MSB)							
10	SOURCE STREAM DEVICE TRANSFER LENGTH							
11								(LSB)
12	Reserved							
13	(MSB)							
14	DESTINATION STREAM DEVICE TRANSFER LENGTH							
15								(LSB)
16	(MSB)							
19	BYTE COUNT							(LSB)

The DESCRIPTOR TYPE CODE field is described in 7.2.5 and 7.2.7.1. Two DESCRIPTOR TYPE CODE values use the segment descriptor format shown in table 33 and described in this subclause.

For descriptor type code 03h (stream→stream) or descriptor type code 0Eh (stream→stream+application client), the copy manager shall copy the data from the source stream device identified by the SOURCE TARGET DESCRIPTOR INDEX field to the destination stream device identified by the DESTINATION TARGET DESCRIPTOR INDEX field. Data shall be read from the source stream device starting at the current position of the source stream device. Data shall be written to the destination stream device starting at the current position of the destination stream device. The BYTE COUNT field defines the number of bytes to be processed (see 7.2.7.2) by the copy manager. The copy manager shall perform read operations as necessary to supply the source data, and as many write operations as possible using the destination data.

For descriptor type code 0Eh (stream→stream+application client), the copy manager also shall hold a copy of the processed data for delivery to the application client upon completion of the EXTENDED COPY command in response to a RECEIVE COPY RESULTS command with RECEIVE DATA service action as described in 7.14.3. The minimum amount of held data supported by the copy manager is returned in the response data for the RECEIVE COPY RESULTS command with OPERATING PARAMETERS service action (see 7.14.4). If the copy manager supports the 0Eh descriptor type code it also shall support the RECEIVE COPY RESULTS command with RECEIVE DATA service action.

The CAT bit is described in 7.2.7.2.

The DESCRIPTOR LENGTH field shall contain 16 (0010h). The SOURCE TARGET DESCRIPTOR INDEX and DESTINATION TARGET DESCRIPTOR INDEX fields are described in 7.2.7.1.

The SOURCE STREAM DEVICE TRANSFER LENGTH field specifies the amount of data to be read from the source stream device on each read operation. See 7.2.6.8 for a description of how data in the SOURCE STREAM DEVICE TRANSFER LENGTH field in the segment descriptor interacts with data in the STREAM BLOCK LENGTH field in the device type specific target descriptor parameters for the source sequential-access device type.

The DESTINATION STREAM DEVICE TRANSFER LENGTH field specifies the amount of data to be written to the destination stream device on each write operation. See 7.2.6.8 for a description of how data in the DESTINATION STREAM DEVICE TRANSFER LENGTH field in the segment descriptor interacts with data in the STREAM BLOCK LENGTH field in the device type specific target descriptor parameters for the destination sequential-access device type.

The BYTE COUNT field specifies the number of bytes that shall be processed for this segment descriptor. A value of zero shall not be considered as an error, and shall specify that no source blocks shall be read and no source data shall be processed. However, a value of zero shall specify that any residual destination data from a previous segment shall be written if possible to the destination in whole-block transfers, and any residual data shall be handled as described in 7.2.7.2.

7.2.7.7 Inline data to stream device operation

The segment descriptor format shown in table 34 instructs the copy manager to write inline data from the EXTENDED COPY parameter list to a stream device.

Table 34 — Inline data to stream device segment descriptor

Bit / Byte	7	6	5	4	3	2	1	0
0	DESCRIPTOR TYPE CODE (04h)							
1	Reserved							CAT
2	(MSB)			DESCRIPTOR LENGTH (0010h)				
3								(LSB)
4	Reserved							
5	Reserved							
6	(MSB)			DESTINATION TARGET DESCRIPTOR INDEX				
7								(LSB)
8	Reserved							
9	(MSB)			STREAM DEVICE TRANSFER LENGTH				
10								
11								(LSB)
12	(MSB)			INLINE DATA OFFSET				
15								(LSB)
16	(MSB)			INLINE DATA NUMBER OF BYTES				
19								(LSB)

The DESCRIPTOR TYPE CODE field is described in 7.2.5 and 7.2.7.1. Descriptor type code 04h (inline→stream) instructs the copy manager to write inline data from the EXTENDED COPY parameter list to a stream device. The inline data shall be read from the optional inline data at the end of the EXTENDED COPY parameter list. The data shall be written to the destination stream device identified by the DESTINATION TARGET DESCRIPTOR INDEX field starting at the current position of the stream device. Any residual destination data from a previous segment descriptor shall be written before the data of the current segment descriptor. Any residual source data from a previous segment descriptor shall not be processed (see 7.2.7.2), and shall be handled as residual source data.

The CAT bit is described in 7.2.7.2.

The DESCRIPTOR LENGTH field shall contain 16 (0010h). The DESTINATION TARGET DESCRIPTOR INDEX field is described in 7.2.7.1.

The STREAM DEVICE TRANSFER LENGTH field specifies the amount of data to be written to the stream device on each write operation. See 7.2.6.8 for a description of how data in the STREAM DEVICE TRANSFER LENGTH field in the segment descriptor interacts with data in the STREAM BLOCK LENGTH field in the device type specific target descriptor parameters for the destination sequential-access device type.

The value in the INLINE DATA OFFSET field is added to the location of the first byte of inline data in the EXTENDED COPY parameter list (see table 15) to locate the first byte of inline data to be written to the stream device. The INLINE DATA OFFSET value shall be a multiple of 4.

The INLINE DATA NUMBER OF BYTES field specifies the number of bytes of inline data that are to be transferred to the stream device. A value of zero shall not be considered an error.

If the sum of the INLINE DATA OFFSET and the INLINE DATA NUMBER OF BYTES values exceeds the value in the INLINE DATA LENGTH field (see table 15), the copy manager shall terminate the command with a CHECK CONDITION

status. The sense key shall be set to COPY ABORTED and the additional sense code shall be set to INLINE DATA LENGTH EXCEEDED.

7.2.7.8 Embedded data to stream device operation

The segment descriptor format shown in table 35 instructs the copy manager to write embedded data from the segment descriptor to a stream device.

Table 35 — Embedded data to stream device segment descriptor

Bit / Byte	7	6	5	4	3	2	1	0
0	DESCRIPTOR TYPE CODE (05h)							
1	Reserved							CAT
2	(MSB)			DESCRIPTOR LENGTH (n-4)				
3								(LSB)
4	Reserved							
5	Reserved							
6	(MSB)			DESTINATION TARGET DESCRIPTOR INDEX				
7								(LSB)
8	Reserved							
9	(MSB)			STREAM DEVICE TRANSFER LENGTH				
10								
11								(LSB)
12	(MSB)			EMBEDDED DATA NUMBER OF BYTES				
13								(LSB)
14				Reserved				
15								
16				EMBEDDED DATA				
n								

The DESCRIPTOR TYPE CODE field is described in 7.2.5 and 7.2.7.1. Descriptor type code 05h (embedded→stream) instructs the copy manager to write embedded data from the segment descriptor to a stream device. The embedded data shall be read from the segment descriptor. The data shall be written to the destination stream device identified by the DESTINATION TARGET DESCRIPTOR INDEX field starting at the current position of the stream device. Any residual destination data from a previous segment descriptor shall be written before the data of the current segment descriptor. Any residual source data from a previous segment descriptor shall not be processed (see 7.2.7.2), and shall be handled as residual source data.

The CAT bit is described in 7.2.7.2.

The DESCRIPTOR LENGTH field shall contain the length in bytes of the fields that follow the DESCRIPTOR LENGTH field, including the embedded data. The value in the DESCRIPTOR LENGTH field shall be a multiple of 4.

The DESTINATION TARGET DESCRIPTOR INDEX field is described in 7.2.7.1.

The STREAM DEVICE TRANSFER LENGTH field specifies the amount of data to be written to the stream device on each write operation. See 7.2.6.8 for a description of how data in the STREAM DEVICE TRANSFER LENGTH field in the segment descriptor interacts with data in the STREAM BLOCK LENGTH field in the device type specific target descriptor parameters for the destination sequential-access device type.

The EMBEDDED DATA NUMBER OF BYTES field specifies the number of bytes of embedded data that are to be transferred to the stream device. A value of zero shall not be considered an error. The EMBEDDED DATA NUMBER OF BYTES value shall be less than or equal to the DESCRIPTOR LENGTH value minus 12.

7.2.7.9 Stream device to discard operation

The segment descriptor format shown in table 36 instructs the copy manager to read data from a stream device and not copy it to any destination device.

Table 36 — Stream device to discard segment descriptor

Bit / Byte	7	6	5	4	3	2	1	0
0	DESCRIPTOR TYPE CODE (06h or 0Fh)							
1	Reserved							CAT
2	(MSB)			DESCRIPTOR LENGTH (000Ch)				
3								(LSB)
4	(MSB)			SOURCE TARGET DESCRIPTOR INDEX				
5								(LSB)
6	Reserved							
7	Reserved							
8	Reserved							
9	(MSB)			STREAM DEVICE TRANSFER LENGTH				
10								
11								(LSB)
12	(MSB)			NUMBER OF BYTES				
15								(LSB)

The DESCRIPTOR TYPE CODE field is described in 7.2.5 and 7.2.7.1. Two DESCRIPTOR TYPE CODE values use the segment descriptor format shown in table 36 and described in this subclause.

For descriptor type code 06h (stream→discard) or descriptor type code 0Fh (stream→discard+application client), the copy manager shall read data as necessary from the source stream device identified by the SOURCE TARGET DESCRIPTOR INDEX field starting at the current position of the source stream device. The number of bytes indicated by the NUMBER OF BYTES field shall be removed from the source data, starting with any residual source data from the previous segment.

For descriptor type code 06h (stream→discard) the removed data shall be discarded and not written to any destination device. For descriptor type code 0Fh (stream→discard+application client) the removed data shall be held for delivery to the application client upon completion of the EXTENDED COPY command in response to a RECEIVE COPY RESULTS command with RECEIVE DATA service action as described in 7.14.3. The minimum amount of held data supported by the copy manager is returned in the response data for the RECEIVE COPY RESULTS command with OPERATING PARAMETERS service action (see 7.14.4). If the copy manager supports the 0Fh (stream→discard+application client) descriptor type code it also shall support the RECEIVE COPY RESULTS command with RECEIVE DATA service action.

The CAT bit is described in 7.2.7.2.

The DESCRIPTOR LENGTH field shall contain 12 (000Ch). The DESTINATION TARGET DESCRIPTOR INDEX field is described in 7.2.7.1.

The SOURCE STREAM DEVICE TRANSFER LENGTH field specifies the amount of data to be read from the source stream device on each read operation. See 7.2.6.8 for a description of how data in the SOURCE STREAM DEVICE TRANSFER

LENGTH field in the segment descriptor interacts with data in the STREAM BLOCK LENGTH field in the device type specific target descriptor parameters for the source sequential-access device type.

The NUMBER OF BYTES field specifies the number of bytes to be removed from the source data.

7.2.7.10 Verify device operation

The segment descriptor format shown in table 37 instructs the copy manager to verify the accessibility of a SCSI device.

Table 37 — Verify device operation segment descriptor

Bit / Byte	7	6	5	4	3	2	1	0
0	DESCRIPTOR TYPE CODE (07h)							
1	Reserved							
2	(MSB)			DESCRIPTOR LENGTH (0008h)				
3								(LSB)
4	(MSB)			SOURCE TARGET DESCRIPTOR INDEX				
5								(LSB)
6				Reserved				
7								
8	Reserved							TUR
9	Reserved							
11								

The DESCRIPTOR TYPE CODE field is described in 7.2.5 and 7.2.7.1. Descriptor type code 07h instructs the copy manager to verify the accessibility of the device identified by the SOURCE TARGET DESCRIPTOR INDEX field.

The DESCRIPTOR LENGTH field shall contain 8 (0008h). The SOURCE TARGET DESCRIPTOR INDEX field is described in 7.2.7.1.

Support for a value of one in the TUR (Test Unit Ready) bit is optional. If a TUR value of one is supported and the TUR bit contains one, then a TEST UNIT READY command (see 7.25) shall be used to determine the readiness of the device. If a TUR value of one is not supported and the TUR bit contains one, then the EXTENDED COPY command shall be terminated with a CHECK CONDITION status. The sense key shall be set to COPY ABORTED and the additional sense code shall be set to INVALID FIELD IN PARAMETER LIST. The SENSE-KEY SPECIFIC field shall be set as described in 7.2.3. If the TUR bit contains zero, then the accessibility should be verified without disturbing established unit attention or ACA conditions, for example, using the INQUIRY command (see 7.3).

7.2.7.11 Block device with offset to stream device operation

The segment descriptor format shown in table 38 is used to instruct the copy manager to move data from a block device with a byte offset to a stream device or vice versa.

Table 38 — Block device with offset to or from stream device segment descriptor

Bit / Byte	7	6	5	4	3	2	1	0
0	DESCRIPTOR TYPE CODE (08h or 09h)							
1	Reserved							CAT
2	(MSB)			DESCRIPTOR LENGTH (0018h)				
3								(LSB)
4	(MSB)			SOURCE TARGET DESCRIPTOR INDEX				
5								(LSB)
6	(MSB)			DESTINATION TARGET DESCRIPTOR INDEX				
7								(LSB)
8	Reserved							
9	(MSB)			STREAM DEVICE TRANSFER LENGTH				
10								
11								(LSB)
12	(MSB)			NUMBER OF BYTES				
15								(LSB)
16	(MSB)			BLOCK DEVICE LOGICAL BLOCK ADDRESS				
23								(LSB)
24	Reserved							
25	Reserved							
26	(MSB)			BLOCK DEVICE BYTE OFFSET				
27								(LSB)

The DESCRIPTOR TYPE CODE field is described in 7.2.5 and 7.2.7.1. Descriptor type code 08h (block<o>→stream) instructs the copy manager to copy the data from the source block device identified by the SOURCE TARGET DESCRIPTOR INDEX field to the destination stream device identified by the DESTINATION TARGET DESCRIPTOR INDEX field using data starting at the location identified by the BLOCK DEVICE BYTE OFFSET field in the logical block identified by the BLOCK DEVICE LOGICAL BLOCK ADDRESS field and continuing for the number of bytes specified in the NUMBER OF BYTES field. The data shall be written to the stream device starting at the current position of the media.

The CAT bit is described in 7.2.7.2.

The DESCRIPTOR LENGTH field shall contain 24 (0018h). The SOURCE TARGET DESCRIPTOR INDEX and DESTINATION TARGET DESCRIPTOR INDEX fields are described in 7.2.7.1.

The STREAM DEVICE TRANSFER LENGTH field specifies the amount of data to be written on each write operation to the stream device. See 7.2.6.8 for a description of how data in the STREAM DEVICE TRANSFER LENGTH field in the segment descriptor interacts with data in the STREAM BLOCK LENGTH field in the device type specific target descriptor parameters for the sequential-access device type.

The NUMBER OF BYTES field specifies the number bytes to be read. A value of zero indicates that no bytes shall be transferred in this segment. This shall not be considered as an error.

The BLOCK DEVICE LOGICAL BLOCK ADDRESS field specifies the starting logical block address on the source block device for this segment.

The BLOCK DEVICE BYTE OFFSET field specifies the offset into the first source block at which to begin reading bytes.

7.2.7.12 Stream device to block device with offset operation

The segment descriptor format shown in table 38 (see 7.2.7.11) also is used to instruct the copy manager to move data from a stream device to a block device with a byte offset.

The DESCRIPTOR TYPE CODE field is described in 7.2.5 and 7.2.7.1. Descriptor type code 09h (stream→block<o>) instructs the copy manager to copy the data from the source stream device identified by the SOURCE TARGET DESCRIPTOR INDEX field to the destination block device identified by the DESTINATION TARGET DESCRIPTOR INDEX field using the stream data starting at the current position of the stream device. The data shall be written starting at the location identified by the BLOCK DEVICE BYTE OFFSET field in the logical block identified by the BLOCK DEVICE LOGICAL BLOCK ADDRESS field and continuing for the number of bytes specified in the NUMBER OF BYTES field.

The content of the starting logical block on the destination device before the starting offset shall be preserved. The content on the ending logical block beyond the end of the transfer shall be preserved. The copy manager may implement this operation by reading the starting and ending logical blocks, modifying a portion of the blocks as required, and writing the full blocks to the destination device.

The CAT bit is described in 7.2.7.2.

The DESCRIPTOR LENGTH field shall contain 24 (0018h). The SOURCE TARGET DESCRIPTOR INDEX and DESTINATION TARGET DESCRIPTOR INDEX fields are described in 7.2.7.1.

The STREAM DEVICE TRANSFER LENGTH field specifies the amount of data to be written on each write operation to the stream device. See 7.2.6.8 for a description of how data in the STREAM DEVICE TRANSFER LENGTH field in the segment descriptor interacts with data in the STREAM BLOCK LENGTH field in the device type specific target descriptor parameters for the sequential-access device type.

The NUMBER OF BYTES field specifies the number bytes to be read. A value of zero indicates that no bytes shall be transferred in this segment. This shall not be considered as an error.

The BLOCK DEVICE LOGICAL BLOCK ADDRESS field specifies the starting logical block address on the destination block device for this segment.

The BLOCK DEVICE BYTE OFFSET field is the offset into the first destination block at which to begin writing data to the destination block device.

7.2.7.13 Block device with offset to block device with offset operation

The segment descriptor format shown in table 39 instructs the copy manager to move data from a block device with a byte offset to a block device with a byte offset.

Table 39 — Block device with offset to block device with offset segment descriptor

Bit Byte	7	6	5	4	3	2	1	0
0	DESCRIPTOR TYPE CODE (0Ah)							
1	Reserved							CAT
2	(MSB)			DESCRIPTOR LENGTH (001Ch)				
3								(LSB)
4	(MSB)			SOURCE TARGET DESCRIPTOR INDEX				
5								(LSB)
6	(MSB)			DESTINATION TARGET DESCRIPTOR INDEX				
7								(LSB)
8	(MSB)			NUMBER OF BYTES				
11								(LSB)
12	(MSB)			SOURCE BLOCK DEVICE LOGICAL BLOCK ADDRESS				
19								(LSB)
20	(MSB)			DESTINATION BLOCK DEVICE LOGICAL BLOCK ADDRESS				
27								(LSB)
28	(MSB)			SOURCE BLOCK DEVICE BYTE OFFSET				
29								(LSB)
30	(MSB)			DESTINATION BLOCK DEVICE BYTE OFFSET				
31								(LSB)

The DESCRIPTOR TYPE CODE field is described in 7.2.5 and 7.2.7.1. Descriptor type code 0Ah (block<o>→block<o>) instructs the copy manager to copy the data from the source block device identified by the SOURCE TARGET DESCRIPTOR INDEX field to the destination block device identified by the DESTINATION TARGET DESCRIPTOR INDEX field using data starting at the location identified by the source BLOCK DEVICE BYTE OFFSET field in the logical block identified by the SOURCE BLOCK DEVICE LOGICAL BLOCK ADDRESS field and continuing for the number of bytes specified in the NUMBER OF BYTES field. The data shall be written starting at the location identified by the DESTINATION BLOCK DEVICE BYTE OFFSET field in the logical block identified by the DESTINATION BLOCK DEVICE LOGICAL BLOCK ADDRESS field.

The content of the starting logical block on the destination device before the starting offset shall be preserved. The content on the ending logical block beyond the end of the transfer shall be preserved. The copy manager may implement this operation by reading the starting and ending logical blocks, modifying a portion of the blocks as required, and writing the full blocks to the destination device.

The CAT bit is described in 7.2.7.2.

The DESCRIPTOR LENGTH field shall contain 28 (001Ch). The SOURCE TARGET DESCRIPTOR INDEX and DESTINATION TARGET DESCRIPTOR INDEX fields are described in 7.2.7.1.

The NUMBER OF BYTES field specifies the number bytes to be read. A value of zero indicates that no bytes shall be transferred in this segment. This shall not be considered as an error.

The SOURCE BLOCK DEVICE LOGICAL BLOCK ADDRESS field specifies the starting address on the source block device for this segment.

The DESTINATION BLOCK DEVICE LOGICAL BLOCK ADDRESS field specifies the starting logical block address on the destination block device for this segment.

The SOURCE BLOCK DEVICE BYTE OFFSET field specifies the offset into the first source block at which to begin reading bytes.

The DESTINATION BLOCK DEVICE BYTE OFFSET field is the offset into the first destination block at which to begin writing data to the destination block device.

7.2.7.14 Write filemarks operation

The segment descriptor format shown in table 40 instructs the copy manager to write filemarks or setmarks on the destination tape device.

Table 40 — Write filemarks operation segment descriptor

Bit / Byte	7	6	5	4	3	2	1	0
0	DESCRIPTOR TYPE CODE (10h)							
1	Reserved							
2	(MSB)			DESCRIPTOR LENGTH (0008h)				
3								(LSB)
4	Reserved							
5	Reserved							
6	(MSB)			DESTINATION TARGET DESCRIPTOR INDEX				
7								(LSB)
8	Reserved						WSMK	Reserved
9	(MSB)			TRANSFER LENGTH				
10								
11								(LSB)

The DESCRIPTOR TYPE CODE field is described in 7.2.5 and 7.2.7.1. Descriptor type code 10h (filemark→tape) instructs the copy manager to write filemarks or setmarks to the destination tape device identified by the DESTINATION TARGET DESCRIPTOR INDEX field starting at the current position of the tape device. If the PERIPHERAL DEVICE TYPE field in the target descriptor identified by the DESTINATION TARGET DESCRIPTOR INDEX field does not contain 01h, the copy manager shall terminate the command with a CHECK CONDITION status. The sense key shall be set to COPY ABORTED and the additional sense code shall be set to INVALID OPERATION FOR COPY SOURCE OR DESTINATION.

The DESCRIPTOR LENGTH field shall contain 8 (0008h). The DESTINATION TARGET DESCRIPTOR INDEX field is described in 7.2.7.1.

If the write setmark (WSMK) bit is one, the TRANSFER LENGTH field specifies the number of setmarks to be written. If the WSMK bit is zero, the TRANSFER LENGTH field specifies the number of filemarks to be written.

7.2.7.15 Space operation

The segment descriptor format shown in table 41 instructs the copy manager to send a SPACE command (see SSC) to the destination tape device.

Table 41 — Space operation segment descriptor

Bit / Byte	7	6	5	4	3	2	1	0
0	DESCRIPTOR TYPE CODE (11h)							
1	Reserved							
2	(MSB)			DESCRIPTOR LENGTH (0008h)				
3								(LSB)
4	Reserved							
5	Reserved							
6	(MSB)			DESTINATION TARGET DESCRIPTOR INDEX				
7								(LSB)
8	Reserved					CODE		
9	(MSB)			COUNT				
10								
11								(LSB)

The DESCRIPTOR TYPE CODE field is described in 7.2.5 and 7.2.7.1. Descriptor type code 11h (space→tape) instructs the copy manager to send a SPACE command to the destination tape device identified by the DESTINATION TARGET DESCRIPTOR INDEX field. If the PERIPHERAL DEVICE TYPE field in the target descriptor identified by the DESTINATION TARGET DESCRIPTOR INDEX field does not contain 01h, the copy manager shall terminate the command with a CHECK CONDITION status. The sense key shall be set to COPY ABORTED and the additional sense code shall be set to INVALID OPERATION FOR COPY SOURCE OR DESTINATION.

The DESCRIPTOR LENGTH field shall contain 8 (0008h). The DESTINATION TARGET DESCRIPTOR INDEX field is described in 7.2.7.1.

The CODE and COUNT fields contents in the SPACE command sent to the destination tape device shall be copied from the CODE and COUNT fields in the segment descriptor. All other fields in the SPACE command sent to the destination tape device that affect the positioning of the tape shall be set to zero.

7.2.7.16 Locate operation

The segment descriptor format shown in table 42 instructs the copy manager to send a LOCATE command (see SSC) to the destination tape device.

Table 42 — Locate operation segment descriptor

Bit / Byte	7	6	5	4	3	2	1	0
0	DESCRIPTOR TYPE CODE (12h)							
1	Reserved							
2	(MSB)			DESCRIPTOR LENGTH (0008h)				
3								(LSB)
4	Reserved							
5	Reserved							
6	(MSB)			DESTINATION TARGET DESCRIPTOR INDEX				
7								(LSB)
8	(MSB)			BLOCK ADDRESS				
11								(LSB)

The DESCRIPTOR TYPE CODE field is described in 7.2.5 and 7.2.7.1. Descriptor type code 12h (locate→tape) instructs the copy manager to send a LOCATE command to the destination tape device identified by the DESTINATION TARGET DESCRIPTOR INDEX field. If the PERIPHERAL DEVICE TYPE field in the target descriptor identified by the DESTINATION TARGET DESCRIPTOR INDEX field does not contain 01h, the copy manager shall terminate the command with a CHECK CONDITION status. The sense key shall be set to COPY ABORTED and the additional sense code shall be set to INVALID OPERATION FOR COPY SOURCE OR DESTINATION.

The DESCRIPTOR LENGTH field shall contain 8 (0008h). The DESTINATION TARGET DESCRIPTOR INDEX field is described in 7.2.7.1.

The BLOCK ADDRESS field contents in the LOCATE command sent to the destination tape device shall be copied from the BLOCK ADDRESS field in the segment descriptor. All other fields in the LOCATE command sent to the destination tape device that affect the positioning of the tape shall be set to zero.

NOTE 9 The restrictions described above for the LOCATE command limit the operation to locating SCSI logical block addresses in the current tape partition.

7.2.7.17 Tape device image copy operation

The segment descriptor format shown in table 43 instructs the copy manager to perform an image copy from the source tape device to the destination tape device.

Table 43 — Tape device image copy segment descriptor

Bit / Byte	7	6	5	4	3	2	1	0
0	DESCRIPTOR TYPE CODE (13h)							
1	Reserved							
2	(MSB)			DESCRIPTOR LENGTH (0008h)				
3								(LSB)
4	(MSB)			SOURCE TARGET DESCRIPTOR INDEX				
5								(LSB)
6	(MSB)			DESTINATION TARGET DESCRIPTOR INDEX				
7								(LSB)
8	(MSB)			COUNT				
11								(LSB)

The DESCRIPTOR TYPE CODE field is described in 7.2.5 and 7.2.7.1. Descriptor type code 13h (<i>tape→<i>tape) instructs the copy manager to create a compatible image of the source device medium identified by the SOURCE TARGET DESCRIPTOR INDEX field on the destination device medium identified by the DESTINATION TARGET DESCRIPTOR INDEX field beginning at their current positions. If the PERIPHERAL DEVICE TYPE field in the target descriptor identified by the SOURCE TARGET DESCRIPTOR INDEX field or the DESTINATION TARGET DESCRIPTOR INDEX field does not contain 01h, the copy manager shall terminate the command with a CHECK CONDITION status. The sense key shall be set to COPY ABORTED and the additional sense code shall be set to INVALID OPERATION FOR COPY SOURCE OR DESTINATION.

The DESCRIPTOR LENGTH field shall contain 8 (0008h). The SOURCE TARGET DESCRIPTOR INDEX and DESTINATION TARGET DESCRIPTOR INDEX fields are described in 7.2.7.1.

The tape image copy operation terminates when:

a) the source device encounters an end-of-partition as defined by the source device;
b) the source device encounters an end-of-data as defined by the source device (i.e., BLANK CHECK sense key);
c) the copy manager has copied the number of consecutive filemarks specified in the count field from the source device to the destination device; or
d) the copy manager has copied the number of consecutive filemarks and/or setmarks specified in the count field from the source device to the destination device, if the RSMK bit in the device configuration page (see SSC) of the source device is on.

A COUNT field of zero indicates that the EXTENDED COPY command shall not terminate due to any number of consecutive filemarks or setmarks. Other error or exception conditions (e.g., early-warning, end-of-partition on destination device) may cause the EXTENDED COPY command to terminate prior to completion. In such cases, it is not possible to calculate a residue, so the information field in the sense data shall be set to zero.

7.2.7.18 Register key operation

The segment descriptor format shown in table 44 instructs the copy manager to register a persistent reservations key (see 5.5.3.4) with the device identified by the DESTINATION TARGET DESCRIPTOR INDEX field.

Table 44 — Register key segment descriptor

Bit / Byte	7	6	5	4	3	2	1	0
0	DESCRIPTOR TYPE CODE (14h)							
1	Reserved							
2	(MSB)			DESCRIPTOR LENGTH (0018h)				
3								(LSB)
4	Reserved							
5	Reserved							
6	(MSB)			DESTINATION TARGET DESCRIPTOR INDEX				
7								(LSB)
8	(MSB)			RESERVATION KEY				
15								(LSB)
16	(MSB)			SERVICE ACTION RESERVATION KEY				
23								(LSB)
24	Reserved							
27								

The DESCRIPTOR TYPE CODE field is described in 7.2.5 and 7.2.7.1. Descriptor type code 14h instructs the copy manager to register a persistent reservations key with the device identified by the DESTINATION TARGET DESCRIPTOR INDEX field using a PERSISTENT RESERVE OUT command with a REGISTER service action (see 7.11.2).

The DESCRIPTOR LENGTH field shall contain 24 (0018h). The DESTINATION TARGET DESCRIPTOR INDEX field is described in 7.2.7.1.

The RESERVATION KEY and SERVICE ACTION RESERVATION KEY field contents in the PERSISTENT RESERVE OUT command sent to the destination device shall be copied from the RESERVATION KEY and SERVICE ACTION RESERVATION KEY fields in the segment descriptor.

NOTE 10 The initiator sending the EXTENDED COPY command may need to remove the reservation key held by the copy manager as described in 5.5.3.6 prior to sending the EXTENDED COPY command.

7.3 INQUIRY command

7.3.1 INQUIRY command introduction

The INQUIRY command (see table 45) requests that information regarding parameters of the target and a component logical unit be sent to the application client. Options allow the application client to request additional information about the target and logical unit (see 7.3.4) or information about SCSI commands supported by the device server (see 7.3.5).

Table 45 — INQUIRY command

Bit / Byte	7	6	5	4	3	2	1	0
0	OPERATION CODE (12h)							
1	Reserved						CMDDT	EVPD
2	PAGE OR OPERATION CODE							
3	Reserved							
4	ALLOCATION LENGTH							
5	CONTROL							

An enable vital product data (EVPD) bit of one specifies that the device server shall return the vital product data specified by the PAGE OR OPERATION CODE field.

A command support data (CMDDT) bit of one specifies that the device server shall return the optional command support data specified by the PAGE OR OPERATION CODE field. If the device server does not support returning command data and this bit is set to one, the device server shall return CHECK CONDITION status with the sense key set to ILLEGAL REQUEST and an additional sense code of INVALID FIELD IN CDB. Details of the command support data may be found in 7.3.5.

NOTE 11 An application client may receive a CHECK CONDITION status response with the sense key set to ILLEGAL REQUEST upon sending an INQUIRY command with the CMDDT bit set to one to some SCSI-2 device servers, since this bit was reserved in SCSI-2.

If both the EVPD and CMDDT bits are zero, the device server shall return the standard INQUIRY data (see 7.3.2). If the PAGE OR OPERATION CODE field is not zero when both EVPD and CMDDT are zero, the device server shall return CHECK CONDITION status with the sense key set to ILLEGAL REQUEST and an additional sense code of INVALID FIELD IN CDB.

If both the EVPD and CMDDT bits are one, the device server shall return CHECK CONDITION status with the sense key set to ILLEGAL REQUEST and an additional sense code of INVALID FIELD IN CDB.

When the EVPD bit is one, the PAGE OR OPERATION CODE field specifies which page of vital product data information the device server shall return (see 8.4).

When the CMDDT bit is one, the PAGE OR OPERATION CODE field specifies the SCSI operation code for which device server shall return command support data (see 7.3.5).

The INQUIRY command shall return CHECK CONDITION status only when the device server is unable to return the requested INQUIRY data.

If an INQUIRY command is received from an initiator with a pending unit attention condition (i.e., before the device server reports CHECK CONDITION status), the device server shall perform the INQUIRY command and shall not clear the unit attention condition (see SAM-2).

The INQUIRY data should be returned even though the device server is not ready for other commands. To minimize delays after a hard reset or power-up condition, the standard INQUIRY data should be available without

incurring any media access delays. If the device server does store some of the INQUIRY data on the media, it may return zeros or ASCII spaces (20h) in those fields until the data is available from the media.

The INQUIRY data may change as the target executes its initialization sequence. For example, the target may contain a minimum command set in its nonvolatile memory and may load its final firmware from the media when it becomes ready. After the target has loaded the firmware, it may support more options and therefore return different supported options information in the INQUIRY data.

If the standard INQUIRY data changes for any reason, the device server shall generate a unit attention condition for all initiators (see SAM-2). The device server shall set the additional sense code to INQUIRY DATA HAS CHANGED. If INQUIRY VPD data changes for any reason, the device server may generate a unit attention condition for all initiators (see SAM-2), setting the additional sense code to INQUIRY DATA HAS CHANGED.

NOTE 12 The INQUIRY command may be used by an application client after a hard reset or power-up condition to determine the device types for system configuration.

7.3.2 Standard INQUIRY data

The standard INQUIRY data (see table 46) shall contain at least 36 bytes.

Table 46 — Standard INQUIRY data format

Bit / Byte	7	6	5	4	3	2	1	0
0	PERIPHERAL QUALIFIER			PERIPHERAL DEVICE TYPE				
1	RMB	Reserved						
2	VERSION							
3	AERC	Obsolete	NORMACA	HISUP	RESPONSE DATA FORMAT			
4	ADDITIONAL LENGTH (n-4)							
5	SCCS	Reserved						
6	BQUE	ENCSERV	VS	MULTIP	MCHNGR	Obsolete	Obsolete	ADDR16†
7	RELADR	Obsolete	WBUS16†	SYNC†	LINKED	Obsolete	CMDQUE	VS
8	(MSB)	VENDOR IDENTIFICATION						
15								(LSB)
16	(MSB)	PRODUCT IDENTIFICATION						
31								(LSB)
32	(MSB)	PRODUCT REVISION LEVEL						
35								(LSB)
36	Vendor specific							
55								
56	Reserved				CLOCKING†		QAS†	IUS†
57	Reserved							
58	(MSB)	VERSION DESCRIPTOR 1						
59								(LSB)
	. . .							
72	(MSB)	VERSION DESCRIPTOR 8						
73								(LSB)
74	Reserved							
95								
	Vendor specific parameters							
96	Vendor specific							
n								

Note: † The meanings of these fields are specific to SPI-3 (see 7.3.3). For protocols other than the SCSI Parallel Interface, these fields are reserved.

The PERIPHERAL QUALIFIER and PERIPHERAL DEVICE TYPE fields identify the device currently connected to the logical unit. If the target is not capable of supporting a device on this logical unit, the device server shall set this field to 7Fh (PERIPHERAL QUALIFIER set to 011b and PERIPHERAL DEVICE TYPE set to 1Fh).

The peripheral qualifier is defined in table 47 and the peripheral device type is defined in table 48.

Table 47 — Peripheral qualifier

Qualifier	Description
000b	The specified peripheral device type is currently connected to this logical unit. If the device server is unable to determine whether or not a physical device is currently connected, it also shall use this peripheral qualifier when returning the INQUIRY data. This peripheral qualifier does not mean that the device is ready for access by the initiator.
001b	The device server is capable of supporting the specified peripheral device type on this logical unit. However, the physical device is not currently connected to this logical unit.
010b	Reserved
011b	The device server is not capable of supporting a physical device on this logical unit. For this peripheral qualifier the peripheral device type shall be set to 1Fh to provide compatibility with previous versions of SCSI. All other peripheral device type values are reserved for this peripheral qualifier.
1xxb	Vendor specific

Table 48 — Peripheral device type

Code	Doc. [a]	Description
00h	SBC	Direct-access device (e.g., magnetic disk)
01h	SSC	Sequential-access device (e.g., magnetic tape)
02h	SSC	Printer device
03h	SPC-2	Processor device
04h	SBC	Write-once device (e.g., some optical disks)
05h	MMC-2	CD-ROM device
06h	SCSI-2	Scanner device
07h	SBC	Optical memory device (e.g., some optical disks)
08h	SMC	Medium changer device (e.g., jukeboxes)
09h	SCSI-2	Communications device
0Ah - 0Bh		Defined by ASC IT8 (Graphic arts pre-press devices)
0Ch	SCC-2	Storage array controller device (e.g., RAID)
0Dh	SES	Enclosure services device
0Eh	RBC	Simplified direct-access device (e.g., magnetic disk)
0Fh	OCRW	Optical card reader/writer device
10h		Reserved [b]
11h	OSD	Object-based Storage Device
12h - 1Eh		Reserved
1Fh		Unknown or no device type

[a] All standards are subject to revision, and parties to agreements based on this standard are encouraged to investigate the possibility of applying the most recent editions of the listed standards.

[b] Type code 10h is reserved for use by Bridging Expanders.

A removable medium (RMB) bit of zero indicates that the medium is not removable. A RMB bit of one indicates that the medium is removable.

The VERSION field indicates the implemented version of this standard and is defined in table 49.

Table 49 — Version

Code	Description		
00h	The device does not claim conformance to any standard.		
02h - 03h	The device complies to American National Standards.		
04h	The device complies to this standard.		
80h	The device complies to ISO/IEC 9316:1995.		
82h - 83h	The device complies to ISO/IEC 9316:1995 and to American National Standards.		
84h	The device complies to ISO/IEC 9316:1995 and to this standard.		
Code	**Description**	**Code**	**Description**
1h	Obsolete (SCSI=001b)	5h - 7h	Reserved
08h - 0Ch	Obsolete (ECMA=001b)	0Dh - 3Fh	Reserved
40h - 44h	Obsolete (ISO=01b)	45h - 47h	Reserved
48h - 4Ch	Obsolete (ISO=01b & ECMA=001b)	4Dh - 7Fh	Reserved
81h	Obsolete (SCSI=001b)	85h - 87h	Reserved
88h - 8Ch	Obsolete (ECMA=001b)	8Dh - FFh	Reserved
ISO/IEC 9316:1995 is SCSI-2			

The asynchronous event reporting capability (AERC) bit indicates that the target supports the asynchronous event reporting capability as defined in SAM-2. The AERC bit is qualified by the PERIPHERAL DEVICE TYPE field as follows:

a) Processor device-type definition: An AERC bit of one indicates that the processor device is capable of accepting asynchronous event reports. An AERC bit of zero indicates that the processor device does not support asynchronous event reports; or
b) All other device-types: This bit is reserved.

Details of the asynchronous event reporting support are protocol specific.

The Normal ACA Supported bit (NORMACA) of one indicates that the device server supports setting the NACA bit to one in the CONTROL byte of the CDB (see SAM-2). A NORMACA bit of zero indicates that the device server does not support setting the NACA bit to one.

A hierarchical support (HISUP) bit of zero indicates the target does not use the hierarchical addressing model to assign LUNs to logical units. A HISUP bit of one indicates the target uses the hierarchical addressing model to assign LUNs to logical units. When the HISUP bit is one, the device server shall support the REPORT LUNS command (see 7.19). When the HISUP bit is zero, the device server may support the REPORT LUNS command.

A RESPONSE DATA FORMAT field value of two indicates that the data shall be in the format specified in this standard. Response data format values less than two are obsolete. Response data format values greater than two are reserved.

The ADDITIONAL LENGTH field shall specify the length in bytes of the parameters. If the ALLOCATION LENGTH of the CDB is too small to transfer all of the parameters, the ADDITIONAL LENGTH shall not be adjusted to reflect the truncation.

An SCC Supported (SCCS) bit of one indicates that the device contains an embedded storage array controller component. See SCC-2 for details about storage array controller devices. An SCCS bit of zero indicates that the device does not contain an embedded storage array controller component.

The basic queuing (BQUE) bit shall be zero if the CMDQUE bit is one.

When the CMDQUE bit is zero, the BQUE bit shall have the following meaning. A BQUE bit of zero indicates that the device does not support tagged tasks (command queuing) for this logical unit. A value of one indicates that the device supports, for this logical unit, the basic task management model defined by SAM-2.

An Enclosure Services (ENCSERV) bit of one indicates that the device contains an embedded enclosure services component. See SES for details about enclosure services, including a device model for an embedded enclosure services device. An ENCSERV bit of zero indicates that the device does not contain an embedded enclosure services component.

A Multi Port (MULTIP) bit of one indicates that this is a multi-port (two or more ports) device and conforms to the SCSI multi-port device requirements found in the applicable standards (e.g., SAM-2, a protocol standard and possibly provisions of a command set standard). A value of zero indicates that this device has a single port and does not implement the multi-port requirements.

A medium changer (MCHNGR) bit of one indicates that the device is associated with or attached to a medium transport element. See SMC for details about medium changers, including a device model for an attached medium changer device. The MCHNGR bit is valid only when the RMB bit is equal to one. A MCHNGR bit of zero indicates that the device is not embedded within or attached to a medium transport element.

A relative addressing (RELADR) bit of one indicates that the device server supports the relative addressing mode. If this bit is set to one, the linked command (LINKED) bit shall also be set to one; since relative addressing is only allowed with linked commands. A RELADR bit of zero indicates the device server does not support relative addressing.

A linked command (LINKED) bit of one indicates that the device server supports linked commands (see SAM-2). A value of zero indicates the device server does not support linked commands.

A command queuing (CMDQUE) bit of one indicates that the device supports tagged tasks (command queuing) for this logical unit (see SAM-2). A value of zero indicates the device server may support tagged tasks for this logical unit (see the BQUE bit, above). Table 50 summarizes the relationship of the BQUE and CMDQUE bits.

Table 50 — Relationship of BQUE and CMDQUE bits

BQUE	CMDQUE	Description
0	0	No command queuing of any kind supported.
0	1	Command queuing with all types of task tags supported.
1	0	Basic task management model supported (see SAM-2)
1	1	Illegal combination of BQUE and CMDQUE bits.

ASCII data fields shall contain only graphic codes (i.e., code values 20h through 7Eh). Left-aligned fields shall place any unused bytes at the end of the field (highest offset) and the unused bytes shall be filled with space characters (20h). Right-aligned fields shall place any unused bytes at the start of the field (lowest offset) and the unused bytes shall be filled with space characters (20h).

The VENDOR IDENTIFICATION field contains eight bytes of ASCII data identifying the vendor of the product. The data shall be left aligned within this field.

NOTE 13 It is intended that this field provide a unique vendor identification of the manufacturer of the SCSI device. In the absence of a formal registration procedure, T10 maintains a list of vendor identification codes in use. Vendors are requested to voluntarily submit their identification codes to T10 to prevent duplication of codes (see Annex D).

The PRODUCT IDENTIFICATION field contains sixteen bytes of ASCII data as defined by the vendor. The data shall be left-aligned within this field.

The PRODUCT REVISION LEVEL field contains four bytes of ASCII data as defined by the vendor. The data shall be left-aligned within this field.

The VERSION DESCRIPTOR fields provide for identifying up to eight standards to which the device claims conformance. The value in each VERSION DESCRIPTOR field shall be selected from table 51. All version descriptor values not listed in table 51 are reserved. Technical Committee T10 of NCITS maintains an electronic copy of the information in table 51 on its world wide web site (http://www.t10.org/). In the event that the T10 world wide web site is no longer active, access may be possible via the NCITS world wide web site (http://www.ncits.org), the ANSI world wide web site (http://www.ansi.org), the IEC site (http://www.iec.ch/), the ISO site (http://www.iso.ch/), or the

ISO/IEC JTC 1 web site (http://www.jtc1.org/). It is recommended that the first version descriptor be used for the SCSI architecture standard, followed by the physical standard, followed by the physical/mapping protocol if any, followed by the appropriate SPC version, followed by the device type command set, followed by a secondary command set if any.

Table 51 — Version descriptor values (part 1 of 4)

Standard	Version Descriptor Value
EPI ANSI NCITS TR-23:1999	0B3Ch
EPI (no version claimed)	0B20h
EPI T10/1134 revision 16	0B3Bh
Fast-20 (no version claimed)	0AC0h
Fast-20 ANSI X3.277:1996	0ADCh
Fast-20 T10/1071 revision 06	0ADBh
FC-AL (no version claimed)	0D40h
FC-AL ANSI X3.272:1996	0D5Ch
FC-AL-2 (no version claimed)	0D60h
FC-AL-2 ANSI NCITS.332:1999	0D7Ch
FC-AL-2 T11/1133 revision 7.0	0D61h
FC-FLA (no version claimed)	1320h
FC-FLA ANSI NCITS TR-20:1998	133Ch
FC-FLA T11/1235 revision 7	133Bh
FC-FS (no version claimed)	0DA0h
FC-FS T11/1331 revision 1.2	0DB7h
FCP (no version claimed)	08C0h
FCP ANSI X3.269:1996	08DCh
FCP T10/0993 revision 12	08DBh
FC-PH (no version claimed)	0D20h
FC-PH ANSI X3.230:1994	0D3Bh
FC-PH ANSI X3.230:1994 with Amnd 1 ANSI X3.230/AM1:1996	0D3Ch
FC-PH-3 (no version claimed)	0D80h
FC-PH-3 ANSI X3.303-1998	0D9Ch
FC-PLDA (no version claimed)	1340h
FC-PLDA ANSI NCITS TR-19:1998	135Ch
FC-PLDA T11/1162 revision 2.1	135Bh
FCP-2 (no version claimed)	0900h
FCP-2 T10/1144 revision 4	0901h
FC-Tape (no version claimed)	1300h
FC-Tape ANSI NCITS TR-24:1999	131Ch
FC-Tape T11/1315 revision 1.17	131Bh
FC-Tape T11/1315 revision 1.16	1301h
IEEE 1394 (no version claimed)	14A0h
ANSI IEEE 1394:1995	14BDh
IEEE 1394a (no version claimed)	14C0h
IEEE 1394b (no version claimed)	14E0h
iSCSI (no version claimed)	0960h
MMC (no version claimed)	0140h
Annex C contains the version descriptor value assignments in numeric order.	

Table 51 — Version descriptor values (part 2 of 4)

Standard	Version Descriptor Value
MMC ANSI X3.304:1997	015Ch
MMC T10/1048 revision 10a	015Bh
MMC-2 (no version claimed)	0240h
MMC-2 ANSI NCITS.333:2000	025Ch
MMC-2 T10/1228 revision 11a	025Bh
MMC-2 T10/1228 revision 11	0255h
MMC-3 (no version claimed)	02A0h
OCRW (no version claimed)	0280h
OCRW ISO/IEC 14776-381	029Eh
OSD (no version claimed)	0340h
OSD T10/1355 revision 0	0341h
RBC (no version claimed)	0220h
RBC ANSI NCITS.330:2000	023Ch
RBC T10/1240 revision 10a	0238h
RMC (no version claimed)	02C0h
SAM (no version claimed)	0020h
SAM ANSI X3.270:1996	003Ch
SAM T10/0994 revision 18	003Bh
SAM-2 (no version claimed)	0040h
SBC (no version claimed)	0180h
SBC ANSI NCITS.306:1998	019Ch
SBC T10/0996 revision 08c	019Bh
SBC-2 (no version claimed)	0320h
SBP-2 (no version claimed)	08E0h
SBP-2 ANSI NCITS.325:1999	08FCh
SBP-2 T10/1155 revision 04	08FBh
SCC (no version claimed)	0160h
SCC ANSI X3.276:1997	017Ch
SCC T10/1047 revision 06c	017Bh
SCC-2 (no version claimed)	01E0h
SCC-2 ANSI NCITS.318:1998	01FCh
SCC-2 T10/1125 revision 04	01FBh
SES (no version claimed)	01C0h
SES ANSI NCITS.305:1998	01DCh
SES T10/1212 revision 08b	01DBh
SES ANSI NCITS.305:1998 w/ Amendment ANSI NCITS.305/AM1:2000	01DEh
SES T10/1212 revision 08b w/ Amendment ANSI NCITS.305/AM1:2000	01DDh
SIP (no version claimed)	08A0h
SIP ANSI X3.292:1997	08BCh
SIP T10/0856 revision 10	08BBh
SMC (no version claimed)	01A0h
SMC ANSI NCITS.314:1998	01BCh
SMC T10/0999 revision 10a	01BBh
Annex C contains the version descriptor value assignments in numeric order.	

Table 51 — Version descriptor values (part 3 of 4)

Standard	Version Descriptor Value
SMC-2 (no version claimed)	02E0h
SPC (no version claimed)	0120h
SPC ANSI X3.301:1997	013Ch
SPC T10/0995 revision 11a	013Bh
SPC-2 (no version claimed)	0260h
SPC-2 T10/1236 revision 12	0267h
SPC-2 T10/1236 revision 18	0269h
SPC-3 (no version claimed)	0300h
SPI (no version claimed)	0AA0h
SPI ANSI X3.253:1995	0ABAh
SPI T10/0855 revision 15a	0AB9h
SPI ANSI X3.253:1995 with SPI Amnd ANSI X3.253/AM1:1998	0ABCh
SPI T10/0855 revision 15a with SPI Amnd revision 3a	0ABBh
SPI-2 (no version claimed)	0AE0h
SPI-2 ANSI X3.302:1999	0AFCh
SPI-2 T10/1142 revision 20b	0AFBh
SPI-3 (no version claimed)	0B00h
SPI-3 ANSI NCITS.336:2000	0B1Ch
SPI-3 T10/1302-D revision 14	0B1Ah
SPI-3 T10/1302-D revision 10	0B18h
SPI-3 T10/1302-D revision 13a	0B19h
SPI-4 (no version claimed)	0B40h
SRP (no version claimed)	0940h
SSA-PH2 (no version claimed)	1360h
SSA-PH2 ANSI X3.293:1996	137Ch
SSA-PH2 T10.1/1145 revision 09c	137Bh
SSA-PH3 (no version claimed)	1380h
SSA-PH3 ANSI NCITS.307:1998	139Ch
SSA-PH3 T10.1/1146 revision 05b	139Bh
SSA-S2P (no version claimed)	0880h
SSA-S2P ANSI X3.294:1996	089Ch
SSA-S2P T10.1/1121 revision 07b	089Bh
SSA-S3P (no version claimed)	0860h
SSA-S3P ANSI NCITS.309:1998	087Ch
SSA-S3P T10.1/1051 revision 05b	087Bh
SSA-TL1 (no version claimed)	0840h
SSA-TL1 ANSI X3.295:1996	085Ch
SSA-TL1 T10.1/0989 revision 10b	085Bh
SSA-TL2 (no version claimed)	0820h
SSA-TL2 ANSI NCITS.308:1998	083Ch
SSA-TL2 T10.1/1147 revision 05b	083Bh
SSC (no version claimed)	0200h
SSC ANSI NCITS.335:2000	021Ch
Annex C contains the version descriptor value assignments in numeric order.	

Table 51 — Version descriptor values (part 4 of 4)

Standard	Version Descriptor Value
SSC T10/0997 revision 22	0207h
SSC T10/0997 revision 17	0201h
SSC-2 (no version claimed)	0360h
SST (no version claimed)	0920h
Version Descriptor Not Supported or No Standard Identified	0000h
Annex C contains the version descriptor value assignments in numeric order.	

7.3.3 SCSI Parallel Interface specific INQUIRY data

Portions of bytes 6 and 7 and all of byte 56 of the standard INQUIRY data shall be used only by the SCSI Parallel Interface. These fields are noted in table 46. For details on how the SPI-specific fields relate to the SCSI Parallel Interface see SPI-n (where n is 2 or greater). Table 52 shows just the SPI-specific standard INQUIRY fields. The definitions of the SCSI Parallel Interface specific fields shall be as follows.

Table 52 — SPI-specific standard INQUIRY bits

Bit / Byte	7	6	5	4	3	2	1	0
6	see table 46							ADDR16
7	see table 46		WBUS16	SYNC	see table 46	Obsolete	see table 46	
	. . .							
56	Reserved				CLOCKING		QAS	IUS

A wide SCSI address 16 (ADDR16) bit of one indicates that the target supports 16-bit wide SCSI addresses. A value of zero indicates that the device does not support 16-bit wide SCSI addresses.

A wide bus 16 (WBUS16) bit of one indicates that the target supports 16-bit wide data transfers. A value of zero indicates that the device does not support 16-bit wide data transfers.

A synchronous transfer (SYNC) bit of one indicates that the target supports synchronous data transfer. A value of zero indicates the device does not support synchronous data transfer.

The obsolete bit 2 in byte 7 indicates whether the target supports an obsolete data transfers management mechanism defined in SPI-2.

Table 53 defines the relationships between the ADDR16 and WBUS16 bits.

Table 53 — Maximum logical device configuration table

ADDR16	WBUS16	Description
0	0	8 bit wide data path on a single cable with 8 SCSI IDs supported
0	1	16 bit wide data path on a single cable with 8 SCSI IDs supported
1	1	16 bit wide data path on a single cable with 16 SCSI IDs supported

The CLOCKING field shall not apply to asynchronous transfers and is defined in table 54.

Table 54 — CLOCKING field

Code	Description
00b	Indicates the device server supports only ST
01b	Indicates the device server supports only DT
10b	Reserved
11b	Indicates the device server supports ST and DT

A quick arbitration and selection supported (QAS) bit of one indicates that the device server supports quick arbitration and selection. A value of zero indicates that the device server does not support quick arbitration and selection.

An information units supported (IUS) bit of one indicates that the device server supports information unit transfers. A value of zero indicates that the device server does not support information unit transfers.

NOTE 14 The acronyms ST and DT and the terms 'quick arbitration and selection' and 'information units' are defined in SPI-3, SPI-4, and possibly later revisions of the SCSI parallel interface standard.

7.3.4 Vital product data

The application client requests the vital product data information by setting the EVPD bit to one and specifying the page code of the desired vital product data. See 8.4 for details about vital product data. The information returned consists of configuration data (e.g., vendor identification, product identification, model, serial number), manufacturing data (e.g., plant and date of manufacture), field replaceable unit data and other vendor specific or device specific data. If the device server does not implement the requested page it shall return CHECK CONDITION status. A sense key shall be set to ILLEGAL REQUEST and the additional sense code shall be set to INVALID FIELD IN CDB.

NOTES

15 The device server should have the ability to process the INQUIRY command even when an error occurs that prohibits normal command completion. In such a case CHECK CONDITION status should be returned for commands other than INQUIRY or REQUEST SENSE. The sense data returned may contain the field replaceable unit code. The vital product data may be obtained for the failing device using the INQUIRY command.

16 This standard defines a format that allows device-independent application client software to display the vital product data returned by the INQUIRY command. The contents of the data may be vendor specific, and may be unusable without detailed information about the device.

17 This standard does not define the location or method of storing the vital product data. The retrieval of the data may require completion of initialization operations within the device, that may induce delays before the data is available to the application client. Time-critical requirements are an implementation consideration and are not addressed in this standard.

7.3.5 Command support data

Implementation of command support data is optional. The application client may request the command support data information by setting the CMDDT bit to one and specifying the SCSI operation code of the desired CDB.

If the device server implements the requested SCSI operation code, it shall return the data defined in table 55. If the device server does not implement the requested SCSI operation code it shall return the peripheral qualifier and type byte and 001b in the SUPPORT field.

Table 55 — Command support data format

Bit / Byte	7	6	5	4	3	2	1	0
0	PERIPHERAL QUALIFIER			PERIPHERAL DEVICE TYPE				
1	Reserved					SUPPORT		
2	VERSION							
3	Reserved							
4	Reserved							
5	CDB SIZE (m - 5)							
6	CDB USAGE DATA							
m								

The PERIPHERAL QUALIFIER, the PERIPHERAL DEVICE TYPE, and the VERSION fields are defined in 7.3.2.

Table 56 defines the values and meanings of the SUPPORT field.

Table 56 — SUPPORT values and meanings

Support	Description
000b	Data about the requested SCSI operation code is not currently available.
001b	The device server does not support the tested SCSI operation code. All data after byte 1 is undefined.
010b	Reserved
011b	The device server supports the tested SCSI operation code in conformance with a SCSI standard. The data format conforms to the definition in table 55.
100b	Vendor specific
101b	The device server supports the tested SCSI operation code in a vendor specific manner. The data format conforms to the definition in table 55.
110b	Vendor specific
111b	Reserved

If the SUPPORT field contains 000b, all data after byte 1 is not valid. One possible reason for SUPPORT being 000b is the device server's inability to retrieve information stored on the media. When this is the case, a subsequent request for command support data may be successful.

The CDB SIZE field shall contain the number of bytes in the CDB for the operation code being queried, and the size of the CDB USAGE DATA field in the return data.

NOTE 18 The CDB SIZE field is provided for the convenience of the application client. In most cases, the size is known from the operation code group.

The CDB USAGE DATA field shall contain information about the CDB for the operation code being queried. The first byte of the CDB usage data shall contain the operation code for the operation being queried. All bytes except the first byte of the CDB usage data shall contain a usage map for bits in the CDB for the operation code being queried.

The bits in the usage map shall have a one-for-one correspondence to the CDB for the operation code being queried. If the device server evaluates a bit in the CDB for the operation code being queried, the usage map shall contain a one in the corresponding bit position. If any bit representing part of a field is returned as one all bits for the field shall be returned as one. If the device server ignores or treats as reserved a bit in the CDB for the operation code being queried, the usage map shall contain a zero in the corresponding bit position. The usage map bits for a given CDB field shall all have the same value.

For example, the CDB usage bit map for the SEND DIAGNOSTIC command of a device server that implements only the default self-test capability is: 1Dh, 04h, 00h, 00h, 00h, 07h. This example assumes that SAM-2 defines uses for only the low-order three bits of the CONTROL byte. Note that the first byte contains the operation code and the remaining bytes contain the usage map.

7.4 LOG SELECT command

The LOG SELECT command (see table 57) provides a means for an application client to manage statistical information maintained by the device about the device or its logical units. Device servers that implement the LOG SELECT command shall also implement the LOG SENSE command. Structures in the form of log parameters within log pages are defined as a way to manage the log data. The LOG SELECT command provides for sending zero or more log pages via the Data-Out Buffer. This standard defines the format of the log pages, but does not define the exact conditions and events that are logged.

Table 57 — LOG SELECT command

Bit / Byte	7	6	5	4	3	2	1	0
0	OPERATION CODE (4Ch)							
1	Reserved						PCR	SP
2	PC		Reserved					
3	Reserved							
4	Reserved							
5	Reserved							
6	Reserved							
7	(MSB)	PARAMETER LIST LENGTH						
8								(LSB)
9	CONTROL							

A parameter code reset (PCR) bit of one and a parameter list length of zero shall cause all implemented parameters to be set to the target-defined default values (e.g., zero). If the PCR bit is one and the parameter list length is greater than zero, the command shall be terminated with CHECK CONDITION status. The sense key shall be set to ILLEGAL REQUEST and the additional sense code shall be set to INVALID FIELD IN CDB. A PCR bit of zero specifies that the log parameters shall not be reset.

A save parameters (SP) bit of one indicates that after performing the specified LOG SELECT operation the target shall save to nonvolatile memory all parameters identified as savable by the DS bit in the log page (see 8.2). A SP bit of zero specifies that parameters shall not be saved.

Saving of log parameters is optional and indicated for each log parameter by the DS bit in the page. Log parameters also may be saved at vendor specific times subject to the TSD bit (see 8.2) in the log parameter or the GLTSD bit in the control mode page (see 8.3.6). If the target does not implement saved parameters for any log parameter and the SP bit is set to one, the command shall be terminated with CHECK CONDITION status. The sense key shall be set to ILLEGAL REQUEST, and the additional sense code set to INVALID FIELD IN CDB.

It is not an error to set the SP bit to one and to set the DS bit of a log parameter to one. In this case, the parameter value for that log parameter is not saved.

The page control (PC) field defines the type of parameter values to be selected. The PC field is defined in table 58.

Table 58 — Page control field

PC	LOG SELECT parameter values	LOG SENSE parameter values
00b	Current threshold values	Threshold values
01b	Current cumulative values	Cumulative values
10b	Default threshold values	Default threshold values
11b	Default cumulative values	Default cumulative values

The current cumulative values may be updated by the target or by the application client using the LOG SELECT command to reflect the cumulative number of events experienced by the target. Fields in the parameter control byte (see 8.2) of each log parameter control the updating and saving of the current cumulative parameters.

The device server shall set the current threshold parameters to the default threshold values in response to a LOG SELECT command with the PC field set to 10b and the parameter list length field set to zero.

The device server shall set all cumulative parameters to their default values in response to a LOG SELECT command with the PC field set to 11b and the parameter list length field set to zero.

The current threshold value may only be modified by the application client via the LOG SELECT command. If the application client attempts to change current threshold values that are not available or not implemented for that log parameter, then the device server shall terminate the LOG SELECT command with CHECK CONDITION status, the sense key set to ILLEGAL REQUEST, and the additional sense code set to INVALID FIELD IN PARAMETER LIST. The saving of current threshold parameters and the criteria for the current threshold being met are controlled by bits in the parameter control byte (see 8.2).

NOTE 19 Pages or log parameters that are not available may become available at some later time (e.g., after the device has become ready).

The PARAMETER LIST LENGTH field specifies the length in bytes of the parameter list that shall be located in the Data-Out Buffer. A parameter list length of zero indicates that no pages shall be transferred. This condition shall not be considered an error. If an application client sends page codes or parameter codes within the parameter list that are reserved or not implemented by the target, the device server shall terminate the LOG SELECT command with CHECK CONDITION status. The sense key shall be set to ILLEGAL REQUEST and the additional sense code set to INVALID FIELD IN PARAMETER LIST.

If a parameter list length results in the truncation of any log parameter, the device server shall terminate the command with CHECK CONDITION status. The sense key shall be set to ILLEGAL REQUEST and the additional sense code set to INVALID FIELD IN CDB.

The application client should send pages in ascending order by page code value if the Data-Out Buffer contains multiple pages. If the Data-Out Buffer contains multiple log parameters within a page, they should be sent in ascending order by parameter code value. The device server shall return CHECK CONDITION status if the application client sends pages out of order or parameter codes out of order. The sense key shall be set to ILLEGAL REQUEST and the additional sense code set to INVALID FIELD IN PARAMETER LIST.

NOTE 20 Initiators should issue LOG SENSE commands prior to issuing LOG SELECT commands to determine supported pages and page lengths.

The target may provide independent sets of log parameters for each logical unit or for each combination of logical units and initiators. If the target does not support independent sets of log parameters and any log parameters are changed that affect other initiators, then the device server shall generate a unit attention condition for all initiators except the one that issued the LOG SELECT command (see SAM-2). This unit attention condition shall be returned with an additional sense code of LOG PARAMETERS CHANGED.

If an application client sends a log parameter that is not supported by the target, the device server shall terminate the command with CHECK CONDITION status, set the sense key to ILLEGAL REQUEST, and set the additional sense code to INVALID FIELD IN PARAMETER LIST.

Additional information about the LOG SELECT command may be found in informative Annex A.

7.5 LOG SENSE command

The LOG SENSE command (see table 59) provides a means for the application client to retrieve statistical or other operational information maintained by the device about the device or its logical units. It is a complementary command to the LOG SELECT command.

Table 59 — LOG SENSE command

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

The parameter pointer control (PPC) bit controls the type of parameters requested from the device server:

a) A PPC bit of one indicates that the device server shall return a log page with parameter code values that have changed since the last LOG SELECT or LOG SENSE command. The device server shall return only those parameter codes following the PARAMETER POINTER field.
b) A PPC bit of zero indicates that the log parameter requested from the device server shall begin with the parameter code specified in the PARAMETER POINTER field and return the number of bytes specified by the ALLOCATION LENGTH field in ascending order of parameter codes from the specified log page. A PPC bit of zero and a PARAMETER POINTER field of zero shall cause all available log parameters for the specified log page to be returned to the application client subject to the specified allocation length.

Saving parameters is an optional function of the LOG SENSE command. If the target does not implement saving log parameters and if the save parameters (SP) bit is one, then the device server shall return CHECK CONDITION status, set the sense key to ILLEGAL REQUEST, and set the additional sense code to INVALID FIELD IN CDB.

An SP bit of zero indicates the device server shall perform the specified LOG SENSE command and shall not save any log parameters. If saving log parameters is implemented, an SP bit of one indicates that the device server shall perform the specified LOG SENSE command and shall save all log parameters identified as savable by the DS bit (see 8.2) to a nonvolatile, vendor specific location.

The page control (PC) field defines the type of parameter values to be selected (see 7.4 for the definition of the page control field). The parameter values returned by a LOG SENSE command are determined as follows:

a) The specified parameter values at the last update (i.e., in response to a LOG SELECT or LOG SENSE command or done automatically by the target for cumulative values);
b) The saved values, if an update has not occurred since the last power-on or hard reset condition and saved parameters are implemented; or
c) The default values, if an update has not occurred since the last power-on or hard reset condition and saved values are not available or not implemented.

The PAGE CODE field identifies which page of data is being requested (see 8.2). If the page code is reserved or not implemented, the device server shall terminate the command with CHECK CONDITION status. The sense key shall be set to ILLEGAL REQUEST with the additional sense code set to INVALID FIELD IN CDB.

The PARAMETER POINTER field allows the application client to request parameter data beginning from a specific parameter code to the maximum allocation length or the maximum parameter code supported by the target, whichever is less. If the value of the PARAMETER POINTER field is larger than the largest available parameter code known to the device server for the specified page, the device server shall terminate the command with CHECK CONDITION status. The sense key shall be set to ILLEGAL REQUEST and the additional sense code shall be set to INVALID FIELD IN CDB.

Log parameters within the specified log page shall be transferred in ascending order according to parameter code.

Additional information about the LOG SENSE command may be found in Annex A.

7.6 MODE SELECT(6) command

The MODE SELECT(6) command (see table 60) provides a means for the application client to specify medium, logical unit, or peripheral device parameters to the device server. Device servers that implement the MODE SELECT(6) command shall also implement the MODE SENSE(6) command. Application clients should issue MODE SENSE(6) prior to each MODE SELECT(6) to determine supported pages, page lengths, and other parameters.

Table 60 — MODE SELECT(6) command

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

If a target supports saved pages, it may save only one copy of the page for each logical unit and have it apply to all initiators, or it may save separate copies for each initiator for each logical unit. Multiple port implementations may save one copy per logical unit and have it apply to all initiators on all ports or save a separate copy per logical unit for each initiator on each port. If separate copies are saved, the target shall maintain separate current values for each combination of initiator and logical unit that it detects. Pages that are common to all initiators are not required to have multiple copies.

If an application client sends a MODE SELECT command that changes any parameters applying to other initiators, the device server shall generate a unit attention condition for all initiators except the one that issued the MODE SELECT command (see SAM-2). The device server shall set the additional sense code to MODE PARAMETERS CHANGED.

The target may provide for independent sets of parameters for each attached logical unit or for each combination of logical unit and initiator. If independent sets of parameters are implemented, and a third-party reservation is requested, the device server shall transfer the set of parameters in effect for the initiator that sent the RESERVE command to the parameters used for commands from the third-party device (see 7.21.3).

A page format (PF) bit of zero indicates that all parameters after the block descriptors are vendor specific. A PF bit of one indicates that the MODE SELECT parameters following the header and block descriptor(s) are structured as pages of related parameters and are as specified in this standard.

A save pages (SP) bit of zero indicates the device server shall perform the specified MODE SELECT operation, and shall not save any pages. If the target implements no distinction between current and saved pages and the SP bit is zero, the command shall be terminated with CHECK CONDITION status. The sense key shall be set to ILLEGAL REQUEST, and the additional sense code set to INVALID FIELD IN CDB. An SP bit of one indicates that the device server shall perform the specified MODE SELECT operation, and shall save to a nonvolatile vendor specific location all the savable pages including any sent in the Data-Out Buffer. The SP bit is optional, even when mode pages are supported by the target. Pages that are saved are identified by the parameter savable bit that is returned in the page header by the MODE SENSE command (see 8.3). If the PS bit is set to one in the MODE SENSE data then the page shall be savable by issuing a MODE SELECT command with the SP bit set to one. If the target does not implement saved pages and the SP bit is set to one, the command shall be terminated with CHECK CONDITION status. The sense key shall be set to ILLEGAL REQUEST, and the additional sense code set to INVALID FIELD IN CDB.

The PARAMETER LIST LENGTH field specifies the length in bytes of the mode parameter list that shall be contained in the Data-Out Buffer. A parameter list length of zero indicates that the Data-Out Buffer shall be empty. This condition shall not be considered as an error.

The device server shall terminate the command with CHECK CONDITION status if the parameter list length results in the truncation of any mode parameter header, mode parameter block descriptor(s), or mode page. The sense key shall be set to ILLEGAL REQUEST, and the additional sense code shall be set to PARAMETER LIST LENGTH ERROR.

The mode parameter list for the MODE SELECT and MODE SENSE commands is defined in 8.3. Parts of each mode parameter list are defined in a device-type dependent manner. Definitions for the parts of each mode parameter list that are uniquely for each device-type may be found in the applicable command standards (see 3.1.12).

The device server shall terminate the MODE SELECT command with CHECK CONDITION status, set the sense key to ILLEGAL REQUEST, set the additional sense code to INVALID FIELD IN PARAMETER LIST, and shall not change any mode parameters for the following conditions:

a) If the application client sets any field that is reported as not changeable by the device server to a value other than its current value;
b) If the application client sets any field in the mode parameter header or block descriptor(s) to an unsupported value;
c) If an application client sends a mode page with a page length not equal to the page length returned by the MODE SENSE command for that page;
d) If the application client sends a unsupported value for a mode parameter and rounding is not implemented for that mode parameter; or
e) If the application client sets any reserved field in the mode parameter list to a non-zero value.

If the application client sends a value for a mode parameter that is outside the range supported by the device server and rounding is implemented for that mode parameter, the device server handles the condition by either:

a) rounding the parameter to an acceptable value and terminate the command as described in 5.3; or
b) terminating the command with CHECK CONDITION status, the sense key set to ILLEGAL REQUEST, and set the additional sense code to INVALID FIELD IN PARAMETER LIST.

A device server may alter any mode parameter in any mode page, even those reported as non-changeable, as a result of changes to other mode parameters.

The device server validates the non-changeable mode parameters against the current values that existed for those mode parameters prior to the MODE SELECT command.

NOTE 21 The current values calculated by the device server may affect the application client's operation. The application client may issue a MODE SENSE command after each MODE SELECT command, to determine the current values.

7.7 MODE SELECT(10) command

The MODE SELECT(10) command (see table 61) provides a means for the application client to specify medium, logical unit, or peripheral device parameters to the device server. See the MODE SELECT(6) command (7.6) for a description of the fields and operation of this command. Application clients should issue MODE SENSE(10) prior to each MODE SELECT(10) to determine supported mode pages, mode page lengths, and other parameters. Device servers that implement the MODE SELECT(10) command shall also implement the MODE SENSE(10) command.

Table 61 — MODE SELECT(10) command

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

7.8 MODE SENSE(6) command

7.8.1 MODE SENSE(6) command introduction

The MODE SENSE(6) command (see table 62) provides a means for a device server to report parameters to an application client. It is a complementary command to the MODE SELECT(6) command. Device servers that implement the MODE SENSE(6) command shall also implement the MODE SELECT(6) command.

Table 62 — MODE SENSE(6) command

Bit / Byte	7	6	5	4	3	2	1	0
0	Operation code (1Ah)							
1	Reserved				DBD	Reserved		
2	PC		PAGE CODE					
3	Reserved							
4	ALLOCATION LENGTH							
5	CONTROL							

A disable block descriptors (DBD) bit of zero indicates that the device server may return zero or more block descriptors in the returned MODE SENSE data (see 8.3). A DBD bit of one specifies that the device server shall not return any block descriptors in the returned MODE SENSE data.

The page control (PC) field defines the type of mode parameter values to be returned in the mode pages. The PC field is defined in table 63.

Table 63 — Page control field

Code	Type of parameter	Reference
00b	Current values	7.8.2
01b	Changeable values	7.8.3
10b	Default values	7.8.4
11b	Saved values	7.8.5

The PC field only affects the mode parameters within the mode pages, however the PS bit, PAGE CODE and PAGE LENGTH fields should return current values since they have no meaning when used with other types. The mode parameter header and mode parameter block descriptor should return current values.

Some SCSI devices may not distinguish between current and saved mode parameters and report identical values in response to a PC field of either 00b or 11b. See also the description of the save pages (SP) bit in the MODE SELECT command.

The PAGE CODE field specifies which mode page(s) to return. Mode page code usage is defined in table 64.

Table 64 — Mode page code usage for all devices

Page Code	Description
00h	Vendor specific (does not require page format)
01h - 1Fh	See specific device-types
20h - 3Eh	Vendor specific (page format required)
3Fh	Return all mode pages

An application client may request any one or all of the supported mode pages from the device server. If an application client issues a MODE SENSE command with a page code value not implemented by the target, the device server shall return CHECK CONDITION status and shall set the sense key to ILLEGAL REQUEST and the additional sense code to INVALID FIELD IN CDB.

A page code of 3Fh indicates that all mode pages implemented by the target shall be returned to the application client. If the mode parameter list exceeds 256 bytes for a MODE SENSE(6) command or 65 536 bytes for a MODE SENSE(10) command, the device server shall return CHECK CONDITION status and the sense key shall be set to ILLEGAL REQUEST and the additional sense code set to INVALID FIELD IN CDB.

Mode page 00h, if implemented, shall be returned after all other mode pages.

Mode pages should be returned in ascending page code order except for mode page 00h.

If the PC field and the PAGE CODE field are both set to zero the device server should return a mode parameter header and block descriptor (if applicable).

The mode parameter list for all device types for MODE SELECT and MODE SENSE is defined in 8.3. Parts of the mode parameter list are specifically defined for each device type. Definitions for the parts of each mode parameter list that are unique for each device-type may be found in the applicable command standards (see 3.1.12).

7.8.2 Current values

A PC field value of 00b requests that the device server return the current values of the mode parameters. The current values returned are:

a) the current values of the mode parameters established by the last successful MODE SELECT command;
b) the saved values of the mode parameters if a MODE SELECT command has not successfully completed since the last power-on or hard reset condition; or
c) the default values of the mode parameters, if saved values, are not available or not supported.

7.8.3 Changeable values

A PC field value of 01b requests that the device server return a mask denoting those mode parameters that are changeable. In the mask, the fields of the mode parameters that are changeable shall be set to all one bits and the fields of the mode parameters that are non-changeable (i.e., defined by the target) shall be set to all zero bits.

Implementation of changeable page parameters is optional. If the target does not implement changeable parameters pages and the device server receives a MODE SENSE command with 01b in the PC field, the command shall be terminated with CHECK CONDITION status. The sense key shall be set to ILLEGAL REQUEST, and the additional sense code set to INVALID FIELD IN CDB.

An attempt to change a non-changeable mode parameter using the MODE SELECT command shall result in an error condition (see 7.6).

The application client should issue a MODE SENSE command with the PC field set to 01b and the PAGE CODE field set to 3Fh to determine which mode pages are supported, which mode parameters within the mode pages are changeable, and the supported length of each mode page prior to issuing any MODE SELECT commands.

7.8.4 Default values

A PC field value of 10b requests that the device server return the default values of the mode parameters. Unsupported parameters shall be set to zero. Default values should be accessible even if the device is not ready.

7.8.5 Saved values

A PC field value of 11b requests that the device server return the saved values of the mode parameters. Implementation of saved page parameters is optional. Mode parameters not supported by the target shall be set to zero. If saved values are not implemented, the command shall be terminated with CHECK CONDITION status, the sense key set to ILLEGAL REQUEST and the additional sense code set to SAVING PARAMETERS NOT SUPPORTED.

The method of saving parameters is vendor specific. The parameters are preserved in such a manner that they are retained when the device is powered down. All savable pages should be considered saved when a MODE SELECT command issued with the SP bit set to one has returned a GOOD status or after the successful completion of a FORMAT UNIT command.

7.8.6 Initial responses

After a power-up condition or hard reset condition, the device server shall respond in the following manner:

a) If default values are requested, report the default values;
b) If saved values are requested, report valid restored mode parameters, or restore the mode parameters and report them. If the saved values of the mode parameters are not able to be accessed from the nonvolatile vendor specific location, terminate the command with CHECK CONDITION status and set the sense key set to NOT READY. If saved parameters are not implemented respond as defined in 7.8.5; or
c) If current values are requested and the current values of the mode parameters have not been sent by the application client (via a MODE SELECT command), the device server may return either the default or saved values, as defined above. If current values have been sent, the current values shall be reported.

7.9 MODE SENSE(10) command

The MODE SENSE(10) command (see table 65) provides a means for a device server to report parameters to an application client. It is a complementary command to the MODE SELECT(10) command. Device servers that implement the MODE SENSE(10) command shall also implement the MODE SELECT(10) command.

Table 65 — MODE SENSE(10) command

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

If the Long LBA Accepted (LLBAA) bit is one, the device server is allowed to return parameter data with the LONGLBA bit equal to one (see 8.3.3). If LLBAA is zero, the LONGLBA bit shall be zero in the parameter data returned by the device server.

See the MODE SENSE(6) command (7.8) for a description of the other fields and operation of this command.

7.10 PERSISTENT RESERVE IN command

7.10.1 PERSISTENT RESERVE IN command introduction

The PERSISTENT RESERVE IN command (see table 66) is used to obtain information about persistent reservations and reservation keys that are active within a device server. This command is used in conjunction with the PERSISTENT RESERVE OUT command (see 7.11).

Table 66 — PERSISTENT RESERVE IN command

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

The PERSISTENT RESERVE IN parameter data includes a field that indicates the number of parameter data bytes available to be returned. The ALLOCATION LENGTH field in the CDB indicates how much space has been allocated for the returned parameter list. An allocation length that is not sufficient to contain the entire parameter list shall not be considered an error. If the complete list is required, the application client should send a new PERSISTENT RESERVE IN command with allocation length large enough to contain the entire list.

7.10.2 PERSISTENT RESERVE IN service actions

7.10.2.1 Summary of PERSISTENT RESERVE IN service actions

The service action codes for the PERSISTENT RESERVE IN command are defined in table 67.

Table 67 — PERSISTENT RESERVE IN service action codes

Code	Name	Description
00h	READ KEYS	Reads all registered Reservation Keys
01h	READ RESERVATION	Reads the current persistent reservations
02h - 1Fh	Reserved	Reserved

7.10.2.2 Read Keys

The READ KEYS service action requests that the device server return a parameter list containing a header and a list of each currently registered initiator's reservation key. If multiple initiators have registered with the same key, then that key value shall be listed multiple times, once for each such registration.

For more information on READ KEYS, see 5.5.3.3.2.

7.10.2.3 Read Reservations

The READ RESERVATIONS service action requests that the device server return a parameter list containing a header and the persistent reservations, if any, that are present in the device server. Multiple persistent reservations may be returned only if element reservations are present.

For more information on READ RESERVATION see 5.5.3.3.3.

7.10.3 PERSISTENT RESERVE IN parameter data for READ KEYS

The format for the parameter data provided in response to a PERSISTENT RESERVE IN command with the READ KEYS service action is shown in table 68.

Table 68 — PERSISTENT RESERVE IN parameter data for READ KEYS

Bit / Byte	7	6	5	4	3	2	1	0
0	(MSB)			GENERATION				
3								(LSB)
4	(MSB)			ADDITIONAL LENGTH (n-7)				
7								(LSB)
	Reservation key list							
8	(MSB)			First reservation key				
15								(LSB)
	. . .							
n-7	(MSB)			Last reservation key				
n								(LSB)

The GENERATION field shall contain a 32-bit counter maintained by the device server that shall be incremented every time a PERSISTENT RESERVE OUT command requests a REGISTER, a REGISTER AND IGNORE EXISTING KEY, a CLEAR, a PREEMPT, or a PREEMPT AND ABORT service action. The counter shall not be incremented by a PERSISTENT RESERVE IN command, by a PERSISTENT RESERVE OUT command that performs a RESERVE or RELEASE service action, or by a PERSISTENT RESERVE OUT command that is terminated due to an error or reservation conflict. Regardless of the APTPL bit value the generation value shall be set to zero as part of the power on reset process.

The ADDITIONAL LENGTH field contains a count of the number of bytes in the Reservation key list. If the allocation length specified by the PERSISTENT RESERVE IN command is not sufficient to contain the entire parameter list, then only the first portion of the list (byte 0 to the allocation length) shall be sent to the application client. The incremental remaining bytes shall be truncated, although the ADDITIONAL LENGTH field shall still contain the actual number of bytes in the reservation key list without consideration of any truncation resulting from an insufficient allocation length. This shall not be considered an error.

The reservation key list contains the 8-byte reservation keys for all initiators that have registered through all ports with the device server.

7.10.4 PERSISTENT RESERVE IN parameter data for READ RESERVATION

7.10.4.1 Format of PERSISTENT RESERVE IN parameter data for READ RESERVATION

The format for the parameter data provided in response to a PERSISTENT RESERVE IN command with the READ RESERVATION service action is shown in table 69.

Table 69 — PERSISTENT RESERVE IN parameter data for READ RESERVATION

Bit / Byte	7	6	5	4	3	2	1	0
0	(MSB)							
3				GENERATION				(LSB)
4	(MSB)							
7				ADDITIONAL LENGTH (n-7)				(LSB)
8	(MSB)							
n				Reservation descriptor(s) (see table 70)				(LSB)

The GENERATION field shall be as defined for the PERSISTENT RESERVE IN READ KEYS parameter data (see 7.10.3).

The ADDITIONAL LENGTH field contains a count of the number of bytes to follow in reservation descriptor(s). If the allocation length specified by the PERSISTENT RESERVE IN command is not sufficient to contain the entire parameter list, then only the first portion of the list (byte 0 to the allocation length) shall be sent to the application client. The incremental remaining bytes shall be truncated, although the ADDITIONAL LENGTH field shall still contain the actual number of bytes of reservation descriptor(s) and shall not be affected by the truncation. This shall not be considered an error.

The format of the reservation descriptors is defined in table 70. There shall be a reservation descriptor for the persistent reservation, if any, present in the logical unit and a reservation descriptor for each element, if any, having a persistent reservation.

Table 70 — PERSISTENT RESERVE IN reservation descriptor

Bit / Byte	7	6	5	4	3	2	1	0
0	(MSB)							
7				RESERVATION KEY				(LSB)
8	(MSB)							
11				SCOPE-SPECIFIC ADDRESS				(LSB)
12	Reserved							
13	SCOPE				TYPE			
14				Obsolete				
15								

If a persistent reservation is present in the logical unit that does not contain elements, there shall be a single reservation descriptor in the list of parameter data returned by the device server in response to the PERSISTENT RESERVE IN command with a READ RESERVATION service action. The reservation descriptor for each reservation shall contain the RESERVATION KEY under which the persistent reservation is held. The TYPE and SCOPE of each persistent reservation as present in the PERSISTENT RESERVE OUT command that created the persistent reservation shall be returned (see 7.10.4.2 and 7.10.4.3).

If a persistent reservation is present in the logical unit that does contain elements, there shall be a reservation descriptor in the list of parameter data returned by the device server in response to the PERSISTENT RESERVE

IN command with a READ RESERVATION service action for the LU_SCOPE persistent reservation that is held, if any, and each ELEMENT_SCOPE persistent reservation that may be held. The reservation descriptor shall contain the RESERVATION KEY under which the persistent reservation is held. The TYPE and SCOPE of the persistent reservation as present in the PERSISTENT RESERVE OUT command that created the persistent reservation shall be returned (see 7.10.4.2 and 7.10.4.3).

If the SCOPE is an ELEMENT_SCOPE reservation, the SCOPE-SPECIFIC ADDRESS field shall contain the element address, zero filled in the most significant bits to fit the field. If the SCOPE is a LU_SCOPE reservation, the SCOPE-SPECIFIC ADDRESS field shall be set to zero. The obsolete field in Bytes 14 and 15 was defined in a previous standard.

7.10.4.2 Persistent reservations Scope

7.10.4.2.1 Summary of persistent reservations Scope

The value in the SCOPE field shall indicate whether a persistent reservation applies to an entire logical unit or to an element. The values in the SCOPE field are defined in table 71.

Table 71 — Persistent reservation scope codes

Code	Name	Description
0h	LU_SCOPE	Persistent reservation applies to the full logical unit
1h		Obsolete
2h	ELEMENT_SCOPE	Persistent reservation applies to the specified element
3h - Fh	Reserved	Reserved

7.10.4.2.2 Logical unit scope

A SCOPE field value of LU_SCOPE shall indicate that the persistent reservation applies to the entire logical unit. The LU_SCOPE scope shall be implemented by all device servers that implement PERSISTENT RESERVE OUT.

7.10.4.2.3 Element scope

A SCOPE field value of ELEMENT_SCOPE shall indicate that the persistent reservation applies to the element of the logical unit defined by the SCOPE-SPECIFIC ADDRESS field in the PERSISTENT RESERVE OUT parameter list. An element is defined by the SMC-2 standard. The ELEMENT_SCOPE scope is optional for all device servers that implement PERSISTENT RESERVE OUT.

7.10.4.3 Persistent Reservations Type

The value in the TYPE field shall specify the characteristics of the persistent reservation being established for all data blocks within the element or within the logical unit. Table 72 defines the characteristics of the different type values. For each persistent reservation type, table 72 lists code value and describes the required device server support. In table 72, the description of required device server support is divided into two paragraphs. The first paragraph defines the required handling for read operations. The second paragraph defines the required handling for write operations.

Table 72 — Persistent reservation type codes

Code	Name	Description
0h		Obsolete
1h	Write Exclusive	**Reads Shared**: Any application client on any initiator may initiate tasks that request transfers from the storage medium or cache of the logical unit to the initiator. **Writes Exclusive**: Any task from any initiator other than the initiator holding the persistent reservation that requests a transfer from the initiator to the storage medium or cache of the logical unit shall be terminated with RESERVATION CONFLICT status.
2h		Obsolete
3h	Exclusive Access	**Reads Exclusive**: Any task from any initiator other than the initiator holding the persistent reservation that requests a transfer from the storage medium or cache of the logical unit to the initiator shall be terminated with RESERVATION CONFLICT status. **Writes Exclusive**: Any task from any initiator other than the initiator holding the persistent reservation that requests a transfer from the initiator to the storage medium or cache of the logical unit shall be terminated with RESERVATION CONFLICT status.
4h		Obsolete
5h	Write Exclusive – Registrants Only	**Reads Shared**: Any application client on any initiator may initiate tasks that request transfers from the storage medium or cache of the logical unit to the initiator. **Writes Exclusive:** A task that requests a transfer to the storage medium or cache of the logical unit from an initiator that is not currently registered with the device server shall be terminated with RESERVATION CONFLICT status.
6h	Exclusive Access – Registrants Only	**Reads Exclusive**: A task that requests a transfer from the storage medium or cache of the logical unit to an initiator that is not currently registered with the device server shall be terminated with RESERVATION CONFLICT status. **Writes Exclusive:** A task that requests a transfer to the storage medium or cache of the logical unit from an initiator that is not currently registered with the device server shall be terminated with RESERVATION CONFLICT status.
7h - Fh	Reserved	

7.11 PERSISTENT RESERVE OUT command

7.11.1 PERSISTENT RESERVE OUT command introduction

The PERSISTENT RESERVE OUT command (see table 73) is used to request service actions that reserve a logical unit or element for the exclusive or shared use of a particular initiator. The command uses other service actions to manage and remove such reservations. The command shall be used in conjunction with the PERSISTENT RESERVE IN command and shall not be used with the RESERVE and RELEASE commands.

Initiators performing PERSISTENT RESERVE OUT service actions are identified by a reservation key provided by the application client. An application client may use the PERSISTENT RESERVE IN command to obtain the reservation key for the initiator holding a persistent reservation and may use the PERSISTENT RESERVE OUT command to preempt that reservation.

Table 73 — PERSISTENT RESERVE OUT command

Bit / Byte	7	6	5	4	3	2	1	0
0	OPERATION CODE (5Fh)							
1	Reserved			SERVICE ACTION				
2	SCOPE				TYPE			
3	Reserved							
4	Reserved							
5	Reserved							
6	Reserved							
7	(MSB)			PARAMETER LIST LENGTH (18h)				
8								(LSB)
9	CONTROL							

If a PERSISTENT RESERVE OUT command is attempted, but there are insufficient device server resources to complete the operation, the device server shall return a CHECK CONDITION status. The sense key shall be set to ILLEGAL REQUEST and the additional sense data shall be set to INSUFFICIENT REGISTRATION RESOURCES.

The PERSISTENT RESERVE OUT command contains fields that specify a persistent reservation service action, the intended scope of the persistent reservation, and the restrictions caused by the persistent reservation. The TYPE and SCOPE fields are defined in 7.10.4.2 and 7.10.4.3. If a SCOPE field specifies a scope that is not implemented, the device server shall return a CHECK CONDITION status. The sense key shall be set to ILLEGAL REQUEST and additional sense data shall be set to INVALID FIELD IN CDB.

Fields contained in the PERSISTENT RESERVE OUT parameter list specify the information required to perform a particular persistent reservation service action.

The parameter list shall be 24 bytes in length and the PARAMETER LIST LENGTH field shall contain 24 (18h). If the parameter list length is not 24, the device server shall return a CHECK CONDITION status. The sense key shall be set to ILLEGAL REQUEST and the additional sense data shall be set to PARAMETER LIST LENGTH ERROR.

7.11.2 PERSISTENT RESERVE OUT Service Actions

When processing the PERSISTENT RESERVE OUT service actions, the device server shall increment the generation value as specified in 7.10.3.

The PERSISTENT RESERVE OUT command service actions are defined in table 74.

Table 74 — PERSISTENT RESERVE OUT service action codes

Code	Name	Description	GENERATION field incremented (see 7.10.3)
00h	REGISTER	Register a reservation key with the device server (see 5.5.3.4).	Yes
01h	RESERVE	Creates a persistent reservation having a specified SCOPE and TYPE (see 5.5.3.5). The SCOPE and TYPE of a persistent reservation are defined in 7.10.4.2 and 7.10.4.3.	No
02h	RELEASE	Releases the selected reservation for the requesting initiator (see 5.5.3.6.2).	No
03h	CLEAR	Clears all reservation keys and all persistent reservations (see 5.5.3.6.5).	Yes
04h	PREEMPT	Preempts persistent reservations from another initiator (see 5.5.3.6.3).	Yes
05h	PREEMPT AND ABORT	Preempts persistent reservations from another initiator and aborts all tasks for all initiators registered with the specified reservation key (see 5.5.3.6.3 and 5.5.3.6.4).	Yes
06h	REGISTER AND IGNORE EXISTING KEY	Register a reservation key with the device server (see 5.5.3.4).	Yes
07h - 1Fh	Reserved		

The parameter list values for each service action are specified in 7.11.3.

7.11.3 PERSISTENT RESERVE OUT parameter list

The parameter list required to perform the PERSISTENT RESERVE OUT command is defined in table 75. All fields shall be sent on all PERSISTENT RESERVE OUT commands, even if the field is not required for the specified service action and scope values.

Table 75 — PERSISTENT RESERVE OUT parameter list

Bit / Byte	7	6	5	4	3	2	1	0
0	(MSB)			RESERVATION KEY				
7								(LSB)
8	(MSB)			SERVICE ACTION RESERVATION KEY				
15								(LSB)
16	(MSB)			SCOPE-SPECIFIC ADDRESS				
19								(LSB)
20				Reserved				APTPL
21				Reserved				
22	(MSB)			Obsolete				
23								(LSB)

The obsolete field in Bytes 22 and 23 was defined in a previous standard for use with an obsolete scope (see table 71). If the obsolete scope is not supported Bytes 22 and 23 should be zero.

The RESERVATION KEY field contains an 8-byte value provided by the application client to the device server to identify the initiator that is the source of the PERSISTENT RESERVE OUT command. The device server shall verify that the contents of the RESERVATION KEY field in a PERSISTENT RESERVE OUT command parameter data matches the registered reservation key for the initiator from which the task was received, except for:

a) the REGISTER AND IGNORE EXISTING KEY service action where the RESERVATION KEY field shall be ignored; and
b) the REGISTER service action for an unregistered initiator where the RESERVATION KEY field shall contain zero.

Except as noted above, when a PERSISTENT RESERVE OUT command specifies a RESERVATION KEY field other than the reservation key registered for the initiator the device server shall return a RESERVATION CONFLICT status. Except as noted above, the reservation key of the initiator shall be verified to be correct regardless of the SERVICE ACTION and SCOPE field values.

The SERVICE ACTION RESERVATION KEY field contains information needed for four service actions; the REGISTER, REGISTER AND IGNORE EXISTING KEY, PREEMPT, and PREEMPT AND ABORT service actions. For the REGISTER and REGISTER AND IGNORE EXISTING KEY service action, the SERVICE ACTION RESERVATION KEY field contains the new reservation key to be registered. For the PREEMPT and PREEMPT AND ABORT service actions, the SERVICE ACTION RESERVATION KEY field contains the reservation key of the persistent reservations that are being preempted. The SERVICE ACTION RESERVATION KEY field is ignored for all other service actions.

If the scope is an ELEMENT_SCOPE reservation, the SCOPE-SPECIFIC ADDRESS field shall contain the element address, zero filled in the most significant bits to fit the field. If the service action is REGISTER, REGISTER AND IGNORE EXISTING KEY, or CLEAR or if the scope is a LU_SCOPE reservation, the SCOPE-SPECIFIC ADDRESS field shall be set to zero.

The Activate Persist Through Power Loss (APTPL) bit shall be valid only for the REGISTER, or the REGISTER AND IGNORE EXISTING KEY service action. In all other cases, the APTPL bit shall be ignored. Support for an APTPL bit equal to one is optional. If a device server that does not support the APTPL bit value of one receives that value in a REGISTER or a REGISTER AND IGNORE EXISTING KEY service action, the device server shall return a CHECK

CONDITION status. The sense key shall be set to ILLEGAL REQUEST and additional sense code shall be set to INVALID FIELD IN PARAMETER LIST.

If the last valid APTPL bit value received by the device server is zero, the loss of power in the target shall release the persistent reservation for all logical units and remove all reservation keys (see 5.5.3.4). If the last valid APTPL bit value received by the device server is one, the logical unit shall retain any persistent reservation(s) that may be present and all reservation keys for all initiators even if power is lost and later returned (see 5.5.3.2).

Table 76 summarizes which fields are set by the application client and interpreted by the device server for each service action and scope value. The APTPL bit in the PERSISTENT RESERVE OUT parameter list, specified in the previous paragraph, is not summarized in table 76.

Table 76 — PERSISTENT RESERVE OUT service actions and valid parameters

Service action	Allowed SCOPE	Parameters			
		TYPE	RESERVATION KEY	SERVICE ACTION RESERVATION KEY	SCOPE-SPECIFIC ADDRESS
REGISTER	ignored	ignored	valid	valid	ignored
REGISTER AND IGNORE EXISTING KEY	ignored	ignored	ignored	valid	ignored
RESERVE	LU_SCOPE ELEMENT_SCOPE	valid valid	valid	ignored ignored	ignored valid (element)
RELEASE	LU_SCOPE ELEMENT_SCOPE	valid valid	valid	ignored ignored	ignored valid (element)
CLEAR	ignored	ignored	valid	ignored	ignored
PREEMPT	LU_SCOPE ELEMENT_SCOPE	valid valid	valid	valid valid	ignored valid (element)
PREEMPT & ABORT	LU_SCOPE ELEMENT_SCOPE	valid valid	valid	valid valid	ignored valid (element)

7.12 PREVENT ALLOW MEDIUM REMOVAL command

The PREVENT ALLOW MEDIUM REMOVAL command (see table 77) requests that the target enable or disable the removal of the medium in the logical unit. The logical unit shall not allow medium removal if any initiator currently has medium removal prevented.

Table 77 — PREVENT ALLOW MEDIUM REMOVAL command

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

Table 78 defines the PREVENT field values and their meanings.

Table 78 — PREVENT ALLOW MEDIUM REMOVAL PREVENT field

PREVENT	Description
00b	Medium removal shall be allowed from both the data transport element and the attached medium changer (if any).
01b	Medium removal shall be prohibited from the data transport element but allowed from the attached medium changer (if any).
10b	Medium removal shall be allowed for the data transport element but prohibited for the attached medium changer.
11b	Medium removal shall be prohibited for both the data transport element and the attached medium changer.

PREVENT values 10b and 11b are valid only when the RMB bit and the MCHNGR bit are both equal to one in the standard INQUIRY data.

The prevention of medium removal shall begin when any application client issues a PREVENT ALLOW MEDIUM REMOVAL command with a PREVENT field of 01b or 11b (i.e., medium removal prevented). The prevention of medium removal for the logical unit shall terminate:

a) after all initiators with application clients that previously prevented medium removal issue PREVENT ALLOW MEDIUM REMOVAL commands with a PREVENT field of 00b or 10b, and the device server has successfully performed a synchronize cache operation; or
b) upon a hard reset condition.

For an initiator that has executed a PERSISTENT RESERVE OUT command with a service action of RESERVE, REGISTER AND IGNORE EXISTING KEY, or REGISTER service action, the PREVENT field shall be set to zero as part of the uninterrupted sequence of events performed by a PERSISTENT RESERVE OUT command with a service action of PREEMPT AND ABORT using that initiator's registration value in the SERVICE ACTION RESERVATION KEY field. This allows an initiator to override the prevention of medium removal function for an initiator that is no longer operating correctly.

While a prevention of medium removal condition is in effect, the target shall inhibit mechanisms that normally allow removal of the medium by an operator.

7.13 READ BUFFER command

7.13.1 READ BUFFER command introduction

The READ BUFFER command (see table 79) is used in conjunction with the WRITE BUFFER command as a diagnostic function for testing memory in the SCSI device and the integrity of the service delivery subsystem. This command shall not alter the medium.

Table 79 — READ BUFFER command

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

The function of this command and the meaning of fields within the CDB depend on the contents of the MODE field. The MODE field is defined in table 80.

Table 80 — READ BUFFER MODE field

MODE	Description	Implementation requirements
0000b	Combined header and data	Optional
0001b	Vendor specific	Vendor specific
0010b	Data	Optional
0011b	Descriptor	Optional
0100b - 1001b	Reserved	Reserved
1010b	Echo buffer	Optional
1011b	Echo buffer descriptor	Optional
1100b - 1111b	Reserved	Reserved

7.13.2 Combined header and data mode (0000b)

In this mode, a four-byte header followed by data bytes is returned to the application client in the Data-In Buffer. The BUFFER ID and the BUFFER OFFSET fields are reserved.

The four-byte READ BUFFER header (see table 81) is followed by data bytes from the buffer.

Table 81 — READ BUFFER header

Bit / Byte	7	6	5	4	3	2	1	0
0	Reserved							
1	BUFFER CAPACITY							
3								

The BUFFER CAPACITY field specifies the total number of data bytes available in the buffer. This number is not reduced to reflect the allocation length; nor is it reduced to reflect the actual number of bytes written using the WRITE BUFFER command. Following the READ BUFFER header, the device server shall transfer data from the buffer. The device server shall terminate filling the Data-In Buffer when allocation length bytes of header plus data have been transferred or when all available header and buffer data have been transferred to the application client, whichever is less.

7.13.3 Vendor specific mode (0001b)

In this mode, the meanings of the BUFFER ID, BUFFER OFFSET, and ALLOCATION LENGTH fields are not specified by this standard.

7.13.4 Data mode (0010b)

In this mode, the Data-In Buffer is filled only with logical unit buffer data. The BUFFER ID field identifies a specific buffer within the logical unit from which data shall be transferred. The vendor assigns buffer ID codes to buffers within the logical unit. Buffer ID zero shall be supported. If more than one buffer is supported, additional buffer ID codes shall be assigned contiguously, beginning with one. Buffer ID code assignments for the READ BUFFER command shall be the same as for the WRITE BUFFER command. If an unsupported buffer ID code is selected, the device server shall return CHECK CONDITION status, shall set the sense key to ILLEGAL REQUEST, and set the additional sense code to INVALID FIELD IN CDB.

The device server shall terminate filling the Data-In Buffer when allocation length bytes have been transferred or when all the available data from the buffer has been transferred to the application client, whichever amount is less.

The BUFFER OFFSET field contains the byte offset within the specified buffer from which data shall be transferred. The application client should conform to the offset boundary requirements returned in the READ BUFFER descriptor (see 7.13.5). If the device server is unable to accept the specified buffer offset, it shall return CHECK CONDITION status, shall set the sense key to ILLEGAL REQUEST, and set the additional sense code to INVALID FIELD IN CDB.

7.13.5 Descriptor mode (0011b)

In this mode, a maximum of four bytes of READ BUFFER descriptor information is returned. The device server shall return the descriptor information for the buffer specified by the BUFFER ID field (see the description of the buffer ID in 7.13.4). If there is no buffer associated with the specified buffer ID, the device server shall return all zeros in the READ BUFFER descriptor. The BUFFER OFFSET field is reserved in this mode. The allocation length should be set to four or greater. The device server shall transfer the lesser of the allocation length or four bytes of READ BUFFER descriptor. The READ BUFFER descriptor is defined as shown in table 82.

Table 82 — READ BUFFER descriptor

Bit / Byte	7	6	5	4	3	2	1	0
0	OFFSET BOUNDARY							
1	BUFFER CAPACITY							
2								
3								

The OFFSET BOUNDARY field returns the boundary alignment within the selected buffer for subsequent WRITE BUFFER and READ BUFFER commands. The value contained in the OFFSET BOUNDARY field shall be interpreted as a power of two.

The value contained in the BUFFER OFFSET field of subsequent WRITE BUFFER and READ BUFFER commands should be a multiple of $2^{\text{offset boundary}}$ as shown in table 83.

Table 83 — Buffer offset boundary

Offset boundary	$2^{\text{Offset boundary}}$	Buffer offsets
0h	$2^0 = 1$	Byte boundaries
1h	$2^1 = 2$	Even-byte boundaries
2h	$2^2 = 4$	Four-byte boundaries
3h	$2^3 = 8$	Eight-byte boundaries
4h	$2^4 = 16$	16-byte boundaries
.	.	.
FFh	Not applicable	0 is the only supported buffer offset

The BUFFER CAPACITY field shall return the size of the selected buffer in bytes.

NOTE 22 In a system employing multiple application clients, a buffer may be altered between the WRITE BUFFER and READ BUFFER commands by another application client. Buffer testing applications should insure that only a single application client is active. Use of reservations to all logical units on the device or linked commands may be helpful in avoiding buffer alteration between these two commands.

7.13.6 Read Data from echo buffer (1010b)

In this mode the device server transfers data to the application client from the echo buffer. The echo buffer shall transfer the same data as when the WRITE BUFFER command with the mode field set to echo buffer was issued. The BUFFER ID and BUFFER OFFSET fields are ignored in this mode.

The READ BUFFER command shall return up to the number of bytes of data as received in the prior echo buffer mode WRITE BUFFER command from the same initiator. If the allocation length is insufficient to accommodate the number of bytes of data as received in the prior echo buffer mode WRITE BUFFER command, the data returned shall be truncated as described in 4.3.4.6, and this shall not be considered an error. If a prior echo buffer mode WRITE BUFFER command was not successfully completed the echo buffer mode READ BUFFER command shall terminate with a CHECK CONDITION status, the sense key shall be set to ILLEGAL REQUEST and the additional sense code to COMMAND SEQUENCE ERROR. If the data in the echo buffer has been overwritten by another initiator the target shall terminate the echo buffer mode READ BUFFER command with a

CHECK CONDITION status, the sense key shall be set to ABORTED COMMAND and the additional sense code to ECHO BUFFER OVERWRITTEN.

The initiator may send a READ BUFFER command requesting the echo buffer descriptor prior to a WRITE BUFFER command.

If an echo buffer mode WRITE BUFFER command is successful then the initiator may send multiple echo buffer mode READ BUFFER commands to read the echo buffer data multiple times.

7.13.7 Echo buffer descriptor mode (1011b)

In this mode, a maximum of four bytes of READ BUFFER descriptor information is returned. The device server shall return the descriptor information for the echo buffer. If there is no echo buffer implemented, the device server shall return all zeros in the READ BUFFER descriptor. The BUFFER OFFSET field is reserved in this mode. The allocation length should be set to four or greater. The device server shall transfer the lesser of the allocation length or four bytes of READ BUFFER descriptor. The READ BUFFER descriptor is defined as shown in table 84.

Table 84 — Echo buffer descriptor

Bit / Byte	7	6	5	4	3	2	1	0
0	Reserved							EBOS
1	Reserved							
2	Reserved			(MSB)				
3	BUFFER CAPACITY							(LSB)

The BUFFER CAPACITY field shall return the size of the echo buffer in bytes aligned to a four-byte boundary. The maximum echo buffer size is 4096 bytes.

If the echo buffer is implemented then the echo buffer descriptor shall be implemented.

An echo buffer overwritten supported (EBOS) bit of one indicates either:

a) the target returns the ECHO BUFFER OVERWRITTEN additional sense code if the data being read from the echo buffer is not the data previously written by the same initiator, or
b) the target ensures echo buffer data from each initiator is the same as that previously written by the same initiator.

An EBOS bit of zero specifies that the echo buffer may be overwritten by other initiators or intervening commands.

7.14 RECEIVE COPY RESULTS command

7.14.1 RECEIVE COPY RESULTS command introduction

The RECEIVE COPY RESULTS command (see table 85) provides a means for the application client to receive information about the copy manager or the results of a previous or current EXTENDED COPY command (see 7.2).

Table 85 — RECEIVE COPY RESULTS command

Bit / Byte	7	6	5	4	3	2	1	0
0	OPERATION CODE (84h)							
1	Reserved			SERVICE ACTION				
2	LIST IDENTIFIER							
3	Reserved							
4	Reserved							
5	Reserved							
6	Reserved							
7	Reserved							
8	Reserved							
9	Reserved							
10	(MSB)							
11				ALLOCATION LENGTH				
12								
13								(LSB)
14	Reserved							
15	CONTROL							

The service actions defined for the RECEIVE COPY RESULTS command are shown in table 86.

Table 86 — RECEIVE COPY RESULTS service action codes

Code	Name	Description	Returns Data While EXTENDED COPY Is In Progress
00h	COPY STATUS	Return the current copy status of the EXTENDED COPY command identified by the LIST IDENTIFIER field.	Yes
01h	RECEIVE DATA	Return the held data read by EXTENDED COPY command identified by the LIST IDENTIFIER field.	No
03h	OPERATING PARAMETERS	Return copy manager operating parameters.	Yes
04h	FAILED SEGMENT DETAILS	Return copy target device sense data and other information about the progress of processing a segment descriptor whose processing was not completed during processing of the EXTENDED COPY command identified by the LIST IDENTIFIER field.	No
05h-1Eh	Reserved		
1Fh	Vendor Specific		

The LIST IDENTIFIER field identifies the EXTENDED COPY command (see 7.2) about which information is desired. The RECEIVE COPY RESULTS command shall return information from the EXTENDED COPY command received from the same initiator with a list identifier that matches the list identifier specified in the RECEIVE COPY RESULTS CDB. If no EXTENDED COPY command known to the copy manager has a matching list identifier, then the command shall be terminated with CHECK CONDITION status, the sense key shall be set to ILLEGAL REQUEST, and the additional sense code shall be set to INVALID FIELD IN CDB.

If the LIST IDENTIFIER field identifies the EXTENDED COPY command that had the NRCR bit set to 1 in the parameter data (see 7.2), the copy manager may respond to a RECEIVE COPY RESULTS command in the same manner it would if the EXTENDED COPY command had never been received.

The actual length of the RECEIVE COPY RESULTS parameter data is available in the AVAILABLE DATA parameter data field. The ALLOCATION LENGTH field in the CDB indicates how much space has been allocated for the returned parameter list. If the length is not sufficient to contain the entire parameter list, the first portion of the list shall be returned. This shall not be considered an error. If the remainder of the list is required, the application client should send a new RECEIVE COPY RESULTS command with an ALLOCATION LENGTH field large enough to contain the entire parameter list.

7.14.2 COPY STATUS service action

In response to the COPY STATUS service action, the copy manager shall return the current status of the EXTENDED COPY command (see 7.2) identified by the LIST IDENTIFIER field in the CDB. Table 87 shows the format of the information returned by the copy manager in response to the COPY STATUS service action. If a device server supports the EXTENDED COPY command, it shall also support the RECEIVE COPY RESULTS command with COPY STATUS service action.

Table 87 — Parameter data for the COPY STATUS service action

Bit / Byte	7	6	5	4	3	2	1	0
0	(MSB)			AVAILABLE DATA (00000008h)				
3								(LSB)
4	HDD			STATUS				
5	(MSB)			SEGMENTS PROCESSED				
6								(LSB)
7				TRANSFER COUNT UNITS				
8	(MSB)			TRANSFER COUNT				
11								(LSB)

After completion of an EXTENDED COPY command, the copy manager shall preserve all data returned by a COPY STATUS service action for a vendor specific period of time. The copy manager shall discard the COPY STATUS data when:

a) a RECIEVE COPY RESULTS command with COPY STATUS service action is received from the same initiator with a matching list identifier;
b) when another EXTENDED COPY command is received from the same initiator and the list identifier matches the list identifier associated with the data preserved for the COPY STATUS service action;
c) when the copy manager detects a hard reset condition; or
d) when the copy manager requires the resources used to preserve the data.

The AVAILABLE DATA field shall contain the number of bytes present in the parameter data that follows.

The held data discarded (HDD) bit indicates whether held data has been discarded. If HDD is one, held data has been discarded as described in 7.14.4. If HDD is zero, held data has not been discarded.

The STATUS field contains the current status of the EXTENDED COPY command identified by the LIST IDENTIFIER field in the CDB as defined in table 88.

Table 88 — COPY STATUS STATUS values

STATUS value	Meaning
00h	Operation in progress
01h	Operation completed without errors
02h	Operation completed with errors
04h - 7Fh	Reserved

The SEGMENTS PROCESSED field contains the number of segments the copy manager has processed for the EXTENDED COPY command identified by the LIST IDENTIFIER field in the CDB including the segment currently being processed. This field shall be zero if the copy manager has not yet begun processing segment descriptors.

The TRANSFER COUNT UNITS field specifies the units for the TRANSFER COUNT field as defined in table 89.

Table 89 — COPY STATUS TRANSFER COUNT UNITS values

Value	Meaning	Binary Multiplier Name [a]	Multiplier to convert TRANSFER COUNT field to bytes
00h	Bytes		1
01h	Ki-bytes	Kilobinary	2^{10} or 1024
02h	Mi-bytes	Megabinary	2^{20}
03h	Gi-bytes	Gigabinary	2^{30}
04h	Ti-bytes	Terabinary	2^{40}
05h	Pi-bytes	Petabinary	2^{50}
06h	Ei-bytes	Exabinary	2^{60}
07h - FFh	Reserved		

[a] This nomenclature is defined in IEC 60027-2 (2000).

The TRANSFER COUNT field specifies the amount of data written to a destination device for the EXTENDED COPY command identified by the LIST IDENTIFIER field in the CDB prior to receiving the RECEIVE COPY RESULTS command with COPY STATUS service action.

7.14.3 RECEIVE DATA service action

If the copy manager supports those segment descriptors require data to be held for transfer to the application client, then the RECEIVE DATA service action causes the copy manager to return the held data using the format shown in table 90. If a copy manager supports any of the segment descriptor type codes that require data to be held for the application client (see 7.2.5), then it shall also support the RECEIVE COPY RESULTS command with RECEIVE DATA service action.

If the LIST IDENTIFIER field of a RECEIVE COPY RESULTS CDB identifies an EXTENDED COPY command that still is being processed by the copy manager, the command shall be terminated with a CHECK CONDITION status. The sense key shall be set to ILLEGAL REQUEST and the additional sense code shall be set to INVALID FIELD IN CDB.

Table 90 — Parameter data for the RECEIVE DATA service action

Bit / Byte	7	6	5	4	3	2	1	0
0	(MSB)							
3				AVAILABLE DATA (n-4)				(LSB)
4								
n				HELD DATA				

Following completion of an EXTENDED COPY command, the copy manager shall preserve all data returned by a RECIEVE DATA service action for a vendor specific period of time. The application client should issue a RECEIVE COPY RESULTS command with RECEIVE DATA service action as soon as practical following completion of the EXTENDED COPY command to insure that the data is not discarded by the copy manager. The copy manager shall discard the buffered inline data:

a) after all data held for a specific EXTENDED COPY command has been successfully transferred to the application client;
b) when a RECIEVE COPY RESULTS command with RECEIVE DATA service action has been received from the same initiator with a matching list identifier, with the ALLOCATION LENGTH field set to zero;
c) when another EXTENDED COPY command is received from the same initiator and the list identifier matches the list identifier associated with the data preserved for RECEIVE DATA service action;
d) when the copy manager detects a hard reset condition; or
e) when the copy manager requires the resources used to preserve the data.

The AVAILABLE DATA field shall contain the number of bytes of held data available for delivery to the application client. If the amount of held data sent to the application client is reduced due to insufficient allocation length, the AVAILABLE DATA field shall not be altered and the held data shall not be discarded.

The HELD DATA field contains the data held by the copy manager for delivery to the application client as prescribed by several segment descriptor type codes. Unless the copy manager's held data limit (see 7.14.4) is exceeded, the first byte held in response to the first segment descriptor in the EXTENDED COPY parameter list prescribing the holding of data (called the oldest byte held) is returned in byte 4. The last byte held in response to the last segment descriptor in the EXTENDED COPY parameter list prescribing the holding of data (called the newest byte held) is returned in byte n.

7.14.4 OPERATING PARAMETERS service action

In response to the OPERATING PARAMETERS service action, the copy manager shall return its operating parameter information in the format shown in table 91. If a device server supports the EXTENDED COPY command (see 7.2), then it shall also support the RECEIVE COPY RESULTS command with OPERATING PARAMETERS service action.

Table 91 — Parameter data for the OPERATING PARAMETERS service action

Bit / Byte	7	6	5	4	3	2	1	0
0	(MSB)							
3				AVAILABLE DATA (n-4)				(LSB)
4								
7				Reserved				
8	(MSB)							
9				MAXIMUM TARGET DESCRIPTOR COUNT				(LSB)
10	(MSB)							
11				MAXIMUM SEGMENT DESCRIPTOR COUNT				(LSB)
12	(MSB)							
15				MAXIMUM DESCRIPTOR LIST LENGTH				(LSB)
16	(MSB)							
19				MAXIMUM SEGMENT LENGTH				(LSB)
20	(MSB)							
23				MAXIMUM INLINE DATA LENGTH				(LSB)
24	(MSB)							
27				HELD DATA LIMIT				(LSB)
28	(MSB)							
31				MAXIMUM STREAM DEVICE TRANSFER SIZE				(LSB)
32								
35				Reserved				
36				MAXIMUM CONCURRENT COPIES				
37				DATA SEGMENT GRANULARITY (log 2)				
38				INLINE DATA GRANULARITY (log 2)				
39				HELD DATA GRANULARITY (log 2)				
40								
42				Reserved				
43				IMPLEMENTED DESCRIPTOR LIST LENGTH (n-43)				
44								
n				List of implemented descriptor type codes (ordered)				

The AVAILABLE DATA field shall contain the number of bytes following the AVAILABLE DATA field in the parameter data (i.e., the total number of parameter data bytes minus 4).

The MAXIMUM TARGET COUNT field contains the maximum number of target descriptors that the copy manager allows in a single EXTENDED COPY target descriptor list.

The MAXIMUM SEGMENT COUNT field contains the maximum number of segment descriptors that the copy manager allows in a single EXTENDED COPY segment descriptor list.

The MAXIMUM DESCRIPTOR LIST LENGTH field contains the maximum length, in bytes, of the target descriptor list and segment descriptor list. This length includes the embedded data but excludes inline data that follows the descriptors.

The MAXIMUM SEGMENT LENGTH field indicates the length, in bytes, of the largest amount of data that the copy manager supports writing via a single segment. Bytes introduced as a result of the PAD bit being set to one (see 7.2.7) are not counted towards this limit. A value of zero indicates that the copy manager places no limits on the amount of data written by a single segment.

The MAXIMUM INLINE DATA LENGTH field indicates the length, in bytes, of the largest amount of inline data that the copy manager supports in the EXTENDED COPY parameter list. This does not include data included as embedded data within the segment descriptors. The MAXIMUM INLINE DATA LENGTH field applies only to segment descriptors containing the 04h descriptor type code (see 7.2.7.7). The field shall be set to zero when the 04h descriptor type code is not supported by the copy manager.

The HELD DATA LIMIT field indicates the length, in bytes, of the minimum amount of data the copy manager guarantees to hold for return to the application client via the RECEIVE COPY RESULTS command with RECEIVE DATA service action (see 7.14.3). If the processing of segment descriptors requires more data to be held, the copy manager may discard some of the held data in a vendor specific manner that retains the held bytes from the most recently processed segment descriptors. The discarding of held data bytes shall not be considered an error. If held data is discarded, the hdd bit shall be set as described in 7.14.2.

The MAXIMUM CONCURRENT COPIES field contains the maximum number of EXTENDED COPY commands supported for concurrent processing by the copy manager.

The DATA SEGMENT GRANULARITY field indicates the length of the smallest data block that copy manager permits in a non-inline segment descriptor (i.e., segment descriptors with type codes other than 04h). The amount of data transferred by a single segment descriptor shall be a multiple of the granularity. The DATA SEGMENT GRANULARITY value is expressed as a power of two. Bytes introduced as a result of the PAD bit being set to one (see 7.2.7) are not counted towards the data length granularity.

The INLINE DATA GRANULARITY field indicates the length of the of the smallest block of inline data that the copy manager permits being written by a segment descriptor containing the 04h descriptor type code (see 7.2.7.7). The amount of inline data written by a single segment descriptor shall be a multiple of the granularity. The INLINE DATA GRANULARITY value is expressed as a power of two. Bytes introduced as a result of the PAD bit being set to one (see 7.2.7) are not counted towards the length granularity.

If the copy manager encounters a data or inline segment descriptor that violates either the data segment granularity or the inline data granularity, the EXTENDED COPY command shall be terminated with a CHECK CONDITION status. The sense key shall be set to COPY ABORTED and the additional sense code shall be set to COPY SEGMENT GRANULARITY VIOLATION.

The HELD DATA GRANULARITY field indicates the length of the smallest block of held data that the copy manager shall transfer to the application client in response to a RECEIVE COPY RESULTS command with RECEIVE DATA service action (see 7.14.3). The amount of data held by the copy manager in response to any one segment descriptor shall be a multiple of this granularity. The HELD DATA GRANULARITY value is expressed as a power of two.

The MAXIMUM STREAM DEVICE TRANSFER SIZE field indicates the maximum transfer size, in bytes, supported for stream devices.

The IMPLEMENTED DESCRIPTOR LIST LENGTH field contains the length, in bytes, of the list of implemented descriptor type codes.

The list of implemented descriptor type codes contains one byte for each segment or target DESCRIPTOR TYPE CODE value (see 7.2.5) supported by the copy manager, with a unique supported DESCRIPTOR TYPE CODE value in each byte. The DESCRIPTOR TYPE CODE values shall appear in the list in ascending numerical order.

7.14.5 FAILED SEGMENT DETAILS service action

In response to the FAILED SEGMENT DETAILS service action, the copy manager shall return details of the segment processing failure that caused termination of the EXTENDED COPY command (see 7.2) identified by the LIST IDENTIFIER field in the CDB. Table 92 shows the format of the information returned by the copy manager in response to a FAILED SEGMENT DETAILS service action. If a device server supports the EXTENDED COPY command (see 7.4), then it shall also support the RECEIVE COPY RESULTS command with FAILED SEGMENT DETAILS service action.

When processing of an EXTENDED COPY command is aborted and processing of a segment descriptor is incomplete, the copy manager shall preserve details about the progress in processing of that descriptor. These details enable the application client to obtain information it needs to determine the state in which target devices (in particular stream devices) have been left by incomplete processing.

If the LIST IDENTIFIER field of a RECEIVE COPY RESULTS CDB identifies an EXTENDED COPY command that still is being processed by the copy manager, the command shall be terminated with a CHECK CONDITION status. The sense key shall be set to ILLEGAL REQUEST and the additional sense code shall be set to INVALID FIELD IN CDB.

Table 92 — Parameter data for the FAILED SEGMENT DETAILS service action

Bit / Byte	7	6	5	4	3	2	1	0
0	(MSB)							
3				AVAILABLE DATA (n-4)				(LSB)
4								
55				Reserved				
56				EXTENDED COPY COMMAND STATUS				
57				Reserved				
58	(MSB)							
59				SENSE DATA LENGTH (n-59)				(LSB)
60								
n				SENSE DATA				

The application client should issue a RECEIVE COPY RESULTS command with FAILED SEGMENT DETAILS service action immediately following failure of the EXTENDED COPY command to insure that the information is not discarded by the copy manager. The copy manager shall discard the failed segment details:

a) after all failed segment details held for a specific EXTENDED COPY command have been successfully transferred to the application client;
b) when a RECEIVE COPY RESULTS command with FAILED SEGMENT DETAILS service action has been received from the same initiator with a matching list identifier, with the ALLOCATION LENGTH field set to zero;
c) when another EXTENDED COPY command is received from the same initiator using the same list identifier;
d) when the copy manager detects a hard reset condition; or
e) when the copy manager requires the resources used to preserve the data.

The AVAILABLE DATA field shall contain the number of bytes of failed segment details available for delivery to the application client. If the amount of failed segment details data sent to the application client is reduced due to insufficient allocation length, the AVAILABLE DATA field shall not be altered and the failed segment details shall not be discarded. If no failed segment details data is available for the specified list identifier then the AVAILABLE DATA field shall be set to zero and no data beyond the AVAILABLE DATA field shall be returned.

The COPY COMMAND STATUS field contains the SCSI status value that was returned for the EXTENDED COPY command identified by the LIST IDENTIFIER field in the CDB.

The SENSE DATA LENGTH field indicates how many bytes of sense data are present in the SENSE DATA field.

The SENSE DATA field contains a copy of the sense data that the copy manager prepared as part of terminating the EXTENDED COPY command identified by the list identifier with a CHECK CONDITION status.

NOTE 23 Specific uses of the reserved bytes 4 to 55 are under discussion for SPC-3.

7.15 RECEIVE DIAGNOSTIC RESULTS command

The RECEIVE DIAGNOSTIC RESULTS command (see table 93) requests that data be sent to the application client after completion of a SEND DIAGNOSTIC command (see 7.23). If optional page formats are supported and the PCV bit is one, the PAGE CODE field specifies the format of the returned data, and there is no relationship to a previous SEND DIAGNOSTIC command.

Table 93 — RECEIVE DIAGNOSTIC RESULTS command

Bit / Byte	7	6	5	4	3	2	1	0
0	OPERATION CODE (1Ch)							
1	Reserved							PCV
2	PAGE CODE							
3	(MSB)	ALLOCATION LENGTH						
4								(LSB)
5	CONTROL							

A page code valid (PCV) bit of zero indicates that the most recent SEND DIAGNOSTIC command shall define the data returned by this command. Optionally, a PCV bit of one indicates that the contents of the PAGE CODE field shall define the data returned by this command. Page code values are defined in 8.1 or in another command set standard (see 3.1.12).

NOTES

24 To insure that the diagnostic command information is not destroyed by a command sent from another initiator the logical unit should be reserved.

25 Although diagnostic software is generally device-specific, this command and the SEND DIAGNOSTIC command provide a means to isolate the operating system software from the device-specific diagnostic software. The operating system may remain device-independent.

See 8.1 for RECEIVE DIAGNOSTIC RESULTS page format definitions.

7.16 RELEASE(10) command

7.16.1 RELEASE(10) command introduction

The RELEASE(10) command (see table 94) is used to release a previously reserved logical unit.

Table 94 — RELEASE(10) command

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

The RESERVE and RELEASE commands provide a basic mechanism for contention resolution in multiple-initiator systems. See 5.5.1 for a general description of reservations and the commands that manage them. A reservation may only be released by a RELEASE command from the initiator that made it. It is not an error for an application client to attempt to release a reservation that is not currently valid, or is held by another initiator. In this case, the device server shall return GOOD status without altering any other reservation.

Byte 1 Bit 0 and Byte 2 provide an obsolete way to release previously reserved extents within a logical unit. If Byte 1, Bit 0 is equal to one, device servers not implementing the obsolete capability shall terminate the command with CHECK CONDITION status and the sense key shall be set to ILLEGAL REQUEST.

7.16.2 Logical unit release

Logical unit reservation release is mandatory if the RELEASE(10) command is implemented. This command shall cause the device server to terminate all non-third-party logical unit reservations that are active from the initiator to the specified logical unit.

7.16.3 Third-party release

Third-party release allows an application client to release a logical unit that was previously reserved using third-party reservation (see 7.21.3). Third-party release is intended for use in multiple-initiator systems that use the EXTENDED COPY command.

If the third-party (3RDPTY) bit is zero, then a third-party release is not requested. If the 3RDPTY bit is zero then the LONGID bit and the PARAMETER LIST LENGTH field shall be ignored. If the 3RDPTY bit is one then the device server shall release the specified logical unit, but only if the initiator ID, 3RDPTY bit, and THIRD-PARTY DEVICE ID are identical when compared to the RESERVE command that established the reservation.

If the 3RDPTY bit is one the device server shall not modify the mode parameters for commands received from the third-party device even if the device server implements the transfer of mode parameters with a third-party RESERVE command.

NOTE 26 If a target implements independent storage of mode parameters for each initiator, a third-party RESERVE command copies the current mode parameters for the initiator that sent the RESERVE command to the current mode parameters for the initiator specified as the third-party device (e.g., a copy manager SCSI device). A unit attention condition notifies the third-party

of the changed mode parameters due to the reservation. A successful third-party RELEASE command does not change the third-party devices' current mode parameters back to their previous values. The third-party device may issue MODE SENSE and MODE SELECT commands to query and modify the mode parameters.

If the THIRD-PARTY DEVICE ID value associated with the reservation release is smaller than 255, the LONGID bit may be zero and the ID value sent in the CDB THIRD-PARTY DEVICE ID field. Device ID formats are protocol specific. If the LONGID bit is zero, the PARAMETER LIST LENGTH field shall be set to zero. If the THIRD-PARTY DEVICE ID is greater than 255, the LONGID bit shall be one.

Device servers that support device IDs greater than 255 shall accept commands with LONGID equal to one. Device servers whose devices IDs are limited to 255 or smaller may reject commands with LONGID equal to one with CHECK CONDITION status and a sense key of ILLEGAL REQUEST.

If the LONGID bit is one, the parameter list length shall be eight, and the parameter list shall have the format shown in table 95. If the LONGID bit is one, the THIRD-PARTY DEVICE ID field in the CDB shall be ignored. If the LONGID bit is one and the parameter list length is not eight, the device server shall return a CHECK CONDITION status with a sense key of ILLEGAL REQUEST.

Table 95 — RELEASE(10) parameter list

Bit / Byte	7	6	5	4	3	2	1	0
0	THIRD-PARTY DEVICE ID							
7								

7.17 RELEASE(6) command

The RELEASE(6) command (see table 96) is used to release a previously reserved logical unit. This subclause describes only those instances where the RELEASE(6) command differs from the RELEASE(10) command. Except for the instances described in this subclause, the RELEASE(6) command shall function exactly like the RELEASE(10) command (see 7.16).

Table 96 — RELEASE(6) command

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

The RELEASE(6) command shall not release third-party reservations.

Obsolete Bits 1 through 4 of Byte 1 provided a method, limited to device addresses 0 through 7, to handle third-party reservations in earlier versions of the SCSI standard. The obsolete method has been replaced by the RESERVE(10) and RELEASE(10).

Byte 1 Bit 0 and Byte 2 provide an obsolete way to release previously reserved extents within a logical unit. If Byte 1, Bit 0 is equal to one, device servers not implementing the obsolete capability shall terminate the command with CHECK CONDITION status and the sense key shall be set to ILLEGAL REQUEST.

7.18 REPORT DEVICE IDENTIFIER command

The REPORT DEVICE IDENTIFIER command (see table 97) 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. Additional MAINTENANCE IN and MAINTENANCE OUT service actions are defined in SCC-2 and in this standard.

The MAINTENANCE IN service actions defined only in SCC-2 shall apply only to SCSI devices that return a device type of 0Ch or the SCCS bit equal to one in their standard INQUIRY data. When a SCSI device returns a device type of 0Ch or the SCCS bit equal to one in its standard INQUIRY data, the implementation requirements for the SCC-2 MAINTENANCE IN service actions shall be as specified in SCC-2. Otherwise the MAINTENANCE IN service action definitions and implementation requirements stated in this standard shall apply.

Table 97 — REPORT DEVICE IDENTIFIER command

Bit / Byte	7	6	5	4	3	2	1	0
0	OPERATION CODE (A3h)							
1	Reserved			SERVICE ACTION (05h)				
2	Reserved							
3	Reserved							
4	Restricted							
5								
6	(MSB)							
7	ALLOCATION LENGTH							
8								
9								(LSB)
10	Reserved						Restricted	Reserved
11	CONTROL							

SCC-2 defines specific usages for bytes 4 and 5, and bit 1 in byte 10, however these fields are reserved for the REPORT DEVICE IDENTIFIER command defined by this standard.

The ALLOCATION LENGTH field indicates how many bytes have been allocated for the returned parameter data. If the length is not sufficient to contain all the parameter data, the first portion of the data shall be returned. This shall not be considered an error. The actual length of the parameter data is available in the IDENTIFIER LENGTH field in the parameter data. If the remainder of the parameter data is required, the application client should send a new REPORT DEVICE IDENTIFIER command with an ALLOCATION LENGTH field large enough to contain all the data.

The REPORT DEVICE IDENTIFIER parameter list (see table 98) contains a four-byte field that contains the length in bytes of the parameter list and the logical unit's identifier.

Table 98 — REPORT DEVICE IDENTIFIER parameter list

Bit / Byte	7	6	5	4	3	2	1	0
0	(MSB)							
1								
2				IDENTIFIER LENGTH (n-4)				
3								(LSB)
4								
n				IDENTIFIER				

The IDENTIFIER LENGTH field specifies the length in bytes of the IDENTIFIER field. If the ALLOCATION LENGTH field in the CDB is too small to transfer all of the identifier, the length shall not be adjusted to reflect the truncation. The identifier length shall initially equal zero, and shall be changed only by a successful SET DEVICE IDENTIFIER command.

The IDENTIFIER field shall contain a vendor specific value. The value reported shall be the last value written by a successful SET DEVICE IDENTIFIER command. The value of the identifier shall be changed only by a SET DEVICE IDENTIFIER command. The identifier value shall persist through resets, power cycles, media format operations, and media replacement.

The logical unit shall return the same Identifier to all initiators.

Processing a REPORT DEVICE IDENTIFIER may require the enabling of a nonvolatile memory within the logical unit. If the nonvolatile memory is not ready, the device server shall return CHECK CONDITION status, rather than wait for the device to become ready. The sense key shall be set to NOT READY and the additional sense code shall be set as described in table 117 (see 7.24). This information should allow the application client to determine the action required to cause the device server to become ready.

7.19 REPORT LUNS command

The REPORT LUNS command (see table 99) requests that the peripheral device logical unit inventory be sent to the application client. The logical unit inventory is a list that shall include the logical unit numbers of all logical units having a PERIPHERAL QUALIFIER value of 000b (see 7.3.2). Logical unit numbers for logical units with PERIPHERAL QUALIFIER values of 100b, 101b, 110b, or 111b may optionally be included in the logical unit inventory. A SCSI device that is capable of supporting a LUN address other than zero shall support a REPORT LUNS command that is addressed to logical unit zero. Support of the REPORT LUNS command by logical units other than logical unit zero is optional. Support of the REPORT LUNS command on devices having only a single logical unit with the logical unit number of zero is optional.

Table 99 — REPORT LUNS command

Bit / Byte	7	6	5	4	3	2	1	0
0	OPERATION CODE (A0h)							
1	Reserved							
2	Reserved							
3	Reserved							
4	Reserved							
5	Reserved							
6	(MSB)			ALLOCATION LENGTH				
7								
8								
9								(LSB)
10	Reserved							
11	CONTROL							

The allocation length should be at least 16 bytes. If the allocation length is not sufficient to contain the entire logical unit inventory, the device server shall report as many logical unit number values as fit in the specified allocation length. This shall not be considered an error.

NOTE 27 Devices compliant with SPC return CHECK CONDITION status with sense key ILLEGAL REQUEST and additional sense code set to INVALID FIELD IN CDB when the allocation length is less than 16 bytes.

The REPORT LUNS command shall return CHECK CONDITION status only when the device server is unable to return the requested report of the logical unit inventory.

If a REPORT LUNS command is received from an initiator with a pending unit attention condition (i.e., before the device server reports CHECK CONDITION status), the device server shall perform the REPORT LUNS command. If the unit attention condition was established because of a change in the logical unit inventory, that unit attention condition shall be cleared for that initiator by the REPORT LUNS command. Unit attention conditions established for other reasons shall not be cleared by the REPORT LUNS command (see SAM-2).

The REPORT LUNS data should be returned even though the device server is not ready for other commands. To minimize delays after a hard reset or power-up condition, the default report of the logical unit inventory should be available without incurring any media access delays. The default report of the logical unit inventory shall contain at least LUN 0.

If the logical unit inventory changes for any reason, including completion of initialization, removal of a logical unit, or creation of a logical unit, the device server shall generate a unit attention command for all initiators (see SAM-2). The device server shall set the additional sense code to REPORTED LUNS DATA HAS CHANGED.

The execution of a REPORT LUNS command to any valid and installed logical unit shall clear the REPORTED LUNS DATA HAS CHANGED unit attention condition for all logical units of that target with respect to the requesting initiator. A valid and installed logical unit is one having a PERIPHERAL QUALIFIER of 000b in the standard INQUIRY data (see 7.3.2).

The device server shall report those devices in the logical unit inventory using the format shown in table 100.

Table 100 — REPORT LUNS parameter list format

Bit / Byte	7	6	5	4	3	2	1	0
0	(MSB)							
3				LUN LIST LENGTH (N-7)				(LSB)
4	(MSB)							
7				Reserved				(LSB)
				LUN list				
8	(MSB)							
15				First LUN				(LSB)
				. . .				
n-7	(MSB)							
n				Last LUN				(LSB)

The LUN LIST LENGTH field shall contain the length in bytes of the LUN list that is available to be transferred. The LUN list length is the number of logical unit numbers in the logical unit inventory multiplied by eight. If the allocation length in the CDB is too small to transfer information about the entire logical unit inventory, the LUN list length value shall not be adjusted to reflect the truncation.

7.20 REQUEST SENSE command

7.20.1 REQUEST SENSE command introduction

The REQUEST SENSE command (see table 101) requests that the device server transfer sense data to the application client.

Table 101 — REQUEST SENSE command

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

Sense data shall be available and cleared under the conditions defined in SAM-2. If the device server has no other sense data available to return, it shall return a sense key of NO SENSE and an additional sense code of NO ADDITIONAL SENSE INFORMATION.

If the device server is in the standby power condition or idle power condition when a REQUEST SENSE command is received and there is no ACA or CA condition, the device server shall return a sense key of NO SENSE and an additional sense code of LOW POWER CONDITION ON. On completion of the command the logical unit shall return to the same power condition that was active before the REQUEST SENSE command was received. A REQUEST SENSE command shall not reset any active power condition timers.

The device server shall return CHECK CONDITION status for a REQUEST SENSE command only to report exception conditions specific to the command itself. For example:

a) An invalid field value is detected in the CDB;
b) An unrecovered parity error is detected by the service delivery subsystem; or
c) A target malfunction that prevents return of the sense data.

If a recovered error occurs during the processing of the REQUEST SENSE command, the device server shall return the sense data with GOOD status. If a device server returns CHECK CONDITION status for a REQUEST SENSE command, the sense data may be invalid.

NOTE 28 The sense data appropriate to the selection of an invalid logical unit is defined in SAM-2.

Device servers shall be capable of returning eighteen bytes of data in response to a REQUEST SENSE command. If the allocation length is eighteen or greater, and a device server returns less than eighteen bytes of data, the application client should assume that the bytes not transferred would have been zeros had the device server returned those bytes. Application clients may determine how much sense data has been returned by examining the ALLOCATION LENGTH field in the CDB and the ADDITIONAL SENSE LENGTH field in the sense data. Device servers shall not adjust the additional sense length to reflect truncation if the allocation length is less than the sense data available.

7.20.2 Sense data format

The sense data format for response codes 70h (current errors) and 71h (deferred errors) are defined in table 102.

Table 102 — Response codes 70h and 71h sense data format

Bit / Byte	7	6	5	4	3	2	1	0
0	VALID	RESPONSE CODE (70h or 71h)						
1	Obsolete							
2	FILEMARK	EOM	ILI	Reserved	SENSE KEY			
3	(MSB)	INFORMATION						
6								(LSB)
7	ADDITIONAL SENSE LENGTH (n-7)							
8	(MSB)	COMMAND-SPECIFIC INFORMATION						
11								(LSB)
12	ADDITIONAL SENSE CODE							
13	ADDITIONAL SENSE CODE QUALIFIER							
14	FIELD REPLACEABLE UNIT CODE							
15	SKSV	SENSE-KEY SPECIFIC						
17								
18	Additional sense bytes							
n								

A VALID bit of zero indicates that the INFORMATION field is not as defined in this standard. A VALID bit of one indicates the INFORMATION field contains valid information as defined in this standard. Device servers shall implement the VALID bit.

Response code value 70h (current errors) is described in 7.20.4. Device servers shall implement response code 70h. Response code value 71h (deferred errors) is described in 7.20.5. Implementation of response code 71h is optional. Response code 7Fh is for a vendor specific sense data formats. Response code values of 72h to 7Eh and 00h to 6Fh are reserved.

The obsolete byte 1 contained information used by the COPY command.

The FILEMARK bit is mandatory for sequential-access devices, and this bit is reserved for all other device types. A FILEMARK bit of one indicates that the current command has read a filemark or setmark. The ADDITIONAL SENSE CODE field may be used to indicate whether a filemark or setmark was read. Reporting of setmarks is optional and indicated by the RSMK bit for sequential-access devices in the configuration parameters page. (See SSC.)

The end-of-medium (EOM) bit is mandatory for sequential-access and printer devices, and this bit is reserved for all other device types. An EOM bit of one indicates that an end-of-medium condition (e.g., end-of-partition, beginning-of-partition, or out-of-paper) exists. For sequential-access devices, this bit indicates that the unit is at or past the early-warning if the direction was forward, or that the command was not completed because beginning-of-partition was encountered if the direction was reverse. (See SSC.)

An incorrect length indicator (ILI) bit of one usually indicates that the requested logical block length did not match the logical block length of the data on the medium. Examples of other conditions indicated by the ILI bit being set to one include media interchange incompatibilities where the recorded logical block length is too large for the device server to read.

The SENSE KEY, ADDITIONAL SENSE CODE and ADDITIONAL SENSE CODE QUALIFIER fields provide a hierarchy of information. The intention of the hierarchy is to provide a top-down approach for an application client to determine

information relating to the error and exception conditions. The sense key provides generic categories in which error and exception conditions may be reported. Application clients typically use sense keys for high level error recovery procedures. Additional sense codes provide further detail describing the sense key. Additional sense code qualifiers add further detail to the additional sense code. The additional sense code and additional sense code qualifier may be used by application clients where sophisticated error recovery procedures require detailed information describing the error and exception conditions.

The SENSE KEY field is mandatory and indicates generic information describing an error or exception condition. The sense keys are defined in 7.20.6.

The contents of the INFORMATION field is device-type or command specific and is defined within the appropriate standard for the device type or command of interest. Device servers shall implement the INFORMATION field. Unless specified otherwise, this field contains:

a) the unsigned logical block address associated with the sense key, for direct-access devices (device type 0), write-once devices (device type 4), CD-ROM devices (device type 5), and optical memory devices (device type 7). If the logical block address value cannot be represented in four bytes, the VALID bit shall be set to zero;
b) the difference (residue) of the requested length minus the actual length in either bytes or blocks, as determined by the command, for sequential-access devices (device type 1), printer devices (device type 2), processor devices (device type 3) and some direct access device commands, except as defined for d) below. Negative values are indicated by two's complement notation;
c) the difference (residue) of the requested number of blocks minus the actual number of blocks copied or compared for the current segment descriptor of an EXTENDED COPY command; or
d) for sequential-access devices operating in buffered modes 1h or 2h that detect an unrecoverable write error when unwritten data blocks, filemarks, or setmarks remain in the buffer, the value of the INFORMATION field for all commands shall be:
   A) the total number of data blocks, filemarks, and setmarks in the buffer if the device is in fixed block mode (i.e., BLOCK LENGTH field of the MODE SENSE block descriptor is non-zero and the FIXED bit of the WRITE command is one); or
   B) the number of bytes in the buffer, including filemarks and setmarks, if the device is in variable mode (i.e., the FIXED bit of the WRITE command is zero).

For additional information on the use of the INFORMATION field by sequential-access devices see SSC.

The ADDITIONAL SENSE LENGTH field indicates the number of additional sense bytes to follow. If the allocation length of the CDB is too small to transfer all of the additional sense bytes, the additional sense length is not adjusted to reflect the truncation.

The COMMAND-SPECIFIC INFORMATION field contains information that depends on the command that encountered the exception condition. Further meaning for this field is defined within the command description. The COMMAND-SPECIFIC INFORMATION field is mandatory if the device server supports any of the following commands: EXTENDED COPY and REASSIGN BLOCKS.

The additional sense code (ASC) indicates further information related to the error or exception condition reported in the SENSE KEY field. Device servers shall support the ADDITIONAL SENSE CODE field. Support of the additional sense codes not explicitly required by this standard is optional. A list of additional sense codes is in 7.20.6. If the device server does not have further information related to the error or exception condition, the additional sense code is set to NO ADDITIONAL SENSE INFORMATION.

The additional sense code qualifier (ASCQ) indicates detailed information related to the additional sense code. The additional sense code qualifier is optional. If the error or exception condition is reportable by the device, the value returned shall be as specified in 7.20.6. If the device server does not have detailed information related to the error or exception condition, the additional sense code qualifier is set to zero.

Non-zero values in the FIELD REPLACEABLE UNIT CODE field are used to define a device-specific mechanism or unit that has failed. A value of zero in this field shall indicate that no specific mechanism or unit has been identified to have failed or that the data is not available. The FIELD REPLACEABLE UNIT CODE field is optional. The format of this information is not specified by this standard. Additional information about the field replaceable unit may be available in the ASCII information page (see 8.4.3), if supported by the device server.

The SENSE-KEY SPECIFIC bytes are described in 7.20.3, below.

The additional sense bytes may contain command specific data, peripheral device specific data, or vendor specific data that further defines the nature of the CHECK CONDITION status.

7.20.3 Sense-key specific

A sense-key specific valid (SKSV) bit of one indicates the SENSE-KEY SPECIFIC field contains valid information as defined in this standard. The SKSV bit and SENSE-KEY SPECIFIC field are optional. The definition of this field is determined by the value of the SENSE KEY field. This field is reserved for sense keys not described below. An SKSV value of zero indicates that this field is not as defined by this standard.

If the sense key is ILLEGAL REQUEST and the SKSV bit is set to one, then the SENSE-KEY SPECIFIC field shall be as defined as shown in table 103. The FIELD POINTER field indicates which parameters in the CDB or the data parameters are in error.

Table 103 — Field pointer bytes

Bit / Byte	7	6	5	4	3	2	1	0
15	SKSV	C/D	Reserved		BPV	BIT POINTER		
16	(MSB)	FIELD POINTER						
17								(LSB)

A command data (C/D) bit of one indicates that the illegal parameter is in the CDB. A C/D bit of zero indicates that the illegal parameter is in the data parameters sent by the application client in the Data-Out Buffer.

A bit pointer valid (BPV) bit of zero indicates that the value in the BIT POINTER field is not valid. A BPV bit of one indicates that the BIT POINTER field specifies which bit of the byte designated by the FIELD POINTER field is in error. When a multiple-bit field is in error, the BIT POINTER field shall point to the most-significant (left-most) bit of the field.

The FIELD POINTER field indicates which byte of the CDB or of the parameter data was in error. Bytes are numbered starting from zero, as shown in the tables describing the commands and parameters. When a multiple-byte field is in error, the field pointer shall point to the most-significant (i.e., left-most) byte of the field. If several consecutive bytes are reserved, each shall be treated as a single-byte field.

NOTE 29 Bytes identified as being in error are not necessarily the place that has to be changed to correct the problem.

If the sense key is RECOVERED ERROR, HARDWARE ERROR or MEDIUM ERROR and if the SKSV bit is one, the SENSE-KEY SPECIFIC field shall be as shown in table 104.

Table 104 — Actual retry count bytes

Bit / Byte	7	6	5	4	3	2	1	0
15	SKSV	Reserved						
16	(MSB)	ACTUAL RETRY COUNT						
17								(LSB)

The ACTUAL RETRY COUNT field returns vendor specific information on the actual number of retries of the recovery algorithm used in attempting to recover an error or exception condition.

NOTE 30 This field should be computed in the same way as the retry count fields within the error recovery page of the MODE SELECT command.

If the sense key is NOT READY or NO SENSE and the SKSV bit is one, the SENSE-KEY SPECIFIC field shall be as shown in table 105.

Table 105 — Progress indication bytes

Bit / Byte	7	6	5	4	3	2	1	0
15	SKSV	Reserved						
16	(MSB)	PROGRESS INDICATION						
17								(LSB)

The PROGRESS INDICATION field is a percent complete indication in which the returned value is the numerator that has 65 536 (10000h) as its denominator. The progress indication shall be based upon the total operation.

NOTE 31 It is intended that the progress indication be time related. However, since for example format time varies with the number of defects encountered, etc., it is reasonable for the device server to assign values to various steps within the process. The granularity of these steps should be small enough to provide reasonable assurances to the application client that progress is being made.

If the sense key is COPY ABORTED and the SKSV bit is one, the SENSE-KEY SPECIFIC field shall be as shown in table 106.

Table 106 — Segment pointer bytes

Bit / Byte	7	6	5	4	3	2	1	0
15	SKSV	Reserved	SD	Reserved	BPV	BIT POINTER		
16	(MSB)	FIELD POINTER						
17								(LSB)

The segment descriptor (SD) bit indicates whether the field pointer is with reference to the start of the parameter list or to the start of a segment descriptor. An SD value of zero indicates that the field pointer is relative to the start of the parameter list. An SD value of one indicates that the field pointer is relative to the start of the segment descriptor indicated by the third and fourth bytes of the COMMAND SPECIFIC INFORMATION field (see 7.2.3).

A bit pointer valid (BPV) bit of zero indicates that the value in the BIT POINTER field is not valid. A BPV bit of one indicates that the BIT POINTER field specifies which bit of the byte designated by the FIELD POINTER field is in error. When a multiple-bit field is in error, the BIT POINTER field shall point to the most-significant (i.e., left-most) bit of the field.

The FIELD POINTER field indicates which byte of the parameter list or segment descriptor was in error.

NOTE 32 If the parameter list is in excess of 65 528 bytes in length and SD is 0, the FIELD POINTER value may not fit in two bytes provided by the sense key specific format definition.

7.20.4 Current errors

Response code 70h (current error) indicates that the CHECK CONDITION status returned is the result of an error or exception condition on the task that returned the CHECK CONDITION status or a protocol specific failure condition. This includes errors generated during processing of the command. It also includes errors not related to any command that are first observed during processing of a command (e.g., disk servo-mechanism failure, off-track errors, or power-up test errors).

7.20.5 Deferred errors

Response code 71h (deferred error) indicates that the CHECK CONDITION status returned is the result of an error or exception condition that occurred during processing of a previous command for which GOOD status has already

been returned. Such commands are associated with use of the immediate bit and with some forms of caching. Device servers that implement these features shall implement deferred error reporting.

The deferred error indication may be sent at a time selected by the device server through use of the asynchronous event reporting mechanism (see SAM-2), if AER is supported by both the application client and device server.

If AER is not supported, the deferred error may be indicated by returning CHECK CONDITION status to an application client on the appropriate initiator as described later in this subclause. A subsequent REQUEST SENSE command shall return the deferred error sense information.

If the task terminates with CHECK CONDITION status and the sense data describes a deferred error the command for the terminated task shall not have been processed. After the device server detects a deferred error condition, it shall return a deferred error according to the following rules:

a) If no external intervention is necessary to recover a deferred error, a deferred error indication shall not be posted unless required by the error handling parameters of a MODE SELECT command. The occurrence of the error may be logged if statistical or error logging is supported;
b) If it is possible to associate a deferred error with an initiator and with a particular function or a particular subset of data, and the error is either unrecovered or required to be reported by the mode parameters, a deferred error indication shall be returned to an application client on the initiator associated with the error. If an application client on an initiator other than the initiator associated with the error attempts access to the particular function or subset of data associated with the deferred error and the TST field equals 000b (see 8.3.6), the command attempting the access shall be responded to according to the requirements in SAM-2. If an application client on an initiator other than the initiator associated with the error attempts access to the particular function or subset of data associated with the deferred error and the TST field equals 001b, the command attempting the access shall not be blocked by the deferred error and the cause of the deferred error may result in an error being reported for the command attempting the access;
c) If the device server is unable to associate a deferred error with an initiator or with a particular subset of data, the device server shall return a deferred error indication to an application client on each initiator. If multiple deferred errors have accumulated for an initiator, only the last error shall be returned;
d) If the device server is unable to associate a deferred error with a particular logical unit, the device server shall return a deferred error indication to an application client associated with any logical unit on the appropriate initiator; or
e) If a task has never entered the enabled task state, and a deferred error occurs, the task shall be terminated with CHECK CONDITION status and deferred error information posted in the sense data. If a deferred error occurs after a task has entered the enabled task state and the task is affected by the error, the task shall be terminated by CHECK CONDITION status and the current error information shall be returned in the sense data. In this case, if the current error information does not adequately define the deferred error condition, a deferred error may be returned after the current error information has been recovered. If a deferred error occurs after a task has entered the enabled task state and the task completes successfully, the device server may choose to return the deferred error information after the completion of the current command in conjunction with a subsequent command that has not begun processing.

NOTE 33 A deferred error may indicate that an operation was unsuccessful long after GOOD status was returned. If the application client is unable to replicate or recover from other sources the data that is being written using buffered write operations, synchronization commands should be performed before the critical data is destroyed in the host. This is necessary to be sure that recovery actions may be taken if deferred errors do occur in the storing of the data. If AER is not implemented, the synchronizing process should provide the necessary commands to allow returning CHECK CONDITION status and subsequent returning of deferred error sense information after all buffered operations are guaranteed to be complete.

7.20.6 Sense key and sense code definitions

The sense keys are defined in table 107.

Table 107 — Sense key descriptions

Sense key	Description
0h	**NO SENSE.** Indicates that there is no specific sense key information to be reported. This may occur for a successful command or for a command that receives CHECK CONDITION status because one of the FILEMARK, EOM, or ILI bits is set to one.
1h	**RECOVERED ERROR.** Indicates that the last command completed successfully, with some recovery action performed by the device server. Details may be determinable by examining the additional sense bytes and the INFORMATION field. When multiple recovered errors occur during one command, the choice of which error to report (first, last, most severe, etc.) is vendor specific.
2h	**NOT READY.** Indicates that the logical unit addressed cannot be accessed. Operator intervention may be required to correct this condition.
3h	**MEDIUM ERROR.** Indicates that the command terminated with a non-recovered error condition that was probably caused by a flaw in the medium or an error in the recorded data. This sense key may also be returned if the device server is unable to distinguish between a flaw in the medium and a specific hardware failure (i.e., sense key 4h).
4h	**HARDWARE ERROR.** Indicates that the device server detected a non-recoverable hardware failure (e.g., controller failure, device failure, or parity error) while performing the command or during a self test.
5h	**ILLEGAL REQUEST.** Indicates that there was an illegal parameter in the CDB or in the additional parameters supplied as data for some commands (e.g., FORMAT UNIT or SEARCH DATA). If the device server detects an invalid parameter in the CDB, then it shall terminate the command without altering the medium. If the device server detects an invalid parameter in the additional parameters supplied as data, then the device server may have already altered the medium.
6h	**UNIT ATTENTION.** Indicates that the removable medium may have been changed or the target has been reset. See SAM-2 for more detailed information about the unit attention condition.
7h	**DATA PROTECT.** Indicates that a command that reads or writes the medium was attempted on a block that is protected from this operation. The read or write operation is not performed.
8h	**BLANK CHECK.** Indicates that a write-once device or a sequential-access device encountered blank medium or format-defined end-of-data indication while reading or a write-once device encountered a non-blank medium while writing.
9h	**VENDOR SPECIFIC.** This sense key is available for reporting vendor specific conditions.
Ah	**COPY ABORTED.** Indicates an EXTENDED COPY command was aborted due to an error condition on the source device, the destination device, or both (see 7.2.3).
Bh	**ABORTED COMMAND.** Indicates that the device server aborted the command. The application client may be able to recover by trying the command again.
Ch	Obsolete
Dh	**VOLUME OVERFLOW.** Indicates that a buffered SCSI device has reached the end-of-partition and data may remain in the buffer that has not been written to the medium. One or more RECOVER BUFFERED DATA command(s) may be issued to read the unwritten data from the buffer. (See SSC.)
Eh	**MISCOMPARE.** Indicates that the source data did not match the data read from the medium.
Fh	Reserved

The additional sense codes and additional sense code qualifiers are defined in table 108.

Table 108 — ASC and ASCQ assignments (part 1 of 11)

D - DIRECT ACCESS DEVICE (SBC)
. T - SEQUENTIAL ACCESS DEVICE (SSC)
. L - PRINTER DEVICE (SSC)
. P - PROCESSOR DEVICE (SPC-2)
. . W - WRITE ONCE READ MULTIPLE DEVICE (SBC)
. . R - C/DVD DEVICE (MMC-2)
. . S - SCANNER DEVICE (SCSI-2)
. . . O - OPTICAL MEMORY DEVICE (SBC)
. . . M - MEDIA CHANGER DEVICE (SMC)
. . . C - COMMUNICATION DEVICE (SCSI-2)
. . . . A - STORAGE ARRAY DEVICE (SCC)
. . . . E - ENCLOSURE SERVICES DEVICE (SES)
. . . . B - SIMPLIFIED DIRECT-ACCESS DEVICE (RBC)
. . . . . K - OPTICAL CARD READER/WRITER DEVICE (OCRW)

Device Column key
blank = code not used
not blank = code used

ASC	ASCQ	D	T	L	P	W	R	S	O	M	C	A	E	B	K	Description
67h	02h											A				ADD LOGICAL UNIT FAILED
13h	00h	D				W			O					B	K	ADDRESS MARK NOT FOUND FOR DATA FIELD
12h	00h	D				W			O					B	K	ADDRESS MARK NOT FOUND FOR ID FIELD
67h	08h											A				ASSIGN FAILURE OCCURRED
27h	03h		T				R									ASSOCIATED WRITE PROTECT
47h	04h	D	T	L	P	W	R	S	O	M	C	A	E	B	K	ASYNCHRONOUS INFORMATION PROTECTION ERROR DETECTED
67h	06h											A				ATTACHMENT OF LOGICAL UNIT FAILED
00h	11h						R									AUDIO PLAY OPERATION IN PROGRESS
00h	12h						R									AUDIO PLAY OPERATION PAUSED
00h	14h						R									AUDIO PLAY OPERATION STOPPED DUE TO ERROR
00h	13h						R									AUDIO PLAY OPERATION SUCCESSFULLY COMPLETED
66h	00h							S								AUTOMATIC DOCUMENT FEEDER COVER UP
66h	01h							S								AUTOMATIC DOCUMENT FEEDER LIFT UP
00h	04h		T					S								BEGINNING-OF-PARTITION/MEDIUM DETECTED
0Ch	06h	D	T			W			O					B		BLOCK NOT COMPRESSIBLE
14h	04h		T													BLOCK SEQUENCE ERROR
29h	03h	D	T	L	P	W	R	S	O	M	C	A	E	B	K	BUS DEVICE RESET FUNCTION OCCURRED
11h	0Eh	D	T			W	R		O					B		CANNOT DECOMPRESS USING DECLARED ALGORITHM
30h	06h	D	T			W	R		O					B		CANNOT FORMAT MEDIUM - INCOMPATIBLE MEDIUM
30h	02h	D	T			W	R		O					B	K	CANNOT READ MEDIUM - INCOMPATIBLE FORMAT
30h	01h	D	T			W	R		O					B	K	CANNOT READ MEDIUM - UNKNOWN FORMAT
30h	08h						R									CANNOT WRITE - APPLICATION CODE MISMATCH
30h	05h	D	T			W	R		O					B	K	CANNOT WRITE MEDIUM - INCOMPATIBLE FORMAT
30h	04h	D	T			W	R		O					B	K	CANNOT WRITE MEDIUM - UNKNOWN FORMAT
52h	00h		T													CARTRIDGE FAULT
73h	00h						R									CD CONTROL ERROR
24h	01h	D	T	L	P	W	R	S	O	M	C	A	E	B	K	CDB DECRYPTION ERROR
3Fh	02h	D	T	L	P	W	R	S	O	M	C			B	K	CHANGED OPERATING DEFINITION
11h	06h					W	R		O					B		CIRC UNRECOVERED ERROR
30h	03h	D	T				R								K	CLEANING CARTRIDGE INSTALLED
30h	07h	D	T	L		W	R	S	O	M		A	E	B	K	CLEANING FAILURE
00h	17h	D	T	L		W	R	S	O	M		A	E	B	K	CLEANING REQUESTED
4Ah	00h	D	T	L	P	W	R	S	O	M	C	A	E	B	K	COMMAND PHASE ERROR
2Ch	00h	D	T	L	P	W	R	S	O	M	C	A	E	B	K	COMMAND SEQUENCE ERROR
6Eh	00h											A				COMMAND TO LOGICAL UNIT FAILED
2Fh	00h	D	T	L	P	W	R	S	O	M	C	A	E	B	K	COMMANDS CLEARED BY ANOTHER INITIATOR
3Fh	04h	D	T			W	R		O	M	C	A	E	B	K	COMPONENT DEVICE ATTACHED
0Ch	04h	D	T			W			O					B		COMPRESSION CHECK MISCOMPARE ERROR
27h	06h						R									CONDITIONAL WRITE PROTECT
67h	00h											A				CONFIGURATION FAILURE
67h	01h											A				CONFIGURATION OF INCAPABLE LOGICAL UNITS FAILED
5Dh	25h	D												B		CONTROLLER IMPENDING FAILURE ACCESS TIMES TOO HIGH
5Dh	27h	D												B		CONTROLLER IMPENDING FAILURE CHANNEL PARAMETRICS
5Dh	28h	D												B		CONTROLLER IMPENDING FAILURE CONTROLLER DETECTED

Annex C contains the ASC and ASCQ assignments in numeric order.

Table 108 — ASC and ASCQ assignments (part 2 of 11)

D - DIRECT ACCESS DEVICE (SBC)
. T - SEQUENTIAL ACCESS DEVICE (SSC)
. L - PRINTER DEVICE (SSC)
. P - PROCESSOR DEVICE (SPC-2)
. . W - WRITE ONCE READ MULTIPLE DEVICE (SBC)
. . R - C/DVD DEVICE (MMC-2)
. . S - SCANNER DEVICE (SCSI-2)
. . . O - OPTICAL MEMORY DEVICE (SBC)
. . . M - MEDIA CHANGER DEVICE (SMC)
. . . C - COMMUNICATION DEVICE (SCSI-2)
. . . . A - STORAGE ARRAY DEVICE (SCC)
. . . . E - ENCLOSURE SERVICES DEVICE (SES)
. . . . B - SIMPLIFIED DIRECT-ACCESS DEVICE (RBC)
. . . . . K - OPTICAL CARD READER/WRITER DEVICE (OCRW)

Device Column key
blank = code not used
not blank = code used

ASC	ASCQ	D	T	L	P	W	R	S	O	M	C	A	E	B	K	Description
5Dh	22h	D												B		CONTROLLER IMPENDING FAILURE DATA ERROR RATE TOO HIGH
5Dh	2Ch	D												B		CONTROLLER IMPENDING FAILURE DRIVE CALIBRATION RETRY COUNT
5Dh	21h	D												B		CONTROLLER IMPENDING FAILURE DRIVE ERROR RATE TOO HIGH
5Dh	20h	D												B		CONTROLLER IMPENDING FAILURE GENERAL HARD DRIVE FAILURE
5Dh	23h	D												B		CONTROLLER IMPENDING FAILURE SEEK ERROR RATE TOO HIGH
5Dh	2Ah	D												B		CONTROLLER IMPENDING FAILURE SEEK TIME PERFORMANCE
5Dh	2Bh	D												B		CONTROLLER IMPENDING FAILURE SPIN-UP RETRY COUNT
5Dh	26h	D												B		CONTROLLER IMPENDING FAILURE START UNIT TIMES TOO HIGH
5Dh	29h	D												B		CONTROLLER IMPENDING FAILURE THROUGHPUT PERFORMANCE
5Dh	24h	D												B		CONTROLLER IMPENDING FAILURE TOO MANY BLOCK REASSIGNS
2Bh	00h	D	T	L	P	W	R	S	O		C				K	COPY CANNOT EXECUTE SINCE HOST CANNOT DISCONNECT
6Fh	00h						R									COPY PROTECTION KEY EXCHANGE FAILURE - AUTHENTICATION FAILURE
6Fh	02h						R									COPY PROTECTION KEY EXCHANGE FAILURE - KEY NOT ESTABLISHED
6Fh	01h						R									COPY PROTECTION KEY EXCHANGE FAILURE - KEY NOT PRESENT
26h	0Dh	D	T	L	P	W	R	S	O		C				K	COPY SEGMENT GRANULARITY VIOLATION
0Dh	05h	D	T	L	P	W	R	S	O		C	A			K	COPY TARGET DEVICE DATA OVERRUN
0Dh	04h	D	T	L	P	W	R	S	O		C	A			K	COPY TARGET DEVICE DATA UNDERRUN
0Dh	02h	D	T	L	P	W	R	S	O		C	A			K	COPY TARGET DEVICE NOT REACHABLE
67h	07h											A				CREATION OF LOGICAL UNIT FAILED
2Ch	04h						R									CURRENT PROGRAM AREA IS EMPTY
2Ch	03h						R									CURRENT PROGRAM AREA IS NOT EMPTY
30h	09h						R									CURRENT SESSION NOT FIXATED FOR APPEND
5Dh	35h	D												B		DATA CHANNEL IMPENDING FAILURE ACCESS TIMES TOO HIGH
5Dh	37h	D												B		DATA CHANNEL IMPENDING FAILURE CHANNEL PARAMETRICS
5Dh	38h	D												B		DATA CHANNEL IMPENDING FAILURE CONTROLLER DETECTED
5Dh	32h	D												B		DATA CHANNEL IMPENDING FAILURE DATA ERROR RATE TOO HIGH
5Dh	3Ch	D												B		DATA CHANNEL IMPENDING FAILURE DRIVE CALIBRATION RETRY COUNT
5Dh	31h	D												B		DATA CHANNEL IMPENDING FAILURE DRIVE ERROR RATE TOO HIGH
5Dh	30h	D												B		DATA CHANNEL IMPENDING FAILURE GENERAL HARD DRIVE FAILURE
5Dh	33h	D												B		DATA CHANNEL IMPENDING FAILURE SEEK ERROR RATE TOO HIGH
5Dh	3Ah	D												B		DATA CHANNEL IMPENDING FAILURE SEEK TIME PERFORMANCE
5Dh	3Bh	D												B		DATA CHANNEL IMPENDING FAILURE SPIN-UP RETRY COUNT
5Dh	36h	D												B		DATA CHANNEL IMPENDING FAILURE START UNIT TIMES TOO HIGH
5Dh	39h	D												B		DATA CHANNEL IMPENDING FAILURE THROUGHPUT PERFORMANCE
5Dh	34h	D												B		DATA CHANNEL IMPENDING FAILURE TOO MANY BLOCK REASSIGNS
26h	05h	D	T	L	P	W	R	S	O	M	C	A		B	K	DATA DECRYPTION ERROR
0Ch	05h	D	T			W			O					B		DATA EXPANSION OCCURRED DURING COMPRESSION
69h	00h											A				DATA LOSS ON LOGICAL UNIT
41h	00h	D														DATA PATH FAILURE (SHOULD USE 40 NN)
47h	01h	D	T	L	P	W	R	S	O	M	C	A	E	B	K	DATA PHASE CRC ERROR DETECTED
4Bh	00h	D	T	L	P	W	R	S	O	M	C	A	E	B	K	DATA PHASE ERROR

Annex C contains the ASC and ASCQ assignments in numeric order.

Table 108 — ASC and ASCQ assignments (part 3 of 11)

D - DIRECT ACCESS DEVICE (SBC)
. T - SEQUENTIAL ACCESS DEVICE (SSC)
. L - PRINTER DEVICE (SSC)
. P - PROCESSOR DEVICE (SPC-2)
. . W - WRITE ONCE READ MULTIPLE DEVICE (SBC)
. . R - C/DVD DEVICE (MMC-2)
. . S - SCANNER DEVICE (SCSI-2)
. . . O - OPTICAL MEMORY DEVICE (SBC)
. . . M - MEDIA CHANGER DEVICE (SMC)
. . . C - COMMUNICATION DEVICE (SCSI-2)
. . . . A - STORAGE ARRAY DEVICE (SCC)
. . . . E - ENCLOSURE SERVICES DEVICE (SES)
. . . . B - SIMPLIFIED DIRECT-ACCESS DEVICE (RBC)
. . . . . K - OPTICAL CARD READER/WRITER DEVICE (OCRW)

Device Column key
blank = code not used
not blank = code used

ASC	ASCQ	D	T	L	P	W	R	S	O	M	C	A	E	B	K	Description
11h	07h					W			O					B		DATA RE-SYNCHRONIZATION ERROR
16h	03h	D				W			O					B	K	DATA SYNC ERROR - DATA AUTO-REALLOCATED
16h	01h	D				W			O					B	K	DATA SYNC ERROR - DATA REWRITTEN
16h	04h	D				W			O					B	K	DATA SYNC ERROR - RECOMMEND REASSIGNMENT
16h	02h	D				W			O					B	K	DATA SYNC ERROR - RECOMMEND REWRITE
16h	00h	D				W			O					B	K	DATA SYNCHRONIZATION MARK ERROR
11h	0Dh	D	T			W	R		O					B		DE-COMPRESSION CRC ERROR
71h	00h		T													DECOMPRESSION EXCEPTION LONG ALGORITHM ID
70h	NNh		T													DECOMPRESSION EXCEPTION SHORT ALGORITHM ID OF NN
19h	00h	D							O						K	DEFECT LIST ERROR
19h	03h	D							O						K	DEFECT LIST ERROR IN GROWN LIST
19h	02h	D							O						K	DEFECT LIST ERROR IN PRIMARY LIST
19h	01h	D							O						K	DEFECT LIST NOT AVAILABLE
1Ch	00h	D							O					B	K	DEFECT LIST NOT FOUND
32h	01h	D				W			O					B	K	DEFECT LIST UPDATE FAILURE
3Fh	05h	D	T			W	R		O	M	C	A	E	B	K	DEVICE IDENTIFIER CHANGED
29h	04h	D	T	L	P	W	R	S	O	M	C	A	E	B	K	DEVICE INTERNAL RESET
40h	NNh	D	T	L	P	W	R	S	O	M	C	A	E	B	K	DIAGNOSTIC FAILURE ON COMPONENT NN (80H-FFH)
66h	02h							S								DOCUMENT JAM IN AUTOMATIC DOCUMENT FEEDER
66h	03h							S								DOCUMENT MISS FEED AUTOMATIC IN DOCUMENT FEEDER
6Fh	05h						R									DRIVE REGION MUST BE PERMANENT/REGION RESET COUNT ERROR
3Fh	0Fh	D	T	L	P	W	R	S	O	M	C	A	E	B	K	ECHO BUFFER OVERWRITTEN
72h	04h						R									EMPTY OR PARTIALLY WRITTEN RESERVED TRACK
34h	00h	D	T	L	P	W	R	S	O	M	C	A	E	B	K	ENCLOSURE FAILURE
35h	00h	D	T	L	P	W	R	S	O	M	C	A	E	B	K	ENCLOSURE SERVICES FAILURE
35h	03h	D	T	L	P	W	R	S	O	M	C	A	E	B	K	ENCLOSURE SERVICES TRANSFER FAILURE
35h	04h	D	T	L	P	W	R	S	O	M	C	A	E	B	K	ENCLOSURE SERVICES TRANSFER REFUSED
35h	02h	D	T	L	P	W	R	S	O	M	C	A	E	B	K	ENCLOSURE SERVICES UNAVAILABLE
3Bh	0Fh						R									END OF MEDIUM REACHED
63h	00h						R									END OF USER AREA ENCOUNTERED ON THIS TRACK
00h	05h		T	L				S								END-OF-DATA DETECTED
14h	03h		T													END-OF-DATA NOT FOUND
00h	02h		T					S								END-OF-PARTITION/MEDIUM DETECTED
51h	00h		T				R		O							ERASE FAILURE
51h	01h						R									ERASE FAILURE - INCOMPLETE ERASE OPERATION DETECTED
0Dh	00h	D	T	L	P	W	R	S	O		C	A			K	ERROR DETECTED BY THIRD PARTY TEMPORARY INITIATOR
0Ah	00h	D	T	L	P	W	R	S	O	M	C	A	E	B	K	ERROR LOG OVERFLOW
11h	10h						R									ERROR READING ISRC NUMBER
11h	0Fh						R									ERROR READING UPC/EAN NUMBER
11h	02h	D	T			W	R	S	O					B	K	ERROR TOO LONG TO CORRECT
38h	06h													B		ESN - DEVICE BUSY CLASS EVENT
38h	04h													B		ESN - MEDIA CLASS EVENT
38h	02h													B		ESN - POWER MANAGEMENT CLASS EVENT
38h	00h													B		EVENT STATUS NOTIFICATION
03h	02h		T													EXCESSIVE WRITE ERRORS

Annex C contains the ASC and ASCQ assignments in numeric order.

Table 108 — ASC and ASCQ assignments (part 4 of 11)

D - DIRECT ACCESS DEVICE (SBC)
. T - SEQUENTIAL ACCESS DEVICE (SSC)
. L - PRINTER DEVICE (SSC)
. P - PROCESSOR DEVICE (SPC-2)
. . W - WRITE ONCE READ MULTIPLE DEVICE (SBC)
. . R - C/DVD DEVICE (MMC-2)
. . S - SCANNER DEVICE (SCSI-2)
. . . O - OPTICAL MEMORY DEVICE (SBC)
. . . M - MEDIA CHANGER DEVICE (SMC)
. . . C - COMMUNICATION DEVICE (SCSI-2)
. . . . A - STORAGE ARRAY DEVICE (SCC)
. . . . E - ENCLOSURE SERVICES DEVICE (SES)
. . . . B - SIMPLIFIED DIRECT-ACCESS DEVICE (RBC)
. . . . . K - OPTICAL CARD READER/WRITER DEVICE (OCRW)

Device Column key
blank = code not used
not blank = code used

ASC	ASCQ	D	T	L	P	W	R	S	O	M	C	A	E	B	K	Description
67h	04h											A				EXCHANGE OF LOGICAL UNIT FAILED
3Bh	07h			L												FAILED TO SENSE BOTTOM-OF-FORM
3Bh	06h			L												FAILED TO SENSE TOP-OF-FORM
5Dh	00h	D	T	L	P	W	R	S	O	M	C	A	E	B	K	FAILURE PREDICTION THRESHOLD EXCEEDED
5Dh	FFh	D	T	L	P	W	R	S	O	M	C	A	E	B	K	FAILURE PREDICTION THRESHOLD EXCEEDED (FALSE)
00h	01h		T													FILEMARK DETECTED
14h	02h		T													FILEMARK OR SETMARK NOT FOUND
5Dh	65h	D												B		FIRMWARE IMPENDING FAILURE ACCESS TIMES TOO HIGH
5Dh	67h	D												B		FIRMWARE IMPENDING FAILURE CHANNEL PARAMETRICS
5Dh	68h	D												B		FIRMWARE IMPENDING FAILURE CONTROLLER DETECTED
5Dh	62h	D												B		FIRMWARE IMPENDING FAILURE DATA ERROR RATE TOO HIGH
5Dh	6Ch	D												B		FIRMWARE IMPENDING FAILURE DRIVE CALIBRATION RETRY COUNT
5Dh	61h	D												B		FIRMWARE IMPENDING FAILURE DRIVE ERROR RATE TOO HIGH
5Dh	60h	D												B		FIRMWARE IMPENDING FAILURE GENERAL HARD DRIVE FAILURE
5Dh	63h	D												B		FIRMWARE IMPENDING FAILURE SEEK ERROR RATE TOO HIGH
5Dh	6Ah	D												B		FIRMWARE IMPENDING FAILURE SEEK TIME PERFORMANCE
5Dh	6Bh	D												B		FIRMWARE IMPENDING FAILURE SPIN-UP RETRY COUNT
5Dh	66h	D												B		FIRMWARE IMPENDING FAILURE START UNIT TIMES TOO HIGH
5Dh	69h	D												B		FIRMWARE IMPENDING FAILURE THROUGHPUT PERFORMANCE
5Dh	64h	D												B		FIRMWARE IMPENDING FAILURE TOO MANY BLOCK REASSIGNS
09h	02h					W	R		O						K	FOCUS SERVO FAILURE
31h	01h	D		L			R		O					B		FORMAT COMMAND FAILED
58h	00h								O							GENERATION DOES NOT EXIST
1Ch	02h	D							O					B	K	GROWN DEFECT LIST NOT FOUND
5Dh	15h	D												B		HARDWARE IMPENDING FAILURE ACCESS TIMES TOO HIGH
5Dh	17h	D												B		HARDWARE IMPENDING FAILURE CHANNEL PARAMETRICS
5Dh	18h	D												B		HARDWARE IMPENDING FAILURE CONTROLLER DETECTED
5Dh	12h	D												B		HARDWARE IMPENDING FAILURE DATA ERROR RATE TOO HIGH
5Dh	1Ch	D												B		HARDWARE IMPENDING FAILURE DRIVE CALIBRATION RETRY COUNT
5Dh	11h	D												B		HARDWARE IMPENDING FAILURE DRIVE ERROR RATE TOO HIGH
5Dh	10h	D												B		HARDWARE IMPENDING FAILURE GENERAL HARD DRIVE FAILURE
5Dh	13h	D												B		HARDWARE IMPENDING FAILURE SEEK ERROR RATE TOO HIGH
5Dh	1Ah	D												B		HARDWARE IMPENDING FAILURE SEEK TIME PERFORMANCE
5Dh	1Bh	D												B		HARDWARE IMPENDING FAILURE SPIN-UP RETRY COUNT
5Dh	16h	D												B		HARDWARE IMPENDING FAILURE START UNIT TIMES TOO HIGH
5Dh	19h	D												B		HARDWARE IMPENDING FAILURE THROUGHPUT PERFORMANCE
5Dh	14h	D												B		HARDWARE IMPENDING FAILURE TOO MANY BLOCK REASSIGNS
27h	01h	D	T			W	R		O					B	K	HARDWARE WRITE PROTECTED
09h	04h	D	T			W	R		O					B		HEAD SELECT FAULT
00h	06h	D	T	L	P	W	R	S	O	M	C	A	E	B	K	I/O PROCESS TERMINATED
10h	00h	D				W			O					B	K	ID CRC OR ECC ERROR
5Eh	03h	D	T	L	P	W	R	S	O		C	A			K	IDLE CONDITION ACTIVATED BY COMMAND
5Eh	01h	D	T	L	P	W	R	S	O		C	A			K	IDLE CONDITION ACTIVATED BY TIMER
22h	00h	D														ILLEGAL FUNCTION (USE 20 00, 24 00, OR 26 00)

Annex C contains the ASC and ASCQ assignments in numeric order.

Table 108 — ASC and ASCQ assignments (part 5 of 11)

D - DIRECT ACCESS DEVICE (SBC)
. T - SEQUENTIAL ACCESS DEVICE (SSC)
. L - PRINTER DEVICE (SSC)
. P - PROCESSOR DEVICE (SPC-2)
. . W - WRITE ONCE READ MULTIPLE DEVICE (SBC)
. . R - C/DVD DEVICE (MMC-2)
. . S - SCANNER DEVICE (SCSI-2)
. . . O - OPTICAL MEMORY DEVICE (SBC)
. . . M - MEDIA CHANGER DEVICE (SMC)
. . . C - COMMUNICATION DEVICE (SCSI-2)
. . . . A - STORAGE ARRAY DEVICE (SCC)
. . . . E - ENCLOSURE SERVICES DEVICE (SES)
. . . . B - SIMPLIFIED DIRECT-ACCESS DEVICE (RBC)
. . . . . K - OPTICAL CARD READER/WRITER DEVICE (OCRW)

Device Column key
blank = code not used
not blank = code used

ASC	ASCQ	D	T	L	P	W	R	S	O	M	C	A	E	B	K	Description
64h	00h						R									ILLEGAL MODE FOR THIS TRACK
2Ch	05h													B		ILLEGAL POWER CONDITION REQUEST
28h	01h	D	T			W	R		O	M				B		IMPORT OR EXPORT ELEMENT ACCESSED
30h	00h	D	T			W	R		O	M				B	K	INCOMPATIBLE MEDIUM INSTALLED
11h	08h		T													INCOMPLETE BLOCK READ
0Dh	03h	D	T	L	P	W	R	S	O		C	A			K	INCORRECT COPY TARGET DEVICE TYPE
47h	03h	D	T	L	P	W	R	S	O	M	C	A	E	B	K	INFORMATION UNIT CRC ERROR DETECTED
6Ah	00h											A				INFORMATIONAL, REFER TO LOG
48h	00h	D	T	L	P	W	R	S	O	M	C	A	E	B	K	INITIATOR DETECTED ERROR MESSAGE RECEIVED
26h	0Bh	D	T	L	P	W	R	S	O		C				K	INLINE DATA LENGTH EXCEEDED
3Fh	03h	D	T	L	P	W	R	S	O	M	C	A	E	B	K	INQUIRY DATA HAS CHANGED
55h	04h	D	T	L	P	W	R	S	O	M		A	E			INSUFFICIENT REGISTRATION RESOURCES
55h	02h	D	T	L	P	W	R	S	O	M		A	E		K	INSUFFICIENT RESERVATION RESOURCES
55h	03h	D	T	L	P	W	R	S	O	M	C	A	E			INSUFFICIENT RESOURCES
2Eh	00h						R									INSUFFICIENT TIME FOR OPERATION
44h	00h	D	T	L	P	W	R	S	O	M	C	A	E	B	K	INTERNAL TARGET FAILURE
21h	02h						R									INVALID ADDRESS FOR WRITE
3Dh	00h	D	T	L	P	W	R	S	O	M	C	A	E		K	INVALID BITS IN IDENTIFY MESSAGE
24h	02h		T													INVALID CDB FIELD WHILE IN EXPLICIT BLOCK ADDRESS MODEL
24h	03h		T													INVALID CDB FIELD WHILE IN IMPLICIT BLOCK ADDRESS MODEL
2Ch	02h							S								INVALID COMBINATION OF WINDOWS SPECIFIED
20h	00h	D	T	L	P	W	R	S	O	M	C	A	E	B	K	INVALID COMMAND OPERATION CODE
21h	01h	D	T			W	R		O	M				B	K	INVALID ELEMENT ADDRESS
24h	00h	D	T	L	P	W	R	S	O	M	C	A	E	B	K	INVALID FIELD IN CDB
26h	00h	D	T	L	P	W	R	S	O	M	C	A	E	B	K	INVALID FIELD IN PARAMETER LIST
49h	00h	D	T	L	P	W	R	S	O	M	C	A	E	B	K	INVALID MESSAGE ERROR
26h	0Ch	D	T	L	P	W	R	S	O		C				K	INVALID OPERATION FOR COPY SOURCE OR DESTINATION
64h	01h						R									INVALID PACKET SIZE
26h	04h	D	T	L	P	W	R	S	O	M	C	A	E	B	K	INVALID RELEASE OF PERSISTENT RESERVATION
11h	05h					W	R		O					B		L-EC UNCORRECTABLE ERROR
60h	00h							S								LAMP FAILURE
5Bh	02h	D	T	L	P	W	R	S	O	M					K	LOG COUNTER AT MAXIMUM
5Bh	00h	D	T	L	P	W	R	S	O	M					K	LOG EXCEPTION
5Bh	03h	D	T	L	P	W	R	S	O	M					K	LOG LIST CODES EXHAUSTED
2Ah	02h	D	T	L		W	R	S	O	M	C	A	E		K	LOG PARAMETERS CHANGED
21h	00h	D	T			W	R		O	M				B	K	LOGICAL BLOCK ADDRESS OUT OF RANGE
08h	03h	D	T				R		O	M				B	K	LOGICAL UNIT COMMUNICATION CRC ERROR (ULTRA-DMA/32)
08h	00h	D	T	L		W	R	S	O	M	C	A	E	B	K	LOGICAL UNIT COMMUNICATION FAILURE
08h	02h	D	T	L		W	R	S	O	M	C	A	E	B	K	LOGICAL UNIT COMMUNICATION PARITY ERROR
08h	01h	D	T	L		W	R	S	O	M	C	A	E	B	K	LOGICAL UNIT COMMUNICATION TIME-OUT
05h	00h	D	T	L		W	R	S	O	M	C	A	E	B	K	LOGICAL UNIT DOES NOT RESPOND TO SELECTION
4Ch	00h	D	T	L	P	W	R	S	O	M	C	A	E	B	K	LOGICAL UNIT FAILED SELF-CONFIGURATION
3Eh	03h	D	T	L	P	W	R	S	O	M	C	A	E	B	K	LOGICAL UNIT FAILED SELF-TEST
3Eh	01h	D	T	L	P	W	R	S	O	M	C	A	E	B	K	LOGICAL UNIT FAILURE
5Dh	02h						R									LOGICAL UNIT FAILURE PREDICTION THRESHOLD EXCEEDED
3Eh	00h	D	T	L	P	W	R	S	O	M	C	A	E	B	K	LOGICAL UNIT HAS NOT SELF-CONFIGURED YET

Annex C contains the ASC and ASCQ assignments in numeric order.

Table 108 — ASC and ASCQ assignments (part 6 of 11)

D - DIRECT ACCESS DEVICE (SBC)
. T - SEQUENTIAL ACCESS DEVICE (SSC)
. L - PRINTER DEVICE (SSC)
. P - PROCESSOR DEVICE (SPC-2)
. . W - WRITE ONCE READ MULTIPLE DEVICE (SBC)
. . R - C/DVD DEVICE (MMC-2)
. . S - SCANNER DEVICE (SCSI-2)
. . . O - OPTICAL MEMORY DEVICE (SBC)
. . . M - MEDIA CHANGER DEVICE (SMC)
. . . C - COMMUNICATION DEVICE (SCSI-2)
. . . . A - STORAGE ARRAY DEVICE (SCC)
. . . . E - ENCLOSURE SERVICES DEVICE (SES)
. . . . B - SIMPLIFIED DIRECT-ACCESS DEVICE (RBC)
. . . . . K - OPTICAL CARD READER/WRITER DEVICE (OCRW)

Device Column key
blank = code not used
not blank = code used

ASC	ASCQ	D	T	L	P	W	R	S	O	M	C	A	E	B	K	Description
04h	01h	D	T	L	P	W	R	S	O	M	C	A	E	B	K	LOGICAL UNIT IS IN PROCESS OF BECOMING READY
68h	00h											A				LOGICAL UNIT NOT CONFIGURED
04h	00h	D	T	L	P	W	R	S	O	M	C	A	E	B	K	LOGICAL UNIT NOT READY, CAUSE NOT REPORTABLE
04h	04h	D	T	L			R		O					B		LOGICAL UNIT NOT READY, FORMAT IN PROGRESS
04h	02h	D	T	L	P	W	R	S	O	M	C	A	E	B	K	LOGICAL UNIT NOT READY, INITIALIZING CMD. REQUIRED
04h	08h						R									LOGICAL UNIT NOT READY, LONG WRITE IN PROGRESS
04h	03h	D	T	L	P	W	R	S	O	M	C	A	E	B	K	LOGICAL UNIT NOT READY, MANUAL INTERVENTION REQUIRED
04h	07h	D	T	L	P	W	R	S	O	M	C	A	E	B	K	LOGICAL UNIT NOT READY, OPERATION IN PROGRESS
04h	05h	D	T			W			O	M	C	A		B	K	LOGICAL UNIT NOT READY, REBUILD IN PROGRESS
04h	06h	D	T			W			O	M	C	A		B	K	LOGICAL UNIT NOT READY, RECALCULATION IN PROGRESS
04h	09h	D	T	L	P	W	R	S	O	M	C	A	E	B	K	LOGICAL UNIT NOT READY, SELF-TEST IN PROGRESS
25h	00h	D	T	L	P	W	R	S	O	M	C	A	E	B	K	LOGICAL UNIT NOT SUPPORTED
27h	02h	D	T			W	R		O					B	K	LOGICAL UNIT SOFTWARE WRITE PROTECTED
3Eh	04h	D	T	L	P	W	R	S	O	M	C	A	E	B	K	LOGICAL UNIT UNABLE TO UPDATE SELF-TEST LOG
5Eh	00h	D	T	L	P	W	R	S	O		C	A			K	LOW POWER CONDITION ON
15h	01h	D	T	L		W	R	S	O	M				B	K	MECHANICAL POSITIONING ERROR
3Bh	16h						R									MECHANICAL POSITIONING OR CHANGER ERROR
5Dh	01h						R							B		MEDIA FAILURE PREDICTION THRESHOLD EXCEEDED
53h	00h	D	T	L		W	R	S	O	M				B	K	MEDIA LOAD OR EJECT FAILED
6Fh	04h						R									MEDIA REGION CODE IS MISMATCHED TO LOGICAL UNIT REGION
3Fh	11h	D	T			W	R		O	M				B		MEDIUM AUXILIARY MEMORY ACCESSIBLE
3Bh	0Dh	D	T			W	R		O	M				B	K	MEDIUM DESTINATION ELEMENT FULL
31h	00h	D	T			W	R		O					B	K	MEDIUM FORMAT CORRUPTED
3Fh	10h	D	T			W	R		O	M				B		MEDIUM LOADABLE
3Bh	13h	D	T			W	R		O	M				B	K	MEDIUM MAGAZINE INSERTED
3Bh	14h	D	T			W	R		O	M				B	K	MEDIUM MAGAZINE LOCKED
3Bh	11h	D	T			W	R		O	M				B	K	MEDIUM MAGAZINE NOT ACCESSIBLE
3Bh	12h	D	T			W	R		O	M				B	K	MEDIUM MAGAZINE REMOVED
3Bh	15h	D	T			W	R		O	M				B	K	MEDIUM MAGAZINE UNLOCKED
3Ah	00h	D	T	L		W	R	S	O	M				B	K	MEDIUM NOT PRESENT
3Ah	03h	D	T			W	R		O	M				B		MEDIUM NOT PRESENT - LOADABLE
3Ah	04h	D	T			W	R		O	M				B		MEDIUM NOT PRESENT - MEDIUM AUXILIARY MEMORY ACCESSIBLE
3Ah	01h	D	T			W	R		O	M				B	K	MEDIUM NOT PRESENT - TRAY CLOSED
3Ah	02h	D	T			W	R		O	M				B	K	MEDIUM NOT PRESENT - TRAY OPEN
53h	02h	D	T			W	R		O	M				B	K	MEDIUM REMOVAL PREVENTED
3Bh	0Eh	D	T			W	R		O	M				B	K	MEDIUM SOURCE ELEMENT EMPTY
43h	00h	D	T	L	P	W	R	S	O	M	C	A	E	B	K	MESSAGE ERROR
3Fh	01h	D	T	L	P	W	R	S	O	M	C	A	E	B	K	MICROCODE HAS BEEN CHANGED
1Dh	00h	D	T			W	R		O					B	K	MISCOMPARE DURING VERIFY OPERATION
11h	0Ah	D	T						O					B	K	MISCORRECTED ERROR
2Ah	01h	D	T	L		W	R	S	O	M	C	A	E	B	K	MODE PARAMETERS CHANGED
67h	03h											A				MODIFICATION OF LOGICAL UNIT FAILED
69h	01h											A				MULTIPLE LOGICAL UNIT FAILURES
07h	00h	D	T	L		W	R	S	O	M				B	K	MULTIPLE PERIPHERAL DEVICES SELECTED
11h	03h	D	T			W		S	O					B	K	MULTIPLE READ ERRORS
67h	09h											A				MULTIPLY ASSIGNED LOGICAL UNIT

Annex C contains the ASC and ASCQ assignments in numeric order.

Table 108 — ASC and ASCQ assignments (part 7 of 11)

D - DIRECT ACCESS DEVICE (SBC)
. T - SEQUENTIAL ACCESS DEVICE (SSC)
. L - PRINTER DEVICE (SSC)
. P - PROCESSOR DEVICE (SPC-2)
. . W - WRITE ONCE READ MULTIPLE DEVICE (SBC)
. . R - C/DVD DEVICE (MMC-2)
. . S - SCANNER DEVICE (SCSI-2)
. . . O - OPTICAL MEMORY DEVICE (SBC)
. . . M - MEDIA CHANGER DEVICE (SMC)
. . . C - COMMUNICATION DEVICE (SCSI-2)
. . . . A - STORAGE ARRAY DEVICE (SCC)
. . . . E - ENCLOSURE SERVICES DEVICE (SES)
. . . . B - SIMPLIFIED DIRECT-ACCESS DEVICE (RBC)
. . . . . K - OPTICAL CARD READER/WRITER DEVICE (OCRW)

Device Column key
blank = code not used
not blank = code used

ASC	ASCQ	D	T	L	P	W	R	S	O	M	C	A	E	B	K	Description
00h	00h	D	T	L	P	W	R	S	O	M	C	A	E	B	K	NO ADDITIONAL SENSE INFORMATION
00h	15h						R									NO CURRENT AUDIO STATUS TO RETURN
32h	00h	D				W			O					B	K	NO DEFECT SPARE LOCATION AVAILABLE
11h	09h		T													NO GAP FOUND
01h	00h	D				W			O					B	K	NO INDEX/SECTOR SIGNAL
72h	05h						R									NO MORE TRACK RESERVATIONS ALLOWED
06h	00h	D				W	R		O	M				B	K	NO REFERENCE POSITION FOUND
02h	00h	D				W	R		O	M				B	K	NO SEEK COMPLETE
03h	01h		T													NO WRITE CURRENT
28h	00h	D	T	L	P	W	R	S	O	M	C	A	E	B	K	NOT READY TO READY CHANGE, MEDIUM MAY HAVE CHANGED
00h	16h	D	T	L	P	W	R	S	O	M	C	A	E	B	K	OPERATION IN PROGRESS
5Ah	01h	D	T			W	R		O	M				B	K	OPERATOR MEDIUM REMOVAL REQUEST
5Ah	00h	D	T	L	P	W	R	S	O	M				B	K	OPERATOR REQUEST OR STATE CHANGE INPUT
5Ah	03h	D	T			W	R		O			A		B	K	OPERATOR SELECTED WRITE PERMIT
5Ah	02h	D	T			W	R		O			A		B	K	OPERATOR SELECTED WRITE PROTECT
61h	02h							S								OUT OF FOCUS
4Eh	00h	D	T	L	P	W	R	S	O	M	C	A	E	B	K	OVERLAPPED COMMANDS ATTEMPTED
2Dh	00h		T													OVERWRITE ERROR ON UPDATE IN PLACE
63h	01h						R									PACKET DOES NOT FIT IN AVAILABLE SPACE
3Bh	05h			L												PAPER JAM
1Ah	00h	D	T	L	P	W	R	S	O	M	C	A	E	B	K	PARAMETER LIST LENGTH ERROR
26h	01h	D	T	L	P	W	R	S	O	M	C	A	E	B	K	PARAMETER NOT SUPPORTED
26h	02h	D	T	L	P	W	R	S	O	M	C	A	E	B	K	PARAMETER VALUE INVALID
2Ah	00h	D	T	L		W	R	S	O	M	C	A	E	B	K	PARAMETERS CHANGED
69h	02h											A				PARITY/DATA MISMATCH
1Fh	00h	D							O						K	PARTIAL DEFECT LIST TRANSFER
03h	00h	D	T	L		W		S	O					B	K	PERIPHERAL DEVICE WRITE FAULT
27h	05h		T				R									PERMANENT WRITE PROTECT
2Ch	06h						R									PERSISTENT PREVENT CONFLICT
27h	04h		T				R									PERSISTENT WRITE PROTECT
50h	02h		T													POSITION ERROR RELATED TO TIMING
3Bh	0Ch		T					S								POSITION PAST BEGINNING OF MEDIUM
3Bh	0Bh							S								POSITION PAST END OF MEDIUM
15h	02h	D	T			W	R		O					B	K	POSITIONING ERROR DETECTED BY READ OF MEDIUM
73h	01h						R									POWER CALIBRATION AREA ALMOST FULL
73h	03h						R									POWER CALIBRATION AREA ERROR
73h	02h						R									POWER CALIBRATION AREA IS FULL
29h	01h	D	T	L	P	W	R	S	O	M	C	A	E	B	K	POWER ON OCCURRED
29h	00h	D	T	L	P	W	R	S	O	M	C	A	E	B	K	POWER ON, RESET, OR BUS DEVICE RESET OCCURRED
5Eh	41h													B		POWER STATE CHANGE TO ACTIVE
5Eh	47h													B	K	POWER STATE CHANGE TO DEVICE CONTROL
5Eh	42h													B		POWER STATE CHANGE TO IDLE
5Eh	45h													B		POWER STATE CHANGE TO SLEEP
5Eh	43h													B		POWER STATE CHANGE TO STANDBY
42h	00h	D														POWER-ON OR SELF-TEST FAILURE (SHOULD USE 40 NN)
1Ch	01h	D							O					B	K	PRIMARY DEFECT LIST NOT FOUND

Annex C contains the ASC and ASCQ assignments in numeric order.

Table 108 — ASC and ASCQ assignments (part 8 of 11)

D - DIRECT ACCESS DEVICE (SBC)
. T - SEQUENTIAL ACCESS DEVICE (SSC)
. L - PRINTER DEVICE (SSC)
. P - PROCESSOR DEVICE (SPC-2)
. . W - WRITE ONCE READ MULTIPLE DEVICE (SBC)
. . R - C/DVD DEVICE (MMC-2)
. . S - SCANNER DEVICE (SCSI-2)
. . . O - OPTICAL MEMORY DEVICE (SBC)
. . . M - MEDIA CHANGER DEVICE (SMC)
. . . C - COMMUNICATION DEVICE (SCSI-2)
. . . . A - STORAGE ARRAY DEVICE (SCC)
. . . . E - ENCLOSURE SERVICES DEVICE (SES)
. . . . B - SIMPLIFIED DIRECT-ACCESS DEVICE (RBC)
. . . . . K - OPTICAL CARD READER/WRITER DEVICE (OCRW)

Device Column key
blank = code not used
not blank = code used

ASC	ASCQ	D	T	L	P	W	R	S	O	M	C	A	E	B	K	Description
73h	05h						R									PROGRAM MEMORY AREA IS FULL
73h	04h						R									PROGRAM MEMORY AREA UPDATE FAILURE
40h	00h	D														RAM FAILURE (SHOULD USE 40 NN)
15h	00h	D	T	L		W	R	S	O	M				B	K	RANDOM POSITIONING ERROR
11h	11h						R									READ ERROR - LOSS OF STREAMING
6Fh	03h						R									READ OF SCRAMBLED SECTOR WITHOUT AUTHENTICATION
3Bh	0Ah							S								READ PAST BEGINNING OF MEDIUM
3Bh	09h							S								READ PAST END OF MEDIUM
11h	01h	D	T			W	R	S	O					B	K	READ RETRIES EXHAUSTED
20h	04h		T													READ TYPE OPERATION WHILE IN WRITE CAPABLE STATE
6Ch	00h											A				REBUILD FAILURE OCCURRED
6Dh	00h											A				RECALCULATE FAILURE OCCURRED
14h	01h	D	T			W	R		O					B	K	RECORD NOT FOUND
14h	06h	D	T			W			O					B	K	RECORD NOT FOUND - DATA AUTO-REALLOCATED
14h	05h	D	T			W			O					B	K	RECORD NOT FOUND - RECOMMEND REASSIGNMENT
14h	00h	D	T	L		W	R	S	O					B	K	RECORDED ENTITY NOT FOUND
18h	08h						R									RECOVERD DATA WITH LINKING
18h	02h	D				W	R		O					B	K	RECOVERED DATA - DATA AUTO-REALLOCATED
18h	05h	D				W	R		O					B	K	RECOVERED DATA - RECOMMEND REASSIGNMENT
18h	06h	D				W	R		O					B	K	RECOVERED DATA - RECOMMEND REWRITE
17h	05h	D				W	R		O					B	K	RECOVERED DATA USING PREVIOUS SECTOR ID
18h	03h						R									RECOVERED DATA WITH CIRC
18h	07h	D				W			O					B	K	RECOVERED DATA WITH ECC - DATA REWRITTEN
18h	01h	D				W	R		O					B	K	RECOVERED DATA WITH ERROR CORR. & RETRIES APPLIED
18h	00h	D	T			W	R		O					B	K	RECOVERED DATA WITH ERROR CORRECTION APPLIED
18h	04h						R									RECOVERED DATA WITH L-EC
17h	03h	D	T			W	R		O					B	K	RECOVERED DATA WITH NEGATIVE HEAD OFFSET
17h	00h	D	T			W	R	S	O					B	K	RECOVERED DATA WITH NO ERROR CORRECTION APPLIED
17h	02h	D	T			W	R		O					B	K	RECOVERED DATA WITH POSITIVE HEAD OFFSET
17h	01h	D	T			W	R	S	O					B	K	RECOVERED DATA WITH RETRIES
17h	04h					W	R		O					B		RECOVERED DATA WITH RETRIES AND/OR CIRC APPLIED
17h	06h	D				W			O					B	K	RECOVERED DATA WITHOUT ECC - DATA AUTO-REALLOCATED
17h	09h	D				W	R		O					B	K	RECOVERED DATA WITHOUT ECC - DATA REWRITTEN
17h	07h	D				W	R		O					B	K	RECOVERED DATA WITHOUT ECC - RECOMMEND REASSIGNMENT
17h	08h	D				W	R		O					B	K	RECOVERED DATA WITHOUT ECC - RECOMMEND REWRITE
1Eh	00h	D				W			O					B	K	RECOVERED ID WITH ECC CORRECTION
3Fh	06h	D	T			W	R		O	M	C	A	E	B		REDUNDANCY GROUP CREATED OR MODIFIED
3Fh	07h	D	T			W	R		O	M	C	A	E	B		REDUNDANCY GROUP DELETED
6Bh	01h											A				REDUNDANCY LEVEL GOT BETTER
6Bh	02h											A				REDUNDANCY LEVEL GOT WORSE
2Ah	05h	D	T	L	P	W	R	S	O	M	C	A	E			REGISTRATIONS PREEMPTED
67h	05h											A				REMOVE OF LOGICAL UNIT FAILED
3Fh	0Eh	D	T	L	P	W	R	S	O	M	C	A	E			REPORTED LUNS DATA HAS CHANGED
3Bh	08h		T													REPOSITION ERROR
2Ah	03h	D	T	L	P	W	R	S	O	M	C	A	E		K	RESERVATIONS PREEMPTED
2Ah	04h	D	T	L	P	W	R	S	O	M	C	A	E			RESERVATIONS RELEASED

Annex C contains the ASC and ASCQ assignments in numeric order.

Table 108 — ASC and ASCQ assignments (part 9 of 11)

D - DIRECT ACCESS DEVICE (SBC)
. T - SEQUENTIAL ACCESS DEVICE (SSC)
. L - PRINTER DEVICE (SSC)
. P - PROCESSOR DEVICE (SPC-2)
. . W - WRITE ONCE READ MULTIPLE DEVICE (SBC)
. . R - C/DVD DEVICE (MMC-2)
. . S - SCANNER DEVICE (SCSI-2)
. . . O - OPTICAL MEMORY DEVICE (SBC)
. . . M - MEDIA CHANGER DEVICE (SMC)
. . . C - COMMUNICATION DEVICE (SCSI-2)
. . . . A - STORAGE ARRAY DEVICE (SCC)
. . . . E - ENCLOSURE SERVICES DEVICE (SES)
. . . . B - SIMPLIFIED DIRECT-ACCESS DEVICE (RBC)
. . . . . K - OPTICAL CARD READER/WRITER DEVICE (OCRW)

Device Column key
blank = code not used
not blank = code used

ASC	ASCQ	D	T	L	P	W	R	S	O	M	C	A	E	B	K	Description
36h	00h			L												RIBBON, INK, OR TONER FAILURE
73h	06h						R									RMA/PMA IS ALMOST FULL
37h	00h	D	T	L		W	R	S	O	M	C	A	E	B	K	ROUNDED PARAMETER
5Ch	00h	D							O							RPL STATUS CHANGE
39h	00h	D	T	L		W	R	S	O	M	C	A	E		K	SAVING PARAMETERS NOT SUPPORTED
62h	00h							S								SCAN HEAD POSITIONING ERROR
29h	02h	D	T	L	P	W	R	S	O	M	C	A	E	B	K	SCSI BUS RESET OCCURRED
47h	00h	D	T	L	P	W	R	S	O	M	C	A	E	B	K	SCSI PARITY ERROR
47h	02h	D	T	L	P	W	R	S	O	M	C	A	E	B	K	SCSI PARITY ERROR DETECTED DURING ST DATA PHASE
54h	00h				P											SCSI TO HOST SYSTEM INTERFACE FAILURE
45h	00h	D	T	L	P	W	R	S	O	M	C	A	E	B	K	SELECT OR RESELECT FAILURE
3Bh	00h		T	L												SEQUENTIAL POSITIONING ERROR
5Dh	45h	D												B		SERVO IMPENDING FAILURE ACCESS TIMES TOO HIGH
5Dh	47h	D												B		SERVO IMPENDING FAILURE CHANNEL PARAMETRICS
5Dh	48h	D												B		SERVO IMPENDING FAILURE CONTROLLER DETECTED
5Dh	42h	D												B		SERVO IMPENDING FAILURE DATA ERROR RATE TOO HIGH
5Dh	4Ch	D												B		SERVO IMPENDING FAILURE DRIVE CALIBRATION RETRY COUNT
5Dh	41h	D												B		SERVO IMPENDING FAILURE DRIVE ERROR RATE TOO HIGH
5Dh	40h	D												B		SERVO IMPENDING FAILURE GENERAL HARD DRIVE FAILURE
5Dh	43h	D												B		SERVO IMPENDING FAILURE SEEK ERROR RATE TOO HIGH
5Dh	4Ah	D												B		SERVO IMPENDING FAILURE SEEK TIME PERFORMANCE
5Dh	4Bh	D												B		SERVO IMPENDING FAILURE SPIN-UP RETRY COUNT
5Dh	46h	D												B		SERVO IMPENDING FAILURE START UNIT TIMES TOO HIGH
5Dh	49h	D												B		SERVO IMPENDING FAILURE THROUGHPUT PERFORMANCE
5Dh	44h	D												B		SERVO IMPENDING FAILURE TOO MANY BLOCK REASSIGNS
72h	00h						R									SESSION FIXATION ERROR
72h	03h						R									SESSION FIXATION ERROR - INCOMPLETE TRACK IN SESSION
72h	01h						R									SESSION FIXATION ERROR WRITING LEAD-IN
72h	02h						R									SESSION FIXATION ERROR WRITING LEAD-OUT
00h	03h		T													SETMARK DETECTED
3Bh	04h			L												SLEW FAILURE
5Dh	03h						R									SPARE AREA EXHAUSTION PREDICTION THRESHOLD EXCEEDED
3Fh	08h	D	T			W	R		O	M	C	A	E	B		SPARE CREATED OR MODIFIED
3Fh	09h	D	T			W	R		O	M	C	A	E	B		SPARE DELETED
5Dh	55h	D												B		SPINDLE IMPENDING FAILURE ACCESS TIMES TOO HIGH
5Dh	57h	D												B		SPINDLE IMPENDING FAILURE CHANNEL PARAMETRICS
5Dh	58h	D												B		SPINDLE IMPENDING FAILURE CONTROLLER DETECTED
5Dh	52h	D												B		SPINDLE IMPENDING FAILURE DATA ERROR RATE TOO HIGH
5Dh	5Ch	D												B		SPINDLE IMPENDING FAILURE DRIVE CALIBRATION RETRY COUNT
5Dh	51h	D												B		SPINDLE IMPENDING FAILURE DRIVE ERROR RATE TOO HIGH
5Dh	50h	D												B		SPINDLE IMPENDING FAILURE GENERAL HARD DRIVE FAILURE
5Dh	53h	D												B		SPINDLE IMPENDING FAILURE SEEK ERROR RATE TOO HIGH
5Dh	5Ah	D												B		SPINDLE IMPENDING FAILURE SEEK TIME PERFORMANCE
5Dh	5Bh	D												B		SPINDLE IMPENDING FAILURE SPIN-UP RETRY COUNT
5Dh	56h	D												B		SPINDLE IMPENDING FAILURE START UNIT TIMES TOO HIGH
5Dh	59h	D												B		SPINDLE IMPENDING FAILURE THROUGHPUT PERFORMANCE

Annex C contains the ASC and ASCQ assignments in numeric order.

Table 108 — ASC and ASCQ assignments (part 10 of 11)

D - DIRECT ACCESS DEVICE (SBC)
. T - SEQUENTIAL ACCESS DEVICE (SSC)
. L - PRINTER DEVICE (SSC)
. P - PROCESSOR DEVICE (SPC-2)
. . W - WRITE ONCE READ MULTIPLE DEVICE (SBC)
. . R - C/DVD DEVICE (MMC-2)
. . S - SCANNER DEVICE (SCSI-2)
. . . O - OPTICAL MEMORY DEVICE (SBC)
. . . M - MEDIA CHANGER DEVICE (SMC)
. . . C - COMMUNICATION DEVICE (SCSI-2)
. . . . A - STORAGE ARRAY DEVICE (SCC)
. . . . E - ENCLOSURE SERVICES DEVICE (SES)
. . . . B - SIMPLIFIED DIRECT-ACCESS DEVICE (RBC)
. . . . . K - OPTICAL CARD READER/WRITER DEVICE (OCRW)

Device Column key
blank = code not used
not blank = code used

ASC	ASCQ	D	T	L	P	W	R	S	O	M	C	A	E	B	K	Description
5Dh	54h	D												B		SPINDLE IMPENDING FAILURE TOO MANY BLOCK REASSIGNS
09h	03h					W	R		O							SPINDLE SERVO FAILURE
5Ch	02h	D							O							SPINDLES NOT SYNCHRONIZED
5Ch	01h	D							O							SPINDLES SYNCHRONIZED
5Eh	04h	D	T	L	P	W	R	S	O		C	A			K	STANDBY CONDITION ACTIVATED BY COMMAND
5Eh	02h	D	T	L	P	W	R	S	O		C	A			K	STANDBY CONDITION ACTIVATED BY TIMER
6Bh	00h											A				STATE CHANGE HAS OCCURRED
1Bh	00h	D	T	L	P	W	R	S	O	M	C	A	E	B	K	SYNCHRONOUS DATA TRANSFER ERROR
55h	01h	D							O					B	K	SYSTEM BUFFER FULL
55h	00h				P											SYSTEM RESOURCE FAILURE
4Dh	NNh	D	T	L	P	W	R	S	O	M	C	A	E	B	K	TAGGED OVERLAPPED COMMANDS (NN = QUEUE TAG)
33h	00h		T													TAPE LENGTH ERROR
3Bh	03h			L												TAPE OR ELECTRONIC VERTICAL FORMS UNIT NOT READY
3Bh	01h		T													TAPE POSITION ERROR AT BEGINNING-OF-MEDIUM
3Bh	02h		T													TAPE POSITION ERROR AT END-OF-MEDIUM
3Fh	00h	D	T	L	P	W	R	S	O	M	C	A	E	B	K	TARGET OPERATING CONDITIONS HAVE CHANGED
0Dh	01h	D	T	L	P	W	R	S	O		C	A			K	THIRD PARTY DEVICE FAILURE
5Bh	01h	D	T	L	P	W	R	S	O	M					K	THRESHOLD CONDITION MET
26h	03h	D	T	L	P	W	R	S	O	M	C	A	E		K	THRESHOLD PARAMETERS NOT SUPPORTED
3Eh	02h	D	T	L	P	W	R	S	O	M	C	A	E	B	K	TIMEOUT ON LOGICAL UNIT
26h	08h	D	T	L	P	W	R	S	O		C				K	TOO MANY SEGMENT DESCRIPTORS
26h	06h	D	T	L	P	W	R	S	O		C				K	TOO MANY TARGET DESCRIPTORS
2Ch	01h							S								TOO MANY WINDOWS SPECIFIED
09h	00h	D	T			W	R		O					B		TRACK FOLLOWING ERROR
09h	01h					W	R		O						K	TRACKING SERVO FAILURE
29h	06h	D	T	L	P	W	R	S	O	M	C	A	E	B	K	TRANSCEIVER MODE CHANGED TO LVD
29h	05h	D	T	L	P	W	R	S	O	M	C	A	E	B	K	TRANSCEIVER MODE CHANGED TO SINGLE-ENDED
61h	01h							S								UNABLE TO ACQUIRE VIDEO
57h	00h						R									UNABLE TO RECOVER TABLE-OF-CONTENTS
26h	0Ah	D	T	L	P	W	R	S	O		C				K	UNEXPECTED INEXACT SEGMENT
53h	01h		T													UNLOAD TAPE FAILURE
08h	04h	D	T	L	P	W	R	S	O		C				K	UNREACHABLE COPY TARGET
11h	00h	D	T			W	R	S	O					B	K	UNRECOVERED READ ERROR
11h	04h	D				W			O					B	K	UNRECOVERED READ ERROR - AUTO REALLOCATE FAILED
11h	0Bh	D				W			O					B	K	UNRECOVERED READ ERROR - RECOMMEND REASSIGNMENT
11h	0Ch	D				W			O					B	K	UNRECOVERED READ ERROR - RECOMMEND REWRITE THE DATA
46h	00h	D	T	L	P	W	R	S	O	M	C			B	K	UNSUCCESSFUL SOFT RESET
35h	01h	D	T	L	P	W	R	S	O	M	C	A	E	B	K	UNSUPPORTED ENCLOSURE FUNCTION
26h	09h	D	T	L	P	W	R	S	O		C				K	UNSUPPORTED SEGMENT DESCRIPTOR TYPE CODE
26h	07h	D	T	L	P	W	R	S	O		C				K	UNSUPPORTED TARGET DESCRIPTOR TYPE CODE
59h	00h								O							UPDATED BLOCK READ
61h	00h							S								VIDEO ACQUISITION ERROR
65h	00h	D	T	L	P	W	R	S	O	M	C	A	E	B	K	VOLTAGE FAULT
3Fh	0Ah	D	T			W	R		O	M	C	A	E	B	K	VOLUME SET CREATED OR MODIFIED
3Fh	0Ch	D	T			W	R		O	M	C	A	E	B	K	VOLUME SET DEASSIGNED
3Fh	0Bh	D	T			W	R		O	M	C	A	E	B	K	VOLUME SET DELETED

Annex C contains the ASC and ASCQ assignments in numeric order.

Table 108 — ASC and ASCQ assignments (part 11 of 11)

D - DIRECT ACCESS DEVICE (SBC)
. T - SEQUENTIAL ACCESS DEVICE (SSC)
. L - PRINTER DEVICE (SSC)
. P - PROCESSOR DEVICE (SPC-2)
. . W - WRITE ONCE READ MULTIPLE DEVICE (SBC)
. . R - C/DVD DEVICE (MMC-2)
. . S - SCANNER DEVICE (SCSI-2)
. . . O - OPTICAL MEMORY DEVICE (SBC)
. . . M - MEDIA CHANGER DEVICE (SMC)
. . . C - COMMUNICATION DEVICE (SCSI-2)
. . . . A - STORAGE ARRAY DEVICE (SCC)
. . . . E - ENCLOSURE SERVICES DEVICE (SES)
. . . . B - SIMPLIFIED DIRECT-ACCESS DEVICE (RBC)
. . . . . K - OPTICAL CARD READER/WRITER DEVICE (OCRW)

Device Column key
blank = code not used
not blank = code used

ASC	ASCQ	D	T	L	P	W	R	S	O	M	C	A	E	B	K	Description
3Fh	0Dh	D	T			W	R		O	M	C	A	E	B	K	VOLUME SET REASSIGNED
0Bh	00h	D	T	L	P	W	R	S	O	M	C	A	E	B	K	WARNING
0Bh	02h	D	T	L	P	W	R	S	O	M	C	A	E	B	K	WARNING - ENCLOSURE DEGRADED
0Bh	01h	D	T	L	P	W	R	S	O	M	C	A	E	B	K	WARNING - SPECIFIED TEMPERATURE EXCEEDED
50h	00h		T													WRITE APPEND ERROR
50h	01h		T													WRITE APPEND POSITION ERROR
0Ch	00h		T				R	S								WRITE ERROR
0Ch	02h	D				W			O					B	K	WRITE ERROR - AUTO REALLOCATION FAILED
0Ch	09h						R									WRITE ERROR - LOSS OF STREAMING
0Ch	0Ah						R									WRITE ERROR - PADDING BLOCKS ADDED
0Ch	03h	D				W			O					B	K	WRITE ERROR - RECOMMEND REASSIGNMENT
0Ch	01h														K	WRITE ERROR - RECOVERED WITH AUTO REALLOCATION
0Ch	08h						R									WRITE ERROR - RECOVERY FAILED
0Ch	07h						R									WRITE ERROR - RECOVERY NEEDED
27h	00h	D	T			W	R		O					B	K	WRITE PROTECTED
20h	05h		T													WRITE TYPE OPERATION WHILE IN READ CAPABLE STATE
31h	02h						R									ZONED FORMATTING FAILED DUE TO SPARE LINKING
80h Through FFh	xxh						\ > /									Vendor specific.
xxh Through xxh	80h FFh						\ > /									Vendor specific QUALIFICATION OF STANDARD ASC.
																ALL CODES NOT SHOWN ARE RESERVED.

Annex C contains the ASC and ASCQ assignments in numeric order.

7.21 RESERVE(10) command

7.21.1 RESERVE(10) command introduction

The RESERVE(10) command (see table 109) is used to reserve a logical unit.

Table 109 — RESERVE(10) command

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

The RESERVE and RELEASE commands provide the basic mechanism for contention resolution in multiple-initiator systems. The third-party reservation allows logical units to be reserved for another specified SCSI device. See 5.5.1 for a general description of reservations and the commands that manage them.

If the RESERVE(10) command is implemented, then the RELEASE(10) also shall be implemented.

Byte 1 Bit 0 and Byte 2 provide an obsolete way to reserve extents within a logical unit. If Byte 1, Bit 0 is equal to one, device servers not implementing the obsolete capability shall terminate the command with CHECK CONDITION status and the sense key shall be set to ILLEGAL REQUEST.

7.21.2 Logical unit reservation

Logical unit reservations are mandatory if the RESERVE(10) command is implemented. This command shall request that the entire logical unit be reserved for the exclusive use of the initiator until the reservation is superseded by another valid RESERVE command from the same initiator or until released by a RELEASE command from the same initiator that made the reservation, by a TARGET RESET task management function performed by any initiator, by a hard reset condition, or by a power on cycle. A logical unit reservation shall not be granted if the logical unit is reserved by another initiator. It shall be permissible for an initiator to reserve a logical unit that is currently reserved by that initiator. If the LONGID bit or the 3RDPTY bit is zero then the PARAMETER LIST LENGTH field shall be ignored.

If the logical unit is reserved for another initiator, the device server shall return RESERVATION CONFLICT status.

After honoring a logical unit reservation, the device server shall check each newly received command for reservation conflicts. See 5.5.1.

For multiple port implementations, devices on other ports (i.e., the ports that do not include the initiator to which the reservation has been granted) also shall be denied access to the logical unit as described in the preceding paragraph.

7.21.3 Third-party reservation

The third-party reservation for the RESERVE(10) command allows an application client to reserve a logical unit within a logical unit for another SCSI device. The SCSI port through which the SCSI device being reserved shall be within the same SCSI domain and use the same SCSI protocol as the SCSI port through which the RESERVE(10) command is received. If the SCSI ports are not within the same SCSI domain or do not both have the same SCSI protocol the device server shall return a CHECK CONDITION status with a sense key of ILLEGAL REQUEST. Third-party reservations are intended for use in multiple initiator systems that use the EXTENDED COPY command.

If the third-party (3RDPTY) bit is zero, then a third-party reservation is not requested. If the 3RDPTY bit is zero then the LONGID bit shall be ignored. If the 3RDPTY bit is one then the device server shall reserve the specified logical unit for the SCSI device specified in the THIRD-PARTY DEVICE ID field. Device ID formats are protocol specific. The device server shall preserve the reservation until it is superseded by another valid RESERVE command from the initiator that made the reservation or until it is released by the same initiator, by a TARGET RESET task management function performed by any initiator, a hard reset condition, or by a power on cycle. The device server shall ignore any attempt to release the reservation made by any other initiator.

After a third-party reservation has been granted, the initiator that sent the RESERVE command shall be treated like any other initiator. Reservation conflicts shall occur in all cases where another initiator is not allowed access due to the reservation.

If independent sets of mode parameters are implemented, a third-party reservation shall cause the device server to transfer the set of mode parameters in effect for the application client that sent the RESERVE command to the mode parameters used for commands from the third-party device. Any subsequent command issued by the third-party device shall be executed according to the mode parameters in effect for the application client that sent the RESERVE command.

NOTE 34 This transfer of the mode parameters is applicable to device servers that store mode information independently for different initiators. This mechanism allows an application client to set the mode parameters of a target for the use of a copy master (i.e., the third-party device). The third-party copy master may subsequently issue a MODE SELECT command to modify the mode parameters.

If the THIRD-PARTY DEVICE ID value associated with the reservation release is smaller than 255, the LONGID bit may be zero and the ID value sent in the CDB. Device ID formats are protocol specific. If the THIRD-PARTY DEVICE ID is greater than 255, the LONGID bit shall be one. If the LONGID bit is one, the THIRD-PARTY DEVICE ID field in the CDB shall be ignored. If the LONGID bit is one, the parameter list length shall be at least eight. If the LONGID bit is one and the parameter list length is less than eight, the device server shall return a CHECK CONDITION status with a sense key of ILLEGAL REQUEST.

Device servers that support device IDs greater than 255 shall accept commands with LONGID equal to one. Device servers whose devices IDs are limited to 255 or smaller may reject commands with LONGID equal to one with CHECK CONDITION status and a sense key of ILLEGAL REQUEST.

If the LONGID bit is one, the parameter list length shall be eight, and the parameter list shall have the format shown in table 110. If the LONGID bit is one and the parameter list length is not eight, the device server shall return a CHECK CONDITION status with a sense key of ILLEGAL REQUEST.

Table 110 — RESERVE(10) ID only parameter list

Bit / Byte	7	6	5	4	3	2	1	0
0	(MSB)							
	THIRD-PARTY DEVICE ID							
7								(LSB)

7.21.4 Superseding reservations

Superseding reservations is mandatory if the RELEASE(10) command is implemented. An application client that holds a current logical unit reservation may modify that reservation by issuing another RESERVE command to the same logical unit. The superseding RESERVE command shall release the previous reservation state when the new reservation request is granted. The current reservation shall not be modified if the superseding reservation request is not granted. If the superseding reservation cannot be granted because of conflicts with a previous reservation, other than the reservation being superseded, then the device server shall return RESERVATION CONFLICT status.

NOTE 35 Superseding reservations allow the SCSI device ID in a third-party reservation to be changed. This capability is necessary for certain situations when using the EXTENDED COPY command.

7.22 RESERVE(6) command

The RESERVE(6) command (see table 111) is used to reserve a logical unit. This subclause describes only those instances where the RESERVE(6) command differs from the RESERVE(10) command. Except for the instances described in this subclause, the RESERVE(6) command shall function exactly like the RESERVE(10) command (see 7.21).

Table 111 — RESERVE(6) command

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

Obsolete Bits 1 through 4 of Byte 1 provided a method, limited to device addresses 0 through 7, to handle third-party reservations in earlier versions of the SCSI standard. The obsolete method has been replaced by the RESERVE(10) and RELEASE(10).

Byte 1 Bit 0 and Bytes 2 through 4 provide an obsolete way to reserve extents within a logical unit. If Byte 1, Bit 0 is equal to one, device servers not implementing the obsolete capability shall terminate the command with CHECK CONDITION status and the sense key shall be set to ILLEGAL REQUEST.

7.23 SEND DIAGNOSTIC command

The SEND DIAGNOSTIC command (see table 112) requests the device server to perform diagnostic operations on the target, on the logical unit, or on both. Targets that support this command shall implement, at a minimum, the default self-test feature (i.e., the SELFTEST bit equal to one and a parameter list length of zero). When the SELFTEST bit is zero and the SELF-TEST CODE field contains 000b, this command is usually followed by a RECEIVE DIAGNOSTIC RESULTS (see 7.15) command.

Table 112 — SEND DIAGNOSTIC command

Bit / Byte	7	6	5	4	3	2	1	0
0	OPERATION CODE (1Dh)							
1	SELF-TEST CODE			PF	Reserved	SELFTEST	DEVOFFL	UNITOFFL
2	Reserved							
3	(MSB)			PARAMETER LIST LENGTH				
4								(LSB)
5	CONTROL							

When the SELFTEST bit is one the SELF-TEST CODE field shall contain 000b. When the SELFTEST bit is zero, the contents of SELF-TEST CODE field are specified in table 113.

Table 113 — SELF-TEST CODE field values

Value	Name	Description
000b		This value shall be used when the SELFTEST bit is set to one or if the SEND DIAGNOSTIC command is not invoking one of the other self-test functions such as enclosure services (see SES) or the Translate Address page (see SBC).
001b	Background short self-test	The device server shall start its short self-test (see 5.4.2) in the background mode (see 5.4.3.2). The PARAMETER LIST LENGTH field shall contain zero.
010b	Background extended self-test	The device server shall start its extended self-test (see 5.4.2) in the background mode (see 5.4.3.2). The PARAMETER LIST LENGTH field shall contain zero.
011b	Reserved	
100b	Abort background self-test	The device server shall abort the current self-test running in background mode. The PARAMETER LIST LENGTH field shall contain zero. This value is only valid if a previous SEND DIAGNOSTIC command specified a background self-test function and that self-test has not completed. If either of these conditions is not met, then the device server shall return a CHECK CONDITION status with a sense key of ILLEGAL REQUEST and an additional sense code of INVALID FIELD IN CDB.
101b	Foreground short self-test	The device server shall start its short self-test (see 5.4.2) in the foreground mode (see 5.4.3.1). The PARAMETER LIST LENGTH field shall contain zero.
110b	Foreground extended self-test	The device server shall start its extended self-test (see 5.4.2) in the foreground mode (see 5.4.3.1). The PARAMETER LIST LENGTH field shall contain zero.
111b	Reserved	

A page format (PF) bit of one specifies that the SEND DIAGNOSTIC parameters and any parameters returned by a following RECEIVE DIAGNOSTIC RESULTS command shall conform to the page structure as specified in this standard. See 8.1 for the definition of diagnostic pages.

A PF bit of zero indicates that all SEND DIAGNOSTIC parameters are vendor specific. If the content of the PARAMETER LIST LENGTH field is zero and the SEND DIAGNOSTIC command will not be followed by a corre-

sponding RECEIVE DIAGNOSTIC RESULTS command then the PF bit shall be zero. The implementation of the PF bit is optional.

A self-test (SELFTEST) bit of one directs the device server to complete the target's default self-test. If the self-test successfully passes, the command shall be terminated with GOOD status; otherwise, the command shall be terminated with CHECK CONDITION status and the sense key shall be set to HARDWARE ERROR.

A SELFTEST bit of zero requests that the device server perform the diagnostic operation specified by the SELF-TEST CODE field or in the parameter list. The diagnostic operation might or might not require the device server to return parameter data that contains diagnostic results. If the return of parameter data is not required, the return of GOOD status indicates successful completion of the diagnostic operation. If the return of parameter data is required, the device server shall either:

a) perform the requested diagnostic operation, prepare the parameter data to be returned, and indicate completion by returning GOOD status. The application client issues a RECEIVE DIAGNOSTIC RESULTS command to recover the parameter data; or
b) accept the parameter list, and if no errors are detected in the parameter list, return GOOD status. The requested diagnostic operation and the preparation of the parameter data to be returned are performed upon receipt of a RECEIVE DIAGNOSTIC RESULTS command.

A UNITOFFL bit of one grants permission to the device server to perform diagnostic operations that may affect the user accessible medium on the logical unit (e.g., write operations to the user accessible medium, or repositioning of the medium on sequential access devices). The implementation of the UNITOFFL bit is optional. A UNITOFFL bit of zero prohibits any diagnostic operations that may be detected by subsequent tasks. When the SELFTEST bit is zero, the UNITOFFL bit shall be ignored.

A DEVOFFL bit of one grants permission to the device server to perform diagnostic operations that may affect all the logical units on a target (e.g., alteration of reservations, log parameters, or sense data). The implementation of the DEVOFFL bit is optional. A DEVOFFL bit of zero prohibits diagnostic operations that may be detected by subsequent tasks. When the SELFTEST bit is zero, the DEVOFFL bit shall be ignored.

The PARAMETER LIST LENGTH field specifies the length in bytes of the parameter list that shall be transferred from the application client to the device server. A parameter list length of zero indicates that no data shall be transferred. This condition shall not be considered an error. If the specified parameter list length results in the truncation of one or more pages (PF bit set to one) the device server shall return CHECK CONDITION status with a sense key of ILLEGAL REQUEST and an additional sense code of INVALID FIELD IN CDB.

NOTE 36 To ensure that the diagnostic command information is not destroyed by a command sent from another initiator, either the SEND DIAGNOSTIC command should be linked to the RECEIVE DIAGNOSTIC RESULTS command or the logical unit should be reserved.

7.24 SET DEVICE IDENTIFIER command

The SET DEVICE IDENTIFIER command (see table 114) requests that the device identifier information in the logical unit be set to the value received in the SET DEVICE IDENTIFIER parameter list. As defined in the SCC-2 standard, the SET DEVICE IDENTIFIER command is the SET PERIPHERAL DEVICE/COMPONENT DEVICE IDENTIFIER service action of the MAINTENANCE OUT command. Additional MAINTENANCE IN and MAINTENANCE OUT service actions are defined in SCC-2 and in this standard.

The MAINTENANCE OUT service actions defined only in SCC-2 shall apply only to SCSI devices that return a device type of 0Ch or the SCCS bit equal to one in their standard INQUIRY data. When a SCSI device returns a device type of 0Ch or the SCCS bit equal to one in its standard INQUIRY data, the implementation requirements for the SCC-2 MAINTENANCE OUT service actions shall be as specified in SCC-2. Otherwise the MAINTENANCE OUT service action definitions and implementation requirements stated in this standard shall apply.

On successful completion of a SET DEVICE IDENTIFIER command a unit attention shall be generated for all initiators except the one that issued the service action. When reporting the unit attention condition the additional sense code shall be set to DEVICE IDENTIFIER CHANGED.

Table 114 — SET DEVICE IDENTIFIER command

Bit / Byte	7	6	5	4	3	2	1	0
0	OPERATION CODE (A4h)							
1	Reserved			SERVICE ACTION (06h)				
2	Reserved							
3	Reserved							
4	Restricted							
5								
6	(MSB)							
7	PARAMETER LIST LENGTH							
8								
9								(LSB)
10	Reserved						Restricted	Reserved
11	CONTROL							

SCC-2 defines specific usages for bytes 4 and 5, and bit 1 in byte 10, however these fields are reserved for the SET DEVICE IDENTIFIER command defined by this standard.

The PARAMETER LIST LENGTH field specifies the length in bytes of the identifier that shall be transferred from the application client to the device server. The maximum value for this field shall be 512 bytes. A parameter list length of zero indicates that no data shall be transferred, and that subsequent REPORT DEVICE IDENTIFIER commands shall return an Identifier length of zero. Logical units that implement this command shall be capable of accepting a parameter list length of 64 bytes or less. If the parameter list length exceeds 64 bytes and the logical unit is not capable of storing the requested number of bytes, then the device server shall return CHECK CONDITION status with the sense key set to ILLEGAL REQUEST and an additional sense code of INVALID FIELD IN CDB.

The SET DEVICE IDENTIFIER parameter list (see table 115) contains the identifier to be set by the addressed logical unit.

Table 115 — SET DEVICE IDENTIFIER parameter list

Bit / Byte	7	6	5	4	3	2	1	0
0	IDENTIFIER							
n								

The IDENTIFIER field is a value selected by the application client using mechanisms outside the scope of this standard to be returned in subsequent REPORT DEVICE IDENTIFIER commands.

7.25 TEST UNIT READY command

The TEST UNIT READY command (see table 116) provides a means to check if the logical unit is ready. This is not a request for a self-test. If the logical unit is able to accept an appropriate medium-access command without returning CHECK CONDITION status, this command shall return a GOOD status. If the logical unit is unable to become operational or is in a state such that an application client action (e.g., START UNIT command) is required to make the unit ready, the device server shall return CHECK CONDITION status with a sense key of NOT READY.

Table 116 — TEST UNIT READY command

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

Table 117 defines the suggested GOOD and CHECK CONDITION status responses to the TEST UNIT READY command. Other conditions, including deferred errors, may result in other responses (e.g., BUSY or RESERVATION CONFLICT status).

Table 117 — Preferred TEST UNIT READY responses

Status	Sense Key	Additional Sense Code
GOOD	NO SENSE	NO ADDITIONAL SENSE INFORMATION or other valid additional sense code.
CHECK CONDITION	ILLEGAL REQUEST	LOGICAL UNIT NOT SUPPORTED
CHECK CONDITION	NOT READY	LOGICAL UNIT DOES NOT RESPOND TO SELECTION
CHECK CONDITION	NOT READY	MEDIUM NOT PRESENT
CHECK CONDITION	NOT READY	LOGICAL UNIT NOT READY, CAUSE NOT REPORTABLE
CHECK CONDITION	NOT READY	LOGICAL UNIT IS IN PROCESS OF BECOMING READY
CHECK CONDITION	NOT READY	LOGICAL UNIT NOT READY, INITIALIZING COMMAND REQUIRED
CHECK CONDITION	NOT READY	LOGICAL UNIT NOT READY, MANUAL INTERVENTION REQUIRED
CHECK CONDITION	NOT READY	LOGICAL UNIT NOT READY, FORMAT IN PROGRESS

7.26 WRITE BUFFER command

7.26.1 WRITE BUFFER command introduction

The WRITE BUFFER command (see table 118) is used in conjunction with the READ BUFFER command as a diagnostic function for testing logical unit memory in the target SCSI device and the integrity of the service delivery subsystem. Additional modes are provided for downloading microcode and for downloading and saving microcode.

Table 118 — WRITE BUFFER command

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

This command shall not alter any medium of the logical unit when the data mode or the combined header and data mode is specified.

The function of this command and the meaning of fields within the CDB depend on the contents of the MODE field. The MODE field is defined in table 119.

Table 119 — WRITE BUFFER MODE field

MODE	Description	Implementation requirements
0000b	Write combined header and data	Optional
0001b	Vendor specific	Vendor specific
0010b	Write data	Optional
0011b	Reserved	Reserved
0100b	Download microcode	Optional
0101b	Download microcode and save	Optional
0110b	Download microcode with offsets	Optional
0111b	Download microcode with offsets and save	Optional
1000b - 1001b	Reserved	Reserved
1010b	Echo buffer	Optional
1011b - 1111b	Reserved	Reserved

NOTES
37 Modes 0000b and0 001b are not recommended.
38 When downloading microcode with buffer offsets, the WRITE BUFFER command mode should be 0110b or 0111b.

7.26.2 Combined header and data mode (0000b)

In this mode, data to be transferred is preceded by a four-byte header. The four-byte header consists of all reserved bytes. The BUFFER ID and the BUFFER OFFSET fields shall be zero. The PARAMETER LIST LENGTH field specifies the maximum number of bytes that shall be transferred from the Data-Out Buffer. This number includes four bytes of header, so the data length to be stored in the device server's buffer is parameter list length minus four. The application client should attempt to ensure that the parameter list length is not greater than four plus the BUFFER CAPACITY field value (see 7.13.2) that is returned in the header of the READ BUFFER command (mode 0000b). If the parameter list length exceeds the buffer capacity the device server shall return CHECK CONDITION status and shall set the sense key to ILLEGAL REQUEST.

7.26.3 Vendor specific mode (0001b)

In this mode, the meaning of the BUFFER ID, BUFFER OFFSET, and PARAMETER LIST LENGTH fields are not specified by this standard.

7.26.4 Data mode (0010b)

In this mode, the Data-Out Buffer contains buffer data destined for the logical unit. The BUFFER ID field identifies a specific buffer within the logical unit. The vendor assigns buffer ID codes to buffers within the logical unit. Buffer ID zero shall be supported. If more than one buffer is supported, additional buffer ID codes shall be assigned contiguously, beginning with one. If an unsupported buffer ID code is selected, the device server shall return CHECK CONDITION status and shall set the sense key to ILLEGAL REQUEST with an additional sense code of INVALID FIELD IN CDB.

Data are written to the logical unit buffer starting at the location specified by the BUFFER OFFSET field. The application client should conform to the offset boundary requirements returned in the READ BUFFER descriptor. If the device server is unable to accept the specified buffer offset, it shall return CHECK CONDITION status and it shall set the sense key to ILLEGAL REQUEST with an additional sense code of INVALID FIELD IN CDB.

The PARAMETER LIST LENGTH field specifies the maximum number of bytes that shall be transferred from the Data-Out Buffer to be stored in the specified buffer beginning at the buffer offset. The application client should attempt to ensure that the parameter list length plus the buffer offset does not exceed the capacity of the specified buffer. The capacity of the buffer may be determined by the BUFFER CAPACITY field in the READ BUFFER descriptor. If the BUFFER OFFSET and PARAMETER LIST LENGTH fields specify a transfer in excess of the buffer capacity, the device server shall return CHECK CONDITION status and shall set the sense key to ILLEGAL REQUEST with an additional sense code of INVALID FIELD IN CDB.

7.26.5 Download microcode mode (0100b)

If the logical unit is unable to accept this command because of some device condition, the device server shall terminate each WRITE BUFFER command with this mode (0100b) with a CHECK CONDITION status, a sense key of ILLEGAL REQUEST, and shall set the additional sense code to COMMAND SEQUENCE ERROR.

In this mode, vendor specific microcode or control information shall be transferred to the control memory space of the logical unit. After a power-cycle or reset, the device operation shall revert to a vendor specific condition. The meanings of the BUFFER ID, BUFFER OFFSET, and PARAMETER LIST LENGTH fields are not specified by this standard and are not required to be zero-filled. When the microcode download has completed successfully the device server shall generate a unit attention condition for all initiators except the one that issued the WRITE BUFFER command (see SAM-2). The additional sense code shall be set to MICROCODE HAS BEEN CHANGED.

7.26.6 Download microcode and save mode (0101b)

If the logical unit is unable to accept this command because of some device condition, the device server shall terminate each WRITE BUFFER command with this mode (0101b) with a CHECK CONDITION status, a sense key of ILLEGAL REQUEST, and shall set the additional sense code to COMMAND SEQUENCE ERROR.

In this mode, vendor specific microcode or control information shall be transferred to the logical unit and, if the WRITE BUFFER command is completed successfully, also shall be saved in a nonvolatile memory space (semiconductor, disk, or other). The downloaded code shall then be effective after each power-cycle and reset

until it is supplanted in another download microcode and save operation. The meanings of the BUFFER ID, BUFFER OFFSET, and PARAMETER LIST LENGTH fields are not specified by this standard and are not required to be zero-filled. When the download microcode and save command has completed successfully the device server shall generate a unit attention condition (see SAM-2) for all initiators except the one that issued the WRITE BUFFER command. When reporting the unit attention condition, the device server shall set the additional sense code to MICROCODE HAS BEEN CHANGED.

7.26.7 Download microcode with offsets (0110b)

In this mode, the application client may split the transfer of the vendor specific microcode or control information over two or more WRITE BUFFER commands. If the logical unit is unable to accept this command because of some device condition, the device server shall terminate each WRITE BUFFER command with this mode (0110b) with a CHECK CONDITION status, a sense key of ILLEGAL REQUEST, and shall set the additional sense code to COMMAND SEQUENCE ERROR.

If the last WRITE BUFFER command of a set of one or more commands completes successfully, the microcode or control information shall be transferred to the control memory space of the logical unit. After a power-cycle or reset, the device shall revert to a vendor specific condition. In this mode, the Data-Out Buffer contains vendor specific, self-describing microcode or control information.

Since the downloaded microcode or control information may be sent using several commands, when the logical unit detects the last download microcode with offsets and save mode WRITE BUFFER command has been received, the device server shall perform any logical unit required verification of the complete set of downloaded microcode or control information prior to returning GOOD status for the last command. After the last command completes successfully the device server shall generate a unit attention condition (see SAM-2) for all initiators except the one that issued the set of WRITE BUFFER commands. When reporting the unit attention condition, the device server shall set the additional sense code to MICROCODE HAS BEEN CHANGED.

If the complete set of WRITE BUFFER commands required to effect a microcode or control information change (one or more commands) are not received before a reset or power-on cycle occurs, the change shall not be effective and the new microcode or control information shall be discarded.

The BUFFER ID field identifies a specific buffer within the logical unit. The vendor assigns buffer ID codes to buffers within the logical unit. A buffer ID value of zero shall be supported. If more than one buffer is supported, additional buffer ID codes shall be assigned contiguously, beginning with one. If an unsupported buffer ID code is identified, the device server shall return CHECK CONDITION status and shall set the sense key to ILLEGAL REQUEST with an additional sense code of INVALID FIELD IN CDB.

The microcode or control information are written to the logical unit buffer starting at the location specified by the BUFFER OFFSET field. The application client shall send commands that conform to the offset boundary requirements (see 7.13.5). If the device server is unable to accept the specified buffer offset, it shall return CHECK CONDITION status and it shall set the sense key to ILLEGAL REQUEST with an additional sense code of INVALID FIELD IN CDB.

The PARAMETER LIST LENGTH field specifies the maximum number of bytes that shall be present in the Data-Out Buffer to be stored in the specified buffer beginning at the buffer offset. The application client should attempt to ensure that the parameter list length plus the buffer offset does not exceed the capacity of the specified buffer. The capacity of the buffer may be determined by the BUFFER CAPACITY field in the READ BUFFER descriptor. If the BUFFER OFFSET and PARAMETER LIST LENGTH fields specify a transfer in excess of the buffer capacity, the device server shall return CHECK CONDITION status and shall set the sense key to ILLEGAL REQUEST with an additional sense code of INVALID FIELD IN CDB.

7.26.8 Download microcode with offsets and save mode (0111b)

In this mode, the initiator may split the transfer of the vendor specific microcode or control information over two or more WRITE BUFFER commands. If the logical unit is unable to accept this command because of some device condition, the device server shall terminate each mode 0111b WRITE BUFFER command with a CHECK CONDITION status, a sense key of ILLEGAL REQUEST, and shall set the additional sense code to COMMAND SEQUENCE ERROR.

If the last WRITE BUFFER command of a set of one or more commands completes successfully, the microcode or control information shall be saved in a nonvolatile memory space (e.g., semiconductor, disk, or other). The saved downloaded microcode or control information shall then be effective after each power-cycle and reset until it is supplanted by another download microcode with save operation or download microcode with offsets and save operation. In this mode, the Data-Out Buffer contains vendor specific, self-describing microcode or control information.

Since the downloaded microcode or control information may be sent using several commands, when the logical unit detects the last download microcode with offsets and save mode WRITE BUFFER command has been received, the device server shall perform any logical unit required verification of the complete set of downloaded microcode or control information prior to returning GOOD status for the last command. After the last command completes successfully the device server shall generate a unit attention condition (see SAM-2) for all initiators except the one that issued the set of WRITE BUFFER commands. When reporting the unit attention condition, the device server shall set the additional sense code to MICROCODE HAS BEEN CHANGED.

If the complete set of WRITE BUFFER commands required to effect a microcode or control information change (one or more commands) are not received before a reset or power-on cycle occurs, the change shall not be effective and the new microcode or control information shall be discarded.

The BUFFER ID field identifies a specific buffer within the logical unit. The vendor assigns buffer ID codes to buffers within the logical unit. A buffer ID value of zero shall be supported. If more than one buffer is supported, additional buffer ID codes shall be assigned contiguously, beginning with one. If an unsupported buffer ID code is identified, the device server shall return CHECK CONDITION status and shall set the sense key to ILLEGAL REQUEST with an additional sense code of INVALID FIELD IN CDB.

The microcode or control information are written to the logical unit buffer starting at the location specified by the BUFFER OFFSET field. The application client shall conform to the offset boundary requirements. If the device server is unable to accept the specified buffer offset, it shall return CHECK CONDITION status and it shall set the sense key to ILLEGAL REQUEST with an additional sense code of INVALID FIELD IN CDB.

The PARAMETER LIST LENGTH field specifies the maximum number of bytes that shall be present in the Data-Out Buffer to be stored in the specified buffer beginning at the buffer offset. The application client should attempt to ensure that the parameter list length plus the buffer offset does not exceed the capacity of the specified buffer. The capacity of the buffer may be determined by the BUFFER CAPACITY field in the READ BUFFER descriptor. If the BUFFER OFFSET and PARAMETER LIST LENGTH fields specify a transfer in excess of the buffer capacity, the device server shall return CHECK CONDITION status and shall set the sense key to ILLEGAL REQUEST with an additional sense code of INVALID FIELD IN CDB.

7.26.9 Write data to echo buffer (1010b)

In this mode the device server transfers data from the application client and stores it in an echo buffer. An echo buffer is assigned in the same manner by the target as it would for a write operation. Data shall be sent aligned on four-byte boundaries. The BUFFER ID and BUFFER OFFSET fields are ignored in this mode.

NOTE 39 It is recommended that the target assign echo buffers on a per initiator basis to limit the number of exception conditions that may occur in a multi-initiator environment.

Upon successful completion of a WRITE BUFFER command the data shall be preserved in the echo buffer unless there is an intervening command to any logical unit in which case it may be changed.

The PARAMETER LIST LENGTH field specifies the maximum number of bytes that shall be transferred from the Data-Out Buffer to be stored in the echo buffer. The application client should attempt to ensure that the parameter list length does not exceed the capacity of the echo buffer. The capacity of the echo buffer may be determined by the BUFFER CAPACITY field in the READ BUFFER echo buffer descriptor. If the PARAMETER LIST LENGTH field specifies a transfer in excess of the buffer capacity, the device server shall return CHECK CONDITION status and shall set the sense key to ILLEGAL REQUEST with an additional sense code of INVALID FIELD IN CDB.

8 Parameters for all device types

8.1 Diagnostic parameters

8.1.1 Diagnostic page format and page codes for all device types

This subclause describes the diagnostic page structure and the diagnostic pages that are applicable to all SCSI devices. Pages specific to each device type are described in the command standard (see 3.1.12) that applies to that device type.

A SEND DIAGNOSTIC command with a PF bit of one specifies that the SEND DIAGNOSTIC parameter list consists of zero or more diagnostic pages and that the data returned by the subsequent RECEIVE DIAGNOSTIC RESULTS command shall use the diagnostic page format defined in table 120. A RECEIVE DIAGNOSTIC RESULTS command with a PCV bit of one specifies that the device server return a diagnostic page using the format defined in table 120.

Table 120 — Diagnostic page format

Bit / Byte	7	6	5	4	3	2	1	0
0	PAGE CODE							
1	Reserved							
2	(MSB)			PAGE LENGTH (n-3)				
3								(LSB)
4	Diagnostic parameters							
n								

Each diagnostic page defines a function or operation that the device server shall perform as a result of a SEND DIAGNOSTIC command or the information being returned as a result of a RECEIVE DIAGNOSTIC RESULTS with the PCV bit equal to one. The page contains a page header followed by the data that is formatted according to the page code specified.

Device servers that implement diagnostic pages are only required to accept a single diagnostic page per command.

The PAGE CODE field identifies which diagnostic page is being sent as a result of a SEND DIAGNOSTIC command, requested as a result of a RECEIVE DIAGNOSTIC RESULTS command with the PCV bit equal to one, or returned as a result of a RECEIVE DIAGNOSTIC RESULTS parameter data. The page codes are defined in table 121.

Table 121 — Diagnostic page codes

Page Code	Description	Reference
00h	Supported diagnostics pages	8.1.2
01h	Configuration	SES
02h	Enclosure Status/Control	SES
03h	Help Text	SES
04h	String In/Out	SES
05h	Threshold In/Out	SES
06h	Array Status/Control	SES
07h	Element Descriptor	SES
08h	Short Enclosure Status	SES
09h - 0Fh	Reserved for SES	SES
10h - 3Fh	Pages that apply to all device types	
40h - 7Fh	See specific device type for definition	
80h - FFh	Vendor specific pages	

The PAGE LENGTH field specifies the length in bytes of the diagnostic parameters that follow this field. If the application client sends a page length that results in the truncation of any parameter, the device server shall terminate the command with CHECK CONDITION status. The sense key shall be set to ILLEGAL REQUEST with the additional sense code set to INVALID FIELD IN PARAMETER LIST.

The diagnostic parameters are defined for each page code. The diagnostic parameters within a page may be defined differently in a SEND DIAGNOSTIC command than in a RECEIVE DIAGNOSTIC RESULTS command.

8.1.2 Supported diagnostic pages

The supported diagnostics page (see table 122) returns the list of diagnostic pages implemented by the device server. This page shall be implemented if the device server implements the page format option of the SEND DIAGNOSTIC and RECEIVE DIAGNOSTIC RESULTS commands.

Table 122 — Supported diagnostic pages

Bit / Byte	7	6	5	4	3	2	1	0
0	PAGE CODE (00h)							
1	Reserved							
2	(MSB)			PAGE LENGTH (n-3)				
3								(LSB)
4	SUPPORTED PAGE LIST							
n								

The definition of this page for the SEND DIAGNOSTIC command includes only the first four bytes. If the PAGE LENGTH field is not zero, the device server shall terminate the SEND DIAGNOSTIC command with CHECK CONDITION status. The sense key shall be set to ILLEGAL REQUEST with an additional sense code of INVALID FIELD IN PARAMETER LIST. This page instructs the device server to make available the list of all supported diagnostic pages to be returned by a subsequent RECEIVE DIAGNOSTIC RESULTS command.

The definition of this page for the RECEIVE DIAGNOSTIC RESULTS command includes the list of diagnostic pages supported by the device server.

The PAGE LENGTH field specifies the length in bytes of the following supported page list.

The SUPPORTED PAGE LIST field shall contain a list of all diagnostic page codes implemented by the device server in ascending order beginning with page code 00h.

8.2 Log parameters

8.2.1 Log page structure and page codes for all device types

This subclause describes the log page structure and the log pages that are applicable to all SCSI devices. Pages specific to each device type are described in the command standard (see 3.1.12) that applies to that device type. The LOG SELECT command supports the ability to send zero or more log pages. The LOG SENSE command (see 7.5) returns a single log page specified in the PAGE CODE field of the CDB.

Each log page begins with a four-byte page header followed by zero or more variable-length log parameters defined for that page. The log page format is defined in table 123.

Table 123 — Log page format

Bit / Byte	7	6	5	4	3	2	1	0
0	PAGE CODE							
1	Reserved							
2	(MSB)			PAGE LENGTH (n-3)				
3								(LSB)
	Log parameter(s)							
4				Log parameter (First)				
x+3				(Length x)				
				. .				
n-y+1				Log parameter (Last)				
n				(Length y)				

The PAGE CODE field identifies which log page is being transferred.

The PAGE LENGTH field specifies the length in bytes of the following log parameters. If the application client sends a page length that results in the truncation of any parameter, the device server shall terminate the command with CHECK CONDITION status. The sense key shall be set to ILLEGAL REQUEST with the additional sense code set to INVALID FIELD IN PARAMETER LIST.

Most log pages contain one or more special data structures called log parameters (see table 124). Log parameters may be data counters of (a) particular event(s), the conditions under which certain operations were performed, or list parameters that contain a character string description of a particular event.

Table 124 — Log parameter

Bit / Byte	7	6	5	4	3	2	1	0
0	(MSB)			PARAMETER CODE				
1								(LSB)
2	DU	DS	TSD	ETC	TMC		LBIN	LP
3	PARAMETER LENGTH (n-3)							
4				PARAMETER VALUE				
n								

Each log parameter begins with a four-byte parameter header followed by one or more bytes of PARAMETER VALUE data.

The PARAMETER CODE field identifies the log parameter being transferred for that log page.

The DU, DS, TSD, ETC, TMC, LBIN, and LP fields are collectively referred to as the PARAMETER CONTROL byte. These fields are described below in this subclause.

For cumulative log parameter values, indicated by the PC field of the LOG SELECT and LOG SENSE commands, the disable update (DU) bit is defined as follows:

a) A zero value indicates that the device server shall update the log parameter value to reflect all events that should be noted by that parameter; or
b) A one value indicates that the device server shall not update the log parameter value except in response to a LOG SELECT command that specifies a new value for the parameter.

NOTE 40 When updating cumulative log parameter values, a device server may use volatile memory to hold these values until a LOG SELECT or LOG SENSE command is received with an SP bit of one or a target-defined event occurs. As a result the updated cumulative log parameter values may be lost if a power cycle occurs.

The DU bit is not defined for threshold values, indicated by the PC field of the LOG SENSE command, nor for list parameters as indicated by the LP bit. The device server shall ignore the value of the DU bit in any log parameters received with a LOG SELECT command.

A disable save (DS) bit of zero indicates that the target supports saving for that log parameter. The device server shall save the current cumulative or the current threshold parameter value, depending on the value in the PC field of the CDB, in response to a LOG SELECT or LOG SENSE command with an SP bit of one. A DS bit of one indicates that the target does not support saving that log parameter in response to a LOG SELECT or LOG SENSE command with an SP bit of one.

A target save disable (TSD) bit of zero indicates that the target provides a target-defined method for saving log parameters. This implicit saving operation shall be done frequently enough to insure that the cumulative parameter values retain statistical significance (i.e., across power cycles). A TSD bit of one indicates that either the target does not provide a target-defined method for saving log parameters or the target-defined method has been disabled individually by an application client setting the TSD bit to one. An application client may disable the target-defined method for saving all log parameters without changing any TSD bits. See the GLTSD bit in the control mode page (see 8.3.6).

An enable threshold comparison (ETC) bit of one indicates that a comparison to the threshold value is performed whenever the cumulative value is updated. An ETC bit of zero indicates that a comparison is not performed. The value of the ETC bit is the same for cumulative and threshold parameters.

The threshold met criteria (TMC) field (see table 125) defines the basis for comparison of the cumulative and threshold values. The TMC field is valid only if the ETC bit is one. The value of the TMC field is the same for cumulative and threshold parameters.

Table 125 — Threshold met criteria

Code	Basis for comparison		
00b	Every update of the cumulative value		
01b	Cumulative value	equal	threshold value
10b	Cumulative value	not equal	threshold value
11b	Cumulative value	greater than	threshold value

If the ETC bit is one and the result of the comparison is true, a unit attention condition shall be generated for all initiators. When reporting the unit attention condition, the device server shall set the sense key to UNIT ATTENTION and set the additional sense code to THRESHOLD CONDITION MET.

The LBIN bit is only valid if the LP bit is set to one. If the LP bit is one and the LBIN bit is zero then the list parameter is a string of ASCII graphic codes (i.e., code values 20h through 7Eh). If the LP bit is one and the LBIN bit is one then the list parameter is a list of binary information.

The list parameter (LP) bit indicates the format of the log parameter. If an application client attempts to set the value of the LP bit to a value other than the one returned for the same parameter in the LOG SENSE command, the device server shall terminate the command with CHECK CONDITION status. The sense key shall be set to ILLEGAL REQUEST with the additional sense code set to INVALID FIELD IN PARAMETER LIST.

An LP bit of zero indicates that the parameter is a data counter. Data counters are associated with one or more events. The data counter is updated whenever one of these events occurs by incrementing the counter value, if each data counter has associated with it a target-defined maximum value. Upon reaching this maximum value, the data counter shall not be incremented (i.e., it does not wrap). When a data counter reaches its maximum value, the device server shall set the associated DU bit to one. If the data counter is at or reaches its maximum value during the execution of a command, the device server shall complete the command. If the command completes correctly, except for the data counter being at its maximum value, and if the RLEC bit of the control mode page (see 8.3.6) is set to one; then the device server shall terminate the command with CHECK CONDITION status and set the sense key to RECOVERED ERROR with the additional sense code set to LOG COUNTER AT MAXIMUM.

An LP bit of one indicates that the parameter is a list parameter. List parameters are not counters and thus the ETC and TMC fields shall be set to zero.

If more than one list parameter is defined in a single log page, the following rules apply to assigning parameter codes:

a) The parameter updated last shall have a higher parameter code than the previous parameter, except as defined in rule b);
b) When the maximum parameter code value supported by the target is reached, the device server shall assign the lowest parameter code value to the next log parameter (i.e., wrap-around parameter codes). If the associated command completes correctly, except for the parameter code being at its maximum value, and if the RLEC bit of the control mode page (see 8.3.6) is set to one; then the device server shall terminate the command with CHECK CONDITION status and set the sense key to RECOVERED ERROR with the additional sense code set to LOG LIST CODES EXHAUSTED.

NOTE 41 List parameters may be used to store the locations of defective blocks in the following manner. When a defective block is identified, a list parameter is updated to reflect the location and cause of the defect. When the next defect is encountered, the list parameter with the next higher parameter code is updated to record this defect. The size of the page may be made vendor specific to accommodate memory limitations. It is recommended that one or more data counter parameters be defined for the page to keep track of the number of valid list parameters and the parameter code of the parameter with the oldest recorded defect. This technique may be adapted to record other types of information.

The parameter length field specifies the length in bytes of the following parameter value. If the application client sends a parameter length value that results in the truncation of the parameter value, the device server shall terminate the command with CHECK CONDITION status. The sense key shall be set to ILLEGAL REQUEST with the additional sense code set to INVALID FIELD IN PARAMETER LIST.

If the application client sends a log parameter value that is outside the range supported by the target, and rounding is implemented for that parameter, the device server may either:

a) round to an acceptable value and terminate the command as described in 5.3; or
b) terminate the command with CHECK CONDITION status and set the sense key to ILLEGAL REQUEST with the additional sense code set to INVALID FIELD IN PARAMETER LIST.

When any counter in a log page reaches its maximum value, incrementing of all counters in that log page shall cease until reinitialized by the application client via a LOG SELECT command. If the RLEC bit of the control mode page is one, then the device server shall report the exception condition.

The page code assignments for the log pages are listed in table 126.

Table 126 — Log page codes

Page Code	Description	Reference
0Fh	Application client page	
01h	Buffer over-run/under-run page	8.2.3
03h	Error counter page (read) page	8.2.4
04h	Error counter page (read reverse) page	8.2.4
05h	Error counter page (verify) page	8.2.4
02h	Error counter page (write) page	8.2.4
0Bh	Last *n* deferred errors or asynchronous events page	8.2.5
07h	Last *n* error events page	8.2.6
06h	Non-medium error page	8.2.7
10h	Self-test results page	8.2.8
0Eh	Start-stop cycle counter page	8.2.9
00h	Supported log pages	8.2.10
0Dh	Temperature page	8.2.11
08h - 0Ah	Reserved (may be used by specific device types)	
0Ch	Reserved (may be used by specific device types)	
11h - 2Fh	Reserved (may be used by specific device types)	
3Fh	Reserved	
30h - 3Eh	Vendor specific pages	

8.2.2 Application client page

The application client page (see table 127) provides a place for application clients to store information. The page code for the application client page is 0Fh.

Table 127 — Application client page

Bit / Byte	7	6	5	4	3	2	1	0
0	PAGE CODE (0Fh)							
1	Reserved							
2	(MSB)			PAGE LENGTH (n-3)				
3								(LSB)
	Application client log parameters							
4				First application client log parameter				
				⋮				
				Last application client log parameter				
n								

The PAGE CODE and PAGE LENGTH fields are described in 8.2.1.

Parameter codes 0000h through 0FFFFh are for general usage application client data. The intended use for this information is to aid in describing the system configuration and system problems, but the exact definition of the data is application client specific. The general usage application client data parameters all have the format shown in table 128.

Table 128 — General usage application client parameter data

Bit / Byte	7	6	5	4	3	2	1	0
0	(MSB)							
1	PARAMETER CODE							(LSB)
2	DU	DS	TSD	ETC	TMC		LBIN	LP
3	PARAMETER LENGTH (FCh)							
4								
255	GENERAL USAGE PARAMETER BYTES							

For general usage application client data, the value in the PARAMETER CODE field shall be between 0000h and 0FFFh. The first supported general usage application client parameter code shall be 0000h and additional supported parameters shall be sequentially numbered. If any general usage parameter codes are implemented, the device shall support at least 64 general usage parameter descriptors and they shall be parameter codes 0000h through 003Fh.

For the general usage application client parameter, the PARAMETER LENGTH value for each parameter shall be FCh.

The state of the log parameter control bits for parameters 0000h through 0FFFh is specified in table 129.

Table 129 — Parameter control bits for general usage parameters (0000h through 0FFFh)

Bit	Value	Description
DU	1	Value provided by application client
DS	0	Device server supports saving of parameter
TSD	0	Device server manages saving of parameter
ETC	0	No threshold comparison is made on this value
TMC	xx	Ignored when ETC is 0
LBIN	1	The parameter is in binary format
LP	1	The parameter is a list parameter

The values stored in the GENERAL USAGE PARAMETER BYTES represent data sent to the device server in a previous LOG SELECT command. If a previous LOG SELECT command has not occurred, the data is vendor specific.

In the application client page, parameter codes 1000h through FFFFh are reserved.

8.2.3 Buffer over-run/under-run page

The buffer over-run/under-run page (page code 01h) defines 24 data counters that may be used to record the number of buffer over-runs or under-runs for the logical unit. A target that implements this page may implement one or more of the defined data counters.

A buffer over-run or under-run may occur when an initiator does not transmit data to or from the target's buffer fast enough to keep up with reading or writing the media. The cause of this problem is protocol specific. A buffer over-run condition may occur during a read operation when a buffer full condition prevents continued transfer of data from the media to the buffer. A buffer under-run condition may occur during a write operation when a buffer empty condition prevents continued transfer of data to the media from the buffer. Most devices incur a delay at this point while the media is repositioned.

Table 130 defines the PARAMETER CODE field for the buffer over-run/under-run counters.

Table 130 — Parameter code field for buffer over-run/under-run counters

Bit / Byte	7	6	5	4	3	2	1	0
0	Reserved							
1	COUNT BASIS			CAUSE				TYPE

The PARAMETER CODE field for buffer over-run/under-run counters is a 16-bit value comprised of eight reserved bits, a three-bit COUNT BASIS field (see table 131), a four-bit CAUSE field (see table 132), and a one-bit TYPE field. These are concatenated to determine the value of the parameter code for that log parameter. For example, a counter for parameter code value of 0023h specifies a count basis of 001b; a cause of 0001b; and a type of 1b; this counter is incremented once per command that experiences an over-run due to the SCSI bus being busy.

The COUNT BASIS field defines the criteria for incrementing the counter. The criteria are defined in table 131.

Table 131 — Count basis definition

Count basis	Description
000b	Undefined
001b	Per command
010b	Per failed reconnect
011b	Per unit of time
100b - 111b	Reserved

NOTE 42 The per unit of time count basis is device type specific. Direct-access devices typically use a latency period (i.e., one revolution of the medium) as the unit of time.

The CAUSE field indicates the reason that the over-run or under-run occurred. The following causes are defined in table 132.

Table 132 — Cause field definition

Cause	Description
0h	Undefined
1h	Bus busy
2h	Transfer rate too slow
3h - Fh	Reserved

The TYPE field indicates whether the counter records under-runs or over-runs. A value of zero specifies a buffer under-run condition and a value of one specifies a buffer over-run condition.

The counters contain the total number of times buffer over-run or under-run conditions have occurred since the last time the counter was cleared. The counter shall be incremented for each occurrence of an under-run or over-run condition and may be incremented more than once for multiple occurrences during the execution of a single command.

8.2.4 Error counter pages

This subclause defines the optional error counter pages for write errors (page code 02h), read errors (page code 03h), read reverse errors (page code 04h) and verify errors (page code 05h). The log page format is defined in 8.2.1. A page may return one or more log parameters that record events defined by the parameter codes. Table 133 defines the parameter codes for the error counter pages. Support of each log parameter is optional.

Table 133 — Parameter codes for error counter pages

Parameter code	Description
0000h	Errors corrected without substantial delay
0001h	Errors corrected with possible delays
0002h	Total (e.g., rewrites or rereads)
0003h	Total errors corrected
0004h	Total times correction algorithm processed
0005h	Total bytes processed
0006h	Total uncorrected errors
0007h - 7FFFh	Reserved
8000h - FFFFh	Vendor specific

NOTE 43 The exact definition of the error counters is not part of this standard. These counters should not be used to compare products because the products may define errors differently.

8.2.5 Last *n* deferred errors or asynchronous events page

The last *n* deferred errors or asynchronous events page (page code 0Bh) provides for a number of deferred errors or asynchronous events sense data records using the list parameter format of the log page. The number of these deferred errors or asynchronous events records supported, *n*, is vendor specific. Each deferred error or asynchronous event record contains SCSI sense data for a deferred error or asynchronous event that has occurred. The parameter code associated with the record indicates the relative time at which the deferred error or asynchronous event occurred. A higher parameter code indicates that the deferred error or asynchronous event occurred later in time.

The content of the parameter value field of each log parameter is the SCSI sense data describing the deferred error.

The fields DU, TSD, ETC, and TMC are reserved and shall be set to zero. The LBIN bit shall be set to one to indicate binary information. The LP bit shall be set to one to indicate a list parameter.

8.2.6 Last *n* error events page

The last *n* error events page (page code 07h) provides for a number of error-event records using the list parameter format of the log page. The number of these error-event records supported, *n*, is vendor specific. Each error-event record contains vendor specific diagnostic information for a single error encountered by the device. The parameter code associated with error-event record indicates the relative time at which the error occurred. A higher parameter code indicates that the error event occurred later in time.

The content of the parameter value field of each log parameter is an ASCII character string that may describe the error event. The exact contents of the character string is not defined by this standard.

When the last supported parameter code is used by an error-event record, the recording on this page of all subsequent error information shall cease until one or more of the list parameters with the highest parameter codes have been reinitialized. If the RLEC bit of the control mode page (see 8.3.6) is set to one, the device server shall return CHECK CONDITION status with the sense key set to RECOVERED ERROR and the additional sense code set to LOG LIST CODES EXHAUSTED. Alternatively, the device server may report this condition via asynchronous event notification (see SAM-2).

8.2.7 Non-medium error page

The non-medium error page (page code 06h) provides for summing the occurrences of recoverable error events other than write, read, or verify failures. No discrimination among the various types of events is provided by parameter code (see table 134). Vendor specific discrimination may be provided through the vendor specific parameter codes.

Table 134 — Non-medium error event parameter codes

Parameter code	Description
0000h	Non-medium error count
0001h - 7FFFh	Reserved
8000h - FFFFh	Vendor specific error counts

8.2.8 Self-test results page

The self-test results log page (see table 135) provides the results from the twenty most recent self-tests (see 5.4). Results from the most recent self-test or the self-test currently in progress shall be reported in the first self-test log parameter; results from the second most recent self-test shall be reported in the second self-test log parameter; etc. If fewer than twenty self-tests have occurred, the unused self-test log parameter entries shall be zero filled.

Table 135 — Self-test results page

Bit / Byte	7	6	5	4	3	2	1	0
0	PAGE CODE (10h)							
1	Reserved							
2	(MSB)			PAGE LENGTH (190h)				
3								(LSB)
Self-test results log parameters								
4	First self-test results log parameter							
23								
	. . .							
384	Twentieth self-test results log parameter							
403								

The PAGE CODE and PAGE LENGTH fields are described in 8.2.1.

Table 136 shows the format of one self-test log parameter.

Table 136 — Self-test results log parameter format

Bit / Byte	7	6	5	4	3	2	1	0
0	(MSB)			PARAMETER CODE (0001h to 0014h)				
1								(LSB)
2	DU	DS	TSD	ETC	TMC		LBIN	LP
3	PARAMETER LENGTH (10h)							
4	SELF-TEST CODE			Reserved	SELF-TEST RESULTS			
5	SELF-TEST NUMBER							
6	(MSB)			TIMESTAMP				
7								(LSB)
8	(MSB)			ADDRESS OF FIRST FAILURE				
15								(LSB)
16	reserved				SENSE KEY			
17	ADDITIONAL SENSE CODE							
18	ADDITIONAL SENSE CODE QUALIFIER							
19	Vendor specific							

The PARAMETER CODE field identifies the log parameter being transferred. The PARAMETER CODE field for the results of the most recent self-test shall contain 0001h; the PARAMETER CODE field for the results of the second most recent test shall contain 0002h; etc.

The values of the log parameter control bits for self-test results log parameters is specified in table 137.

Table 137 — Parameter control bits for self-test results log parameters

Bit	Value	Description
DU	0	Value provided by device server
DS	0	Device server supports saving of parameter
TSD	0	Device server manages saving of parameter
ETC	0	No threshold comparison is made on this value
TMC	xx	Ignored when ETC is 0
LBIN	1	The parameter is in binary format
LP	1	The parameter is a list parameter

The PARAMETER LENGTH field shall contain 10h.

The SELF-TEST CODE field contains the value in the SELF-TEST CODE field of the SEND DIAGNOSTICS command that initiated this self-test (see 7.23).

Table 138 defines the content of the SELF-TEST RESULTS field.

Table 138 — Self-test results values

Value	Description
0h	The self-test completed without error.
1h	The background self-test was aborted by the application client using a SEND DIAGNOSTICS command (see 7.23) with the SELF-TEST CODE field set to 100b (Abort background self-test).
2h	The self-test routine was aborted by an application client using a method other than a SEND DIAGNOSTICS command with the SELF-TEST CODE field set to 100b (e.g., by a task management function, by a reset, or by issuing an exception command as defined in 5.4.3).
3h	An unknown error occurred while the device server was executing the self-test and the device server was unable to complete the self-test.
4h	The self-test completed with a failure in a test segment, and the test segment that failed is not known.
5h	The first segment of the self-test failed.
6h	The second segment of the self-test failed.
7h	Another segment of the self-test failed (see the SELF-TEST SEGMENT NUMBER field).
8h-Eh	Reserved
Fh	The self-test is in progress.

The SELF-TEST NUMBER field identifies the self-test that failed and consists of either:

a) the number of the segment that failed during the self-test; or
b) the number of the test that failed and the number of the segment in which the test was run, using a vendor specific method for placing the two values in the one field.

When the segment in which the failure occurred cannot or need not be identified, the SELF-TEST NUMBER field shall contain 00h.

The TIMESTAMP field contains the total accumulated power-on hours for the device server at the time the self-test was completed. If the test is still in progress, the content of the TIMESTAMP field shall be zero. If the power-on hours for the device server at the time the self-test was completed is greater than FFFFh then the content of the TIMESTAMP field shall be FFFFh.

The ADDRESS OF FIRST FAILURE field contains information that locates the failure on the media. If the logical unit implements logical blocks, the content of the ADDRESS OF FIRST FAILURE field is the first logical block address where a self-test error occurred. This implies nothing about the quality of any other logical block on the logical unit, since the testing during which the error occurred may not have been performed in a sequential manner. This value shall not change (e.g., as the result of block reassignment). The content of the ADDRESS OF FIRST FAILURE field shall be FFFFFFFFFFFFFFFFh if no errors occurred during the self-test or if the error that occurred is not related to an identifiable media address.

The SENSE KEY, ADDITIONAL SENSE CODE, and ADDITIONAL SENSE CODE QUALIFIER fields may contain a hierarchy of additional information relating to error or exception conditions that occurred during the self-test represented in the same format used by the sense data (see 7.20).

8.2.9 Start-stop cycle counter page

This subclause defines the optional start-stop cycle counter page (page code 0Eh). A device server that implements the start-stop cycle counter page shall implement one or more of the defined parameters. Table 139 shows the start-stop cycle counter page with all parameters present.

Table 139 — Start-stop cycle counter page

Bit / Byte	7	6	5	4	3	2	1	0
0	PAGE CODE (0Eh)							
1	Reserved							
2	(MSB)			PAGE LENGTH (24h)				
3								(LSB)
4	(MSB)			PARAMETER CODE 0001h Date of Manufacture				
5								(LSB)
6	DU	DS	TSD	ETC	TMC		LBIN	LP
7	PARAMETER LENGTH (06h)							
8	(MSB)			YEAR OF MANUFACTURE (4 ASCII characters)				
11								(LSB)
12	(MSB)			WEEK OF MANUFACTURE (2 ASCII characters)				
13								(LSB)
14	(MSB)			PARAMETER CODE 0002h Accounting Date				
15								(LSB)
16	DU	DS	TSD	ETC	TMC		LBIN	LP
17	PARAMETER LENGTH (06h)							
18	(MSB)			ACCOUNTING DATE YEAR (4 ASCII characters)				
21								(LSB)
22	(MSB)			ACCOUNTING DATE WEEK (2 ASCII characters)				
23								(LSB)
24	(MSB)			PARAMETER CODE 0003h Specified cycle count over device lifetime				
25								(LSB)
26	DU	DS	TSD	ETC	TMC		LBIN	LP
27	PARAMETER LENGTH (04h)							
28	(MSB)			SPECIFIED CYCLE COUNT OVER DEVICE LIFETIME (4-byte binary number)				
31								(LSB)
32	(MSB)			PARAMETER CODE 0004h Accumulated start-stop cycles				
33								(LSB)
34	DU	DS	TSD	ETC	TMC		LBIN	LP
35	PARAMETER LENGTH (04h)							
36	(MSB)			ACCUMULATED START-STOP CYCLES (4-byte binary number)				
39								(LSB)

The year and week in the year that the device was manufactured shall be set in the parameter field defined by parameter code 0001h. The date of manufacture shall not be saveable by the application client using the LOG SELECT command (i.e., the log parameter DS bit shall be one). The date is expressed in numeric ASCII

characters (30h – 39h) in the form YYYYWW, as shown in table 139. The state of the parameter control bits for parameter 0001h is specified in table 140.

Table 140 — Parameter control bits for date of manufacture parameter (0001h)

Bit	Value	Description
DU	0	Value provided by device server
DS	1	Device server does not support saving of parameter
TSD	0	Device server manages saving of parameter
ETC	0	No threshold comparison is made on this value
TMC	xx	Ignored when ETC is 0
LBIN	0	The parameter is in ASCII format
LP	1	The parameter is a list parameter

The accounting date specified by parameter code 0002h may be saved using a LOG SELECT command to indicate when the device was placed in service. If the parameter is not yet set or is not settable, the default value placed in the parameter field shall be 6 ASCII blank characters (20h). The field shall not be checked for validity by the device server. The state of the parameter control bits for parameter 0002h is specified in table 141.

Table 141 — Parameter control bits for accounting date parameter (0002h)

Bit	Value	Description
DU	0	Value provided by device server
DS	0 or 1	Device server optionally supports saving of parameter
TSD	0	Device server manages saving of parameter
ETC	0	No threshold comparison is made on this value
TMC	xx	Ignored when ETC is 0
LBIN	0	The parameter is in ASCII format
LP	1	The parameter is a list parameter

The specified cycle count over device lifetime (parameter code 0003h) is a parameter provided by the device server. The specified cycle count over device lifetime parameter shall not be saveable by the application client using the LOG SELECT command (i.e., the log parameter DS bit shall be one). The parameter value is a 4-byte binary number. The value indicates how many stop-start cycles may typically be executed over the lifetime of the device without degrading the device's operation or reliability outside the limits specified by the manufacturer of the device. The state of the parameter control bits for parameter 0003h is specified in table 142.

Table 142 — Parameter control bits for start-stop cycle counter parameters (0003h and 0004h)

Bit	Value	Description
DU	0	Value provided by device server
DS	1	Device server does not support saving of parameter
TSD	0	Device server manages saving of parameter
ETC	0	No threshold comparison is made on this value
TMC	xx	Ignored when ETC is 0
LBIN	1	The parameter is in binary format
LP	1	The parameter is a list parameter

The accumulated start-stop cycles (parameter code 0004h) is a parameter provided by the device server. The accumulated start-stop cycles parameter shall not be saveable by the application client using the LOG SELECT command (i.e., the log parameter DS bit shall be one). The parameter value is a 4-byte binary number. The value indicates how many start-stop cycles the device has detected since its date of manufacture. The time at which the count is incremented during a start-stop cycle is vendor specific. For rotating magnetic storage devices, a single start-stop cycle is defined as an operational cycle that begins with the disk spindle at rest, continues while the disk accelerates to its normal operational rotational rate, continues during the entire period the disk is rotating,

continues as the disk decelerates toward a resting state, and ends when the disk is no longer rotating. For devices without a spindle or with multiple spindles, the definition of a single start-stop cycle is vendor specific. The count is incremented by one for each complete start-stop cycle. No comparison with the value of parameter 0003h shall be performed by the device server. The state of the control bits for parameter 0004h is specified in table 142.

8.2.10 Supported log pages

The supported log page (see table 143) returns the list of log pages implemented by the target. Targets that implement the LOG SENSE command shall implement this log page.

Table 143 — Supported log pages

Bit / Byte	7	6	5	4	3	2	1	0
0	PAGE CODE (00h)							
1	Reserved							
2	(MSB)			PAGE LENGTH (n-3)				
3								(LSB)
4	SUPPORTED PAGE LIST							
n								

This page is not defined for the LOG SELECT command. This log page returns the list of supported log pages for the specified logical unit.

The PAGE LENGTH field specifies the length in bytes of the following supported page list.

The SUPPORTED PAGE LIST field shall contain a list of all log page codes implemented by the target in ascending order beginning with page code 00h.

8.2.11 Temperature page

This subclause defines the optional temperature log page (page code 0Dh). A device server that implements the temperature page shall implement parameter 0000h. Parameter 0001h is optional and may be either omitted or set to a value indicating that the parameter is not defined. Table 144 shows the temperature page with all parameters present.

Table 144 — Temperature page (part 1 of 2)

Bit / Byte	7	6	5	4	3	2	1	0
0	PAGE CODE (0Dh)							
1	Reserved							
2	(MSB)			PAGE LENGTH (0Ch)				
3								(LSB)
4	(MSB)			PARAMETER CODE 0000h				
5				Temperature				(LSB)
6	DU	DS	TSD	ETC	TMC		LBIN	LP
7	PARAMETER LENGTH (02h)							
8	Reserved							
9	TEMPERATURE (degrees Celsius)							

Table 144 — Temperature page (part 2 of 2)

Bit / Byte	7	6	5	4	3	2	1	0
10	(MSB)			PARAMETER CODE 0001h				
11				Reference temperature				(LSB)
12	DU	DS	TSD	ETC	TMC		LBIN	LP
13	PARAMETER LENGTH (02h)							
14	Reserved							
15	REFERENCE TEMPERATURE (degrees Celsius)							

The temperature sensed in the device at the time the LOG SENSE command is performed shall be returned in the parameter field defined by parameter code 0000h. The one byte binary value specifies the temperature of the device in degrees Celsius. Temperatures equal to or less than zero degrees Celsius shall be indicated by a value of zero. If the device server is unable to detect a valid temperature because of a sensor failure or other condition, the value returned shall be FFh. The temperature should be reported with an accuracy of plus or minus three Celsius degrees while the device is operating at a steady state within the environmental limits specified for the device. No comparison is performed between the temperature value specified in parameter 0000h and the reference temperature specified in parameter 0001h. The state of the parameter control bits for parameter 0000h is specified in table 145.

Table 145 — Parameter control bits for temperature parameters (0000h and 0001h)

Bit	Value	Description
DU	0	Value provided by device server
DS	1	Device server does not support saving of parameter
TSD	0	Device server manages saving of parameter
ETC	0	No threshold comparison is made on this value
TMC	xx	Ignored when ETC is 0
LBIN	1	The parameter is in binary format
LP	1	The parameter is a list parameter

A reference temperature for the device may optionally be provided by the device using parameter code 0001h. If no reference temperature is provided, the parameter may not be provided in the log page or alternatively, the reference temperature value may be set to the value of FFh. The one byte binary value should reflect the maximum reported sensor temperature in degrees Celsius at which the device is capable of operating continuously without degrading the device's operation or reliability beyond manufacturer accepted limits. The reference temperature may change for vendor specific reasons. The state of the parameter control bits for parameter 0001h is specified in table 145.

8.3 Mode parameters

8.3.1 Mode parameters overview

This subclause describes the block descriptors and the pages used with MODE SELECT and MODE SENSE commands that are applicable to all SCSI devices. Pages specific to each device type are described in the command standard (see 3.1.12) that applies to that device type.

8.3.2 Mode parameter list format

The mode parameter list shown in table 146 contains a header, followed by zero or more block descriptors, followed by zero or more variable-length pages. Parameter lists are defined for each device type.

Table 146 — Mode parameter list

Bit Byte	7	6	5	4	3	2	1	0
0 - n	Mode parameter header							
0 - n	Block descriptor(s)							
0 - n	Page(s)							

8.3.3 Mode parameter header formats

The six-byte CDB mode parameter header is defined in table 147.

Table 147 — Mode parameter header(6)

Bit Byte	7	6	5	4	3	2	1	0
0	MODE DATA LENGTH							
1	MEDIUM TYPE							
2	DEVICE-SPECIFIC PARAMETER							
3	BLOCK DESCRIPTOR LENGTH							

The ten-byte CDB mode parameter header is defined in table 148.

Table 148 — Mode parameter header(10)

Bit Byte	7	6	5	4	3	2	1	0
0	(MSB)	MODE DATA LENGTH						
1								(LSB)
2	MEDIUM TYPE							
3	DEVICE-SPECIFIC PARAMETER							
4	Reserved							LONGLBA
5	Reserved							
6	(MSB)	BLOCK DESCRIPTOR LENGTH						
7								(LSB)

When using the MODE SENSE command, the MODE DATA LENGTH field specifies the length in bytes of the following data that is available to be transferred. The mode data length does not include the number of bytes in the MODE DATA LENGTH field. When using the MODE SELECT command, this field is reserved.

NOTE 44 Targets that support more than 256 bytes of block descriptors and pages may need to implement ten-byte mode commands. The mode data length field in the six-byte CDB header limits the returned data to 256 bytes.

The contents of the MEDIUM TYPE field are unique for each device type. Refer to the mode parameters subclause of the specific device type command standard (see 3.1.12) for definition of these values. Some device types reserve this field.

The DEVICE-SPECIFIC PARAMETER field is unique for each device type. Refer to the mode parameters subclause of the specific device type command standard (see 3.1.12) for definition of this field. Some device types reserve all or part of this field.

The Long LBA (LONGLBA) bit of zero indicates the mode parameter block descriptors are eight bytes long and have the format described in 8.3.4.1 or 8.3.4.2. A LONGLBA bit of one indicates the mode parameter block descriptors are sixteen bytes long and have the format described in 8.3.4.3.

The BLOCK DESCRIPTOR LENGTH field specifies the length in bytes of all the block descriptors. It is equal to the number of block descriptors times eight if the LONGLBA bit is set to zero or times sixteen if the LONGLBA bit is set to one, and does not include pages or vendor specific parameters, if any, that may follow the last block descriptor. A block descriptor length of zero indicates that no block descriptors are included in the mode parameter list. This condition shall not be considered an error.

8.3.4 Mode parameter block descriptor formats

8.3.4.1 General block descriptor format

When the LONGLBA bit is set to zero (see 8.3.3), the mode parameter block descriptor format for all device types except direct-access is shown in table 149.

Table 149 — General mode parameter block descriptor

Bit / Byte	7	6	5	4	3	2	1	0
0	DENSITY CODE							
1	(MSB)							
2				NUMBER OF BLOCKS				
3								(LSB)
4	Reserved							
5	(MSB)							
6				BLOCK LENGTH				
7								(LSB)

Block descriptors specify some of the medium characteristics for all or part of a logical unit. Support for block descriptors is optional. Each block descriptor contains a DENSITY CODE field, a NUMBER OF BLOCKS field, and a BLOCK LENGTH field. Block descriptor values are always current (i.e., saving is not supported). A unit attention condition (see 7.6 and SAM-2) shall be generated when any block descriptor values are changed.

The DENSITY CODE field is unique for each device type. Refer to the mode parameters subclause of the specific device type command standard (see 3.1.12) for definition of this field. Some device types reserve all or part of this field.

The NUMBER OF BLOCKS field specifies the number of logical blocks on the medium to which the DENSITY CODE and BLOCK LENGTH FIELDS apply. A value of zero indicates that all of the remaining logical blocks of the logical unit shall have the medium characteristics specified.

NOTES
45 There may be implicit association between parameters defined in the pages and block descriptors. In this case, the target may change parameters not explicitly sent with the MODE SELECT command. A subsequent MODE SENSE command may be used to detect these changes.
46 The number of remaining logical blocks may be unknown for some device types.

The BLOCK LENGTH field specifies the length in bytes of each logical block described by the block descriptor. For sequential-access devices, a block length of zero indicates that the logical block size written to the medium is specified by the transfer length field in the CDB (see SSC).

8.3.4.2 Direct-access device block descriptor format for LONGLBA=0

When the LONGLBA bit is set to zero (see 8.3.3), the mode parameter block descriptor format for the direct-access device type is shown in table 150.

Table 150 — Direct-access device mode parameter block descriptor

Bit / Byte	7	6	5	4	3	2	1	0
0	(MSB)							
1				NUMBER OF BLOCKS				
2								
3								(LSB)
4	DENSITY CODE							
5	(MSB)							
6				BLOCK LENGTH				
7								(LSB)

This block descriptor format shall apply only to direct-access devices. When the LONGLBA bit is set to zero (see 8.3.3), all other device types shall use the block descriptor format described in 8.3.4.1.

Block descriptors specify some of the medium characteristics for a logical unit. Support for block descriptors is optional. Each block descriptor contains a DENSITY CODE field, a NUMBER OF BLOCKS field, and a BLOCK LENGTH field. A unit attention condition (see 7.6 and SAM-2) shall be generated when any block descriptor values are changed.

The NUMBER OF BLOCKS field specifies the number of logical blocks on the medium to which the DENSITY CODE and BLOCK LENGTH fields apply. A value of zero indicates that all of the remaining logical blocks of the logical unit shall have the medium characteristics specified.

If the SCSI device doesn't support changing its capacity by changing the NUMBER OF BLOCKS field using the MODE SELECT command, the value in the NUMBER OF BLOCKS field is ignored. If the device supports changing its capacity by changing the NUMBER OF BLOCKS field, then the NUMBER OF BLOCKS field is interpreted as follows:

a) If the number of blocks is set to zero, the device shall retain its current capacity if the block size has not changed. If the number of blocks is set to zero and the block size has changed, the device shall be set to its maximum capacity when the new block size takes effect;
b) If the number of blocks is greater than zero and less than or equal to its maximum capacity, the device shall be set to that number of blocks. If the block size has not changed, the device shall not become format corrupted. This capacity setting shall be retained through reset events or power cycles;
c) If the number of blocks field is set to a value greater than the maximum capacity of the device and less than FFFFFFFFh, then the command is terminated with a CHECK CONDITION status. The sense key is set to ILLEGAL REQUEST. The device shall retain its previous block descriptor settings;
d) If the number of blocks is set to FFFFFFFFh, the device shall be set to its maximum capacity. If the block size has not changed, the device shall not become format corrupted. This capacity setting shall be retained through reset events or power cycles.

NOTE 47 There may be implicit association between parameters defined in the pages and block descriptor. For direct-access devices, the block length affects the optimum values (i.e., the values that achieves the best performance) for the sectors per track, bytes per physical sector, track skew factor, and cylinder skew factor fields in the format parameters page. In this case, the target may change parameters not explicitly sent with the MODE SELECT command. A subsequent MODE SENSE command may be used to detect these changes.

The DENSITY CODE field is unique for each device type. Refer to the mode parameters subclause of the specific device type command standard (see 3.1.12) for the definition of this field. Some device types reserve all or part of this field.

The BLOCK LENGTH field specifies the length in bytes of each logical block described by the block descriptor.

8.3.4.3 Long LBA block descriptor format

When the LONGLBA bit is set to one (see 8.3.3), the mode parameter block descriptor format for all device types is shown in table 151.

Table 151 — Long LBA mode parameter block descriptor

Bit / Byte	7	6	5	4	3	2	1	0
0	(MSB)			NUMBER OF BLOCKS				
7								(LSB)
8	DENSITY CODE							
9	Reserved							
10	Reserved							
11	Reserved							
12	(MSB)			BLOCK LENGTH				
15								(LSB)

Block descriptors specify some of the medium characteristics for all or part of a logical unit. Support for block descriptors is optional. Each block descriptor contains a DENSITY CODE field, a NUMBER OF BLOCKS field, and a BLOCK LENGTH field. Block descriptor values are always current (i.e., saving is not supported). A unit attention condition (see 7.6 and SAM-2) shall be generated when any block descriptor values are changed.

The NUMBER OF BLOCKS field specifies the number of logical blocks on the medium to which the DENSITY CODE and BLOCK LENGTH fields apply. A value of zero indicates that all of the remaining logical blocks of the logical unit shall have the medium characteristics specified.

If the SCSI device doesn't support changing its capacity by changing the NUMBER OF BLOCKS field using the MODE SELECT command, the value in the NUMBER OF BLOCKS field is ignored. If the device supports changing its capacity by changing the NUMBER OF BLOCKS field, then the NUMBER OF BLOCKS field is interpreted as follows:

a) If the number of blocks is set to zero, the device shall retain its current capacity if the block size has not changed. If the number of blocks is set to zero and the block size has changed, the device shall be set to its maximum capacity when the new block size takes effect;
b) If the number of blocks is greater than zero and less than or equal to its maximum capacity, the device shall be set to that number of blocks. If the block size has not changed, the device shall not become format corrupted. This capacity setting shall be retained through reset events or power cycles;
c) If the number of blocks field is set to a value greater than the maximum capacity of the device and less than FFFFFFFFFFFFFFFFh, then the command is terminated with a CHECK CONDITION status. The sense key is set to ILLEGAL REQUEST. The device shall retain its previous block descriptor settings;
d) If the number of blocks is set to FFFFFFFFFFFFFFFFh, the device shall be set to its maximum capacity. If the block size has not changed, the device shall not become format corrupted. This capacity setting shall be retained through reset events or power cycles.

NOTE 48 There may be implicit association between parameters defined in the pages and block descriptor. For direct-access devices, the block length affects the optimum values (i.e., the values that achieve the best performance) for the sectors per track, bytes per physical sector, track skew factor, and cylinder skew factor fields in the format parameters page. In this case, the target may change parameters not explicitly sent with the MODE SELECT command. A subsequent MODE SENSE command may be used to detect these changes.

The DENSITY CODE field is unique for each device type. Refer to the mode parameters subclause of the specific device type command standard (see 3.1.12) for the definition of this field. Some device types reserve all or part of this field.

The BLOCK LENGTH field specifies the length in bytes of each logical block described by the block descriptor.

8.3.5 Mode page format and page codes

The mode page format is defined in table 152.

Table 152 — Mode page format

Bit / Byte	7	6	5	4	3	2	1	0
0	PS	Reserved	PAGE CODE					
1	PAGE LENGTH (n-1)							
2	Mode parameters							
n								

Each mode page contains a PAGE CODE field, a PAGE LENGTH field, and a set of mode parameters. The page codes are defined in this subclause and in the mode parameter subclauses in the command standard (see 3.1.12) for the specific device type.

When using the MODE SENSE command, a parameters savable (PS) bit of one indicates that the mode page may be saved by the target in a nonvolatile, vendor specific location. A PS bit of zero indicates that the supported parameters cannot be saved. When using the MODE SELECT command, the PS bit is reserved.

The PAGE CODE field identifies the format and parameters defined for that mode page. Some page codes are defined as applying to all device types and other page codes are defined for the specific device type. The page codes that apply to a specific device type are defined in the command standard (see 3.1.12) for that device type.

When using the MODE SENSE command, if page code 00h (vendor specific page) is implemented, the device server shall return that page last in response to a request to return all pages (page code 3Fh). When using the MODE SELECT command, this page should be sent last.

The PAGE LENGTH field specifies the length in bytes of the mode parameters that follow. If the application client does not set this value to the value that is returned for the page by the MODE SENSE command, the device server shall terminate the command with CHECK CONDITION status. The sense key shall be set to ILLEGAL REQUEST with the additional sense code set to INVALID FIELD IN PARAMETER LIST. The target is permitted to implement a mode page that is less than the full page length defined in this standard, provided no field is truncated and the PAGE LENGTH field correctly specifies the actual length implemented.

The mode parameters for each page are defined in the following subclauses, or in the mode parameters subclause in the command standard (see 3.1.12) for the specific device type. Mode parameters not implemented by the target shall be set to zero.

Table 153 defines the mode pages that are applicable to all device types that implement the MODE SELECT and MODE SENSE commands.

Table 153 — Mode page codes

Page code	Description	Reference
0Ah	Control mode page	8.3.6
02h	Disconnect-reconnect page	8.3.7
1Ch	Informational exceptions control page	8.3.8
09h	obsolete	3.3.7
1Ah	Power condition page	8.3.9
18h	Protocol specific LUN page	8.3.10
19h	Protocol specific port page	8.3.11
01h	(See specific device type)	
03h - 08h	(See specific device type)	
0Bh - 17h	(See specific device type)	
1Bh	(See specific device type)	
1Dh - 1Fh	(See specific device type)	
00h	Vendor specific (does not require page format)	
20h - 3Eh	(See specific device type)	
3Fh	Return all pages (valid only for the MODE SENSE command)	

8.3.6 Control mode page

The control mode page (see table 154) provides controls over several SCSI features that are applicable to all device types such as tagged queuing, asynchronous event reporting, and error logging.

Table 154 — Control mode page

Bit / Byte	7	6	5	4	3	2	1	0
0	PS	Reserved	PAGE CODE (0Ah)					
1	PAGE LENGTH (0Ah)							
2	TST			Reserved			GLTSD	RLEC
3	QUEUE ALGORITHM MODIFIER				Reserved	QERR		DQUE
4	TAS	RAC	Reserved		SWP	RAERP	UAAERP	EAERP
5	Reserved					AUTOLOAD MODE		
6	(MSB)	READY AER HOLDOFF PERIOD						
7								(LSB)
8	(MSB)	BUSY TIMEOUT PERIOD						
9								(LSB)
10	(MSB)	EXTENDED SELF-TEST COMPLETION TIME						
11								(LSB)

A task set type field (TST) specifies the type of task set (see table 155). If the device maintains mode pages per initiator, the TST field, if changeable, shall reflect in all initiator pages the state selected by the most recent MODE SELECT. If the most recent MODE SELECT changes the setting of this field the device server shall establish a unit

attention condition for all initiators except the one that issued the MODE SELECT command (see SAM-2). The device server shall set the additional sense code to MODE PARAMETERS CHANGED.

Table 155 — Task set type

Value	Description
000b	Task set per logical unit for all initiators
001b	Task set per initiator per logical unit
010b - 111b	Reserved

A global logging target save disable (GLTSD) bit of zero allows the target to provide a target-defined method for saving log parameters. A GLTSD bit of one indicates that either the target has disabled the target-defined method for saving log parameters or when set by the initiator specifies that the target-defined method shall be disabled.

A report log exception condition (RLEC) bit of one specifies that the device server shall report log exception conditions as described in 8.2. A RLEC bit of zero specifies that the device server shall not report log exception conditions.

The QUEUE ALGORITHM MODIFIER field (see table 156) specifies restrictions on the algorithm used for reordering tasks having the SIMPLE task attribute.

Table 156 — Queue algorithm modifier

Value	Description
0h	Restricted reordering
1h	Unrestricted reordering allowed
2h - 7h	Reserved
8h - Fh	Vendor specific

A value of zero in the QUEUE ALGORITHM MODIFIER field specifies that the device server shall order the processing sequence of tasks having the SIMPLE task attribute such that data integrity is maintained for that initiator. This means that, if the transmission of new service delivery requests is halted at any time, the final value of all data observable on the medium shall have exactly the same value as it would have if all the tasks had been given the ORDERED task attribute. The restricted reordering value shall be the default value.

A value of one in the QUEUE ALGORITHM MODIFIER field specifies that the device server may reorder the processing sequence of tasks having the SIMPLE task attribute in any manner. Any data integrity exposures related to task sequence order shall be explicitly handled by the application client through the selection of appropriate commands and task attributes.

The queue error management (QERR) field specifies how the device server shall handle blocked tasks when another task receives a CHECK CONDITION status (see table 157). The task set type (see the TST field definition above) defines which tasks are blocked. If TST field equals 000b, then all tasks from all initiators are blocked. If TST field equals 001b, then only tasks from the initiator that receives the CHECK CONDITION status are blocked.

Table 157 — Queue error management (QERR) field

Value	Definition
00b	Blocked tasks in the task set shall resume after an ACA or CA condition is cleared (see SAM-2).
01b	All the blocked tasks in the task set shall be aborted when the CHECK CONDITION status is sent. If the TAS bit is zero, a unit attention condition (see SAM-2) shall be generated for each initiator that had blocked tasks aborted except for the initiator to which the CHECK CONDITION status was sent. The device server shall set the additional sense code to COMMANDS CLEARED BY ANOTHER INITIATOR. If the TAS bit is one, all tasks blocked for initiators other than the initiator for which the CHECK CONDITION status was sent shall be completed with a TASK ABORTED status and no unit attention shall be generated.
10b	Reserved
11b	Blocked tasks in the task set belonging to the initiator to which a CHECK CONDITION status is sent shall be aborted when the status is sent.

A disable queuing (DQUE) bit of zero specifies that tagged queuing shall be enabled if the device server supports tagged queuing. A DQUE bit of one specifies that tagged queuing shall be disabled. Any queued commands received by the device server shall be aborted. The method used to abort queued commands is protocol specific.

The report a check (RAC) bit provides control of reporting long busy conditions or CHECK CONDITION status. A RAC bit of one specifies that a CHECK CONDITION status should be reported rather than a long busy condition (e.g., longer than the BUSY TIMEOUT PERIOD). A RAC bit of zero specifies that long busy conditions (e.g., busy condition during auto contingent allegiance) may be reported.

A task aborted status (TAS) bit of zero specifies that aborted tasks shall be terminated by the device server without any response to the initiator. A TAS bit of one specifies that tasks aborted by the actions of another initiator shall be terminated with a TASK ABORTED status (see SAM-2).

A software write protect (SWP) bit of one specifies that the logical unit shall inhibit writing to the medium after writing all cached or buffered write data, if any. When SWP is one, all commands requiring writes to the medium shall return CHECK CONDITION status and shall set the sense key to DATA PROTECT and the additional sense code to WRITE PROTECTED. When SWP is one and the device model defines a write protect (WP) bit in the DEVICE-SPECIFIC PARAMETER field in the mode parameter header, the WP bit shall be set to one for subsequent MODE SENSE commands. A SWP bit of zero specifies that the logical unit may allow writing to the medium, depending on other write inhibit mechanisms implemented by the logical unit. When the SWP bit is zero, the value of the WP bit, if defined, is device model specific. For a list of commands affected by the SWP bit and details of the WP bit see the command standard (see 3.1.12) for the specific device type.

The RAERP, UAAERP, and EAERP bits enable specific events to be reported via the asynchronous event reporting protocol. When all three bits are zero, the target shall not use asynchronous event reporting. AER is defined in SAM-2.

A ready AER permission (RAERP) bit of one specifies that the device server may issue an asynchronous event report upon completing its initialization sequence instead of generating a unit attention condition. A RAERP bit of zero specifies that the device server shall not issue an asynchronous event report upon completing its initialization sequence.

NOTE 49 If the device server's default value for the RAERP bit is one and it does not implement saved parameters or include a hardware switch, then it may be impossible to disable the initialization sequence asynchronous event reporting.

A unit attention AER permission (UAAERP) bit of one specifies that the device server may issue an asynchronous event report instead of creating a unit attention condition upon detecting a unit attention condition event, other than

upon completing an initialization sequence. A UAAERP bit of zero specifies that the device server shall not issue an asynchronous event reporting instead of creating a unit attention condition.

An error AER permission (EAERP) bit of one specifies that the device server may issue an asynchronous event report upon detecting a deferred error condition instead of waiting to report the deferred error on the next command. An EAERP bit of zero specifies that the device server shall not report deferred error conditions via an asynchronous event reporting.

The AUTOLOAD MODE field specifies the action to be taken by a removable medium device server when a medium is inserted. For devices other than removable medium devices, this field is reserved. Table 158 shows the usage of the AUTOLOAD MODE field.

Table 158 — AUTOLOAD MODE field

Value	Definition
000b	Medium shall be loaded for full access.
001b	Medium shall be loaded for medium auxiliary memory access only.
010b	Medium shall not be loaded.
011b - 111b	Reserved

The READY AER HOLDOFF PERIOD field specifies the minimum time in milliseconds after the target starts its initialization sequence that it shall delay before attempting to issue an asynchronous event report. This value may be rounded up as defined in 5.3.

The BUSY TIMEOUT PERIOD field specifies the maximum time, in 100 milliseconds increments, that the initiator allows for the target to remain busy for unanticipated conditions that are not a routine part of commands from the initiator. This value may be rounded down as defined in 5.3. A 0000h value in this field is undefined by this standard. An FFFFh value in this field is defined as an unlimited period.

The EXTENDED SELF-TEST COMPLETION TIME field contains advisory data that an application client may use to determine the time in seconds that the device server requires to complete an extended self-test when the device server is not interrupted by subsequent commands and no errors occur during execution of the self-test. The application client should expect this time to increase significantly if other commands are sent to the logical unit while a self-test is in progress or if errors occur during execution of the self-test. Device servers supporting SELF-TEST CODE field values other than 000b for the SEND DIAGNOSTIC command (see 7.23), shall support the EXTENDED SELF-TEST COMPLETION TIME field.

8.3.7 Disconnect-reconnect page

The disconnect-reconnect page (see table 159) provides the application client the means to tune the performance of the service delivery subsystem. The name for this mode page, disconnect-reconnect, comes from the SCSI parallel bus. A SCSI device based on any of the protocols may use appropriate parameters in the disconnect-reconnect mode page. The parameters appropriate to each protocol and their interpretation for that protocol may be specified in the individual protocol documents.

Table 159 — Disconnect-reconnect page

Bit / Byte	7	6	5	4	3	2	1	0
0	PS	Reserved	PAGE CODE (02h)					
1	PAGE LENGTH (0Eh)							
2	BUFFER FULL RATIO							
3	BUFFER EMPTY RATIO							
4	(MSB)	BUS INACTIVITY LIMIT						
5								(LSB)
6	(MSB)	DISCONNECT TIME LIMIT						
7								(LSB)
8	(MSB)	CONNECT TIME LIMIT						
9								(LSB)
10	(MSB)	MAXIMUM BURST SIZE						
11								(LSB)
12	EMDP	FAIR ARBITRATION			DIMM	DTDC		
13	Reserved							
14	(MSB)	FIRST BURST SIZE						
15								(LSB)

The device server communicates the parameter values in this mode page to the service delivery subsystem. Similarly the application client may also communicate parameter values to the service delivery subsystem. This communication is internal to the initiator or target device and is outside the scope of SCSI.

If a parameter that is not appropriate for the specific protocol implemented by the SCSI device is non-zero, the device server shall return CHECK CONDITION status. The sense key shall be set to ILLEGAL REQUEST and the additional sense code set to ILLEGAL FIELD IN PARAMETER LIST.

An interconnect tenancy is a period of time during which a SCSI device owns or may access the interconnect. For example, on arbitrated interconnects, a tenancy typically begins when a SCSI device successfully arbitrates for the interconnect and ends when the SCSI device releases the interconnect for use by other devices. Data and other information transfers take place during interconnect tenancies.

The BUFFER FULL RATIO field indicates to the device server, during read operations, how full the buffer should be prior to requesting an interconnect tenancy. Device servers that do not implement the requested ratio should round down to the nearest implemented ratio as defined in 5.3.

The BUFFER EMPTY RATIO field indicates to the device server, during write operations, how empty the buffer should be prior to requesting an interconnect tenancy. Device servers that do not implement the requested ratio should round down to the nearest implemented ratio as defined in 5.3.

The buffer full and buffer empty ratios are numerators of a fractional multiplier that has 256 as its denominator. A value of zero indicates that the target determines when to request an interconnect tenancy consistent with the disconnect time limit parameter. These parameters are advisory to the target.

NOTE 50 As an example, consider a device server with ten 512-byte buffers and a specified buffer full ratio of 3Fh. The formula is: INTEGER((ratio/256)*number of buffers). Therefore in this example INTEGER((3Fh/256)*10) = 2. During the read operations described in this example, the device server should request an interconnect tenancy whenever two or more buffers are full.

The BUS INACTIVITY LIMIT field indicates the maximum time that the target is permitted to maintain an interconnect tenancy without data or information transfer. If the bus inactivity limit is exceeded the device server shall conclude the interconnect tenancy, within the restrictions placed on it by the applicable SCSI protocol. The contents of the DTDC field in this mode page also shall affect the duration of an interconnect tenancy. This value may be rounded as defined in 5.3. A value of zero indicates that there is no bus inactivity limit. Different protocols specify different units of measure for the bus inactivity limit.

The DISCONNECT TIME LIMIT field indicates the minimum time that the target shall wait between interconnect tenancies. This value may be rounded as defined in 5.3. A value of zero indicates that there is no disconnect time limit. Different protocols specify different units of measure for the disconnect time limit.

The CONNECT TIME LIMIT field indicates the maximum duration of a single interconnect tenancy. If the connect time limit is exceeded the device server shall conclude the interconnect tenancy, within the restrictions placed on it by the applicable SCSI protocol. The contents of the DTDC field in this mode page also shall affect the duration of an interconnect tenancy. This value may be rounded as defined in 5.3. A value of zero indicates that there is no connect time limit. Different protocols specify different units of measure for the connect time limit.

The MAXIMUM BURST SIZE field indicates the maximum amount of data that the device server shall transfer during a single data transfer operation. This value is expressed in increments of 512 bytes (e.g., a value of one means 512 bytes, two means 1024 bytes, etc.). The relationship (if any) between data transfer operations and interconnect tenancies is specified in the individual protocol documents. A value of zero indicates there is no limit on the amount of data transferred per data transfer operation.

The enable modify data pointers (EMDP) bit indicates whether or not the initiator allows the data transfer to be re-ordered by the target. If the EMDP bit is zero, the target shall not re-order the data transfer. If the EMDP bit is one, the target is allowed to re-order the data transfer.

The FAIR ARBITRATION field indicates whether the target should use fair or unfair arbitration when requesting an interconnect tenancy. The field may be used to indicate different fairness methods as specified in the individual protocol documents.

A disconnect immediate (DIMM) bit of zero indicates that the target may transfer data for a command during the same interconnect tenancy in which it receives the command. Whether or not the target does so may depend upon the target's internal algorithms, the rules of the applicable SCSI protocol, and settings of the other parameters in this mode page. A disconnect immediate (DIMM) bit of one indicates that the target shall not transfer data for a command during the same interconnect tenancy in which it receives the command.

The data transfer disconnect control (DTDC) field (see table 160) defines other restrictions on when multiple interconnect tenancies are permitted. A non-zero value in the DTDC field shall take precedence over other interconnect tenancy controls represented by other fields in this mode page.

Table 160 — Data transfer disconnect control

DTDC	Description
000b	Data transfer disconnect control is not used. Interconnect tenancies are controlled by other fields in this page.
001b	A target shall transfer all data for a command within a single interconnect tenancy.
010b	Reserved
011b	A target shall transfer all data for a command and complete the command within a single interconnect tenancy.
100b - 111b	Reserved

The FIRST BURST SIZE field indicates the maximum amount of data that may be transferred to the target for a command along with the command. This value is expressed in increments of 512 bytes; a value of one means 512 bytes, two means 1024 bytes, etc. A value of zero indicates that there is no first burst size limit.

8.3.8 Informational exceptions control page

The informational exceptions control page (see table 161) defines the methods used by the target to control the reporting and the operations of specific informational exception conditions. This page shall only apply to informational exceptions that report an additional sense code of FAILURE PREDICTION THRESHOLD EXCEEDED or WARNING to the application client.

Informational exception conditions occur as the result of vendor specific events within a target. An informational exception condition may occur asynchronous to any commands issued by an application client.

Table 161 — Informational exceptions control page

Bit / Byte	7	6	5	4	3	2	1	0
0	PS	Reserved	PAGE CODE (1Ch)					
1	PAGE LENGTH (0Ah)							
2	PERF	Reserved	EBF	EWASC	DEXCPT	TEST	Reserved	LOGERR
3	Reserved				MRIE			
4	(MSB)	INTERVAL TIMER						
7								(LSB)
8	(MSB)	REPORT COUNT						
11								(LSB)

The log errors bit (LOGERR) of zero indicates that the logging of informational exception conditions by a device server is vendor specific. A LOGERR bit of one indicates the device server shall log informational exception conditions.

A TEST bit of one shall create a test device failure at the next interval time, as specified by the INTERVAL TIMER field, if the DEXCPT bit is set to zero. When the TEST bit is one, the MRIE and REPORT COUNT fields shall apply as if the TEST bit were zero. The test device failure shall be reported with an additional sense code of FAILURE PREDICTION THRESHOLD EXCEEDED (FALSE). If both the TEST and the DEXCPT bits are one, the device server shall terminate the MODE SELECT command with CHECK CONDITION status. The sense key shall be set to ILLEGAL REQUEST with the additional sense code set to INVALID FIELD IN PARAMETER LIST. A TEST bit of zero shall instruct the device server not to generate any test device failure notifications.

A disable exception control (DEXCPT) bit of zero indicates the failure prediction threshold exceeded reporting shall be enabled. The method for reporting the failure prediction threshold exceeded when the DEXCPT bit is set to zero is determined from the MRIE field. A DEXCPT bit of one indicates the target shall disable reporting of the failure prediction threshold exceeded. The MRIE field is ignored when DEXCPT is set to one and EWASC is set to zero.

An enable warning (EWASC) bit of zero indicates the target shall disable reporting of the warning. The MRIE field is ignored when DEXCPT is set to one and EWASC is set to zero. An EWASC bit of one indicates warning reporting shall be enabled. The method for reporting the warning when the EWASC bit is set to one is determined from the MRIE field.

If background functions are supported, an Enable Background Function (EBF) bit of one indicates the target shall enable background functions. An EBF bit of zero indicates the target shall disable the functions.

For the purposes of the EBF bit, background functions are defined as idle time functions that may impact performance that are performed by a target operating without errors but do not impact the reliability of the target (e.g., read scan).

A performance (PERF) bit of zero indicates that informational exception operations that are the cause of delays are acceptable. A PERF bit of one indicates the device server shall not cause delays while doing informational exception operations. A PERF bit set to one may cause the device server to disable some or all of the informational exceptions operations, thereby limiting the reporting of informational exception conditions.

The method of reporting informational exceptions field (MRIE) indicates the methods that shall be used by the device server to report informational exception conditions (see table 162). The priority of reporting multiple information exceptions is vendor specific.

Table 162 — Method of reporting informational exceptions (MRIE) field (part 1 of 2)

MRIE	Description
0h	**No reporting of informational exception condition:** This method instructs the device server to not report information exception conditions.
1h	**Asynchronous event reporting:** This method instructs the device server to report informational exception conditions by using the rules for asynchronous event reporting as described in SAM-2 and the relevant protocol standard. The sense key shall be set to RECOVERED ERROR and the additional sense code shall indicate the cause of the informational exception condition.
2h	**Generate unit attention:** This method instructs the device server to report informational exception conditions by returning a CHECK CONDITION status. The sense key shall be set to UNIT ATTENTION and the additional sense code shall indicate the cause of the informational exception condition. The command that has the CHECK CONDITION shall not be executed before the informational exception condition is reported.
3h	**Conditionally generate recovered error:** This method instructs the device server to report informational exception conditions, if the reporting of recovered errors is allowed, by returning a CHECK CONDITION status. If the TEST bit equals zero, the status may be returned on any command after the informational exception condition occurs. If the TEST bit equals one, the status shall be returned on the next command that is normally capable of returning an informational exception condition when the TEST bit equals zero. The sense key shall be set to RECOVERED ERROR and the additional sense code shall indicate the cause of the informational exception condition. A command that has the CHECK CONDITION shall complete without error before any informational exception condition may be reported.
4h	**Unconditionally generate recovered error:** This method instructs the device server to report informational exception conditions, regardless of the value of the per bit of the error recovery mode page, by returning a CHECK CONDITION status. If the TEST bit equals zero, the status may be returned on any command after the informational exception condition occurs. If the TEST bit equals one, the status shall be returned on the next command that is normally capable of returning an informational exception condition when the TEST bit equals zero. The sense key shall be set to RECOVERED ERROR and the additional sense code shall indicate the cause of the informational exception condition. The command that has the CHECK CONDITION shall complete without error before any informational exception condition may be reported.

Table 162 — Method of reporting informational exceptions (MRIE) field (part 2 of 2)

MRIE	Description
5h	**Generate no sense:** This method instructs the device server to report informational exception conditions by returning a CHECK CONDITION status. If the TEST bit equals zero, the status may be returned on any command after the informational exception condition occurs. If the TEST bit equals one, the status shall be returned on the next command that is normally capable of returning an informational exception condition when the TEST bit equals zero. The sense key shall be set to NO SENSE and the additional sense code shall indicate the cause of the informational exception condition. The command that has the CHECK CONDITION shall complete without error before any informational exception condition may be reported.
6h	**Only report informational exception condition on request:** This method instructs the device server to preserve the informational exception(s) information. To find out about information exception conditions the application client polls the device server by issuing an unsolicited REQUEST SENSE command. The sense key shall be set to NO SENSE and the additional sense code shall indicate the cause of the informational exception condition.
7h - Bh	Reserved
Ch - Fh	Vendor specific

The INTERVAL TIMER field indicates the period in 100 millisecond increments for reporting that a informational exception condition has occurred. The device server shall not report informational exception conditions more frequently than the time specified by the INTERVAL TIMER field and as soon as possible after the timer interval has elapsed. After the informational exception condition has been reported the interval timer shall be restarted. A value of zero or FFFFFFFFh in the INTERVAL TIMER field shall indicate the timer interval is vendor specific.

The REPORT COUNT field indicates the number of times to report an informational exception condition to the application client. A value of zero in the REPORT COUNT field indicates there is no limit on the number of times the device server reports an informational exception condition.

The maintaining of the INTERVAL TIMER and the REPORT COUNT fields across power cycles and/or resets by the target are vendor specific.

8.3.9 Power condition page

The power condition page (see table 163) provides the application client the means to control the behavior of a logical unit in a manner that reduces the power required to operate. There shall be no notification to the initiator that a logical unit has entered into one of the power conditions. The application client may determine if a power condition is in effect by issuing a REQUEST SENSE command (see 7.20). In addition to the power condition page, the power conditions may be controlled by the START STOP UNIT command (see SBC). If both methods are being used on the same logical unit then any START STOP UNIT commands power condition request shall override the power condition pages power control.

No power condition shall affect the supply of any power required for proper operation of the service delivery subsystem.

On the receipt of a command the device server shall adjust itself to the power condition that allows the command to execute. The timer that maps to this power condition and any lower power condition timers shall be reset on receipt of the command. On completion of the command the timer associated with this power condition shall be restarted.

Logical units that contain cache memory shall implicitly perform a SYNCHRONIZE CACHE command (see SBC) for the entire medium prior to entering into any power condition that prevents access the media (e.g., the spindle being stopped).

The logical unit shall use the power condition page to control the power conditions after a power on or a hard reset until a START STOP UNIT command is received that sets power conditions.

Table 163 — Power condition page

Bit / Byte	7	6	5	4	3	2	1	0
0	PS	Reserved	PAGE CODE (1Ah)					
1	PAGE LENGTH (0Ah)							
2	Reserved							
3	Reserved						IDLE	STANDBY
4	(MSB)	IDLE CONDITION TIMER						
7								(LSB)
8	(MSB)	STANDBY CONDITION TIMER						
11								(LSB)

An IDLE bit of one indicates that the logical unit shall use the IDLE CONDITION TIMER field to determine the length of inactivity time to wait before entering the idle condition. An IDLE bit of zero indicates that the logical unit shall not enter the idle condition.

A STANDBY bit of one indicates that the logical unit shall use the STANDBY CONDITION TIMER field to determine the length of inactivity time to wait before entering the standby condition. A STANDBY bit of zero indicates that the logical unit shall not enter the standby condition.

The IDLE CONDITION TIMER field indicates the inactivity time in 100 ms increments that the logical unit shall wait before entering the idle condition.

If the IDLE bit is one, a value of zero in the idle condition timer indicates the logical unit shall enter the idle condition on completion of any command.

The STANDBY CONDITION TIMER field indicates the inactivity time in 100 millisecond increments that the logical unit shall wait before entering the standby condition. This timer shall only count if the idle condition timer is equal to zero.

If the STANDBY bit is one and the IDLE bit is zero, a value of zero in the standby condition timer indicates the logical unit shall enter the standby condition on completion of any command.

If the STANDBY bit is one and the IDLE bit is one, a value of zero in the standby condition timer indicates the logical unit shall enter the standby condition when the idle condition timer equals zero.

Figure 3 shows graphically the relationships between the different power conditions and their timers.

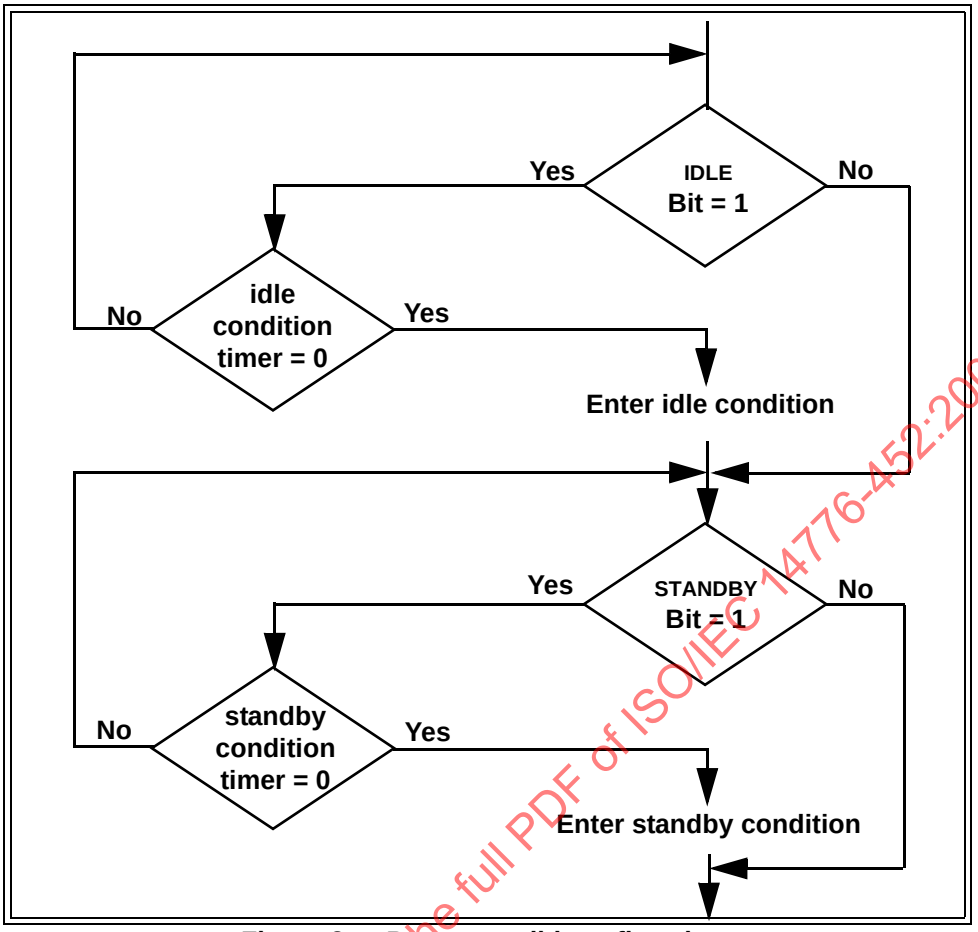

Figure 3 — Power conditions flowchart

8.3.10 Protocol specific LUN page

The protocol specific LUN page (see table 164) provides protocol specific controls that are associated with a logical unit. See the protocol standard (see 3.1.41) for definition of the protocol specific mode parameters.

Table 164 — Protocol specific LUN page

Bit Byte	7	6	5	4	3	2	1	0
0	PS	Reserved	PAGE CODE (18h)					
1	PAGE LENGTH (n-1)							
2	Reserved				PROTOCOL IDENTIFIER			
3	Protocol specific mode parameters							
n								

The PROTOCOL IDENTIFIER field indicates the protocol to which the page applies. For MODE SENSE commands, the device server shall set the PROTOCOL IDENTIFIER field to one of the values shown in table 165 to indicate the protocol used by its service delivery subsystem. For MODE SELECT commands, the application client shall set the PROTOCOL IDENTIFIER field to one of the values shown in table 165 indicating the protocol to which the protocol specific mode parameters apply. If a device server receives a page containing a protocol identifier value other than the one used by its service delivery subsystem, it shall terminate the command with a CHECK CONDITION status.

The sense key shall be set to ILLEGAL REQUEST and the additional sense code shall be set to INVALID FIELD IN PARAMETER LIST.

Table 165 — PROTOCOL IDENTIFIER values

Protocol Identifier	Description
0	Fibre Channel (FCP-n)
1	Parallel SCSI (SPI-n)
2	SSA (SSA-S2P or SSA-S3P)
3	IEEE 1394 (SBP-2)
4	SRP
5	iSCSI
7 - 15	Reserved

8.3.11 Protocol specific port page

The protocol specific port page (see table 166) provides protocol specific controls that are associated with a port. See the protocol standard (see 3.1.41) for definition of the protocol specific mode parameters.

Table 166 — Protocol specific port page

Bit / Byte	7	6	5	4	3	2	1	0
0	PS	Reserved	PAGE CODE (19h)					
1	PAGE LENGTH (n-1)							
2	Reserved				PROTOCOL IDENTIFIER			
3	Protocol specific mode parameters							
n								

The PROTOCOL IDENTIFIER field indicates the protocol to which the page applies. For MODE SENSE commands, the device server shall set the PROTOCOL IDENTIFIER field to one of the values shown in table 165 to indicate the protocol used by its service delivery subsystem. For MODE SELECT commands, the application client shall set the PROTOCOL IDENTIFIER field to one of the values shown in table 165 indicating the protocol to which the protocol specific mode parameters apply. If a device server receives a page containing a protocol identifier value other than the one used by its service delivery subsystem, it shall terminate the command with a CHECK CONDITION status. The sense key shall be set to ILLEGAL REQUEST and the additional sense code shall be set to INVALID FIELD IN PARAMETER LIST.

8.4 Vital product data parameters

8.4.1 Vital product data parameters overview and page codes

This subclause describes the vital product data page structure and the vital product data pages (see table 167) that are applicable to all SCSI devices. These pages are optionally returned by the INQUIRY command (see 7.3).

Table 167 — Vital product data page codes

Page code	Description	Reference	Support Requirements
82h	ASCII implemented operating definition page	8.4.2	Optional
01h - 7Fh	ASCII information page	8.4.3	Optional
83h	Device identification page	8.4.4	Mandatory
81h	Obsolete	3.3.7	
00h	Supported vital product data pages	8.4.5	Mandatory
80h	Unit serial number page	8.4.6	Optional
84h - BFh	Reserved		
C0h - FFh	Vendor specific		

8.4.2 ASCII implemented operating definition page

The ASCII implemented operation definition page (see table 168) contains operating definition description data for all operating definitions implemented by the target.

Table 168 — ASCII implemented operating definition

Bit / Byte	7	6	5	4	3	2	1	0
0	PERIPHERAL QUALIFIER			PERIPHERAL DEVICE TYPE				
1	PAGE CODE (82h)							
2	Reserved							
3	PAGE LENGTH (n-3)							
4	ASCII OPERATING DEFINITION DESCRIPTION LENGTH (m-4)							
5	ASCII OPERATING DEFINITION DESCRIPTION DATA							
m								
m+1	Vendor specific description data							
n								

The PERIPHERAL QUALIFIER field and the PERIPHERAL DEVICE TYPE field are as defined in 7.3.2.

The PAGE LENGTH field specifies the length of the following page data. If the allocation length is less than the length of the data to be returned, the page length shall not be adjusted to reflect the truncation.

The ASCII OPERATING DEFINITION DESCRIPTION LENGTH field specifies the length in bytes of the ASCII OPERATING DEFINITION DESCRIPTION DATA field that follows. If the allocation length is less than the length of data to be returned, the ASCII operating definition description length shall not be adjusted to reflect the truncation. A value of zero in this field indicates that no ASCII operating definition description data is available.

The ASCII OPERATING DEFINITION DESCRIPTION DATA field contains the ASCII operating definition description data for the device server. The data in this field shall be formatted in lines (or character strings). Each line shall contain only graphic codes (i.e., code values 20h through 7Eh) and shall be terminated with a NULL (00h) character. The text is vendor specific.

8.4.3 ASCII information page

The ASCII information page (see table 169) contains information for the field replaceable unit code returned in the REQUEST SENSE data (see 7.20.2).

Table 169 — ASCII information page

Bit / Byte	7	6	5	4	3	2	1	0
0	PERIPHERAL QUALIFIER			PERIPHERAL DEVICE TYPE				
1	PAGE CODE (01h - 7Fh)							
2	Reserved							
3	PAGE LENGTH (n-3)							
4	ASCII LENGTH (m-4)							
5	ASCII INFORMATION							
m								
m+1	Vendor specific information							
n								

The PERIPHERAL QUALIFIER field and the PERIPHERAL DEVICE TYPE field are defined in 7.3.2.

The PAGE CODE field contains the same value as in the PAGE OR OPERATION CODE field of the INQUIRY CDB (see 7.3) and is associated with the FIELD REPLACEABLE UNIT CODE field returned by the REQUEST SENSE command.

NOTE 51 The FIELD REPLACEABLE UNIT CODE field in the sense data provides for 255 possible codes, while the page code field provides for only 127 possible codes. For that reason it is not possible to return ASCII information pages for the upper code values.

The PAGE LENGTH field specifies the length of the following page data. If the allocation length of the CDB is too small to transfer all of the page, the page length shall not be adjusted to reflect the truncation.

The ASCII LENGTH field specifies the length in bytes of the ASCII INFORMATION field that follows. If the allocation length is less than the length of the data to be returned, the ASCII length shall not be adjusted to reflect the truncation. A value of zero in this field indicates that no ASCII information is available for the specified page code.

The ASCII INFORMATION field contains ASCII information concerning the field replaceable unit identified by the page code. The data in this field shall be formatted in one or more character string lines. Each line shall contain only graphic codes (i.e., code values 20h through 7Eh) and shall be terminated with a NULL (00h) character.

The contents of the vendor specific information field is not defined in this standard.

8.4.4 Device identification page

The device identification page (see table 170) provides the means to retrieve zero or more identification descriptors applying to the logical unit. Logical units may have more than one identification descriptor (e.g., if several types or associations of identifier are supported).

Device identifiers, if any, shall be assigned to the peripheral device (e.g., a disk drive) and not to the currently mounted media, in the case of removable media devices. Media identification is outside the scope of this standard. Operating systems are expected to use the device identifiers during system configuration activities to determine whether alternate paths exist for the same peripheral device.

NOTE 52 In the case of virtual logical units (e.g., volume sets as defined by SCC-2), the Identifier field (see table 171) should be a concatenation of all the bytes in an IEEE Registered Extended name. The IEEE Registered Extended name has a code of 3h in the Identifier type field and an NAA value of 0110b as defined in FC-FS.

Table 170 — Device identification page

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

The PERIPHERAL QUALIFIER field and the PERIPHERAL DEVICE TYPE field in table 170 are as defined in 7.3.2.

Each Identification descriptor (see table 171) contains information identifying the logical unit, physical device, or access path used by the command and returned parameter data. The ASSOCIATION field indicates the entity that the Identification descriptor describes. If a physical or logical device returns an Identification descriptor with the ASSOCIATION field set to 0h, it shall return the same descriptor when it is accessed through any other path.

Table 171 — Identification descriptor

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

The CODE SET field specifies the code set used for the identifier field, as described in table 172. This field is intended to be an aid to software that displays the identifier field.

Table 172 — Code set

Value	Description
0h	Reserved
1h	The identifier field shall contain binary values.
2h	The identifier field shall contain ASCII graphic codes (i.e., code values 20h through 7Eh)
3h - Fh	Reserved

The ASSOCIATION field specifies the entity with which the IDENTIFIER field is associated, as described in table 173.

Table 173 — Association

Value	Description
0h	The IDENTIFIER field is associated with the addressed physical or logical device.
1h	The IDENTIFIER field is associated with the port that received the request.
2h - 3h	Reserved

The IDENTIFIER TYPE field specifies the format and assignment authority for the identifier, as described in table 174. At least one identification descriptor shall contain 1h, 2h, or 3h in the IDENTIFIER TYPE field and 0h in the ASSOCIATION field. At least one identification descriptor should contain 2h or 3h in the IDENTIFIER TYPE field and 0h in the ASSOCIATION field.

Table 174 — Identifier type

Value	Description
0h	No assignment authority was used and consequently there is no guarantee that the identifier is globally unique (i.e., the identifier is vendor specific)
1h	The first eight bytes of the identifier field are a Vendor ID (see Annex D). The organization associated with the Vendor ID is responsible for ensuring that the remainder of the identifier field is unique. One recommended method of constructing the remainder of the identifier field is to concatenate the product identification field from the standard INQUIRY data field and the product serial number field from the unit serial number page.
2h	The identifier field contains a canonical form IEEE Extended Unique Identifier, 64-bit (EUI-64). In this case, the identifier length field shall be set to eight. Note that the IEEE guidelines for EUI-64 specify a method for unambiguously encapsulating an IEEE 48-bit identifier within an EUI-64.
3h	The identifier field contains an FC-FS Name_Identifier. Any FC-FS identifier may be used, including one of the four based on a Canonical form IEEE company_id.
4h	If the ASSOCIATION value is 1h, the IDENTIFIER value contains a four-byte binary number identifying the port relative to other ports in the device using the values shown table 175. In this case, the CODE SET field shall be set to 1h and the IDENTIFIER LENGTH field shall be set to 4. If the ASSOCIATION value is not 1h, use of this identifier type is reserved.
5h - Fh	Reserved

Table 175 — Relative port identifier values

Value	Description
0h	Reserved
1h	Relative port 1, historically known as port A
2h	Relative port 2, historically known as port B
3h-7FFFFFFFh	Relative port 3 through 2 147 483 647
80000000h-FFFFFFFFh	Reserved

The IDENTIFIER LENGTH field specifies the length in bytes of the IDENTIFIER field. If the allocation length field of the CDB is too small to transfer all of the identifier, the identifier length shall not be adjusted to reflect the truncation.

The IDENTIFIER field contains the identifier as described by the ASSOCIATION, IDENTIFIER TYPE, CODE SET, and IDENTIFIER LENGTH fields.

The example described in this paragraph and shown in table 176 is not a normative part of this standard. This example of a complete device identification VPD page assumes that the product is a direct-access device with an T10 Vendor ID of “XYZ_Corp”, a product identification of “Super Turbo Disk”, and a product serial number of “2034589345”. Furthermore, it is assumed that the manufacturer has been assigned a 24-bit IEEE company_id of 01ABCDh by the IEEE Registration Authority Committee and that the manufacture has assigned a 24-bit

extension_identifier of 234567h to this logical unit. The combined 48-bit identifier is reported in the 64-bit format as defined by the IEEE 64-bit Global Identifier (EUI-64) standard. The data returned in the device identification VPD page for this logical unit is shown in table 176.

Table 176 — Device identification page example

Bytes	Hexadecimal values				ASCII values
00 – 15	00 83 00 32	02 01 00 22	58 59 5A 5F	43 6F 72 70	...2..."XYZ_Corp
16 – 31	53 75 70 65	72 20 54 75	72 62 6F 20	44 69 73 6B	Super Turbo Disk
32 – 47	32 30 33 34	35 38 39 33	34 35 01 02	00 08 01 AB	2034589345.....
48 – 53	CD FF FF 23	45 67			.....

NOTE 1 Non-printing ASCII characters are shown as '.'.
NOTE 2 Byte 00 is the beginning of the VPD page (see table 170).
NOTE 3 Byte 04 is the beginning of the Identification descriptor for the Vendor ID based identifier (Identifier type 1, see table 174).
NOTE 4 Byte 42 is the beginning of the Identification descriptor for the EUI-64 identifier (Identifier type 2, see table 174).

8.4.5 Supported vital product data pages

This contains a list of the vital product data page codes supported by the target or logical unit (see table 177). If a device server supports any vital product data pages, it also shall support this vital product data page.

Table 177 — Supported vital product data pages

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

The PERIPHERAL QUALIFIER field and the PERIPHERAL DEVICE TYPE field are defined in 7.3.2.

The PAGE LENGTH field specifies the length of the supported page list. If the allocation length is too small to transfer all of the page, the page length shall not be adjusted to reflect the truncation.

The SUPPORTED PAGE LIST field shall contain a list of all vital product data page codes (see 8.4) implemented for the target or logical unit in ascending order beginning with page code 00h.

8.4.6 Unit serial number page

This page (see table 178) provides a product serial number for the target or logical unit.

Table 178 — Unit serial number page

Bit / Byte	7	6	5	4	3	2	1	0
0	PERIPHERAL QUALIFIER			PERIPHERAL DEVICE TYPE				
1	PAGE CODE (80h)							
2	Reserved							
3	PAGE LENGTH (n-3)							
4	PRODUCT SERIAL NUMBER							
n								

The PERIPHERAL QUALIFIER field and the PERIPHERAL DEVICE TYPE field are defined in 7.3.2.

The PAGE LENGTH field specifies the length of the product serial number. If the allocation length is too small to transfer all of the page, the page length shall not be adjusted to reflect the truncation.

The PRODUCT SERIAL NUMBER field contains ASCII data that is vendor-assigned serial number. The least significant ASCII character of the serial number shall appear as the last byte in the Data-In Buffer. If the product serial number is not available, the device server shall return ASCII spaces (20h) in this field.

9 Commands for processor type devices

9.1 Summary of commands for processor type devices

The commands for processor type devices shall be as listed in table 179.

Table 179 — Commands for processor devices

Command name	Operation code	Type	Reference
Obsolete	40h	OB	
Obsolete	39h	OB	
Obsolete	18h	OB	
Obsolete	3Ah	OB	
EXTENDED COPY	83h	O	7.2
INQUIRY	12h	M	7.3
LOG SELECT	4Ch	O	7.4
LOG SENSE	4Dh	O	7.5
PERSISTENT RESERVE IN	5Eh	O	7.10
PERSISTENT RESERVE OUT	5Fh	O	7.11
READ BUFFER	3Ch	O	7.13
RECEIVE	08h	O	9.2
RECEIVE COPY RESULTS	84h	O	7.14
RECEIVE DIAGNOSTIC RESULTS	1Ch	O	7.15
RELEASE(10)	57h	O	7.16
RELEASE(6)	17h	O	7.17
REPORT LUNS	A0h	O	7.19
REQUEST SENSE	03h	M	7.20
RESERVE(10)	56h	O	7.21
RESERVE(6)	16h	O	7.22
SEND	0Ah	O	9.3
SEND DIAGNOSTIC	1Dh	M	7.23
TEST UNIT READY	00h	M	7.25
WRITE BUFFER	3Bh	O	7.26

Key: M = Command implementation is mandatory.
O = Command implementation is optional.
OB = Command implementation is defined in a previous standard

The following operation codes are vendor specific: 02h, 05h, 06h, 09h, 0Ch, 0Dh, 0Eh, 0Fh, 10h, 11h, 13h, 14h, 19h, C0h through FFh. All remaining operation codes for processor devices are reserved.

9.2 RECEIVE command

The RECEIVE command (see table 180) requests that the device server transfer data to the initiator. The contents of the data are not defined by this standard.

Table 180 — RECEIVE command

Bit / Byte	7	6	5	4	3	2	1	0
0	OPERATION CODE (08h)							
1	Reserved							
2	(MSB)							
3				TRANSFER LENGTH				
4								(LSB)
5	CONTROL							

The TRANSFER LENGTH field specifies the length in bytes of data that shall be transferred to the Data-In Buffer. A transfer length of zero indicates that no data shall be sent. This condition shall not be considered an error.

9.3 SEND command

The SEND command (see table 181) requests that the device server transfer data from the initiator.

Table 181 — SEND command

Bit / Byte	7	6	5	4	3	2	1	0
0	OPERATION CODE (0Ah)							
1	Reserved							AER
2	(MSB)							
3				TRANSFER LENGTH				
4								(LSB)
5	CONTROL							

An asynchronous event reporting (AER) bit of one indicates that the data to be transferred conforms to AER data format as defined in table 182. A SEND command with an AER bit of one shall be only issued to logical unit zero. An AER bit of zero indicates that the data to be transferred are vendor specific.

The TRANSFER LENGTH field specifies the length in bytes of data that shall be transferred from the Data-Out Buffer. A transfer length of zero indicates that no data shall be sent. This condition shall not be considered an error.

Table 182 — SEND command – AER data format

Bit / Byte	7	6	5	4	3	2	1	0
0	SCSI-3	Reserved						
1	Reserved							
3								
4	(MSB)	LUN						
11								(LSB)
12	Sense data byte (0)							
n+12	Sense data byte (n)							

If the SCSI-3 bit is zero, then the AEN data format, as defined by the SCSI-2 standard, shall be used. If the SCSI-3 bit is one, then the AER data format shown in table 182 shall be used.

The LUN field shall contain the logical unit number on which the asynchronous event occurred. The LUN field shall have the properties defined in SAM-2.

The sense data bytes shall have the format defined in 7.20.2.

10 Parameters for processor type devices

10.1 Diagnostic parameters

This subclause defines the descriptors and pages for diagnostic parameters used with processor type devices.

The diagnostic page codes for processor devices are defined in table 183.

Table 183 — Processor diagnostic page codes

Page Code	Description	Reference
00h	Supported diagnostics pages	8.1.2
01h - 3Fh	Reserved (for pages that apply to all device types)	
40h - 7Fh	Reserved	
80h - FFh	Vendor specific pages	

10.2 Log parameters

This subclause defines the descriptors and pages for log parameters used with processor type devices.

The log page codes for processor devices are defined in table 184.

Table 184 — Processor log page codes

Page Code	Description	Reference
01h	Buffer over-run/under-run page	8.2.3
0Bh	Last *n* deferred errors or asynchronous events page	8.2.5
07h	Last *n* error events page	8.2.6
06h	Non-medium error page	8.2.7
00h	Supported log pages	8.2.10
02h - 05h	Reserved	
08h - 0Ah	Reserved	
0Ch - 2Fh	Reserved	
3Fh	Reserved	
30h - 3Eh	Vendor specific pages	

10.3 Vital product data parameters

This subclause defines the descriptors and pages for vital product data parameters used with processor type devices.

The vital product data page codes for processor devices are defined in table 185.

Table 185 — Processor vital product data page codes

Page code	Description	Reference	Support Requirements
82h	ASCII implemented operating definition page	8.4.2	Optional
01h - 7Fh	ASCII information page	8.4.3	Optional
83h	Device identification page	8.4.4	Mandatory
00h	Supported vital product data pages	8.4.5	Mandatory
80h	Unit serial number page	8.4.6	Optional
C0h - FFh	Vendor specific		

Annex A
(Informative)

Procedures for logging operations in SCSI

A.1 Procedures for logging operations in SCSI introduction

This annex provides guidance in the use of the LOG SELECT and LOG SENSE commands defined in clause 7. This annex does not replace the descriptions in clause 7 and is not intended to conflict with clause 7. The purpose of this annex is to provide more information to gain a more uniform implementation of the SCSI logging functions.

A.2 Logging operations terminology

A.2.1
list parameter:
A parameter value that consists of a string of ASCII graphic codes or a binary value.

A.2.2
log page:
A page made up of one or more log parameters.

A.2.3
log parameter:
Log information that is made up of a parameter code, a parameter control byte, and a parameter value.

A.2.4
parameter code:
A unique identifier that is used to distinguish between the different log parameters within a single log page.

A.2.5
parameter control byte:
Used to tell the device server how to update, save, use thresholds, determine format, etc. of the parameter value.

A.2.6
parameter pointer field:
Contains a parameter code.

A.2.7
parameter value:
A counter, cumulative, threshold, or ASCII value.

A.2.8
GT:
Greater Than

A.2.9
NV:
Not Valid

A.3 LOG SENSE command

The LOG SENSE command may be used to do two functions. One is to allow the device server to save the log parameters in a log page to nonvolatile storage. The other is to allow an application client to receive the value of the current log parameters for a given log page.

Table A.1 lists the definitions of the LOG SENSE CDB fields.

Table A.1 — LOG SENSE Command CDB fields

LOG SENSE CDB Values			
PPC bit	**SP** bit	**PC** field	**Description**
0	-	--	Indicates that the log parameter requested from the device server begin with the parameter code specified by the PARAMETER POINTER field in ascending order of parameter codes from the specified log page.
1	-	--	Indicates that the device server returns a log page consisting only of the log parameters in which a log parameter value has changed since the last LOG SELECT or LOG SENSE command. The device server returns only those log parameters with a parameter code greater than or equal to the parameter code specified by the PARAMETER POINTER field.
-	0	--	Indicates that the device server performs the specified LOG SENSE command and does not save any log parameters.
-	1	--	Indicates that the device server performs the specified LOG SENSE command and saves all log parameters identified as savable by the DS bit to a nonvolatile vendor specific location if allowed. See the table A.3 to determine the interaction between the SP and DS bits to see what 'allowed' means.
-	-	00	Indicates that the device server returns current threshold values.
-	-	01	Indicates that the device server returns current cumulative values.
-	-	10	Indicates that the device server returns default threshold values.
-	-	11	Indicates that the device server returns default cumulative values.