ISO/IEC 14776-454

Edition 1.0 2018-04

INTERNATIONAL STANDARD

colour inside

Information technology – Small Computer System Interface (SCSI) – Part 454: SCSI Primary Commands – 4 (SPC-4)

ISO/IEC 14776-454:2018-04(en)

IEC Central Office
3, rue de Varembé
CH-1211 Geneva 20
Switzerland

Tel.: +41 22 919 02 11
info@iec.ch
www.iec.ch

About the IEC
The International Electrotechnical Commission (IEC) is the leading global organization that prepares and publishes International Standards for all electrical, electronic and related technologies.

About IEC publications
The technical content of IEC publications is kept under constant review by the IEC. Please make sure that you have the latest edition, a corrigenda or an amendment might have been published.

IEC Catalogue - webstore.iec.ch/catalogue
The stand-alone application for consulting the entire bibliographical information on IEC International Standards, Technical Specifications, Technical Reports and other documents. Available for PC, Mac OS, Android Tablets and iPad.

IEC publications search - webstore.iec.ch/advsearchform
The advanced search enables to find IEC publications by a variety of criteria (reference number, text, technical committee,…). It also gives information on projects, replaced and withdrawn publications.

IEC Just Published - webstore.iec.ch/justpublished
Stay up to date on all new IEC publications. Just Published details all new publications released. Available online and also once a month by email.

Electropedia - www.electropedia.org
The world's leading online dictionary of electronic and electrical terms containing 21 000 terms and definitions in English and French, with equivalent terms in 16 additional languages. Also known as the International Electrotechnical Vocabulary (IEV) online.

IEC Glossary - std.iec.ch/glossary
67 000 electrotechnical terminology entries in English and French extracted from the Terms and Definitions clause of IEC publications issued since 2002. Some entries have been collected from earlier publications of IEC TC 37, 77, 86 and CISPR.

IEC Customer Service Centre - webstore.iec.ch/csc
If you wish to give us your feedback on this publication or need further assistance, please contact the Customer Service Centre: sales@iec.ch.

ISO/IEC 14776-454

Edition 1.0 2018-04

INTERNATIONAL STANDARD

colour inside

Information technology – Small Computer System Interface (SCSI) – Part 454: SCSI Primary Commands – 4 (SPC-4)

INTERNATIONAL ELECTROTECHNICAL COMMISSION

ICS 35.200

ISBN 978-2-8322-5595-7

Contents

Tables

Figures

INFORMATION TECHNOLOGY – SMALL COMPUTER SYSTEM INTERFACE (SCSI) –

Part 454: SCSI Primary Commands - 4 (SPC-4)

FOREWORD

1) ISO (the International Organization for Standardization) and IEC (the 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. In the field of information technology, ISO and IEC have established a joint technical committee, ISO/IEC JTC 1.
2) The formal decisions or agreements of IEC and ISO on technical matters express, as nearly as possible, an international consensus of opinion on the relevant subjects since each technical committee has representation from all interested IEC National Committees and ISO member bodies.
3) IEC, ISO and ISO/IEC publications have the form of recommendations for international use and are accepted by IEC National Committees and ISO member bodies in that sense. While all reasonable efforts are made to ensure that the technical content of IEC, ISO and ISO/IEC publications is accurate, IEC or ISO cannot be held responsible for the way in which they are used or for any misinterpretation by any end user.
4) In order to promote international uniformity, IEC National Committees and ISO member bodies undertake to apply IEC, ISO and ISO/IEC publications transparently to the maximum extent possible in their national and regional publications. Any divergence between any ISO, IEC or ISO/IEC publication and the corresponding national or regional publication should be clearly indicated in the latter.
5) ISO and IEC do not provide any attestation of conformity. Independent certification bodies provide conformity assessment services and, in some areas, access to IEC marks of conformity. ISO or IEC are not responsible for any services carried out by independent certification bodies.
6) All users should ensure that they have the latest edition of this publication.
7) 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 National Committees or ISO member bodies 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, ISO or ISO/IEC publications.
8) 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.
9) Attention is drawn to the possibility that some of the elements of this ISO/IEC publication may be the subject of patent rights. ISO and IEC shall not be held responsible for identifying any or all such patent rights.

International Standard ISO/IEC 14776-454 was prepared by subcommittee 25: Interconnection of information technology equipment, of ISO/IEC joint technical committee 1: Information technology.

The list of all currently available parts of the ISO/IEC 14776 series, under the general title *Information technology – Small computer system interface (SCSI)*, can be found on the IEC web site.

This International Standard has been approved by vote of the member bodies and the voting results may be obtained from the address given on the second title page.

This publication has been drafted in accordance with the ISO/IEC Directives, Part 2 except as described in 3.4 and 3.5.

A bilingual version of this publication may be issued at a later date.

IMPORTANT - The 'colour inside' logo on the cover page of this publication indicates that it contains colours which are considered to be useful for the correct understanding of its contents. Users should therefore print this document using a colour printer.

INTRODUCTION

The SCSI command set is designed to provide efficient peer-to-peer operation of SCSI devices (e.g., disks, tapes, media changers) by an operating system. The SCSI command set assumes an underlying command-response protocol.

The SCSI command set provides multiple operating systems concurrent control over one or more SCSI devices. However, proper coordination of activities between the multiple operating systems is critical to avoid data corruption. Commands that assist with coordination between multiple operating systems are described in this standard. However, details of the coordination are beyond the scope of the SCSI command set.

This standard defines the device model for all SCSI devices. This standard defines the SCSI commands that are basic to every device model and the SCSI commands that may apply to any device model.

With any technical document there may arise questions of interpretation as new products are implemented. INCITS has established procedures to issue technical opinions concerning the standards developed by INCITS. These procedures may result in SCSI Technical Information Bulletins being published by INCITS.

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.

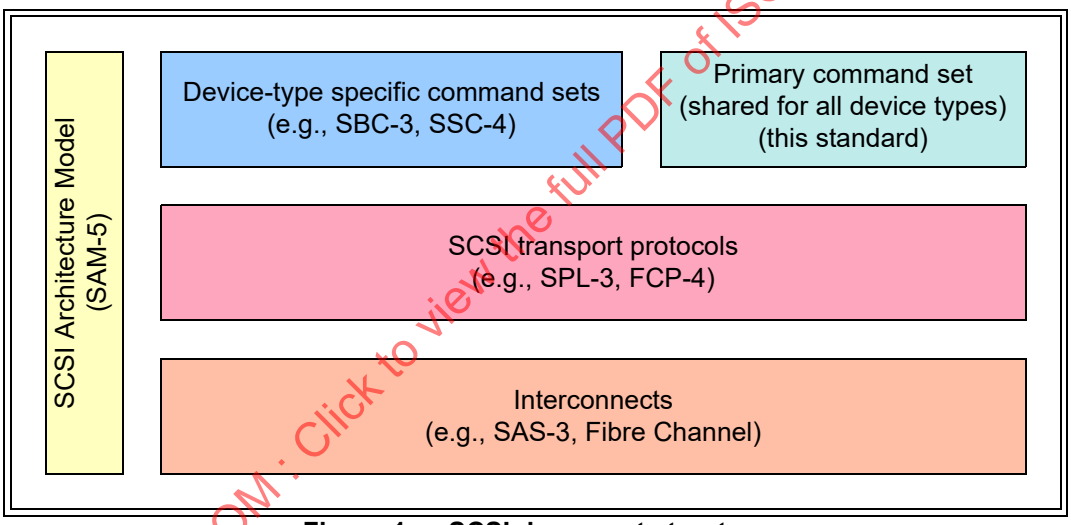


Figure 1 — SCSI document structure

The SCSI document structure in figure 1 is intended to show the general applicability of the documents to one another. Figure 1 is not intended to imply a relationship such as a hierarchy, protocol stack, or system architecture.

SCSI Architecture Model: Defines the SCSI systems model, the functional partitioning of the SCSI standard set and requirements applicable to all SCSI implementations and implementation standards.

Device-Type Specific Command Sets: Implementation standards that define specific device types including a device model for each device type. These standards specify the required commands and behaviors that are specific to a given device type and prescribe the requirements to be followed by a SCSI initiator device when sending commands to a SCSI target device having the specific device type. The commands and behaviors for a specific device type may include by reference commands and behaviors that are shared by all SCSI devices.

Shared Command Set: An implementation standard that defines a model for all SCSI device types. This standard specifies the required commands and behavior that is common to all SCSI devices, regardless of

device type, and prescribes the requirements to be followed by a SCSI initiator device when sending commands to any SCSI target device.

SCSI Transport Protocols: Implementation standards that define the requirements for exchanging information so that different SCSI devices are capable of communicating.

Interconnects: Implementation standards that define the communications mechanism employed by the SCSI transport protocols. These standards may describe the electrical and signaling requirements essential for SCSI devices to interoperate over a given interconnect. Interconnect standards may allow the interconnection of devices other than SCSI devices in ways that are outside the scope of this standard.

The term SCSI is used to refer to ISO/IEC 14776 (all parts).

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

INFORMATION TECHNOLOGY – SMALL COMPUTER SYSTEM INTERFACE (SCSI) –

Part 454: SCSI Primary Commands - 4 (SPC-4)

1 Scope

ISO/IEC 14776 (all parts) provides for many different types of SCSI devices (e.g., disks, tapes, media changers). This standard defines a device model that is applicable to all SCSI devices. Other command standards expand on the general SCSI device model in ways appropriate to specific types of SCSI devices.

ISO/IEC 14776 (all parts) specifies the interfaces, functions, and operations necessary to ensure interoperability between conforming SCSI implementations. This standard is a functional description. Conforming implementations employ any design technique that does not violate interoperability.

This standard defines the SCSI commands that are mandatory and optional for all SCSI devices. Support for any feature defined in this standard is optional unless otherwise stated. This standard also defines the SCSI commands that may apply to any device model.

The following commands, parameter data, and features defined in previous versions of the SPC standard are made obsolete by this standard:

a) the TARGET RESET supported (TRS) bit and the WAKEUP supported (WAKES) bit in the REPORT SUPPORTED TASK MANAGEMENT FUNCTIONS parameter data;
b) code value 10b (i.e., Per initiator port) in the MODE PAGE POLICY field in the mode page policy descriptor in the Mode Page Policy VPD page;
c) the removable medium devices with an attached medium changer model, MCHNGR bit in the standard INQUIRY data, the MOVE MEDIUM ATTACHED command in disks and tapes, and the READ ELEMENT STATUS ATTACHED command in disks and tapes;
d) linked commands;
e) the PPC bit in the LOG SENSE command;
f) the NUL bit in EXTENDED COPY command target descriptors (i.e., CSCD descriptors in this standard);
g) EXTENDED COPY support for the processing of setmarks by sequential-access devices;
h) READ BUFFER commands with the MODE field set to 00h and 1Ah; and
i) WRITE BUFFER commands with the MODE field set to 00h, 1Ah, and 1Bh.

2 Normative references

The following documents are referred to in the text in such a way that some or all of their content constitutes requirements 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.

ISO/IEC 14776-115, *Information technology - Small Computer System Interface (SCSI) - Part 115: Parallel Interface - 5 (SPI-5)*

ISO/IEC 14776-153, *Information technology - Small Computer System Interface (SCSI) - Part 153: Serial Attached SCSI - 2.1 (SAS-2.1)*

ISO/IEC 14776-322, *Information technology - Small Computer System Interface (SCSI) - Part 322: SCSI Block Commands - 2 (SBC-2)*

ISO/IEC 14776-323, *Information technology - Small Computer System Interface (SCSI) - Part 323: SCSI Block Commands - 3 (SBC-3)*

ISO/IEC 14776-331, *Information technology - Small Computer System Interface (SCSI) - Part 331: Stream Commands (SSC)*

ISO/IEC 14776-414, *Information technology - Small Computer System Interface (SCSI) - Part 414: SCSI Architecture Model - 4 (SAM-4)*

ISO/IEC 14776-452, *Information technology - Small Computer System Interface (SCSI) - Part 452: SCSI Primary Commands - 2 (SPC-2)*

ISO/IEC 14776-453, *Information technology - Small Computer System Interface (SCSI) - Part 453: SCSI Primary Commands - 3 (SPC-3)*

ISO 5807:1985, *Information processing - Documentation symbols and conventions for data, program and system flowcharts, program network charts and system resources charts*

INCITS 318-1998, *SCSI Controller Commands - 2 (SCC-2)*

INCITS 365, *Information technology SCSI RDMA Protocol (SRP)*

INCITS 375-2004, *Information technology - Serial Bus Protocol 3 (SBP-3)*

INCITS 457, *Information technology - Serial Attached SCSI - 2 (SAS-2)* [1)]

INCITS 468-2010, *Information technology - Multi-Media Command Set - 6 (MMC-6)*

INCITS 468-2010/AM 1-2012, *Information technology - Multi-Media Command Set 6 - Amendment 1 (MMC 6 AM1)*

INCITS 470-2011, *Information technology - Fibre Channel - Framing and Signaling Interface - 3 (FC-FS-3)*

INCITS 477-2011, *Information technology - Fibre Channel - Link Services - 2 (FC-LS-2)*

INCITS 481-2011, *Information technology - Fibre Channel Protocol for SCSI - 4 (FCP-4)* (planned as ISO/IEC 14776-224)

INCITS 484-2012, *Information technology SCSI Media Changer Commands 3 (SMC-3)*

INCITS 489, *Information technology - SCSI Over PCI Express® (SOP)*

INCITS 492, *Information technology - SAS Protocol Layer - 3 (SPL-3)* (planned as ISO/IEC 14776-263)

INCITS 496-2012, *Information technology - Fibre Channel - Security Protocols - 2 (FC-SP-2)*

INCITS 496-2012/AM 1-2015, *Information technology - Fibre Channel - Security Protocols - 2/Amendment 1 - (FC-SP-2/AM1)*

INCITS 515, *Information technology - Small Computer System Interface (SCSI) - SCSI Architecture Model - 5 (SAM-5)* (planned as ISO/IEC 14776-415)

INCITS 516, *Information technology - SCSI Stream Commands - 4 (SSC-4)*

1) SAS-2.1 supersedes SAS-2 but does not contain the information that this standard references normatively in 7.5.13.

INCITS 517, *Information technology - SCSI / ATA Translation - 3 (SAT-3)*

ANSI/IEEE 1394a-2000, *High Performance Serial Bus (supplement to ANSI/IEEE 1394-1995)*

ANSI/IEEE 1619.1-2007, *Standard for Authenticated Encryption with Length Expansion for Storage Devices*

ANSI/IEEE 1667:2009, *Standard Protocol for Authentication in Host Attachments of Transient Storage Devices*

INCITS 4-1986 (R2002), *Information Systems - Coded Character Sets - 7-Bit American National Standard Code for Information Interchange (7-Bit ASCII)*

ISO/IEC 13213, *Information technology - Microprocessor systems - Control and Status Registers (CSR) Architecture for microcomputer buses*

ISO/IEC 646:1991, *Information technology - ISO 7-bit coded character set for information interchange*

ISO/IEC 8859-1:1998, *Information technology - 8-bit single-byte coded graphic character sets - Part 1: Latin alphabet No. 1*

ISO/IEC 8859-2:1999, *Information technology - 8-bit single-byte coded graphic character sets - Part 2: Latin alphabet No. 2*

ISO/IEC 8859-3:1999, *Information technology - 8-bit single-byte coded graphic character sets - Part 3: Latin alphabet No. 3*

ISO/IEC 8859-4:1998, *Information technology - 8-bit single-byte coded graphic character sets - Part 4: Latin alphabet No. 4*

ISO/IEC 8859-5:1999, *Information technology - 8-bit single-byte coded graphic character sets - Part 5: Latin/Cyrillic alphabet*

ISO/IEC 8859-6:1999, *Information technology - 8-bit single-byte coded graphic character sets - Part 6: Latin/Arabic alphabet*

ISO/IEC 8859-7:2003, *Information technology - 8-bit single-byte coded graphic character sets - Part 7: Latin/Greek alphabet*

ISO/IEC 8859-8:1999, *Information technology - 8-bit single-byte coded graphic character sets - Part 8: Latin/Hebrew alphabet*

ISO/IEC 8859-9:1999, *Information technology - 8-bit single-byte coded graphic character sets - Part 9: Latin alphabet No. 5*

ISO/IEC 8859-10:1998, *Information technology - 8-bit single-byte coded graphic character sets - Part 10: Latin alphabet* No. 6

ISO/IEC 10646:2017, *Information technology - Universal Coded Character Set (UCS)*

Video Performance Guarantee Profile 1 Revision 1.0 (VPG1), May 24, 2011

NOTE 1 - For more information about documents published by the CFA, see http://www.compactflash.org/.

RFC 791, *Internet Protocol - DARPA Internet Program - Protocol Specification*

RFC 1321, *The MD5 Message-Digest Algorithm*

RFC 2104, *HMAC: Keyed-Hashing for Message Authentication*

RFC 2279, *UTF-8, a transformation format of ISO 10646*

RFC 2410, *The NULL Encryption Algorithm and Its Use With IPsec*

RFC 2437, *PKCS #1: RSA Cryptography Specifications Version 2.0*

RFC 2616, *Hypertext Transfer Protocol - HTTP/1.1*

RFC 3526, *More Modular Exponential (MODP) Diffie-Hellman groups for Internet Key Exchange (IKE)*

RFC 3566, *The AES-XCBC-MAC-96 Algorithm and Its Use With IPsec*

RFC 3602, *The AES-CBC Cipher Algorithm and Its Use with IPsec*

RFC 3766, *Determining Strengths For Public Keys Used For Exchanging Symmetric Keys*

RFC 4086, *Randomness Requirements for Security*

RFC 4106, *The Use of Galois/Counter Mode (GCM) in IPsec Encapsulating Security Payload*

RFC 4291, *IP Version 6 Addressing Architecture*

RFC 4303, *IP Encapsulating Security Payload (ESP)*

RFC 4306, *Internet Key Exchange (IKEv2) Protocol*

RFC 4309, *Using Advanced Encryption Standard (AES) CCM Mode with IPsec Encapsulating Security Payload*

RFC 4434, *The AES-XCBC-PRF-128 Algorithm for the Internet Key Exchange Protocol (IKE)*

RFC 4595, *Use of IKEv2 in the Fibre Channel Security Association Management Protocol*

RFC 4718, *IKEv2 Clarifications and Implementation Guidelines*

RFC 4753, *ECP Groups for IKE and IKEv2*

RFC Errata for RFC 4753, *http://rfc-editor.org/errata_search.php?rfc=4753*

RFC 4754, *IKE and IKEv2 Authentication Using the Elliptic Curve Digital Signature Algorithm (ECDSA)*

RFC 4868, *Using HMAC-SHA-256, HMAC-SHA-384, and HMAC-SHA-512 with IPsec*

RFC 5280, *Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile*

RFC 5282, *Using Authenticated Encryption Algorithms with the Encrypted Payload of the Internet Key Exchange version 2 (IKEv2) Protocol*

RFC 5755, *An Internet Attribute Certificate Profile for Authorization*

RFC 6818, *Updates to the Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile*

RFC 7143, *iSCSI Protocol (Consolidated)*

RFC 7144, *Internet Small Computer Systems Interface (iSCSI) SCSI Features Update*

NOTE 2 - For more information on the RFCs and standards published by the Internet Engineering Task Force (IETF), see http://www.ietf.org/.

NIST SP (Special Publication) 800-38D, *Recommendation for Block Cipher Modes of Operation: Galois/Counter (GCM) Mode for Confidentiality and Authentication and GMAC*

NIST SP 800-90 A, *Recommendation for Random Number Generation Using Deterministic Random Bit Generators*

FIPS 140-2, *Security Requirements for Cryptographic Modules*

FIPS 140-2, *Annex C: Approved Random Number Generators*

FIPS 140-3, *Security Requirements for Cryptographic Modules (in development)*

FIPS 180-4, *Secure Hash Standard*

FIPS 198-1, *The Keyed-Hash Message Authentication Code (HMAC)*

NOTE 3 - For more information on NIST and FIPS standards published by the National Institute of Standards and Technology, see http://csrc.nist.gov/publications/.

Universal Serial Bus 3.0 Specification Revision 1.0 (USB-3), November 12, 2008

NOTE 4 - For more information on publications by the USB Implementers Forum, see http://www.usb.org/.

3 Terms, definitions, symbols, abbreviations, and conventions

3.1 Terms and definitions

3.1.1
access control list
ACL
data used by a SCSI target device to configure access rights for initiator ports according to the access controls state of the SCSI target device

Note 1 to entry: See 8.3.1.3.

3.1.2
access control list entry
ACE
one entry in the access control list (see 3.1.1)

3.1.3
access controls
SCSI target device feature that restricts initiator port access to specific logical units

Note 1 to entry: Information is modified about logical units in the parameter data of the INQUIRY command (see 6.6) and REPORT LUNS command (see 6.33).

Note 2 to entry: See 8.3.1.

3.1.4
access controls coordinator
entity within a SCSI target device that coordinates the management and enforcement of access controls (see 8.3.1) for all logical units within the SCSI target device

Note 1 to entry: The access controls coordinator is always addressable through the ACCESS CONTROLS well known logical unit (see 8.3) and LUN 0.

3.1.5
active power condition
power condition in which a device server is capable of completing the processing of all its supported commands

Note 1 to entry: See 5.11.4.

3.1.6
additional authenticated data
AAD
bytes of data input to a combined mode encryption algorithm (e.g., AES-GCM) as part of integrity checking computations

EXAMPLE: The IKEv2-SCSI Encrypted payload (see 7.7.3.5.11) uses AAD.

3.1.7
additional sense code
combination of the ADDITIONAL SENSE CODE field and the ADDITIONAL SENSE CODE QUALIFIER field in sense data (see 4.5)

3.1.8
alias list
list of alias values (see 3.1.9) and their associated designations (see 3.1.41) maintained by the device server and managed by the CHANGE ALIASES command (see 6.2) and REPORT ALIASES command (see 6.30)

3.1.9
alias value
numeric value associated to a designation (see 3.1.41) in the alias list (see 3.1.8)

Note 1 to entry: Used in command or parameter data to reference a SCSI target device or SCSI target port.

Note 2 to entry: See 6.2.2.

3.1.10
application client
object that is the source of SCSI commands

Note 1 to entry: See SAM-5.

3.1.11
ASCII format list log parameter
log parameter that contains ASCII data in a list log parameter format

Note 1 to entry: See 7.3.2.2.2.4.

3.1.12
attribute
single unit of MAM (see 3.1.88) information

3.1.13
auto contingent allegiance
ACA
task set condition established following the return of CHECK CONDITION status if the NACA bit is set to one in the CONTROL byte

Note 1 to entry: See SAM-5.

3.1.14
background function
background scan operation (see SBC-3) or device specific background function (see 5.3)

3.1.15
binary format list log parameter
log parameter that contains binary data in a list log parameter format

Note 1 to entry: See 7.3.2.2.2.5.

3.1.16
blocked command
command that is in the blocked state (see SAM-5)

Note 1 to entry: Commands become blocked if an ACA condition occurs and the blocked state ends when the ACA condition is cleared.

3.1.17
bounded data counter log parameter
log parameter that contains one saturating counter (see 3.1.134) value for which the establishment of thresholds is allowed

Note 1 to entry: Effects on other log parameters in the same log page are determined by the format and linking field in the parameter control byte.

Note 2 to entry: See 7.3.2.2.2.2.

3.1.18
byte
B
sequence of eight contiguous bits considered as a unit

3.1.19
cache memory
temporary and often volatile data storage area outside the area accessible by application clients

Note 1 to entry: Cache memory may contain a subset of the data stored in the non-volatile data storage area.

3.1.20
CbCS capability
descriptor that specifies access to a logical unit or volume (see SSC-4) for specific commands

Note 1 to entry: A CbCS capability is a component of a CbCS credential (see 3.1.20) and a CbCS extension descriptor (see 5.13.6.7.16).

Note 2 to entry: See 5.13.6.7.13.

3.1.21
CbCS capability key
shared key (see 3.1.159) that is cryptographically associated with a specific CbCS capability (see 3.1.20)

3.1.22
CbCS credential
combination of a CbCS capability (see 3.1.20) and a CbCS capability key (see 3.1.21)

Note 1 to entry: A CbCS credential is returned by a RECEIVE CREDENTIAL command (see 6.26).

Note 2 to entry: See 5.13.6.7.12.

3.1.23
CbCS Management Application Client class
CbCS (see 5.13.6.7) class whose objects manage, or an object that manages CbCS shared keys (see 5.13.6.7.11)

Note 1 to entry: See 5.13.6.7.4.

3.1.24
CbCS Management Device Server class
CbCS (see 5.13.6.7) class whose objects prepare, or an object that prepares CbCS credentials (see 5.13.6.7.12)

Note 1 to entry: A CbCS credential is prepared in response to a RECEIVE CREDENTIAL command (see 6.26) request from a secure CDB originator (see 5.13.6.7.5).

Note 2 to entry: See 5.13.6.7.3.

3.1.25
CbCS master key
shared key (see 3.1.159) from which CbCS working keys (see 3.1.28) are derived that is composed of an authentication key component and a generation key component

Note 1 to entry: See 5.13.6.7.11.

3.1.26
CbCS security token
random nonce (see 3.1.119) that is chosen by the enforcement manager (see 5.13.6.7.7) for a given I_T nexus and returned to a secure CDB originator (see 5.13.6.7.5)

Note 1 to entry: See 5.13.6.7.10.

3.1.27
CbCS working keys
shared keys (see 3.1.159) that are used to cryptographically associate a CbCS capability (see 3.1.20) with a CbCS capability key (see 3.1.21)

Note 1 to entry: See 5.13.6.7.11.

3.1.28
code set enumeration
coded value used in one field to indicate the format of data in other fields or descriptors

Note 1 to entry: See 4.3.3.

3.1.29
command
request describing a unit of work to be performed by a device server

Note 1 to entry: See SAM-5.

3.1.30
command descriptor block
CDB
structure used to communicate commands from an application client to a device server (see 4.2)

Note 1 to entry: A CDB may have a fixed length of up to 16 bytes or a variable length of between 12 and 260 bytes.

3.1.31
command standard
SCSI standard that defines the model, commands, and parameter data for a device type

EXAMPLE: SBC-3, SSC-4, SMC-3, MMC-6, and SES-2 are examples of command standards.

3.1.32
company_id
synonym for OUI (see 3.1.99)

3.1.33
copy manager
object (see SAM-5) that processes third-party copy commands (see 5.16.3) and manages copy operations (see 5.16.4.3)

3.1.34
copy operation
foreground or background operation that results from the processing of a third-party copy command

Note 1 to entry: See 5.16.4.3.

3.1.35
copy source or copy destination
CSCD
source or destination of a copy operation (see 6.4.6) performed by an EXTENDED COPY(LID4) command (see 6.4) or an EXTENDED COPY(LID1) command (see 6.5)

3.1.36
cryptographic command sequence
CCS
defined sequence of SECURITY PROTOCOL IN commands (see 6.40) and SECURITY PROTOCOL OUT commands (see 6.41) that realize the cryptographic protocol of a specified security operation

EXAMPLE: The SECURITY PROTOCOL IN commands and SECURITY PROTOCOL OUT commands in the Key Exchange step and Authentication step, if any, of an IKEv2-SCSI (see 3.1.67) SA creation transaction (see 3.1.128).

3.1.37
data counter log parameter
bounded data counter log parameter (see 7.3.2.2.2.2) or unbounded data counter log parameter (see 7.3.2.2.2.3)

3.1.38
Data-In Buffer
buffer specified by the application client to receive data from the device server during the processing of a command

Note 1 to entry: See SAM-5.

3.1.39
Data-Out Buffer
buffer specified by the application client to supply data that is sent from the application client to the device server during the processing of a command

Note 1 to entry: See SAM-5.

3.1.40
deferred error
CHECK CONDITION status and sense data that is returned as the result of an error or exception condition that occurred during processing of a previous command for which GOOD status or CONDITION MET status has already been returned

Note 1 to entry: See 4.5.7.

3.1.41
designation
distinguishing name, identifier, or title

3.1.42
designation
<access controls> name and optional identifier information that specifies a SCSI target device or SCSI target port for association with an alias value (see 3.1.9) in the alias list (see 3.1.8) as described in 6.2.2

3.1.43
device server
object within a logical unit that processes SCSI commands according to the rules of command management

Note 1 to entry: See SAM-5.

3.1.44
device service request
request, submitted by an application client, conveying a SCSI command to a device server

Note 1 to entry: See SAM-5.

3.1.45
device service response
response returned to an application client by a device server on completion of a SCSI command

Note 1 to entry: See SAM-5.

3.1.46
device type
type of peripheral device (i.e., device model) implemented by the device server and indicated to the application client by the contents of the PERIPHERAL DEVICE TYPE field in the standard INQUIRY data (see 6.6.2)

3.1.47
element
addressable physical component of a medium changer SCSI device that may serve as the location of a removable unit of data storage medium

Note 1 to entry: See SMC-3.

3.1.48
enabled command
command that is being processed by the device server and is progressing towards completion

Note 1 to entry: See SAM-4.

3.1.49
Encapsulating Security Payload for SCSI
ESP-SCSI
method for transferring encrypted and/or integrity checked parameter data in Data-In Buffers and/or Data-Out Buffers based on Encapsulating Security Payload (see RFC 4303)

Note 1 to entry: See 5.13.7.

3.1.50
Enforcement Manager class
command security (see 5.13.6) class whose objects are, or an object that is consulted by the Secure CDB Processor class (see 3.1.147) or Secure CDB Processor class objects to determine if the processing of a command is permitted or prohibited

Note 1 to entry: The Enforcement Manager class is equivalent to the Policy Enforcement Point in the policy model defined by RFC 2753 and similar policy-based authorization standards.

Note 2 to entry: See 5.13.6.4.

3.1.51
Error history I_T nexus
I_T nexus for which the device server has reserved access to the error history snapshot (see 3.1.52)

3.1.52
Error history snapshot
contents of the error history at a specific point in time

Note 1 to entry: See 5.5.2.

3.1.53
Extended CDB
XCDB
CDB that contains another CDB and additional information to support extra processing (e.g., security features)

Note 1 to entry: See 4.2.4.

3.1.54
EUI-48
48-bit identifier that is globally unique

Note 1 to entry: The IEEE maintains a tutorial describing EUI-48 at http://standards.ieee.org/develop/regauth/tut/eui.pdf.

3.1.55
EUI-64
64-bit identifier that is globally unique

Note 1 to entry: The IEEE maintains a tutorial describing EUI-64 at http://standards.ieee.org/develop/regauth/tut/eui.pdf.

3.1.56
faulted I_T nexus
I_T nexus on which a CHECK CONDITION status was returned that resulted in the establishment of an ACA

Note 1 to entry: The faulted I_T nexus condition is cleared when the ACA condition is cleared.

Note 2 to entry: See SAM-5.

3.1.57
field
group of one or more contiguous bits, a part of a larger structure such as a CDB (see 3.1.30) or sense data (see 3.1.155)

3.1.58
hard reset
condition resulting from the events defined by SAM-5 in which the SCSI device performs the hard reset operations described in SAM-5, this standard, and the applicable command standards

3.1.59
hashed message authentication code
HMAC
type of message authentication code that is calculated using the cryptographic hash function as defined in FIPS 198-1 in combination with a secret key

3.1.60
host
SCSI device with the characteristics of a primary computing device

Note 1 to entry: A host is typically a personal computer, workstation, server, minicomputer, mainframe computer, or auxiliary computing device.

Note 2 to entry: A host includes one or more SCSI initiator devices.

3.1.61
IEEE company_id
synonym for OUI (see 3.1.99)

3.1.62
I_T nexus
nexus between a SCSI initiator port and a SCSI target port

Note 1 to entry: See SAM-5.

3.1.63
I_T nexus loss
condition resulting from the events defined by SAM-5

Note 1 to entry: In response to this condition, the SCSI device performs the I_T nexus loss operations described in SAM-5, this standard, and other applicable standards.

3.1.64
I_T_L nexus
nexus between a SCSI initiator port, a SCSI target port, and a logical unit

Note 1 to entry: See SAM-5.

3.1.65
I_T_L_Q nexus transaction
information transferred between SCSI ports in a single data structure with defined boundaries (e.g., an information unit)

3.1.66
idle power condition
power condition in which a device server is capable of completing the processing of all its supported commands, except those that require the logical unit be in the active power condition (see 3.1.5) to be capable of completing the command with GOOD status

Note 1 to entry: See 5.11.5.

3.1.67
IKEv2-SCSI
Internet Key Exchange protocol version 2 for SCSI

Note 1 to entry: See 5.13.4.

3.1.68
IKEv2-SCSI keys
shared keys (see 3.1.159) used to provide security for IKEv2-SCSI operations (e.g., creation and deletion of the SA)

Note 1 to entry: IKEv2-SCSI keys used after SA creation is complete are maintained in the MGMT_DATA SA parameter (see 3.1.131).

Note 2 to entry: The shared key names used by IKEv2-SCSI are listed in 5.13.4.4.

3.1.69
IKEv2-SCSI CCS
CCS (see 3.1.36) that is the Key Exchange step and Authentication step, if any, of an IKEv2-SCSI (see 3.1.67) SA creation transaction (see 3.1.128)

3.1.70
information unit
IU
delimited and sequenced set of information in a format appropriate for transport by the service delivery subsystem (e.g., a CDB for a specific SCSI command)

3.1.71
initiator port
synonymous with SCSI initiator port (see 3.1.139)

3.1.72
initiator port identifier
value by which a SCSI initiator port (see 3.1.139) is referenced within a SCSI domain (see SAM-5)

3.1.73
initiator port name
name (see 3.1.91) of a SCSI initiator port

3.1.74
integrity check value
value used to cryptographically validate the integrity of a specified set of bytes that contain specified data based on a shared key (see 3.1.159)

3.1.75
internet assigned numbers authority
IANA
service that assigns and manages registries of code values (i.e., code points) for the IETF (see http://www.ietf.org) and other users of the public internet (see http://www.iana.org)

EXAMPLE: The IANA maintains code values used by Internet Key Exchange version 2 (IKEv2) at https://www.iana.org/assignments/ikev2-parameters/ikev2-parameters.xhtml.

3.1.76
Internet protocol domain name
name of a computer or hierarchy of computers within the domain name system defined by the IETF (see RFC 1035 and RFC 1591)

3.1.77
Internet protocol number
coded value assigned to identify protocols that layer on the Internet protocol (see RFC 791)

EXAMPLE: The Internet protocol number assigned to the transmission control protocol (TCP, see RFC 793) is six.

Note 1 to entry: The IANA maintains a list of Internet protocol number assignments at https://www.iana.org/assignments/protocol-numbers/protocol-numbers.xhtml.

3.1.78
key derivation function
KDF
algorithm that is used to derive cryptographic keying material from a shared secret and other information

3.1.79
least significant bit
LSB
<in a binary code> bit or bit position with the smallest numerical weighting in a group of bits that, when taken as a whole, represent a numerical value

EXAMPLE – In the number 0001b, the bit that is set to one.

3.1.80
left-aligned
type of field containing ASCII data in which unused bytes are placed at the end of the field (i.e., highest offset) and are filled with ASCII space (20h) characters

Note 1 to entry: See 4.3.1.

3.1.81
logical unit
externally addressable entity within a SCSI target device that implements a SCSI device model and contains a device server

Note 1 to entry: See SAM-5.

3.1.82
logical unit access control descriptor
LUACD
structure within an ACE (see 3.1.2) that identifies a logical unit to which access is allowed and specifies the LUN by which the logical unit is to be accessed

Note 1 to entry: See 8.3.1.3.3.

3.1.83
logical unit inventory
list of the logical unit numbers reported by a REPORT LUNS command (see 6.33)

3.1.84
logical unit number
LUN
encoded 64-bit identifier for a logical unit

Note 1 to entry: See SAM-5.

3.1.85
logical unit reset
condition resulting from the events defined by SAM-5

Note 1 to entry: In response to this condition, the logical unit performs the logical unit reset operations described in SAM-5, this standard, and other applicable standards.

3.1.86
low power condition
any power condition other than the active power condition (see 5.11.4)

3.1.87
medium
physical entity that stores data in a nonvolatile manner (i.e., retained through a power cycle) in accordance with commands processed by the device server

3.1.88
medium auxiliary memory
MAM
auxiliary memory residing in a volume (e.g., a tape cartridge) that is accessible to the device server

Note 1 to entry: See 7.4.

3.1.89
media changer
device that mechanizes the movement of volumes to and from the SCSI device that records on or reads from the media

Note 1 to entry: See SMC-3.

3.1.90
most significant bit
MSB
<in a binary code> bit or bit position with the largest numerical weighting in a group of bits that, when taken as a whole, represent a numerical value

EXAMPLE: In the number 1000b, the bit that is set to one.

3.1.91
name
label of an object that is unique within a specified context and should never change

EXAMPLE: The term name and world wide identifier (WWID) may be interchangeable.

3.1.92
network address authority
NAA
field within a name (see 3.1.91) that specifies the format and length of that name

Note 1 to entry: See 7.8.6.6 and FC-FS-3.

3.1.93
nexus
relationship between two SCSI devices, and the SCSI initiator port and SCSI target port objects within those SCSI devices

Note 1 to entry: See SAM-5.

3.1.94
non-volatile cache memory
cache memory (see 3.1.19) that retains data through any power cycle

3.1.95
nonce
value that is used one and only one time to provide uniqueness to a value (e.g., a secure cryptographic key) in whose derivation it participates or to uniquely identify a single instance of something (e.g., a timestamp) exchanged between an application client and a device server

3.1.96
null-padded
type of field in which unused bytes are placed at the end of the field (i.e., highest offset) and are filled with ASCII null (00h) characters

Note 1 to entry: See 4.3.2.

3.1.97
null-terminated
type of field in which the last used byte (i.e., highest offset) is required to contain an ASCII null (00h) character

Note 1 to entry: See 4.3.2.

3.1.98
one
logical true condition of a variable

3.1.99
organizationally unique identifier
OUI
numeric identifier that is assigned by the IEEE such that no assigned identifiers are identical

Note 1 to entry: OUI is equivalent to company_id or IEEE company_id.

Note 2 to entry: The IEEE prefers OUI for EUI-48 identifiers (see 3.1.54) and company_id for EUI-64 identifiers (see 3.1.55). However, the numeric identifier is called an OUI when that identifier is assigned by the IEEE.

Note 3 to entry: The IEEE maintains a tutorial describing the OUI at http://standards.ieee.org/develop/regauth/tut/eui.pdf.

3.1.100
page
regular parameter structure or format used by several commands and identified with a value known as a page code

3.1.101
peripheral device
object that connects to a logical unit

3.1.102
peripheral device identifying information
value associated with a peripheral device accessed via the REPORT IDENTIFYING INFORMATION command (see 6.32) and SET IDENTIFYING INFORMATION command (see 6.43)

Note 1 to entry: See 5.6.

3.1.103
peripheral device text identifying information
UTF-8 (see 3.1.186) format string associated with a peripheral device accessed via the REPORT IDENTIFYING INFORMATION command (see 6.32) and SET IDENTIFYING INFORMATION command (see 6.43)

Note 1 to entry: See 5.6.

3.1.104
peripheral device type
synonym for device type (see 3.1.46)

3.1.105
persist through power loss
capability associated with some features that allows an application client to request that a device server maintain information regarding that feature across power failures

3.1.106
persistent reservation holder
I_T nexus(es) that are allowed to release or change a persistent reservation without preempting it

Note 1 to entry: See 5.12.10.

3.1.107
plaintext
unencrypted bytes

3.1.108
power cycle
power being removed from and later applied to a SCSI device

3.1.109
power on
condition resulting from the events defined by SAM-5 in which the SCSI device performs the power on operations described in SAM-5, this standard, and the applicable command standards

3.1.110
pre-shared key
shared key (see 3.1.159) that is established by a method outside the scope of this standard prior to the initiation of the function that requires its use

Note 1 to entry: The requirements on pre-shared keys that are particular to this standard are described in 5.13.4.3.2.

Note 2 to entry: Some of the RFCs referenced by this standard use the name pre-shared secret for the same information that this standard calls a pre-shared key.

3.1.111
primary target port asymmetric access state
characteristic that defines the behavior of a target port and the allowable command set for a logical unit when commands and task management functions are routed through the target port maintaining that state

Note 1 to entry: See 5.15.2.1.

3.1.112
primary target port group
set of target ports that are in the same primary target port asymmetric access state (see 3.1.111) at all times

Note 1 to entry: See 5.15.2.1.

3.1.113
primary target port group asymmetric access state
primary target port asymmetric access state (see 3.1.111) common to the set of target ports in a primary target port group (see 3.1.112)

Note 1 to entry: See 5.15.2.1.

3.1.114
protection information
one or more fields appended to each logical block that contain a CRC and device type specific information

Note 1 to entry: See SBC-3 and SSC-4.

3.1.115
protocol identifier
coded value used in various fields to identify the SCSI transport protocol to which other fields apply

Note 1 to entry: See 7.6.1.

3.1.116
protocol specific
requirement that is defined by a SCSI transport protocol standard (see 3.1.143)

Note 1 to entry: See SAM-5.

3.1.117
protocol standard
synonymous with SCSI transport protocol standard (see 3.1.143)

3.1.118
proxy token
identifier for a logical unit that may be generated to share access to that logical unit in the presence of access controls

Note 1 to entry: See 8.3.1.6.2.

3.1.119
random nonce
secure random number (see 4.4) that has a negligible chance of repeating and whose uses include providing significant uniqueness and randomness in cryptographic calculations

3.1.120
request for comment
RFC
name given to standards developed by the Internet Engineering Task Force

3.1.121
registered
condition that exists for an I_T nexus following the successful completion of a PERSISTENT RESERVE OUT command with a REGISTER service action, REGISTER AND IGNORE EXISTING KEY service action (see 5.12.7), or REGISTER AND MOVE service action (see 5.12.8) and lasting until the registration is removed (see 5.12.11)

3.1.122
registrant
I_T nexus that is registered (see 3.1.121)

3.1.123
right-aligned
type of field containing ASCII data in which unused bytes are placed at the start of the field (i.e., lowest offset) and are filled with ASCII space (20h) characters

Note 1 to entry: See 4.3.1.

3.1.124
relative port identifier
identifier for a SCSI port (see 3.1.140) that is unique within a SCSI device (see 3.1.135)

Note 1 to entry: Application clients may use the SCSI Ports VPD page (see 7.8.14) to determine relative port identifier values.

Note 2 to entry: See SAM-5.

3.1.125
relative initiator port identifier
relative port identifier (see 3.1.124) for a SCSI initiator port (see 3.1.139)

3.1.126
relative target port identifier
relative port identifier (see 3.1.124) for a SCSI target port (see 3.1.142)

3.1.127
ROD token
kind of token (see 3.1.178) that is a representation of data

Note 1 to entry: See 5.16.6.

3.1.128
SA creation transaction
any sequence of SECURITY PROTOCOL IN commands (see 6.40) and SECURITY PROTOCOL OUT commands (see 6.41) that is used to create an SA between an application client and device server

Note 1 to entry: A CCS (see 3.1.36) is a kind of SA creation transaction.

3.1.129
SA generation
computation and initialization of the SA parameter values (see 3.1.131) required to create an SA (see 3.1.152)

Note 1 to entry: SA generation is performed separately by the application client and device server after all SCSI commands required to create an SA have been performed without error

Note 2 to entry: See 5.13.4.9.

3.1.130
SA keys
shared keys (see 3.1.159) that are maintained in the USAGE_DATA SA parameter (see 3.1.131)

Note 1 to entry: SA keys are used to provide security for the operations that use the SA (e.g., encryption of command parameter data).

Note 2 to entry: The shared key names used by IKEv2-SCSI are listed in 5.13.4.4.

3.1.131
SA parameters
parameters stored by both an application client and a device server

Note 1 to entry: SA parameters are associated with one SA (see 3.1.152) and identified by a pair of SAIs (see 3.1.153).

Note 2 to entry: See 5.13.2.2..

3.1.132
SA participant
application client or device server that participates in the creation or use of an SA (see 3.1.152)

3.1.133
salt bytes
sequence of bytes whose contents are unpredictable in advance of their use

Note 1 to entry: Salt bytes are kept secret prior to their use, and disclosed during their use (e.g., bytes added to a cryptographic operation to increase the entropy in the result (see RFC 4106)).

3.1.134
saturating counter
counter that remains at its maximum value after reaching its maximum value

3.1.135
SCSI device
device that contains one or more SCSI ports that are each connected to a service delivery subsystem and supports a SCSI application protocol

Note 1 to entry: See SAM-5.

3.1.136
SCSI device name
name (see 3.1.91) of a SCSI device that is world wide unique within the protocol of a SCSI domain (see 3.1.137) in which the SCSI device has SCSI ports (see 3.1.140)

Note 1 to entry: The SCSI device name may be made available to other SCSI devices or SCSI ports in protocol specific ways.

Note 2 to entry: See SAM-5.

3.1.137
SCSI domain
interconnection of two or more SCSI devices and a service delivery subsystem

3.1.138
SCSI initiator device
SCSI device (see 3.1.135) containing application clients (see 3.1.10) and SCSI initiator ports (see 3.1.139)

Note 1 to entry: SCSI initiator devices originate device service and task management requests (see SAM-5) to be processed by a SCSI target device (see 3.1.141) and receive device service and task management responses from SCSI target devices.

Note 2 to entry: See SAM-5.

3.1.139
SCSI initiator port
SCSI initiator device (see SAM-5) object that acts as the connection between application clients and a service delivery subsystem through which requests and responses are routed

Note 1 to entry: See SAM-5.

3.1.140
SCSI port
SCSI initiator port (see 3.1.139) or SCSI target port (see 3.1.142)

3.1.141
SCSI target device
SCSI device (see 3.1.135) containing logical units (see 3.1.81) and SCSI target ports (see 3.1.142)

Note 1 to entry: SCSI target devices receive device service and task management requests (see SAM-5) for processing and send device service and task management responses to SCSI initiator devices.

Note 2 to entry: See SAM-5.

3.1.142
SCSI target port
target port
SCSI target device (see SAM-5) object that acts as the connection between device servers and task managers and a service delivery subsystem through which requests and responses are routed

Note 1 to entry: See SAM-5.

3.1.143
SCSI transport protocol standard
SCSI standard that defines a SCSI transport protocol (e.g., FCP-4, SPL-3, SRP, or SBP-3)

3.1.144
secondary target port asymmetric access state
characteristic indicating a condition that affects the way in which an individual target port participates in its assigned primary target port group (see 3.1.112)

Note 1 to entry: See 5.15.2.1.

3.1.145
secondary target port group
one or more target ports that are in the same secondary target port asymmetric access state (see 3.1.144)

Note 1 to entry: See 5.15.2.1.

3.1.146
Secure CDB Originator class
command security (see 5.13.6) class whose objects are, or an object that is an application client that originates secure CDBs

Note 1 to entry: See 5.13.6.2.

3.1.147
Secure CDB Processor class
command security (see 5.13.6) class whose objects are, or an object that is a device server that processes secure CDBs

Note 1 to entry: See 5.13.6.3.

3.1.148
Secure Content Storage Association
SCSA
organization that develops specifications for protecting digital media content

Note 1 to entry: See http://www.vidity.com.

3.1.149
Secure Digital Card Association
SD Card
organization that establishes and maintains standards for secure digital memory card applications

Note 1 to entry: See http://www.sdcard.org/.

3.1.150
secure hash algorithm
SHA
secure hash algorithm (e.g., SHA-1) defined in FIPS 180-4, Secure Hash Standard (SHS)

3.1.151
secure random number
random number that is generated in ways that protect it from security attacks

Note 1 to entry: See 4.4.

3.1.152
security association
SA
relationship and associated security processing between an application client and a device server that is used to apply security functions (e.g., integrity checking, encryption) to data that is transferred in either direction

Note 1 to entry: See 5.13.2.

3.1.153
security association index
SAI
number representing the parameters for an SA as stored internally by the application client or device server

Note 1 to entry: In other security models, this value is called the security parameters index (SPI).

Note 2 to entry: See 5.13.2.

3.1.154
Security Manager class
command security (see 5.13.6) class whose objects maintain, or an object that maintains information about which secure CDBs may be originated by which secure CDB originator (see 3.1.146) application clients to which secure CDB processor (see 3.1.147) device servers

Note 1 to entry: The Security Manager responds to requests to allow the origination of secure CDBs from secure CDB originator application clients, and updates secure CDB permissions and prohibitions in appropriate enforcement manager (see 3.1.50) members.

Note 2 to entry: The Security Manager is equivalent to the Policy Decision Point in the policy model defined by RFC 2753 and similar policy-based authorization standards.

Note 3 to entry: See 5.13.6.5.

3.1.155
sense data
data describing an error, exceptional condition, or completion information

Note 1 to entry: A device server delivers sense data to an application client in the same I_T_L_Q nexus transaction (see 3.1.65) as the status or as parameter data in response to a REQUEST SENSE command (see 6.39).

Note 2 to entry: See 4.5.

3.1.156
sense key
contents of the SENSE KEY field in sense data (see 4.5)

3.1.157
service action
extension of the operation code (see 4.2.5.1) used to define a command

Note 1 to entry: See 4.2.5.2.

3.1.158
service delivery subsystem
that part of a SCSI domain that transmits service requests to a logical unit or SCSI target device and returns logical unit or SCSI target device responses to a SCSI initiator device

Note 1 to entry: See SAM-5.

3.1.159
shared key
SK
cryptographically protected secret (e.g., with more randomness (see RFC 4089) than a password) that is known only to a defined and limited set of entities (e.g., one application client and one device server)

Note 1 to entry: The shared key names used by IKEv2-SCSI are listed in 5.13.4.4.

3.1.160
SK_ai
shared key (see 3.1.159) used for integrity checking data in a Data-Out Buffer (e.g., integrity checking the Encrypted payload in an IKEv2-SCSI Authentication step SECURITY PROTOCOL OUT command (see 5.13.4.7.2))

3.1.161
SK_ar
shared key (see 3.1.159) used for integrity checking data in a Data-In Buffer (e.g., integrity checking the Encrypted payload in an IKEv2-SCSI Authentication step SECURITY PROTOCOL IN command (see 5.13.4.7.3))

3.1.162
SK_d
shared key (see 3.1.159) KDF input material use to generate the shared keys stored in the KEYMAT SA Parameter (see 3.1.131)

3.1.163
SK_ei
shared key (see 3.1.159) used for encrypting data in a Data-Out Buffer (e.g., encrypting the Encrypted payload in an IKEv2-SCSI Authentication step SECURITY PROTOCOL OUT command (see 5.13.4.7.2))

3.1.164
SK_er
shared key (see 3.1.159) used for encrypting data in a Data-In Buffer (e.g., encrypting the Encrypted payload in an IKEv2-SCSI Authentication step SECURITY PROTOCOL IN command (see 5.13.4.7.3))

3.1.165
SK_pi
shared key (see 3.1.159) used to construct the IKEv2-SCSI Data-Out Buffer Authentication payload (see 5.13.4.7.2)

3.1.166
SK_pr
shared key (see 3.1.159) used to construct the IKEv2-SCSI Data-In Buffer Authentication payload (see 5.13.4.7.3)

3.1.167
standby power condition
power condition in which a device server is capable of completing the processing of all its supported commands, except those that require the logical unit be in the idle power condition (see 3.1.66) or active power condition (see 3.1.5) to be capable of completing the command with GOOD status

Note 1 to entry: See 5.11.6.

3.1.168
status
one byte of response information that contains a coded value defined in SAM-5, transferred from a device server to an application client upon completion of each command

Note 1 to entry: See SAM-5.

3.1.169
storage networking industry association
SNIA
organization that develops and promotes standards for storage management and other storage-related functions

Note 1 to entry: See http://www.snia.org/.

3.1.170
system
one or more SCSI domains operating as a single configuration

3.1.171
target port asymmetric access state
primary target port asymmetric access state (see 3.1.111) or secondary target port asymmetric access state (see 3.1.144)

3.1.172
target port group
primary target port group (see 3.1.112) or secondary target port group (see 3.1.145)

3.1.173
target port identifier
value by which a SCSI target port (see 3.1.142) is referenced within a SCSI domain (see SAM-5)

3.1.174
target port name
name (see 3.1.91) of a SCSI target port

3.1.175
task set
group of commands within a logical unit, whose interaction is dependent on the task management (i.e., queuing) and ACA rules

Note 1 to entry: The Control mode page (see 7.5.8) defines management capabilities for task sets.

Note 2 to entry: See SAM-5.

3.1.176
TCP port numbers
one of the data necessary to establish a TCP connection

Note 1 to entry: TCP port numbers may be assigned to protocols that layer on TCP by IANA. IANA maintains a list of TCP port number assignments at http://www.iana.org/assignments/port-numbers.

3.1.177
third-party command
command sent to one SCSI device requesting that an operation be performed involving two other SCSI devices

EXAMPLE: The EXTENDED COPY command may perform copy operations between two or more SCSI devices none of which are the SCSI device to which the EXTENDED COPY command was sent.

3.1.178
token
representation of a collection of data (e.g., management data, security data)

3.1.179
Trusted Computing Group
TCG
organization that develops and promotes open standards for hardware-enabled trusted computing and security technologies

Note 1 to entry: See https://www.trustedcomputinggroup.org.

3.1.180
unbounded data counter log parameter
log parameter that contains one or more saturating counter values (see 3.1.134) or wrapping counter values (see 3.1.190) that do not affect other log parameters in the same log page and for which the establishment of thresholds is not allowed

Note 1 to entry: See 7.3.2.2.2.3.

3.1.181
unit attention condition
asynchronous status information that a logical unit establishes to report to the initiator ports associated with one or more I_T nexuses

Note 1 to entry: See SAM-5.

3.1.182
universal time
UT
time at longitude zero, colloquially known as Greenwich Mean Time

Note 1 to entry: See http://aa.usno.navy.mil/faq/docs/UT.php.

3.1.183
URI Schemes
syntax specifications for UTF-8 (see 3.1.186) format strings that identify resources (see RFC 3986)

Note 1 to entry: IANA maintains a list of schemes for URI and URL names at: https://www.iana.org/assignments/uri-schemes/uri-schemes.xhtml.

3.1.184
user data
data contained in logical blocks (e.g., see SBC-3) that is not protection information (see 3.1.114)

Note 1 to entry: Each command standard (see 3.1.31) may provide its own definition of user data.

3.1.185
user data segment
contiguous sequence of logical blocks

Note 1 to entry: See SBC-3.

3.1.186
UTF-8
character set that is a transformation format of the character set defined by ISO 10646

Note 1 to entry: See RFC 2279.

3.1.187
volatile cache memory
cache memory (see 3.1.19) that does not retain data through power cycles

3.1.188
well known logical unit
logical unit that only does specific functions in a SCSI target device

Note 1 to entry: Well known logical units allow an application client to issue requests to receive and manage specific information usually relating to a SCSI target device.

Note 2 to entry: See clause 8 and SAM-5.

3.1.189
word
sequence of 16 contiguous bits considered as a unit

3.1.190
wrapping counter
counter that wraps back to zero after reaching its maximum value

3.1.191
zero
logical false condition of a variable

3.1.192
zero-padded
type of field in which unused bytes are placed at the end of the field (i.e., highest offset) and are filled with zeros

Note 1 to entry: See 4.3.2.

3.2 Abbreviations and symbols

3.2.1 Abbreviations

Abbreviations used in this standard:

Abbreviation	Meaning
AAD	Additional Authenticated Data (see 3.1.6)
ACE	Access Control list Entry (see 3.1.2)
ACL	Access Control List (see 3.1.1)
ACA	Auto Contingent Allegiance (see 3.1.13)
ADC-3	Automation/Drive Interface - Commands - 3 (see bibliography)
ADT-2	Automation/Drive Interface - Transport Protocol - 2 (see bibliography)
AES	Advanced Encryption Standards
AES-GCM	Advanced Encryption Standard - Galois Counter Mode (see RFC 4106)
ASC	Additional Sense Code (see 4.5)
ASCII	American Standard Code for Information Interchange
ASCQ	Additional Sense Code Qualifier (see 4.5)
ASN.1	Abstract Syntax Notation One
ATA	AT Attachment (see www.t13.org)
ATAPI	AT Attachment with Packet Interface (see www.t13.org)
CbCS	Capability-based Command Security (see 5.13.6.7)
CCS	Cryptographic Command Sequence (see 3.1.36)
CDB	Command Descriptor Block (see 3.1.30)
CFA	CompactFlash Association
CRC	Cyclic Redundancy Check
CSCD	Copy Source or Copy Destination (see 3.1.35)
CSEC	Communications Security Establishment Canada
D_ID	Destination Identifier (defined in FC-FS-3)
ECP	Elliptic Curve group modulo Prime (see RFC 4753)
ECDSA	Elliptic Curve Digital Signature Algorithm (see RFC 4754)
ESP-SCSI	Encapsulating Security Payload for SCSI (see 3.1.49)
EUI-48	Extended Unique Identifier, a 48-bit globally unique identifier (see 3.1.54)
EUI-64	Extended Unique Identifier, a 64-bit globally unique identifier (see 3.1.55)
FC-FS-3	Fibre Channel Framing and Signaling Interface - 3 (see clause 2)
FC-LS-2	Fibre Channel Link Services - 2 (see clause 2)
FC-SP-2	Fibre Channel Security Protocols - 2 (see clause 2)
FCP-4	Fibre Channel Protocol for SCSI - 4 (see clause 2)
FIPS	Federal Information Processing Standard
HMAC	Hashed Message Authentication Code (see 3.1.59)
HTTP	Hypertext Transfer Protocol (see RFC 2616)
IANA	Internet Assigned Numbers Authority (see 3.1.75)
ID	Identifier or Identification
IEEE	Institute of Electrical and Electronics Engineers
IKEv2	Internet Key Exchange version 2
IP	Internet Protocol
IPv4	Internet Protocol version 4
IPv6	Internet Protocol version 6
iSCSI	Internet SCSI
IU	Information Unit
KDF	Key Derivation Function (see 3.1.78)
LBA	Logical Block Address

Abbreviation	Meaning
LID1	List Identifier is 1 byte (i.e., a third-party copy command with a one-byte LIST IDENTIFIER field)
LID4	List Identifier is 4 bytes (i.e., a third-party copy command with a four-byte LIST IDENTIFIER field)
LSB	Least Significant Bit (see 3.1.79)
LUACD	Logical Unit Access Control Descriptor (see 3.1.82)
LUN	Logical Unit Number (see 3.1.84)
MAC	Message Authentication Code
MAM	Medium Auxiliary Memory (see 3.1.88)
MMC-6	SCSI Multi-Media Commands - 6 and MMC - 6 Amendment 1 (see clause 2)
MODP	Modular exponential (see RFC 3526)
MSB	Most Significant Bit (see 3.1.90)
NAA	Network Address Authority (see 3.1.92)
OCRW	SCSI Specification for Optical Card Reader/Writer
OSD	Object-based Storage Devices Commands
OUI	Organizationally Unique Identifier (see 3.1.99)
PKI	Public Key Infrastructure (see RFC 5280 and RFC 6818)
PRF	Pseudo-Random Function (see RFC 4306)
RAID	Redundant Array of Independent Disks
RBC	SCSI Reduced Block Commands (see bibliography)
RDMA	Remote Direct Memory Access (see SRP)
RFC	Request For Comments (see 3.1.120)
ROD	Representation Of Data (see 5.16.6)
SA	Security Association (see 3.1.152)
SAI	Security Association Index (see 3.1.153)
SAM-2	SCSI Architecture Model - 2 (see bibliography)
SAM-4	SCSI Architecture Model - 4 (see clause 2)
SAM-5	SCSI Architecture Model - 5 (see clause 2)
SAS-2	Serial Attached SCSI - 2 (see clause 2)
SAT-3	SCSI / ATA Translation -3 (see clause 2)
SAUT	SA Usage Type (see 7.7.3.5.14)
SBC-2	SCSI Block Commands - 2 (see clause 2)
SBC-3	SCSI Block Commands - 3 (see clause 2)
SBC-4	SCSI Block Commands - 4
SBP-3	Serial Bus Protocol - 3 (see clause 2)
SCC-2	SCSI Controller Commands - 2 (see clause 2)
SCSA	Secure Content Storage Association (see 3.1.148)
SCSI	Small Computer System Interface
SES-2	SCSI Enclosure Services - 2 (see bibliography)
SES-3	SCSI Enclosure Services - 3 (see bibliography)
SHA-1	Secure Hash Algorithm, 160 bits (see 3.1.150)
SHA-256	Secure Hash Algorithm, 256 bits (see 3.1.150)
SHA-384	Secure Hash Algorithm, 384 bits (see 3.1.150)
SHA-512	Secure Hash Algorithm, 512 bits (see 3.1.150)
SK	Shared Key (see 3.1.159)
SMC-3	SCSI Media Changer Commands -3
SNIA	Storage Networking Industry Association (see 3.1.169)
SPC-2	SCSI Primary Commands - 2 (see clause 2)
SPC-3	SCSI Primary Commands - 3 (see clause 2)
SPC-4	SCSI Primary Commands - 4 (this standard)
SPI-5	SCSI Parallel Interface - 5 (see clause 2)

Abbreviation	Meaning
SPL-2	SAS Protocol Layer - 2 (see bibliography)
SPL-3	SAS Protocol Layer - 3 (see clause 2)
SRP	SCSI RDMA Protocol (see clause 2)
SSC-4	SCSI Stream Commands - 4 (see clause 2)
TCG	Trusted Computing Group (see 3.1.179)
TCP	Transmission Control Protocol (see RFC 793)
UFS	Universal Flash Storage (developed by JEDEC)
UML	Unified Modeling Language (developed by OMG)
URI	Uniform Resource Identifier (see RFC 3986 and 3.1.183)
URL	Uniform Resource Locator (see RFC 3986 and 3.1.183)
UT	Universal time (see 3.1.182)
USB	Universal Serial Bus
USB-3	Universal Serial Bus - 3
VPD	Vital Product Data (see 7.8)
VPG1	Video Performance Guarantee Profile 1
VS	Vendor Specific (see 3.3.12)
XCDB	Extended CDB (see 3.1.53)
ZBC	Zoned Block Commands (see clause 2)

3.2.2 Symbols

Symbols used in this standard:

Symbol	Meaning
||	concatenation
n/a	not applicable
x, xx	any valid value for a bit or field
&	grammatical and
→	segment descriptor transfer data (see 3.7.5)

3.2.3 Mathematical operators

Mathematical operators used in this standard:

Mathematical operator	Meaning
+	plus
–	minus
/	divided by
×	multiplied by
<	less than
≤	less than or equal to
>	greater than
≥	greater than or equal to
≠	not equal to
||	concatenation

3.3 Keywords

3.3.1
invalid
keyword used to describe an illegal or unsupported bit, byte, word, field or code value

Note 1 to entry: Receipt by the device server of an invalid bit, byte, word, field or code value shall be reported as an error.

3.3.2
mandatory
keyword indicating an item that is required to be implemented as defined in this standard

3.3.3
may
keyword indicating flexibility of choice with no implied preference

3.3.4
may not
keyword indicating flexibility of choice with no implied preference

3.3.5
obsolete
keyword indicating that an item was defined in prior SCSI standards but has been removed from this standard

3.3.6
option, optional
keywords that describe features that are not required to be implemented by this standard

Note 1 to entry: If any optional feature defined by this standard is implemented, then it shall be implemented as defined in this standard.

3.3.7
prohibited
keyword used to describe a feature, function, or coded value that is defined in a non-SCSI standard (i.e., a standard that is not a member of the SCSI family of standards) to which this standard makes a normative reference where the use of said feature, function, or coded value is not allowed for implementations of this standard

3.3.8
reserved
keyword referring to bits, bytes, words, fields and code values that are set aside for future standardization

Note 1 to entry: A reserved bit, byte, word or field shall be set to zero, or in accordance with a future extension to this standard.

Note 2 to entry: Recipients are not required to check reserved bits, bytes, words or fields for zero values.

Note 3 to entry: Receipt of reserved code values in defined fields shall be reported as an error.

3.3.9
restricted
keyword referring to bits, bytes, words, and fields that are set aside for other identified standardization purposes

Note 1 to entry: A restricted bit, byte, word, or field shall be treated as a reserved bit, byte, word or field in the context where the restricted designation appears.

3.3.10
shall
keyword indicating a mandatory requirement

Note 1 to entry: Designers are required to implement all such mandatory requirements to ensure interoperability with other products that conform to this standard.

3.3.11
should
keyword indicating flexibility of choice with a strongly preferred alternative

3.3.12
vendor specific (VS)
something (e.g., a bit, field, code value) that is not defined by this standard

Note 1 to entry: Specification of the referenced item is determined by the SCSI device vendor and may be used differently in various implementations.

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 described 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 letters 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.

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

Lists sequenced by lowercase or uppercase letters show no ordering relationship between the listed items.

EXAMPLE 1 – The following list shows no relationship between the colors named:

a) red, specificity one of the following colors:
   A) crimson; or
   B) amber;
b) blue; or
c) green.

Lists sequenced by numbers show an ordering relationship between the listed items.

EXAMPLE 2 – The following list shows the order in which a page is meant to be read:

1) top;
2) middle; and
3) bottom.

Lists are associated with an introductory paragraph or phrase, and are numbered relative to that paragraph or phrase (i.e., all lists begin with an a) or 1) entry).

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 Numeric and character conventions

3.5.1 Numeric conventions

A binary number is represented in this standard by any sequence of digits comprised of only the Arabic numerals 0 and 1 immediately followed by a lower-case b (e.g., 0101b). Underscores or spaces may be included in binary number representations to increase readability or delineate field boundaries (e.g., 0 0101 1010b or 0_0101_1010b).

A hexadecimal number is represented in this standard by any sequence of digits comprised of only the Arabic numerals 0 through 9 and/or the upper-case English letters A through F immediately followed by a lower-case h (e.g., FA23h). Underscores or spaces may be included in hexadecimal number representations to increase readability or delineate field boundaries (e.g., B FD8C FA23h or B_FD8C_FA23h).

A decimal number is represented in this standard by any sequence of digits comprised of only the Arabic numerals 0 through 9 not immediately followed by a lower-case b or lower-case h (e.g., 25).

A range of numeric values is represented in this standard in the form “a to z”, where a is the first value included in the range, all values between a and z are included in the range, and z is the last value included in the range (e.g., the representation “0h to 3h” includes the values 0h, 1h, 2h, and 3h).

This standard uses the following conventions for representing decimal numbers:

a) the decimal separator (i.e., separating the integer and fractional portions of the number) is a period;
b) the thousands separator (i.e., separating groups of three digits in a portion of the number) is a space;
c) the thousands separator is used in both the integer portion and the fraction portion of a number; and
d) the decimal representation for a year is 1999 not 1 999.

Table 1 shows some examples of decimal numbers represented using various conventions.

Table 1 — Numbering conventions examples

French	English	This standard
0,6	0.6	0.6
3,141 592 65	3.14159265	3.141 592 65
1 000	1,000	1 000
1 323 462,95	1,323,462.95	1 323 462.95

A decimal number represented in this standard with an overline over one or more digits following the decimal point is a number where the overlined digits are infinitely repeating (e.g., $666.\overline{6}$ means 666.666 666… or 666 2/3 and $12.\overline{142\ 857}$ means 12.142 857 142 857… or 12 1/7).

3.5.2 Units of measure

This standard represents values using both decimal units of measure and binary units of measure. Values are represented by the following formats:

a) for values based on decimal units of measure:
   1) numerical value (e.g., 100);
   2) space;
   3) prefix symbol and unit:
      1) decimal prefix symbol (e.g., M) (see table 2); and
      2) unit abbreviation (e.g., B);

and

b) for values based on binary units of measure:
   1) numerical value (e.g., 1 024);
   2) space;
   3) prefix symbol and unit:
      1) binary prefix symbol (e.g., Gi) (see table 2); and
      2) unit abbreviation (e.g., b).

Table 2 compares the prefix, symbols, and power of the binary and decimal units.

Table 2 — Comparison of decimal prefixes and binary prefixes

Decimal			Binary		
Prefix name	**Prefix symbol**	**Power (base-10)**	**Prefix name**	**Prefix symbol**	**Power (base-2)**
kilo	k	10^3	kibi	Ki	2^{10}
mega	M	10^6	mebi	Mi	2^{20}
giga	G	10^9	gibi	Gi	2^{30}
tera	T	10^{12}	tebi	Ti	2^{40}
peta	P	10^{15}	pebi	Pi	2^{50}
exa	E	10^{18}	exbi	Ei	2^{60}
zetta	Z	10^{21}	zebi	Zi	2^{70}
yotta	Y	10^{24}	yobi	Yi	2^{80}

3.5.3 Byte encoded character strings conventions

When this standard requires one or more bytes to contain specific encoded characters, the specific characters are enclosed in single quotation marks. The single quotation marks identify the start and end of the characters that are required to be encoded but are not themselves to be encoded. The characters that are to be encoded are shown in the case that is to be encoded.

An ASCII space character (i.e., 20h) may be represented in a string by the character '¬' (e.g., 'SCSI¬device').

The encoded characters and the single quotation marks that enclose them are preceded by text that specifies the character encoding methodology and the number of characters required to be encoded.

EXAMPLE - Using the notation described in this subclause, stating that the eleven ASCII characters 'SCSI device' represent encoded characters is the same as writing out the following sequence of byte values: 53h 43h 53h 49h 20h 64h 65h 76h 69h 63h 65h.

3.6 Bit and byte ordering

This subclause describes the representation of fields in a table that defines the format of a SCSI structure (e.g., the format of a CDB).

If a field consists of more than one bit and contains a single value (e.g., a number), the least significant bit (LSB) is shown on the right and the most significant bit (MSB) is shown on the left (e.g., in a byte, bit 7 is the MSB and is shown on the left, and bit 0 is the LSB and is shown on the right). The MSB and LSB are not labeled if the field consists of 8 or fewer bits.

If a field consists of more than one byte and contains a single value, the byte containing the MSB is stored at the lowest address and the byte containing the LSB is stored at the highest address (i.e., big-endian byte ordering). The MSB and LSB are labeled.

If a field consists of more than one byte and always contains multiple fields each with their own values (e.g., a descriptor), then there is no MSB and LSB of the field itself and thus there are no MSB and LSB labels. Each individual field has an MSB and LSB that are labeled as appropriate in the table, if any, that describes the format of the sub-structure having multiple fields.

If a field that consists of more than one byte contains either multiple fields each with their own values or a single value, then:

a) the MSB and LSB are labeled and they apply as described in this subclause whenever the field contains a single value; and
b) the labels on constituent fields supersede the single value labels whenever the field contains multiple fields.

If a field contains a text string (e.g., ASCII or UTF-8), the MSB label is the MSB of the first character and the LSB label is the LSB of the last character.

When required for clarity, multiple byte fields may be represented with only two rows in a table. This condition is represented by values in the byte number column not increasing by one in each subsequent table row, thus indicating the presence of additional bytes.

3.7 Notation conventions

3.7.1 Notation for procedure calls

In this standard, the model for functional interfaces between objects is a procedure call. 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 call.

Procedure Name: A descriptive name for the function modeled by the procedure call. When the procedure call model is used to describe a SCSI transport protocol service, the procedure name is the same as the service name.

Input-1, Input-2, …: A comma-separated list of names identifying caller-supplied input arguments.

Output-1, Output-2, …: A comma-separated list of names identifying output arguments to be returned by the procedure call.

"[…]": Brackets enclosing optional or conditional arguments.

This notation allows arguments to be specified as inputs and outputs. An interface between entities may require only inputs. If a procedure call has no output arguments, the word OUT, preceding comma, and associated pair of balanced parentheses are omitted.

The following is an example of a procedure call specification:

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

Where:

Found = Flag
Flag: if set to one, indicates that a matching item was located.

Input Arguments:

Pattern = … /* Definition of Pattern argument */
Argument containing the search pattern.

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

Output Arguments:

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

3.7.2 Notation for state diagrams

All state diagrams use the notation shown in figure 2.

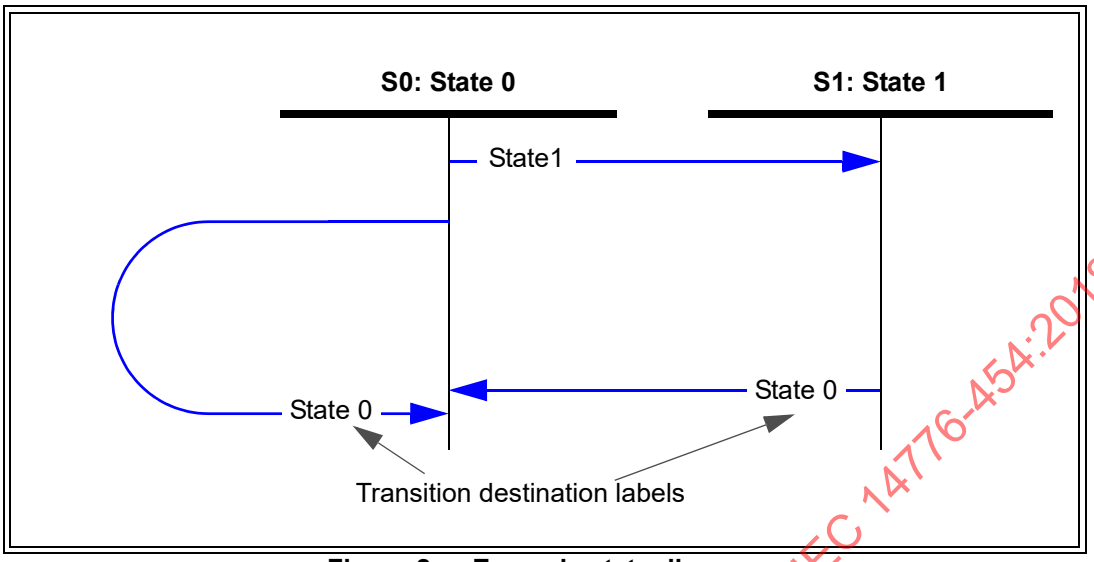


Figure 2 — Example state diagram

The state diagram is followed by subclauses describing the states and state transitions.

Each state and state transition is described in the list with particular attention to the conditions that cause the transition to occur and special conditions related to the transition.

A system described in this manner has the following properties:

a) time elapses only within discrete states; and
b) state transitions are logically instantaneous.

3.7.3 Notation for flowcharts

This standard uses flowcharts that ISO 5807:1985 defines as program flowcharts. Figure 3 shows an example flowchart.

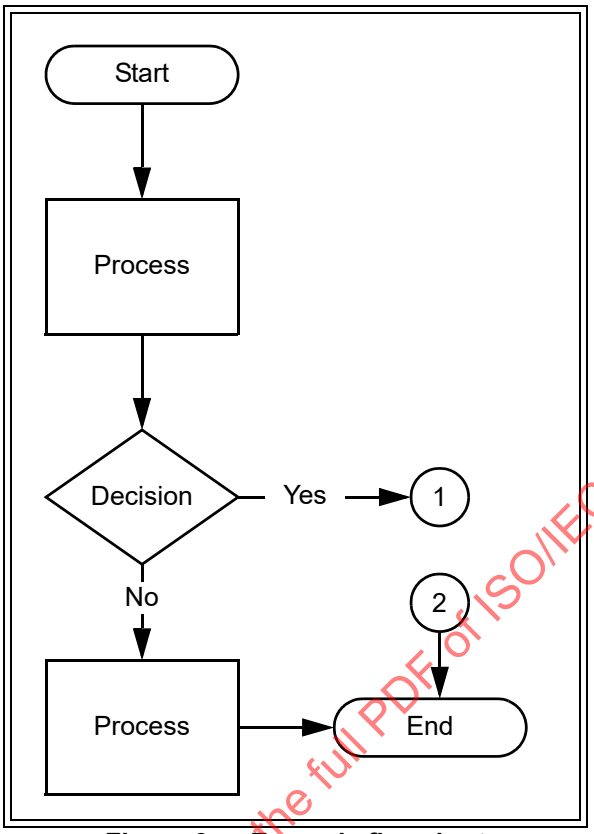


Figure 3 — Example flowchart

The following types of symbols are shown in figure 3:

a) a termination (e.g., start and end) symbol;
b) a process symbol;
c) a decision symbol; and
d) a reference (e.g., 1 and 2) symbol.

A termination symbol shows the starting point for the flowchart or the ending point for the flowchart.

A process symbol shows any kind of processing function that occurs as a result of entering this condition from a previous symbol.

A decision symbol shows a point in the progression of the flowchart from which there is more than one exit possibility of which only one is satisfied by the condition described within the decision symbol.

A reference symbol shows a connection to or from another flowchart and has the same number in both the source flowchart and destination flowchart.

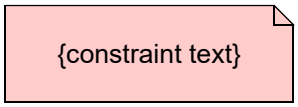
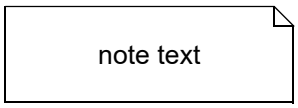
3.7.4 Notation for UML figures

3.7.4.1 Overview

This standard contains class diagram figures that use notation that is based on UML.

Some class diagrams contain constraints or notes that use the notion shown in table 3.

Table 3 — Class diagram constraints and notes notation

Notation	Description
	The presence of the curly brackets defines constraint that is a normative requirement. An example of a constraint is shown in figure 4.
	The absence of curly brackets defines a note that is informative. An example of a note is shown in figure 6.

The notation used to denote multiplicity of instances of classes and attributes in class diagrams is shown in table 4.

Table 4 — Class diagram multiplicity notation

Notation	Description
not specified	The number of instances of a class or an attribute is not specified.
1	One instance of the class or attribute exists.
0..*	Zero or more instances of the class or attribute exist.
1..*	One or more instances of the class or attribute exist.
0..1	Zero or one instance of the class or attribute exists.
n..m	n to m instances of the class or attribute exist (e.g., 2..8).
x,n..m	Multiple disjoint instances of the class or attribute exist (e.g., 2, 8..15).

Class diagrams show:

a) two or more classes (see 3.7.4.2); and
b) one or more of the following relationships between them:
   A) association (see 3.7.4.3);
   B) aggregation (see 3.7.4.4);
   C) generalization (see 3.7.4.5); and
   D) dependency (see 3.7.4.6).

3.7.4.2 Class notation

The notation used for classes is shown in table 5.

Table 5 — Class diagram notation for classes

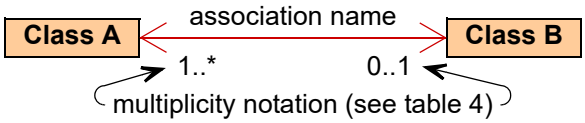
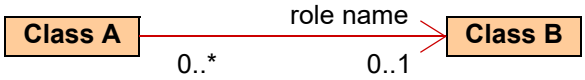
Notation	Description
Class Name **Class Name** **Class Name**	A class with no attributes or operations
Class Name Attribute01[1] Attribute02[1] **Class Name** Attribute01[1] Attribute02[1]	A class with attributes and no operations
Class Name Operation01() Operation02()	A class with operations and no attributes
Class Name Attribute01[1] Attribute02[1] Operation01() Operation02()	A class with attributes and operations
Class Name Attribute01[1..*] Attribute02[1] Operation01() Operation02()	A class with attributes that have a specified multiplicity (see table 4 in 3.7.4.1) and operations

3.7.4.3 Class association relationships notation

The notation used to denote association (i.e., "knows about") relationships between classes is shown in table 6. Unless the two classes in an association relationship also have an aggregation relationship (see 3.7.4.4),

association relationships have multiplicity notation (see table 4 in 3.7.4.1) at each end of the relationship line.

Table 6 — Class diagram notation for associations

Notation	Description
	Class A knows about Class B (i.e., read as "Class A association name Class B") and Class B knows about Class A (i.e., read as "Class B association name Class A")
	Class B knows about Class A (i.e., read as "Class B knows about Class A") but Class A does not know about Class B
	Class A knows about Class B (i.e., read as "Class A uses the role name attribute of Class B") but Class B does not know about Class A
Note: The use of role names and association names are optional.	

Several example association relationships between classes are shown in figure 4.

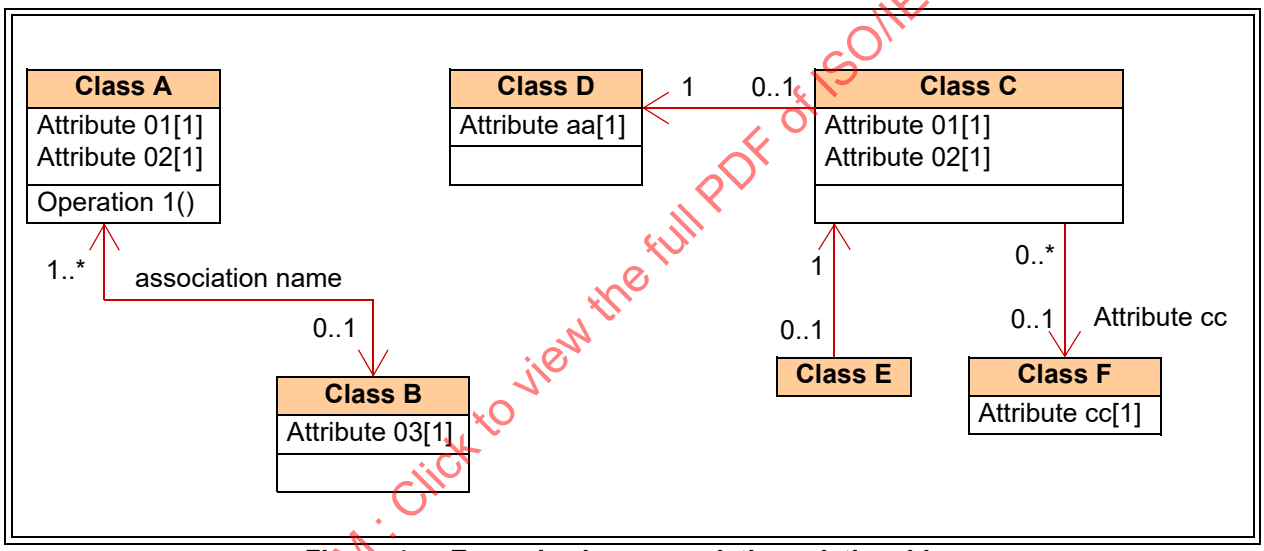


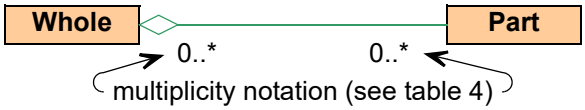
Figure 4 — Example class association relationships

3.7.4.4 Class aggregation relationships notation

The aggregation relationship is a specific type of association (see table 6). The notation used to denote aggregation (i.e., "is a part of" or "contains") relationships between classes is shown in table 7. Aggregation

relationships always include multiplicity notation (see table 4 in 3.7.4.1) at each end of the relationship line.

Table 7 — Class diagram notation for aggregations

Notation	Description
	The Part class is part of the Whole class and may continue to exist even if the Whole class is removed (i.e., read as "the Whole contains the Part.")
	The Part class is part of the Whole class, shall only belong to one Whole class, and shall not continue to exist if the Whole class is removed (i.e., read as "the Whole contains the Part.")

Several example aggregation relationships between classes are shown in figure 5.

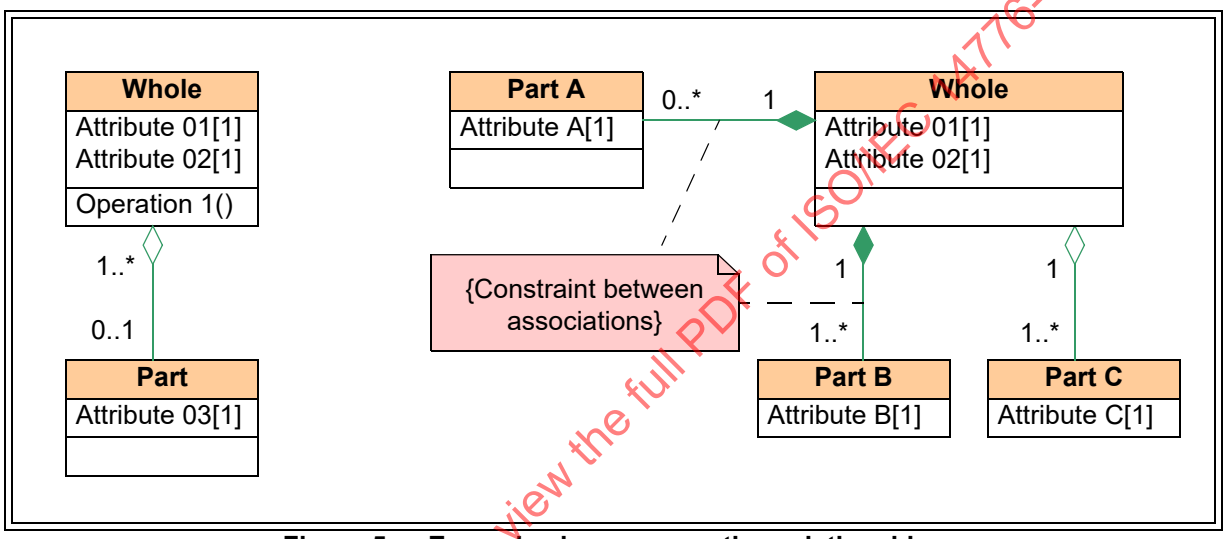


Figure 5 — Example class aggregation relationships

3.7.4.5 Class generalization relationships notation

The notation used to denote generalization (i.e., "is a kind of") relationships between classes is shown in table 8.

Table 8 — Class diagram notation for generalizations

Notation	Description
	Subclass is a kind of superclass. A subclass shares all the attributes and operations of the superclass (i.e., the subclass inherits from the superclass).

Several example generalization relationships between classes are shown in figure 6.

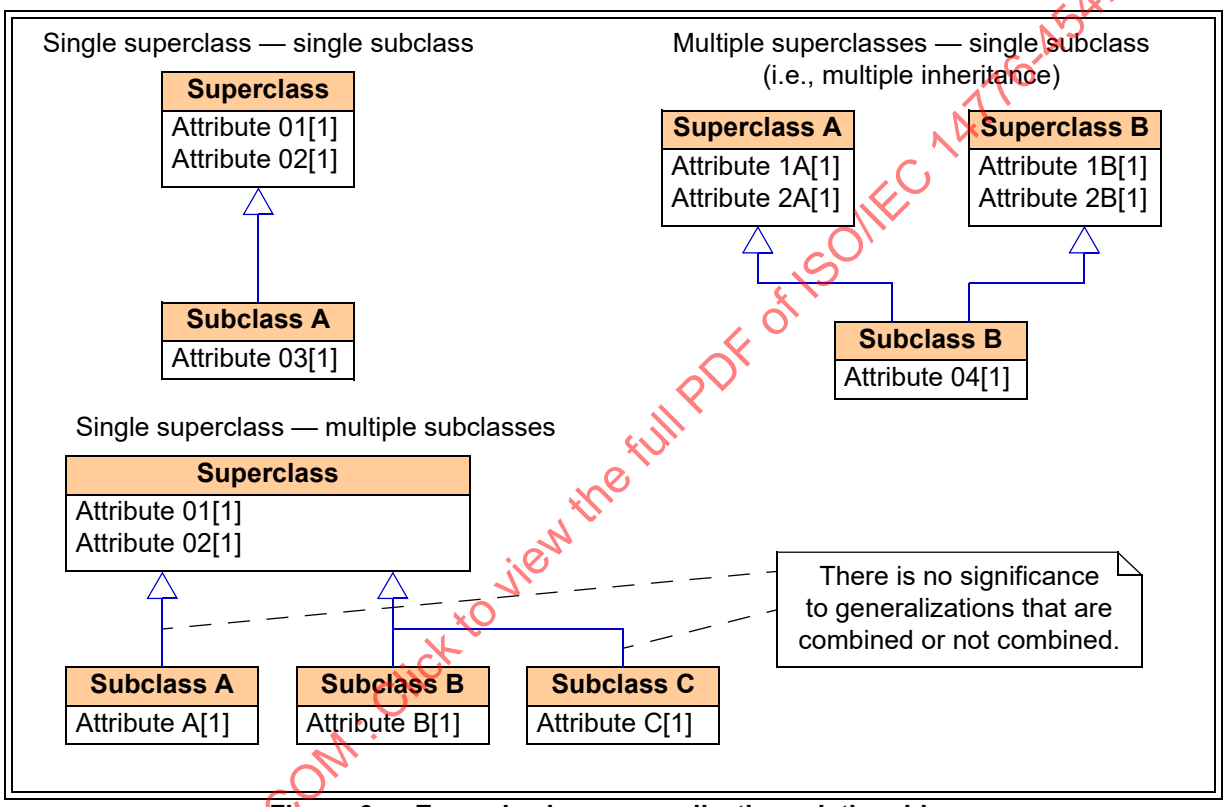


Figure 6 — Example class generalization relationships

3.7.4.6 Class dependency relationships notation

The notation used to denote dependency (i.e., "depends on") relationships between classes is shown in table 9.

Table 9 — Class diagram notation for dependencies

Notation	Description
	Class A depends on class B. A change in class B may cause a change in class A.

An example dependency relationship between classes is shown in figure 7.

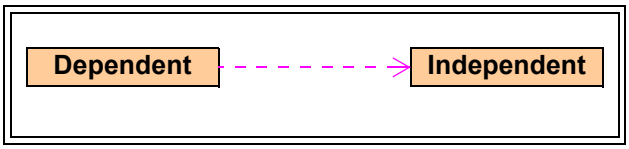


Figure 7 — Example class dependency relationships

3.7.4.7 Object notation

The notation used for objects is shown in table 10.

Table 10 — Notation for objects

Notation	Description
label : Class Name	Notation for a named object with no attributes
label : Class Name Attribute01 = x Attribute02 = y	Notation for a named object with attributes
: Class Name	Notation for an anonymous object with no attributes
: Class Name Attribute01 = x Attribute02 = y	Notation for an anonymous object with attributes

3.7.5 Notation for EXTENDED COPY command segment descriptors

The segment descriptors (see 6.4.6) contained in the parameter list for an EXTENDED COPY command (see 6.4) request the copy manager to perform operations on copy sources and copy destinations. The names of segment descriptors indicate this by inserting a right pointing arrow (i.e., →) between a brief description of the copy source and a brief description of the copy destination (e.g., block→stream).

If the segment descriptor requests the transfer of data to the copy destination and another location, an ampersand (i.e., &) is appended to the copy destination with a description of the additional location following the ampersand (e.g., block→stream&application client\).

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 6). This standard defines the parameters that are basic to more than one device model (see clause 7).

4.2 Command Descriptor Block

4.2.1 CDB usage and structure

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

If an object in a logical unit (e.g., a device server) validates reserved CDB fields and receives a reserved field within the CDB that is not zero, then the logical unit shall terminate the command with CHECK CONDITION status, with the sense key set to ILLEGAL REQUEST, and the additional sense code set to INVALID FIELD IN CDB.

If a logical unit receives a reserved CDB code value in a field other than the OPERATION CODE field, then the logical unit shall terminate the command with CHECK CONDITION status, with the sense key set to ILLEGAL REQUEST, and the additional sense code set to INVALID FIELD IN CDB.

The fixed length CDB formats are described in 4.2.2. The variable length CDB formats are described in 4.2.3. The XCDB format is described in 4.2.4. The CDB fields that are common to most commands are described in 4.2.5. The fields shown in 4.2.2 and 4.2.3 and described in 4.2.5 are used consistently by most commands. Except for the OPERATION CODE field, the SERVICE ACTION field, if any, and the CONTROL byte, the actual usage of any field is described in the standard defining that command. If a device server receives a CDB containing an operation code that is invalid or not supported, the device server shall terminate the command with CHECK CONDITION status, with the sense key set to ILLEGAL REQUEST, and the additional sense code set to INVALID COMMAND OPERATION CODE.

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

The XCDB format (see 4.2.4) is a CDB in which additional information is appended to another CDB. The requirements in this subclause apply to the XCDB and the CDB contained in the XCDB.

4.2.2 Fixed length CDB formats

4.2.2.1 Formats for 6-byte CDBs

4.2.2.1.1 Generic 6-byte CDB format

Table 11 shows the generic format of a 6-byte CDB.

Table 11 — Generic CDB format for 6-byte commands

Bit / Byte	7	6	5	4	3	2	1	0
0	OPERATION CODE							
1	miscellaneous CDB information							
•••								
4								
5	CONTROL							

The OPERATION CODE field is defined in 4.2.5.1.

Miscellaneous CDB information is defined by the standard that defines the command.

The CONTROL byte is defined in SAM-5.

4.2.2.1.2 Typical 6-byte CDB format

Table 12 shows the typical format of a 6-byte CDB, including the location of:

a) the TRANSFER LENGTH field, if any;
b) the PARAMETER LIST LENGTH field, if any; and
c) the ALLOCATION LENGTH field, if any.

Table 12 — Typical CDB format for 6-byte commands

Bit / Byte	7	6	5	4	3	2	1	0
0	OPERATION CODE							
1	miscellaneous CDB information							
•••								
3								
4	TRANSFER LENGTH (if any) PARAMETER LIST LENGTH (if any) ALLOCATION LENGTH (if any)							
5	CONTROL							

The OPERATION CODE field is defined in 4.2.5.1.

Miscellaneous CDB information is defined by the standard that defines the command.

The TRANSFER LENGTH field, if any, is defined in 4.2.5.4.

The PARAMETER LIST LENGTH field, if any, is defined in 4.2.5.5.

The ALLOCATION LENGTH field, if any, is defined in 4.2.5.6.

The CONTROL byte is defined in SAM-5.

4.2.2.2 Formats for 10-byte CDBs

4.2.2.2.1 Generic 10-byte CDB format

Table 13 shows the generic format of a 10-byte CDB, including the location of the SERVICE ACTION field, if any.

Table 13 — Generic CDB format for 10-byte commands

Bit / Byte	7	6	5	4	3	2	1	0
0	OPERATION CODE							
1	miscellaneous CDB information			SERVICE ACTION (if any) miscellaneous CDB information (if any)				
2	miscellaneous CDB information							
•••								
8								
9	CONTROL							

The OPERATION CODE field is defined in 4.2.5.1.

Miscellaneous CDB information is defined by the standard that defines the command.

The SERVICE ACTION field, if any, is defined in 4.2.5.2.

The CONTROL byte is defined in SAM-5.

4.2.2.2.2 Typical 10-byte CDB format

Table 14 shows the typical format of a 10-byte CDB, including the location of:

a) the SERVICE ACTION field, if any;
b) the LOGICAL BLOCK ADDRESS field, if any;
c) the TRANSFER LENGTH field, if any;
d) the PARAMETER LIST LENGTH field, if any; and
e) the ALLOCATION LENGTH field, if any.

Table 14 — Typical CDB format for 10-byte commands

Bit / Byte	7	6	5	4	3	2	1	0
0	OPERATION CODE							
1	miscellaneous CDB information			SERVICE ACTION (if any)				
2	(MSB)							
•••				LOGICAL BLOCK ADDRESS (if any)				
5								(LSB)
6				miscellaneous CDB information				
7	(MSB)			TRANSFER LENGTH (if any) PARAMETER LIST LENGTH (if any)				
8				ALLOCATION LENGTH (if any)				(LSB)
9	CONTROL							

The OPERATION CODE field is defined in 4.2.5.1.

Miscellaneous CDB information is defined by the standard that defines the command.

The SERVICE ACTION field, if any, is defined in 4.2.5.2.

The LOGICAL BLOCK ADDRESS field, if any, is defined in 4.2.5.3.

The TRANSFER LENGTH field, if any, is defined in 4.2.5.4.

The PARAMETER LIST LENGTH field, if any, is defined in 4.2.5.5.

The ALLOCATION LENGTH field, if any, is defined in 4.2.5.6.

The CONTROL byte is defined in SAM-5.

4.2.2.3 Formats for 12-byte CDBs

4.2.2.3.1 Generic 12-byte CDB format

Table 15 shows the generic format of a 12-byte CDB, including the location of the SERVICE ACTION field, if any.

Table 15 — Generic CDB format for 12-byte commands

Bit / Byte	7	6	5	4	3	2	1	0
0	OPERATION CODE							
1	miscellaneous CDB information			SERVICE ACTION (if any) miscellaneous CDB information (if any)				
2	miscellaneous CDB information							
•••								
10								
11	CONTROL							

The OPERATION CODE field is defined in 4.2.5.1.

Miscellaneous CDB information is defined by the standard that defines the command.

The SERVICE ACTION field, if any, is defined in 4.2.5.2.

The CONTROL byte is defined in SAM-5.

4.2.2.3.2 Typical 12-byte CDB format

Table 16 shows the typical format of a 12-byte CDB, including the location of:

a) the SERVICE ACTION field, if any;
b) the LOGICAL BLOCK ADDRESS field, if any;
c) the TRANSFER LENGTH field, if any;
d) the PARAMETER LIST LENGTH field, if any; and
e) the ALLOCATION LENGTH field, if any.

Table 16 — Typical CDB format for 12-byte commands

Bit / Byte	7	6	5	4	3	2	1	0
0	OPERATION CODE							
1	miscellaneous CDB information			SERVICE ACTION (if any)				
2	(MSB)			LOGICAL BLOCK ADDRESS (if any)				
•••								
5								(LSB)
6	(MSB)			TRANSFER LENGTH (if any) PARAMETER LIST LENGTH (if any) ALLOCATION LENGTH (if any)				
•••								
9								(LSB)
10	miscellaneous CDB information							
11	CONTROL							

The OPERATION CODE field is defined in 4.2.5.1.

Miscellaneous CDB information is defined by the standard that defines the command.

The SERVICE ACTION field, if any, is defined in 4.2.5.2.

The LOGICAL BLOCK ADDRESS field, if any, is defined in 4.2.5.3.

The TRANSFER LENGTH field, if any, is defined in 4.2.5.4.

The PARAMETER LIST LENGTH field, if any, is defined in 4.2.5.5.

The ALLOCATION LENGTH field, if any, is defined in 4.2.5.6.

The CONTROL byte is defined in SAM-5.

4.2.2.3.3 MAINTENANCE IN CDB format

Table 17 shows the generic format of a CDB that uses the MAINTENANCE IN operation code. If the PERIPHERAL DEVICE TYPE field is set to 0Ch (i.e., storage array controller device) or the SCCS bit is set to one in the standard INQUIRY data (see 6.6.2), then the MAINTENANCE IN service action definition in SCC-2 for the specified service action, if any, applies.

Table 17 — Generic CDB format for MAINTENANCE IN commands

Bit / Byte	7	6	5	4	3	2	1	0
0	OPERATION CODE (A3h)							
1	miscellaneous CDB information			SERVICE ACTION				
2	miscellaneous CDB information							
•••								
10								
11	CONTROL							

The OPERATION CODE field is defined in 4.2.5.1 and shall be set as shown in table 17 for any command that uses the MAINTENANCE IN CDB format.

Miscellaneous CDB information is defined by the standard that defines the command that uses the MAINTENANCE IN CDB format.

The SERVICE ACTION field is defined in 4.2.5.2, and in the standard that defines the command that uses the MAINTENANCE IN CDB format. A numeric ordered listing of service actions associated with the MAINTENANCE IN CDB format is provided in F.3.3.

The CONTROL byte is defined in SAM-5.

4.2.2.3.4 MAINTENANCE OUT CDB format

Table 18 shows the generic format of a CDB that uses the MAINTENANCE OUT operation code. If the PERIPHERAL DEVICE TYPE field is set to 0Ch (i.e., storage array controller device) or the SCCS bit is set to one in the standard INQUIRY data (see 6.6.2), then the MAINTENANCE OUT service action definition in SCC-2 for the specified service action, if any, applies.

Table 18 — Generic CDB format for MAINTENANCE OUT commands

Bit / Byte	7	6	5	4	3	2	1	0
0	OPERATION CODE (A4h)							
1	miscellaneous CDB information			SERVICE ACTION				
2	miscellaneous CDB information							
•••								
10								
11	CONTROL							

The OPERATION CODE field is defined in 4.2.5.1 and shall be set as shown in table 18 for any command that uses the MAINTENANCE OUT CDB format.

Miscellaneous CDB information is defined by the standard that defines the command that uses the MAINTENANCE OUT CDB format.

The SERVICE ACTION field is defined in 4.2.5.2, and in the standard that defines the command that uses the MAINTENANCE OUT CDB format. A numeric ordered listing of service actions associated with the MAINTENANCE OUT CDB format is provided in F.3.3.

The CONTROL byte is defined in SAM-5.

4.2.2.3.5 SERVICE ACTION IN(12) CDB format

Table 19 shows the generic format of a CDB that uses the SERVICE ACTION IN(12) operation code.

Table 19 — Generic CDB format for SERVICE ACTION IN(12) commands

Bit / Byte	7	6	5	4	3	2	1	0
0	OPERATION CODE (ABh)							
1	miscellaneous CDB information			SERVICE ACTION				
2	miscellaneous CDB information							
•••								
10								
11	CONTROL							

The OPERATION CODE field is defined in 4.2.5.1 and shall be set as shown in table 19 for any command that uses the SERVICE ACTION IN(12) CDB format.

Miscellaneous CDB information is defined by the standard that defines the command that uses the SERVICE ACTION IN(12) CDB format.

The SERVICE ACTION field is defined in 4.2.5.2, and in the standard that defines the command that uses the SERVICE ACTION IN(12) CDB format. A numeric ordered listing of service actions associated with the

SERVICE ACTION IN(12) CDB format is provided in table F.5 (see F.3.4).

The CONTROL byte is defined in SAM-5.

4.2.2.3.6 SERVICE ACTION OUT(12) CDB format

Table 20 shows the generic format of a CDB that uses the SERVICE ACTION OUT(12) operation code.

Table 20 — Generic CDB format for SERVICE ACTION OUT(12) commands

Bit / Byte	7	6	5	4	3	2	1	0
0	OPERATION CODE (A9h)							
1	miscellaneous CDB information			SERVICE ACTION				
2	miscellaneous CDB information							
•••								
10								
11	CONTROL							

The OPERATION CODE field is defined in 4.2.5.1 and shall be set as shown in table 20 for any command that uses the SERVICE ACTION OUT(12) CDB format.

Miscellaneous CDB information is defined by the standard that defines the command that uses the SERVICE ACTION OUT(12) CDB format.

The SERVICE ACTION field is defined in 4.2.5.2, and in the standard that defines the command that uses the SERVICE ACTION OUT(12) CDB format. A numeric ordered listing of service actions associated with the SERVICE ACTION OUT(12) CDB format is provided in table F.5 (see F.3.4).

The CONTROL byte is defined in SAM-5.

4.2.2.4 Formats for 16-byte CDBs

4.2.2.4.1 Generic 16-byte CDB format

Table 21 shows the generic format of a 16-byte CDB, including the location of the SERVICE ACTION field, if any.

Table 21 — Generic CDB format for 16-byte commands

Bit / Byte	7	6	5	4	3	2	1	0
0	OPERATION CODE							
1	miscellaneous CDB information			SERVICE ACTION (if any) miscellaneous CDB information (if any)				
2	miscellaneous CDB information							
•••								
14								
15	CONTROL							

The OPERATION CODE field is defined in 4.2.5.1.

Miscellaneous CDB information is defined by the standard that defines the command.

The SERVICE ACTION field, if any, is defined in 4.2.5.2.

The CONTROL byte is defined in SAM-5.

4.2.2.4.2 Typical 16-byte CDB format, if eight-byte LBAs not supported

Table 22 shows the typical format of a 16-byte CDB for commands that do not support eight-byte LBA values, including the location of:

a) the SERVICE ACTION field, if any;
b) the LOGICAL BLOCK ADDRESS field, if any;
c) the TRANSFER LENGTH field, if any;
d) the PARAMETER LIST LENGTH field, if any; and
e) the ALLOCATION LENGTH field, if any.

Table 22 — Typical CDB format for 16-byte commands, if eight-byte LBAs not supported

Bit Byte	7	6	5	4	3	2	1	0
0	OPERATION CODE							
1	miscellaneous CDB information			SERVICE ACTION (if any)				
2	(MSB)							
•••				LOGICAL BLOCK ADDRESS (if any)				
5								(LSB)
6								
•••				miscellaneous CDB information				
9								
10	(MSB)			TRANSFER LENGTH (if any) PARAMETER LIST LENGTH (if any) ALLOCATION LENGTH (if any)				
•••								
13								(LSB)
14	miscellaneous CDB information							
15	CONTROL							

The OPERATION CODE field is defined in 4.2.5.1.

Miscellaneous CDB information is defined by the standard that defines the command.

The SERVICE ACTION field, if any, is defined in 4.2.5.2.

The LOGICAL BLOCK ADDRESS field, if any, is defined in 4.2.5.3.

The TRANSFER LENGTH field, if any, is defined in 4.2.5.4.

The PARAMETER LIST LENGTH field, if any, is defined in 4.2.5.5.

The ALLOCATION LENGTH field, if any, is defined in 4.2.5.6.

The CONTROL byte is defined in SAM-5.

4.2.2.4.3 Typical 16-byte CDB format with eight-byte LBAs supported

Table 23 shows the typical format of a 16-byte CDB for commands that support eight-byte LBA values, including the location of:

a) the SERVICE ACTION field, if any;
b) the LOGICAL BLOCK ADDRESS field;
c) the TRANSFER LENGTH field, if any;
d) the PARAMETER LIST LENGTH field, if any; and
e) the ALLOCATION LENGTH field, if any.

Table 23 — Typical CDB format for 16-byte commands with eight-byte LBAs supported

Bit / Byte	7	6	5	4	3	2	1	0
0	OPERATION CODE							
1	miscellaneous CDB information			SERVICE ACTION (if any)				
2	(MSB)							
•••				LOGICAL BLOCK ADDRESS				
9								(LSB)
10	(MSB)			TRANSFER LENGTH (if any)				
•••				PARAMETER LIST LENGTH (if any)				
13				ALLOCATION LENGTH (if any)				(LSB)
14	miscellaneous CDB information							
15	CONTROL							

The OPERATION CODE field is defined in 4.2.5.1.

Miscellaneous CDB information is defined by the standard that defines the command.

The SERVICE ACTION field, if any, is defined in 4.2.5.2.

The LOGICAL BLOCK ADDRESS field is defined in 4.2.5.3.

The TRANSFER LENGTH field, if any, is defined in 4.2.5.4.

The PARAMETER LIST LENGTH field, if any, is defined in 4.2.5.5.

The ALLOCATION LENGTH field, if any, is defined in 4.2.5.6.

The CONTROL byte is defined in SAM-5.

4.2.2.4.4 SERVICE ACTION IN(16) CDB format

Table 24 shows the generic format of a CDB that uses the SERVICE ACTION IN(16) operation code.

Table 24 — Generic CDB format for SERVICE ACTION IN(16) commands

Bit / Byte	7	6	5	4	3	2	1	0
0	OPERATION CODE (9Eh)							
1	miscellaneous CDB information			SERVICE ACTION				
2	miscellaneous CDB information							
•••								
14								
15	CONTROL							

The OPERATION CODE field is defined in 4.2.5.1 and shall be set as shown in table 24 for any command that uses the SERVICE ACTION IN(16) CDB format.

Miscellaneous CDB information is defined by the standard that defines the command that uses the SERVICE ACTION IN(16) CDB format.

The SERVICE ACTION field is defined in 4.2.5.2, and in the standard that defines the command that uses the SERVICE ACTION IN(16) CDB format. A numeric ordered listing of service actions associated with the SERVICE ACTION IN(16) CDB format is provided in table F.6 (see F.3.4).

The CONTROL byte is defined in SAM-5.

4.2.2.4.5 SERVICE ACTION OUT(16) CDB format

Table 25 shows the generic format of a CDB that uses the SERVICE ACTION OUT(16) operation code.

Table 25 — Generic CDB format for SERVICE ACTION OUT(16) commands

Bit / Byte	7	6	5	4	3	2	1	0
0	OPERATION CODE (9Fh)							
1	miscellaneous CDB information			SERVICE ACTION				
2	miscellaneous CDB information							
•••								
14								
15	CONTROL							

The OPERATION CODE field is defined in 4.2.5.1 and shall be set as shown in table 25 for any command that uses the SERVICE ACTION OUT(16) CDB format.

Miscellaneous CDB information is defined by the standard that defines the command that uses the SERVICE ACTION OUT(16) CDB format.

The SERVICE ACTION field is defined in 4.2.5.2, and in the standard that defines the command that uses the SERVICE ACTION OUT(16) CDB format. A numeric ordered listing of service actions associated with the SERVICE ACTION OUT(16) CDB format is provided in table F.6 (see F.3.4).

The CONTROL byte is defined in SAM-5.

4.2.2.4.6 SERVICE ACTION BIDIRECTIONAL CDB format

Table 26 shows the generic format of a CDB that uses the SERVICE ACTION BIDIRECTIONAL operation code.

Table 26 — Generic CDB format for SERVICE ACTION BIDIRECTIONAL commands

Bit Byte	7	6	5	4	3	2	1	0
0	OPERATION CODE (9Dh)							
1	miscellaneous CDB information			SERVICE ACTION				
2	miscellaneous CDB information							
•••								
14								
15	CONTROL							

The OPERATION CODE field is defined in 4.2.5.1 and shall be set as shown in table 26 for any command that uses the SERVICE ACTION BIDIRECTIONAL CDB format.

Miscellaneous CDB information is defined by the standard that defines the command that uses the SERVICE ACTION BIDIRECTIONAL CDB format.

The SERVICE ACTION field is defined in 4.2.5.2, and in the standard that defines the command that uses the SERVICE ACTION BIDIRECTIONAL CDB format. A numeric ordered listing of service actions associated with the SERVICE ACTION BIDIRECTIONAL CDB format is provided in table F.7 (see F.3.5).

The CONTROL byte is defined in SAM-5.

4.2.3 Variable length CDB formats

4.2.3.1 Generic variable length CDB format

Table 27 shows the generic format of a variable length CDB.

Table 27 — Generic variable length CDB

Bit Byte	7	6	5	4	3	2	1	0
0	OPERATION CODE (7Fh)							
1	CONTROL							
2	miscellaneous CDB information							
•••								
6								
7	ADDITIONAL CDB LENGTH (n-7)							
8	(MSB)			SERVICE ACTION				
9								(LSB)
10	miscellaneous CDB information							
•••								
n								

The OPERATION CODE field is defined in 4.2.5.1 and shall be set as shown in table 27 for any command that uses the variable length CDB format.

The ADDITIONAL CDB LENGTH field specifies the number of additional CDB bytes that follow. The value in the ADDITIONAL CDB LENGTH field shall be a multiple of four. 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, then the command shall be terminated with CHECK CONDITION status, with the sense key set to ILLEGAL REQUEST, and the additional sense code set to INVALID FIELD IN CDB.

The CONTROL byte is defined in SAM-5.

Miscellaneous CDB information is defined by the standard that defines the command that uses the variable length CDB format.

The SERVICE ACTION field is defined in 4.2.5.2, and in the standard that defines the command that uses the variable length CDB format. The SERVICE ACTION field is required in the variable length CDB format. Lists of service action values ranges associated with the variable length CDB format are provided in F.3.6.

4.2.3.2 Typical 32-byte variable length CDB format

Table 28 shows the typical format of a 32-byte CDB for commands that support eight-byte LBA values, including the location of:

a) the SERVICE ACTION field;
b) the LOGICAL BLOCK ADDRESS field, if any;
c) the TRANSFER LENGTH field, if any;
d) the PARAMETER LIST LENGTH field, if any; and
e) the ALLOCATION LENGTH field, if any.

Table 28 — Typical variable length CDB format for 32-byte commands

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

The OPERATION CODE field is defined in 4.2.5.1 and shall be set as shown in table 28 for any command that uses the typical variable length CDB format for 32-byte commands.

The CONTROL byte is defined in SAM-5.

Miscellaneous CDB information is defined by the standard that defines the command.

The ADDITIONAL CDB LENGTH field is defined in 4.2.3.1 and shall be set as shown in table 28 for any command that uses the typical variable length CDB format for 32-byte commands.

The SERVICE ACTION field is defined in 4.2.5.2.

The LOGICAL BLOCK ADDRESS field, if any, is defined in 4.2.5.3.

The TRANSFER LENGTH field, if any, is defined in 4.2.5.4.

The PARAMETER LIST LENGTH field, if any, is defined in 4.2.5.5.

The ALLOCATION LENGTH field, if any, is defined in 4.2.5.6.

4.2.4 Extended CDBs

4.2.4.1 XCDB model

Any SCSI CDB except an XCDB may be extended in an XCDB. An XCDB shall be extended by adding additional XCDB descriptors to the existing XCDB.

XCDB descriptors may be:

a) added by application clients and removed by device servers; or
b) added and removed by entities outside the scope of this standard that transport or process CDBs and XCDBs during their transfer from an application client to a device server.

4.2.4.2 The XCDB format

Table 29 shows the format of an XCDB. In an XCDB, the CONTROL byte (see SAM-5) is the CONTROL byte in the CDB field.

Table 29 — XCDB format

Bit / Byte	7	6	5	4	3	2	1	0
0	OPERATION CODE (7Eh)							
1	Reserved							
2	(MSB)			LENGTH (n-3)				
3								(LSB)
4	CDB							
•••								
i								
	XCDB descriptors							
k	XCDB descriptor [first]							
•••								
	⋮							
	XCDB descriptor [last]							
•••								
n								

The OPERATION CODE field is defined in 4.2.5.1 and shall be set as shown in table 29 for an XCDB.

The LENGTH field specifies the number of bytes that follow in the XCDB.

The CDB field contains the CDB to which XCDB descriptors are being appended.

Each XCDB descriptor (see table 30) contains fields associated with a specified extension type (see table 31). The command shall be terminated with CHECK CONDITION status, with the sense key set to ILLEGAL REQUEST, and the additional sense code set to INVALID XCDB, if:

a) an XCDB does not contain at least one XCDB descriptor;
b) more than one XCDB descriptor contains the same extension type; or
c) the order of extension types in the XCDB descriptors differs from that shown in table 31.

Table 30 — XCDB descriptor format

Bit / Byte	7	6	5	4	3	2	1	0
0	EXTENSION TYPE							
1	Extension parameters							
•••								
n								

The EXTENSION TYPE field (see table 31) specifies the size and format of the extension parameters that follow in the XCDB descriptor.

Table 31 — EXTENSION TYPE field

Code	Descriptor Order [a]	Description	Extension size (bytes)	Reference
40h	first	CbCS extension descriptor	140	5.13.6.7.16
all others	Reserved			

[a] The order in which XCDB descriptors appear in an XCDB is arranged so that all the XCDB descriptors that follow an XCDB descriptor defined in a future version of this standard are also XCDB descriptors defined in a future version of this standard (i.e., after encountering one unrecognized XCDB descriptor, all subsequent XCDB descriptors are also going to be unrecognized).

The extension parameters contain additional information whose processing is associated with the CDB in the CDB field. The number, content, and size of the extension parameters depends on the contents of the EXTENSION TYPE field.

4.2.5 Common CDB fields

4.2.5.1 Operation code

The first byte of a SCSI CDB shall contain the OPERATION CODE field (see table 32) specifying the command being requested by the CDB.

Some operation codes are further qualified by a service action code (see 4.2.5.2). In such cases, the operation code and service action code combine to specify the command being requested. The location of the SERVICE ACTION field in the CDB varies depending on the operation code value. Operation codes and service action codes are defined in this standard and command standards.

Table 32 — OPERATION CODE field

Code	Length of CDB	Generic CDB format	Reference
00h to 1Fh	6 byte command	see table 11	4.2.2.1
20h to 5Fh	10 byte command	see table 13	4.2.2.2
60h to 7Dh	Reserved		
7Eh	variable (i.e., extended)	see table 29	4.2.4.2
7Fh	variable	see table 27	4.2.3.1
80h to 9Fh	16 byte command	see table 21	4.2.2.4
A0h to BFh	12 byte command	see table 15	4.2.2.3
C0h to FFh	vendor specific		

4.2.5.2 Service action

The SERVICE ACTION field provides further qualification for the OPERATION CODE field for some commands, allowing for:

a) unrelated commands that share the same operation code (e.g., the REPORT SUPPORTED OPERATION CODES command and the REPORT TARGET PORT GROUPS command); and
b) a set of related functions that share the same operation code (e.g., the PERSISTENT RESERVE IN command).

4.2.5.3 Logical block address

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

The typical 10-byte CDB format and typical 12-byte CDB format allow 32-bit LOGICAL BLOCK ADDRESS fields. The 16-byte CDB has two typical formats:

a) one allows a 32-bit LOGICAL BLOCK ADDRESS field (see table 22); and
b) the other allows a 64-bit LOGICAL BLOCK ADDRESS field (see table 23).

The typical 32-byte variable length CDB format (see table 28) allows a 64-bit LOGICAL BLOCK ADDRESS field.

LOGICAL BLOCK ADDRESS fields in additional parameter data have their length specified for each occurrence. See the specific command descriptions.

4.2.5.4 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.

In commands that use multiple bytes for the TRANSFER LENGTH field, a transfer length of zero specifies that no data transfer shall take place. This condition shall not be considered an error. A value of one or greater specifies the number of blocks or bytes that shall be transferred.

Refer to the specific command description for further information.

4.2.5.5 Parameter list length

The PARAMETER LIST LENGTH field is used to specify the number of bytes available for transfer in 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 specifies that no data shall be transferred. This condition shall not be considered an error, unless otherwise specified.

4.2.5.6 Allocation length

The ALLOCATION LENGTH field specifies the maximum number of bytes or blocks that an application client has allocated in the Data-In Buffer. The ALLOCATION LENGTH field specifies bytes unless a different requirement is stated in the command definition.

An allocation length of zero specifies that no data shall be transferred. This condition shall not be considered an error.

The device server shall terminate transfers to the Data-In Buffer when the number of bytes or blocks specified by the ALLOCATION LENGTH field 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 variable length data (e.g., 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 (e.g., a PARAMETER DATA LENGTH field, a PAGE LENGTH field, a DESCRIPTOR LENGTH field, an AVAILABLE DATA field), then 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 states otherwise.

If the amount of information that is available to be transferred exceeds the maximum value that the ALLOCATION LENGTH field in combination with other fields in the CDB is capable of specifying, then no data shall be transferred and the command shall be terminated with CHECK CONDITION status, with the sense key set to ILLEGAL REQUEST, and the additional sense code set to INVALID FIELD IN CDB.

4.3 Data field requirements

4.3.1 ASCII data field requirements

ASCII data fields shall contain only ASCII printable characters (i.e., code values 20h to 7Eh) and may be terminated with one or more ASCII null (00h) characters.

ASCII data fields described as being left-aligned shall have any unused bytes at the end of the field (i.e., highest offset) and the unused bytes shall be filled with ASCII space characters (20h).

ASCII data fields described as being right-aligned shall have any unused bytes at the start of the field (i.e., lowest offset) and the unused bytes shall be filled with ASCII space characters (20h).

4.3.2 Null data field termination and zero padding requirements

A data field that is described as being null-terminated shall have:

1) zero or more bytes that contain ASCII or UTF-8 characters that are not the null (i.e., 00h) character;
2) followed by one byte that contains the ASCII/UTF-8 null character; and
3) followed by zero or more bytes whose contents are ignored.

A data field may be specified to be a fixed length. The length specified for a data field may be greater than the length required to contain the contents of the field. A data field may be specified to have a length that is a multiple of a given value (e.g., a multiple of four bytes). When such fields are described as being null-padded, the bytes, if any, between the end of the data and the end of the field data shall contain ASCII or UTF-8 null (00h) characters. When such fields are described as being zero-padded, the bytes, if any, between the end of the data and the end of the field data shall contain zeros.

NOTE 5 - There is no difference between the pad byte contents in null-padded and zero-padded fields. The difference is in the format of the other bytes in the field.

A data field that is described as being both null-terminated and null-padded shall have at least one byte containing an ASCII or UTF-8 null (00h) character in the end of the field (i.e., highest offset) and may have more than one byte containing ASCII or UTF-8 null characters to meet the specified field length requirements. If more than one byte in a null-terminated, null-padded field contains the ASCII or UTF-8 null character, then all the bytes containing the ASCII or UTF-8 null character shall be at the end of the field (i.e., only the highest offsets).

4.3.3 Variable type data field requirements

Parameter lists may contain fields or descriptors in which data may be represented in different formats. To indicate which format is being used a field may be defined that contains a code set enumeration (see table 33).

Table 33 — Code set enumeration

Code	Description
0h	Reserved
1h	The associated fields or descriptors contain binary values
2h	The associated fields or descriptors contain ASCII printable characters (i.e., code values 20h to 7Eh)
3h	The associated fields or descriptors contain UTF-8 codes
4h to Fh	Reserved

4.3.4 Port identifier field requirements

The contents of RELATIVE PORT IDENTIFIER fields, RELATIVE INITIATOR PORT IDENTIFIER fields, and RELATIVE TARGET PORT IDENTIFIER fields are defined in table 34.

Table 34 — 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 to FFFFh	Relative port 3 through 65 535

4.4 Secure random numbers

Secure random numbers should be generated as specified by RFC 4086 (e.g., see NIST SP 800-90 A).

If the same random number source is used to generate random numbers for multiple purposes (e.g., nonces and secret keys), then interactions between the two shall not be allowed to compromise secrecy. If the value sequence generated by the common random number source is predictable to any degree, then the random number values that are transmitted outside the SCSI device may provide information about the random number values that the SCSI device maintains internally, based on the reasonable assumption that an adversary knows the order in which the random numbers are obtained from the common random number source. SCSI devices shall eliminate sources of such predictability.

Compliance with RFC 4086 is one method for achieving the required independence between random number values.

4.5 Sense data

4.5.1 Sense data introduction

Sense data shall be returned in the same I_T_L_Q nexus transaction as the status and as parameter data in response to the REQUEST SENSE command (see 6.39). Sense data returned in the same I_T_L_Q nexus transaction as the status shall be either fixed format or descriptor format sense data format based on the value of the D_SENSE bit in the Control mode page (see 7.5.8). The REQUEST SENSE command may be used to request either fixed format sense data or descriptor format sense data.

The first byte of all sense data contains the RESPONSE CODE field (see table 35) that indicates the report type and format of the sense data.

Table 35 — Sense data response codes

Response Code	Report type		Sense data format	
	Description	Reference	Description	Reference
00h to 6Fh	Reserved			
70h	Current information	4.5.6	Fixed	4.5.3
71h	Deferred error	4.5.7	Fixed	4.5.3
72h	Current information	4.5.6	Descriptor	4.5.2
73h	Deferred error	4.5.7	Descriptor	4.5.2
74h to 7Eh	Reserved			
7Fh	Vendor specific			

If sense data is returned in the same I_T_L_Q nexus transaction as the status, the RESPONSE CODE field shall be set to 70h in all unit attention condition sense data in which:

a) the ADDITIONAL SENSE CODE field is set to 29h; or
b) the additional sense code is set to MODE PARAMETERS CHANGED.

4.5.2 Descriptor format sense data

4.5.2.1 Descriptor format sense data overview

The descriptor format sense data for response codes 72h (i.e., current information) and 73h (i.e., deferred errors) is defined in table 36.

Table 36 — Descriptor format sense data

Bit / Byte	7	6	5	4	3	2	1	0
0	Reserved	RESPONSE CODE (72h or 73h)						
1	Reserved				SENSE KEY			
2	ADDITIONAL SENSE CODE							
3	ADDITIONAL SENSE CODE QUALIFIER							
4	SDAT_OVFL	Reserved						
5	Reserved							
6								
7	ADDITIONAL SENSE LENGTH (n-7)							
	Sense data descriptor list							
8	Sense data descriptor (see table 37) [first]							
•••								
	⋮							
	Sense data descriptor (see table 37) [last]							
•••								
n								

The contents of the RESPONSE CODE field indicate the report type and format of the sense data (see 4.5.1). For descriptor format sense data, the RESPONSE CODE field shall be set to 72h or 73h.

The SENSE KEY field, ADDITIONAL SENSE CODE field, and ADDITIONAL SENSE CODE QUALIFIER field provide a hierarchy of information. The hierarchy provides a top-down approach for an application client to determine information relating to the reported condition.

The SENSE KEY field indicates generic information describing a reported condition. The sense keys are defined in 4.5.8.

The ADDITIONAL SENSE CODE field indicates further information related to the condition reported in the SENSE KEY field. Support of the additional sense codes not required by this standard is optional. For a list of additional sense codes see 4.5.8. If the device server does not have further information related to the reported condition, the ADDITIONAL SENSE CODE field shall be set to zero.

The ADDITIONAL SENSE CODE QUALIFIER field indicates detailed information related to the condition reported in the ADDITIONAL SENSE CODE field. If the condition is reported by the device server, the value returned shall be as defined in 4.5.8. If the device server does not have detailed information related to the reported condition, the ADDITIONAL SENSE CODE QUALIFIER field shall be set to zero.

A sense data overflow (SDAT_OVFL) bit set to one indicates that the device server truncated the sense data to ensure that the sense data length is less than or equal to the length specified in the MAXIMUM SENSE DATA LENGTH field in the Control Extension mode page (see 7.5.9). A SDAT_OVFL bit set to zero indicates that the

device server did not truncate the sense data to ensure that the sense data length is less than or equal to the length specified in the MAXIMUM SENSE DATA LENGTH field in the Control Extension mode page.

The ADDITIONAL SENSE LENGTH field indicates the number of additional sense bytes that follow. The additional sense length shall be less than or equal to 244 (i.e., limiting the total length of the sense data to 252 bytes). If the sense data is being transferred as parameter data by a REQUEST SENSE command, then the contents of the ADDITIONAL SENSE LENGTH field are not altered based on the allocation length (see 4.2.5.6). If the sense data is being transferred in the same I_T_L_Q nexus transaction as the status and the sense data is longer than the length specified in the MAXIMUM SENSE DATA LENGTH field in the Control Extension mode page (see 7.5.9), then the device server shall ensure the number of sense data bytes is less than or equal to the length specified in the MAXIMUM SENSE DATA LENGTH field in the Control Extension mode page by discarding entire descriptors (i.e., not including a partial descriptor). The maximum total length of the sense data transferred by the device server is indicated in the MAXIMUM SUPPORTED SENSE DATA LENGTH field in the Extended INQUIRY VPD page (see 7.8.7).

Sense data descriptors (see table 37) provide specific sense information. A given type of sense data descriptor shall be included in the sense data only if that descriptor contains valid information.

Table 37 — Sense data descriptor format

Bit / Byte	7	6	5	4	3	2	1	0
0	DESCRIPTOR TYPE							
1	ADDITIONAL LENGTH (n-1)							
2	Sense data descriptor specific							
•••								
n								

The DESCRIPTOR TYPE field (see table 38) identifies the type of sense data descriptor.

Table 38 — DESCRIPTOR TYPE field

Code	Description	Number of descriptors allowed (maximum)	Reference
00h	Information	1 [a]	4.5.2.2
01h	Command specific information	1 [a]	4.5.2.3
02h	Sense key specific	1 [a]	4.5.2.4
03h	Field replaceable unit	1 [a]	4.5.2.5
04h	Stream commands	1	SSC-4
05h	Block commands	1 [a]	SBC-3

[a] The direct-access block device sense data descriptor, if used, is used by a direct-access block device instead of the information, command specific information, sense key specific, field replaceable unit, and block commands sense data descriptors. See SBC-3.

Table 38 — DESCRIPTOR TYPE field

Code	Description	Number of descriptors allowed (maximum)	Reference
06h	OSD object identification	1	OSD
07h	OSD response integrity check value	1	OSD
08h	OSD attribute identification	1	OSD
09h	ATA Status Return	1	SAT-3
0Ah	Another progress indication	32	4.5.2.6
0Bh	User data segment referral	1	SBC-3
0Ch	Forwarded sense data	2	4.5.2.7
0Dh	Direct-access block device	1 [a]	SBC-3
0Eh to 7Fh	Reserved		
80h to FFh	Vendor specific		4.5.2.8

[a] The direct-access block device sense data descriptor, if used, is used by a direct-access block device instead of the information, command specific information, sense key specific, field replaceable unit, and block commands sense data descriptors. See SBC-3.

The ADDITIONAL LENGTH field indicates the number of sense data descriptor specific bytes that follow in the sense data descriptor.

4.5.2.2 Information sense data descriptor

The information sense data descriptor (see table 39) is included in the descriptor format sense data if device-type information or command specific information is available as defined in this standard or a command standard. See 4.5.4 for device server requirements regarding how values are returned in the INFORMATION field.

Table 39 — Information sense data descriptor format

Bit / Byte	7	6	5	4	3	2	1	0
0	DESCRIPTOR TYPE (00h)							
1	ADDITIONAL LENGTH (0Ah)							
2	VALID (1b)			Reserved				
3	Reserved							
4	(MSB)							
•••	INFORMATION							
11								(LSB)

The DESCRIPTOR TYPE and ADDITIONAL LENGTH fields are described in 4.5.2.1 and shall be set as shown in table 39 for the information sense data descriptor.

The VALID bit shall be set to one.

NOTE 6 - In SPC-2, in the fixed format sense data (see 4.5.3), and in sense data descriptors other than the information sense data descriptor that contain a VALID bit and an INFORMATION field (e.g., the direct-access

block device sense data descriptor (see SBC-3)), the VALID bit indicates whether the contents of the INFORMATION field are valid as defined by a command standard. Since the contents of the INFORMATION field are valid whenever the information sense data descriptor is included in the sense data, the only legal value for the VALID bit in the information sense data descriptor is one.

The contents of the INFORMATION field are device-type or command specific and are defined in a command standard.

4.5.2.3 Command-specific information sense data descriptor

The command-specific information sense data descriptor (see table 40) is included in the descriptor format sense data if sense data information is available that depends on the command for which the reported condition occurred. See 4.5.5 for device server requirements regarding how values are returned in the COMMAND-SPECIFIC INFORMATION field.

Table 40 — Command-specific information sense data descriptor format

Bit / Byte	7	6	5	4	3	2	1	0
0	DESCRIPTOR TYPE (01h)							
1	ADDITIONAL LENGTH (0Ah)							
2	Reserved							
3	Reserved							
4	(MSB)							
•••	COMMAND-SPECIFIC INFORMATION							
11								(LSB)

The DESCRIPTOR TYPE and ADDITIONAL LENGTH fields are described in 4.5.2.1 and shall be set as shown in table 40 for the command-specific information sense data descriptor.

The contents of the COMMAND-SPECIFIC INFORMATION field are command specific, and are defined in this standard or a command standard.

4.5.2.4 Sense key specific sense data descriptor

4.5.2.4.1 Sense key specific sense data descriptor overview

The sense key specific sense data descriptor (see table 41) is included in the descriptor format sense data if additional information is available about the reported condition described in 4.5.2.1. The format and content of the sense key specific information depends on the value in the SENSE KEY field (see 4.5.2.1).

Table 41 — Sense key specific sense data descriptor format

Bit / Byte	7	6	5	4	3	2	1	0
0	DESCRIPTOR TYPE (02h)							
1	ADDITIONAL LENGTH (06h)							
2	Reserved							
3	Reserved							
4	SKSV (1b)			sense key specific information (see table 42)				
5								
6								
7	Reserved							

The DESCRIPTOR TYPE and ADDITIONAL LENGTH fields are described in 4.5.2.1 and shall be set as shown in table 41 for the sense-key specific sense data descriptor.

The sense-key specific valid (SKSV) bit shall be set to one.

NOTE 7 - In SPC-2, in the fixed format sense data (see 4.5.3), and in sense data descriptors other than the sense key specific sense data descriptor that contain a VALID bit and an INFORMATION field (e.g., the direct-access block device sense data descriptor (see SBC-3)), the SKSV bit indicates whether the sense key specific information is valid as defined by a command standard. Since the sense key specific information is valid whenever a sense key specific sense data descriptor is included in the sense data, the only legal value for the SKSV bit in the sense key specific sense data descriptor is one.

The content and format of the sense key specific information (see table 42) is determined by the value of the SENSE KEY field (see 4.5.2.1).

Table 42 — Sense key specific information definitions

Sense key	Sense key specific information	Reference
ILLEGAL REQUEST	Field pointer	4.5.2.4.2
HARDWARE ERROR, MEDIUM ERROR, or RECOVERED ERROR	Actual retry count	4.5.2.4.3
NO SENSE or NOT READY	Progress indication	4.5.2.4.4
COPY ABORTED	Segment pointer	4.5.2.4.5
UNIT ATTENTION	Unit attention condition queue overflow	4.5.2.4.6
All other sense keys	In descriptor format sense data: a) the sense key specific sense data descriptor shall not appear in the sense data descriptor list; and b) in other sense data descriptors that contain an SKSV bit and sense key specific information (e.g., the direct-access block device sense data descriptor (see SBC-3)), the SKSV bit shall be set to zero. In fixed format sense data (see 4.5.3), the SKSV bit shall be set to zero.	

4.5.2.4.2 Field pointer sense key specific information

If the sense key is ILLEGAL REQUEST, the sense key specific information (see table 41) shall have the content and format shown in table 43.

Table 43 — Field pointer sense key specific information

Bit / Byte	7	6	5	4	3	2	1	0
0	SKSV	C/D	Reserved		BPV	BIT POINTER		
1	(MSB)	FIELD POINTER						
2								(LSB)

The SKSV bit is described in 4.5.2.4.1 for descriptor format sense data and in 4.5.3 for fixed format sense data.

A command data (C/D) bit set to one indicates that the illegal parameter is in the CDB. A C/D bit set to zero indicates that the illegal parameter is in the data parameters transferred by the application client in the Data-Out Buffer.

A bit pointer valid (BPV) bit set to zero indicates that the value in the BIT POINTER field is not valid. A BPV bit set to one indicates that the BIT POINTER field specifies which bit of the byte designated by the FIELD POINTER field is in error. If a multiple-bit field is in error, the BIT POINTER field shall point to the first bit (i.e., the left-most bit) of the field. If several consecutive bits are reserved, each bit should be treated as a single-bit 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. If a multiple-byte field is in error, the field pointer shall point to the first byte (i.e., the left-most byte) of the field. If several consecutive bytes are reserved, each shall be treated as a single-byte field.

NOTE 8 - The byte or bytes identified as being in error may not be the bytes that need to be changed to correct the problem.

4.5.2.4.3 Actual retry count sense key specific information

If the sense key is HARDWARE ERROR, MEDIUM ERROR, or RECOVERED ERROR, then the sense key specific information (see table 41) shall have the content and format shown in table 44.

Table 44 — Actual retry count sense key specific information

Bit / Byte	7	6	5	4	3	2	1	0
0	SKSV	Reserved						
1	(MSB)	ACTUAL RETRY COUNT						
2								(LSB)

The SKSV bit is described in 4.5.2.4.1 for descriptor format sense data and in 4.5.3 for fixed format sense data.

The ACTUAL RETRY COUNT field contains vendor specific information on the number of retries of the recovery algorithm used in attempting to recover an error or exception condition. This field should be computed in the same way as the retry count fields within the Read-Write Error Recovery mode page (see SBC-3, SSC-4, and MMC-6).

4.5.2.4.4 Progress indication sense key specific information

If the sense key is NO SENSE or NOT READY, the sense key specific information (see table 41) shall have the content and format shown in table 45.

Table 45 — Progress indication sense key specific information

Bit / Byte	7	6	5	4	3	2	1	0
0	SKSV	Reserved						
1	(MSB)	PROGRESS INDICATION						
2								(LSB)

The SKSV bit is described in 4.5.2.4.1 for descriptor format sense data and in 4.5.3 for fixed format sense data.

The PROGRESS INDICATION field is a percent complete indication in which the value is a numerator that has 65 536 (10000h) as its denominator. The progress indication shall be based upon the total operation. The progress indication numerator should be time related; however, this is not an absolute requirement.

EXAMPLE - Since format time varies with the number of defects encountered, etc., the device server may assign values to various steps within the process, and use these values as the progress indication numerator. The granularity of these steps should be small enough to provide reasonable assurances to the application client that progress is being made.

4.5.2.4.5 Segment pointer sense key specific information

If the sense key is COPY ABORTED, the sense key specific information (see table 41) shall have the content and format shown in table 46.

Table 46 — Segment pointer sense key specific information

Bit Byte	7	6	5	4	3	2	1	0
0	SKSV	Reserved	SD	Reserved	BPV	BIT POINTER		
1	(MSB)	FIELD POINTER						
2								(LSB)

The SKSV bit is described in 4.5.2.4.1 for descriptor format sense data and in 4.5.3 for fixed format sense data.

The segment descriptor (SD) bit indicates whether the field pointer is relative to the start of the parameter list or to the start of a segment descriptor. An SD bit set to zero indicates that the field pointer is relative to the start of the parameter list. An SD bit set to 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 5.16.7.4).

A bit pointer valid (BPV) bit set to zero indicates that the value in the BIT POINTER field is not valid. A BPV bit set to one indicates that the BIT POINTER field specifies which bit of the byte designated by the FIELD POINTER field is in error. If 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 being processed when the error or exception condition was detected.

If the SD bit is set to zero and the byte in error has an offset greater than FFFFh, the FIELD POINTER field shall be set to FFFFh and the BPV bit shall be set to zero.

4.5.2.4.6 Unit attention condition queue overflow sense key specific information

If the sense key is UNIT ATTENTION, the sense key specific information (see table 41) shall have the content and format shown in table 47.

Table 47 — Unit attention condition queue overflow sense key specific information

Bit Byte	7	6	5	4	3	2	1	0
0	SKSV	Reserved						OVERFLOW
1	Reserved							
2								

The SKSV bit is described in 4.5.2.4.1 for descriptor format sense data and in 4.5.3 for fixed format sense data.

An OVERFLOW bit set to one indicates that the unit attention condition queue has overflowed. An OVERFLOW bit set to zero indicates that the unit attention condition queue has not overflowed.

4.5.2.5 Field replaceable unit sense data descriptor

The field replaceable unit sense data descriptor (see table 48) is included in the descriptor format sense data if information is available about a component associated with the sense data.

Table 48 — Field replaceable unit sense data descriptor format

Bit / Byte	7	6	5	4	3	2	1	0
0	DESCRIPTOR TYPE (03h)							
1	ADDITIONAL LENGTH (02h)							
2	Reserved							
3	FIELD REPLACEABLE UNIT CODE							

The DESCRIPTOR TYPE and ADDITIONAL LENGTH fields are described in 4.5.2.1 and shall be set as shown in table 48 for the field replaceable unit sense data descriptor.

Non-zero values in the FIELD REPLACEABLE UNIT CODE field are used to identify a component associated with the sense data. A value of zero in this field indicates that no specific component has been associated with the sense data or that the data is not available. 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 VPD page (see 7.8.3), if supported by the device server.

4.5.2.6 Another progress indication sense data descriptor

If the sense key is set to NO SENSE or NOT READY, the another progress indication sense data descriptor (see table 49) may be included in the descriptor format sense data to provide a progress indication for one operation other than the one described by the non-descriptor fields in 4.5.2.1. The sense data should include one another progress indication sense data descriptor for each operation for which the device server is able to report progress other than the operation described by the non-descriptor fields in 4.5.2.1.

Table 49 — Another progress indication sense data descriptor format

Bit / Byte	7	6	5	4	3	2	1	0
0	DESCRIPTOR TYPE (0Ah)							
1	ADDITIONAL LENGTH (06h)							
2	ANOTHER SENSE KEY							
3	ANOTHER ADDITIONAL SENSE CODE							
4	ANOTHER ADDITIONAL SENSE CODE QUALIFIER							
5	Reserved							
6	(MSB)			ANOTHER PROGRESS INDICATION				
7								(LSB)

The DESCRIPTOR TYPE field and ADDITIONAL LENGTH field are described in 4.5.2.1 and shall be set as shown in table 49 for the progress indications sense data descriptor.

The ANOTHER SENSE KEY field indicates generic information about the operation for which this another progress indication sense data descriptor provides a progress indication. A list of sense key values is in 4.5.8.

The ANOTHER ADDITIONAL SENSE CODE field indicates further information about the operation for which this another progress indication sense data descriptor provides a progress indication. A list of additional sense codes is in 4.5.8.

The ANOTHER ADDITIONAL SENSE CODE QUALIFIER field indicates detailed information related to the additional sense code for the operation for which this another progress indication sense data descriptor provides a progress indication. The value returned in the ADDITIONAL SENSE CODE QUALIFIER field shall be as defined in 4.5.8.

The ANOTHER PROGRESS INDICATION field indicates a percent complete for the operation indicated by the ANOTHER SENSE KEY field, the ANOTHER ADDITIONAL SENSE CODE field, and the ANOTHER ADDITIONAL SENSE CODE QUALIFIER field. The value in the ANOTHER PROGRESS INDICATION field shall be as defined in 4.5.2.4.4.

4.5.2.7 Forwarded sense data

Forwarded sense data descriptors (see table 50) are included in the descriptor format sense data if status and sense data is available from another device server as part of command completion (e.g., an exception condition returned by a copy target device in an EXTENDED COPY(LID4) command (see 6.4) or EXTENDED COPY(LID1) command (see 6.5) during segment descriptor processing).

Table 50 — Forwarded sense data descriptor format

Bit / Byte	7	6	5	4	3	2	1	0
0	DESCRIPTOR TYPE (0Ch)							
1	ADDITIONAL LENGTH (n-1)							
2	FSDT	Reserved				SENSE DATA SOURCE		
3	FORWARDED STATUS							
4	FORWARDED SENSE DATA							
•••								
n								

The DESCRIPTOR TYPE field is described in 4.5.2.1, and shall be set as shown in table 50 for the forwarded sense data descriptor.

The ADDITIONAL LENGTH field shall be set to:

a) a minimum of 22, if the FORWARDED SENSE DATA field contains fixed format sense data (i.e., the RESPONSE CODE field (see 4.5.1) in the forwarded sense data is 70h or 71h);
b) a minimum of 10, if the FORWARDED SENSE DATA field contains descriptor format sense data (i.e., the RESPONSE CODE field in the forwarded sense data is 72h or 73h); and
c) two less than a multiple of four (i.e., the FORWARDED SENSE DATA field is a multiple of four bytes in length).

A forwarded sense data truncated (FSDT) bit set to one indicates that the contents of the FORWARDED SENSE DATA field have been truncated (i.e., the FORWARDED SENSE DATA field does not contain all of the sense data that was supplied). An FSDT bit set to zero indicates that the contents of the FORWARDED SENSE DATA field have not been truncated.

The SENSE DATA SOURCE field (see table 51) indicates the supplier of the forwarded sense data.

Table 51 — SENSE DATA SOURCE field

Code	Description
0h	Unknown
1h	EXTENDED COPY command copy source (see 5.16.7.2)
2h	EXTENDED COPY command copy destination (see 5.16.7.2)
all others	Reserved

The FORWARDED STATUS field contains the status code (see SAM-5) returned by the supplier of the forwarded sense data at the same time that the forwarded sense data was returned.

The FORWARDED SENSE DATA field contains the forwarded sense data. The FORWARDED SENSE DATA field is a zero-padded (see 4.3.2) field whose length is a multiple of four bytes.

4.5.2.8 Vendor specific sense data descriptors

Vendor specific sense data descriptors (see table 52) may be included in the descriptor format sense data if vendor specific data is available that further defines the nature of the reported condition.

Table 52 — Vendor specific sense data descriptor format

Bit / Byte	7	6	5	4	3	2	1	0
0	DESCRIPTOR TYPE (80h to FFh)							
1	ADDITIONAL LENGTH (n-1)							
2	Vendor specific							
•••								
n								

The DESCRIPTOR TYPE field and ADDITIONAL LENGTH field are described in 4.5.2.1. The DESCRIPTOR TYPE field shall be set as shown in table 52 for the vendor specific sense data descriptor.

4.5.3 Fixed format sense data

The fixed format sense data for response codes 70h (current information) and 71h (deferred errors) is defined in table 53.

Table 53 — Fixed format sense data

Bit / Byte	7	6	5	4	3	2	1	0
0	VALID	RESPONSE CODE (70h or 71h)						
1	Obsolete							
2	FILEMARK	EOM	ILI	SDAT_OVFL	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 information (see 4.5.2.4)						
16								
17								
18	Additional sense bytes							
•••								
n								

A VALID bit set to zero indicates that the INFORMATION field is not defined in this standard or any other command standard. A VALID bit set to one indicates the INFORMATION field contains valid information as defined in this standard or a command standard. See 4.5.4 for device server requirements regarding the VALID bit.

The contents of the RESPONSE CODE field indicate the report type and format of the sense data (see 4.5.1). For fixed format sense data, the RESPONSE CODE field shall be set to 70h or 71h.

The meaning of the FILEMARK bit is device-type or command specific (e.g., see the SSC-4 READ command and SPACE command for examples of FILEMARK bit usage) and the bit is defined in a command standard.

The meaning of the end-of-medium (EOM) bit is device-type or command specific (e.g., see the SSC-4 READ command, SPACE command, and WRITE command for examples of EOM bit usage) and the bit is defined in a command standard.

The meaning of the incorrect length indicator (ILI) bit is device-type or command specific (e.g., see the SSC-4 READ command an example of ILI bit usage) and the bit is defined in a command standard.

The SDAT_OVFL bit, SENSE KEY field, ADDITIONAL SENSE CODE field, and ADDITIONAL SENSE CODE QUALIFIER field are described in 4.5.2.1.

The contents of the INFORMATION field are device-type or command specific and are defined in a command standard. See 4.5.4 for device server requirements regarding how values are returned in the INFORMATION field.

The ADDITIONAL SENSE LENGTH field indicates the number of additional sense bytes that follow. The additional sense length shall be less than or equal to 244 (i.e., limiting the total length of the sense data to 252 bytes). If the sense data is being returned as parameter data by a REQUEST SENSE command, then the contents of the ADDITIONAL SENSE LENGTH field are not altered based on the allocation length (see 4.2.5.6). If the sense data is being returned in the same I_T_L_Q nexus transaction as the status and the sense data is longer than the length specified in the MAXIMUM SENSE DATA LENGTH field in the Control Extension mode page (see 7.5.9), then the device server shall ensure the number of sense data bytes is less than or equal to the length specified in the MAXIMUM SENSE DATA LENGTH field in the Control Extension mode page. The maximum total length of the sense data returned by the device server is indicated in the MAXIMUM SUPPORTED SENSE DATA LENGTH field in the Extended INQUIRY VPD page (see 7.8.7).

The contents of the COMMAND-SPECIFIC INFORMATION field are command specific, and are defined in this standard or a command standard. The COMMAND-SPECIFIC INFORMATION field should be ignored in sense data:

a) for a command or operation for which the COMMAND-SPECIFIC INFORMATION field is not defined; or
b) that is not related to a command or operation (e.g., pollable sense data (see 5.10)).

See 4.5.5 for device server requirements regarding how values are returned in the COMMAND-SPECIFIC INFORMATION field.

The FIELD REPLACEABLE UNIT CODE field is described in 4.5.2.5.

A sense-key specific valid (SKSV) bit set to one indicates the sense key specific information is valid as defined in this standard. An SKSV bit set to zero indicates that the content and format of the sense key specific information is not as defined by this standard.

The sense key specific information is described in 4.5.2.4.

The additional sense bytes may contain:

a) information related to conditions detected during the processing of an EXTENDED COPY command (see 5.16.7.4); and/or
b) vendor specific data that further defines the nature of the reported condition.

4.5.4 Returning a value in the INFORMATION field in the sense data

To return a value less than or equal to FFFF_FFFFh in the INFORMATION field:

a) if fixed format sense data (see 4.5.3) is being returned, the device server shall set the VALID bit to one and shall set the INFORMATION field to the value; and
b) if descriptor format sense data (see 4.5.2) is being returned and a sense data descriptor that contains a VALID bit and an INFORMATION field is being returned (e.g., the information descriptor (see 4.5.2.2) or the direct-access block device sense data descriptor (see SBC-3)), then the device server shall set the VALID bit to one, the first four bytes of the INFORMATION field to zero, and the next four bytes of the INFORMATION field to the value.

To return a value greater than FFFF_FFFFh in the INFORMATION field:

a) if fixed format sense data (see 4.5.3) is being returned, the device server shall set the VALID bit to zero and shall set the INFORMATION field to a vendor specific value. The value is not able to be reported; and

b) if descriptor format sense data (see 4.5.2) is being returned and a sense data descriptor that contains a VALID bit and an INFORMATION field is being returned (e.g., the information descriptor (see 4.5.2.2) or the direct-access block device sense data descriptor (see SBC-3)), then the device server shall set the VALID bit to one and shall set the INFORMATION field to the value.

To return sense data and not return a value in the INFORMATION field:

a) if fixed format sense data (see 4.5.3) is being returned, the device server shall set the VALID bit to zero and shall set the INFORMATION field to a vendor specific value; and
b) if descriptor format sense data (see 4.5.2) is being returned, then:
   A) the device server shall not return an information descriptor (see 4.5.2.2); and
   B) if a sense data descriptor other than an information descriptor that contains a VALID bit and an INFORMATION field is being returned (e.g., the direct-access block device sense data descriptor (see SBC-3)), then the device server shall set the VALID bit to zero and shall set the INFORMATION field to a vendor specific value.

4.5.5 Returning a value in the COMMAND-SPECIFIC INFORMATION field in the sense data

To return a value less than or equal to FFFF_FFFFh in the COMMAND-SPECIFIC INFORMATION field:

a) if fixed format sense data (see 4.5.3) is being returned, the device server shall set the COMMAND-SPECIFIC INFORMATION field to the value; and
b) if descriptor format sense data (see 4.5.2) is being returned and a sense data descriptor that contains a COMMAND-SPECIFIC INFORMATION field is being returned (e.g., the command-specific information descriptor (see 4.5.2.3) or the direct-access block device sense data descriptor (see SBC-3)), then the device server shall set the first four bytes of the COMMAND-SPECIFIC INFORMATION field to zero, and the next four bytes of the COMMAND-SPECIFIC INFORMATION field to the value.

To return a value greater than FFFF_FFFFh in the COMMAND-SPECIFIC INFORMATION field:

a) if fixed format sense data (see 4.5.3) is being returned, the device server shall set the COMMAND-SPECIFIC INFORMATION field to a vendor specific value. The value is not able to be reported; and
b) if descriptor format sense data (see 4.5.2) is being returned and a sense data descriptor that contains a COMMAND-SPECIFIC INFORMATION field is being returned (e.g., the command-specific information descriptor (see 4.5.2.3) or the direct-access block device sense data descriptor (see SBC-3)), then the device server shall set the COMMAND-SPECIFIC INFORMATION field to the value.

To return sense data and not return a value in the COMMAND-SPECIFIC INFORMATION field:

a) if fixed format sense data (see 4.5.3) is being returned, the device server shall set the COMMAND-SPECIFIC INFORMATION field to a vendor specific value; and
b) if descriptor format sense data (see 4.5.2) is being returned, then:
   A) the device server shall not return a command-specific information descriptor (see 4.5.2.3); and
   B) if a sense data descriptor that contains a COMMAND-SPECIFIC INFORMATION field is being returned (e.g., the direct-access block device sense data descriptor (see SBC-3)), then the device server shall set the COMMAND-SPECIFIC INFORMATION field to a vendor specific value.

4.5.6 Current information

Response codes 70h and 72h (i.e., current information) indicate that the sense data is:

a) the result of an error, exception condition, or protocol specific failure that is associated with CHECK CONDITION status; or
b) additional information that is associated with a status other than CHECK CONDITION.

Current information includes:

a) errors generated during processing of a command terminated with CHECK CONDITION status;
b) errors not related to any command that are detected during processing of a command (e.g., disk servomechanism failures, off-track errors, or power-up test errors); and
c) referral information (see SBC-3) associated with GOOD status.

4.5.7 Deferred errors

Response codes 71h and 73h (deferred error) indicate that the sense data is the result of an error or exception condition that occurred during processing of a previous command for which GOOD status or CONDITION MET status has already been returned. Such commands are associated with the 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 may be indicated by returning CHECK CONDITION status to an application client accessed through a defined I_T nexus as described in this subclause.

If a command terminates with CHECK CONDITION status and the sense data describes a deferred error, the terminated command shall not have been processed. After the device server detects a deferred error condition, the device server 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 returned unless required by the error handling parameters of a mode page (e.g., the Informational Exceptions mode page defined by SBC-3 and SSC-4). The occurrence of the error may be logged;
b) if it is possible to associate a deferred error with an I_T nexus 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, then a deferred error indication shall be returned for a command received on the I_T nexus associated with the deferred error. If a command received on an I_T nexus other than the I_T nexus associated with the deferred error attempts to access the particular function or subset of data associated with the deferred error and the TST field equals 000b (see 7.5.8), then the device server shall complete the command with BUSY status or ACA ACTIVE status according to the requirements in SAM-5. If a command received on an I_T nexus other than the I_T nexus associated with the deferred error attempts to access the particular function or subset of data associated with the deferred error and the TST field equals 001b, then 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 I_T nexus or with a particular subset of data, then the device server shall return a deferred error for one command received on each I_T nexus. If multiple deferred errors have accumulated for an I_T nexus, then:
   A) one error shall be returned; and
   B) only the last error should be returned;
d) if the SCSI target device is unable to associate a deferred error with a particular logical unit, the SCSI target device shall establish a deferred error for every logical unit and shall return the deferred error for one command received on each appropriate I_T nexus associated with each logical unit; or
e) if a command has never been an enabled command, and a deferred error occurs, the device server shall terminate the command with CHECK CONDITION status and deferred error information returned in the sense data. If a deferred error occurs after a command becomes an enabled command and the command is affected by the error, then the device server shall terminate the command with 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 still exist after the current error information has been returned. If a deferred error occurs after a command has become an enabled command and the command completes successfully, then 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 9 - 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 cached or buffered write operations, then synchronization commands should be performed before the critical data is destroyed. This is necessary for actions taken when deferred errors occur in the storing of the data. The synchronizing process should provide the necessary commands to allow returning CHECK CONDITION status and subsequent returning of deferred error sense information after all cached or buffered operations are completed.

4.5.8 Sense key and additional sense code definitions

The sense keys are defined in table 54.

Table 54 — Sense key descriptions (part 1 of 2)

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 is terminated with CHECK CONDITION status (e.g., as a result of the FILEMARK bit, EOM bit, or ILI bit being set to one).
1h	**RECOVERED ERROR:** Indicates that the command completed successfully, with some recovery action performed by the device server. Details may be determined by examining the sense data (e.g., the INFORMATION field). If multiple recovered errors occur during one command, the choice of which error to report (e.g., first, last, most severe) is vendor specific.
2h	**NOT READY:** Indicates that the logical unit is not accessible. Operator intervention may be required to correct this condition.
3h	**MEDIUM ERROR:** Indicates that the command terminated with a non-recovered error condition that may have been 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: a) the command was addressed to an incorrect logical unit number (see SAM-5); b) the command had an invalid task attribute (see SAM-5); c) the command was addressed to a logical unit whose current configuration prohibits processing the command; d) there was an illegal parameter in the CDB; or e) there was an illegal parameter in the additional parameters supplied as data for some commands (e.g., PERSISTENT RESERVE OUT). If the device server detects an invalid parameter in the CDB, the device server 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 a unit attention condition has been established (e.g., the removable medium may have been changed, a logical unit reset occurred). See SAM-5.
7h	**DATA PROTECT:** Indicates that a command that reads or writes the medium was attempted on a block that is protected. The read or write operation was not performed.
8h	**BLANK CHECK:** Indicates that blank or non-blank medium was encountered when not expected.
9h	**VENDOR SPECIFIC:** This sense key is available for reporting vendor specific conditions.

Table 54 — Sense key descriptions (part 2 of 2)

Sense Key	Description
Ah	**COPY ABORTED:** Indicates a third-party copy command (see 5.16.3) was aborted after some data was transferred but before all data was transferred.
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	Reserved
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-4.)
Eh	**MISCOMPARE:** Indicates that the source data did not match the data read from the medium.
Fh	**COMPLETED:** Indicates there is command completed sense data (see SAM-5) to be reported. This may occur for a successful command.

The additional sense codes (i.e., the ADDITIONAL SENSE CODE field and ADDITIONAL SENSE CODE QUALIFIER field values in sense data) are defined in table 55.

Table 55 — ASC and ASCQ assignments (part 1 of 18)

D – Direct Access Block Device (SBC-3)
. T – Sequential Access Device (SSC-4)
. L – Printer Device (SSC)
. P – Processor Device (SPC-2)
. . W – Write Once Block Device (SBC)
. . R – C/DVD Device (MMC-6)
. . O – Optical Memory Block Device (SBC)
. . . M – Media Changer Device (SMC-3)
. . . A – Storage Array Device (SCC-2)
. . . E – SCSI Enclosure Services device (SES-3)
. . . . B – Simplified Direct-Access (Reduced Block) device (RBC)
. . . . K – Optical Card Reader/Writer device (OCRW)
. . . . V – Automation/Device Interface device (ADC-3)
. . . . . F – Object-based Storage Device (OSD-2)

Device Column key
blank = code not used
not blank = code used

ASC	ASCQ	DTLPWROMAEBKVF	Description
20h	0Bh	DT PWROMAEBK	ACCESS DENIED - ACL LUN CONFLICT
20h	08h	DT PWROMAEBK	ACCESS DENIED - ENROLLMENT CONFLICT
20h	01h	DT PWROMAEBK	ACCESS DENIED - INITIATOR PENDING-ENROLLED
20h	09h	DT PWROMAEBK	ACCESS DENIED - INVALID LU IDENTIFIER
20h	03h	DT PWROMAEBK	ACCESS DENIED - INVALID MGMT ID KEY
20h	0Ah	DT PWROMAEBK	ACCESS DENIED - INVALID PROXY TOKEN
20h	02h	DT PWROMAEBK	ACCESS DENIED - NO ACCESS RIGHTS
4Bh	03h	DT PWROMAEBK	ACK/NAK TIMEOUT
67h	02h	A	ADD LOGICAL UNIT FAILED
13h	00h	D W O BK	ADDRESS MARK NOT FOUND FOR DATA FIELD
12h	00h	D W O BK	ADDRESS MARK NOT FOUND FOR ID FIELD
67h	08h	A	ASSIGN FAILURE OCCURRED
27h	03h	T R	ASSOCIATED WRITE PROTECT
2Ah	06h	DTLPWROMAEBKVF	ASYMMETRIC ACCESS STATE CHANGED
47h	04h	DTLPWROMAEBKVF	ASYNCHRONOUS INFORMATION PROTECTION ERROR DETECTED
44h	71h	DT B	ATA DEVICE FAILED SET FEATURES
67h	0Bh	DT B	ATA DEVICE FEATURE NOT ENABLED
00h	1Dh	DT B	ATA PASS THROUGH INFORMATION AVAILABLE
00h	21h	D	ATOMIC COMMAND ABORTED DUE TO ACA
67h	06h	A	ATTACHMENT OF LOGICAL UNIT FAILED

Table 55 — ASC and ASCQ assignments (part 2 of 18)

D – Direct Access Block Device (SBC-3)
. T – Sequential Access Device (SSC-4)
. L – Printer Device (SSC)
. P – Processor Device (SPC-2)
. . W – Write Once Block Device (SBC)
. . R – C/DVD Device (MMC-6)
. . O – Optical Memory Block Device (SBC)
. . . M – Media Changer Device (SMC-3)
. . . A – Storage Array Device (SCC-2)
. . . E – SCSI Enclosure Services device (SES-3)
. . . . B – Simplified Direct-Access (Reduced Block) device (RBC)
. . . . K – Optical Card Reader/Writer device (OCRW)
. . . . V – Automation/Device Interface device (ADC-3)
. . . . . F – Object-based Storage Device (OSD-2)

Device Column key
blank = code not used
not blank = code used

ASC	ASCQ	D	T	L	P	W	R	O	M	A	E	B	K	V	F	Description
21h	06h	D														ATTEMPT TO READ INVALID DATA
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
74h	40h	D	T				R		M	A	E	B	K	V		AUTHENTICATION FAILED
66h	00h															AUTOMATIC DOCUMENT FEEDER COVER UP
66h	01h															AUTOMATIC DOCUMENT FEEDER LIFT UP
55h	06h	D	T			W	R	O	M			B				AUXILIARY MEMORY OUT OF SPACE
11h	12h	D	T			W	R	O	M			B				AUXILIARY MEMORY READ ERROR
0Ch	0Bh	D	T			W	R	O	M			B				AUXILIARY MEMORY WRITE ERROR
00h	04h		T													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	O	M	A	E	B	K	V	F	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
2Ah	09h	D														CAPACITY DATA HAS CHANGED
52h	00h		T													CARTRIDGE FAULT
73h	00h						R									CD CONTROL ERROR
24h	01h	D	T	L	P	W	R	O		A	E	B	K	V	F	CDB DECRYPTION ERROR
3Fh	02h	D	T	L	P	W	R	O	M			B	K			CHANGED OPERATING DEFINITION
11h	06h					W	R	O				B				CIRC UNRECOVERED ERROR
30h	03h	D	T				R		M				K			CLEANING CARTRIDGE INSTALLED
30h	07h	D	T	L		W	R	O	M	A	E	B	K	V	F	CLEANING FAILURE
30h	0Ah	D	T			W	R	O		A	E	B	K			CLEANING REQUEST REJECTED
00h	17h	D	T	L		W	R	O	M	A	E	B	K	V	F	CLEANING REQUESTED
30h	13h								M							CLEANING VOLUME EXPIRED
4Ah	00h	D	T	L	P	W	R	O	M	A	E	B	K	V	F	COMMAND PHASE ERROR
2Ch	00h	D	T	L	P	W	R	O	M	A	E	B	K	V	F	COMMAND SEQUENCE ERROR
2Eh	01h	D				W		O	M			B				COMMAND TIMEOUT BEFORE PROCESSING
2Eh	02h	D				W		O	M			B				COMMAND TIMEOUT DURING PROCESSING
2Eh	03h	D				W		O	M			B				COMMAND TIMEOUT DURING PROCESSING DUE TO ERROR RECOVERY
6Eh	00h									A						COMMAND TO LOGICAL UNIT FAILED
2Fh	00h	D	T	L	P	W	R	O	M	A	E	B	K	V	F	COMMANDS CLEARED BY ANOTHER INITIATOR
2Fh	02h	D	T	L	P	W	R	O	M	A	E	B	K	V	F	COMMANDS CLEARED BY DEVICE SERVER
2Fh	01h	D														COMMANDS CLEARED BY POWER LOSS NOTIFICATION
3Fh	04h	D	T			W	R	O	M	A	E	B	K			COMPONENT DEVICE ATTACHED

Table 55 — ASC and ASCQ assignments (part 3 of 18)

D – Direct Access Block Device (SBC-3)
. T – Sequential Access Device (SSC-4)
. L – Printer Device (SSC)
. P – Processor Device (SPC-2)
. . W – Write Once Block Device (SBC)
. . R – C/DVD Device (MMC-6)
. . O – Optical Memory Block Device (SBC)
. . . M – Media Changer Device (SMC-3)
. . . . A – Storage Array Device (SCC-2)
. . . . E – SCSI Enclosure Services device (SES-3)
. . . . . B – Simplified Direct-Access (Reduced Block) device (RBC)
. . . . . K – Optical Card Reader/Writer device (OCRW)
. . . . . V – Automation/Device Interface device (ADC-3)
. . . . . F – Object-based Storage Device (OSD-2)
. . . . . .

Device Column key
blank = code not used
not blank = code used

ASC	ASCQ	D	T	L	P	W	R	O	M	A	E	B	K	V	F	Description
0Ch	04h	D				W		O				B				COMPRESSION CHECK MISCOMPARE ERROR
27h	06h						R								F	CONDITIONAL WRITE PROTECT
67h	00h									A						CONFIGURATION FAILURE
67h	01h									A						CONFIGURATION OF INCAPABLE LOGICAL UNITS FAILED
6Fh	07h						R									CONFLICT IN BINDING NONCE RECORDING
00h	1Eh	D	T				R		M	A	E	B	K	V		CONFLICTING SA CREATION REQUEST
4Bh	07h	D	T		P	W	R	O	M	A	E	B	K		F	CONNECTION LOST
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
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	O					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	O					K			COPY SEGMENT GRANULARITY VIOLATION
0Dh	05h	D	T	L	P	W	R	O		A			K			COPY TARGET DEVICE DATA OVERRUN
0Dh	04h	D	T	L	P	W	R	O		A			K			COPY TARGET DEVICE DATA UNDERRUN
0Dh	02h	D	T	L	P	W	R	O		A			K			COPY TARGET DEVICE NOT REACHABLE
67h	07h									A						CREATION OF LOGICAL UNIT FAILED
74h	04h		T													CRYPTOGRAPHIC INTEGRITY VALIDATION FAILED
73h	10h						R									CURRENT POWER CALIBRATION AREA ALMOST FULL
73h	11h						R									CURRENT POWER CALIBRATION AREA IS FULL
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

Table 55 — ASC and ASCQ assignments (part 4 of 18)

D – Direct Access Block Device (SBC-3)
. T – Sequential Access Device (SSC-4)
. L – Printer Device (SSC)
. P – Processor Device (SPC-2)
. . W – Write Once Block Device (SBC)
. . R – C/DVD Device (MMC-6)
. . O – Optical Memory Block Device (SBC)
. . . M – Media Changer Device (SMC-3)
. . . A – Storage Array Device (SCC-2)
. . . E – SCSI Enclosure Services device (SES-3)
. . . . B – Simplified Direct-Access (Reduced Block) device (RBC)
. . . . K – Optical Card Reader/Writer device (OCRW)
. . . . V – Automation/Device Interface device (ADC-3)
. . . . . F – Object-based Storage Device (OSD-2)

Device Column key
blank = code not used
not blank = code used

ASC	ASCQ	D	T	L	P	W	R	O	M	A	E	B	K	V	F	Description
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
55h	0Ah								M							DATA CURRENTLY UNAVAILABLE
26h	05h	D	T	L	P	W	R	O		A		B	K			DATA DECRYPTION ERROR
26h	10h		T													DATA DECRYPTION KEY FAIL LIMIT REACHED
2Ah	0Dh		T													DATA ENCRYPTION CAPABILITIES CHANGED
74h	21h		T													DATA ENCRYPTION CONFIGURATION PREVENTED
2Ah	13h		T													DATA ENCRYPTION KEY INSTANCE COUNTER HAS CHANGED
2Ah	11h		T													DATA ENCRYPTION PARAMETERS CHANGED BY ANOTHER I_T NEXUS
2Ah	12h		T													DATA ENCRYPTION PARAMETERS CHANGED BY VENDOR SPECIFIC EVENT
0Ch	05h	D	T				W		O			B				DATA EXPANSION OCCURRED DURING COMPRESSION
69h	00h									A						DATA LOSS ON LOGICAL UNIT
4Bh	05h	D	T		P	W	R	O	M	A	E	B	K			DATA OFFSET ERROR
41h	00h	D														DATA PATH FAILURE (SHOULD USE 40 NN)
47h	01h	D	T	L	P	W	R	O	M	A	E	B	K	V	F	DATA PHASE CRC ERROR DETECTED
4Bh	00h	D	T	L	P	W	R	O	M	A	E	B	K	V	F	DATA PHASE ERROR
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
53h	0Ch								M							DATA TRANSFER DEVICE ERROR - EJECT FAILED
53h	0Dh								M							DATA TRANSFER DEVICE ERROR - LIBRARY COMMUNICATION FAILED
53h	09h								M							DATA TRANSFER DEVICE ERROR - LOAD FAILED
53h	0Ah								M							DATA TRANSFER DEVICE ERROR - UNLOAD FAILED
53h	0Bh								M							DATA TRANSFER DEVICE ERROR - UNLOAD MISSING
3Bh	1Bh								M							DATA TRANSFER DEVICE INSERTED
3Bh	1Ah								M							DATA TRANSFER DEVICE REMOVED
4Bh	0Ah	D	T		P	W	R	O	M	A	E	B	K		F	DATA-IN BUFFER ERROR
4Bh	09h	D	T		P	W	R	O	M	A	E	B	K		F	DATA-IN BUFFER OVERFLOW - DATA BUFFER DESCRIPTOR AREA
4Bh	08h	D	T		P	W	R	O	M	A	E	B	K		F	DATA-IN BUFFER OVERFLOW - DATA BUFFER SIZE
4Bh	0Dh	D	T		P	W	R	O	M	A	E	B	K		F	DATA-OUT BUFFER ERROR
4Bh	0Ch	D	T		P	W	R	O	M	A	E	B	K		F	DATA-OUT BUFFER OVERFLOW - DATA BUFFER DESCRIPTOR AREA
4Bh	0Bh	D	T		P	W	R	O	M	A	E	B	K		F	DATA-OUT BUFFER OVERFLOW - DATA BUFFER SIZE
11h	0Dh	D	T			W	R	O				B				DE-COMPRESSION CRC ERROR
71h	00h		T													DECOMPRESSION EXCEPTION LONG ALGORITHM ID

Table 55 — ASC and ASCQ assignments (part 5 of 18)

D – Direct Access Block Device (SBC-3)
. T – Sequential Access Device (SSC-4)
. L – Printer Device (SSC)
. P – Processor Device (SPC-2)
. . W – Write Once Block Device (SBC)
. . R – C/DVD Device (MMC-6)
. . O – Optical Memory Block Device (SBC)
. . . M – Media Changer Device (SMC-3)
. . . A – Storage Array Device (SCC-2)
. . . E – SCSI Enclosure Services device (SES-3)
. . . . B – Simplified Direct-Access (Reduced Block) device (RBC)
. . . . K – Optical Card Reader/Writer device (OCRW)
. . . . V – Automation/Device Interface device (ADC-3)
. . . . . F – Object-based Storage Device (OSD-2)

Device Column key
blank = code not used
not blank = code used

ASC	ASCQ	D T L P W R O M A E B K V F	Description
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
0Ch	0Fh	R	DEFECTS IN ERROR WINDOW
3Fh	05h	D T W R O M A E B K	DEVICE IDENTIFIER CHANGED
29h	04h	D T L P W R O M A E B K V F	DEVICE INTERNAL RESET
40h	NNh	D T L P W R O M A E B K V F	DIAGNOSTIC FAILURE ON COMPONENT NN (80h-FFh)
74h	08h	D T R M E V F	DIGITAL SIGNATURE VALIDATION FAILURE
66h	02h		DOCUMENT JAM IN AUTOMATIC DOCUMENT FEEDER
66h	03h		DOCUMENT MISS FEED AUTOMATIC IN DOCUMENT FEEDER
6Fh	05h	R	DRIVE REGION MUST BE PERMANENT/REGION RESET COUNT ERROR
53h	07h	M	DUPLICATE VOLUME IDENTIFIER
3Fh	0Fh	D T L P W R O M A E B K V F	ECHO BUFFER OVERWRITTEN
3Bh	18h	M	ELEMENT DISABLED
3Bh	19h	M	ELEMENT ENABLED
53h	08h	M	ELEMENT STATUS UNKNOWN
72h	04h	R	EMPTY OR PARTIALLY WRITTEN RESERVED TRACK
34h	00h	D T L P W R O M A E B K V F	ENCLOSURE FAILURE
35h	05h	D T L W R O M A E B K V F	ENCLOSURE SERVICES CHECKSUM ERROR
35h	00h	D T L P W R O M A E B K V F	ENCLOSURE SERVICES FAILURE
35h	03h	D T L P W R O M A E B K V F	ENCLOSURE SERVICES TRANSFER FAILURE
35h	04h	D T L P W R O M A E B K V F	ENCLOSURE SERVICES TRANSFER REFUSED
35h	02h	D T L P W R O M A E B K V F	ENCLOSURE SERVICES UNAVAILABLE
74h	0Ah	T	ENCRYPTED BLOCK NOT RAW READ ENABLED
74h	0Dh	T	ENCRYPTION ALGORITHM DISABLED
74h	09h	T	ENCRYPTION MODE MISMATCH ON READ
74h	07h	T	ENCRYPTION PARAMETERS NOT USEABLE
3Bh	0Fh	R	END OF MEDIUM REACHED
63h	00h	R	END OF USER AREA ENCOUNTERED ON THIS TRACK
00h	05h	T L	END-OF-DATA DETECTED
14h	03h	T	END-OF-DATA NOT FOUND
00h	02h	T	END-OF-PARTITION/MEDIUM DETECTED
51h	00h	D T R O	ERASE FAILURE
51h	01h	D R	ERASE FAILURE - INCOMPLETE ERASE OPERATION DETECTED
00h	18h	T	ERASE OPERATION IN PROGRESS
74h	05h	T	ERROR DECRYPTING DATA
0Dh	00h	D T L P W R O A K	ERROR DETECTED BY THIRD PARTY TEMPORARY INITIATOR
2Ah	0Ah	D T	ERROR HISTORY I_T NEXUS CLEARED
2Ah	0Bh	D T	ERROR HISTORY SNAPSHOT RELEASED
0Ah	00h	D T L P W R O M A E B K V F	ERROR LOG OVERFLOW

Table 55 — ASC and ASCQ assignments (part 6 of 18)

D – Direct Access Block Device (SBC-3)
. T – Sequential Access Device (SSC-4)
. L – Printer Device (SSC)
. P – Processor Device (SPC-2)
. . W – Write Once Block Device (SBC)
. . R – C/DVD Device (MMC-6)
. . O – Optical Memory Block Device (SBC)
. . . M – Media Changer Device (SMC-3)
. . . A – Storage Array Device (SCC-2)
. . . E – SCSI Enclosure Services device (SES-3)
. . . . B – Simplified Direct-Access (Reduced Block) device (RBC)
. . . . K – Optical Card Reader/Writer device (OCRW)
. . . . V – Automation/Device Interface device (ADC-3)
. . . . . F – Object-based Storage Device (OSD-2)

Device Column key
blank = code not used
not blank = code used

ASC	ASCQ	D	T	L	P	W	R	O	M	A	E	B	K	V	F	Description
11h	10h						R									ERROR READING ISRC NUMBER
11h	0Fh						R									ERROR READING UPC/EAN NUMBER
2Ah	0Ch														F	ERROR RECOVERY ATTRIBUTES HAVE CHANGED
11h	02h	D	T			W	R	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
67h	04h									A						EXCHANGE OF LOGICAL UNIT FAILED
00h	20h	D	T		P							B				EXTENDED COPY INFORMATION AVAILABLE
74h	6Fh		T													EXTERNAL DATA ENCRYPTION CONTROL ERROR
74h	6Eh		T													EXTERNAL DATA ENCRYPTION CONTROL TIMEOUT
74h	61h													V		EXTERNAL DATA ENCRYPTION KEY MANAGER ACCESS ERROR
74h	62h													V		EXTERNAL DATA ENCRYPTION KEY MANAGER ERROR
74h	63h													V		EXTERNAL DATA ENCRYPTION KEY NOT FOUND
74h	64h													V		EXTERNAL DATA ENCRYPTION REQUEST NOT AUTHORIZED
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	O	M	A	E	B	K	V	F	FAILURE PREDICTION THRESHOLD EXCEEDED
5Dh	FFh	D	T	L	P	W	R	O	M	A	E	B	K	V	F	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
28h	02h						R									FORMAT-LAYER MAY HAVE CHANGED
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

Table 55 — ASC and ASCQ assignments (part 7 of 18)

		D – Direct Access Block Device (SBC-3) . T – Sequential Access Device (SSC-4) . L – Printer Device (SSC) . P – Processor Device (SPC-2) . . W – Write Once Block Device (SBC) . . R – C/DVD Device (MMC-6) . . O – Optical Memory Block Device (SBC) . . . M – Media Changer Device (SMC-3) . . . A – Storage Array Device (SCC-2) . . . E – SCSI Enclosure Services device (SES-3) . . . . B – Simplified Direct-Access (Reduced Block) device (RBC) . . . . K – Optical Card Reader/Writer device (OCRW) . . . . V – Automation/Device Interface device (ADC-3) . . . . . F – Object-based Storage Device (OSD-2)	Device Column key blank = code not used not blank = code used
ASC	ASCQ	D T L P W R O M A E B K V F	Description
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 O M A E B K V F	I/O PROCESS TERMINATED
10h	00h	D W O B K	ID CRC OR ECC ERROR
5Eh	03h	D T L P W R O A K	IDLE CONDITION ACTIVATED BY COMMAND
5Eh	01h	D T L P W R O A K	IDLE CONDITION ACTIVATED BY TIMER
5Eh	06h	D T L P W R O A K	IDLE_B CONDITION ACTIVATED BY COMMAND
5Eh	05h	D T L P W R O A K	IDLE_B CONDITION ACTIVATED BY TIMER
5Eh	08h	D T L P W R O A K	IDLE_C CONDITION ACTIVATED BY COMMAND
5Eh	07h	D T L P W R O A K	IDLE_C CONDITION ACTIVATED BY TIMER
20h	0Ch	T	ILLEGAL COMMAND WHEN NOT IN APPEND-ONLY MODE
20h	06h	T	ILLEGAL COMMAND WHILE IN EXPLICIT ADDRESS MODE
20h	07h	T	ILLEGAL COMMAND WHILE IN IMPLICIT ADDRESS MODE
20h	04h	T	ILLEGAL COMMAND WHILE IN WRITE CAPABLE STATE
22h	00h	D	ILLEGAL FUNCTION (USE 20 00, 24 00, OR 26 00)
64h	00h	R	ILLEGAL MODE FOR THIS TRACK
2Ch	05h	B	ILLEGAL POWER CONDITION REQUEST
2Ah	07h	D T L P W R O M A E B K V F	IMPLICIT ASYMMETRIC ACCESS STATE TRANSITION FAILED
28h	01h	D T W R O M B	IMPORT OR EXPORT ELEMENT ACCESSED
28h	03h	M	IMPORT/EXPORT ELEMENT ACCESSED, MEDIUM CHANGED
30h	00h	D T W R O M B K	INCOMPATIBLE MEDIUM INSTALLED
30h	12h	M	INCOMPATIBLE VOLUME QUALIFIER
30h	11h	M	INCOMPATIBLE VOLUME TYPE
11h	08h	T	INCOMPLETE BLOCK READ
26h	11h	T	INCOMPLETE KEY-ASSOCIATED DATA SET
0Ch	10h	D	INCOMPLETE MULTIPLE ATOMIC WRITE OPERATIONS
0Dh	03h	D T L P W R O A K	INCORRECT COPY TARGET DEVICE TYPE
74h	03h	T	INCORRECT DATA ENCRYPTION KEY
74h	0Bh	T	INCORRECT ENCRYPTION PARAMETERS
0Eh	02h	D T P W R O M A E B K F	INFORMATION UNIT TOO LONG
0Eh	01h	D T P W R O M A E B K F	INFORMATION UNIT TOO SHORT
47h	03h	D T L P W R O M A E B K V F	INFORMATION UNIT iuCRC ERROR DETECTED
6Ah	00h	A	INFORMATIONAL, REFER TO LOG
48h	00h	D T L P W R O M A E B K V F	INITIATOR DETECTED ERROR MESSAGE RECEIVED

Table 55 — ASC and ASCQ assignments (part 8 of 18)

D – Direct Access Block Device (SBC-3)
. T – Sequential Access Device (SSC-4)
. L – Printer Device (SSC)
. P – Processor Device (SPC-2)
. . W – Write Once Block Device (SBC)
. . R – C/DVD Device (MMC-6)
. . O – Optical Memory Block Device (SBC)
. . . M – Media Changer Device (SMC-3)
. . . A – Storage Array Device (SCC-2)
. . . E – SCSI Enclosure Services device (SES-3)
. . . . B – Simplified Direct-Access (Reduced Block) device (RBC)
. . . . K – Optical Card Reader/Writer device (OCRW)
. . . . V – Automation/Device Interface device (ADC-3)
. . . . . F – Object-based Storage Device (OSD-2)
. . . . .

Device Column key
blank = code not used
not blank = code used

ASC	ASCQ	D	T	L	P	W	R	O	M	A	E	B	K	V	F	Description
4Bh	06h	D			P	W	R	O	M	A	E	B	K			INITIATOR RESPONSE TIMEOUT
26h	0Bh	D	T	L	P	W	R	O					K			INLINE DATA LENGTH EXCEEDED
3Fh	03h	D	T	L	P	W	R	O	M	A	E	B	K	V	F	INQUIRY DATA HAS CHANGED
3Fh	15h	D	T	L	P	W	R		M	A	E	B	K			INSPECT REFERRALS SENSE DESCRIPTORS
55h	05h	D			P	W	R	O	M	A	E	B	K			INSUFFICIENT ACCESS CONTROL RESOURCES
6Fh	06h						R									INSUFFICIENT BLOCK COUNT FOR BINDING NONCE RECORDING
55h	0Bh	D	T	L	P	W	R	O	M	A	E	B	K	V	F	INSUFFICIENT POWER FOR OPERATION
55h	04h	D	T	L	P	W	R	O	M	A	E		K			INSUFFICIENT REGISTRATION RESOURCES
55h	02h	D	T	L	P	W	R	O	M	A	E		K			INSUFFICIENT RESERVATION RESOURCES
55h	03h	D	T	L	P	W	R	O	M	A	E		K			INSUFFICIENT RESOURCES
55h	0Ch	D	T		P							B				INSUFFICIENT RESOURCES TO CREATE ROD
55h	0Dh	D	T		P							B				INSUFFICIENT RESOURCES TO CREATE ROD TOKEN
2Eh	00h	D				W	R	O	M			B				INSUFFICIENT TIME FOR OPERATION
55h	0Eh	D														INSUFFICIENT ZONE RESOURCES
44h	00h	D	T	L	P	W	R	O	M	A	E	B	K	V	F	INTERNAL TARGET FAILURE
21h	02h						R									INVALID ADDRESS FOR WRITE
3Dh	00h	D	T	L	P	W	R	O	M	A	E		K			INVALID BITS IN IDENTIFY MESSAGE
2Ch	02h															INVALID COMBINATION OF WINDOWS SPECIFIED
20h	00h	D	T	L	P	W	R	O	M	A	E	B	K	V	F	INVALID COMMAND OPERATION CODE
26h	0Fh														F	INVALID DATA-OUT BUFFER INTEGRITY CHECK VALUE
21h	01h	D	T			W	R	O	M			B	K			INVALID ELEMENT ADDRESS
24h	00h	D	T	L	P	W	R	O	M	A	E	B	K	V	F	INVALID FIELD IN CDB
0Eh	03h	D	T		P		R		M	A	E	B	K		F	INVALID FIELD IN COMMAND INFORMATION UNIT
26h	00h	D	T	L	P	W	R	O	M	A	E	B	K	V	F	INVALID FIELD IN PARAMETER LIST
0Eh	00h	D	T		P	W	R	O	M	A	E	B	K		F	INVALID INFORMATION UNIT
49h	00h	D	T	L	P	W	R	O	M	A	E	B	K	V	F	INVALID MESSAGE ERROR
26h	0Ch	D	T	L	P	W	R	O					K			INVALID OPERATION FOR COPY SOURCE OR DESTINATION
64h	01h						R									INVALID PACKET SIZE
26h	0Eh	D	T		P	W	R	O	M	A	E	B	K			INVALID PARAMETER WHILE PORT IS ENABLED
26h	04h	D	T	L	P	W	R	O	M	A	E	B	K	V	F	INVALID RELEASE OF PERSISTENT RESERVATION
74h	12h	D	T				R		M	A	E	B	K	V		INVALID SA USAGE
4Bh	01h	D	T		P	W	R	O	M	A	E	B	K			INVALID TARGET PORT TRANSFER TAG RECEIVED
23h	00h	D	T		P							B				INVALID TOKEN OPERATION, CAUSE NOT REPORTABLE
23h	0Ah	D	T		P							B				INVALID TOKEN OPERATION, INVALID TOKEN LENGTH
23h	03h	D	T		P							B				INVALID TOKEN OPERATION, REMOTE ROD TOKEN CREATION NOT SUPPORTED
23h	02h	D	T		P							B				INVALID TOKEN OPERATION, REMOTE TOKEN USAGE NOT SUPPORTED
23h	08h	D	T		P							B				INVALID TOKEN OPERATION, TOKEN CANCELLED
23h	05h	D	T		P							B				INVALID TOKEN OPERATION, TOKEN CORRUPT
23h	09h	D	T		P							B				INVALID TOKEN OPERATION, TOKEN DELETED
23h	07h	D	T		P							B				INVALID TOKEN OPERATION, TOKEN EXPIRED
23h	06h	D	T		P							B				INVALID TOKEN OPERATION, TOKEN REVOKED
23h	04h	D	T		P							B				INVALID TOKEN OPERATION, TOKEN UNKNOWN

Table 55 — ASC and ASCQ assignments (part 9 of 18)

D – Direct Access Block Device (SBC-3)
. T – Sequential Access Device (SSC-4)
. L – Printer Device (SSC)
. P – Processor Device (SPC-2)
. . W – Write Once Block Device (SBC)
. . R – C/DVD Device (MMC-6)
. . O – Optical Memory Block Device (SBC)
. . . M – Media Changer Device (SMC-3)
. . . A – Storage Array Device (SCC-2)
. . . E – SCSI Enclosure Services device (SES-3)
. . . . B – Simplified Direct-Access (Reduced Block) device (RBC)
. . . . K – Optical Card Reader/Writer device (OCRW)
. . . . V – Automation/Device Interface device (ADC-3)
. . . . . F – Object-based Storage Device (OSD-2)

Device Column key
blank = code not used
not blank = code used

ASC	ASCQ	DTLPWROMAEBKVF	Description
23h	01h	DT P B	INVALID TOKEN OPERATION, UNSUPPORTED TOKEN TYPE
21h	03h	R	INVALID WRITE CROSSING LAYER JUMP
24h	08h	DT R MAEBKV	INVALID XCDB
29h	07h	DTLPWROMAEBKVF	I_T NEXUS LOSS OCCURRED
11h	05h	WRO B	L-EC UNCORRECTABLE ERROR
60h	00h		LAMP FAILURE
14h	07h	T	LOCATE OPERATION FAILURE
00h	19h	T	LOCATE OPERATION IN PROGRESS
5Bh	02h	DTLPWROM K	LOG COUNTER AT MAXIMUM
5Bh	00h	DTLPWROM K	LOG EXCEPTION
5Bh	03h	DTLPWROM K	LOG LIST CODES EXHAUSTED
2Ah	02h	DTL WROMAE K	LOG PARAMETERS CHANGED
21h	00h	DT WRO BK	LOGICAL BLOCK ADDRESS OUT OF RANGE
10h	02h	DT W O	LOGICAL BLOCK APPLICATION TAG CHECK FAILED
10h	01h	DT W O	LOGICAL BLOCK GUARD CHECK FAILED
10h	04h	T	LOGICAL BLOCK PROTECTION ERROR ON RECOVER BUFFERED DATA
10h	05h	T	LOGICAL BLOCK PROTECTION METHOD ERROR
10h	03h	DT W O	LOGICAL BLOCK REFERENCE TAG CHECK FAILED
74h	71h	DT R M E V	LOGICAL UNIT ACCESS NOT AUTHORIZED
08h	03h	DT ROM BK	LOGICAL UNIT COMMUNICATION CRC ERROR (ULTRA-DMA/32)
08h	00h	DTL WROMAEBKVF	LOGICAL UNIT COMMUNICATION FAILURE
08h	02h	DTL WROMAEBKVF	LOGICAL UNIT COMMUNICATION PARITY ERROR
08h	01h	DTL WROMAEBKVF	LOGICAL UNIT COMMUNICATION TIME-OUT
05h	00h	DTL WROMAEBKVF	LOGICAL UNIT DOES NOT RESPOND TO SELECTION
4Ch	00h	DTLPWROMAEBKVF	LOGICAL UNIT FAILED SELF-CONFIGURATION
3Eh	03h	DTLPWROMAEBKVF	LOGICAL UNIT FAILED SELF-TEST
3Eh	01h	DTLPWROMAEBKVF	LOGICAL UNIT FAILURE
5Dh	02h	R	LOGICAL UNIT FAILURE PREDICTION THRESHOLD EXCEEDED
3Eh	00h	DTLPWROMAEBKVF	LOGICAL UNIT HAS NOT SELF-CONFIGURED YET
04h	01h	DTLPWROMAEBKVF	LOGICAL UNIT IS IN PROCESS OF BECOMING READY
04h	0Ah	DTLPWROMAEBKVF	LOGICAL UNIT NOT ACCESSIBLE, ASYMMETRIC ACCESS STATE TRANSITION
04h	0Bh	DTLPWROMAEBKVF	LOGICAL UNIT NOT ACCESSIBLE, TARGET PORT IN STANDBY STATE
04h	0Ch	DTLPWROMAEBKVF	LOGICAL UNIT NOT ACCESSIBLE, TARGET PORT IN UNAVAILABLE STATE
68h	00h	A	LOGICAL UNIT NOT CONFIGURED
04h	18h	M	LOGICAL UNIT NOT READY, A DOOR IS OPEN
04h	1Ch	DT MAEB	LOGICAL UNIT NOT READY, ADDITIONAL POWER USE NOT YET GRANTED
04h	10h	DT WROM B	LOGICAL UNIT NOT READY, AUXILIARY MEMORY NOT ACCESSIBLE
04h	17h	M	LOGICAL UNIT NOT READY, CALIBRATION REQUIRED
04h	00h	DTLPWROMAEBKVF	LOGICAL UNIT NOT READY, CAUSE NOT REPORTABLE
04h	1Dh	D	LOGICAL UNIT NOT READY, CONFIGURATION IN PROGRESS
04h	16h	M	LOGICAL UNIT NOT READY, CONFIGURATION REQUIRED

Table 55 — ASC and ASCQ assignments (part 10 of 18)

D – Direct Access Block Device (SBC-3)
. T – Sequential Access Device (SSC-4)
. L – Printer Device (SSC)
. P – Processor Device (SPC-2)
. . W – Write Once Block Device (SBC)
. . R – C/DVD Device (MMC-6)
. . O – Optical Memory Block Device (SBC)
. . . M – Media Changer Device (SMC-3)
. . . A – Storage Array Device (SCC-2)
. . . E – SCSI Enclosure Services device (SES-3)
. . . . B – Simplified Direct-Access (Reduced Block) device (RBC)
. . . . K – Optical Card Reader/Writer device (OCRW)
. . . . V – Automation/Device Interface device (ADC-3)
. . . . . F – Object-based Storage Device (OSD-2)

Device Column key
blank = code not used
not blank = code used

ASC	ASCQ	DTLPWROMAEBKVF	Description
04h	04h	DTL RO B	LOGICAL UNIT NOT READY, FORMAT IN PROGRESS
04h	21h	DTLPWROMAEBKVF	LOGICAL UNIT NOT READY, HARD RESET REQUIRED
04h	02h	DTLPWROMAEBKVF	LOGICAL UNIT NOT READY, INITIALIZING COMMAND REQUIRED
04h	20h	DTLPWROMAEBKVF	LOGICAL UNIT NOT READY, LOGICAL UNIT RESET REQUIRED
04h	08h	R	LOGICAL UNIT NOT READY, LONG WRITE IN PROGRESS
04h	03h	DTLPWROMAEBKVF	LOGICAL UNIT NOT READY, MANUAL INTERVENTION REQUIRED
04h	1Eh	D	LOGICAL UNIT NOT READY, MICROCODE ACTIVATION REQUIRED
04h	1Fh	DTLPWROMAEBKVF	LOGICAL UNIT NOT READY, MICROCODE DOWNLOAD REQUIRED
04h	11h	DT WRO AEB VF	LOGICAL UNIT NOT READY, NOTIFY (ENABLE SPINUP) REQUIRED
04h	12h	M V	LOGICAL UNIT NOT READY, OFFLINE
04h	19h	M	LOGICAL UNIT NOT READY, OPERATING IN SEQUENTIAL MODE
04h	07h	DTLPWROMAEBKVF	LOGICAL UNIT NOT READY, OPERATION IN PROGRESS
04h	22h	DTLPWROMAEBKVF	LOGICAL UNIT NOT READY, POWER CYCLE REQUIRED
04h	05h	DT W O A BK F	LOGICAL UNIT NOT READY, REBUILD IN PROGRESS
04h	06h	DT W O A BK	LOGICAL UNIT NOT READY, RECALCULATION IN PROGRESS
04h	15h	M	LOGICAL UNIT NOT READY, ROBOTICS DISABLED
04h	13h	DT R MAEBKV	LOGICAL UNIT NOT READY, SA CREATION IN PROGRESS
04h	1Bh	D B	LOGICAL UNIT NOT READY, SANITIZE IN PROGRESS
04h	0Eh	DT R MAEBKVF	LOGICAL UNIT NOT READY, SECURITY SESSION IN PROGRESS
04h	09h	DTLPWROMAEBKVF	LOGICAL UNIT NOT READY, SELF-TEST IN PROGRESS
04h	14h	D B	LOGICAL UNIT NOT READY, SPACE ALLOCATION IN PROGRESS
04h	1Ah	D B	LOGICAL UNIT NOT READY, START STOP UNIT COMMAND IN PROGRESS
04h	0Dh	F	LOGICAL UNIT NOT READY, STRUCTURE CHECK REQUIRED
25h	00h	DTLPWROMAEBKVF	LOGICAL UNIT NOT SUPPORTED
27h	02h	DT WRO BK	LOGICAL UNIT SOFTWARE WRITE PROTECTED
00h	1Fh	DT B	LOGICAL UNIT TRANSITIONING TO ANOTHER POWER CONDITION
3Eh	04h	DTLPWROMAEBKVF	LOGICAL UNIT UNABLE TO UPDATE SELF-TEST LOG
5Eh	00h	DTLPWRO A K	LOW POWER CONDITION ON
55h	08h	T	MAXIMUM NUMBER OF SUPPLEMENTAL DECRYPTION KEYS EXCEEDED
15h	01h	DTL WROM BK	MECHANICAL POSITIONING ERROR
3Bh	16h	R	MECHANICAL POSITIONING OR CHANGER ERROR
5Dh	01h	R B	MEDIA FAILURE PREDICTION THRESHOLD EXCEEDED
53h	00h	DTL WROM BK	MEDIA LOAD OR EJECT FAILED
6Fh	04h	R	MEDIA REGION CODE IS MISMATCHED TO LOGICAL UNIT REGION
3Fh	11h	DT WROM B	MEDIUM AUXILIARY MEMORY ACCESSIBLE
55h	09h	M	MEDIUM AUXILIARY MEMORY NOT ACCESSIBLE
3Bh	0Dh	DT WROM BK	MEDIUM DESTINATION ELEMENT FULL
31h	00h	DT WRO BK	MEDIUM FORMAT CORRUPTED
3Fh	10h	DT WROM B	MEDIUM LOADABLE
3Bh	13h	DT WROM BK	MEDIUM MAGAZINE INSERTED
3Bh	14h	DT WROM BK	MEDIUM MAGAZINE LOCKED
3Bh	11h	DT WROM BK	MEDIUM MAGAZINE NOT ACCESSIBLE

Table 55 — ASC and ASCQ assignments (part 11 of 18)

D – Direct Access Block Device (SBC-3)
. T – Sequential Access Device (SSC-4)
. L – Printer Device (SSC)
. P – Processor Device (SPC-2)
. . W – Write Once Block Device (SBC)
. . R – C/DVD Device (MMC-6)
. . O – Optical Memory Block Device (SBC)
. . . M – Media Changer Device (SMC-3)
. . . A – Storage Array Device (SCC-2)
. . . E – SCSI Enclosure Services device (SES-3)
. . . . B – Simplified Direct-Access (Reduced Block) device (RBC)
. . . . K – Optical Card Reader/Writer device (OCRW)
. . . . V – Automation/Device Interface device (ADC-3)
. . . . . F – Object-based Storage Device (OSD-2)

Device Column key
blank = code not used
not blank = code used

ASC	ASCQ	D	T	L	P	W	R	O	M	A	E	B	K	V	F	Description
3Bh	12h	D				W		O	M			B	K			MEDIUM MAGAZINE REMOVED
3Bh	15h	D				W		O	M			B	K			MEDIUM MAGAZINE UNLOCKED
30h	10h						R									MEDIUM NOT FORMATTED
3Ah	00h	D	T	L		W		O	M			B	K			MEDIUM NOT PRESENT
3Ah	03h	D	T			W		O	M			B				MEDIUM NOT PRESENT - LOADABLE
3Ah	04h	D	T			W		O				B				MEDIUM NOT PRESENT - MEDIUM AUXILIARY MEMORY ACCESSIBLE
3Ah	01h	D	T			W		O	M			B	K			MEDIUM NOT PRESENT - TRAY CLOSED
3Ah	02h	D	T			W		O	M			B	K			MEDIUM NOT PRESENT - TRAY OPEN
53h	02h	D	T			W		O	M			B	K			MEDIUM REMOVAL PREVENTED
53h	03h								M							MEDIUM REMOVAL PREVENTED BY DATA TRANSFER ELEMENT
2Ah	15h		T						M					V		MEDIUM REMOVAL PREVENTION PREEMPTED
3Bh	0Eh	D	T			W		O	M			B	K			MEDIUM SOURCE ELEMENT EMPTY
53h	04h		T													MEDIUM THREAD OR UNTHREAD FAILURE
43h	00h	D	T	L	P	W	R	O	M	A	E	B	K	V	F	MESSAGE ERROR
3Fh	01h	D	T	L	P	W	R	O	M	A	E	B	K	V	F	MICROCODE HAS BEEN CHANGED
3Fh	16h	D	T		P	W	R	O	M	A	E	B	K	V	F	MICROCODE HAS BEEN CHANGED WITHOUT RESET
1Dh	00h	D	T			W		O				B	K			MISCOMPARE DURING VERIFY OPERATION
1Dh	01h	D										B				MISCOMPARE VERIFY OF UNMAPPED LBA
11h	0Ah	D	T					O				B	K			MISCORRECTED ERROR
2Ah	01h	D	T	L		W	R	O	M	A	E	B	K	V	F	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	O	M			B	K			MULTIPLE PERIPHERAL DEVICES SELECTED
11h	03h	D	T			W		O				B	K			MULTIPLE READ ERRORS
0Ch	0Eh	D	T			W		O				B	K			MULTIPLE WRITE ERRORS
67h	09h									A						MULTIPLY ASSIGNED LOGICAL UNIT
4Bh	04h	D	T		P	W	R	O	M	A	E	B	K			NAK RECEIVED
00h	00h	D	T	L	P	W	R	O	M	A	E	B	K	V	F	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	07h						R									NO MORE TEST ZONE EXTENSIONS ARE ALLOWED
72h	05h						R									NO MORE TRACK RESERVATIONS ALLOWED
06h	00h	D				W		O	M			B	K			NO REFERENCE POSITION FOUND
02h	00h	D				W		O				B	K			NO SEEK COMPLETE
03h	01h		T													NO WRITE CURRENT
24h	06h														F	NONCE NOT UNIQUE
24h	07h														F	NONCE TIMESTAMP OUT OF RANGE
28h	00h	D	T	L	P	W	R	O	M	A	E	B	K	V	F	NOT READY TO READY CHANGE, MEDIUM MAY HAVE CHANGED
2Ch	0Bh		T													NOT RESERVED
00h	16h	D	T	L	P	W	R	O	M	A	E	B	K	V	F	OPERATION IN PROGRESS
5Ah	01h	D	T			W		O	M			B	K			OPERATOR MEDIUM REMOVAL REQUEST
5Ah	00h	D	T	L	P	W		O				B	K			OPERATOR REQUEST OR STATE CHANGE INPUT

Table 55 — ASC and ASCQ assignments (part 12 of 18)

D – Direct Access Block Device (SBC-3)
. T – Sequential Access Device (SSC-4)
. L – Printer Device (SSC)
. P – Processor Device (SPC-2)
. . W – Write Once Block Device (SBC)
. . R – C/DVD Device (MMC-6)
. . O – Optical Memory Block Device (SBC)
. . . M – Media Changer Device (SMC-3)
. . . A – Storage Array Device (SCC-2)
. . . E – SCSI Enclosure Services device (SES-3)
. . . . B – Simplified Direct-Access (Reduced Block) device (RBC)
. . . . K – Optical Card Reader/Writer device (OCRW)
. . . . V – Automation/Device Interface device (ADC-3)
. . . . . F – Object-based Storage Device (OSD-2)

Device Column key
blank = code not used
not blank = code used

ASC	ASCQ	D	T	L	P	W	R	O	M	A	E	B	K	V	F	Description
5Ah	03h	D				W	R	O		A		B	K			OPERATOR SELECTED WRITE PERMIT
5Ah	02h	D				W	R	O		A		B	K			OPERATOR SELECTED WRITE PROTECT
2Ch	0Ch	D														ORWRITE GENERATION DOES NOT MATCH
61h	02h															OUT OF FOCUS
4Eh	00h	D	T	L	P	W	R	O	M	A	E	B	K	V	F	OVERLAPPED COMMANDS ATTEMPTED
2Dh	00h		T													OVERWRITE ERROR ON UPDATE IN PLACE
20h	05h		T													Obsolete
24h	02h		T													Obsolete
24h	03h		T													Obsolete
63h	01h						R									PACKET DOES NOT FIT IN AVAILABLE SPACE
3Bh	05h			L												PAPER JAM
1Ah	00h	D	T	L	P	W	R	O	M	A	E	B	K	V	F	PARAMETER LIST LENGTH ERROR
26h	01h	D	T	L	P	W	R	O	M	A	E	B	K	V	F	PARAMETER NOT SUPPORTED
26h	02h	D	T	L	P	W	R	O	M	A	E	B	K	V	F	PARAMETER VALUE INVALID
2Ah	00h	D	T	L		W	R	O	M	A	E	B	K	V	F	PARAMETERS CHANGED
69h	02h									A						PARITY/DATA MISMATCH
1Fh	00h	D						O					K			PARTIAL DEFECT LIST TRANSFER
2Ch	0Ah														F	PARTITION OR COLLECTION CONTAINS USER OBJECTS
4Bh	14h	D	T		P	W	R	O	M	A	E	B	K		F	PCIE ACS VIOLATION
4Bh	10h	D	T		P	W	R	O	M	A	E	B	K		F	PCIE COMPLETER ABORT
4Bh	0Fh	D	T		P	W	R	O	M	A	E	B	K		F	PCIE COMPLETION TIMEOUT
4Bh	12h	D	T		P	W	R	O	M	A	E	B	K		F	PCIE ECRC CHECK FAILED
4Bh	0Eh	D	T		P	W	R	O	M	A	E	B	K		F	PCIE FABRIC ERROR
4Bh	11h	D	T		P	W	R	O	M	A	E	B	K		F	PCIE POISONED TLP RECEIVED
4Bh	15h	D	T		P	W	R	O	M	A	E	B	K		F	PCIE TLP PREFIX BLOCKED
4Bh	13h	D	T		P	W	R	O	M	A	E	B	K		F	PCIE UNSUPPORTED REQUEST
03h	00h	D	T	L		W	R	O				B	K			PERIPHERAL DEVICE WRITE FAULT
27h	05h		T				R									PERMANENT WRITE PROTECT
2Ch	06h						R									PERSISTENT PREVENT CONFLICT
44h	01h	D	T		P				M	A	E	B	K	V	F	PERSISTENT RESERVATION INFORMATION LOST
27h	04h		T				R									PERSISTENT WRITE PROTECT
47h	06h	D	T						M	A	E	B	K	V	F	PHY TEST FUNCTION IN PROGRESS
50h	02h		T													POSITION ERROR RELATED TO TIMING
3Bh	0Ch		T													POSITION PAST BEGINNING OF MEDIUM
3Bh	0Bh															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	O	M	A	E	B	K	V	F	POWER ON OCCURRED
29h	00h	D	T	L	P	W	R	O	M	A	E	B	K	V	F	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

Table 55 — ASC and ASCQ assignments (part 13 of 18)

D – Direct Access Block Device (SBC-3)
. T – Sequential Access Device (SSC-4)
. L – Printer Device (SSC)
. P – Processor Device (SPC-2)
. . W – Write Once Block Device (SBC)
. . R – C/DVD Device (MMC-6)
. . O – Optical Memory Block Device (SBC)
. . . M – Media Changer Device (SMC-3)
. . . A – Storage Array Device (SCC-2)
. . . E – SCSI Enclosure Services device (SES-3)
. . . . B – Simplified Direct-Access (Reduced Block) device (RBC)
. . . . K – Optical Card Reader/Writer device (OCRW)
. . . . V – Automation/Device Interface device (ADC-3)
. . . . . F – Object-based Storage Device (OSD-2)

Device Column key
blank = code not used
not blank = code used

ASC	ASCQ	D	T	L	P	W	R	O	M	A	E	B	K	V	F	Description
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)
2Ch	07h	D	T	L	P	W	R	O	M	A	E	B	K	V	F	PREVIOUS BUSY STATUS
2Ch	09h	D	T	L	P	W	R	O	M		E	B	K	V	F	PREVIOUS RESERVATION CONFLICT STATUS
2Ch	08h	D	T	L	P	W	R	O	M	A	E	B	K	V	F	PREVIOUS TASK SET FULL STATUS
1Ch	01h	D						O				B	K			PRIMARY DEFECT LIST NOT FOUND
2Ah	08h	D	T			W	R	O	M	A	E	B	K	V	F	PRIORITY CHANGED
73h	05h						R									PROGRAM MEMORY AREA IS FULL
73h	04h						R									PROGRAM MEMORY AREA UPDATE FAILURE
00h	07h		T													PROGRAMMABLE EARLY WARNING DETECTED
47h	05h	D	T	L	P	W	R	O	M	A	E	B	K	V	F	PROTOCOL SERVICE CRC ERROR
55h	07h														F	QUOTA ERROR
40h	00h	D														RAM FAILURE (SHOULD USE 40 NN)
15h	00h	D	T	L		W	R	O	M			B	K			RANDOM POSITIONING ERROR
73h	17h						R									RDZ IS FULL
21h	07h	D														READ BOUNDARY VIOLATION
11h	13h	D	T	L	P	W	R	O		A	E	B	K	V	F	READ ERROR - FAILED RETRANSMISSION REQUEST
11h	14h	D														READ ERROR - LBA MARKED BAD BY APPLICATION CLIENT
11h	11h						R									READ ERROR - LOSS OF STREAMING
6Fh	03h						R									READ OF SCRAMBLED SECTOR WITHOUT AUTHENTICATION
3Bh	0Ah															READ PAST BEGINNING OF MEDIUM
3Bh	09h															READ PAST END OF MEDIUM
3Bh	17h														F	READ PAST END OF USER OBJECT
11h	01h	D	T			W	R	O				B	K			READ RETRIES EXHAUSTED
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	O				B	K			RECORDED ENTITY NOT FOUND
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
18h	08h						R									RECOVERED DATA WITH LINKING
17h	03h	D	T			W	R	O				B	K			RECOVERED DATA WITH NEGATIVE HEAD OFFSET
17h	00h	D	T			W	R	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

Table 55 — ASC and ASCQ assignments (part 14 of 18)

D – Direct Access Block Device (SBC-3)
. T – Sequential Access Device (SSC-4)
. L – Printer Device (SSC)
. P – Processor Device (SPC-2)
. . W – Write Once Block Device (SBC)
. . R – C/DVD Device (MMC-6)
. . O – Optical Memory Block Device (SBC)
. . . M – Media Changer Device (SMC-3)
. . . A – Storage Array Device (SCC-2)
. . . E – SCSI Enclosure Services device (SES-3)
. . . . B – Simplified Direct-Access (Reduced Block) device (RBC)
. . . . K – Optical Card Reader/Writer device (OCRW)
. . . . V – Automation/Device Interface device (ADC-3)
. . . . . F – Object-based Storage Device (OSD-2)

Device Column key
blank = code not used
not blank = code used

ASC	ASCQ	D	T	L	P	W	R	O	M	A	E	B	K	V	F	Description
17h	01h	D				W	R	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	A	E	B				REDUNDANCY GROUP CREATED OR MODIFIED
3Fh	07h	D	T			W	R	O	M	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	O	M	A	E					REGISTRATIONS PREEMPTED
67h	05h									A						REMOVE OF LOGICAL UNIT FAILED
3Fh	0Eh	D	T	L	P	W	R	O	M	A	E					REPORTED LUNS DATA HAS CHANGED
3Bh	08h		T													REPOSITION ERROR
2Ah	03h	D	T	L	P	W	R	O	M	A	E		K			RESERVATIONS PREEMPTED
2Ah	04h	D	T	L	P	W	R	O	M	A	E					RESERVATIONS RELEASED
2Ch	0Dh	D														RESET WRITE POINTER NOT ALLOWED
00h	1Ah		T													REWIND OPERATION IN PROGRESS
36h	00h			L												RIBBON, INK, OR TONER FAILURE
73h	06h						R									RMA/PMA IS ALMOST FULL
72h	06h						R									RMZ EXTENSION IS NOT ALLOWED
37h	00h	D	T	L		W	R	O	M	A	E	B	K	V	F	ROUNDED PARAMETER
5Ch	00h	D						O								RPL STATUS CHANGE
2Ah	14h	D	T				R		M	A	E	B	K	V		SA CREATION CAPABILITIES DATA HAS CHANGED
74h	30h	D	T				R		M	A	E	B	K	V		SA CREATION PARAMETER NOT SUPPORTED
74h	10h	D	T				R		M	A	E	B	K	V		SA CREATION PARAMETER VALUE INVALID
74h	11h	D	T				R		M	A	E	B	K	V		SA CREATION PARAMETER VALUE REJECTED
31h	03h	D										B				SANITIZE COMMAND FAILED
39h	00h	D	T	L		W	R	O	M	A	E		K			SAVING PARAMETERS NOT SUPPORTED
62h	00h															SCAN HEAD POSITIONING ERROR
29h	02h	D	T	L	P	W	R	O	M	A	E	B	K	V	F	SCSI BUS RESET OCCURRED
47h	00h	D	T	L	P	W	R	O	M	A	E	B	K	V	F	SCSI PARITY ERROR
47h	02h	D	T	L	P	W	R	O	M	A	E	B	K	V	F	SCSI PARITY ERROR DETECTED DURING ST DATA PHASE
54h	00h				P											SCSI TO HOST SYSTEM INTERFACE FAILURE
24h	04h														F	SECURITY AUDIT VALUE FROZEN
74h	79h	D														SECURITY CONFLICT IN TRANSLATED DEVICE
74h	00h		T													SECURITY ERROR
24h	05h														F	SECURITY WORKING KEY FROZEN
45h	00h	D	T	L	P	W	R	O	M	A	E	B	K	V	F	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

Table 55 — ASC and ASCQ assignments (part 15 of 18)

		D – Direct Access Block Device (SBC-3) . T – Sequential Access Device (SSC-4) . L – Printer Device (SSC) . P – Processor Device (SPC-2) . . W – Write Once Block Device (SBC) . . R – C/DVD Device (MMC-6) . . O – Optical Memory Block Device (SBC) . . . M – Media Changer Device (SMC-3) . . . A – Storage Array Device (SCC-2) . . . E – SCSI Enclosure Services device (SES-3) . . . . B – Simplified Direct-Access (Reduced Block) device (RBC) . . . . K – Optical Card Reader/Writer device (OCRW) . . . . V – Automation/Device Interface device (ADC-3) . . . . . F – Object-based Storage Device (OSD-2)	Device Column key blank = code not used not blank = code used
ASC	ASCQ	D T L P W R O M A E B K V F	Description
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	1Bh	T	SET CAPACITY OPERATION IN PROGRESS
67h	0Ah	D T L P W R O M A E B K V F	SET TARGET PORT GROUPS COMMAND FAILED
00h	03h	T	SETMARK DETECTED
3Bh	04h	L	SLEW FAILURE
47h	7Fh	D T P W R O M A E B K	SOME COMMANDS CLEARED BY ISCSI PROTOCOL EVENT
2Fh	03h	D T L P W R O M A E B K V F	SOME COMMANDS CLEARED BY QUEUING LAYER EVENT
27h	07h	D B	SPACE ALLOCATION FAILED WRITE PROTECT
5Dh	03h	R	SPARE AREA EXHAUSTION PREDICTION THRESHOLD EXCEEDED
3Fh	08h	D T W R O M A E B	SPARE CREATED OR MODIFIED
3Fh	09h	D T W R O M 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
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 O A K	STANDBY CONDITION ACTIVATED BY COMMAND
5Eh	02h	D T L P W R O A K	STANDBY CONDITION ACTIVATED BY TIMER
5Eh	0Ah	D T L P W R O A K	STANDBY_Y CONDITION ACTIVATED BY COMMAND
5Eh	09h	D T L P W R O A K	STANDBY_Y CONDITION ACTIVATED BY TIMER

Table 55 — ASC and ASCQ assignments (part 16 of 18)

D – Direct Access Block Device (SBC-3)
. T – Sequential Access Device (SSC-4)
. L – Printer Device (SSC)
. P – Processor Device (SPC-2)
. . W – Write Once Block Device (SBC)
. . R – C/DVD Device (MMC-6)
. . O – Optical Memory Block Device (SBC)
. . . M – Media Changer Device (SMC-3)
. . . A – Storage Array Device (SCC-2)
. . . E – SCSI Enclosure Services device (SES-3)
. . . . B – Simplified Direct-Access (Reduced Block) device (RBC)
. . . . K – Optical Card Reader/Writer device (OCRW)
. . . . V – Automation/Device Interface device (ADC-3)
. . . . . F – Object-based Storage Device (OSD-2)
. . . . .

Device Column key
blank = code not used
not blank = code used

ASC	ASCQ	DTLPWROMAEBKVF	Description
6Bh	00h	A	STATE CHANGE HAS OCCURRED
68h	01h	D	SUBSIDIARY LOGICAL UNIT NOT CONFIGURED
1Bh	00h	DTLPWROMAEBKVF	SYNCHRONOUS DATA TRANSFER ERROR
55h	01h	D O BK	SYSTEM BUFFER FULL
55h	00h	P	SYSTEM RESOURCE FAILURE
4Dh	NNh	DTLPWROMAEBKVF	TAGGED OVERLAPPED COMMANDS (NN = TASK 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	DTLPWROMAEBKVF	TARGET OPERATING CONDITIONS HAVE CHANGED
38h	07h	D	THIN PROVISIONING SOFT THRESHOLD REACHED
0Dh	01h	DTLPWRO A K	THIRD PARTY DEVICE FAILURE
5Bh	01h	DTLPWROM K	THRESHOLD CONDITION MET
26h	03h	DTLPWROMAE K	THRESHOLD PARAMETERS NOT SUPPORTED
3Eh	02h	DTLPWROMAEBKVF	TIMEOUT ON LOGICAL UNIT
2Ah	10h	DT M E V	TIMESTAMP CHANGED
3Bh	1Ch	T	TOO MANY LOGICAL OBJECTS ON PARTITION TO SUPPORT OPERATION
26h	08h	DTLPWRO K	TOO MANY SEGMENT DESCRIPTORS
26h	06h	DTLPWRO K	TOO MANY TARGET DESCRIPTORS
2Ch	01h		TOO MANY WINDOWS SPECIFIED
4Bh	02h	DT PWROMAEBK	TOO MUCH WRITE DATA
09h	00h	DT WRO B	TRACK FOLLOWING ERROR
09h	01h	WRO K	TRACKING SERVO FAILURE
29h	06h	DTLPWROMAEBKVF	TRANSCEIVER MODE CHANGED TO LVD
29h	05h	DTLPWROMAEBKVF	TRANSCEIVER MODE CHANGED TO SINGLE-ENDED
61h	01h		UNABLE TO ACQUIRE VIDEO
74h	01h	T	UNABLE TO DECRYPT DATA
74h	0Ch	DT R MAEBKV	UNABLE TO DECRYPT PARAMETER LIST
57h	00h	R	UNABLE TO RECOVER TABLE-OF-CONTENTS
21h	04h	D	UNALIGNED WRITE COMMAND
74h	02h	T	UNENCRYPTED DATA ENCOUNTERED WHILE DECRYPTING
26h	0Ah	DTLPWRO K	UNEXPECTED INEXACT SEGMENT
74h	06h	T	UNKNOWN SIGNATURE VERIFICATION KEY
53h	01h	T	UNLOAD TAPE FAILURE
08h	04h	DTLPWRO K	UNREACHABLE COPY TARGET
11h	00h	DT WRO BK	UNRECOVERED READ ERROR
11h	04h	D W O BK	UNRECOVERED READ ERROR - AUTO REALLOCATE FAILED
11h	0Bh	D W O BK	UNRECOVERED READ ERROR - RECOMMEND REASSIGNMENT
11h	0Ch	D W O BK	UNRECOVERED READ ERROR - RECOMMEND REWRITE THE DATA
46h	00h	DTLPWROM BK	UNSUCCESSFUL SOFT RESET
35h	01h	DTLPWROMAEBKVF	UNSUPPORTED ENCLOSURE FUNCTION
26h	09h	DTLPWRO K	UNSUPPORTED SEGMENT DESCRIPTOR TYPE CODE

Table 55 — ASC and ASCQ assignments (part 17 of 18)

D – Direct Access Block Device (SBC-3)
. T – Sequential Access Device (SSC-4)
. L – Printer Device (SSC)
. P – Processor Device (SPC-2)
. . W – Write Once Block Device (SBC)
. . R – C/DVD Device (MMC-6)
. . O – Optical Memory Block Device (SBC)
. . . M – Media Changer Device (SMC-3)
. . . A – Storage Array Device (SCC-2)
. . . E – SCSI Enclosure Services device (SES-3)
. . . . B – Simplified Direct-Access (Reduced Block) device (RBC)
. . . . K – Optical Card Reader/Writer device (OCRW)
. . . . V – Automation/Device Interface device (ADC-3)
. . . . . F – Object-based Storage Device (OSD-2)
. . . . .

Device Column key
blank = code not used
not blank = code used

ASC	ASCQ	D	T	L	P	W	R	O	M	A	E	B	K	V	F	Description
26h	07h	D	T	L	P	W	R	O					K			UNSUPPORTED TARGET DESCRIPTOR TYPE CODE
59h	00h							O								UPDATED BLOCK READ
26h	12h		T													VENDOR SPECIFIC KEY REFERENCE NOT FOUND
00h	1Ch		T													VERIFY OPERATION IN PROGRESS
09h	05h	D	T				W	R	O			B				VIBRATION INDUCED TRACKING ERROR
61h	00h															VIDEO ACQUISITION ERROR
65h	00h	D	T	L	P	W	R	O	M	A	E	B	K	V	F	VOLTAGE FAULT
53h	05h								M							VOLUME IDENTIFIER INVALID
53h	06h								M							VOLUME IDENTIFIER MISSING
3Fh	0Ah	D	T				W	R	O	M	A	E	B	K		VOLUME SET CREATED OR MODIFIED
3Fh	0Ch	D	T				W	R	O	M	A	E	B	K		VOLUME SET DEASSIGNED
3Fh	0Bh	D	T				W	R	O	M	A	E	B	K		VOLUME SET DELETED
3Fh	0Dh	D	T				W	R	O	M	A	E	B	K		VOLUME SET REASSIGNED
0Bh	00h	D	T	L	P	W	R	O	M	A	E	B	K	V	F	WARNING
0Bh	05h	D	T	L	P	W	R	O		A	E	B	K	V	F	WARNING - BACKGROUND MEDIUM SCAN DETECTED MEDIUM ERROR
0Bh	04h	D	T	L	P	W	R	O		A	E	B	K	V	F	WARNING - BACKGROUND PRE-SCAN DETECTED MEDIUM ERROR
0Bh	03h	D	T	L	P	W	R	O	M	A	E	B	K	V	F	WARNING - BACKGROUND SELF-TEST FAILED
0Bh	07h	D	T	L	P	W	R	O	M	A	E	B	K	V	F	WARNING - DEGRADED POWER TO NON-VOLATILE CACHE
0Bh	09h	D														WARNING - DEVICE STATISTICS NOTIFICATION ACTIVE
0Bh	02h	D	T	L	P	W	R	O	M	A	E	B	K	V	F	WARNING - ENCLOSURE DEGRADED
0Bh	06h	D	T	L	P	W	R	O	M	A	E	B	K	V	F	WARNING - NON-VOLATILE CACHE NOW VOLATILE
0Bh	08h	D	T	L	P	W	R	O	M	A	E	B	K	V	F	WARNING - POWER LOSS EXPECTED
0Bh	01h	D	T	L	P	W	R	O	M	A	E	B	K	V	F	WARNING - SPECIFIED TEMPERATURE EXCEEDED
30h	0Dh		T													WORM MEDIUM - INTEGRITY CHECK
30h	0Ch		T													WORM MEDIUM - OVERWRITE ATTEMPTED
11h	15h	D														WRITE AFTER SANITIZE REQUIRED
50h	00h		T													WRITE APPEND ERROR
50h	01h		T													WRITE APPEND POSITION ERROR
21h	05h	D														WRITE BOUNDARY VIOLATION
0Ch	00h		T				R									WRITE ERROR
0Ch	02h	D				W		O				B	K			WRITE ERROR - AUTO REALLOCATION FAILED
0Ch	09h						R									WRITE ERROR - LOSS OF STREAMING
0Ch	0Dh	D	T	L	P	W	R	O		A	E	B	K	V	F	WRITE ERROR - NOT ENOUGH UNSOLICITED DATA
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
0Ch	0Ch	D	T	L	P	W	R	O		A	E	B	K	V	F	WRITE ERROR - UNEXPECTED UNSOLICITED DATA
27h	00h	D	T				W	R	O			B	K			WRITE PROTECTED
2Ch	0Eh	D														ZONE IS OFFLINE
27h	08h	D														ZONE IS READ ONLY
31h	02h						R									ZONED FORMATTING FAILED DUE TO SPARE LINKING
3Fh	12h	D	T	L	P	W	R		M	A	E	B	K		F	iSCSI IP ADDRESS ADDED

Table 55 — ASC and ASCQ assignments (part 18 of 18)

		D – Direct Access Block Device (SBC-3) . T – Sequential Access Device (SSC-4) . L – Printer Device (SSC) . P – Processor Device (SPC-2) . . W – Write Once Block Device (SBC) . . R – C/DVD Device (MMC-6) . . O – Optical Memory Block Device (SBC) . . . M – Media Changer Device (SMC-3) . . . A – Storage Array Device (SCC-2) . . . E – SCSI Enclosure Services device (SES-3) . . . . B – Simplified Direct-Access (Reduced Block) device (RBC) . . . . K – Optical Card Reader/Writer device (OCRW) . . . . V – Automation/Device Interface device (ADC-3) . . . . . F – Object-based Storage Device (OSD-2)	Device Column key blank = code not used not blank = code used
ASC	ASCQ	D T L P W R O M A E B K V F	Description
3Fh	14h	D T L P W R M A E B K F	iSCSI IP ADDRESS CHANGED
3Fh	13h	D T L P W R M A E B K F	iSCSI IP ADDRESS REMOVED
80h Through FFh	xxh xxh	\ > /	Vendor specific
xxh Through xxh	80h FFh	\ > /	Vendor specific qualification of standard ASC
			All codes not shown are reserved.

5 Model common to all device types

5.1 Introduction to the model common to all device types

5.1.1 Overview

This model describes some of the general characteristics expected of most SCSI devices. This model is not intended to alter any requirements defined elsewhere in SCSI. Devices conforming to this standard shall conform to SAM-5.

5.1.2 Important commands for all SCSI device servers

5.1.2.1 Commands implemented by all SCSI device servers

This standard defines the following mandatory commands (see 6.1):

a) the INQUIRY command (see 5.1.2.3);
b) the REPORT LUNS command (see 5.1.2.4); and
c) the TEST UNIT READY command (see 5.1.2.5).

These commands are used to discover a logical unit's capabilities, to discover the system configuration, and to determine whether a logical unit is ready.

If the device server is able to process commands after an error occurs that prohibits normal command completion, then the device server may include a field replaceable unit code in the sense data when returning CHECK CONDITION status for commands other than the INQUIRY command (see 6.6), the REPORT LUNS command (see 6.33), and the REQUEST SENSE command (see 6.39). If the sense data includes a field replaceable unit code, then an application client may use the INQUIRY command to request the corresponding ASCII Information VPD page (see 7.8.3), if any, which contains ASCII information about the field replaceable unit causing the error.

5.1.2.2 Commands recommended for all SCSI device servers

Support for the REQUEST SENSE command is recommended to provide compatibility with application clients designed to use previous versions of this standard or status polling features defined by command standards.

5.1.2.3 Using the INQUIRY command

The INQUIRY command (see 6.6) may be used by an application client to determine the configuration of a logical unit. Device servers respond with information that includes their device type and standard version and may include the manufacturer's identification, model number and other information.

The Device Identification VPD page (see 7.8.6) returned in response to an INQUIRY command with the EVPD bit set to one and the PAGE CODE field set to 83h contains identifying information for the logical unit, the target port, and the SCSI target device.

The INQUIRY command may return incomplete information until the device serve completes the power on process (see 6.6.1).

5.1.2.4 Using the REPORT LUNS command

The REPORT LUNS command (see 6.33) may be used by an application client to request a logical unit inventory report of the logical units that are accessible to the I_T nexus on which the command is sent.

The application client may select different types of logical unit inventories to be reported. Examples of using different logical unit inventory report types are shown in Annex B.

5.1.2.5 Using the TEST UNIT READY command

The TEST UNIT READY command (see 6.47) allows an application client to poll a logical unit until it is ready without allocating 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.

5.1.2.6 Using the REQUEST SENSE command

The REQUEST SENSE command (see 6.39) may be used by an application client to poll the status of some background operations and to clear interlocked unit attention conditions (see 7.5.8).

5.1.3 Implicit head of queue

Each of the following commands may be processed by the task manager as if the command has a task attribute of HEAD OF QUEUE (see SAM-5) if the command is received with a SIMPLE task attribute or an ORDERED task attribute:

a) INQUIRY; and
b) REPORT LUNS.

See SAM-5 for additional rules on implicit head of queue processing.

5.2 Device clocks and timestamps

A timestamp may be included in data logged or recorded by a device server based on a the contents of a device clock saturating counter described in this subclause.

Device clocks may be managed with:

a) the REPORT TIMESTAMP command (see 6.38);
b) the SET TIMESTAMP command (see 6.46);
c) the Control Extension mode page (see 7.5.9); and
d) methods outside the scope of this standard.

A device clock is initialized by a:

a) power on reset or hard reset that sets a device clock to zero;
b) SET TIMESTAMP command that sets a device clock to a specified value; or
c) method outside the scope of this standard.

The TCMOS bit in the Control Extension mode page (see 7.5.9) specifies whether the device server allows a device clock to be initialized by methods outside the scope of this standard. The SCSIP bit in the Control Extension mode page (see 7.5.9) specifies whether methods outside the scope of this standard take precedence over the SET TIMESTAMP command for initializing a device clock.

After a device clock is initialized, the device server shall increment it by one every millisecond, plus or minus a vendor specific tolerance.

A device clock shall not be affected by an I_T nexus loss or a logical unit reset.

If a device clock is initialized by means other than the SET TIMESTAMP command (see 6.46), the device server shall establish a unit attention condition for the initiator port associated with every I_T nexus (see SAM-5), with the additional sense code set to TIMESTAMP CHANGED.

The format of a device clock value is shown in table 56.

Table 56 — Device clock value format

Bit / Byte	7	6	5	4	3	2	1	0
0	(MSB)							
•••	TIMESTAMP							
5								(LSB)

The TIMESTAMP field contains the value of a device clock.

A data structure that contains a device clock value may indicate the most recent event that initialized that device clock by associating a timestamp origin value (see table 57) with that device clock value.

Table 57 — Timestamp origin value

Code	Description
000b	Device clock initialized to zero at power on or as the result of a hard reset
001b	Reserved
010b	Device clock initialized by the SET TIMESTAMP command (see 6.46)
011b	Device clock initialized by methods outside the scope of this standard
100b to 111b	Reserved

5.3 Device specific background functions

5.3.1 Introduction

Device specific background functions are SCSI target device specific functions that a SCSI target device may perform that have no specific association with application client initiated operations. A device specific background function is initiated if:

a) a device specific event associated with the function occurs (e.g., a timer expires or a counter reaches its limit); and
b) the function is enabled.

If conditions are met to initiate more than one device specific background function (e.g., more than one device specific event occurs), then the order of performing the functions is device specific. Other device server processes (e.g., processing a command) may or may not have any impact on events associated with device specific background functions.

Device specific background functions do not include self-test operations (see 5.14) or background scan operations (see SBC-3).

Device specific background functions:

a) may include background functions for the informational exceptions that are managed using the Informational Exceptions Control mode page (see applicable command standard);

b) may include regular, device specific self testing or saving of device specific data;
c) may require the logical unit to be in a different power condition to be performed (e.g., a logical unit may require being in the active power condition to access the medium for some functions);
d) may be enabled or disabled via the EBF bit in the Informational Exceptions Control mode page;
e) shall be affected by the PERF bit and the LOGERR bit in the Informational Exceptions Control mode page, if the background function is associated with informational exceptions;
f) should be processed relative to power conditions based on the setting in the PM_BG_PRECEDENCE field in the Power Condition mode page (see 7.5.13);
g) may have impact on the SCSI target device's performance (e.g., if a logical unit is performing a background function, and the device server receives a command from an application client that requires access to the logical unit, then the logical unit may take a short period of time (e.g., two seconds) to suspend the background function before the logical unit is able to process the command);
h) shall not affect power condition timers as defined in the Power Condition mode page;
i) shall not affect timers defined in the Background Control mode page (see SBC-3); and
j) shall have no negative impact on the reliability of the logical unit.

5.3.2 Suspending and resuming device specific background functions

The SCSI target device shall suspend a device specific background function in progress if:

a) any of the following are true:
   A) a command or task management function is processed that requires the device specific background function to be suspended; or
   B) a SCSI event (e.g., a reset event) (see SAM-5) occurs that requires the device specific background function to be suspended;
   or
b) all of the following are true:
   A) a power condition timer defined in the Power Condition mode page (see 7.5.13) expires;
   B) the PM_BG_PRECEDENCE field in the Power Condition mode page is set to 10b; and
   C) the SCSI target device is unable to continue performing the device specific background function in the power condition associated with the timer that expired.

The SCSI target device may suspend a device specific background function in progress if:

a) a power condition timer defined in the Power Condition mode page (see 7.5.13) expires;
b) the PM_BG_PRECEDENCE field in the Power Condition mode page is set to 00b; and
c) the SCSI target device is unable to continue performing the device-specific background function in the power condition associated with the timer that expired.

If a device specific background function is suspended, the device server shall not stop any process that causes a device specific background function to be initiated (e.g., not stop any timers or counters associated with device specific background functions).

A suspended device specific background function may be resumed if:

a) there are no commands in the task set to be processed;
b) there are no task management functions to be processed;
c) there are no SCSI events to be processed;
d) no ACA condition (see SAM-5) exists;
e) the PM_BG_PRECEDENCE field in the Power Condition mode page (see 7.5.13) is set to 00b; or
f) the PM_BG_PRECEDENCE field in the Power Condition mode page is set to 10b, and no power condition timer defined in the Power Condition mode page has expired.

5.4 Downloading and activating microcode

5.4.1 Downloading microcode

SCSI target device implementations may use microcode (e.g., firmware) that is stored in nonvolatile storage. Microcode may be changeable by an application client using the WRITE BUFFER command (see 6.49). The WRITE BUFFER command provides multiple methods for downloading microcode to the SCSI target device and activating the microcode.

Downloading microcode involves the following steps:

1) **Download:** the application client transfers complete microcode from the Data-Out buffer to the device server in one or more WRITE BUFFER commands; and
2) **Save:** if defined by the download microcode mode, the device server saves the microcode to nonvolatile storage.

The SCSI target device begins using the new microcode for the first time after it is activated (see 5.4.2) as part of the response to an event defined by the download microcode mode.

After power on or hard reset, the logical unit shall use the last microcode for that logical unit that was saved to nonvolatile storage.

Table 58 defines the WRITE BUFFER download microcode modes with respect to the steps described in this subclause.

Table 58 — WRITE BUFFER download microcode modes

Mode	Down-load [a]	Save [b]	Activate [c]
Download microcode and activate (i.e., 04h)	yes [d]	no	yes
Download microcode, save, and activate (i.e., 05h)	yes [d]	yes	optional
Download microcode with offsets and activate (i.e., 06h)	yes [e]	no	yes
Download microcode with offsets, save, and activate (i.e., 07h)	yes [e]	yes	optional
Download microcode with offsets, save, and defer activate (i.e., 0Eh) [f]	yes [e]	yes	no
Download microcode with offsets, select activation events, save, and defer activate (i.e., 0Dh) [f]	yes [e]	yes	no
Activate deferred microcode (i.e., 0Fh) [g]	no	no	yes

[a] Entries in the Download column are as follows. For modes labeled yes, the application client delivers microcode in the WRITE BUFFER command(s). For modes labeled no, the application client does not deliver microcode with the WRITE BUFFER command.

[b] Entries in the Save column are as follows. For modes labeled yes, the device server shall save the microcode to nonvolatile storage for use after each subsequent power on or hard reset, and shall not return GOOD status for the final command in the WRITE BUFFER sequence (i.e., the series of WRITE BUFFER commands that downloads the microcode) until the microcode has been saved. For modes labeled no, the device server shall discard the microcode on the next power on or hard reset.

[c] Entries in the Activate column are as follows. For modes labeled yes, the device server shall activate the microcode (see 5.4.2) after completion of the final command in the WRITE BUFFER sequence (i.e., the series of WRITE BUFFER commands that downloads the microcode and activates it). For modes labeled optional, the device server may or may not activate the microcode image upon completion of the final command in the WRITE BUFFER sequence. For modes labeled no, the device server shall not activate the microcode upon completion of the final command in the WRITE BUFFER sequence.

[d] The application client delivers microcode in a vendor specific number of WRITE BUFFER commands (i.e., a WRITE BUFFER sequence). The device server should require that the microcode be delivered in a single command. The device server shall perform any required verification of the microcode prior to returning GOOD status for the final WRITE BUFFER command in a sequence (i.e., the WRITE BUFFER command delivering the last part of the microcode).

[e] The application client delivers microcode in one or more WRITE BUFFER commands, specifying a buffer ID and buffer offset in each command. If the device server does not receive the necessary WRITE BUFFER commands required to deliver the complete microcode before a logical unit reset occurs, an I_T nexus loss occurs, or a WRITE BUFFER command specifying a different download microcode mode is processed, then the device server shall discard the new microcode. If the device server determines that it is processing the final WRITE BUFFER command (i.e., the WRITE BUFFER command delivering the last part of the microcode), then the device server shall perform any required verification of the microcode prior to returning GOOD status for the command.

[f] Microcode downloaded with this mode is defined as deferred microcode.

[g] Support for mode 0Fh is mandatory if either mode 0Dh or mode 0Eh is supported.

Table 59 summarizes how the WRITE BUFFER download microcode modes process the BUFFER ID field, the BUFFER OFFSET field, and the PARAMETER LIST LENGTH field in the WRITE BUFFER CDB.

Table 59 — WRITE BUFFER download microcode field processing

Mode	BUFFER ID field, BUFFER OFFSET field, and PARAMETER LIST LENGTH field processing
Download microcode and activate (i.e., 04h)	vendor specific
Download microcode, save, and activate (i.e., 05h)	vendor specific
Download microcode with offsets and activate (i.e., 06h)	6.49.6
Download microcode with offsets, save, and activate (i.e., 07h)	6.49.6
Download microcode with offsets, select activation events, save, and defer activate (i.e., 0Dh)	6.49.6 and 6.49.9
Download microcode with offsets, save, and defer activate (i.e., 0Eh)	6.49.6
Activate deferred microcode (i.e., 0Fh)	ignored

If the device server is unable to process a WRITE BUFFER command with a download microcode mode because of a vendor specific condition (e.g., the device server requires the microcode be delivered in order, and the BUFFER OFFSET field is not equal to the contents of the previous WRITE BUFFER command's BUFFER OFFSET field plus the contents of the previous WRITE BUFFER command's PARAMETER LIST LENGTH field), then the device server shall terminate the command with CHECK CONDITION status, with the sense key set to ILLEGAL REQUEST, and the additional sense code set to COMMAND SEQUENCE ERROR.

If the device server detects a digital signature validation failure while processing a WRITE BUFFER command that downloads microcode, it shall terminate the command with CHECK CONDITION status, with the sense key set to ABORTED COMMAND, and the additional sense code set to DIGITAL SIGNATURE VALIDATION FAILURE.

The MULTI I_T NEXUS MICROCODE DOWNLOAD field (see table 58) in the Extended INQUIRY Data VPD page (see 7.8.7) indicates how the device server handles concurrent attempts to download microcode using the WRITE BUFFER command download microcode modes from multiple I_T nexuses.

Table 60 — MULTI I_T NEXUS MICROCODE DOWNLOAD field (part 1 of 2)

Code	Description
0h	The handling of concurrent WRITE BUFFER download microcode operations from multiple I_T nexus is vendor specific.
1h	For modes that download microcode (see table 58), the device server shall: a) if a WRITE BUFFER command with the BUFFER OFFSET field set to zero is received on any I_T nexus, then the command shall be processed as described elsewhere in this subclause. This shall establish the I_T nexus for the WRITE BUFFER sequence, and cause any microcode downloaded on another I_T nexus to be discarded; b) if a WRITE BUFFER command with the BUFFER OFFSET field set to a non-zero value is received on the established I_T nexus for the WRITE BUFFER sequence, then the command shall be processed as described elsewhere in this subclause; and c) if a WRITE BUFFER command with the BUFFER OFFSET field set to a non-zero value is received on an I_T nexus that is different from the established I_T nexus for the WRITE BUFFER sequence, then the device server shall terminate the command with CHECK CONDITION status, with the sense key set to ILLEGAL REQUEST, and the additional sense code set to COMMAND SEQUENCE ERROR. The WRITE BUFFER sequence for the established I_T nexus should not be affected. If a WRITE BUFFER command with mode 0Fh (i.e., activate deferred microcode) is received on an I_T nexus that is different from the established I_T nexus for the WRITE BUFFER sequence, then the device server shall terminate the WRITE BUFFER command with mode 0Fh with CHECK CONDITION status, with the sense key set to ILLEGAL REQUEST, and the additional sense code set to COMMAND SEQUENCE ERROR. Any deferred microcode shall not be invalidated.
2h	For modes that download microcode (see table 58), the device server shall allow concurrent sequences of WRITE BUFFER commands to be processed as described elsewhere in this subclause on more than one I_T nexus. If a WRITE BUFFER command with mode 0Fh (i.e., activate deferred microcode) is received on an I_T nexus that is different from the established I_T nexus for the WRITE BUFFER sequence, then the device server shall process the command as described elsewhere in this subclause.

Table 60 — MULTI I_T NEXUS MICROCODE DOWNLOAD field (part 2 of 2)

Code	Description
3h	For modes that download microcode (see table 58), the device server shall: a) if a WRITE BUFFER command with the BUFFER OFFSET field set to zero is received on any I_T nexus, then the command shall be processed as described elsewhere in this subclause. This shall establish the I_T nexus for the WRITE BUFFER sequence, and cause any microcode downloaded on another I_T nexus to be discarded; b) if a WRITE BUFFER command with the BUFFER OFFSET field set to a non-zero value is received on the established I_T nexus for the WRITE BUFFER sequence, then the command shall be processed as described elsewhere in this subclause; and c) if a WRITE BUFFER command with the BUFFER OFFSET field set to a non-zero value is received on an I_T nexus that is different from the established I_T nexus for the WRITE BUFFER sequence, then the device server shall terminate the command with CHECK CONDITION status, with the sense key set to ILLEGAL REQUEST, and the additional sense code set to COMMAND SEQUENCE ERROR. The WRITE BUFFER sequence for the established I_T nexus should not be affected. If a WRITE BUFFER command with mode 0Fh (i.e., activate deferred microcode) is received on an I_T nexus that is different from the established I_T nexus for the WRITE BUFFER sequence, then the device server shall process the command as described elsewhere in this subclause.
4h to Fh	Reserved

For all WRITE BUFFER command modes that download microcode (see table 58), the COMMAND SPECIFIC field (see 6.35.4.2) located in the command timeouts descriptor of the parameter data returned by the REPORT SUPPORTED OPERATION CODES command (see 6.35) indicates the maximum time that access to the SCSI device is limited or not possible through any SCSI ports associated with a logical unit that processes a WRITE BUFFER command that activates microcode.

5.4.2 Activating microcode

If the SCSI target device contains multiple logical units, the activation of microcode by one logical unit may change the microcode for other logical units in that SCSI target device.

Activating microcode for a logical unit may result in:

a) a hard reset; or
b) a logical unit reset for each logical unit affected by the microcode download (i.e., logical units that activated microcode or logical units that processed a logical unit reset due to another logical unit activated microcode).

If microcode is activated due to processing a WRITE BUFFER command with a mode that requires activation after processing (i.e., modes 04h (see 6.49.4), 06h (see 6.49.6), and 0Fh (see 6.49.11)), then:

a) if the microcode activation resulted in a hard reset or resulted in a logical unit reset, the device server shall establish a unit attention condition (see SAM-5) for the initiator port associated with every I_T nexus affected by the microcode activation except the I_T nexus on which the WRITE BUFFER command was received with the additional sense code set to MICROCODE HAS BEEN CHANGED; or
b) if the microcode activation did not result in a hard reset and did not result in a logical unit reset, the device server shall establish a unit attention condition for the initiator port associated with every I_T

nexus affected by the microcode activation with the additional sense code set to MICROCODE HAS BEEN CHANGED WITHOUT RESET.

If microcode is activated due to processing a WRITE BUFFER command with a mode that may cause activation after processing (i.e., for modes 05h (see 6.49.5) and 07h (see 6.49.7)), then the device server shall establish a unit attention condition (see SAM-5) based on the setting of the ACTIVATE MICROCODE field in the Extended INQUIRY VPD page (see 7.8.7).

If microcode is activated as a result of a power on or as a result of a hard reset, the device server may establish a unit attention condition (see SAM-5) for the initiator port associated with every I_T nexus with the additional sense code set to MICROCODE HAS BEEN CHANGED in addition to the unit attention condition for the power on or hard reset.

If deferred microcode (see table 58) is activated due to a command defined by its command standard as causing deferred microcode to be activated (e.g., the FORMAT UNIT command and the START STOP UNIT command (see SBC-3)), then the device server:

a) shall establish a unit attention condition (see SAM-5) for the initiator port associated with every I_T nexus affected by the microcode activation:
   A) if the microcode activation resulted in a hard reset or resulted in a logical unit reset, the additional sense code for the established unit attention shall be set to MICROCODE HAS BEEN CHANGED; or
   B) if the microcode activation did not result in a hard reset and did not result in a logical unit reset, the additional sense code for the established unit attention shall be set to MICROCODE HAS BEEN CHANGED WITHOUT RESET;

   and
b) may establish other unit attention condition(s) as defined for the command (e.g., CAPACITY DATA HAS CHANGED for the FORMAT UNIT command).

If new microcode is saved before deferred microcode is activated, then that deferred microcode is not activated.

5.5 Error history

5.5.1 Error history overview

Error history is data collected by a logical unit to aid in troubleshooting errors.

The READ BUFFER command (see 6.18.7) provides a method for retrieving error history from the logical unit (see 5.5.2).

The WRITE BUFFER command (see 6.49.12) provides a method for inserting application client error history into the error history (see 5.5.3) and for clearing the error history (see 5.5.4).

The format of the application client error history is defined by the manufacturer of the application client. The format of the error history, including how the application client error history, if any, is incorporated into the error history, is defined by the manufacturer of the logical unit.

5.5.2 Retrieving error history with the READ BUFFER command

Device servers may allow the error history to be retrieved using a sequence of READ BUFFER commands on one I_T nexus.

Error history is returned using error history snapshots. An error history snapshot is the contents of the error history at a specific point in time, created by the device server at vendor specific times or requested by the application client using the READ BUFFER command with certain buffer IDs.

The I_T nexus being used to retrieve an error history snapshot is called the error history I_T nexus. Only one I_T nexus at a time is allowed to be the error history I_T nexus, and only the error history I_T nexus is allowed to retrieve an error history snapshot.

To retrieve the complete error history, an application client uses one I_T nexus to:

1) create an error history snapshot if one does not already exist, establish the I_T nexus as the error history I_T nexus, and retrieve the error history directory by sending a READ BUFFER command (see 6.18.7.2) with:
   A) the MODE field set to 1Ch (i.e., error history);
   B) the BUFFER ID field set to one of the following:
      a) If the error history I_T nexus is expected to be valid:
         A) 00h (i.e., return error history directory); or
         B) 01h (i.e., return error history directory and create new snapshot);
      b) if the application client has knowledge obtained by means outside the scope of this standard that the error history I_T nexus is no longer valid:
         A) 02h (i.e., return error history directory and establish new error history I_T nexus); or
         B) 03h (i.e., return error history directory, establish new error history I_T nexus, and create new snapshot);
   C) the BUFFER OFFSET field set to 000000h; and
   D) the ALLOCATION LENGTH field set to at least 2 088 (i.e., large enough to transfer the complete error history directory);
2) retrieve the error history. The application client uses a Data-In Buffer size that is a multiple of the offset boundary indicated in the READ BUFFER descriptor (see 6.18.4). For each buffer ID indicated in the error history directory in the range of 10h to EFh, the application client sends one or more READ BUFFER commands (see 6.18.7.3) as follows:
   1) send the first READ BUFFER command with:
      a) the MODE field set to 1Ch (i.e., error history);
      b) the BUFFER ID field set to the buffer ID (i.e., an error history data buffer);
      c) the BUFFER OFFSET field set to 000000h; and
      d) the ALLOCATION LENGTH field set to the size of the Data-In Buffer;
      and
   2) until the number of bytes returned by the previous READ BUFFER command does not equal the specified allocation length and/or the total number of bytes returned from the buffer ID equals the maximum available length indicated in the error history directory, send zero or more additional READ BUFFER commands with:
      a) the MODE field set to 1Ch (i.e., error history);
      b) the BUFFER ID field set to the buffer ID (i.e., an error history data buffer);
      c) the BUFFER OFFSET field set to the previous buffer offset plus the previous allocation length; and
      d) the ALLOCATION LENGTH field set to the size of the Data-In Buffer;
   and
3) clear the error history I_T nexus and, depending on the buffer ID, release the error history snapshot by sending a READ BUFFER command with:
   A) the MODE field set to 1Ch (i.e., error history);
   B) the BUFFER ID field set to:
      a) FEh (i.e., clear error history I_T nexus) (see 6.18.7.4); or
      b) FFh (i.e., clear error history I_T nexus and release snapshot) (see 6.18.7.5);
   C) the BUFFER OFFSET field set to any value allowed by table 234 (see 6.18.7.1) (e.g., 000000h); and
   D) the ALLOCATION LENGTH field set to any value allowed for the chosen BUFFER ID field value (see 6.18.7.4 or 6.18.7.5) (e.g., 000000h).

While an error history snapshot exists, the device server:

a) shall not modify the error history snapshot to reflect any changes to the error history;
b) may or may not record events that are detected into the error history; and
c) if the device server supports the WRITE BUFFER command download application client error history mode (see 6.49.12), shall record the specified application client error history into the error history.

The device server shall clear the established error history I_T nexus and not release the error history snapshot:

a) upon processing of a READ BUFFER command on the error history I_T nexus with:
   A) the MODE field set to 1Ch (i.e., error history); and
   B) the BUFFER ID field set to FEh (i.e., clear error history I_T nexus) (see 6.18.7.4);
   or
b) if an I_T nexus loss occurs on the error history I_T nexus.

The device server shall clear the established error history I_T nexus and release the error history snapshot:

a) upon processing of a READ BUFFER command using the same I_T nexus that was used to establish the snapshot with:
   A) the MODE field set to 1Ch (i.e., error history); and
   B) the BUFFER ID field set to FFh (i.e., clear error history I_T nexus and release snapshot) (see 6.18.7.5);
b) if a power on occurs;
c) if a hard reset occurs; or
d) if a logical unit reset occurs.

The device server shall not replace or release the error history snapshot while the error history I_T nexus is established.

The device server shall implement a vendor specific timer for error history snapshot retrieval. If the vendor specific timer expires, then:

a) the device server shall:
   A) clear the error history I_T nexus; and
   B) establish a unit attention condition for the error history I_T nexus with the additional sense code set to ERROR HISTORY I_T NEXUS CLEARED;
   or
b) the device server shall:
   A) clear the error history I_T nexus;
   B) release the error history snapshot; and
   C) establish a unit attention condition for the error history I_T nexus with the additional sense code set to ERROR HISTORY SNAPSHOT RELEASED.

After an error history snapshot is released, the device server shall resume recording error history for events that are detected.

Error history may also be retrieved by vendor specific methods or other READ BUFFER command sequences that are outside the scope of this standard.

5.5.3 Adding application client error history with the WRITE BUFFER command

An application client adds application client detected error history to the error history collected by a logical unit using a WRITE BUFFER command with the MODE field set to 1Ch (see 6.49.12). The application client error history:

a) may be recovered:
   A) as part of the error history (see 5.5.2); or
   B) by means outside the scope of this standard;
   and
b) is not used for any logical unit related error recovery.

Error history that contains a mix of application client error history and logical unit error history may be used to correlate an application client-detected error with errors detected internally by the logical unit.

Application clients should minimize the amount of error history they store to prevent error history overflows (see 6.49.12).

5.5.4 Clearing error history with the WRITE BUFFER command

An application client clears the portions of the error history that the device server allows to be cleared by sending a WRITE BUFFER command (see 6.49.12) with:

a) the MODE field set to 1Ch (i.e., download error history);
b) the PARAMETER LIST LENGTH field set to 00001Ah;
c) in the parameter list, the CLR bit set to one; and
d) all other fields in the parameter list set as 6.49.12 describes.

Clearing error history shall not:

a) clear the error history I_T nexus, if any; or
b) release the error history snapshot, if any, if it was created with the READ BUFFER command (see 5.5.2).

5.6 Identifying information

The REPORT IDENTIFYING INFORMATION command (see 6.32) and SET IDENTIFYING INFORMATION command (see 6.43) allow an application client to maintain one or more sets of identifying information associated with the peripheral device.

Identifying information shall persist through power cycles (i.e., be stored in non-volatile storage), hard resets, logical unit resets, I_T nexus losses, media format operations, and media replacement.

Table 61 defines the identifying information types.

Table 61 — Identifying information types

Identifying information type	Length	Support [a]
Peripheral device identifying information: a value describing the peripheral device (e.g., an operating system volume label)	0 to 64 bytes	Mandatory
	65 to 512 bytes	Optional
Peripheral device text identifying information: a null-terminated (see 4.3.2) UTF-8 format string providing an informational description of the peripheral device (e.g., a descriptive string entered by a system administrator)	0 to 256 bytes	Optional

[a] These support requirements shall apply only if the REPORT IDENTIFYING INFORMATION command and/or SET IDENTIFYING INFORMATION command are implemented.

Identifying information is changed by:

a) the SET IDENTIFYING INFORMATION command; or
b) a mechanism outside the scope of this standard (e.g., a system administrator may be able to change identifying information through a management interface).

If a mechanism outside the scope of this standard changes the identifying information, then the device server shall establish a unit attention condition for the initiator port associated with every I_T nexus with the additional sense code set to DEVICE IDENTIFIER CHANGED.

5.7 Medium auxiliary memory

Some types of media, especially removable media, include a non-volatile memory referred to as MAM. Medium auxiliary memory is used to store data describing the media and its contents. This standard supports medium auxiliary memory with the READ ATTRIBUTE command (see 6.17) and the WRITE ATTRIBUTE command (see 6.48). These commands are used to retrieve and store information in the medium auxiliary memory in the form of MAM attributes.

A MAM attribute is represented in a format described in 7.4.

There are three types of MAM attributes (see table 62).

Table 62 — Types of MAM attributes

Attribute Type	Attribute Source	Example	Readable with READ ATTRIBUTE	Writable with WRITE ATTRIBUTE
Medium	Permanently stored in the medium auxiliary memory during manufacture.	Media Serial Number	Yes	No
Device	Maintained by the device server.	Load Count	Yes	No
Host	Maintained by the application client.	Backup Date	Yes	Yes

Depending on the attribute type, MAM attributes have the states shown in table 63.

Table 63 — MAM attribute states

Attribute Type	Attribute State	Description
Medium or Device	Read Only	An application client may read the contents of the MAM attribute with the READ ATTRIBUTE command, but an attempt to clear or change the MAM attribute using the WRITE ATTRIBUTE command shall result in the device server terminating the command with CHECK CONDITION status. If the READ ONLY bit (see 7.4.1) is set to one, the attribute is in the read only state.
	Unsupported	The device server does not support the MAM attribute and shall not return this attribute in response to a READ ATTRIBUTE command.
	Unavailable	The MAM attribute exists but is not available at this time. The device server shall not return this attribute in response to a READ ATTRIBUTE command.
Host	Nonexistent	A host attribute does not exist in the medium auxiliary memory until a WRITE ATTRIBUTE command creates it.
	Read/Write	The MAM attribute has been created using the WRITE ATTRIBUTE command. After the MAM attribute has been created, the contents may be altered using subsequent WRITE ATTRIBUTE commands. A read/write MAM attribute may be placed in the nonexistent state using a WRITE ATTRIBUTE command with the attribute length set to zero. If the READ ONLY bit (see 7.4.1) is set to zero, the MAM attribute is in the read/write state.
	Unsupported	The device server does not support the MAM attribute and shall not return this attribute in response to a READ ATTRIBUTE command.

5.8 Parameter rounding

For certain commands, one or more specified parameters may be constrained to a range of values. Device servers may choose to implement only selected values from this range. If the device server receives a value that it does not support, the device server shall:

a) terminate the command (e.g., by returning CHECK CONDITION status with ILLEGAL REQUEST sense key); or
b) round the value received to a supported value.

If 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 the sense key set to RECOVERED ERROR, and the additional sense code set to ROUNDED

PARAMETER. The application client should send 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 (i.e., up or down) is described in the definition of the parameter.

5.9 Parsing variable length parameter lists and parameter data

Parameter lists and parameter data (e.g., diagnostic pages, mode pages, log pages, and VPD pages) often include length fields indicating the size of the parameter list or parameter data (e.g., the MODE DATA LENGTH field in the mode parameter header (see 7.5.5)). Parameter lists and parameter data often include descriptor lists and descriptor length fields containing the length of the descriptors in the descriptor lists (e.g., the DESIGNATOR LENGTH field in the designation descriptor used in the Device Identification VPD page (see 7.8.6.1)).

An application client or device server shall not assume that any length field contains the value defined in a SCSI standard.

If a device server receives a parameter list containing a length field (e.g., a PAGE LENGTH field) and containing more bytes than are defined in the standard to which it was designed (e.g., the device server complies with a version of a SCSI standard defining that a parameter list has 24 bytes, but receives a parameter list containing 36 bytes), then the device server shall terminate the command with CHECK CONDITION status, with the sense key set to ILLEGAL REQUEST, and the additional sense code set to INVALID FIELD IN PARAMETER LIST.

For parameter lists containing a descriptor length field and a descriptor list, if a device server receives more bytes in a descriptor than are defined in the standard to which the device server was designed (e.g., the device server complies with a version of a SCSI standard defining that a descriptor is 12 bytes, but receives a parameter list containing a 16 byte form of that descriptor), then the device server shall terminate the command with CHECK CONDITION status, with the sense key set to ILLEGAL REQUEST, and the additional sense code set to INVALID FIELD IN PARAMETER LIST.

An application client should ignore any bytes of parameter data beyond those defined in the standard to which the application client was designed (e.g., if the application client complies with a version of a SCSI standard defining 24 bytes of parameter data, but receives 36 bytes of parameter data, then the application client should ignore the last 12 bytes or the parameter data).

For additional response bytes containing a descriptor length field and a descriptor list, an application client should ignore any bytes in each descriptor beyond those defined in the standard to which the application client was designed (e.g., if the application client complies with a version of a SCSI standard defining that a descriptor has 24 bytes, but receives parameter data containing a descriptor list with a 36 byte form of that descriptor, then the application client should ignore the last 12 bytes of the descriptor).

5.10 Pollable condition information

5.10.1 Information that does not represent an exception condition

A device server may have information about a condition that does not represent an exception condition. This information is not reported with a CHECK CONDITION status. Instead, this information is reported:

a) as sense data format parameter data in response to a REQUEST SENSE command as described in 5.10.2; or
b) as log parameter data as described in 5.10.3.

5.10.2 REQUEST SENSE pollable sense data

5.10.2.1 Making information available for the REQUEST SENSE command

SCSI target devices are required to make specified information used in the parameter data returned by the REQUEST SENSE command (see 6.39) available whenever that information is applicable. Which sense data is returned in the REQUEST SENSE parameter data is determined at the time the REQUEST SENSE command is processed.

Application clients have no way to control which pollable sense data is returned by a REQUEST SENSE command. Mechanisms that are specialized to a particular function (e.g., log pages, mode pages) should be used to obtain information about that function.

5.10.2.2 Selecting pollable sense data to return

Conditions that are not related to the availability of pollable sense data (e.g., a pending unit attention condition) may cause the device server to ignore all available pollable sense data.

If pollable sense data is available to be returned by a REQUEST SENSE command (see 6.39), the choice of which sense key and additional sense code to return shall be made as follows:

1) sense data with the sense key set to NOT READY and the additional sense code set to:
   1) LOGICAL UNIT NOT READY, INITIALIZING COMMAND REQUIRED (see 5.11.7); and
   2) any other additional sense code that is associated with pollable sense data when combined with a sense key set to NOT READY;

   and
2) sense data with the sense key set to NO SENSE and the additional sense code set:
   1) as defined for the informational exceptions sense data described in the Informational Exceptions Control mode page (see applicable command standard);
   2) to LOGICAL UNIT TRANSITIONING TO ANOTHER POWER CONDITION (see 5.11.8 and SBC-3);
   3) as defined for any power condition related sense data described in 5.11.7; and
   4) any other additional sense code that is associated with pollable sense data when combined with a sense key set to NO SENSE.

5.10.2.3 Returning one or more progress indications

If the sense key is set to NOT READY or NO SENSE in the header of the sense data being returned by a REQUEST SENSE command (see 6.39) and progress indication information is associated with the pollable sense data, if any, in the sense data (see 5.10.2.1), then:

a) if the DESC bit is set to zero in the REQUEST SENSE command, the device server shall set the SKSV bit in the fixed-format sense data to one only if progress indication information is available for the

additional sense code associated with the ADDITIONAL SENSE CODE field and ADDITIONAL SENSE CODE QUALIFIER field; or

b) if the DESC bit is set to one in the REQUEST SENSE command, the device server places progress indications in the sense data as follows:
   A) if progress indication information is available for the additional sense code associated with the ADDITIONAL SENSE CODE field and ADDITIONAL SENSE CODE QUALIFIER field in the sense data header, then the device server shall include one sense key specific sense data descriptor (see 4.5.2.4) that contains the available progress indication information; and
   B) if progress indication information is available for one or more additional sense codes that are not associated with the ADDITIONAL SENSE CODE field and ADDITIONAL SENSE CODE QUALIFIER field in the sense data header, then for each instance of available progress indication information, the device server should include one another progress indication sense data descriptor (see 4.5.2.6) that contains the available progress indication information.

5.10.3 Log parameter pollable device condition information

The following background testing functions report their condition information using log page parameters:

a) self-test operations report whether a test is in progress, but not how far it has progressed using the Self-Test Results log page (see 7.3.17); and
b) direct-access block devices report progress for background pre-scan operations and background medium scan operations using the Background Scan Results log page (see SBC-3).

5.11 Power management

5.11.1 Power management overview

The Power Condition mode page (see 7.5.13) and Power Consumption mode page (see 7.5.14) allow an application client to manage the power utilization of a logical unit in a manner that may reduce power consumption of the SCSI target device.

Power consumption management is described in 5.11.2, including its interactions with power condition management. Power condition management is described in 5.11.3.

A change in the power consumption setting or power condition of any logical unit in a SCSI target device may result in a change in the SCSI target device's power utilization. If a SCSI target device contains multiple logical units, then the SCSI target device's power utilization may not change until a group of the logical units have changed their power consumption in the active power condition (see 5.11.2) or changed to a lower power condition (see 5.11.3). Any grouping or groupings of logical units for power management is outside the scope of this standard.

5.11.2 Power consumption management

Power consumption management allows control of the maximum power consumption (e.g., see USB-3) of a logical unit that is in the active power condition (see 5.11.4).

If power consumption management is supported, the device server shall support:

a) the Power Consumption mode page (see 7.5.14); and
b) the Power Consumption VPD page (see 7.8.11).

The Power Consumption VPD page contains one or more power consumption descriptors that indicate the maximum power consumption levels supported by the device server using:

a) a power consumption identifier; and
b) information about the maximum power consumption associated with that power consumption identifier.

An application client may specify use of one of the maximum power consumption levels indicated by the Power Consumption VPD page by setting the POWER CONSUMPTION IDENTIFIER field in the Power Consumption mode page to the contents of that POWER CONSUMPTION IDENTIFIER field in the Power Consumption VPD page. The SCSI target device shall limit the maximum power consumption while the logical unit is in the active power condition to the value indicated in the power consumption descriptor in the Power Consumption VPD page that is associated with the POWER CONSUMPTION IDENTIFIER field in the Power Consumption mode page.

Power consumption management shall:

a) be used to limit the maximum power consumption while in the active power condition;
b) not affect the power consumed during a change between power conditions; and
c) not affect the power consumed while in a power condition other than active.

5.11.3 Power conditions management

Fields in the Power Condition mode page (see 7.5.13) manage power conditions by enabling and initializing one or more idle condition timers and/or standby condition timers.

Command standards may define the following additional power management features:

a) power conditions (e.g., the stopped power condition in SBC-3);
b) changes to the power condition state machine (e.g., the SSU_PC8:Stopped state in SBC-3);
c) commands that manage power conditions (e.g., a START STOP UNIT command in SBC-3 or MMC-6);
d) changes to commands defined in this standard; or
e) mode pages, log pages, or VPD pages that are associated with managing power conditions.

Transport protocol standards (e.g., SPL-3) may define additional requirements on the states in the power condition state machine defined in this standard or a command standard.

There shall be no notification to the application client that a logical unit has changed from one power condition to another. The response to a REQUEST SENSE command (see 6.39) may indicate whether a logical unit is in a low power condition and which low power condition.

The current power condition of a logical unit may be decreased by:

a) the expiration of a power condition timer;
b) the completion of background functions; or
c) power condition activities described in a command standard.

The current power condition of a logical unit is increased by:

a) the processing of a command that the device server is unable to continue processing while in the current power condition;
b) the processing of a background function that the device server is unable to process while in the current power condition; or
c) power condition activities described in a command standard.

If a device server processes a command that the device server is capable of completing while the logical unit is in a low power condition, then the device server shall not stop any enabled power condition timers, regardless of which power condition the logical unit was in when the device server began processing the command.

If a device server processes a command that the device server is not capable of completing while the logical unit is in a low power condition, then the device server shall stop any running power condition timers. On completion of the command, the device server shall reinitialize all enabled power condition timers based on their values in the Power Condition mode page (see 7.5.13) and start the timers, regardless of which power condition the logical unit was in when the device server began processing the command.

The device server shall process any task management function (see SAM-5), except LOGICAL UNIT RESET, regardless of current power condition, without changing to a different power condition. The power condition timers shall not be affected by task management functions, except LOGICAL UNIT RESET.

The device server may change power conditions or power condition timers while processing a LOGICAL UNIT RESET.

No power condition defined in this standard shall affect the supply of any power required for proper operation of a service delivery subsystem.

Logical units that contain cache memory shall write all cached data to the medium for the logical unit (e.g., as a logical unit would do in response to a SYNCHRONIZE CACHE command as described in SBC-3) prior to entering into any power condition that prevents accessing the media (e.g., before a SCSI target device stops its spindle motor during a change to the standby power condition).

5.11.4 Active power condition

While in the active power condition:

a) the device server is capable of completing the processing of its supported commands, including those that require media access, without the logical unit changing power condition;
b) the device server completes processing of a command in the shortest time when compared to the time required for completion of that command if command processing began while the logical unit was in any of the idle power conditions or standby power conditions; and
c) the SCSI target device may consume more power than while the logical unit is in any of the idle power conditions or standby power conditions (e.g., a disk drive's spindle motor may be active).

A logical unit that is in the active power condition may be affected by power consumption management (see 5.11.2).

5.11.5 Idle power conditions

A device server may support more than one idle power condition (i.e., idle_a, idle_b, and idle_c) to provide progressively lower power consumption (i.e., the following power consumption relationship: idle_a ≥ idle_b ≥ idle_c).

NOTE 10 - The idle_a power condition was referenced as the idle power condition in SPC-3.

While in one of the idle power conditions:

a) the device server is capable of completing the processing of its supported commands, except those that require the logical unit to be in the active power condition to be capable of completing the command with GOOD status (e.g., commands that require media access to complete processing);

b) the device server may take longer to complete processing a command than while the logical unit is in the active power condition (e.g., the device may have to activate some circuitry before completing processing of a command);
c) the power consumed by the SCSI target device while in an idle power condition should be less than the power consumed while the logical unit is in the active power condition and may be greater than the power consumed while the logical unit is in a standby power condition; and
d) the peak power consumption during a change from an idle power condition to the active power condition shall be no more than the typical peak power consumption in the active power condition.

5.11.6 Standby power conditions

A device server may support more than one standby power condition (i.e., standby_y and standby_z) to provide progressively lower power consumption (i.e., the following power consumption relationship: standby_y ≥ standby_z).

NOTE 11 - The standby_z power condition was referenced as the standby power condition in SPC-3.

While in one of the standby power conditions:

a) the device server is not capable of completing the processing of commands that require media access without the logical unit changing to the active power condition. SCSI transport protocol standards may impose additional requirements on command processing while changing to a higher power condition (e.g., the response may be CHECK CONDITION status with sense key set to NOT READY and additional sense bytes set to LOGICAL UNIT NOT READY, NOTIFY (ENABLE SPINUP) REQUIRED instead of GOOD status (see SPL-3));
b) the device server may take longer to complete processing a command than while the logical unit is in the active power condition or one of the idle power conditions (e.g., a disk drive's spindle motor may need to be started);
c) the power consumed by the SCSI target device while in one of the standby power conditions should be less than the power consumed while the logical unit is in the active power condition or any of the idle power conditions; and
d) the peak power consumption during a change from a standby power condition to the active power condition or an idle power condition is not limited by this standard.

5.11.7 Power condition pollable sense data

If the logical unit is in any power condition other than active, the following data shall be available for use by the REQUEST SENSE command while returning pollable sense data (see 5.10.2) and:

a) if the logical unit is in an idle power condition (see 5.11.5), then the sense key shall be set to NO SENSE and the additional sense code set to one of the following:
   A) LOW POWER CONDITION ON if the reason for entry into the idle power condition is unknown;
   B) IDLE CONDITION ACTIVATED BY TIMER if the logical unit entered the idle_a power condition due to the idle_a condition timer (see 5.11.8.4);
   C) IDLE CONDITION ACTIVATED BY COMMAND if the logical unit entered the idle_a power condition due to processing of a command;
   D) IDLE_B CONDITION ACTIVATED BY TIMER if the logical unit entered the idle_b power condition due to the idle_b condition timer (see 5.11.8.4);
   E) IDLE_B CONDITION ACTIVATED BY COMMAND if the logical unit entered the idle_b power condition due to processing of a command;
   F) IDLE_C CONDITION ACTIVATED BY TIMER if the logical unit entered the idle_c power condition due to the idle_c condition timer (see 5.11.8.4); or
   G) IDLE_C CONDITION ACTIVATED BY COMMAND if the logical unit entered the idle_c power condition due to processing of a command;

b) if the logical unit is in a standby power condition (see 5.11.6), then the sense key shall be set to NO SENSE and the additional sense code set to one of the following:
   A) LOW POWER CONDITION ON if the reason for entry into the standby power condition is unknown;
   B) STANDBY_Y CONDITION ACTIVATED BY TIMER if the logical unit entered the standby_y power condition due to the standby_y condition timer (see 5.11.8.5);
   C) STANDBY_Y CONDITION ACTIVATED BY COMMAND if the logical unit entered the standby_y power condition due to processing of a command;
   D) STANDBY CONDITION ACTIVATED BY TIMER if the logical unit entered the standby_z power condition due to the standby_z condition timer (see 5.11.8.5); or
   E) STANDBY CONDITION ACTIVATED BY COMMAND if the logical unit entered the standby_z power condition due to processing of a command;

   or
c) if the logical unit is in the stopped power condition (see SBC-3), then the sense key shall be set to NOT READY and the additional sense code shall be set to LOGICAL UNIT NOT READY, INITIALIZING COMMAND REQUIRED.

NOTE 12 - Device servers that conform to SBC-2 may not provide the pollable sense data described in c).

5.11.8 Power condition state machine

5.11.8.1 Power condition state machine overview

The power condition state machine describes the logical unit power states and transitions resulting from command processing and Power Condition mode page values (see 7.5.13).

The power condition state machine states are shown in table 64.

Table 64 — Power condition state machine states

State	Reference
PC0:Powered_On [a]	5.11.8.2
PC1:Active	5.11.8.3
PC2:Idle	5.11.8.4
PC3:Standby	5.11.8.5
PC4:Active_Wait	5.11.8.6
PC5:Wait_Idle	5.11.8.7
PC6:Wait_Standby	5.11.8.8
[a] PC0:Powered_On is the initial state.	

While in the following power condition state machine states the logical unit may be increasing power usage to enter a higher power condition:

a) PC4:Active_Wait.

While in the following power condition state machine states the logical unit may be decreasing power usage to enter a lower power condition:

a) PC5:Wait_Idle; and
b) PC6:Wait_Standby.

The power condition state machine maintains the timers listed in table 65.

Table 65 — Power condition state machine timers

Timer	Initial value [a]	Enable bit [a, b]
idle_a condition	IDLE_A CONDITION TIMER field	IDLE_A bit
idle_b condition	IDLE_B CONDITION TIMER field	IDLE_B bit
idle_c condition	IDLE_C CONDITION TIMER field	IDLE_C bit
standby_y condition	STANDBY_Y CONDITION TIMER field	STANDBY_Y bit
standby_z condition	STANDBY_Z CONDITION TIMER field	STANDBY_Z bit

[a] These fields and bits are in the Power Condition mode page (see 7.5.13).
[b] In a direct-access block device, the enabled state of these bits may be overridden by a START STOP UNIT command (see SBC-3).

If more than one of the timers listed in table 65 expire at the same time, then only one timer is processed. The processing priority order shall be as follows:

1) standby_z condition timer;
2) standby_y condition timer;
3) idle_c condition timer;
4) idle_b condition timer; and
5) idle_a condition timer.

The power condition state machine shall start in the PC0:Powered_On state after power on.

Figure 8 describes the power condition state machine.

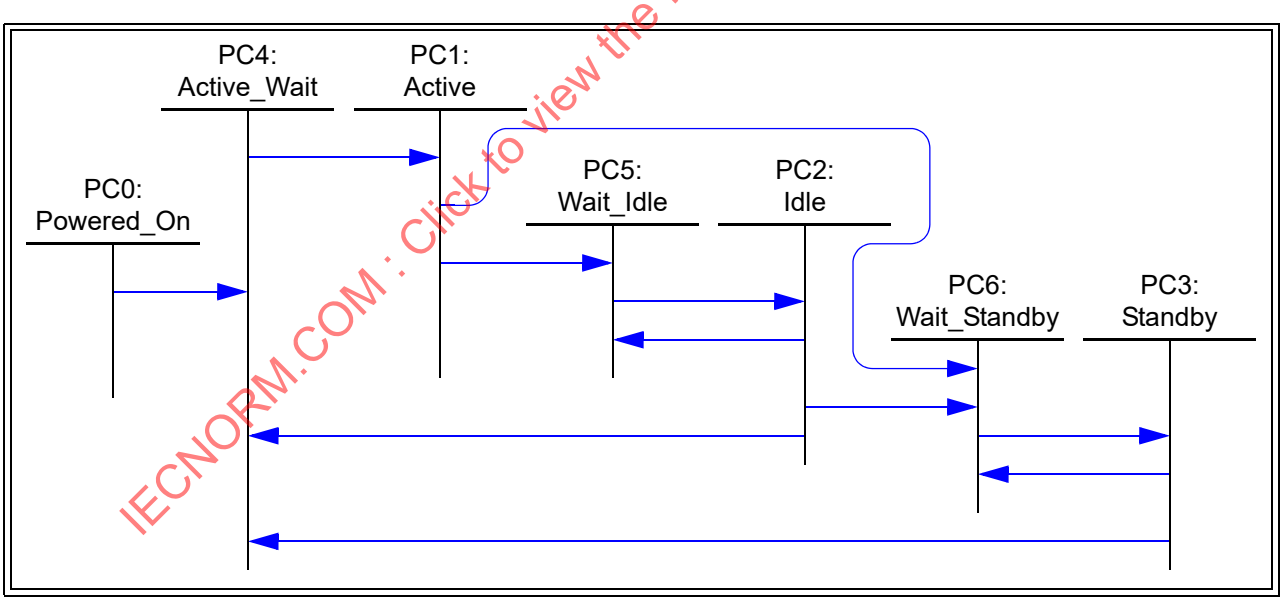


Figure 8 — Power condition state machine

5.11.8.2 PC0:Powered_On state

5.11.8.2.1 PC0:Powered_On state description

The logical unit shall enter this state upon power on.

5.11.8.2.2 Transition PC0:Powered_On to PC4:Active_Wait

This transition shall occur after:

a) the logical unit is ready to begin power on initialization.

The transition shall include a Transitioning From Powered On argument.

5.11.8.3 PC1:Active state

5.11.8.3.1 PC1:Active state description

While in this state, if power on initialization is not complete, then the logical unit shall complete its power on initialization.

While in this state, if power on initialization is complete, then:

a) the logical unit is in the active power condition (see 5.11.4);
b) each enabled idle condition timer is running; and
c) each enabled standby condition timer is running.

5.11.8.3.2 Transition PC1:Active to PC5:Wait_Idle

This transition shall occur if:

a) an idle condition timer is enabled; and
b) that idle condition timer has expired.

The transition shall include a:

a) Transitioning To Idle_a argument, if the highest priority timer (see 5.11.8.1) that expired is the idle_a condition timer;
b) Transitioning To Idle_b argument, if the highest priority timer that expired is the idle_b condition timer; or
c) Transitioning To Idle_c argument, if the highest priority timer that expired is the idle_c condition timer.

5.11.8.3.3 Transition PC1:Active to PC6:Wait_Standby

This transition shall occur if:

a) a standby condition timer is enabled; and
b) that standby condition timer has expired.

The transition shall include a:

a) Transitioning To Standby_z argument, if the highest priority timer (see 5.11.8.1) that expired is the standby_z condition timer; or
b) Transitioning To Standby_y argument, if the highest priority timer that expired is the standby_y condition timer.

5.11.8.4 PC2:Idle state

5.11.8.4.1 PC2:Idle state description

While in this state:

a) the logical unit is in an idle power condition (see 5.11.5);

b) the device server shall provide power condition pollable sense data (see 5.11.7);
c) each enabled idle condition timer that has not expired is running; and
d) each enabled standby condition timer that has not expired is running.

If a lower priority (see 5.11.8.1) idle condition timer is enabled and expires, then that timer is ignored.

5.11.8.4.2 Transition PC2:Idle to PC4:Active_Wait

This transition shall occur if:

a) the device server processes a command that requires the logical unit to be in the PC1:Active state to continue processing that command.

The transition shall include a:

a) Transitioning From Idle argument; and
b) Transitioning From Idle_c argument, if the current power condition is the idle_c power condition.

5.11.8.4.3 Transition PC2:Idle to PC5:Wait_Idle

This transition shall occur if:

a) an idle condition timer is enabled;
b) that idle condition timer has expired; and
c) the priority (see 5.11.8.1) of that idle condition timer is greater than the priority of the idle condition timer associated with the current idle power condition.

The transition shall include a:

a) Transitioning To Idle_b argument, if the highest priority timer that expired is the idle_b condition timer; or
b) Transitioning To Idle_c argument, if the highest priority timer that expired is the idle_c condition timer.

5.11.8.4.4 Transition PC2:Idle to PC6:Wait_Standby

This transition shall occur if:

a) a standby condition timer is enabled; and
b) that standby condition timer has expired.

The transition shall include a:

a) Transitioning To Standby_z argument, if the highest priority timer (see 5.11.8.1) that expired is the standby_z condition timer; or
b) Transitioning To Standby_y argument, if the highest priority timer that expired is the standby_y condition timer.

5.11.8.5 PC3:Standby state

5.11.8.5.1 PC3:Standby state description

While in this state:

a) the logical unit is in a standby power condition (see 5.11.6);
b) the device server shall provide power condition pollable sense data (see 5.11.7);
c) each enabled idle condition timer that has not expired is running; and
d) each enabled standby condition timer that has not expired is running.

If an idle condition timer or a lower priority (see 5.11.8.1) standby condition timer is enabled and expires, then that timer is ignored.

5.11.8.5.2 Transition PC3:Standby to PC4:Active_Wait

This transition shall occur if:

a) the device server processes a command that requires the logical unit to be in the PC1:Active state to continue processing that command.

The transition shall include a Transitioning From Standby argument.

5.11.8.5.3 Transition PC3:Standby to PC6:Wait_Standby

This transition shall occur if:

a) the standby_z condition timer is enabled;
b) the standby_z condition timer expired; and
c) the current power condition is the standby_y power condition.

The transition shall include a Transitioning To Standby_z argument.

5.11.8.6 PC4:Active_Wait state

5.11.8.6.1 PC4:Active_Wait state description

While in this state:

a) each idle condition timer that is enabled and not expired is running;
b) each standby condition timer that is enabled and not expired is running;
c) the device server shall provide power condition pollable sense data (see 5.11.7) with the sense key set to NO SENSE and the additional sense code set to LOGICAL UNIT TRANSITIONING TO ANOTHER POWER CONDITION; and
d) the logical unit is performing the operations required for it to be in the PC1:Active state (e.g., a disk drive spins up its media).

If this state was entered with a Transitioning From Idle argument, then:

a) the device server is capable of processing and completing the same commands that the device server is able to process and complete while in the PC2:Idle state;
b) the peak power consumed in this state shall be no more than the typical peak power consumed in the PC1: Active state; and
c) the device server shall terminate any command that requires the logical unit be in the PC1:Active state to continue processing, with CHECK CONDITION status, with the sense key set to NOT READY and the additional sense code set to LOGICAL UNIT IS IN PROCESS OF BECOMING READY if all of the following are true:
   A) this state was entered with a Transitioning From Idle_c argument; and
   B) the CCF IDLE field in the Power Condition mode page (see 7.5.13) is set to 10b (i.e., enabled).

If this state was entered with a Transitioning From Standby argument, then:

a) the device server is capable of processing and completing the same commands that the device server is able to process and complete while in the PC3:Standby state;
b) the peak power consumption in this state is not limited by this standard; and
c) if the CCF STANDBY field in the Power Condition mode page (see 7.5.13) is set to 10b (i.e., enabled), then the device server shall terminate any command that requires the logical unit be in the PC1:Active state or PC2:Idle state to continue processing, with CHECK CONDITION status, with the sense key

set to NOT READY and the additional sense code set to LOGICAL UNIT IS IN PROCESS OF BECOMING READY.

If this state was entered with a Transitioning From Powered On argument, then:

a) the device server is capable of processing and completing the same commands (except a TEST UNIT READY command) that the device server is able to process and complete while in the PC3:Standby state; and
b) the device server shall terminate with CHECK CONDITION status, with the sense key set to NOT READY and the additional sense code set to LOGICAL UNIT IS IN PROCESS OF BECOMING READY any of the following commands:
   A) any command that requires the logical unit be in the PC1:Active state or PC2:Idle state to continue processing; and
   B) all TEST UNIT READY commands (see 6.47).

If an idle condition timer or a standby condition timer is enabled and expires, then that timer is ignored in this state.

5.11.8.6.2 Transition PC4:Active_Wait to PC1:Active

This transition shall occur if:

a) the logical unit meets the requirements for being in the PC1:Active state.

5.11.8.7 PC5:Wait_Idle state

5.11.8.7.1 PC5:Wait_Idle state description

While in this state:

a) each idle condition timer that is enabled and not expired is running;
b) each standby condition timer that is enabled and not expired is running;
c) the device server shall provide power condition pollable sense data (see 5.11.7) with the sense key set to NO SENSE and the additional sense code set to LOGICAL UNIT TRANSITIONING TO ANOTHER POWER CONDITION;
d) the logical unit is performing the operations required for it to be in the PC2:Idle state (e.g., reducing power usage); and
e) the device server is capable of processing and completing the same commands, except a START STOP UNIT command with the IMMED bit set to zero (see SBC-3), that the device server is able to process and complete in the PC2:Idle state.

If an idle condition timer or a standby condition timer is enabled and expires, then that timer is ignored in this state.

5.11.8.7.2 Transition PC5:Wait_Idle to PC2:Idle

This transition shall occur if:

a) the logical unit meets the requirements for being in:
   A) the idle_a power condition, if this state was entered with a Transitioning To Idle_a argument;
   B) the idle_b power condition, if this state was entered with a Transitioning To Idle_b argument; or
   C) the idle_c power condition, if this state was entered with a Transitioning To Idle_c argument.

5.11.8.8 PC6:Wait_Standby state

5.11.8.8.1 PC6:Wait_Standby state description

While in this state:

a) each idle condition timer that is enabled and not expired is running;
b) each standby condition timer that is enabled and not expired is running;
c) the device server shall provide power condition pollable sense data (see 5.11.7) with the sense key set to NO SENSE and the additional sense code set to LOGICAL UNIT TRANSITIONING TO ANOTHER POWER CONDITION;
d) the logical unit is performing the operations required for it to be in the PC3:Standby state (e.g., reducing power usage); and
e) the device server is capable of processing and completing the same commands, except a START STOP UNIT command with the IMMED bit set to zero (see SBC-3), that the device server is able to process and complete in the PC3:Standby state.

If an idle condition timer or a standby condition timer is enabled and expires, then that timer is ignored in the state.

5.11.8.8.2 Transition PC6:Wait_Standby to PC3:Standby

This transition shall occur if:

a) the logical unit meets the requirements for being in:
   A) the standby_y power condition, if this state was entered with a Transitioning To Standby_y argument; or
   B) the standby_z power condition, if this state was entered with a Transitioning To Standby_z argument.

5.12 Reservations

5.12.1 Persistent Reservations overview

Reservations may be used to allow a device server to process commands from a selected set of I_T nexuses (i.e., combinations of initiator ports accessing target ports) and reject commands from I_T nexuses outside the selected set. The device server uniquely identifies I_T nexuses using protocol specific mechanisms.

Application clients may add or remove I_T nexuses 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 persistent reservations mechanism allows multiple application clients communicating through multiple I_T nexuses to preserve reservation operations across SCSI initiator device failures, which usually involve logical unit resets and involve I_T nexus losses. Persistent reservations persist across recovery actions. Persistent reservations are not reset by hard reset, logical unit reset, or I_T nexus loss.

The persistent reservation held by a failing I_T nexus may be preempted by another I_T nexus as part of its recovery process. Persistent reservations shall be retained by the device server until released, preempted, or cleared by mechanisms defined in this standard. Persistent reservations may be retained if power to the SCSI target device is removed.

The PERSISTENT RESERVE OUT command and PERSISTENT RESERVE IN command provide the basic mechanism for dynamic contention resolution in systems with multiple initiator ports accessing a logical unit.

Before a persistent reservation may be established, the application client shall register a reservation key for each I_T nexus with the device server. Reservation keys allow:

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

The reservation key provides a method for the application client to associate a protocol-independent identifier with a registered I_T nexus. The reservation key is used in the PERSISTENT RESERVE IN command to identify which I_T nexuses are registered and which I_T nexus, if any, holds the persistent reservation. The reservation key is used in the PERSISTENT RESERVE OUT command to register an I_T nexus, to verify the I_T nexus being used for the PERSISTENT RESERVE OUT command is registered, and to specify which registrations or persistent reservation to preempt.

Reservation key values may be used by application clients to identify registered I_T nexuses, using application specific methods that are outside the scope of this standard. This standard provides the ability to register no more than one reservation key per I_T nexus. Multiple initiator ports may use the same reservation key value for a logical unit accessed through the same target ports. An initiator port may use the same reservation key value for a logical unit accessed through different target ports. The logical unit shall maintain a separate reservation key for each I_T nexus, regardless of the reservation key's value.

An application client may register an I_T nexus with multiple logical units in a SCSI target device using any combination of unique or duplicate reservation keys. These rules provide the ability for an application client to preempt multiple I_T nexuses with a single PERSISTENT RESERVE OUT command, but they do not provide the ability for the application client to uniquely identify the I_T nexuses using the PERSISTENT RESERVE commands.

See table 216 in 6.16.2 for a list of PERSISTENT RESERVE OUT service actions. See table 204 in 6.15.1 for a list of PERSISTENT RESERVE IN service actions.

The scope (see 6.15.3.2) of a persistent reservation shall be the entire logical unit.

The type (see 6.15.3.3) of a persistent reservation defines the selected set of I_T nexuses for which the persistent reservation places restrictions on commands.

The details of which commands are allowed under what types of reservations are described in table 66.

In table 66 and table 67 the following keywords are used:

a) **allowed:** Commands received from I_T nexuses not holding the reservation or from I_T nexuses not registered if a registrants only or all registrants type persistent reservation is present should complete normally.
b) **conflict:** Commands received from I_T nexuses not holding the reservation or from I_T nexuses not registered if a registrants only or all registrants type persistent reservation is present shall not be performed and the device server shall complete the command with RESERVATION CONFLICT status.

Commands from I_T nexuses holding a reservation should complete normally. The behavior of commands from registered I_T nexuses if a registrants only or all registrants type persistent reservation is present is defined in table 66 and table 67.

A command shall be checked for reservation conflicts when the device server begins processing of the command. After that check succeeds, the device server shall not complete the command with RESERVATION CONFLICT status due to a subsequent reservation.

The time at which a reservation is established with respect to other commands 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 commands in the task set before the completion of the reservation command. The reservation shall apply to all commands received by the device server after successful completion of the reservation command. Any persistent reserve service action shall be performed as a single indivisible event.

Multiple persistent reserve service actions may be present in the task set at the same time. The order of processing of such service actions is defined by the task set management requirements defined in SAM-5, but each is processed as a single indivisible command without any interleaving of actions that may be required by other reservation commands.

For each command, this standard or a command standard defines the conditions that result in the command being completed with RESERVATION CONFLICT. Command standards define the conditions either in the device model or in the descriptions of each specific command.

Table 66 — SPC-4 commands that are allowed in the presence of various reservations (part 1 of 3)

Command	Addressed logical unit has this type of persistent reservation held by another I_T nexus				
	From any I_T nexus		From registered I_T nexus (RR all types)	From not registered I_T nexus	
	Write Excl	Excl Access		Write Excl RR	Excl Access – RR
ACCESS CONTROL IN	Allowed	Allowed	Allowed	Allowed	Allowed
ACCESS CONTROL OUT	Allowed	Allowed	Allowed	Allowed	Allowed
CHANGE ALIASES	Conflict	Conflict	Allowed	Conflict	Conflict
COPY OPERATION ABORT	Conflict	Conflict	Allowed	Conflict	Conflict
EXTENDED COPY(LID4)	Conflict	Conflict	Allowed	Conflict	Conflict
EXTENDED COPY(LID1)	Conflict	Conflict	Allowed	Conflict	Conflict
INQUIRY	Allowed	Allowed	Allowed	Allowed	Allowed
LOG SELECT	Conflict	Conflict	Allowed	Conflict	Conflict
LOG SENSE	Allowed	Allowed	Allowed	Allowed	Allowed
MANAGEMENT PROTOCOL IN	Allowed	Conflict	Allowed	Allowed	Conflict
MANAGEMENT PROTOCOL OUT	Conflict	Conflict	Allowed	Conflict	Conflict
MODE SELECT(6) / MODE SELECT(10)	Conflict	Conflict	Allowed	Conflict	Conflict
MODE SENSE(6) / MODE SENSE(10)	Allowed[b]	Conflict	Allowed	Allowed[b]	Conflict
PERSISTENT RESERVE IN	Allowed	Allowed	Allowed	Allowed	Allowed
PERSISTENT RESERVE OUT	see table 67				
READ ATTRIBUTE	Allowed[b]	Conflict	Allowed	Allowed[b]	Conflict

Key: **Excl**=Exclusive, **RR**=Registrants Only or All Registrants

[a] Exceptions to the behavior of the RESERVE and RELEASE commands described in SPC-2 are defined in 5.12.3.

[b] Logical units claiming compliance with SPC-2 or SPC-3 may return RESERVATION CONFLICT status in this case. Logical units may report whether certain commands are allowed in the ALLOW COMMANDS field of the parameter data returned by the PERSISTENT RESERVE IN command with REPORT CAPABILITIES service action (see 6.15.4).

Table 66 — SPC-4 commands that are allowed in the presence of various reservations (part 2 of 3)

Command	Addressed logical unit has this type of persistent reservation held by another I_T nexus				
	From any I_T nexus		From registered I_T nexus (RR all types)	From not registered I_T nexus	
	Write Excl	Excl Access		Write Excl RR	Excl Access – RR
READ BUFFER	Allowed[b]	Conflict	Allowed	Allowed[b]	Conflict
READ MEDIA SERIAL NUMBER	Allowed	Allowed	Allowed	Allowed	Allowed
RECEIVE CREDENTIAL	Conflict	Conflict	Allowed	Conflict	Conflict
RECEIVE COPY DATA(LID4)	Allowed	Allowed	Allowed	Allowed	Allowed
RECEIVE COPY DATA(LID1)	Allowed[b]	Allowed[b]	Allowed	Allowed[b]	Allowed[b]
RECEIVE COPY OPERATING PARAMETERS	Allowed[b]	Allowed[b]	Allowed	Allowed[b]	Allowed[b]
RECEIVE COPY FAILURE DETAILS(LID1)	Allowed[b]	Allowed[b]	Allowed	Allowed[b]	Allowed[b]
RECEIVE COPY STATUS(LID4)	Allowed	Allowed	Allowed	Allowed	Allowed
RECEIVE COPY STATUS(LID1)	Allowed[b]	Allowed[b]	Allowed	Allowed[b]	Allowed[b]
RECEIVE ROD TOKEN INFORMATION	Allowed	Allowed	Allowed	Allowed	Allowed
RECEIVE DIAGNOSTIC RESULTS	Allowed[b]	Conflict	Allowed	Allowed[b]	Conflict
RELEASE(6) / RELEASE(10)	As defined in SPC-2 [a]				
REMOVE I_T NEXUS	Conflict	Conflict	Allowed	Conflict	Conflict
REPORT ALIASES	Allowed	Allowed	Allowed	Allowed	Allowed
REPORT ALL ROD TOKENS	Allowed	Allowed	Allowed	Allowed	Allowed
REPORT IDENTIFYING INFORMATION	Allowed	Allowed	Allowed	Allowed	Allowed
REPORT LUNS	Allowed	Allowed	Allowed	Allowed	Allowed
REPORT PRIORITY	Allowed	Allowed	Allowed	Allowed	Allowed
REPORT SUPPORTED OPERATION CODES	Allowed[b]	Allowed[b]	Allowed	Allowed[b]	Allowed[b]
REPORT SUPPORTED TASK MANAGEMENT FUNCTIONS	Allowed[b]	Allowed[b]	Allowed	Allowed[b]	Allowed[b]
REPORT TARGET PORT GROUPS	Allowed	Allowed	Allowed	Allowed	Allowed
REPORT TIMESTAMP	Allowed	Allowed	Allowed	Allowed	Allowed
REQUEST SENSE	Allowed	Allowed	Allowed	Allowed	Allowed
RESERVE(6) / RESERVE(10)	As defined in SPC-2 [a]				

Key: **Excl**=Exclusive, **RR**=Registrants Only or All Registrants

[a] Exceptions to the behavior of the RESERVE and RELEASE commands described in SPC-2 are defined in 5.12.3.

[b] Logical units claiming compliance with SPC-2 or SPC-3 may return RESERVATION CONFLICT status in this case. Logical units may report whether certain commands are allowed in the ALLOW COMMANDS field of the parameter data returned by the PERSISTENT RESERVE IN command with REPORT CAPABILITIES service action (see 6.15.4).

Table 66 — SPC-4 commands that are allowed in the presence of various reservations (part 3 of 3)

Command	Addressed logical unit has this type of persistent reservation held by another I_T nexus				
	From any I_T nexus		From registered I_T nexus (RR all types)	From not registered I_T nexus	
	Write Excl	Excl Access		Write Excl RR	Excl Access – RR
SECURITY PROTOCOL IN	Allowed	Conflict	Allowed	Allowed	Conflict
SECURITY PROTOCOL OUT	Conflict	Conflict	Allowed	Conflict	Conflict
SEND DIAGNOSTIC	Conflict	Conflict	Allowed	Conflict	Conflict
SET IDENTIFYING INFORMATION	Conflict	Conflict	Allowed	Conflict	Conflict
SET PRIORITY	Conflict	Conflict	Allowed	Conflict	Conflict
SET TARGET PORT GROUPS	Conflict	Conflict	Allowed	Conflict	Conflict
SET TIMESTAMP	Conflict	Conflict	Allowed	Conflict	Conflict
TEST UNIT READY	Allowed [b]	Allowed [b]	Allowed	Allowed [b]	Allowed [b]
WRITE ATTRIBUTE	Conflict	Conflict	Allowed	Conflict	Conflict
WRITE BUFFER	Conflict	Conflict	Allowed	Conflict	Conflict

Key: **Excl**=Exclusive, **RR**=Registrants Only or All Registrants

[a] Exceptions to the behavior of the RESERVE and RELEASE commands described in SPC-2 are defined in 5.12.3.

[b] Logical units claiming compliance with SPC-2 or SPC-3 may return RESERVATION CONFLICT status in this case. Logical units may report whether certain commands are allowed in the ALLOW COMMANDS field of the parameter data returned by the PERSISTENT RESERVE IN command with REPORT CAPABILITIES service action (see 6.15.4).

Table 67 — PERSISTENT RESERVE OUT service actions that are allowed in the presence of various reservations

Service action	Addressed logical unit has a persistent reservation held by another I_T nexus	
	Command is from a registered I_T nexus	Command is from a not registered I_T nexus
CLEAR	Allowed	Conflict
PREEMPT	Allowed	Conflict
PREEMPT AND ABORT	Allowed	Conflict
REGISTER	Allowed	Allowed
REGISTER AND IGNORE EXISTING KEY	Allowed	Allowed
REGISTER AND MOVE	Conflict	Conflict
RELEASE	Allowed [a]	Conflict
REPLACE LOST RESERVATION [b]	Allowed	Conflict
RESERVE	Conflict	Conflict

[a] The reservation is not released (see 5.12.11.2.2).

[b] If the device server has not detected that persistent reservation information has been lost, then the command shall be processed as shown in this table. If the device server has detected that persistent reservation information has been lost, then the command shall be processed as described in 5.12.5.4.

5.12.2 Third party persistent reservations

Except for all registrants type reservations, a reservation holder (see 5.12.10) may move the persistent reservation to a third party (e.g., a copy manager supporting the EXTENDED COPY command) using the REGISTER AND MOVE service action (see 5.12.8). A copy manager supporting the EXTENDED COPY command may be instructed to move the persistent reservation to a specified I_T nexus using the third party persistent reservations source I_T nexus segment descriptor (see 6.4.6.18).

5.12.3 Exceptions to SPC-2 RESERVE and RELEASE behavior

This subclause defines exceptions to the behavior of the RESERVE command and RELEASE command defined in SPC-2. The RESERVE command and RELEASE command are obsolete in this standard, except for the behavior defined in this subclause. Device servers that operate using the exceptions described in this subclause shall set the CRH bit to one in the parameter data returned by the REPORT CAPABILITIES service action of the PERSISTENT RESERVE IN command (see 6.15.4).

A RELEASE(6) command or RELEASE(10) command shall complete with GOOD status, but the persistent reservation shall not be released, if the command is received from:

a) an I_T nexus that is a persistent reservation holder (see 5.12.10); or
b) an I_T nexus that is registered if a registrants only or all registrants type persistent reservation is present.

A RESERVE(6) command or RESERVE(10) command shall complete with GOOD status, but no reservation shall be established and the persistent reservation shall not be changed, if the command is received from:

a) an I_T nexus that is a persistent reservation holder; or
b) an I_T nexus that is registered if a registrants only or all registrants type persistent reservation is present.

In all other cases, the device server shall process a RESERVE(6) command, RESERVE(10) command, RELEASE(6) command, or RELEASE(10) command as defined in SPC-2.

5.12.4 Persistent reservations interactions with IKEv2-SCSI SA creation

If a PERSISTENT RESERVE OUT command is received while an IKEv2-SCSI CCS is in progress (see 5.13.4), the device server shall terminate the command with CHECK CONDITION status, with the sense key NOT READY, and the additional sense code set to LOGICAL UNIT NOT READY, SA CREATION IN PROGRESS. The sense key specific additional sense data may be set as described in 5.13.5.

5.12.5 Preserving persistent reservations and registrations

5.12.5.1 Requirements for preserving persistent reservations and registrations

The device server shall preserve the following information for each existing registration across any hard reset, logical unit reset, or I_T nexus loss, and if the persist through power loss capability is enabled (see 5.12.5.2), across any power cycle:

a) for SCSI transport protocols where initiator port names are required, the initiator port name; otherwise, the initiator port identifier;
b) reservation key; and
c) indication of the target port to which the registration was applied.

The device server shall preserve the following information about the existing persistent reservation across any hard reset, logical unit reset, or I_T nexus loss, and if the persist through power loss capability is enabled (see 5.12.5.2), across any power cycle:

a) for SCSI transport protocols where initiator port names are required, the initiator port name; otherwise, the initiator port identifier;
b) reservation key;
c) scope;
d) type; and
e) indication of the target port through which the reservation was established.

NOTE 13 - The scope of a persistent reservation is always LU_SCOPE (see 6.15.3.2). For an all registrants type persistent reservation, preserving the scope and type is sufficient.

5.12.5.2 Preserving persistent reservations and registrations through power loss

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 the PERSISTENT RESERVE OUT parameter data associated with a REGISTER service action, REGISTER AND IGNORE EXISTING KEY service action, or REGISTER AND MOVE service action.

After the application client enables the persist through power loss capability the device server shall preserve the persistent reservation, if any, and all current and future registrations associated with the logical unit to which the REGISTER service action, the REGISTER AND IGNORE EXISTING KEY service action, or the REGISTER AND MOVE 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 service action, REGISTER AND IGNORE EXISTING KEY service action, or REGISTER AND MOVE service action from any application client shall determine the logical unit's behavior in the event of a power loss.

5.12.5.3 Nonvolatile memory considerations for preserving persistent reservations and registrations

The capability of preserving persistent reservations and registrations across power cycles requires logical units to use nonvolatile memory within the SCSI device. Any logical unit 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) REPORT TARGET PORT GROUPS;
f) REQUEST SENSE;
g) START STOP UNIT with the START bit set to one and the POWER CONDITION field set to 0h (see SBC-3); and
h) WRITE BUFFER.

Until nonvolatile memory has become ready after a power cycle, commands other than those listed in this subclause shall be terminated with CHECK CONDITION status, with the sense key set to NOT READY, and the additional sense code set as described in table 336 (see 6.47).

5.12.5.4 Loss of persistent reservation information

5.12.5.4.1 Loss of persistent reservation information overview

While the persist through power loss capability is enabled (see 5.12.5.2), the device server may detect a failure (e.g., a hardware failure in nonvolatile memory) that causes the loss of the preserved persistent reservation information.

The failure detected by the device server may be:

a) recoverable through the combined actions of the device server and application client (e.g., sufficient nonvolatile memory is available to recreate the lost persistent reservation and registrations information) using the processes described in 5.12.5.4.2 (i.e., a recoverable lost persistent reservation); or
b) unrecoverable, except by operator intervention (i.e., an unrecoverable lost persistent reservation).

5.12.5.4.2 Recoverable loss of persistent reservation information

If the device server detects a recoverable lost persistent reservation, the device server shall establish a recoverable lost persistent reservation condition. A recoverable lost persistent reservation condition is a condition in which the device server shall:

a) not terminate a PERSISTENT RESERVE OUT command with the REPLACE LOST RESERVATION service action with RESERVATION CONFLICT status; and
b) terminate with CHECK CONDITION status with the sense key set to DATA PROTECT and the additional sense code set to PERSISTENT RESERVATION INFORMATION LOST all commands other than:
   A) a PERSISTENT RESERVE OUT command with a REPLACE LOST RESERVATION service action; and
   B) those commands listed in 5.12.5.3.

The device server shall clear a recoverable lost persistent reservation condition in response to:

a) the successful processing of a PERSISTENT RESERVE OUT command with the REPLACE LOST RESERVATION service action (see 5.12.11.3); or
b) the recoverable lost persistent reservation becoming an unrecoverable lost persistent reservation.

The device server shall not clear a recoverable lost persistent reservation condition for any reason other than the reasons described in this subclause.

5.12.5.4.3 Unrecoverable loss of persistent reservation information overview

If the device server detects an unrecoverable lost persistent reservation, then the device server:

a) should operate as if it has non volatile memory that is not ready (see 5.12.5.3); or
b) may terminate commands other than those commands listed in 5.12.5.3 with CHECK CONDITION status with the sense key set to HARDWARE ERROR with the additional sense code set to an appropriate value.

5.12.6 Finding persistent reservations and reservation keys

5.12.6.1 Summary of commands for finding persistent reservations and reservation keys

The application client may obtain information about the persistent reservation and the reservation keys (i.e., registrations) that are present within a device server by issuing a PERSISTENT RESERVE IN command with a READ RESERVATION service action, a READ KEYS service action, or a READ FULL STATUS service action.

5.12.6.2 Reporting reservation keys

An application client may send a PERSISTENT RESERVE IN command with READ KEYS service action to determine if any I_T nexuses have been registered with a logical unit through any target port.

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

a) the current PRgeneration value (see 6.15.2); and
b) the reservation key for every I_T nexus that is currently registered regardless of the target port through which the registration occurred.

The PRgeneration value allows the application client to verify that the configuration of the I_T nexuses registered with a logical unit has not been modified (i.e., if the PRgeneration value is not changed, the configuration of the I_T nexus registered with a logical unit has not been modified).

Duplicate reservation keys shall be reported if multiple I_T nexuses are registered using the same reservation key.

If an application client uses a different reservation key for each I_T nexus, the application client may use the reservation key to uniquely identify an I_T nexus.

5.12.6.3 Reporting the persistent reservation

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

In response to a PERSISTENT RESERVE IN command with READ RESERVATION service action the device server shall report the following information for the persistent reservation, if any:

a) the current PRgeneration value (see 6.15.2);
b) the registered reservation key, if any, associated with the I_T nexus that holds the persistent reservation (see 5.12.10). If the persistent reservation is an all registrants type, the registered reservation key reported shall be zero; and
c) the scope and type of the persistent reservation, if any.

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

5.12.6.4 Reporting full status

An application client may send a PERSISTENT RESERVE IN command with READ FULL STATUS service action to receive all information about registrations and the persistent reservation, if any.

In response to a PERSISTENT RESERVE IN command with READ FULL STATUS service action the device server shall report the current PRgeneration value (see 6.15.2) and, for every I_T nexus that is currently registered, the following information:

a) the registered reservation key;
b) whether the I_T nexus is a persistent reservation holder;
c) if the I_T nexus is a persistent reservation holder, the scope and type of the persistent reservation;
d) the relative target port identifier identifying the target port of the I_T nexus; and
e) a TransportID identifying the initiator port of the I_T nexus.

5.12.7 Registering

To establish a persistent reservation the application client shall first register an I_T nexus with the device server. An application client registers with a logical unit by issuing a PERSISTENT RESERVE OUT command with REGISTER service action or REGISTER AND IGNORE EXISTING KEY service action.

If the I_T nexus has an established registration, an application client may remove the reservation key (see 5.12.11.2.3). This is accomplished by issuing a PERSISTENT RESERVE OUT command with a REGISTER service action or a REGISTER AND IGNORE EXISTING KEY service action as shown in table 68 and table 69, respectively.

If an I_T nexus has not yet established a reservation key or the reservation key and registration have been removed, then an application client may register that I_T nexus and zero or more specified unregistered I_T nexuses by issuing a PERSISTENT RESERVE OUT command with REGISTER service action as defined in table 68.

If the I_T nexus has an established registration, the application client may change the reservation key by issuing a PERSISTENT RESERVE OUT command with REGISTER service action as defined in table 68.

Table 68 — Register behaviors for a REGISTER service action

Command I_T nexus status	Parameter list fields [a]			Device server action
	RESERVATION KEY	SERVICE ACTION RESERVATION KEY	SPEC_I_PT	
received on an unregistered I_T nexus	zero	zero	ignore	Do nothing except return GOOD status.
		non-zero	zero	Register the I_T nexus on which the command was received with the value specified in the SERVICE ACTION RESERVATION KEY field.
			one	Register the I_T nexus on which the command was received and each unregistered I_T nexus specified in the parameter list with the value specified in the SERVICE ACTION RESERVATION KEY field. [b]
	non-zero	ignore	ignore	Return RESERVATION CONFLICT status.
received on a registered I_T nexus	Not equal to I_T nexus reservation key	ignore	ignore	Return RESERVATION CONFLICT status.
	Equal to I_T nexus reservation key	zero	zero	Unregister the I_T nexus on which the command was received (see 5.12.11.2.3).
			one	Return CHECK CONDITION status. [c]
		non-zero	zero	Change the reservation key of the I_T nexus on which the command was received to the value specified in the SERVICE ACTION RESERVATION KEY field.
			one	Return CHECK CONDITION status. [c]

[a] For requirements regarding the parameter list fields not shown in this table, see 6.16.3.

[b] If any I_T nexus specified in the parameter list is registered, the device server shall terminate the command with CHECK CONDITION status, with the sense key set to ILLEGAL REQUEST, and the additional sense code set to INVALID FIELD IN PARAMETER LIST. Devices compliant with SPC-3 may return an additional sense code set to INVALID FIELD IN CDB.

[c] The sense key shall be set to ILLEGAL REQUEST, and the additional sense code shall be set to INVALID FIELD IN PARAMETER LIST. Devices compliant with SPC-3 may return an additional sense code set to INVALID FIELD IN CDB.

Alternatively, an application client may establish a reservation key for an I_T nexus without regard for whether one has previously been established by issuing a PERSISTENT RESERVE OUT command with REGISTER AND IGNORE EXISTING KEY service action as defined in table 69.

Table 69 — Register behaviors for a REGISTER AND IGNORE EXISTING KEY service action

Command I_T nexus status	Parameter list fields [a]	
	SERVICE ACTION RESERVATION KEY	Results
received on an unregistered I_T nexus	zero	Do nothing except return GOOD status.
	non-zero	Register the I_T nexus on which the command was received with the value specified in the SERVICE ACTION RESERVATION KEY field.
received on a registered I_T nexus	zero	Unregister the I_T nexus on which the command was received (see 5.12.11.2.3).
	non-zero	Change the reservation key of the I_T nexus on which the command was received to the value specified in the SERVICE ACTION RESERVATION KEY field.

[a] The RESERVATION KEY field is ignored when processing a REGISTER AND IGNORE EXISTING KEY service action. For requirements regarding other parameter list fields not shown in this table, see 6.16.3.

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

In response to a PERSISTENT RESERVE OUT command with a REGISTER service action or a REGISTER AND IGNORE EXISTING KEY service action the device server shall perform a registration for each specified I_T nexus 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) associate the reservation key specified in the SERVICE ACTION RESERVATION KEY field with the I_T nexus being registered, where:
   A) the I_T nexus(es) being registered are shown in table 70; and
   B) regardless of how the I_T nexus initiator port is specified, the association for the initiator port is based on either the initiator port name on SCSI transport protocols where initiator port names are required or the initiator port identifier on SCSI transport protocols where initiator port names are not required;
e) register the reservation key specified in the SERVICE ACTION RESERVATION KEY field without changing any persistent reservation that may exist; and
f) retain the reservation key specified in the SERVICE ACTION RESERVATION KEY field and associated information.

Table 70 — I_T Nexuses being registered

SPEC_I_PT [a]	ALL_TG_PT	I_T nexus(es) being registered	
		Initiator port	Target port
0	0	The port's names or identifiers to be registered are determined from the I_T nexus on which the PERSISTENT RESERVE OUT command was received	
0	1	The port's name or identifier to be registered is determined from the I_T nexus on which the PERSISTENT RESERVE OUT command was received	Register all of the target ports in the SCSI target device
1	0	a) The port's name or identifier to be registered is determined from the I_T nexus on which the PERSISTENT RESERVE OUT command was received; and b) Specified by each TransportID in the additional parameter data (see 6.16.3)	The port's name or identifier to be registered is determined from the I_T nexus on which the PERSISTENT RESERVE OUT command was received
1	1	a) The port's name or identifier to be registered is determined from the I_T nexus on which the PERSISTENT RESERVE OUT command was received; and b) Specified by each TransportID in the additional parameter data	Register all of the target ports in the SCSI target device

[a] If the SPEC_I_PT bit is set to one and the service action is REGISTER AND IGNORE EXISTING KEY, then the device server shall terminate the command with CHECK CONDITION status, with the sense key set to ILLEGAL REQUEST, and the additional sense code set to INVALID FIELD IN PARAMETER LIST.

After the registration request has been processed, the device server shall then allow other PERSISTENT RESERVE OUT commands from the registered I_T nexus to be processed. The device server shall retain the reservation key until the key is changed as described in this subclause or removed as described in 5.12.11.

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

It is not an error for an I_T nexus that is registered to be registered again with the same reservation key or a new reservation key. A registration shall have no effect on any other registrations (e.g., if more than one I_T nexus is registered with the same reservation key and one of those I_T nexuses registers again it has no effect on the other I_T nexus' registrations). A registration that contains a non-zero value in the SERVICE ACTION RESERVATION KEY field shall have no effect on any persistent reservations (i.e., the reservation key for an I_T nexus may be changed without affecting any previously created persistent reservation).

Multiple I_T nexuses may be registered with the same reservation key. An application client may use the same reservation key for other I_T nexuses and logical units.

5.12.8 Registering and moving the reservation

The PERSISTENT RESERVE OUT command REGISTER AND MOVE service action is used to register a specified I_T nexus (see table 71) and move the reservation to establish that I_T nexus as the reservation holder.

Table 71 — Register behaviors for a REGISTER AND MOVE service action

Command I_T nexus status	Parameter list fields [a]			Device server action
	RESERVATION KEY	SERVICE ACTION RESERVATION KEY	UNREG	
received on an unregistered I_T nexus	ignore	ignore	ignore	Return RESERVATION CONFLICT status. [d]
received on the registered I_T nexus of reservation holder	Not equal to I_T nexus reservation key	ignore	ignore	Return RESERVATION CONFLICT status.
	Equal to I_T nexus reservation key	zero	ignore	Return CHECK CONDITION status. [b]
		non-zero [c]	zero	The I_T nexus on which PERSISTENT RESERVE OUT command was received shall remain registered. See this subclause for the registration and the move specifications.
			one	The I_T nexus on which PERSISTENT RESERVE OUT command was received shall be unregistered (see 5.12.11.2.3) upon completion of command processing. See this subclause for the registration and the move specifications.
received on a registered I_T nexus that is not the reservation holder	ignore	ignore	ignore	Return RESERVATION CONFLICT status.

[a] For requirements regarding other parameter list fields not shown in this table see 6.16.4.

[b] The sense key shall be set to ILLEGAL REQUEST, and the additional sense code shall be set to INVALID FIELD IN PARAMETER LIST. Devices compliant with SPC-3 may return an additional sense code set to INVALID FIELD IN CDB.

[c] The application client and backup application should use the same reservation key.

[d] Devices compliant with SPC-3 may return CHECK CONDITION status with the sense key set to ILLEGAL REQUEST and the additional sense code set to INVALID FIELD IN CDB.

If a PERSISTENT RESERVE OUT command with a REGISTER AND MOVE service action is attempted, but there are insufficient device server resources to complete the operation, then the device server shall terminate the command with CHECK CONDITION status, with the sense key set to ILLEGAL REQUEST, and the additional sense code set to INSUFFICIENT REGISTRATION RESOURCES.

If a PERSISTENT RESERVE OUT command with a REGISTER AND MOVE service action is received and the established persistent reservation is a Write Exclusive - All Registrants type reservation or Exclusive

Access - All Registrants type reservation, then the device server shall complete the command with RESERVATION CONFLICT status.

If a PERSISTENT RESERVE OUT command with a REGISTER AND MOVE service action is received and there is no persistent reservation established, then the device server shall terminate the command with CHECK CONDITION status, with the sense key set to ILLEGAL REQUEST, and the additional sense code set to INVALID FIELD IN CDB.

If a PERSISTENT RESERVE OUT command with a REGISTER AND MOVE service action specifies a TransportID that is the same as the initiator port of the I_T nexus on which the command received, then the device server shall terminate the command with CHECK CONDITION status, with the sense key set to ILLEGAL REQUEST, and the additional sense code set to INVALID FIELD IN PARAMETER LIST.

In response to a PERSISTENT RESERVE OUT command with a REGISTER AND MOVE service action the device server shall perform a register and move by doing the following as an uninterrupted series of actions:

a) process the APTPL bit;
b) ignore the contents of the SCOPE and TYPE fields;
c) associate the reservation key specified in the SERVICE ACTION RESERVATION KEY field with the I_T nexus specified as the destination of the register and move, where:
   A) the I_T nexus is specified by the TransportID and the RELATIVE TARGET PORT IDENTIFIER field (see 6.16.4); and
   B) regardless of the TransportID format used, the association for the initiator port is based on either the initiator port name on SCSI transport protocols where initiator port names are required or the initiator port identifier on SCSI transport protocols where initiator port names are not required;
d) register the reservation key specified in the SERVICE ACTION RESERVATION KEY field;
e) retain the reservation key specified in the SERVICE ACTION RESERVATION KEY field and associated information;
f) release the persistent reservation for the persistent reservation holder (i.e., the I_T nexus on which the command was received);
g) move the persistent reservation to the specified I_T nexus using the same scope and type as the persistent reservation released in item f); and
h) if the UNREG bit is set to one, unregister (see 5.12.11.2.3) the I_T nexus on which PERSISTENT RESERVE OUT command was received.

It is not an error for a REGISTER AND MOVE service action to register an I_T nexus that is already registered with the same reservation key or a different reservation key.

5.12.9 Reserving

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

a) RESERVATION KEY set to the value of the reservation key that is registered with the logical unit for the I_T nexus; and
b) TYPE field and SCOPE field set to the persistent reservation being created.

Only one persistent reservation is allowed at a time per logical unit and that persistent reservation has a scope of LU_SCOPE.

If the device server receives a PERSISTENT RESERVE OUT command from an I_T nexus other than a persistent reservation holder (see 5.12.10) that attempts to create a persistent reservation when a persistent reservation already exists for the logical unit, then the device server shall complete the command with RESERVATION CONFLICT status.

If a persistent reservation holder attempts to modify the type or scope of an existing persistent reservation, the device server shall complete the command with RESERVATION CONFLICT status.

If the device server receives a PERSISTENT RESERVE OUT command with RESERVE service action where the TYPE field and the SCOPE field contain the same values as the existing type and scope from a persistent reservation holder, then the device server shall not make any change to the existing persistent reservation and shall complete the command with GOOD status.

See 5.12.1 for information on when a persistent reservation takes effect.

5.12.10 Persistent reservation holder

The persistent reservation holder is determined by the type of the persistent reservation as follows:

a) for a persistent reservation of the type Write Exclusive – All Registrants or Exclusive Access – All Registrants, the persistent reservation holder is any registered I_T nexus; or
b) for all other persistent reservation types, the persistent reservation holder is the I_T nexus:
   A) for which the reservation was established with a PERSISTENT RESERVE OUT command with the RESERVE service action, the PREEMPT service action, the PREEMPT AND ABORT service action, or the REPLACE LOST RESERVATION service action; or
   B) to which the reservation was moved by a PERSISTENT RESERVE OUT command with REGISTER AND MOVE service action.

A persistent reservation holder has its reservation key returned in the parameter data from a PERSISTENT RESERVE IN command with READ RESERVATION service action as follows:

a) for a persistent reservation of the type Write Exclusive – All Registrants or Exclusive Access – All Registrants, the reservation key shall be set to zero; or
b) for all other persistent reservation types, the reservation key shall be set to the registered reservation key for the I_T nexus that holds the persistent reservation.

It is not an error for a persistent reservation holder to send a PERSISTENT RESERVE OUT command with RESERVE service action to the reserved logical unit with TYPE and SCOPE fields that match those of the persistent reservation (see 5.12.9).

A persistent reservation holder is allowed to release the persistent reservation using the PERSISTENT RESERVE OUT command with RELEASE service action (see 5.12.11.2.2).

If the registration of the persistent reservation holder is removed (see 5.12.11.2), the reservation shall be released. If the persistent reservation holder is more than one I_T nexus, the reservation shall not be released until the registrations for all persistent reservation holder I_T nexuses are removed.

5.12.11 Releasing persistent reservations and removing registrations

5.12.11.1 Releasing persistent reservations, removing registrations, and lost reservation information

The application client may use PERSISTENT RESERVE OUT command service actions to:

a) release persistent reservations and remove registrations (see 5.12.11.2); and
b) begin the process of recovering from lost reservation information, if any (see 5.12.5.4 and 5.12.11.3).

5.12.11.2 Service actions that release persistent reservations and remove registrations

5.12.11.2.1 Service actions that release persistent reservations and remove registrations overview

An application client may release or preempt the persistent reservation by issuing one of the following commands through a registered I_T nexus with the RESERVATION KEY field set to the reservation key value that is registered with the logical unit for that I_T nexus:

a) a PERSISTENT RESERVE OUT command with RELEASE service action from a persistent reservation holder (see 5.12.11.2.2);
b) a PERSISTENT RESERVE OUT command with PREEMPT service action specifying the reservation key of the persistent reservation holder or holders (see 5.12.11.2.4);
c) a PERSISTENT RESERVE OUT command with PREEMPT AND ABORT service action specifying the reservation key of the persistent reservation holder or holders (see 5.12.11.2.6);
d) a PERSISTENT RESERVE OUT command with CLEAR service action (see 5.12.11.2.7); or
e) if the I_T nexus is the persistent reservation holder and the persistent reservation is not an all registrants type, then a PERSISTENT RESERVE OUT command with REGISTER service action or REGISTER AND IGNORE EXISTING KEY service action with the SERVICE ACTION RESERVATION KEY field set to zero (see 5.12.11.2.3).

Table 72 defines processing for a persistent reservation released or preempted by an application client based on the reservation type.

Table 72 — Processing for a released or preempted persistent reservation

Reservation Type	Processing
Write Exclusive – Registrants Only or Exclusive Access – Registrants Only	This persistent reservation shall be released if the persistent reservation holder (see 5.12.10) of this reservation type becomes unregistered.
Write Exclusive – All Registrants or Exclusive Access – All Registrants	This persistent reservation shall be released if: a) the registration for the last registered I_T nexus is removed; or b) the type or scope is changed.
Write Exclusive or Exclusive Access	This persistent reservation shall be released if the persistent reservation holder (see 5.12.10) of this reservation type becomes unregistered.

An application client may remove registrations by issuing one of the following commands through a registered I_T nexus with the RESERVATION KEY field set to the reservation key value that is registered with the logical unit for that I_T nexus:

a) a PERSISTENT RESERVE OUT command with PREEMPT service action with the SERVICE ACTION RESERVATION KEY field set to the reservation key (see 5.12.11.2.4) to be removed;
b) a PERSISTENT RESERVE OUT command with PREEMPT AND ABORT service action with the SERVICE ACTION RESERVATION KEY field set to the reservation key (see 5.12.11.2.6) to be removed;
c) a PERSISTENT RESERVE OUT command with CLEAR service action (see 5.12.11.2.7); or
d) a PERSISTENT RESERVE OUT command with REGISTER service action or REGISTER AND IGNORE EXISTING KEY service action with the SERVICE ACTION RESERVATION KEY field set to zero (see 5.12.11.2.3).

After a reservation key (i.e., registration) has been removed, no information shall be reported for that unregistered I_T nexus in subsequent READ KEYS service actions until the I_T nexus is registered again (see 5.12.7).

If the persist through power loss capability is not enabled, loss of power also causes persistent reservations to be released and registrations to be removed. If the most recent APTPL value received by the device server is zero (see 6.16.3), the processing of a power on condition:

a) releases all persistent reservations; and
b) removes all registered reservation keys (see 5.12.7).

5.12.11.2.2 Releasing

Only the persistent reservation holder (see 5.12.10) is allowed to release a persistent reservation.

An application client releases the persistent reservation by issuing a PERSISTENT RESERVE OUT command with RELEASE service action through an I_T nexus that is a persistent reservation holder with the following parameters:

a) RESERVATION KEY field set to the value of the reservation key that is registered with the logical unit for the I_T nexus; and
b) TYPE field and SCOPE field set to match the persistent reservation being released.

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

a) release the persistent reservation;
b) not remove any registration(s);
c) if the NUAR bit (see 7.5.8) is set to zero and the released persistent reservation is either a registrants only type or an all registrants type persistent reservation, then the device server shall establish a unit attention condition for the initiator port associated with every registered I_T nexus other than I_T nexus on which the PERSISTENT RESERVE OUT command with RELEASE service action was received, with the additional sense code set to RESERVATIONS RELEASED;
d) if the NUAR bit is set to one and the released persistent reservation is either a registrants only type or an all registrants type persistent reservation, then the device server shall not establish a unit attention condition; and
e) if the persistent reservation is of any other type, the device server shall not establish a unit attention condition.

The device server shall not alter the established persistent reservation and shall terminate the command with CHECK CONDITION status, with the sense key set to ILLEGAL REQUEST, and the additional sense code set to INVALID RELEASE OF PERSISTENT RESERVATION, for a PERSISTENT RESERVE OUT command that specifies the release of a persistent reservation if:

a) the requesting I_T nexus is a persistent reservation holder (see 5.12.10); and
b) the SCOPE and TYPE fields do not match the scope and type of the established persistent reservation.

If there is no persistent reservation or in response to a persistent reservation release request from a registered I_T nexus that is not a persistent reservation holder (see 5.12.10), then the device server shall do the following:

a) not release the persistent reservation, if any;
b) not remove any registrations; and
c) complete the command with GOOD status.

5.12.11.2.3 Unregistering

An application client may remove a registration for an I_T nexus by issuing a PERSISTENT RESERVE OUT command with REGISTER service action or a REGISTER AND IGNORE EXISTING KEY service action with the SERVICE ACTION RESERVATION KEY field set to zero through that I_T nexus.

If the I_T nexus is a reservation holder, the persistent reservation is of an all registrants type, and the I_T nexus is the last remaining registered I_T nexus, then the device server shall also release the persistent reservation.

If the I_T nexus is the reservation holder and the persistent reservation is of a type other than all registrants, then the device server shall also release the persistent reservation. If the persistent reservation is a registrants only type, then the device server shall establish a unit attention condition for the initiator port associated with every registered I_T nexus except for the I_T nexus on which the PERSISTENT RESERVE OUT command was received, with the additional sense code set to RESERVATIONS RELEASED.

5.12.11.2.4 Preempting

5.12.11.2.4.1 Commands that preempt reservations

A PERSISTENT RESERVE OUT command with PREEMPT service action or PREEMPT AND ABORT service action is used to:

a) preempt (i.e., replace) the persistent reservation and remove registrations (see 5.12.11.2.4.3); or
b) remove registrations (see 5.12.11.2.5).

Table 73 lists the actions taken by the device server based on the current persistent reservation type and the SERVICE ACTION RESERVATION KEY field in the PERSISTENT RESERVE OUT command.

Table 73 — Preempting actions

Reservation Type	Service Action Reservation Key	Device server action	Reference
All Registrants	Zero	Preempt the persistent reservation and remove registrations.	5.12.11.2.4.3
	Not Zero	Remove registrations.	5.12.11.2.5
All other types	Zero	Terminate the command with CHECK CONDITION status, with the sense key set to ILLEGAL REQUEST, and the additional sense code set to INVALID FIELD IN PARAMETER LIST.	
	Reservation holder's reservation key	Preempt the persistent reservation and remove registrations.	5.12.11.2.4.3
	Any other, non-zero reservation key	Remove registrations.	5.12.11.2.5

See figure 9 for a description of how a device server interprets a PREEMPT service action to determine its actions (e.g., preempt the persistent reservation, remove registration, or both preempt the persistent reservation and remove registration).

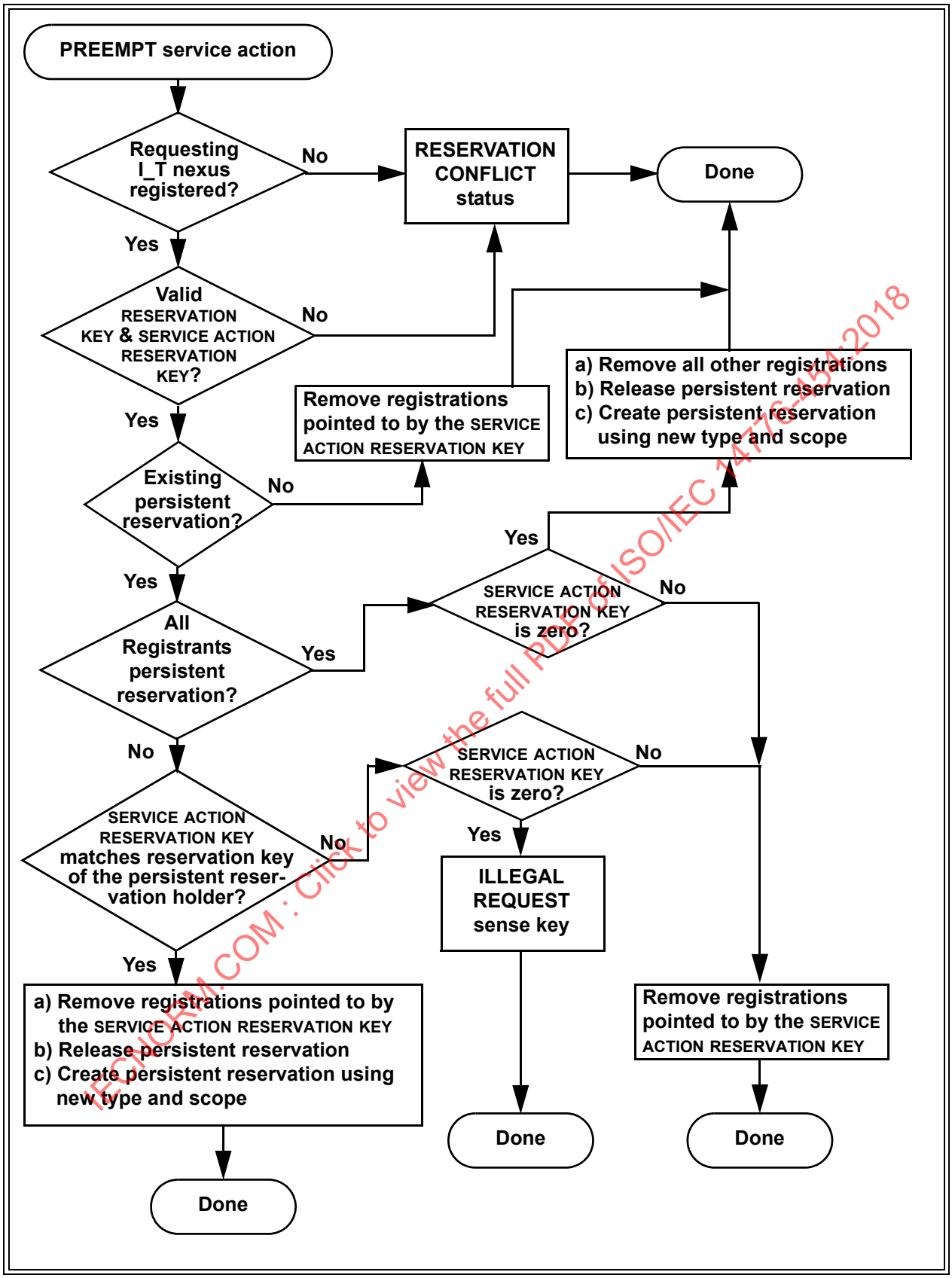


Figure 9 — Device server interpretation of PREEMPT service action

5.12.11.2.4.2 Failed persistent reservation preempt

If the preempting I_T nexus' PREEMPT service action or PREEMPT AND ABORT service action fails (e.g., repeated TASK SET FULL status, repeated BUSY status, SCSI transport protocol time-out, or time-out due to the task set being blocked due to failed initiator port or failed SCSI initiator device), then the application client may send a LOGICAL UNIT RESET task management function to the failing logical unit to remove blocking commands and then resend the preempting service action.

5.12.11.2.4.3 Preempting persistent reservations and registration handling

An application client may preempt the persistent reservation with another persistent reservation by issuing a PERSISTENT RESERVE OUT command with PREEMPT service action or PREEMPT AND ABORT service action through a registered I_T nexus with the following parameters:

a) RESERVATION KEY field set to the value of the reservation key that is registered with the logical unit for the I_T nexus;
b) SERVICE ACTION RESERVATION KEY field set to the value of the reservation key of the persistent reservation to be preempted; and
c) TYPE field and SCOPE field set to define a new persistent reservation. The SCOPE and TYPE of the persistent reservation created by the preempting I_T nexus may be different than those of the persistent reservation being preempted.

If the SERVICE ACTION RESERVATION KEY field identifies a persistent reservation holder (see 5.12.10), the device server shall perform a preempt by doing the following as an uninterrupted series of actions:

a) release the persistent reservation for the holder identified by the SERVICE ACTION RESERVATION KEY field;
b) remove the registrations for all I_T nexuses identified by the SERVICE ACTION RESERVATION KEY field, except the I_T nexus that is being used for the PERSISTENT RESERVE OUT command. If an all registrants persistent reservation is present and the SERVICE ACTION RESERVATION KEY field is set to zero, then all registrations shall be removed except for that of the I_T nexus that is being used for the PERSISTENT RESERVE OUT command;
c) establish a persistent reservation for the preempting I_T nexus using the contents of the SCOPE and TYPE fields;
d) process commands as defined in 5.12.1;
e) establish a unit attention condition for the initiator port associated with every I_T nexus that lost its persistent reservation and/or registration, with the additional sense code set to REGISTRATIONS PREEMPTED; and
f) if the type or scope has changed, then for every I_T nexus whose reservation key was not removed, except for the I_T nexus on which the PERSISTENT RESERVE OUT command was received, the device server shall establish a unit attention condition for the initiator port associated with that I_T nexus, with the additional sense code set to RESERVATIONS RELEASED. If the type or scope have not changed, then no unit attention condition(s) shall be established for this reason.

After the PERSISTENT RESERVE OUT command has been completed with GOOD status, new commands are subject to the persistent reservation restrictions established by the preempting I_T nexus.

The following commands shall be affected in a vendor specific manner either by the restrictions established by the persistent reservation being preempted or by the restrictions established by the preempting I_T nexus:

a) a command received after the arrival, but before the completion of the PERSISTENT RESERVE OUT command with the PREEMPT service action or the PREEMPT AND ABORT service action; or
b) a command that has been placed into a task set by the task manager (see SAM-4) at the time the PERSISTENT RESERVE OUT command with the PREEMPT service action or the PREEMPT AND ABORT service action is received.

Completion status shall be returned for each command unless it was aborted by a PERSISTENT RESERVE OUT command with the PREEMPT AND ABORT service action and TAS bit set to zero in the Control mode page (see 7.5.8).

If an all registrants persistent reservation is not present, it is not an error for the persistent reservation holder to preempt itself (i.e., a PERSISTENT RESERVE OUT with a PREEMPT service action or a PREEMPT AND ABORT service action with the SERVICE ACTION RESERVATION KEY value equal to the persistent reservation holder's reservation key that is received from the persistent reservation holder). In that case, the device server shall establish the new persistent reservation and maintain the registration.

5.12.11.2.5 Removing registrations

If a registered reservation key does not identify a persistent reservation holder (see 5.12.10), an application client may remove the registration(s) without affecting any persistent reservations by issuing a PERSISTENT RESERVE OUT command with PREEMPT service action through a registered I_T nexus with the following parameters:

a) RESERVATION KEY field set to the value of the reservation key that is registered for the I_T nexus; and
b) SERVICE ACTION RESERVATION KEY field set to match the reservation key of the registration or registrations being removed.

If the SERVICE ACTION RESERVATION KEY field does not identify a persistent reservation holder or there is no persistent reservation holder (i.e., there is no persistent reservation), then the device server shall perform a preempt by doing the following in an uninterrupted series of actions:

a) remove the registrations for all I_T nexuses specified by the SERVICE ACTION RESERVATION KEY field;
b) ignore the contents of the SCOPE and TYPE fields;
c) process commands as defined in 5.12.1; and
d) establish a unit attention condition for the initiator port associated with every I_T nexus that lost its registration other than the I_T nexus on which the PERSISTENT RESERVE OUT command was received, with the additional sense code set to REGISTRATIONS PREEMPTED.

If a PERSISTENT RESERVE OUT with a PREEMPT service action or a PREEMPT AND ABORT service action sets the SERVICE ACTION RESERVATION KEY field to a value that does not match any registered reservation key, then the device server shall complete the command with RESERVATION CONFLICT status.

It is not an error for a PERSISTENT RESERVE OUT with a PREEMPT service action or a PREEMPT AND ABORT service action to set the RESERVATION KEY and the SERVICE ACTION RESERVATION KEY to the same value. However, no unit attention condition is established for the I_T nexus on which the PERSISTENT RESERVE OUT command was received. The registration is removed.

5.12.11.2.6 Preempting and aborting

The application client's request for and the device server's responses to a PERSISTENT RESERVE OUT command with PREEMPT AND ABORT service action are identical to the responses to a PREEMPT service action (see 5.12.11.2.4) except for the additions described in this subclause. If no reservation conflict occurred, the device server shall perform the following uninterrupted series of actions:

a) if the persistent reservation is not an all registrants type then:
   A) if the TST field is set to 000b (see 7.5.8) and the faulted I_T nexus, if any, is not the I_T nexus associated with the persistent reservation or registration being preempted, then the task set ACA condition shall be processed as defined in SAM-5;
   B) if the TST field is set to 000b and the faulted I_T nexus, if any, is the I_T nexus associated with the persistent reservation or registration being preempted, then the PERSISTENT RESERVE OUT command shall be processed without regard for the task set ACA condition; or
   C) if the TST field is set to 001b, then the ACA condition shall be processed as defined in SAM-5;

b) perform the uninterrupted series of actions described for the PREEMPT service action (see 5.12.11.2.4);
c) all commands from the I_T nexus(es) associated with the persistent reservations or registrations being preempted (i.e., preempted commands) except the PERSISTENT RESERVE OUT command itself shall be aborted as defined in SAM-5;
d) for each copy operation (see 5.16.4.3) being processed:
   A) the third-party copy command (see 5.16.3), if any, that is associated with that copy operation shall be aborted as described in c); and
   B) that copy operation shall be processed as a background copy operation and aborted as if a COPY OPERATION ABORT command (see 6.3) has been received;
e) after the PERSISTENT RESERVE OUT command with PREEMPT AND ABORT service action has completed, all new commands are subject to the persistent reservation restrictions established by the preempting I_T nexus;
f) if the persistent reservation is not an all registrants type, then the device server shall clear any ACA condition associated with an I_T nexus being preempted and shall abort any commands with an ACA attribute received on that I_T nexus;
g) if the persistent reservation is an all registrants type, then:
   A) if the service action reservation key is set to zero, the device server shall clear any ACA condition and shall abort any commands with an ACA attribute; or
   B) if the service action reservation key is not set to zero, the device server shall do the following for any I_T nexus registered using the specified reservation key:
      a) clear any ACA condition; and
      b) abort any commands with an ACA attribute;

   and
h) for logical units that implement the PREVENT ALLOW MEDIUM REMOVAL command (see SBC-3, SSC-4, and SMC-3), the device server shall perform an action equivalent to the processing of a PREVENT ALLOW MEDIUM REMOVAL command with the PREVENT field equal to zero received on the I_T nexuses associated with the persistent reservation being preempted.

The actions described in this subclause shall be performed for all I_T nexuses that are registered with the non-zero SERVICE ACTION RESERVATION KEY value, without regard for whether the preempted I_T nexuses hold the persistent reservation. If the SERVICE ACTION RESERVATION KEY field is set to zero and an all registrants persistent reservation is present, the device server shall abort all commands for all registered I_T nexuses.

5.12.11.2.7 Clearing

Any application client may release the persistent reservation and remove all registrations from a device server by issuing a PERSISTENT RESERVE OUT command with CLEAR service action through a registered I_T nexus with the following parameter:

a) RESERVATION KEY field set to the value of the reservation key that is registered with the logical unit for the I_T nexus.

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 the persistent reservation, if any;
b) remove all registration(s);
c) ignore the contents of the SCOPE and TYPE fields;
d) for logical units that implement the PREVENT ALLOW MEDIUM REMOVAL command (see SBC-3, SSC-4, and SMC-3), the device server shall perform an action equivalent to the processing of a PREVENT ALLOW MEDIUM REMOVAL command with the PREVENT field equal to zero received on the I_T nexuses associated with the persistent reservation being cleared;
e) continue normal processing of any commands from any I_T nexus that have been accepted by the device server as allowed (i.e., nonconflicting); and

f) establish a unit attention condition for the initiator port associated with every registered I_T nexus other than the I_T nexus on which the PERSISTENT RESERVE OUT command with CLEAR service action was received, with the additional sense code set to RESERVATIONS PREEMPTED.

NOTE 14 - Application clients should not use the CLEAR service action except during recovery operations that are associated with a specific initiator port, since the effect of the CLEAR service action defeats the persistent reservations features that protect data integrity.

5.12.11.3 Replacing lost reservations

A PERSISTENT RESERVE OUT command with the REPLACE LOST RESERVATION service action is used to:

a) begin a recovery process for the lost persistent reservation that is managed by application clients; and
b) cause the device server to stop terminating commands due to a lost persistent reservation (see 5.12.5.4).

If the device server has not detected that persistent reservation information has been lost (see 5.12.5.4), then the device server shall terminate a PERSISTENT RESERVE OUT command with the REPLACE LOST RESERVATION service action with CHECK CONDITION status, with the sense key set to ILLEGAL REQUEST, and the additional sense code set to INVALID RELEASE OF PERSISTENT RESERVATION.

An application client may replace lost reservation information by issuing a PERSISTENT RESERVE OUT command with the REPLACE LOST RESERVATION service action with the following parameters:

a) RESERVATION KEY field set to zero;
b) SERVICE ACTION RESERVATION KEY field set to the value of the new reservation key (i.e., the value used to replace the lost reservation key value); and
c) TYPE field and SCOPE field set to define a new persistent reservation. The scope and type of the new persistent reservation may be different than those of the lost persistent reservation.

To process a valid PERSISTENT RESERVE OUT command with the REPLACE LOST RESERVATION service action the device server shall perform the following as an uninterrupted series of actions:

a) remove the prior registrations for all I_T nexuses, if any, without establishing unit attention conditions;
b) establish a registration for the I_T nexus that is being used for the PERSISTENT RESERVE OUT command using the service action reservation key;
c) release any persistent reservations known to the device server;
d) establish a new persistent reservation for the I_T nexus that is being used for the PERSISTENT RESERVE OUT command using the contents of the SCOPE and TYPE fields;
e) set the PRgeneration value to zero; and
f) stop terminating commands due to a lost persistent reservation (see 5.12.5.4).

After the PERSISTENT RESERVE OUT command with the REPLACE LOST RESERVATION service action has been completed with GOOD status:

a) new commands are subject to the new persistent reservation restrictions established by the command; and
b) until the APTPL bit is set to one in a PERSISTENT RESERVE OUT command with a REGISTER service action or a REGISTER AND IGNORE EXISTING KEY service action that completes with GOOD status, the persist through power loss capability is not enabled (see 5.12.5.2).

5.13 Security features

5.13.1 Security goals and threat model

5.13.1.1 Introduction

In many cases, the security goals and threat model used for the Internet are applicable to SCSI commands. The Internet security goals and threat model found in RFC 3552 as they apply to SCSI are summarized in 5.13.1. Terms, concepts, and classes of security techniques that are defined in RFC 3552 are discussed based on their RFC 3552 definitions without modification in this standard.

The security goals and threat model described in 5.13.1 are valid for all SCSI device types. Command standards may modify this model to handle threats appropriate to specific device types.

5.13.1.2 Security goals

The overall goals of security may be divided into the following categories:

a) communications security (i.e., protecting communications); and
b) system security (e.g., protecting systems from unauthorized usage, inappropriate usage, and denial of service).

These goals interact as a result of communications being carried out by systems, with access to those systems provided through communications channels. A common methodology is to secure the communications first and then provide secure access to systems over the secured communication channels.

Communication security is subdivided into the following primary areas of protection:

a) **confidentiality:** preventing unintended entities from seeing the data;
b) **cryptographic data integrity:** ensuring that the data that arrives is identical to the data that was sent; and
c) **peer entity authentication:** ensuring that the communicating endpoints are the intended peer entities.

Data origin authentication (i.e., ensuring that the received data was sent by the authenticated peer) is the combination of peer entity authentication and cryptographic data integrity.

Non Repudiation enhances data origin authentication with the ability to prove to a third party that the sender sent the data that the receiver received.

Cryptographic data integrity is called data integrity in RFC 3552. The term cryptographic is added in this standard to distinguish the class of integrity protection required to counter malicious attacks from the class of integrity protection required to deal with random data corruption (e.g., caused by cosmic rays or electrical noise). Mechanisms used to deal with random data corruption (e.g., parity bits and CRCs) have minimal value against malicious attacks that are able to modify integrity checks to conceal their modifications to the data. Cryptographic data integrity requires knowledge of a secret key in order to modify an integrity check without that modification being detectable. Systems should provide a high level of assurance that an attacker is unable to learn, guess, discover, or otherwise obtain the required secret key.

In addition to the primary areas, there is another area of control:

a) **authorization:** controlling what an entity is allowed to do. For communications security this is control of the entities with which an entity is allowed to communicate.

A form of authorization is access control (i.e., controlling what an entity is allowed to access).

5.13.1.3 Threat model

Most secured systems are vulnerable to an attacker equipped with sufficient resources, time, and skills. In order to make designing a security system practical, a threat model is defined to describe the capabilities that an attacker is assumed to be able to deploy (e.g., knowledge, computing capability, and ability to control the system).

The main purposes of a threat model are as follows:

a) to identify the threats of concern; and
b) to rule some threats explicitly out of scope.

Most security measures do not provide absolute assurance that an attack has not occurred. Rather, security measures raise the difficulty of accomplishing the attack to beyond the attacker's assumed capabilities and/or resources. Design of security measures that resist attackers with essentially unlimited capabilities (e.g., certain nation-states) is outside the scope of this standard. Security measures that are susceptible to a level of capability available to some attackers may still be useful for deterring attackers who lack that level of capability, especially when combined with non-technical security measures such as physical access controls.

The computational capability of an attacker is treated as a variable because that capability is inherently a moving target as a result of more powerful processors. The computational capability of an attacker influences design aspects (e.g., key length). Well designed security systems are agile in that they are able to operate not only with different key lengths, but also with different cryptographic algorithms.

The Internet threat model described in RFC 3552 is generally applicable to SCSI, and is specifically applicable if Internet Protocols are used by the SCSI transport (e.g., iSCSI, Fibre Channel via FCIP, or Fibre Channel via iFCP). The basic assumptions of the Internet threat model are:

a) end systems engaging in communication are not under the control of the attacker; and
b) the attacker is able to read any communicated data (e.g., data in an IU) and undetectably remove, change, or inject forged IUs, including injection of IUs that appear to be from a known and/or trusted system.

Communications security designs are based on an additional assumption that secrets (e.g., keys) used to secure the communications are protected so that an attacker is unable to learn, guess, discover, or otherwise obtain them. A consequence of this assumption is that attacks against secured communications are assumed to begin without with advance knowledge of the secrets used to secure the communications.

5.13.1.4 Types of attacks

The following types of attacks are considered:

a) passive attacks (i.e., attacks that only require reading IUs); and
b) active attacks (i.e., attacks that require the attacker to change communication and/or engage in communication).

More information on attack types is available in RFC 3552.

Simple passive attacks involve reading communicated data that the attacker was not intended to see (e.g., password, credit card number). More complex passive attacks involve post-processing the communicated data (e.g., checking a challenge-response pair against a dictionary to see if a common word was used as a password).

There are a wide variety of active attacks (e.g., spoofing, replay, insertion, deletion, known plaintext, and modification of communications). Man-in-the-middle attacks are a class of active attacks that involve the attacker inserting itself in the middle of communication, enabling it to intercept all communications without the

knowledge of the communicating parties for various purposes (e.g., insertion, deletion, replay, modification and/or inspection via decryption of the communications).

5.13.1.5 SCSI security considerations

The application of communication security techniques (see RFC 3552) is defined by command standards. This subclause describes specific design considerations in applying the threat model (see 5.13.1.3) to all SCSI device types.

SCSI environments tend not to be fully connected (i.e., there are restrictions on the SCSI device servers with which a SCSI application client is able to communicate) due to the following mechanisms:

a) physical and logical connectivity restrictions (e.g., in SCSI to SCSI gateways across different transports);
b) LUN mapping and masking; and
c) transport zoning.

The resulting connectivity is more limited than the Internet security assumption that an off-path attacker is able to transmit to an arbitrary victim (see RFC 3552).

SCSI security designs are also influenced by SCSI being a client-server distributed service model (see SAM-5) that is realized over a number of different SCSI transport protocols and interconnects.

Security functionality may be defined as part of a command set or at the SCSI transport level. Some SCSI transport protocols (e.g., Fibre Channel and iSCSI) define security functionality that provides confidentiality, cryptographic integrity, and peer entity authentication for all communicated data. However, there are situations in which some or all of those mechanisms are not used and there are SCSI communications whose scope spans more than one SCSI transport protocol (e.g., via a gateway between iSCSI and FCP). Security that is defined by a command set is appropriate for such situations.

5.13.2 Security associations

5.13.2.1 Principles of SAs

Before an application client and device server begin applying security functions (e.g., data integrity checking, data encryption) to messages (i.e., data that is transferred in either direction between them), they perform a security protocol to create at least one SA (see 5.13.2.3). The result of the SA creation protocol is two sets of SA parameters (see 5.13.2.2), one that is maintained by the application client and one that is maintained by the device server.

In this model, SAs decouple the process of creating a security relationship from its usage in processing security functions. This decoupling allows either the creation or the usage of an SA to be upgraded in response to changing security threats without requiring both processes to be upgraded simultaneously.

Figure 10 shows the relationship between application clients and device servers with respect to SAs.

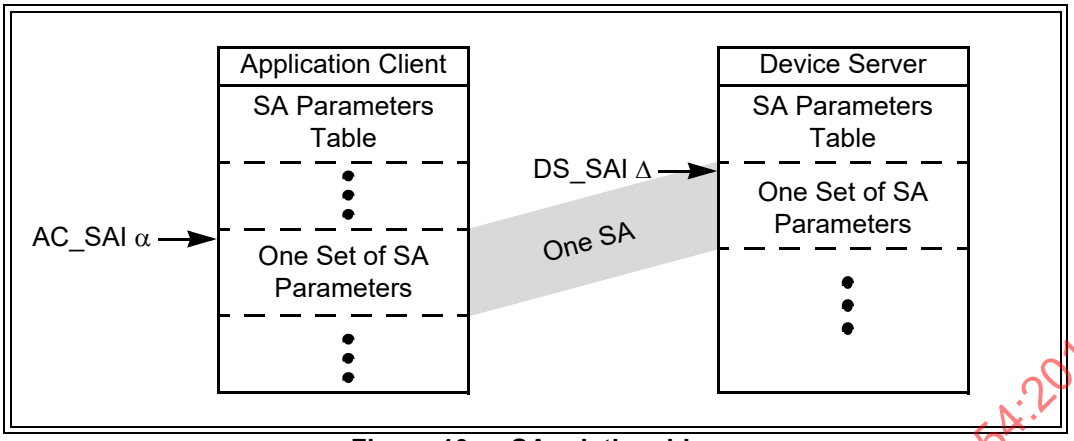


Figure 10 — SA relationships

In both the application client and the device server, the SA parameters are modeled as being stored in an indexed array and the SAI (i.e., the AC_SAI or the DS_SAI in figure 10) identifies one set of SA parameters within that array. The application client and device server are not required to store the parameters for any given SA in the same array locations. In order to support this implementation flexibility, a single SA is modeled as having two different SAI values (i.e., one for the application client and one for the device server).

The device server shall maintain a single SA parameters table for all I_T nexuses.

SAs shall not be preserved across a power cycle, hard reset, or logical unit reset. SAs shall not be affected by an I_T nexus loss.

5.13.2.2 SA parameters

Each SAI shall identify at least the SA parameters defined in table 74. Individual security protocols define how the SA parameters are generated and/or used by that security protocol.

Table 74 — Minimum SA parameters (part 1 of 2)

Name	Description	Size (bytes) [a]		Scope [b]
		Min.	Max.	
SA parameters that identify and manage the SA.				
AC_SAI	The SAI used by the application client to identify the SA. [c]	4	4	Public
DS_SAI	The SAI used by the device server to identify the SA. [c]	4	4	Public
TIMEOUT	The number of seconds that may elapse after the completion of an SA access operation (i.e., SA creation or SA usage by a command) before the device server should discard the state associated with this SA (e.g., the SA parameters). If SA state is discarded because no SA access operations are received during the specified interval, then the device server shall respond to further attempts to access the SA as if the SA had never been created. This parameter shall not be set to zero.	4	4	Public
SA parameters that are incorporated in messages to prevent message replay attacks.				
AC_SQN	A sequence number that is incremented for each response message received by an application client on which a security function is performed and used to detect replay attacks (see 5.13.1.4).	8	8	Public
DS_SQN	A sequence number that is incremented for each request message received by a device server on which a security function is performed and used to detect replay attacks.	8	8	Public

[a] These size values are guidelines. Specific security protocols may place more exacting size requirements on SA parameters.

[b] Public SA parameters may be transferred outside a SCSI device unencrypted. Secret SA parameters shall not be transferred outside a SCSI device. Fields within a protocol specific SA parameter are Shared or Secret as defined by the applicable SA creation protocol.

[c] SAI values between 0 and 255, inclusive, are reserved.

[d] Nonce SA parameters shall be at least half the size of the KEY_SEED SA parameter.

[e] The number of bits of entropy in the KEY_SEED SA parameter should be as close to the number of bits in the KEY_SEED SA parameter as possible (see RFC 3766).

Table 74 — Minimum SA parameters (part 2 of 2)

Name	Description	Size (bytes) [a]		Scope [b]
		Min.	Max.	
SA parameters that are used by security functions to derive the secret keys that are applied to messages (e.g., for encryption).				
AC_NONCE [d]	A random nonce value that is generated by the application client and used as an input to the key derivation security algorithm specified by the KDF_ID SA parameter during the derivation of an encryption key.	16	64	Public
DS_NONCE [d]	A random nonce value that is generated by the device server and used as an input to the key derivation security algorithm specified by the KDF_ID SA parameter during the derivation of an encryption key.	16	64	Public
KEY_SEED [e]	A value that is known only to the application client and device server that are creating this SA that in combination with the applicable nonce is used to derive the KEYMAT SA parameter. The KEY_SEED SA parameter shall be set to zero as part of completing the SA creation function.	16	64	Secret
KDF_ID	A security algorithm (see 5.13.8) coded value that identifies the KDF used by the application client and device server.	4	4	Public
SA parameters that are used by security functions to secure messages between the application client and device server.				
KEYMAT	A value that is known only to the application client and device server that are participating in this SA that may be subdivided into one or more key values that are used in security functions that secure messages. The contents of KEYMAT depend on the USAGE_TYPE SA parameter value.	14	1 024	Secret
SA parameters that are used by SA management functions.				
USAGE_TYPE	A coded value (see table 75) that indicates how the SA is used.	2	2	Public
USAGE_DATA	Information associated with how the SA is used (e.g., cryptographic algorithms and key sizes). The contents of USAGE_DATA depend on the USAGE_TYPE SA parameter value.	0	1 024	Public
MGMT_DATA	SA data that is used in ways defined by the SA creation protocol to perform SA management functions (e.g., deletion of the SA).	0	1 024	Protocol specific

[a] These size values are guidelines. Specific security protocols may place more exacting size requirements on SA parameters.

[b] Public SA parameters may be transferred outside a SCSI device unencrypted. Secret SA parameters shall not be transferred outside a SCSI device. Fields within a protocol specific SA parameter are Shared or Secret as defined by the applicable SA creation protocol.

[c] SAI values between 0 and 255, inclusive, are reserved.

[d] Nonce SA parameters shall be at least half the size of the KEY_SEED SA parameter.

[e] The number of bits of entropy in the KEY_SEED SA parameter should be as close to the number of bits in the KEY_SEED SA parameter as possible (see RFC 3766).

The USAGE_TYPE SA parameter (see table 75) provides an indication of how the SA is to be used.

Table 75 — USAGE_TYPE SA parameter

Code [a]	Description	USAGE_TYPE SA parameter		
		Usage model	Description	Reference
0000h to 0080h	Reserved			
0081h	Tape data encryption	ESP-SCSI [b]	None [c]	SSC-4
0082h to 8000h	Reserved			
8001h	CbCS authentication and credential encryption	ESP-SCSI [b]	None [c]	5.13.6.7
8002h to FFFFh	Reserved			

[a] USAGE_TYPE values between 8000h and CFFFh inclusive place additional constraints on how an SA is to be created as described in 7.7.3.5.14.

[b] ESP-SCSI usage is defined in 5.13.7.

[c] The USAGE DATA LENGTH field in the IKEv2-SCSI SAUT Cryptographic Algorithms payload (see 7.7.3.5.14) shall contain zero.

5.13.2.3 Creating an SA

The SECURITY PROTOCOL IN command (see 6.40) and SECURITY PROTOCOL OUT command (see 6.41) security protocols shown in table 76 are used to create SAs. The process of creating an SA establishes the SA parameter (see 5.13.2.2) values as follows:

a) initial values for:
   A) AC_SQN set as described in 5.13.4.9; and
   B) DS_SQN set as described in 5.13.4.9;
b) unchanging values for the lifetime of the SA:
   A) AC_SAI;
   B) DS_SAI;
   C) TIMEOUT;
   D) KDF_ID;
   E) KEYMAT;
   F) USAGE_TYPE;
   G) USAGE_DATA; and
   H) MGMT_DATA;
   and
c) values that are zero (see 5.13.4.9) upon completion of SA creation:
   A) KEY_SEED;
   B) AC_NONCE; and
   C) DS_NONCE.

Table 76 — Security protocols that create SAs

Security Protocol Code	Description	References
40h	SA creation capabilities	7.7.2
41h	IKEv2-SCSI	5.13.4 and 7.7.3

5.13.3 Key derivation functions

5.13.3.1 KDFs overview

A KDF produces KEYMAT from in input KEY_SEED and STRING as follows:

KEYMAT = KDF(KEY_SEED, STRING)

Where:
KEY_SEED is a SA Parameter (see 5.13.2.2); and
STRING is a specified sequence of bytes.

Table 77 summarizes the KDFs defined by this standard.

Table 77 — KDFs summary

Security Algorithm Code (see table 105)	Description	Reference
8002 0002h	IKEv2-based iterative HMAC KDF based on SHA-1	5.13.3.3
8002 0005h	IKEv2-based iterative HMAC KDF based on SHA-256	5.13.3.3
8002 0006h	IKEv2-based iterative HMAC KDF based on SHA-384	5.13.3.3
8002 0007h	IKEv2-based iterative HMAC KDF based on SHA-512	5.13.3.3
8002 0004h	IKEv2-based iterative KDF based on AES-128 in XCBC mode	5.13.3.4

5.13.3.2 IKEv2-based iterative KDF

To produce a sufficient number of bits in KEYMAT, the IKEv2-based (see RFC 4306) iterative uses the following equation:

$KEYMAT = T_1 \| T_2 \| T_3 \| T_4 \| \ldots \| T_N$

Where:
T_1 = KDF(KEY_SEED, STRING || 01h);
T_2 = KDF(KEY_SEED, T_1 || STRING || 02h);
T_3 = KDF(KEY_SEED, T_2 || STRING || 03h);
T_4 = KDF(KEY_SEED, T_3 || STRING || 04h);
… = …
T_N = KDF(KEY_SEED, T_{N-1} || STRING || Nh);
KDF is the function defined in 5.13.3.1;
KEY_SEED is a SA Parameter (see 5.13.2.2); and
STRING is a specified sequence of bytes.

Protocols that use the IKEv2-based iterative KDF to generate KEYMAT should ensure that the number of KEYMAT bits requested does not cause N to exceed 255. If N reaches 256, then:

1) the requested number of KEYMAT bits is not returned; and
2) the request to produce KEYMAT shall be terminated with an error.

5.13.3.3 HMAC-based KDFs

If the KDF_ID is one of those shown in table 78, the KDF is a combination of the:

a) HMAC function defined in FIPS 198-1;
b) secure hash function shown in table 78 for the specified KDF_ID value; and
c) IKEv2-based iterative KDF technique (see 5.13.3.2).

The technique requires the following inputs from or related to the SA parameters:

a) AC_SAI;
b) DS_SAI;
c) AC_NONCE;
d) DS_NONCE;
e) KEY_SEED; and
f) KEYMAT size, in bits.

The USAGE_TYPE SA parameter and USAGE_DATA SA parameter (see 5.13.2.2) specify the KEYMAT size as part of the security protocol that performs SA creation (see 5.13.2.3).

The IKEv2-based iterative KDF technique (see 5.13.3.2) is applied with the following inputs:

a) IFUNC (see 5.13.3.2) is the HMAC function defined in FIPS 198-1 with the translation of inputs names shown in table 78;
b) KEY_SEED is the KEY_SEED SA parameter; and
c) STRING contains the concatenated contents of the following SA parameters:
   1) AC_NONCE;
   2) DS_NONCE;
   3) AC_SAI; and
   4) DS_SAI.

Table 78 — HMAC-based KDFs

FIPS 198-1 inputs selected by KDF_ID	KDF_ID (see table 77)			
	8002 0002h	**8002 0005h**	**8002 0006h**	**8002 0007h**
H (i.e., hash function)	SHA-1 (see table 79)	SHA-256 (see table 79)	SHA-384 (see table 79)	SHA-512 (see table 79)
B (i.e., hash input block size) [a]	64	64	128	128
L (i.e., hash output block size) [a, b]	20	32	48	64
K (i.e., key)	KEY_SEED SA parameter			
text	STRING as defined in this subclause and used in 5.13.3.2			

[a] In accordance with FIPS 198-1, all sizes are shown in bytes.
[b] The HMAC-based KDFs defined by this standard do not truncate (i.e., FIPS 198-1 Lambda equals the L shown in this table).

Details of the hash functions that act as inputs to the FIPS 198-1 HMAC function are shown in table 79.

Table 79 — Hash functions used by HMAC based on KDF_ID

KDF_ID (see table 77)	Function	Description
8002 0002h	SHA-1	HMAC input *H* is the SHA-1 secure hash function defined in FIPS 180-4.
8002 0005h	SHA-256	HMAC input *H* is the SHA-256 secure hash function defined in FIPS 180-4.
8002 0006h	SHA-384	HMAC input *H* is the SHA-384 secure hash function defined in FIPS 180-4.
8002 0007h	SHA-512	HMAC input *H* is the SHA-512 secure hash function defined in FIPS 180-4.

5.13.3.4 AES-XCBC-PRF-128 IKEv2-based iterative KDF

If the KDF_ID is 8002 0004h, the KDF is a combination of:

a) the AES-XCBC-PRF-128 secure hash function defined in RFC 4434 and RFC 3566; and
b) the IKEv2-based iterative KDF technique (see 5.13.3.2).

The technique requires the following inputs from or related to the SA parameters:

a) AC_SAI;
b) DS_SAI;
c) AC_NONCE;
d) DS_NONCE;
e) KEY_SEED; and
f) the number of KEYMAT bits that are to be produced.

The IKEv2-based iterative KDF (see 5.13.3.2) is applied with the following inputs:

a) IFUNC (see 5.13.3.2) is the AES-XCBC-PRF-128 secure hash function with the translation of inputs names shown in table 80;
b) KEY_SEED is the KEY_SEED SA parameter; and
c) STRING contains the concatenated contents of the following SA parameters:
   1) AC_NONCE;
   2) DS_NONCE;
   3) AC_SAI; and
   4) DS_SAI.

Table 80 — RFC 3566 parameter translations for the KDF based on AES-XCBC-PRF-128

RFC 3566 Parameter	Translation
K (i.e., key)	KEY_SEED SA parameter
M (i.e., message)	STRING as defined in this subclause and used in 5.13.3.2

5.13.4 Using IKEv2-SCSI to create an SA

5.13.4.1 Overview

The IKEv2-SCSI protocol is a subset of the IKEv2 protocol (see RFC 4306) that this standard defines for use in the creation and maintenance of an SA.

An IKEv2-SCSI SA creation transaction shall only be initiated by the application client.

The IKEv2-SCSI protocol creates the following pair of IKE SAs (see RFC 4306):

a) an SA that protects data transferred from the application client to the device server; and
b) an SA that protects data transferred from the device server to the application client.

An IKEv2-SCSI SA creation transaction consists of the following steps:

1) **Device Server Capabilities step** (see 5.13.4.5): The application client determines the device server's cryptographic capabilities;
2) **Key Exchange step** (see 5.13.4.6): The application client and device server:
   A) perform a key exchange;
   B) determine SAIs;
   C) generate the shared keys used for SA management (e.g., SA creation and deletion) (see 5.13.4.8); and
   D) may complete the generation of the SA (see 5.13.4.9);
   and
3) **Authentication step** (see 5.13.4.7): Unless omitted by application client and device server negotiations in the previous steps, the application client and device server:
   A) authenticate:
      a) each other;
      b) the key exchange; and
      c) the capability selection;
      and
   B) complete the generation of the SA (see 5.13.4.9).

The values in the SECURITY PROTOCOL field and the SECURITY PROTOCOL SPECIFIC field in the SECURITY PROTOCOL IN command (see 6.40) and SECURITY PROTOCOL OUT command (see 6.41) identify the step of the IKEv2-SCSI protocol (see 7.7.3.2).

The Key Exchange step and the Authentication step depend on the results from the Device Server Capabilities step in order to create an SA. During the Key Exchange step, the application client and device server perform independent computations to construct the following sets of shared keys:

a) shared keys that are used by the Authentication step;
b) shared keys that are used by the Authentication step and to delete the SA; and
c) shared keys that are used by SCSI usage type specific operations that obtain security from the generated SA.

More details about these shared keys are provided in 5.13.4.4.

An application client may or may not:

a) proceed to the Key Exchange step after the Device Server Capabilities step; or
b) perform a separate Device Server Capabilities step for each IKEv2-SCSI SA creation transaction.

If the device server's capabilities have changed since the Device Server Capabilities step, the Authentication step returns an error, and the Key Exchange step may return an error.

Changes in the device server's capabilities do not take effect until at least one application client has been notified of the new capabilities via the parameter data returned by the Device Server Capabilities step.

After a Device Server Capabilities step, the application client performs SA creation by sending a sequence of two or four IKEv2-SCSI commands over a single I_T_L nexus to the device server. The following commands constitute an IKEv2-SCSI CCS:

a) if the Authentication step is skipped (see 5.13.4.3.4):
   1) a Key Exchange step SECURITY PROTOCOL OUT command (see 5.13.4.6.2); and
   2) a Key Exchange step SECURITY PROTOCOL IN command (see 5.13.4.6.3);
   or
b) if the Authentication step is performed:
   1) a Key Exchange step SECURITY PROTOCOL OUT command (see 5.13.4.6.2);
   2) a Key Exchange step SECURITY PROTOCOL IN command (see 5.13.4.6.3);
   3) an Authentication step SECURITY PROTOCOL OUT command (see 5.13.4.7.2); and
   4) an Authentication step SECURITY PROTOCOL IN command (see 5.13.4.7.3).

The device server shall process each command in the IKEv2-SCSI CCS to completion before returning status. While a command in the IKEv2-SCSI CCS is being processed by the device server, the application client may use the REQUEST SENSE command (see 5.13.5) to ascertain the device server's progress for the command.

If an error is encountered, the device server or application client may abandon the IKEv2-SCSI CCS before the SA is created (see 5.13.4.10).

The device server shall maintain state for the IKEv2-SCSI CCS on a given I_T_L nexus from the time the Key Exchange step SECURITY PROTOCOL OUT command is completed with GOOD status until one of the following occurs:

a) the IKEv2-SCSI CCS completes successfully;
b) the IKEv2-SCSI CCS is abandoned as described in 5.13.4.10;
c) the SA being created by the IKEv2-SCSI CCS is deleted as described in 5.13.4.11;
d) the number of seconds specified in the IKEV2-SCSI PROTOCOL TIMEOUT field of the IKEv2-SCSI Timeout Values payload (see 7.7.3.5.15) in the Key Exchange step SECURITY PROTOCOL OUT parameter data elapses and none of the following commands have been received:
   A) the next command in the IKEv2-SCSI CCS; or
   B) a REQUEST SENSE command;
   or
e) one of the following event related SCSI device conditions (see SAM-5) occurs:
   A) power cycle;
   B) hard reset;
   C) logical unit reset; or
   D) I_T nexus loss.

If the device server receives a SECURITY PROTOCOL OUT command or SECURITY PROTOCOL IN command with the SECURITY PROTOCOL field set to IKEv2-SCSI (i.e., 41h) on an I_T_L nexus other than the one for which IKEv2-SCSI CCS state is being maintained, then:

a) an additional IKEv2-SCSI CCS may be started; or
b) the device server may terminate the command with CHECK CONDITION status, with the sense key set to ABORTED COMMAND, and the additional sense code set to CONFLICTING SA CREATION REQUEST.

Except for the PERSISTENT RESERVE OUT command (see 5.12.4) and the cases described in this subclause, a device server that is maintaining a IKEv2-SCSI CCS state on a particular I_T_L nexus shall not alter its processing of new commands received on that I_T_L nexus.

If all of the following conditions are true:

a) the device server includes the following algorithm descriptors in the IKEv2-SCSI SA Creation Capabilities payload (see 7.7.3.5.12) in the parameter data returned by the Device Server Capabilities step SECURITY PROTOCOL IN command (see 5.13.4.5):
   A) an SA_AUTH_OUT algorithm descriptor (see 7.7.3.6.6) with the ALGORITHM IDENTIFIER field set to SA_AUTH_NONE; and
   B) an SA_AUTH_IN algorithm descriptor (see 7.7.3.6.6) with the ALGORITHM IDENTIFIER field set to SA_AUTH_NONE;

   and
b) the application client sends the following algorithm descriptors to the device server in the IKEv2-SCSI SA Cryptographic Algorithms payload (see 7.7.3.5.13) in the Key Exchange step SECURITY PROTOCOL OUT command (see 5.13.4.6.2):
   A) an SA_AUTH_OUT algorithm descriptor with the ALGORITHM IDENTIFIER field set to SA_AUTH_NONE; and
   B) an SA_AUTH_IN algorithm descriptor with the ALGORITHM IDENTIFIER field set to SA_AUTH_NONE,

then:

a) the Authentication step is skipped;
b) the IKEv2-SCSI CCS consists of the two Key Exchange step commands;
c) the device server requires the IKEv2-SCSI SAUT Cryptographic Algorithms payload (see 7.7.3.5.14) to be present in the parameter data sent by the Key Exchange step SECURITY PROTOCOL OUT command;
d) the device server returns the IKEv2-SCSI SAUT Cryptographic Algorithms payload in the parameter data returned by the Key Exchange step SECURITY PROTOCOL IN command; and
e) SA creation occurs upon the completion of the Key Exchange step.

Operation of an IKEv2-SCSI CCS depends on SA_AUTH_NONE being used in both the Authentication step SECURITY PROTOCOL OUT command and the Authentication step SECUIRTY PROTOCOL IN command, or SA_AUTH_NONE not being used by either command. Processing requirements placed on the SECURITY PROTOCOL OUT command during the Key Exchange step (see 7.7.3.6.6) ensure that this dependency is maintained.

If no other errors are detected and any of the following conditions are true:

a) the device server does not include an SA_AUTH_OUT algorithm descriptor with the ALGORITHM IDENTIFIER field set to SA_AUTH_NONE in the IKEv2-SCSI SA Creation Capabilities payload in the parameter data returned by the Device Server Capabilities step SECURITY PROTOCOL IN command;
b) the device server does not include an SA_AUTH_IN algorithm descriptor with the ALGORITHM IDENTIFIER field set to SA_AUTH_NONE in the IKEv2-SCSI SA Creation Capabilities payload in the parameter data returned by the Device Server Capabilities step SECURITY PROTOCOL IN command; or
c) the application client does not set the ALGORITHM IDENTIFIER field to SA_AUTH_NONE in the SA_AUTH_OUT algorithm descriptor and the SA_AUTH_IN algorithm descriptor in the IKEv2-SCSI SA Cryptographic Algorithms payload sent to the device server in the Key Exchange step SECURITY PROTOCOL OUT command,

then:

a) the Authentication step is processed;
b) the IKEv2-SCSI CCS consists of the two Key Exchange step commands and two Authentication step commands;
c) the device server requires the IKEv2-SCSI SAUT Cryptographic Algorithms payload to be absent from the parameter data sent by the Key Exchange step SECURITY PROTOCOL OUT command;

d) the device server omits the IKEv2-SCSI SAUT Cryptographic Algorithms payload from the parameter data returned by the Key Exchange step SECURITY PROTOCOL IN command;
e) the device server requires the IKEv2-SCSI SAUT Cryptographic Algorithms payload to be present in the parameter data sent by the Authentication step SECURITY PROTOCOL OUT command;
f) the device server returns the IKEv2-SCSI SAUT Cryptographic Algorithms payload in the parameter data returned by the Authentication step SECURITY PROTOCOL IN command; and
g) SA creation occurs upon the completion of the Authentication step.

SA participants should perform the Authentication step unless man-in-the-middle attacks (see 5.13.1.4) are not of concern or are prevented by a means outside the scope of this standard (e.g., physical security of the transport).

Omission of the Authentication step provides no defense against a man-in-the-middle adversary that is capable of modifying SCSI commands. Such an adversary is able to insert itself as an intermediary on the created SA without knowledge of the SA participants, thereby completely subverting the intended security. Omission of the Authentication step is only appropriate in environments where the absence of such adversaries is assured by other means (e.g., a direct physical connection between the systems on which the application client and device server or use of end-to-end security in the SCSI transport protocol such as FC-SP-2).

5.13.4.2 IKEv2-SCSI Protocol summary

This subclause summarizes the IKE-v2-SCSI payloads (see 7.7.3.5) that are exchanged between an application client and a device server during all steps of an IKEv2-SCSI SA creation transaction using message diagrams. Each IKEv2-SCSI step (see 5.13.4.1) is shown in a separate figure. The contents of a payload (e.g., Key Exchange) may not be the same in both directions of transfer.

Figure 11 shows the Device Server Capabilities step (see 5.13.4.5). The Device Server Capabilities step consists of a SECURITY PROTOCOL IN command carrying an IKEv2-SCSI SA Creation Capabilities payload (see 7.7.3.5.12). The IKEv2-SCSI header is not used.

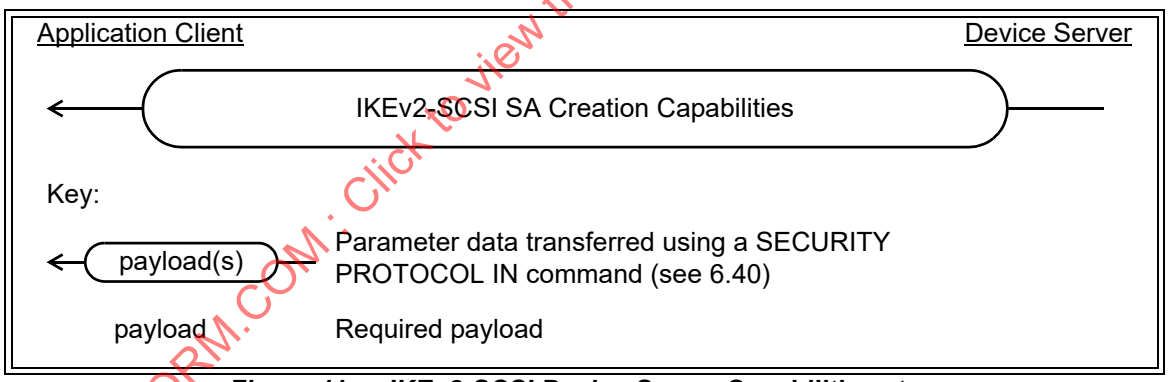


Figure 11 — IKEv2-SCSI Device Server Capabilities step

The IKEv2-SCSI SA Creation Capabilities payload indicates the device server's capabilities for SA creation.

Figure 12 shows the Key Exchange step (see 5.13.4.6). The Key Exchange step consists of a SECURITY PROTOCOL OUT command followed by a SECURITY PROTOCOL IN command.

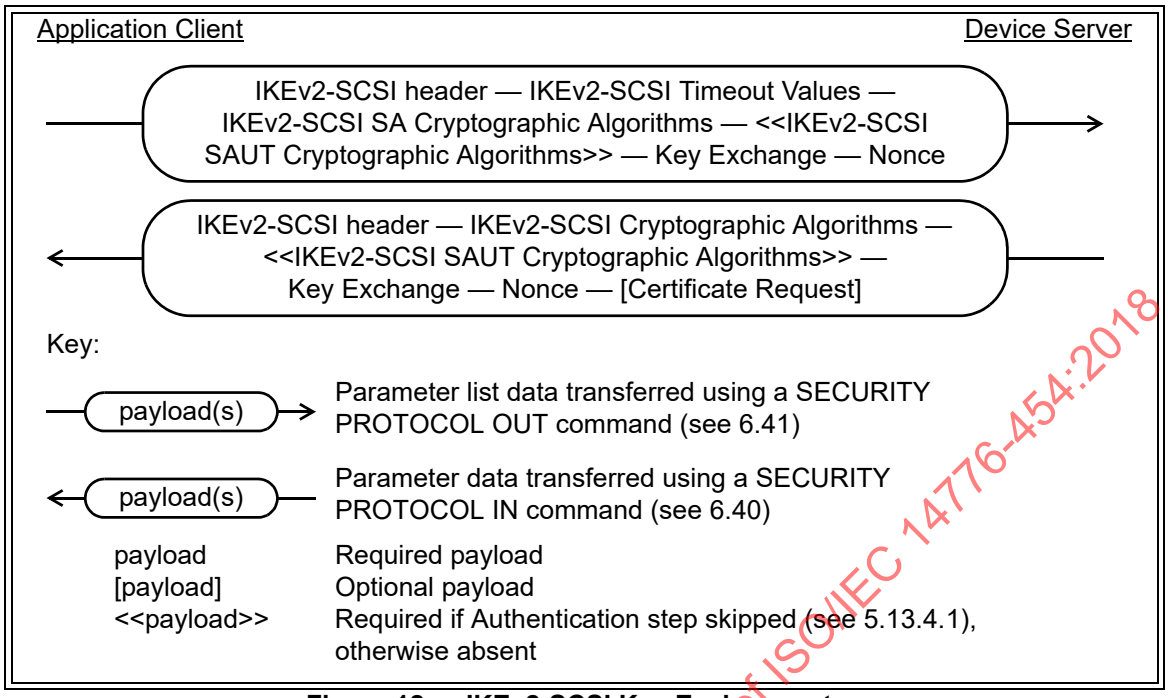


Figure 12 — IKEv2-SCSI Key Exchange step

The IKEv2-SCSI Timeout Values payload (see 7.7.3.5.15) contains timeouts for SA creation and usage.

The IKEv2-SCSI SA Cryptographic Algorithms payloads (see 7.7.3.5.13) are used to select and agree on the cryptographic algorithms used for creating the SA.

If the Authentication step is skipped (see 5.13.4.1), the IKEv2-SCSI SAUT Cryptographic Algorithms payloads (see 7.7.3.5.14) are used to select and agree on usage of the SA and the cryptographic algorithms used by the created SA. If the Authentication step is processed (i.e., not skipped), the SA usage and algorithms selection is performed during the Authentication step.

The Key Exchange payload (see 7.7.3.5.3) and Nonce payload (see 7.7.3.5.8) are part of the key and nonce exchanges that are used to generate the IKEv2-SCSI keys and SA keys.

The Certificate Request payload or payloads (see 7.7.3.5.6) enables the device server to request a certificate from the application client. If the Authentication step is being skipped (see 5.13.4.1), the device server shall not include any Certificate Request payloads in the parameter data. Use of the Certificate Request payload is described in 5.13.4.3.3.4.

Figure 13 shows the Authentication step (see 5.13.4.7). The Authentication step consists of a SECURITY PROTOCOL OUT command followed by a SECURITY PROTOCOL IN command.

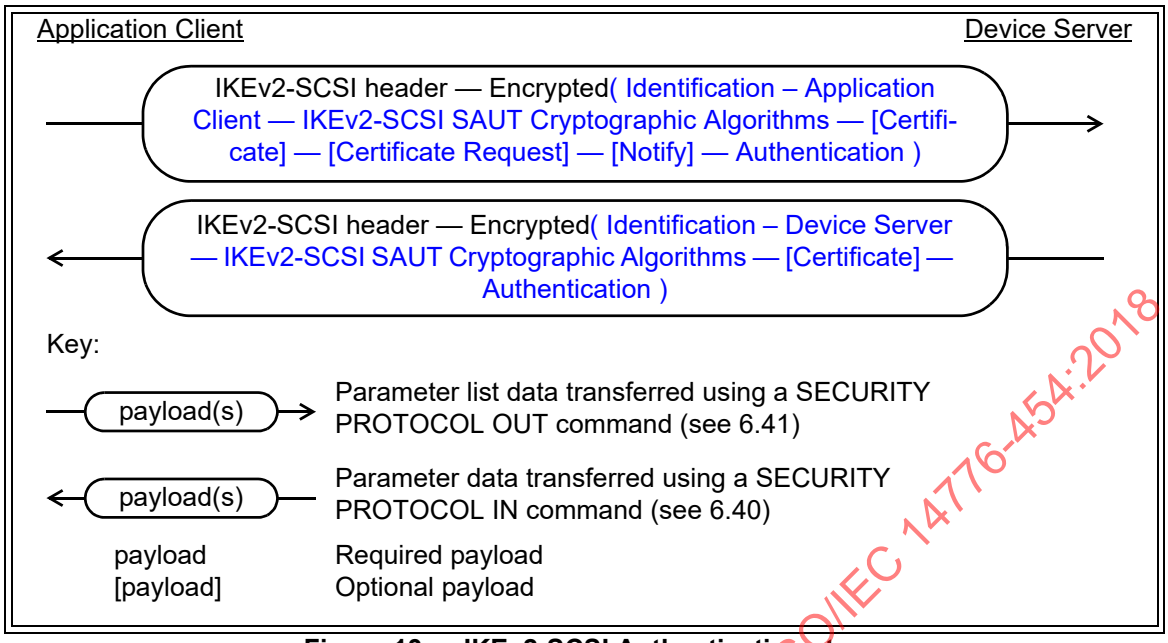


Figure 13 — IKEv2-SCSI Authentication step

An Encrypted payload (see 7.7.3.5.11) contains all other Authentication step payloads that are protected using the cryptographic algorithms determined by the IKEv2-SCSI SA Cryptographic Algorithms payloads (see 7.7.3.5.13) in the Key Exchange step (see figure 12).

The Identification payloads (see 7.7.3.5.4) contain the identities to be authenticated. These identities are not required to be SCSI names or identifiers.

The IKEv2-SCSI SAUT Cryptographic Algorithms payloads (see 7.7.3.5.14) are used to select and agree on usage of the SA and the cryptographic algorithms used by the created SA.

The Certificate payload or payloads (see 7.7.3.5.5) respond to the Certificate Request payload(s) sent by the device server in the Key Exchange step SECURITY PROTOCOL IN command.

The Certificate Request payload or payloads (see 7.7.3.5.6) allows an application client to request the delivery of a Certificate payload (see 7.7.3.5.5) in the parameter data for the Authentication step SECURITY PROTOCOL IN command (see 5.13.4.3.3.4).

The Notify payload (see 7.7.3.5.9) provides a means for the application client to inform the device server that this is the only SA being used between them, and that the device server should discard state for any other SAs created by the same application client.

The Authenticate payloads (see 7.7.3.5.7) authenticate not only the SA participants, but also the entire protocol sequence (e.g., the Authenticate payloads prevent a man-in-the-middle attack from succeeding).

Figure 14 shows the Delete operation (see 5.13.4.11). The Delete operation consists of a SECURITY PROTOCOL OUT command.

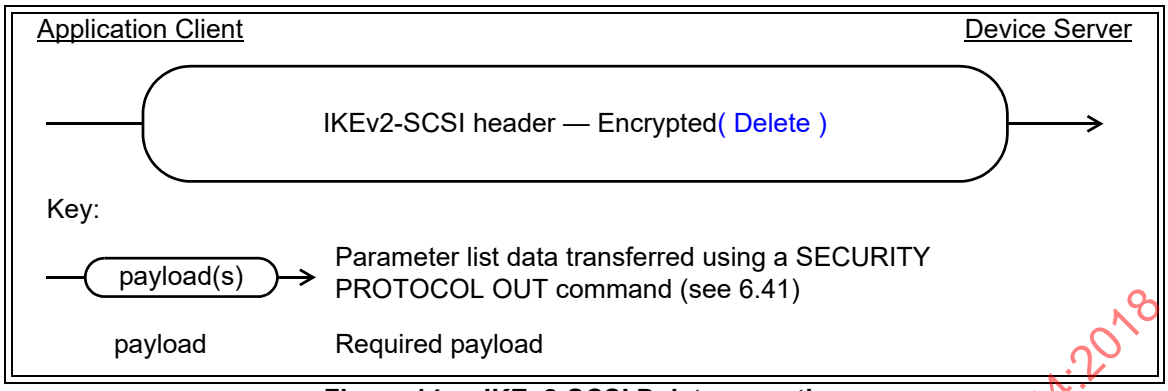


Figure 14 — IKEv2-SCSI Delete operation

An Encrypted payload (see 7.7.3.5.11) contains the Delete payload that is protected using the cryptographic algorithms determined by the IKEv2-SCSI SA Cryptographic Algorithms payloads (see 7.7.3.5.13) in the Key Exchange step that was used to create the SA.

The Delete payload (see 7.7.3.5.10) specifies the SA to be deleted.

5.13.4.3 IKEv2-SCSI Authentication

5.13.4.3.1 Overview

IKEv2-SCSI authentication includes these security functions:

a) the application client and device server each establish an identity by demonstrating knowledge of a secret authentication key associated with that identity;
b) the application client demonstrates knowledge of the current device server capability information; and
c) the application client and device server check the integrity of the current IKEv2-SCSI CCS.

An IKEv2 SCSI authentication algorithm accomplishes these functions by generating and verifying authentication data based on a concatenation of bytes that includes device server capability information and the specified portion of the IKEv2-SCSI parameter data (see 7.7.3.5.7) from the IKEv2-SCSI Key Exchange step (see 5.13.4.6).

An authentication key associated with an identity is used to generate authentication data. IKEv2-SCSI transfers the authentication data in the AUTHENTICATION DATA field of the IKEv2-SCSI Authentication payload (see 7.7.3.5.7). The recipient of an IKEv2-SCSI Authentication payload uses a verification key associated with the identity to verify the authentication data. The identity is:

a) transferred in the IDENTIFICATION DATA field of the appropriate IKEv2-SCSI Identification payload (see 7.7.3.5.4); or
b) obtained from a certificate transferred in the IKEv2-SCSI Certificate payload (see 7.7.3.5.5).

IKEv2-SCSI Authentication is bidirectional (i.e., both the application client and the device server authenticate). IKEv2-SCSI Authentication is skipped when the application client and device server agree to do so during the Key Exchange step (see 5.13.4.3.4).

The following IKEv2-SCSI Authentication methods are defined:

a) **pre-shared key** (see 5.13.4.3.2): the authentication key is also used as the verification key; and
b) **digital signature** (see 5.13.4.3.3): the verification key and authentication key form a public/private key pair. The authentication data is a digital signature based on asymmetric cryptography.

Certificates and the IKEv2-SCSI Certificate payload may be used to provide verification keys for digital signatures to application clients and device servers.

5.13.4.3.2 Pre-shared key authentication

Pre-shared key authentication uses a single cryptographic algorithm to both generate and verify authentication data. A pre-shared key is associated with an identity that is transferred in the IDENTIFICATION DATA field of the appropriate Identification payload (see 7.7.3.5.4). The pre-shared key serves as both the authentication key and the verification key for the identity.

NOTE 15 - A pre-shared key that is not kept secret may compromise the security properties of IKEv2-SCSI.

If pre-shared key authentication is used, then the pre-shared key is the key for the cryptographic algorithm. Authentication data is generated by applying the cryptographic algorithm with this key to the input data (e.g., the applicable concatenation of bytes described in 7.7.3.5.7).

Verification of the authentication data shall consist of:

1) computing the expected contents of the AUTHENTICATION DATA field of the Authentication payload (see 7.7.3.5.7) using the input data and a verification key associated with the identity received in the Identification payload (see 7.7.3.5.4); and
2) comparing the expected contents to the actual contents of the AUTHENTICATION DATA field.

Verification is successful if the expected contents match the actual contents, otherwise verification is not successful.

The pre-shared key requirements in RFC 4306 shall apply to IKEv2-SCSI pre-shared keys, including the following requirements on interfaces for provisioning pre-shared keys:

a) ASCII strings of at least 64 bytes shall be supported;
b) a null terminator shall not be added to any input before it is used as a pre-shared key;
c) a hexadecimal ASCII encoding of the pre-shared key shall be supported; and
d) ASCII encodings other than hexadecimal may be supported. Support for any such encoding shall include specification of the algorithm for translating the encoding to a binary string as part of the interface.

The following requirements for pre-shared keys apply in addition to those found in RFC 4306:

a) a pre-shared key shall be associated with one identity;
b) the same pre-shared key shall not be used to authenticate both an application client and a device server;
c) the same pre-shared key should not be used for a group of application clients or a group of device servers;
d) information about the size of the pre-shared key shall be stored at the same time that the pre-shared key is stored; and
e) the means for provisioning pre-shared keys are outside the scope of this standard.

5.13.4.3.3 Digital signature authentication

5.13.4.3.3.1 Overview

Digital signature authentication uses a matched pair of signature and verification cryptographic algorithms to generate and verify authentication data that is a digital signature. A public/private key pair is associated with an identity. The private key is used as the authentication key for the identity. The public key is used as the verification key for the identity.

NOTE 16 - A private authentication key that is not kept secret may compromise the security properties of IKEv2-SCSI.

If digital signature authentication is used, then the private key is the key for the signature algorithm. A digital signature is generated by applying the signature algorithm with this private key to the input data (e.g., the applicable concatenation of bytes described in 7.7.3.5.7).

Verification of the digital signature shall consist of using the public verification key associated with the identity and the input data to verify the digital signature received as the contents of the AUTHENTICATION DATA field of the Authentication payload (see 7.7.3.5.7). Verification is successful if the digital signature is a valid digital signature over the input data, otherwise verification is not successful.

The means by which an application client or device server obtains a private authentication key are outside the scope of this standard. An identity and associated public verification key are obtained as follows:

a) if certificates are used for digital signature authentication, then the identity and the associated public verification key are obtained from a certificate transferred in the first the IKEv2-SCSI Certificate payload (see 5.13.4.3.3.4); or
b) if certificates are not used for digital signature authentication, then the identity is transferred in the IDENTIFICATION DATA field of the appropriate IKEv2-SCSI Identification payload (see 7.7.3.5.4) and the public verification key may be:
   A) transferred as a raw RSA key in an IKEv2-SCSI Certificate payload (see 7.7.3.5.5); or
   B) obtained by means that are outside the scope of this standard.

If certificates are not used for digital signature authentication, the association between the identity and the public key should be verified by means outside the scope of this standard.

5.13.4.3.3.2 Certificates and digital signature authentication

A certificate (see RFC 5280 and RFC 6818) is a data structure that contains:

a) an identity;
b) a public key for that identity;
c) additional relevant information that may constrain use of the public key;
d) the identity of a certification authority (see RFC 5280 and RFC 6818); and
e) a digital signature generated by that certification authority.

If the identity and associated public key used to verify a digital signature are obtained from a certificate, then the certification path from the certificate to a trust anchor should be validated (see RFC 5280 and RFC 6818). If certification path validation is not successful, verification of the digital signature for that identity shall fail independent of whether the digital signature is valid.

The means by which an application client or device server obtains a trust anchor are outside the scope of this standard.

5.13.4.3.3.3 Example of certificate use for digital signature authentication

An example of certificate use involves an application client or device server that trusts a certification authority. Based on this trust, the public key of that certification authority is used to validate a certificate presented as part of authentication. Successful validation of that certificate establishes that the public key in that certificate is associated with the identity in the certificate. That public key is then used to verify the digital signature in the Authentication payload (see 7.7.3.5.7).

In this example, providing a certificate as part of the IKEv2-SCSI Authentication step (see 5.13.4.7) allows a single certification authority public key to serve as a trust anchor (see RFC 5280 and RFC 6818) for verification of digital signatures for any identity that has been issued a certificate by that certification authority, so that a public key for each identity does not have to be obtained by other means.

Validating a certificate includes multiple checks beyond verifying the signature, and the validation may traverse a certification path composed of multiple certificates (see RFC 5280 and RFC 6818).

5.13.4.3.3.4 Handling of the Certificate Request payload and the Certificate payload

As detailed in this subclause, a Certificate Request payload (see 7.7.3.5.6) in one set of parameter data requests the delivery of a Certificate payload (see 7.7.3.5.5) in the next set of parameter data transferred. The purpose of these IKEv2-SCSI protocol elements is as follows:

a) each SA participant is allowed to require the delivery of a Certificate payload by the other SA participant for use in authentication; and
b) each Certificate Request payload indicates the trust anchors list (see RFC 4306) used by the device server or application client when PKI-based Authentication is being used with certificates that are not self signed (see RFC 5280 and RFC 6818).

The presence of one or more Certificate Request payloads in the Key Exchange step SECURITY PROTOCOL IN command (see 5.13.4.6.3) parameter data indicates that the device server requires the application client to include a Certificate payload in the parameter list for the Authentication step SECURITY PROTOCOL OUT command (see 5.13.4.7.2).

The presence of one or more Certificate Request payloads in the Authentication step SECURITY PROTOCOL OUT command parameter list specifies that the application client requires the device server to return a Certificate payload in the parameter data for the Authentication step SECURITY PROTOCOL IN command (see 5.13.4.7.3).

If any Certificate payloads are included in the parameter data, the first Certificate payload shall contain the public key used to verify the Authentication payload. Additional Certificate payloads may be used to assist in establishing a certification path from the certificate in the first payload to a trust anchor (see RFC 4306, RFC 5280 and RFC 6818).

The application client and device server may use different authentication methods that require or do not require the use of Certificate payloads. The presence or absence of Certificate Request payloads and Certificate payloads may vary in any of the commands described in this subclause.

5.13.4.3.4 Constraints on skipping the Authentication step

In the Device Server Capabilities step (see 5.13.4.5), the parameter data returned by the SECURITY PROTOCOL IN command (see 7.7.2.3.2) contains the IKEv2-SCSI SA Creation Algorithms payload (see 7.7.3.5.12) that contains one or more SA_AUTH_OUT IKEv2-SCSI cryptographic algorithm descriptors (see 7.7.3.6.6) and one or more SA_AUTH_IN IKEv2-SCSI cryptographic algorithm descriptors.

The device server shall allow the Authentication step to be omitted (see 5.13.4.1) if:

a) the ALGORITHM IDENTIFIER field is set to SA_AUTH_NONE (see 7.7.3.6.6) in one of the SA_AUTH_OUT IKEv2-SCSI cryptographic algorithm descriptors returned in the Device Server Capabilities step; and
b) the ALGORITHM IDENTIFIER field is set to SA_AUTH_NONE in one of the SA_AUTH_IN IKEv2-SCSI cryptographic algorithm descriptors returned in the Device Server Capabilities step.

The methods for configuring a device server to return SA_AUTH_NONE are outside the scope of this standard. Device servers shall not be manufactured to return SA_AUTH_NONE as an Authentication payload authentication algorithm type in the Device Server Capabilities step.

In the Key Exchange step SECURITY PROTOCOL OUT command (see 5.13.4.6.2), the application client requests that the Authentication step be omitted by setting the ALGORITHM IDENTIFIER field to SA_AUTH_NONE in:

a) the SA_AUTH_OUT cryptographic algorithm descriptor in the IKEv2-SCSI SA Cryptographic Algorithms payload (see 7.7.3.5.13); and
b) the SA_AUTH_IN cryptographic algorithm descriptor in the IKEv2-SCSI SA Cryptographic Algorithms payload.

To ensure adequate SA security, the application client should not select the SA_AUTH_NONE value as an Authentication payload authentication algorithm type unless:

a) an SA_AUTH_OUT IKEv2-SCSI cryptographic algorithm descriptor and an SA_AUTH_IN IKEv2-SCSI cryptographic algorithm descriptor from the Device Server Capabilities step indicates SA_AUTH_NONE availability; and
b) the application client is configured to omit the Authentication step.

If SA_AUTH_NONE is used, IKEv2-SCSI has no protection against man-in-the-middle attacks. Enabling return of the SA_AUTH_NONE authentication algorithm type in the Device Capabilities step, and allowing an application client to select SA_AUTH_NONE in the Key Exchange step are administrative security policy decisions that absence of authentication is acceptable. Such decisions should only be made in situations where active attacks on IKEv2-SCSI are not of concern (e.g., direct attachment of a SCSI initiator device and a SCSI target device, or an end-to-end secure service delivery subsystem such as Fibre Channel secured by an end-to-end FC-SP-2 SA).

5.13.4.4 Summary of IKEv2-SCSI shared keys nomenclature and shared key sizes

The IKEv2-SCSI shared keys are named as shown in table 81.

Table 81 — IKEv2-SCSI shared key names and SA shared key names

Name	SA parameter that stores this shared key	Description
Shared keys used only during Authentication step		
SK_pi	shall not be stored in any SA parameter	Shared key used to construct the Authentication payload (see 7.7.3.5.7) for the SECURITY PROTOCOL OUT parameter list in the Authentication step (see 5.13.4.7.2).
SK_pr		Shared key used to construct the Authentication payload for the SECURITY PROTOCOL IN parameter data in the Authentication step.
Shared keys used: a) during IKEv2-SCSI SA creation and management; and b) for bytes in a Data-Out Buffer or Data-In Buffer		
SK_ai	MGMT_DATA	Shared key used to integrity check the Encrypted payload in the SECURITY PROTOCOL OUT parameter list in the: a) Authentication step; and b) IKEv2-SCSI Delete operation (see 5.13.4.11).
	KEYMAT	Shared key used to integrity check the contents of a Data-Out Buffer.
SK_ar	MGMT_DATA	Shared key used to integrity check the Encrypted payload (see 7.7.3.5.11) in the SECURITY PROTOCOL IN parameter data in the Authentication step (see 5.13.4.7.3).
	KEYMAT	Shared key used to integrity check the contents of a Data-In Buffer.
SK_ei	MGMT_DATA	Shared key used to encrypt the Encrypted payload in the SECURITY PROTOCOL OUT parameter list in the: a) Authentication step; and b) IKEv2-SCSI Delete operation.
	KEYMAT	Shared key used to encrypt the contents of a Data-Out Buffer.
SK_er	MGMT_DATA	Shared key used to encrypt the Encrypted payload in the SECURITY PROTOCOL IN parameter data in the Authentication step.
	KEYMAT	Shared key used to encrypt the contents of a Data-In Buffer.
Shared key used to construct the SA keys		
SK_d	KEY_SEED	Shared key material that is used as input to the KDF that generates the KEYMAT SA parameter bytes for the SA.

The sizes of the shared keys are determined as shown in table 82.

Table 82 — Shared key size determination

Name	Shared key size determination	
	Separate encryption and integrity checking [a, b]	Combined mode encryption and integrity checking [a, b]
SK_ai and SK_ar [c]	The shared key size is shown in table 553 for the value in the ALGORITHM IDENTIFIER field of the INTEG IKEv2-SCSI cryptographic algorithm descriptor (see 7.7.3.6.4) [b].	zero [d]
SK_er and SK_ei [c]	The shared key size is the value in the KEY LENGTH field of the ENCR IKEv2-SCSI cryptographic algorithm descriptor (see 7.7.3.6.2) [b].	The shared key size is the value in the KEY LENGTH field of the ENCR IKEv2-SCSI cryptographic algorithm descriptor plus the number of salt bytes shown in table 549 (see 7.7.3.6.2) [b].
SK_pi and SK_pr [c]	The shared key size is equal to the PRF output length (see table 551) associated with the value in the ALGORITHM IDENTIFIER field of the PRF IKEv2-SCSI cryptographic algorithm descriptor (see 7.7.3.6.3) in the IKEv2-SCSI SA Cryptographic Algorithms payload (see 7.7.3.5.13).	
SK_d		

[a] The use of combined mode encryption and integrity checking is indicated by the AUTH_COMBINED value in the ALGORITHM IDENTIFIER field in the INTEG IKEv2-SCSI cryptographic algorithm descriptor (see 7.7.3.6.4).

[b] For the shared keys used to create an SA, the algorithm descriptor is located in an IKEv2-SCSI SA Cryptographic Algorithms payload (see 7.7.3.5.13). For the shared keys used after the SA is created (i.e., the KEYMAT SA parameter), the algorithm descriptor is located in an IKEv2-SCSI SAUT Cryptographic Algorithms payload (see 7.7.3.5.14).

[c] To accommodate two shared keys of the specified size, the shared key length shown is doubled for the purposes of shared key generation.

[d] In combined mode encryption and integrity checking, the SK_er and SK_ei are used for both encryption and integrity checking.

5.13.4.5 Device Server Capabilities step

In the Device Server Capabilities step, the application client sends a SECURITY PROTOCOL IN command (see 6.40) with the SECURITY PROTOCOL field set to SA creation capabilities (i.e., 40h) and the SECURITY PROTOCOL SPECIFIC field set to 0101h.

The device server returns the SECURITY PROTOCOL IN parameter data specified by the SECURITY PROTOCOL SPECIFIC field (see 7.7.2.2) and the parameter data (see 7.7.2.3.2) contains an IKEv2-SCSI SA Creation Capabilities payload (see 7.7.3.5.12).

In the Device Server Capabilities step, the device server shall return parameter data containing the IKEv2-SCSI cryptographic algorithm descriptors (see 7.7.3.6) in at least one complete row shown in table 83.

Table 83 — Device Server Capabilities step parameter data requirements

IKEv2-SCSI cryptographic algorithm descriptor					
ENCR (see 7.7.3.6.2)	**PRF (see 7.7.3.6.3)**	**INTEG (see 7.7.3.6.4)**	**D-H (see 7.7.3.6.5)**	**SA_AUTH_OUT (see 7.7.3.6.6)**	**SA_AUTH_IN (see 7.7.3.6.6)**
The ALGORITHM IDENTIFIER field set to 8001 0014h (i.e., AES-GCM) and the KEY LENGTH field set to 0010h	The ALGORITHM IDENTIFIER field set to 8002 0005h (i.e., IKEv2-use based on SHA-256)	The ALGORITHM IDENTIFIER field set to F003 0001h (i.e., AUTH_ COMBINED)	The ALGORITHM IDENTIFIER field set to 8004 000Eh (i.e., 2 048-bit MODP group (finite field D-H))	The ALGORITHM IDENTIFIER field set to 00F9 0001h (i.e., RSA Digital Signature)	The ALGORITHM IDENTIFIER field set to 00F9 0001h (i.e., RSA Digital Signature)
The ALGORITHM IDENTIFIER field set to 8001 000Ch (i.e., AES-CBC) and the KEY LENGTH field set to 0020h	The ALGORITHM IDENTIFIER field set to 8002 0007h (i.e., IKEv2-use based on SHA-512)	The ALGORITHM IDENTIFIER field set to 8003 000Eh (i.e., AUTH_HMAC_ SHA2_ 512_256)	The ALGORITHM IDENTIFIER field set to 8004 0015h (i.e., 521-bit random ECP group)	The ALGORITHM IDENTIFIER field set to 00F9 000Bh (i.e., ECDSA with SHA-512 on the P-521 curve)	The ALGORITHM IDENTIFIER field set to 00F9 000Bh (i.e., ECDSA with SHA-512 on the P-521 curve)

In the Device Server Capabilities step, the device server shall return parameter data containing one SA_AUTH_OUT IKEv2-SCSI cryptographic algorithm descriptor with the ALGORITHM IDENTIFIER field is set to SA_AUTH_NONE and one SA_AUTH_IN IKEv2-SCSI cryptographic algorithm descriptor with the ALGORITHM IDENTIFIER field is set to SA_AUTH_NONE if any of the following are true:

a) the ALGORITHM IDENTIFIER field is set to SA_AUTH_NONE in one of the SA_AUTH_OUT IKEv2-SCSI cryptographic algorithm descriptors returned in the Device Server Capabilities step; or
b) the ALGORITHM IDENTIFIER field is set to SA_AUTH_NONE in one of the SA_AUTH_IN IKEv2-SCSI cryptographic algorithm descriptors returned in the Device Server Capabilities step.

The device server capabilities returned in the SECURITY PROTOCOL IN parameter data may be changed at any time by means that are outside the scope of this standard, However, such changes shall not take effect until at least one application client has been notified of the new capabilities in the parameter data returned by the Device Server Capabilities step SECURITY PROTOCOL IN command. Management applications may ensure that their device server capabilities changes take effect by sending a Device Server Capabilities step SECURITY PROTOCOL IN command to the device server after the changes have been made.

If the device server capabilities change (i.e., upon completion of the processing for a Device Server Capabilities step SECURITY PROTOCOL IN command that reported changed information in its parameter data), then the device server shall establish a unit attention condition for the initiator port associated with every I_T nexus except the I_T nexus on which the Device Server Capabilities step SECURITY PROTOCOL IN command was received (see SAM-5), with the additional sense code set to SA CREATION CAPABILITIES DATA HAS CHANGED.

The Device Server Capabilities step participates in the negotiation to skip the Authentication step as described in 5.13.4.3.4.

NOTE 17 - The Device Server Capabilities step has no IKEv2 exchange equivalent in RFC 4306. This step replaces most of IKEv2's negotiation by having the application client obtain the supported capabilities from the device server.

5.13.4.6 IKEv2-SCSI Key Exchange step

5.13.4.6.1 Overview

The Key Exchange step consists of a Diffie-Hellman key exchange with nonces (see RFC 4306) and is accomplished as follows:

1) a SECURITY PROTOCOL OUT command (see 5.13.4.6.2);
2) a SECURITY PROTOCOL IN command (see 5.13.4.6.3); and
3) key exchange completion (see 5.13.4.6.4)

NOTE 18 - The Key Exchange step corresponds to the IKEv2 IKE_SA_INIT exchange in RFC 4306, except that determination of device server capabilities has been moved to the Device Server Capabilities step.

5.13.4.6.2 Key Exchange step SECURITY PROTOCOL OUT command

To send its key exchange message to the device server, the application client sends a SECURITY PROTOCOL OUT command (see 6.41) with the SECURITY PROTOCOL field set to IKEv2-SCSI (i.e., 41h) and the SECURITY PROTOCOL SPECIFIC field set to 0102h. The parameter list consists of an IKEv2-SCSI header (see 7.7.3.4) and the following:

1) an IKEv2-SCSI Timeout Values payload (see 7.7.3.5.15);
2) an IKEv2-SCSI SA Cryptographic Algorithms payload (see 7.7.3.5.13);
3) if the Authentication step is skipped (see 5.13.4.1), an IKEv2-SCSI SAUT Cryptographic Algorithms payload (see 7.7.3.5.14);
4) a Key Exchange payload (see 7.7.3.5.3); and
5) a Nonce payload (see 7.7.3.5.8).

The IKEv2-SCSI Timeout Values payload contains the inactivity timeouts that apply to this IKEv2-SCSI SA creation transaction and the SA that is created.

The IKEv2-SCSI SA Cryptographic Algorithms payload selects the cryptographic algorithms from among those returned in the Device Server Capabilities step (see 5.13.4.5) to be used in the creation of the SA.

If the Authentication step is skipped, the IKEv2-SCSI SAUT Cryptographic Algorithms payload contains the following information about the SA to be created:

a) the cryptographic algorithms selected by the application client from among those returned in the Device Server Capabilities step; and
b) the usage data (see 7.7.3.5.14), if any, that is specific to the SA.

If the application client is unable to select a set of algorithms that are appropriate for the intended creation and usage of the SA, then the application client should not perform the Key Exchange step to request the creation of an SA.

IKEv2-SCSI SA Cryptographic Algorithms payload error checking requirements that ensure a successful negotiation of SA creation algorithms are described in 7.7.3.5.13 and 7.7.3.6.

IKEv2-SCSI SAUT Cryptographic Algorithms payload error checking requirements that ensure a successful SA creation are described in 7.7.3.5.14 and 7.7.3.6.

The Key Exchange payload contains the application client's Diffie-Hellman value.

The Nonce payload contains the application client's random nonce.

5.13.4.6.3 Key Exchange step SECURITY PROTOCOL IN command

If the Key Exchange step SECURITY PROTOCOL OUT command (see 5.13.4.6.2) completes with GOOD status, then the application client sends a SECURITY PROTOCOL IN command (see 6.40) with the SECURITY PROTOCOL field set to IKEv2-SCSI (i.e., 41h) and the SECURITY PROTOCOL SPECIFIC field set to 0102h to obtain the device server's key exchange message.

The parameter data returned by the device server in response to the SECURITY PROTOCOL IN command shall contain an IKEv2-SCSI header (see 7.7.3.4) and the following:

1) an IKEv2-SCSI SA Cryptographic Algorithms payload (see 7.7.3.5.13);
2) if the Authentication step is skipped (see 5.13.4.1), an IKEv2-SCSI SAUT Cryptographic Algorithms payload (see 7.7.3.5.14);
3) a Key Exchange payload (see 7.7.3.5.3);
4) a Nonce payload (see 7.7.3.5.8); and
5) zero or more Certificate Request payloads (see 7.7.3.5.6).

As part of processing of the Key Exchange step SECURITY PROTOCOL IN command, the device server shall:

a) associate the SECURITY PROTOCOL IN command to the last Key Exchange step SECURITY PROTOCOL OUT command received on the I_T_L nexus. If the device server is maintaining state for at least one IKEv2-SCSI CCS and the device server is unable to establish this association, then the device server shall:
   A) terminate the SECURITY PROTOCOL IN command with CHECK CONDITION status, with the sense key set to NOT READY, and the additional sense code set to LOGICAL UNIT NOT READY, SA CREATION IN PROGRESS; and
   B) continue the IKEv2-SCSI CCS.

   If the device is not maintaining state for at least one IKEv2-SCSI CSS, the device server shall terminate the SECURITY PROTOCOL IN command with CHECK CONDITION status, with the sense key set to ILLEGAL REQUEST, and the additional sense code set to COMMAND SEQUENCE ERROR.
b) return the device server's SAI in the IKEv2-SCSI header IKE_SA DEVICE SERVER SAI field;
c) return the IKEv2-SCSI SA Cryptographic Algorithms payload containing:
   A) the SA creation cryptographic algorithms supplied by the application client in the Key Exchange step SECURITY PROTOCOL OUT command parameter list; and
   B) the device server's SAI in the SAI field (see 7.7.3.5.13);
d) if the Authentication step is skipped, return the IKEv2-SCSI SAUT Cryptographic Algorithms payload containing:
   A) the SA usage cryptographic algorithms supplied by the application client in the Key Exchange step SECURITY PROTOCOL OUT command parameter list; and
   B) the device server's SAI in the SAI field (see 7.7.3.5.14);
e) return information about the completed Diffie-Hellman exchange with the Key Exchange payload; and
f) return the device server's random nonce in the Nonce payload.

If the Key Exchange step SECURITY PROTOCOL IN command (see 5.13.4.6.3) completes with GOOD status, then the application client should copy the device server's SAI from the IKE_SA DEVICE SERVER SAI field in the IKEv2-SCSI header to the state it is maintaining for the IKEv2-SCSI CCS.

Except for the SAI field, the application client should compare the fields in the IKEv2-SCSI SA Cryptographic Algorithms payload and the IKEv2-SCSI SAUT Cryptographic Algorithms payload, if any, to the values sent in the Key Exchange step SECURITY PROTOCOL OUT command (see 5.13.4.6.2). If the application client detects differences in the contents of the payloads other than in the SAI field, then the application client should abandon the IKEv2-SCSI CCS and notify the device server that the IKEv2-SCSI CCS is being abandoned as described in 5.13.4.10.

5.13.4.6.4 Key Exchange step completion

Before completing the Key Exchange step SECURITY PROTOCOL IN command (see 5.13.4.6.3) with GOOD status the device server shall complete the Key Exchange step as described in this subclause.

Upon receipt of GOOD status for the Key Exchange step SECURITY PROTOCOL IN command the application client should complete the Key Exchange step as described in this subclause.

If the Key Exchange step does not end with the IKEv2-SCSI CCS being abandoned (see 5.13.4.10), then the contents of the SA_AUTH_OUT IKEv2-SCSI cryptographic algorithm descriptor and SA_AUTH_IN IKEv2-SCSI cryptographic algorithm descriptor (see 7.7.3.6.6) in the IKEv2-SCSI SA Cryptographic Algorithms payload specify how the shared key exchanged by the Key Exchange step SECURITY PROTOCOL OUT command and the Key Exchange step SECURITY PROTOCOL IN command is used to generate additional shared keys as follows:

a) if the ALGORITHM IDENTIFIER field in both descriptors contain SA_AUTH_NONE, then the SA participants generate the SA, including the generation of the shared keys used for SA management (e.g., SA creation and management) and the shared keys used by the created SA as defined in 5.13.4.9; or
b) if the ALGORITHM IDENTIFIER field in both descriptors contain a value other than SA_AUTH_NONE, then the SA participants generate shared keys (see 5.13.4.8.3) for the following:
   A) seeding the Authentication step generation of the shared keys used by the created SA; and
   B) SA creation and management.

5.13.4.6.5 After the Key Exchange step

Processing of the IKEv2-SCSI CCS subsequent to completion of the Key Exchange step (see 5.13.4.6.4) depends on the contents of the SA_AUTH_OUT IKEv2-SCSI cryptographic algorithm descriptor and SA_AUTH_IN IKEv2-SCSI cryptographic algorithm descriptor (see 7.7.3.6.6) in the IKEv2-SCSI SA Cryptographic Algorithms payload as follows:

a) if the ALGORITHM IDENTIFIER field in both descriptors contain SA_AUTH_NONE, then processing of the IKEv2-SCSI CCS is finished and the generated SA is ready for use; or
b) if the ALGORITHM IDENTIFIER field in both descriptors contain a value other than SA_AUTH_NONE, then processing of the IKEv2-SCSI CCS continues as follows:
   A) the Authentication step is performed (see 5.13.4.7); and
   B) the SA participants generate the SA, including the generation of the shared keys used for SA management and the shared keys used by the created SA as defined in 5.13.4.9.

5.13.4.7 IKEv2-SCSI Authentication step

5.13.4.7.1 Overview

The Authentication step performs the following functions:

a) authenticates both the application client and the device server;
b) protects the previous steps of the protocol; and
c) cryptographically binds the authentication and the previous steps to the created SA.

The Authentication step is accomplished as follows:

1) a SECURITY PROTOCOL OUT command (see 5.13.4.7.2); and
2) a SECURITY PROTOCOL IN command (see 5.13.4.7.3).

The parameter data for both commands shall be encrypted and integrity protected using the algorithms and keys determined in the Key Exchange step (see 5.13.4.6).

NOTE 19 - The Authentication step corresponds to the IKEv2 IKE_AUTH exchange in RFC 4306.

5.13.4.7.2 Authentication step SECURITY PROTOCOL OUT command

To send its authentication message to the device server, the application client sends a SECURITY PROTOCOL OUT command (see 6.41) with the SECURITY PROTOCOL field set to IKEv2-SCSI (i.e., 41h) and the SECURITY PROTOCOL SPECIFIC field set to 0103h. The parameter data consists of the IKEv2-SCSI header (see 7.7.3.4) and an Encrypted payload (see 7.7.3.5.11) that:

a) is integrity checked and encrypted in one of the following ways:
   A) using separate algorithms as follows:
      a) integrity checked using the following:
         A) the algorithm specified by the INTEG IKEv2-SCSI algorithm descriptor (see 7.7.3.6.4) in the IKEv2-SCSI SA Cryptographic Algorithms payload (see 7.7.3.5.13) from the Key Exchange step (see 5.13.4.6); and
         B) the SK_ai shared key (see 5.13.4.4);
         and
      b) encrypted using the following:
         A) the algorithm specified by the ENCR IKEv2-SCSI algorithm descriptor (see 7.7.3.6.2) in the IKEv2-SCSI SA Cryptographic Algorithms payload from the Key Exchange step; and
         B) the SK_ei shared key (see 5.13.4.4);
      or
   B) using a combined integrity check and encryption algorithm that uses the following (i.e., if the INTEG IKEv2-SCSI algorithm descriptor indicates AUTH_COMBINED):
      a) the algorithm specified by the ENCR IKEv2-SCSI algorithm descriptor in the IKEv2-SCSI SA Cryptographic Algorithms payload; and
      b) the SK_ei shared key with additional salt bytes as described in 5.13.4.8.1 and table 82 (see 5.13.4.4);
   and
b) contains the following:
   1) an Identification – Application Client payload (see 7.7.3.5.4);
   2) an IKEv2-SCSI SAUT Cryptographic Algorithms payload (see 7.7.3.5.14);
   3) zero or more Certificate payloads (see 7.7.3.5.5);
   4) zero or more Certificate Request payloads (see 7.7.3.5.6);
   5) zero or one Notify payload (see 7.7.3.5.9); and
   6) an Authentication payload (see 7.7.3.5.7).

Before performing any checks on data contained in the Encrypted payload, the device server shall validate the SECURITY PROTOCOL OUT command parameter data as follows:

a) the device server shall compare the IKE_SA APPLICATION CLIENT SAI field and the IKE_SA DEVICE SERVER SAI field in the IKEv2-SCSI header to the SAI values it is maintaining for the IKEv2-SCSI CCS, if any, being maintained for the I_T_L nexus on which the SECURITY PROTOCOL OUT command was received as described in 7.7.3.4; and
b) the device server shall decrypt and check the integrity of the Encrypted payload as described in 7.7.3.5.11.4.

Errors detected during the decryption and integrity checking of the Encrypted payload shall be handled as described in 7.7.3.8.2.

In the SECURITY PROTOCOL OUT command parameter list, the application client:

a) sends information about the SA usage and cryptographic algorithms;
b) sends its identity in the Identification – Application Client payload;
c) sends information proving knowledge of the secret corresponding to the application client's identity in the Authentication payload; and

d) integrity protects the Key Exchange step and the Authentication step using the Authentication payload.

The application client may include the Notify payload to send an initial contact notification to the device server. If sent, the initial contact notification specifies that the application client has no stored state for any SAs with the device server other than the SA that is being created.

In response to receipt of an initial contact notification, the device server should delete all other SAs that were authenticated with a SECURITY PROTOCOL OUT command that contained the same Identification - Application Client payload data as that which is present in the SECURITY PROTOCOL OUT command that the device server is processing.

If the device server deletes other SAs in response to an initial contact notification, the device server shall do so only after the successful completion of the Authentication step SECURITY PROTOCOL OUT command. If an error occurs during the Authentication SECURITY PROTOCOL OUT command, the device server shall ignore the initial contact notification.

If the device server is unable to proceed with SA creation for any reason (e.g., the verification of the Authentication payload fails), then the device server shall terminate the SECURITY PROTOCOL OUT command with CHECK CONDITION status, with the sense key set to ABORTED COMMAND, and the additional sense code set to an appropriate value. The additional sense code AUTHENTICATION FAILED shall be used if verification of the Authentication payload fails, or if authentication fails for any other reason.

5.13.4.7.3 Authentication step SECURITY PROTOCOL IN command

If the Authentication step SECURITY PROTOCOL OUT command (see 5.13.4.7.2) completes with GOOD status, then the application client sends a SECURITY PROTOCOL IN command (see 6.40) with the SECURITY PROTOCOL field set to IKEv2-SCSI (i.e., 41h) and the SECURITY PROTOCOL SPECIFIC field set to 0103h to obtain the device server's authentication message. The parameter data consists of the IKEv2-SCSI header (see 7.7.3.4) and an Encrypted payload (see 7.7.3.5.11) that:

a) is integrity checked and encrypted in one of the following ways:
   A) using separate algorithms as follows:
      a) integrity checked using the following:
         A) the algorithm specified by the INTEG IKEv2-SCSI algorithm descriptor (see 7.7.3.6.4) in the IKEv2-SCSI SA Cryptographic Algorithms payload (see 7.7.3.5.13) from the Key Exchange step (see 5.13.4.6); and
         B) the SK_ar shared key (see 5.13.4.4);
         and
      b) encrypted using the following:
         A) the algorithm specified by the ENCR IKEv2-SCSI algorithm descriptor (see 7.7.3.6.2) in the IKEv2-SCSI SA Cryptographic Algorithms payload from the Key Exchange step; and
         B) the SK_er shared key (see 5.13.4.4);
      or
   B) using a combined integrity check and encryption algorithm that uses the following (i.e., if the INTEG IKEv2-SCSI algorithm descriptor indicates AUTH_COMBINED):
      a) the algorithm specified by the ENCR IKEv2-SCSI algorithm descriptor in the IKEv2-SCSI SA Cryptographic Algorithms payload from the Key Exchange step; and
      b) the SK_er shared key with additional salt bytes as described in 5.13.4.8.1 and table 82 (see 5.13.4.4);
   and
b) contains the following:
   1) an Identification – Device Server payload (see 7.7.3.5.4);
   2) an IKEv2-SCSI SAUT Cryptographic Algorithms payload (see 7.7.3.5.14);
   3) zero or more Certificate payloads (see 5.13.4.3.3.4); and
   4) an Authentication payload (see 7.7.3.5.7).

In the SECURITY PROTOCOL IN parameter data, the device server shall:

a) confirm information about the SA usage and cryptographic algorithms;
b) send its identity in the Identification – Device Server payload;
c) authenticate its identity; and
d) protect the integrity of the prior step messages using the Authentication payload.

Before completing the SECURITY PROTOCOL IN command with GOOD status, the device server shall generate the SA as described in 5.13.4.9.

The application client should verify the Authentication payload as described in 7.7.3.5.7. The Certificate payload(s) are used as part of this verification for PKI-based authentication. If the Authentication payload is verified and no other error occurs, the application client should generate the SA as described in 5.13.4.9.

If the application client is unable to proceed with SA creation for any reason (e.g., the verification of the Authentication payload fails), then the application client should:

a) not use the SA for any additional activities; and
b) notify the device server that the IKEv2-SCSI CCS is being abandoned as described in 5.13.4.10.

The application client should compare the fields in the IKEv2-SCSI SAUT Cryptographic Algorithms payload to the values sent in the Authentication step SECURITY PROTOCOL OUT command (see 5.13.4.7.2). If the application client detects differences in the contents of the payloads, the application client should abandon the IKEv2-SCSI CCS and notify the device server that the IKEv2-SCSI CCS is being abandoned as described in 5.13.4.10.

5.13.4.8 Generating shared keys

5.13.4.8.1 Overview

If the Authentication step is skipped (see 5.13.4.1), then shared key generation is performed as described in 5.13.4.8.2 and is summarized as follows:

a) the shared keys for SA management (e.g., SA creation and deletion) are generated at the same time as the shared keys used by the created SA; and
b) all the shared keys are generated during completion of the Key Exchange step (see 5.13.4.6.4).

If the Authentication step is processed (i.e., not skipped), then shared key generation is performed as described in 5.13.4.8.3 and is summarized as follows:

a) the shared keys for SA management (e.g., SA creation and deletion) are generated during the completion of the Key Exchange step (see 5.13.4.6.4); and
b) the shared keys used by the created SA are generated during SA generation (see 5.13.4.9).

Regardless of when the SA management shared keys (e.g., used for SA creation and deletion) and shared keys (see 5.13.4.4) used by the created SA are generated, the organization of the shared keys depends on the type of encryption and integrity checking algorithm being used as follows:

a) if an encryption algorithm that requires separate integrity checking is used, then separate shared keys are generated for each algorithm; or
b) if an encryption algorithm that includes integrity checking is used (i.e., if the ALGORITHM IDENTIFIER field in the INTEG IKEv2-SCSI cryptographic algorithm descriptor (see 7.7.3.6.4) contains AUTH_COMBINED), then no shared keys are generated for the integrity checking algorithm but additional key material is generated to act as salt bytes (see table 549 in 7.7.3.6.2) for the combined mode encryption algorithm.

5.13.4.8.2 Generating shared keys when the Authentication step is skipped

If the Authentication step is skipped (see 5.13.4.1), then the shared keys for SA management are generated at the same time as the shared keys for use by the created SA.

As part of completing the Key Exchange step (see 5.13.4.6.4), the SA participants generate all necessary shared keys as follows:

1) generate SKEYSEED (see RFC 4306) as described in 5.13.4.8.4;
2) generate the shared keys used for SA management as described in 5.13.4.8.5;
3) as part of generating the SA (see 5.13.4.9) (i.e., as part of completing the Key Exchange step as described in 5.13.4.6.4), generate the shared keys for use by the created SA as described in 5.13.4.8.6 and store them in the KEYMAT SA parameter.

5.13.4.8.3 Generating shared keys when the Authentication step is processed

If the Authentication step is not skipped (see 5.13.4.1), then:

1) the shared keys for SA management are generated during completion of the Key Exchange step (see 5.13.4.6.4) as follows;
   1) generate SKEYSEED (see RFC 4306) as described in 5.13.4.8.4; and
   2) generate the shared keys used for SA management as described in 5.13.4.8.5;

   and
2) the shared keys for use by the created SA are generated during SA generation (see 5.13.4.9), near the end of processing for the Authentication step (see 5.13.4.7) as described in 5.13.4.8.6 and store them in the KEYMAT SA parameter.

5.13.4.8.4 Initializing shared key generation

5.13.4.8.4.1 Initializing for SA creation shared key generation

The SA parameters are initialized for the KDF function used to generate SA creation shared keys as follows:

1) generate the input to the PRF function by performing the last steps of the key exchange algorithm selected by the ALGORITHM IDENTIFIER field in the D-H IKEv2-SCSI algorithm descriptor (see 7.7.3.6.5) in the IKEv2-SCSI SA Cryptographic Algorithms payload (see 7.7.3.5.13) using at least the contents of one of the following fields as inputs to those last steps:
   A) the KEY EXCHANGE DATA field in the Key Exchange payload (see 7.7.3.5.3) in the Key Exchange SECURITY PROTOCOL OUT command (see 5.13.4.6.2) parameter data; and
   B) the KEY EXCHANGE DATA field in the Key Exchange payload in the Key Exchange SECURITY PROTOCOL IN command (see 5.13.4.6.3) parameter data;
2) generate SKEYSEED (see RFC 4306) using the output from step 1) and the PRF selected by the ALGORITHM IDENTIFIER field in the PRF IKEv2-SCSI cryptographic algorithm descriptor (see 7.7.3.6.3) in the IKEv2-SCSI SA Cryptographic Algorithms payload;
3) store the generated SKEYSEED value in the KEY_SEED SA parameter;
4) store the contents of the IKE_SA APPLICATION CLIENT SAI field from the IKEv2-SCSI Header (see 7.7.3.4) from the Key Exchange step SECURITY PROTOCOL IN command (see 5.13.4.6.3) in the AC_SAI SA parameter;
5) store the contents of the IKE_SA DEVICE SERVER SAI field from the IKEv2-SCSI Header (see 7.7.3.4) from the Key Exchange step SECURITY PROTOCOL IN command (see 5.13.4.6.3) in the DS_SAI SA parameter;
6) store the contents of the NONCE DATA field from the Nonce payload (see 7.7.3.5.8) from the parameter data for the Key Exchange step SECURITY PROTOCOL OUT command (see 5.13.4.6.2) in the AC_NONCE SA parameter; and
7) store the contents of the NONCE DATA field from the Nonce payload (see 7.7.3.5.8) from the parameter data for the Key Exchange step SECURITY PROTOCOL IN command (see 5.13.4.6.3) in the DS_NONCE SA parameter.

5.13.4.8.4.2 Initializing for generation of shared keys used by the created SA

The SA parameters are initialized for the KDF function used to generate the shared keys that are used by the created SA as follows:

1) store the SK_d value that was generated along with the other shared keys used in SA creation (see 5.13.4.8.5) in the KEY_SEED SA parameter;
2) store the contents of the IKE_SA APPLICATION CLIENT SAI field from the IKEv2-SCSI Header (see 7.7.3.4) from the Key Exchange step SECURITY PROTOCOL IN command (see 5.13.4.6.3) in the AC_SAI SA parameter;
3) store the contents of the IKE_SA DEVICE SERVER SAI field from the IKEv2-SCSI Header (see 7.7.3.4) from the Key Exchange step SECURITY PROTOCOL IN command (see 5.13.4.6.3) in the DS_SAI SA parameter;
4) store the contents of the NONCE DATA field from the Nonce payload (see 7.7.3.5.8) from the parameter data for the Key Exchange step SECURITY PROTOCOL OUT command (see 5.13.4.6.2) in the AC_NONCE SA parameter; and
5) store the contents of the NONCE DATA field from the Nonce payload (see 7.7.3.5.8) from the parameter data for the Key Exchange step SECURITY PROTOCOL IN command (see 5.13.4.6.3) in the DS_NONCE SA parameter.

5.13.4.8.5 Generating shared keys used for SA management

The shared keys used for SA management (e.g., SA creation and deletion) are generated using the SKEYSEED generated during initialization (see 5.13.4.8.4) and the steps described in this subclause.

Correct operation of SA management requires that the application client and device server use the same value for each shared key. SA management operations attempted with different values for the same shared key result in errors (e.g., integrity check errors).

Which shared keys are generated for SA management depends on:

a) whether the encryption algorithm includes integrity checking as indicated by the contents of the ALGORITHM IDENTIFIER field in the INTEG IKEv2-SCSI cryptographic algorithm descriptor (see 7.7.3.6.4) of the IKEv2-SCSI SA Cryptographic Algorithms payload (see 7.7.3.5.13), and
b) whether the Authentication step is skipped (see 5.13.4.1).

Using the contents of the initialized SA parameters (see 5.13.4.8.4.1), the INTEG ALGORITHM IDENTIFIER field, and the KDF selected by the ALGORITHM IDENTIFIER field in the PRF IKEv2-SCSI cryptographic algorithm descriptor in the IKEv2-SCSI SA Cryptographic Algorithms payload the following shared keys (see 5.13.4.4) are generated in the order shown:

a) if the INTEG ALGORITHM IDENTIFIER field is not set to AUTH_COMBINED, then generate the following shared keys (see 5.13.4.4):
   A) if the Authentication step is not skipped, generate the following shared keys:
      1) SK_d;
      2) SK_ai;
      3) SK_ar;
      4) SK_ei;
      5) SK_er;
      6) SK_pi; and
      7) SK_pr;

      or
   B) if the Authentication step is skipped, generate the following shared keys:
      1) SK_d;
      2) SK_ai;
      3) SK_ar;
      4) SK_ei; and

5) SK_er;

or

b) if the INTEG ALGORITHM IDENTIFIER field is set to AUTH_COMBINED, then generate the following shared keys:
   A) if the Authentication step is not skipped, generate the following shared keys:
      1) SK_d;
      2) SK_ei with additional salt bytes as described in 5.13.4.8.1 and table 82 (see 5.13.4.4);
      3) SK_er with additional salt bytes as described in 5.13.4.8.1 and table 82 (see 5.13.4.4);
      4) SK_pi; and
      5) SK_pr;

      or
   B) if the Authentication step is skipped, generate the following shared keys:
      1) SK_d;
      2) SK_ei with additional salt bytes as described in 5.13.4.8.1 and table 82 (see 5.13.4.4); and
      3) SK_er with additional salt bytes as described in 5.13.4.8.1 and table 82 (see 5.13.4.4).

The shared keys thus generated are combined with other data and stored in the MGMT_DATA SA parameter as described in 5.13.4.9.

How to determine the sizes of the shared keys to be generated is summarized in table 82 (see 5.13.4.4).

5.13.4.8.6 Generating shared keys for use by the created SA

As part of completing the Authentication step and generating the SA (see 5.13.4.9), the SA participants initialize the SA parameters for performing a KDF (see 5.13.4.8.4.2), and use the KDF selected by the ALGORITHM IDENTIFIER field in the PRF IKEv2-SCSI cryptographic algorithm descriptor in the Key Exchange step IKEv2-SCSI SA Cryptographic Algorithms payload (see 7.7.3.5.13) to generate the following shared keys (see 5.13.4.4) in the order shown and store them in the KEYMAT SA parameter:

a) if the ALGORITHM IDENTIFIER field in the ENCR IKEv2-SCSI cryptographic algorithm descriptor (see 7.7.3.6.2) contains ENCR_NULL in the IKEv2-SCSI SAUT Cryptographic Algorithms payload (see 7.7.3.5.14), then generate the following shared keys:
   1) SK_ai;
   2) SK_ar;

   and
b) if the ALGORITHM IDENTIFIER field in the ENCR IKEv2-SCSI cryptographic algorithm descriptor does not contain ENCR_NULL in the IKEv2-SCSI SAUT Cryptographic Algorithms payload, then generate the following shared keys:
   A) if the ALGORITHM IDENTIFIER field in the INTEG IKEv2-SCSI cryptographic algorithm descriptor (see 7.7.3.6.4) does not contain AUTH_COMBINED in the IKEv2-SCSI SAUT Cryptographic Algorithms payload (see 7.7.3.5.14), then generate the following shared keys:
      1) SK_ai;
      2) SK_ar;
      3) SK_ei; and
      4) SK_er;

      or
   B) if the ALGORITHM IDENTIFIER field in the INTEG IKEv2-SCSI cryptographic algorithm descriptor contains AUTH_COMBINED in the IKEv2-SCSI SAUT Cryptographic Algorithms payload, then generate the following shared keys:
      1) SK_ei with additional salt bytes as described in 5.13.4.8.1 and table 82 (see 5.13.4.4); and
      2) SK_er with additional salt bytes as described in 5.13.4.8.1 and table 82 (see 5.13.4.4).

How to determine the sizes of the shared keys to be generated is summarized in table 82 (see 5.13.4.4).

5.13.4.9 IKEv2-SCSI SA generation

Depending on whether or not the Authentication step was skipped (see 5.13.4.1), the SA participants shall generate shared keys as described in 5.13.4.8.

The SA participants shall initialize the SA parameters as follows:

1) KEYMAT shall be set as follows:
   A) if the Authentication step is skipped, KEYMAT shall be set as described in 5.13.4.8.2; or
   B) if the Authentication step is processed, KEYMAT shall be set as described in 5.13.4.8.3;
   and
2) the other SA parameters shall be set as follows:
   A) AC_SAI shall be set to the value in the IKE_SA APPLICATION CLIENT SAI field in the IKEv2-SCSI header (see 7.7.3.4) in the Key Exchange step SECURITY PROTOCOL OUT command (see 5.13.4.6.2);
   B) DS_SAI shall be set to the value in the IKE_SA DEVICE SERVER SAI field in the IKEv2-SCSI header in the Key Exchange step SECURITY PROTOCOL IN command (see 5.13.4.6.3);
   C) TIMEOUT shall be set to the IKEV2-SCSI SA INACTIVITY TIMEOUT field in the IKEv2-SCSI Timeout Values payload (see 7.7.3.5.15) in the Key Exchange step SECURITY PROTOCOL OUT command;
   D) KDF_ID shall be set to the value in the ALGORITHM IDENTIFIER field in the PRF IKEv2-SCSI cryptographic algorithm descriptor (see 7.7.3.6.3) in the IKEv2-SCSI SA Cryptographic Algorithms payload (see 7.7.3.5.13);
   E) AC_SQN shall be set to one;
   F) DS_SQN shall be set to one;
   G) AC_NONCE shall be set to zero;
   H) DS_NONCE shall be set to zero;
   I) KEY_SEED shall be set to zero;
   J) USAGE_TYPE shall be set to the value in the SA TYPE field in the IKEv2-SCSI SAUT Cryptographic Algorithms payload (see 7.7.3.5.14) in the Key Exchange step SECURITY PROTOCOL OUT command;
   K) USAGE_DATA shall contain at least the following values from the IKEv2-SCSI SAUT Cryptographic Algorithms payload (see 7.7.3.5.14) in either the Key Exchange step SECURITY PROTOCOL OUT command or the Authentication step SECURITY PROTOCOL OUT command:
      a) the ALGORITHM IDENTIFIER field and KEY LENGTH field from the ENCR IKEv2-SCSI cryptographic algorithm descriptor (see 7.7.3.6.2);
      b) the ALGORITHM IDENTIFIER field from the INTEG IKEv2-SCSI cryptographic algorithm descriptor (see 7.7.3.6.4); and
      c) the contents, if any, of the USAGE DATA field;
      and
   L) MGMT_DATA shall contain at least the following values:
      a) from the IKEv2-SCSI SA Cryptographic Algorithms payload (see 7.7.3.5.13) in the Key Exchange step SECURITY PROTOCOL OUT command:
         A) the ALGORITHM IDENTIFIER field and KEY LENGTH field from the ENCR IKEv2-SCSI cryptographic algorithm descriptor (see 7.7.3.6.2); and
         B) the ALGORITHM IDENTIFIER field from the INTEG IKEv2-SCSI cryptographic algorithm descriptor (see 7.7.3.6.4);
      b) from the shared keys generated for SA management (see 5.13.4.8.5), the following shared keys and salt bytes:
         A) the value of SK_ai, if any;
         B) the value of SK_ar, if any;
         C) the value of SK_ei, with additional salt bytes, if any; and
         D) the value of SK_er, with additional salt bytes, if any;
         and
      c) the next value of the MESSAGE ID field in the IKEv2-SCSI header.

NOTE 20 - The inclusion of the algorithm identifiers and key length in MGMT_DATA SA parameter enables the SA to apply the same encryption and integrity algorithms that IKEv2-SCSI negotiated to future IKEv2-SCSI SECURITY PROTOCOL OUT commands and IKEv2-SCSI SECURITY PROTOCOL IN commands, if any.

5.13.4.10 Abandoning an IKEv2-SCSI CCS

The occurrence of errors in either the application client or the device server may require that an IKEv2-SCSI CCS be abandoned.

A device server shall indicate that it has abandoned an IKEv2-SCSI CCS, if any, by terminating an IKEv2-SCSI CCS command (see 5.13.4.1) received on the I_T_L nexus for which the IKEv2-SCSI CCS state is being maintained with any combination of status and sense data other than those shown in table 84.

Table 84 — IKEv2-SCSI command terminations that do not abandon the CCS

IKEv2-SCSI CCS command	Status (Sense Key)	Additional Sense Code	Description
SECURITY PROTOCOL OUT SECURITY PROTOCOL IN	GOOD (n/a)	n/a	Indicates IKEv2-SCSI CCS is progressing as described in this standard
Key Exchange step SECURITY PROTOCOL OUT (see 5.13.4.6.2)	CHECK CONDITION (ABORTED COMMAND)	CONFLICTING SA CREATION REQUEST	At least one IKEv2-SCSI CCS is already active, and attempts to start another are blocked until the first CCS completes
SECURITY PROTOCOL OUT SECURITY PROTOCOL IN	CHECK CONDITION (NOT READY)	LOGICAL UNIT NOT READY, SA CREATION IN PROGRESS	Device server is busy processing another command in the IKEv2-SCSI CCS associated with this I_T_L nexus or a different IKEv2-SCSI CCS associated with a different I_T_L nexus
SECURITY PROTOCOL IN	CHECK CONDITION (ILLEGAL REQUEST)	INVALID FIELD IN CDB	Incorrect SECURITY PROTOCOL IN CDB format
Authentication step SECURITY PROTOCOL OUT (see 5.13.4.7.2)		UNABLE TO DECRYPT PARAMETER LIST	Device server is unable to decrypt the Encrypted payload (see 7.7.3.5.11) or the integrity check fails
SECURITY PROTOCOL OUT		SA CREATION PARAMETER VALUE REJECTED	To adapt to possible denial of service attacks, a condition for which the optimal response includes an additional sense code of SA CREATION PARAMETER VALUE INVALID and the abandonment of the CCS is not causing the CCS to be abandoned

As part of abandoning an IKEv2-SCSI CCS, the device server shall:

a) discard all maintained state (see 5.13.4.1); and
b) prepare to allow a future Key Exchange step SECURITY PROTOCOL OUT command received on any I_T_L nexus to start a new IKEv2-SCSI CCS.

After a device server abandons an IKEv2-SCSI CCS, the device server shall respond to all new IKEv2-SCSI protocol commands as if an IKEv2-SCSI CCS had never been started.

An application client should not abandon an IKEv2-SCSI CCS when the next command in the CCS is a SECURITY PROTOCOL IN command. Instead, the application client should send the appropriate SECURITY PROTOCOL IN command and then abandon the IKEv2-SCSI CCS.

An application client should specify that it has abandoned an IKEv2-SCSI CCS by sending an IKEv2-SCSI Delete operation (see 5.13.4.11) with application client SAI and device server SAI information that matches that of the IKEv2-SCSI CCS being abandoned.

5.13.4.11 Deleting an IKEv2-SCSI SA

As part of deleting an SA, both sets of SA parameters (see 5.13.2.2) are deleted as follows:

1) the application client uses the information in its SA parameters to prepare an IKEv2-SCSI Delete operation that requests deletion of the device server's SA parameters;
2) the application client deletes its SA parameters and any associated data;
3) the application client sends the IKEv2-SCSI Delete operation prepared in step 1) to the device server;
4) in response to the IKEv2-SCSI Delete operation, the device server deletes its SA parameters and any associated data.

The IKEv2-SCSI Delete operation is a SECURITY PROTOCOL OUT command with the SECURITY PROTOCOL field set to IKEv2-SCSI (i.e., 41h) and the SECURITY PROTOCOL SPECIFIC field set to 0104h. The parameter data consists of the IKEv2-SCSI header (see 7.7.3.4) and an Encrypted payload (see 7.7.3.5.11) that contains the one Delete payload (see 7.7.3.5.10).

The Delete payload should conform to the requirements described in 7.7.3.5.10.

The device server shall process a valid Delete operation SECURITY PROTOCOL OUT command regardless of whether or not IKEv2-SCSI CCS state is being maintained for the I_T_L nexus on which the command is received.

The application client SAI and device server SAI information in a valid Delete operation may or may not identify an IKEv2-SCSI CCS for which state is being maintained for the I_T_L nexus on which the command is received (see 7.7.3.5.11).

5.13.5 Security progress indication

The cryptographic calculations required by some security protocols are capable of consuming significant amounts of time in the device server. While cryptographic security calculations are in progress, the device server shall provide pollable REQUEST SENSE data (see 5.10.2) with:

a) the sense key set to NOT READY;
b) the additional sense code set to LOGICAL UNIT NOT READY, SA CREATION IN PROGRESS; and
c) the PROGRESS INDICATION field set to indicate the progress of the device server in performing the necessary cryptographic calculations.

The device server shall not use the progress indication to report the detailed progress of cryptographic computations that take a variable amount of time based on their inputs. The device server may use the progress indication to report synthetic progress that does not reveal the detailed progress of the computation (e.g., divide a constant expected time for the computation by 10 and advance the progress indication in 10% increments based on the elapsed time as compared to the expected time).

The requirements in this subclause apply to implementations of Diffie-Hellman computations and operations involving any keys (e.g., RSA) that optimize operations on large numbers based on the values of inputs (e.g.,

a computational step may be skipped if a bit or set of bits in an input is zero). A progress indication that advances based on the computation structure (e.g., count of computational steps) may reveal the time taken by content-dependent portions of the computation, and reveal information about the inputs.

5.13.6 Command security

5.13.6.1 Overview

SCSI command security defines techniques for protecting against inadvertent or malicious misuse of SCSI commands to gain unauthorized access to logical units.

The following classes are used to specify SCSI command security:

a) Secure CDB Originator class;
b) Security Manager class;
c) Enforcement Manager class; and
d) Secure CDB Processor class.

The relationship between those classes varies depending on the implemented security technique.

5.13.6.2 Secure CDB Originator class

The Secure CDB Originator class is a kind of application client that originates SCSI commands to which it has attached a security CDB extension (see 4.2.4) that allows an enforcement manager to determine if the SCSI command may be processed by the addressed logical unit.

The secure CDB originator interacts with the security manager to determine:

a) the types of the SCSI commands it is allowed to send to the Secure CDB processor; and
b) the content of the security CDB extension to be attached to the SCSI commands.

5.13.6.3 Secure CDB Processor class

The Secure CDB Processor class is a kind of device server that processes SCSI commands that have an attached security extension, if an enforcement manager allows that type of SCSI command from the originating application client to be processed.

The secure CDB processor determines if a SCSI command is allowed to be processed by communicating the following information to the enforcement manager:

a) the CDB of the SCSI command to be processed; and
b) the security CDB extension, if any, attached to the SCSI command to be processed.

The secure CDB processor shall always allow the processing of the following commands when they do not have an attached security CDB extension:

a) INQUIRY;
b) REPORT LUNS;
c) REPORT SUPPORTED OPERATION CODES;
d) REPORT SUPPORTED TASK MANAGEMENT FUNCTIONS;
e) REPORT TARGET PORT GROUPS; and
f) REQUEST SENSE.

5.13.6.4 Enforcement Manager class

The Enforcement Manager class is either contained within a:

a) device server (i.e., has the same LUN as the secure CDB processor); or
b) target device (e.g., has a W_LUN, or vendor specific presence in the SCSI target device).

The enforcement manager determines if the secure CDB processor is allowed to, or prevented from, processing a SCSI command using security information received from the security manager.

If a secure CDB originator has not been authenticated, the enforcement manager shall not make determinations about whether a command from that secure CDB originator is allowed or prevented.

5.13.6.5 Security Manager class

The Security Manager class communicates with the Secure CDB Originator class (see 5.13.6.2) and the Enforcement Manager class (see 5.13.6.4) as shown in table 85.

Table 85 — Security Manager class relationships

Security Manager location	Communications mechanism for …	
	Secure CDB Originator	Enforcement Manager
An application client located in the same SCSI device as the secure CDB originator	Outside the scope of this standard	Via the SCSI domain's service delivery subsystem (see SAM-5)
A device server located in the same SCSI device as the secure CDB processor	Via the SCSI domain's service delivery subsystem (see SAM-5)	Outside the scope of this standard
A SCSI device contained within the same SCSI domain as the secure CDB originator and the secure CDB processor [a]	Via the SCSI domain's service delivery subsystem (see SAM-5)	Via the SCSI domain's service delivery subsystem (see SAM-5)
Not a SCSI device, device server, or application client	Outside the scope of this standard	Outside the scope of this standard

[a] This SCSI device is required to contain an application client and a device server (i.e., to contain both a SCSI Initiator Port class (see SAM-5) and a SCSI Target Port class (see SAM-5)).

The security manager:

a) maintains SCSI command security information for the SCSI domain (e.g., authorization and authentication information);
b) delivers to the enforcement manager the security information required by the enforcement manager to determine if the secure CDB processor is allowed to, or prevented from, processing a SCSI command; and
c) responds to requests from authenticated secure CDB originators to send SCSI commands to a secure CDB processor as follows:
   A) if the secure CDB originator sends its authentication and an authorization request, then the security manager responds with the authorization information necessary for the secure CDB originator to generate security information to be attached to any authorized CDB that is sent to the secure CDB processor; or
   B) if the secure CDB originator sends its authentication, an authorization request, and the security information to be attached to CDBs, then the security manager shall only accept the request if the

secure CDB originator is authorized to send the requested SCSI commands to the requested secure CDB processor.

5.13.6.6 The relationship between SAs and command security

As defined by this standard, SAs provide the following forms of secure communications for selected portions of selected CDB and command parameter data:

a) cryptographic data integrity provided by Message Authentication Code or Integrity Check Value; and
b) confidentially provided by data encryption.

The SAs defined by this standard do not apply to data communicated in the CDBs sent from the secure CDB originator to the secure CDB processor. The function of securing CDBs is performed by the command security features described in 5.13.6. Command security and SAs features may be used in concert to protect both the CDB data and the parameter data.

Authorization information (see 5.13.6.5) includes associations between:

a) permissions to use certain commands and command options; and
b) secure CDB originator identities.

SAs provide a mechanism for satisfying the secure CDB originator authentication requirements placed on enforcement managers (see 5.13.6.4). Every secure CDB originator that is authenticated using an SA is required to be authenticated using a unique SA.

Some command security techniques (e.g., the CbCS technique described in 5.13.6.7) depend on an established secure channel between a secure CDB originator and secure CDB processor. SAs provide a mechanism for establishing such secure channels. If SAs are used in this manner, multiple secure CDB originators may share a common SA; however, such SAs do not authenticate any specific secure CDB originator.

5.13.6.7 Capability-based command security technique

5.13.6.7.1 Overview

CbCS is a credential-based system that manages access to a logical unit by the coordination between shared keys and security parameters set by the CbCS management application client (see 5.13.6.7.4) and credentials generated by the CbCS management device server (see 5.13.6.7.3). The mechanism for coordination between the CbCS management device server and the CbCS management application client is not defined in this standard.

The CbCS protocol enables centralized management of SCSI command security.

CbCS secures access to a logical unit or a volume (see SSC-4) by providing cryptographic integrity of credentials that are added to commands sent to the logical unit. This cryptographic integrity is based on mutual trust and key exchanges between the CbCS management device server, CbCS management application client, and the enforcement manager.

Different levels of protection and security are achieved by using different CbCS methods. The following CbCS methods are defined by this standard:

a) the BASIC CbCS method (see 5.13.6.7.8.2) provides protection against errors but does not prevent unauthorized access caused by means of malicious attacks (e.g., identity spoofing and network attacks); and
b) the CAPKEY CbCS method (see 5.13.6.7.8.3) enforces application client authentication and provides cryptographic integrity of credentials. It protects against the following types of unauthorized access attacks:

A) Without cryptographic message integrity in the service delivery subsystem:
   a) illegal use of credentials beyond their original scope and life span;
   b) forging or stealing credentials; and
   c) using malformed credentials;
   and
B) with cryptographic message integrity in the service delivery subsystem:
   a) network errors and malicious message modifications; and
   b) message replay attacks.

CbCS also supports rapid revocation of credentials, per SCSI target device and per logical unit.

CbCS does not define task management function security.

CbCS (see figure 15) is composed of the following classes:

a) Security Manager class (see 5.13.6.7.2) that contains the following classes:
   A) CbCS Management Device Server class (see 5.13.6.7.3); and
   B) CbCS Management Application Client class (see 5.13.6.7.4);
b) SCSI Initiator Device class (see SAM-5) that contains the following class:
   A) Secure CDB Originator class (see 5.13.6.7.5);
   and
c) SCSI Target Device class (see SAM-5) that contains the following classes:
   A) Secure CDB Processor class (see 5.13.6.7.6); and
   B) Enforcement Manager class (see 5.13.6.7.7).

Figure 15 shows the flow of transactions between the components of a CbCS capable SCSI domain.

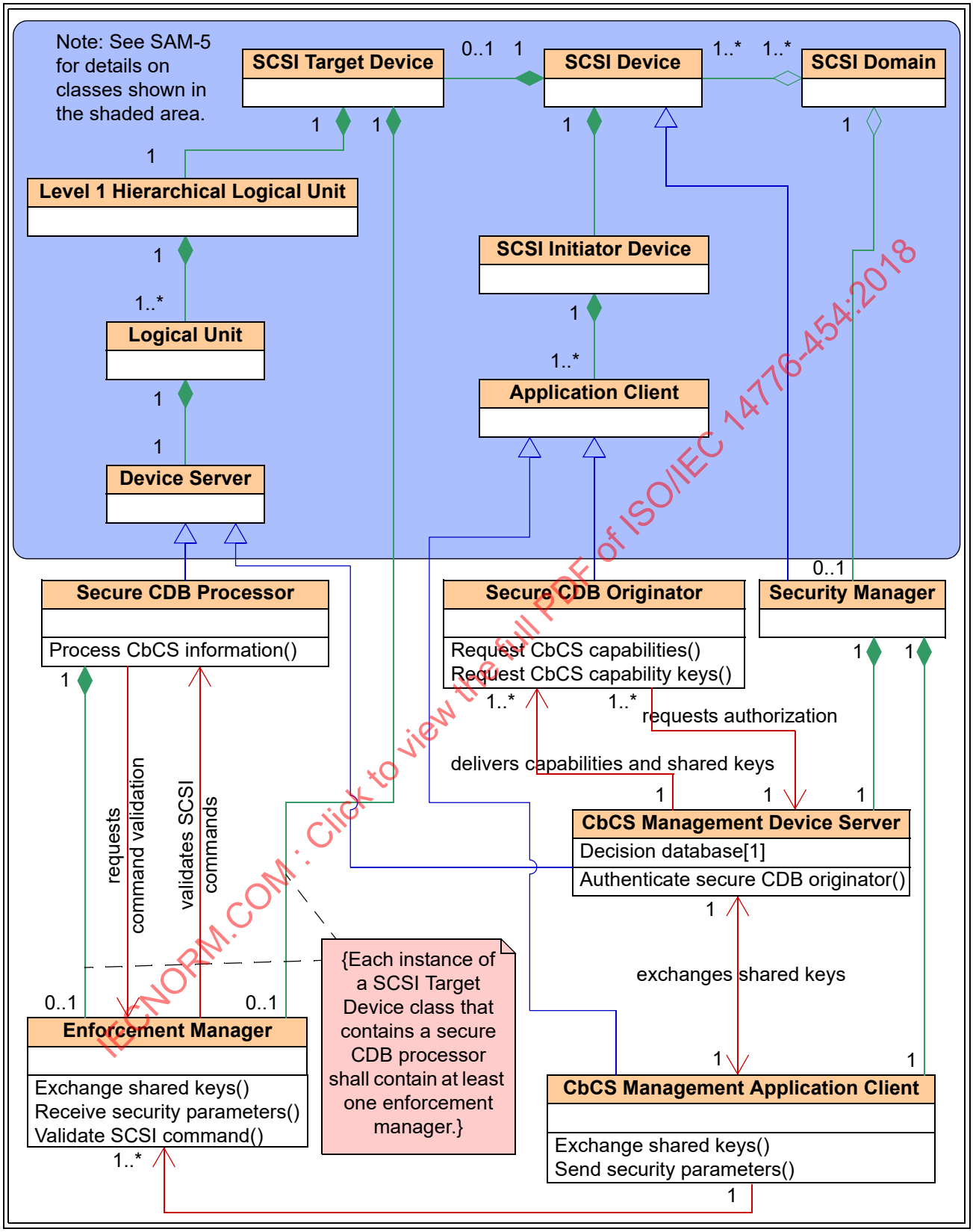


Figure 15 — CbCS overview class diagram

Each instance of CbCS shall contain:

a) one security manager that shall contain:
   A) one CbCS management device server; and
   B) one CbCS management application;
b) one or more SCSI initiator devices that shall contain:
   A) one or more secure CDB originators;
   and
c) one or more SCSI target devices that shall contain:
   A) one secure CDB processor per logical unit; and
   B) the following:
      a) one enforcement manager per secure CDB processor;
      b) one enforcement manager per SCSI target device; or
      c) both.

5.13.6.7.2 Security Manager class

The Security Manager class for the CbCS technique manages access of secure CDB originators to logical units or volumes (see SSC-4). The security manager uses a decision database to obtain the authorization information required for deciding the type and duration of access granted to secure CDB originator to a given logical unit or volume (see SSC-4). The security manager communicates with secure CDB originators to provide them CbCS credentials (see 5.13.6.7.12), and with enforcement managers to exchange shared keys (see 5.13.6.7.11) and send CbCS parameters (see 5.13.6.7.15).

The security manager may be located and may communicate with secure CDB originators and enforcement managers as follows:

a) if the security manager is a SCSI device contained within the same SCSI domain as the secure CDB originator and the enforcement manager, then the security manager shall contain an application client and use the application client to communicate to the enforcement manager, and the security manager shall contain a device server and use the device server to communicate with secure CDB originators;
b) if the security manager is an application client located in the same device as the secure CDB originators, the security manager shall communicate to the enforcement manager via the SCSI domain's service delivery subsystem, and the security manager may communicate with the secure CDB originators by means outside the scope of this standard; and
c) if the security manager is a device server located in the same device as the secure CDB processor, the security manager shall communicate to the secure CDB originators via the SCSI domain's service delivery subsystem, and the security manager may communicate with the enforcement manager by means outside the scope of this standard.

The security manager's device server is called the CbCS management device server (see 5.13.6.7.3). The security manager's application client is called the CbCS management application client (see 5.13.6.7.4).

If the security manager is a SCSI device, the security manager shall perform CbCS management using the CbCS management device server and the CbCS management application client as follows:

a) the CbCS management device server (see 5.13.6.7.3) provides access policy controls to secure CDB originators using policy-coordinated CbCS capabilities; and
b) the CbCS management application client (see 5.13.6.7.4) prevents unsecured access to a logical unit or a volume (see SSC-4) in concert with:
   A) the CbCS management device server (see 5.13.6.7.3);
   B) the enforcement manager; and
   C) the secure CDB processor.

CbCS management is confined to the CbCS management application client and CbCS management device server. The communication of CbCS management information may occur in a manner outside the scope of this standard.

5.13.6.7.3 CbCS Management Device Server class

5.13.6.7.3.1 CbCS Management Device Server class overview

The CbCS Management Device Server class returns a CbCS capability and a CbCS capability key (i.e., Capability-Key) with each CbCS credential giving the secure CDB originator access to:

a) a specific logical unit; or
b) if requested, a specific volume mounted in a specific logical unit (see SSC-4).

This standard defines the RECEIVE CREDENTIAL command (see 6.26) that the secure CDB originator may use to request a CbCS capability and a CbCS capability key from a CbCS management device server.

Commands to and responses from the CbCS management device server are protected by use of an SA whose creation included the Authentication Step (see 6.26).

The CbCS management device server shall set the CBCS bit to zero in any Extended INQUIRY Data VPD page (see 7.8.7) that it returns.

5.13.6.7.3.2 Decision Database attribute

The Decision Database attribute is used to obtain the authorization information required for deciding the type and duration of access granted to a secure CDB originator for a given logical unit or volume (see SSC-4) within a SCSI target device. CbCS Credentials are prepared by the CbCS management device server based on the contents of that Decision Database attribute.

5.13.6.7.4 CbCS Management Application Client class

The CbCS Management Application Client class exchanges shared keys (see 5.13.6.7.11) with and sends CbCS parameters (see 5.13.6.7.15) to the enforcement manager using SECURITY PROTOCOL OUT commands (see 6.41) and SECURITY PROTOCOL IN commands (see 6.40) transferred over the SCSI domain's service delivery subsystem.

The CbCS capability keys are computed by the CbCS management device server using shared keys that are shared between:

a) the enforcement manager;
b) the CbCS management application client; and
c) the CbCS management device server.

The shared keys are managed by the security manager.

5.13.6.7.5 Secure CDB Originator class

The Secure CDB Originator class requests CbCS capabilities and CbCS capability keys from the CbCS management device server for a specific logical unit or volume (see SSC-4). The secure CDB originator sends the CbCS capability and integrity check value to the logical unit's secure CDB processor as part of a CbCS extended CDB as described in 5.13.6.7.16.

For more information on the Secure CDB Originator class, see 5.13.6.2.

5.13.6.7.6 Secure CDB Processor class

The Secure CDB Processor class:

a) receives a CbCS capability descriptor (see 6.26.2.3) from a secure CDB originator in a CbCS extension descriptor (see 5.13.6.7.16);

b) requests the SCSI command be validated by the enforcement manager; and
c) if the Enforcement Manager validates the SCSI command, then the secure CDB processor processes that SCSI command.

The secure CDB processor indicates that CbCS is applied to a logical unit by setting the CBCS bit to one in the Extended INQUIRY Data VPD page (see 7.8.7). If the CBCS bit is set to one, the logical unit shall support the following:

a) Extended CDBs (see 4.2.4);
b) CbCS extension type (see 5.13.6.7.16);
c) SECURITY PROTOCOL IN commands (see 6.40) specifying the CbCS security protocol (see 7.7.4); and
d) SECURITY PROTOCOL OUT commands (see 6.41) specifying the CbCS security protocol (see 7.7.4).

For more information on the Secure CDB Processor class see 5.13.6.3.

5.13.6.7.7 Enforcement Manager class

The Enforcement Manager class:

a) receives shared keys (see 5.13.6.7.11) and CbCS parameters (see 5.13.6.7.15) from the CbCS management application client;
b) authenticates the CbCS capability received from a secure CDB processor with an integrity check value as described in 5.13.6.7.13.2;
c) validates SCSI commands sent by secure CDB originators as described in 5.13.6.7.13.2; and
d) if the CAPKEY CbCS method (see 5.13.6.7.8.3) is supported, supplies one security token (see 5.13.6.7.10) for each active I_T nexus to the secure CDB processor for delivery to the secure CDB originator that is using that I_T nexus.

The enforcement manager may be contained within the secure CDB processor or within the SCSI target device. If the enforcement manager is contained within the secure CDB processor, the shared keys and CbCS parameters the enforcement manager uses pertain to that logical unit. If the enforcement manager is contained within the SCSI target device, the shared keys and CbCS parameters the enforcement manager uses pertain to the SCSI target device, and the SECURITY PROTOCOL well-known logical unit (see 8.5) is used for the commands to exchange shared keys and set CbCS parameters.

If a shared key is stored in a well known logical unit then that key is shared between all logical units within the SCSI target device but shall only used by a logical unit if there has been no shared key assigned to that logical unit (i.e., a shared key assigned to a logical unit always overrides any shared key assigned to a well known logical unit).

For more information on the Enforcement Manager class see 5.13.6.4.

5.13.6.7.8 CbCS methods

5.13.6.7.8.1 Overview

The CbCS methods defined by this standard are summarized in table 86.

Table 86 — CbCS methods

CbCS method	Protection provided by the enforcement manager and secure CDB processor	Level of security provided, including I_T nexus security provided by SCSI transport (e.g., FC-SP-2)	
		Without I_T nexus security	With I_T nexus security
BASIC	Protection against errors provided by verifying that the CbCS capability allows processing, but no validation of the authenticity of the CbCS capability.	No protection against attacks	Same protection as is provided by the SCSI transport on the I_T nexus
CAPKEY	Protection against errors and some attacks by both verifying that the CbCS capability allows processing, and validating the authenticity of the CbCS capability.	CbCS capability authenticity assured, but still subject to network attacks (e.g., replay attacks)	CbCS capability authenticity assured and bound to an I_T nexus; other network attacks (e.g., data privacy) thwarted by SCSI transport security on the I_T nexus

If a secure CDB processor receives a command for a logical unit that has CbCS enabled, the enforcement manager shall validate the command as described in 5.13.6.7.13.2 before any other field in the CDB is validated, including the operation code.

5.13.6.7.8.2 The BASIC CbCS method

The BASIC CbCS method validates that the CbCS capability authorizes the encapsulated command for each CDB. This method provides centrally-managed policy-driven command access control mechanism that enforces authorized access based on capabilities.

The BASIC CbCS method does not validate the authenticity of the CbCS capability.

Preparing CbCS credentials for the BASIC CbCS method does not require the knowledge of CbCS shared keys and may be done by the secure CDB originator without coordination with the CbCS management device server. In the CbCS extension descriptor (see 5.13.6.7.16):

a) the CbCS capability descriptor (see 6.26.2.3) CBCS METHOD field is set to BASIC;
b) the following CbCS capability descriptor fields are ignored:
   A) the KEY VERSION field; and
   B) the INTEGRITY CHECK VALUE ALGORITHM field;
   and
c) the INTEGRITY CHECK VALUE field is set to zero.

The BASIC CbCS method controls access between the secure CDB originator and the secure CDB processor without requiring authentication of the secure CDB originator. This method is sufficient for the secure CDB processor to request that the enforcement manager verify the CbCS capabilities sent by the secure CDB originator.

5.13.6.7.8.3 The CAPKEY CbCS method

The CAPKEY CbCS method provides centrally-managed policy-driven command access control mechanism that enforces authorized access based on capabilities.

In addition, the CAPKEY CbCS method assures the integrity and authenticity of the CbCS capability transferred with each command.

The CAPKEY CbCS method provides for security of commands delivered to the secure CDB processor. When used in conjunction with a secure service delivery subsystem, the method provides additional protection against network attacks (see 5.13.6.7.8.1).

Each capability (see 5.13.6.7.13) is cryptographically associated with a capability key, and the pair is returned to a secure CDB originator (see 5.13.6.7.5) in a credential (see 5.13.6.7.12) in response to a RECEIVE CREDENTIAL command (see 6.26).

If the capability is associated with a specific command using a CbCS extension descriptor (see 5.13.6.7.16), then an integrity check value is computed using the capability key and a CbCS security token (see 5.13.6.7.10). The enforcement manager (see 5.13.6.7.7) validates this integrity check value as described in 5.13.6.7.13.3.

5.13.6.7.9 CbCS trust assumptions

After the logical unit is trusted (i.e., after a secure CDB originator authenticates that it is communicating with a specific logical unit), the secure CDB originator trusts the secure CDB processor and the enforcement manager to do the following:

a) deny any access attempt from any application client that is not authorized by the security manager; and
b) deny access from any application client that does not perform the CbCS protocols and functions defined by this standard.

The CbCS management device server and the CbCS management application client are trusted after:

a) the CbCS management device server is authenticated by the secure CDB originator; and
b) the CbCS management application client is authenticated by the CbCS Enforcement Manager.

The CbCS management device server and the CbCS management application client are trusted to do the following:

a) securely store long-lived shared keys and capability keys;
b) grant credentials to secure CDB originators according to access control policies that are outside the scope of this standard; and
c) perform the defined security functions.

The service delivery subsystem between the secure CDB originator and the secure CDB processor is not trusted. However, the CbCS security model for CbCS methods other than BASIC is defined so that commands generated by the secure CDB originator are processed by the secure CDB processor only after the secure CDB originator interacts with both the CbCS management device server and the secure CDB processor as defined in this standard.

The communications trust requirements shown in table 87 provide a basis for the CbCS trust assumptions.

Table 87 — CbCS communications trust requirement

For connections between	CbCS cryptographic communications trust requirements	Requirement level
secure CDB originator and secure CDB processor	Message integrity [b]	Optional [a]
secure CDB originator and CbCS management application client	Message confidentiality [c] and integrity [b]	Mandatory
CbCS management application client and enforcement manager	Message integrity [b]	Mandatory
CbCS management application client and CbCS management device server	Message confidentiality [c] and integrity [b]	Mandatory

[a] If this requirement is not met, then the conditions that table 86 (see 5.13.6.7.8) show for an I_T nexus without security apply.
[b] Message integrity algorithms are those that table 105 (see 5.13.8) describes as integrity checking (i.e., AUTH) algorithms.
[c] Message confidentiality algorithms are those that table 105 (see 5.13.8) describes as encryption algorithms.

5.13.6.7.10 CbCS security tokens

A CbCS security token is a random nonce that is at least eight bytes in length that is chosen by the enforcement manager (see 5.13.6.7.7) and returned to a secure CDB originator (see 5.13.6.7.5) by the secure CDB processor (see 5.13.6.7.6) in response to a SECURITY PROTOCOL IN command (see 6.40) with the SECURITY PROTOCOL field set to 07h (i.e., CbCS) and the SECURITY PROTOCOL SPECIFIC field set to 3Fh (i.e., the Security Token CbCS page) as described in 7.7.4.3.4.

The security token shall be unique to each instance of an I_T nexus known to the secure CDB processor and enforcement manager. Security tokens shall be reset and maintained as described in this subclause.

Each security token shall contain at least as many bytes as the largest cipher block size for all the integrity checking algorithms supported by the SCSI target device (see 7.7.4.3.3).

If a hard reset, logical unit reset, or I_T nexus loss is detected by the secure CDB processor, the enforcement manager shall be notified of the event once for each affected I_T nexus. In response to such a notification the enforcement manager shall discard the security token, if any, associated with the affected I_T nexus.

After a power on, the enforcement manager shall discard all security tokens, if any, that it had been maintaining before the power on.

In response to a request for a security token for a given I_T nexus from the secure CDB processor, the enforcement manager shall do one of the following:

a) return the security token value, if any, that is being maintained for the specified I_T nexus to the secure CDB processor; or
b) if no security token is being maintained for the specified I_T nexus, then:
   1) a security token shall be prepared;
   2) the security token shall be returned to the secure CDB processor; and
   3) the security token shall be maintained in association with the specified I_T nexus until one of the events described in this subclause causes it to be discarded.

5.13.6.7.11 CbCS shared keys

5.13.6.7.11.1 Overview

Cryptographic integrity checking for CbCS capabilities depends on the following hierarchy of shared keys that is specific to the CbCS model:

1) a master key that is composed of:
   A) an authentication key; and
   B) a generation key;
   and
2) up to 16 working keys whose values are generated from the generation key component of the master key.

Each CbCS shared key set shall support:

a) one master key; and
b) at least two working keys.

Each CbCS shared key in a set has an associated identifier (see 5.13.6.7.11.2) that describes the CbCS shared key to the objects that are managing it without revealing the CbCS shared key's value.

Coordinated sets of CbCS shared keys that conform to this hierarchy are maintained by:

a) the CbCS management application client (see 5.13.6.7.4);
b) the CbCS management device server (see 5.13.6.7.3); and
c) the enforcement manager (see 5.13.6.7.7).

The mechanism for coordinating CbCS shared key sets between the CbCS management application client and the CbCS management device server is outside the scope of this standard.

The following security protocols are provided by this standard for coordinating CbCS shared key sets between the CbCS management application client and the enforcement manager:

a) a Diffie-Hellman key exchange protocol for changing all the components of the master key (see 5.13.6.7.11.4); and
b) mechanisms that invalidate a working key or set a working key based on the generation key component of the current master key (see 5.13.6.7.11.5).

These key management protocols associate a key identifier with each shared key (i.e., master key or working key). The CbCS management application client may use these key identifiers to describe the shared key in some way (e.g., when the shared key was last refreshed or the intended use of the shared key). Key identifiers shall not be used to contain shared key values.

Within any SCSI target device that contains an enforcement manager, sets of CbCS shared keys are maintained as follows:

a) a separate set of per-logical unit CbCS shared keys for a logical unit that has CbCS enabled;
b) one target-wide set of CbCS shared keys that are accessible to all logical units that have CbCS enabled; or
c) both a target-wide set of CbCS shared keys and per-logical unit sets of CbCS shared keys.

The enforcement manager in any logical unit that has CbCS enabled shall have access to at least one set of CbCS shared keys.

If an enforcement manager has access to both a target-wide set of CbCS shared keys and a per-logical unit set of CbCS shared keys, then a working key defined in the per-logical unit set of CbCS shared keys, if any, shall be used instead of the equivalent working key from the target-wide set of CbCS shared keys.

All CbCS shared keys should be retired from active use (i.e., set, changed, or discarded) often enough to thwart key attacks. How often to retire CbCS shared keys from active use is outside the scope of this standard.

The shared keys in a CbCS shared key set and the CbCS pages used to manage them between the CbCS management application client and the enforcement manager are summarized in table 88.

Table 88 — Summary of CbCS shared keys

CbCS shared key	Applicable CbCS page	Data direction	Capability protection	Reference
Master Authentication, and Generation	Current CbCS Parameters [a]	In	Working key	7.7.4.3.5
	Set Master Key – Seed Exchange Set Master Key – Seed Exchange Set Master Key – Change Master Key [b]	Out In Out	Authentication key	7.7.4.5.5 7.7.4.3.6 7.7.4.5.6
Working key	Current CbCS Parameters [a]	In	Working key	7.7.4.3.5
	Invalidate Key Set Key	Out	Authentication key	7.7.4.5.3 7.7.4.5.4

[a] Only key identifiers are returned, not shared key values.
[b] The new authentication key computed during the seed exchange is the authentication key used for this CbCS page.

5.13.6.7.11.2 CbCS shared key identifiers

CbCS shared key identifiers (see table 89) are set by the CbCS pages (see table 88 in 5.13.6.7.11.1) that change the values of a CbCS shared key and reported by the Current CbCS Parameters CbCS page (see 7.7.4.3.5).

Table 89 — CbCS shared key identifier values

Value	Description
0000 0000 0000 0000h [a]	The associated CbCS shared key has not been modified since the SCSI target device was manufactured
0000 0000 0000 0001h to FFFF FFFF FFFF FFFDh	The associated CbCS shared key has a valid value that has been set by the applicable CbCS page
FFFF FFFF FFFF FFFEh [a]	The associated CbCS shared key does not have a valid value
FFFF FFFF FFFF FFFFh [a]	The associated CbCS shared key is not supported

[a] The use of this value in the CbCS pages that change CbCS shared key values is reserved.

5.13.6.7.11.3 Specifying which CbCS shared key to change

The logical unit to which a SECURITY PROTOCOL IN command (see 6.40) or a SECURITY PROTOCOL OUT command (see 6.41) that specifies one of the CbCS pages shown in table 88 (see 5.13.6.7.11.1) is addressed determines which CbCS shared key is modified as follows:

a) if the command is addressed to the SECURITY PROTOCOL well known logical unit (see 8.5), then the CbCS the target-wide (see 5.13.6.7.11.1) set of CbCS shared keys is modified; or

b) if the command is addressed to a logical unit that is not a well known logical unit, then the per-logical unit (see 5.13.6.7.11.1) set of CbCS shared keys for the specified logical unit are modified.

5.13.6.7.11.4 Updating a CbCS master key

Both components of the CbCS master key (i.e., the authentication key and the generation key) are changed using a single Diffie-Hellman exchange CCS as summarized in this subclause. Use of the Diffie-Hellman exchange ensures forward secrecy of the master key.

The CbCS master key update CCS is composed of the following commands:

1) a SECURITY PROTOCOL OUT command processing the Set Master Key – Seed Exchange CbCS page (see 7.7.4.5.5);
2) a SECURITY PROTOCOL IN command processing the Set Master Key – Seed Exchange CbCS page (see 7.7.4.3.6); and
3) a SECURITY PROTOCOL OUT command processing the Set Master Key – Change Master Key CbCS page (see 7.7.4.5.6).

NOTE 21 - The capability key used to access the two Set Master Key – Seed Exchange CbCS pages is different from the capability key used to access the Set Master Key – Change Master Key CbCS page (see 5.13.6.7.12.3).

The device server shall maintain CCS state for only one CbCS master key update CCS for all I_T nexuses. The device server shall terminate the command with CHECK CONDITION status, with the sense key set to ILLEGAL REQUEST, and the additional sense code set to COMMAND SEQUENCE ERROR, if any of the following conditions occur:

a) a command attempts to start a new CbCS master key update CCS while state is being maintained for another; or
b) a sequence of CbCS master key update CCS commands other than the one shown in this subclause is attempted.

The device server shall discard the CbCS master key update CCS state if any of the following occur:

a) a command in the CCS does not complete with GOOD status; or
b) the entire CbCS master key update CCS command sequence shown in this subclause is not completed within ten seconds of the successful completion of processing for the SECURITY PROTOCOL OUT command for the Set Master Key – Seed Exchange CbCS page.

5.13.6.7.11.5 Changing a CbCS working keys

A working key is invalidated using the Invalidate Key CbCS page (see 7.7.4.5.3) and set to a new value using the Set Key CbCS page (see 7.7.4.5.4).

New working keys are computed based on the applicable master key and a random number seed.

Working keys are tracked using CbCS shared key identifiers (see 5.13.6.7.11.2).

5.13.6.7.12 CbCS credentials

5.13.6.7.12.1 Overview

Each CbCS credential authorizes access to:

a) a logical unit; or
b) a specific volume (see SSC-4) mounted in a specific logical unit.

The applicability of CbCS credentials to the BASIC CbCS method (see 5.13.6.7.8.2) is outside the scope of this standard.

The format of a CbCS credential is described in 6.26.2.2.

The primary components of a CbCS credential are as follows:

a) a CbCS capability whose format and preparation are described in 5.13.6.7.13; and
b) the CbCS capability key that is an integrity check value that is computed as follows:
   A) if the credential is to be sent to a secure CDB originator (see 5.13.6.7.5), the CbCS capability key is computed based on a working key as described in 5.13.6.7.12.2; or
   B) if the credential is being prepared for use by the CbCS management application client (see 5.13.6.7.4), then the CbCS capability key is computed based on a working key or the master key as described in 5.13.6.7.12.3.

Regardless of how it is computed, the CbCS capability key is used as follows:

a) by the secure CDB originator to prepare CbCS extension descriptors (see 5.13.6.7.16) sent to the secure CDB processor (see 5.13.6.7.6);
b) by the CbCS management application client to prepare CbCS extension descriptors sent to the enforcement manager (see 5.13.6.7.7); and
c) by the enforcement manager to validate the integrity of capabilities (see 5.13.6.7.13.3) in CbCS extension descriptors received by the secure CDB processor.

5.13.6.7.12.2 CbCS capability key computations for the secure CDB originator

For credentials sent to the secure CDB originator (see 5.13.6.7.5), the CbCS capability key (see 5.13.6.7.12.1) is computed without knowledge of the command for which the CbCS capability key is being prepared using the following inputs:

a) the integrity check value algorithm specified by the INTEGRITY CHECK VALUE ALGORITHM field (see 6.26.2.3) in the CbCS capability descriptor; and
b) the following inputs to this integrity check value algorithm:
   A) all the bytes in the CbCS capability descriptor (see 6.26.2.3); and
   B) the working key specified by the KEY VERSION field (see 6.26.2.3) in the CbCS capability descriptor.

5.13.6.7.12.3 CbCS capability key computations for general use

The computation of the CbCS capability key depends on the command for which the CbCS capability key is being computed as described in this subclause. This computation is more general than the computation described in 5.13.6.7.12.2, but the computation produces the same results because the secure CDB originator should not be allowed to use the commands that produce the exceptional cases described in this subclause.

The CbCS capability key computations described in this subclause are used:

a) for credentials that are to be used by the CbCS management application client (see 5.13.6.7.4); and
b) by the enforcement manager (see 5.13.6.7.7) when validating a CbCS capability descriptor (see 5.13.6.7.13).

When the enforcement manager is validating a CbCS capability descriptor, the command is determined by inspecting the CDB that is being processed.

Based on the command associated with the credential, the CbCS capability key (see 5.13.6.7.12.1) is computed using the following inputs:

a) the integrity check value algorithm specified by the INTEGRITY CHECK VALUE ALGORITHM field (see 6.26.2.3) in the CbCS capability descriptor; and
b) the following inputs to this integrity check value algorithm:
   A) all the bytes in the CbCS capability descriptor (see 6.26.2.3); and
   B) the following CbCS shared key:
      a) if the command is a SECURITY PROTOCOL IN command (see 6.40) or a SECURITY PROTOCOL OUT command (see 6.41) with the SECURITY PROTOCOL field set to 07h (i.e., CbCS) and the SECURITY PROTOCOL SPECIFIC field set to a value that is greater than CFFFh, then the following shared key shall be used:
         A) if the command does not access the Set Master Key - Change Key CbCS page (see 7.7.4.5.6), then the authentication key component of the master key (see 5.13.6.7.11) is used; or
         B) if the command accesses the Set Master Key - Change Key CbCS page, then the authentication key component of the new master key (see 7.7.4.3.6) maintained in the CbCS master key update CCS (see 5.13.6.7.11.4) is used;

         or
      b) if the command is not one of those described in step b) B) a), then the working key specified by the KEY VERSION field (see 6.26.2.3) in the CbCS capability descriptor.

5.13.6.7.13 CbCS capability descriptors

5.13.6.7.13.1 Overview

CbCS capability descriptors are components of:

a) CbCS credentials (see 5.13.6.7.12); and
b) CbCS extension descriptors (see 5.13.6.7.16).

The format of a CbCS capability descriptor is described in 6.26.2.3.

CbCS capability descriptors contain:

a) information about what commands are allowed if the CbCS capability descriptor is associated with a specific CDB via a CbCS extension descriptor;
b) information that identifies a specific logical unit or a specific volume (see SSC-4) mounted in a specific logical unit to which the CbCS capability is bound;
c) a time limit on the validity of the CbCS capability descriptor;
d) information that the CbCS management application client (see 5.13.6.7.4) may use to invalidate one or more CbCS capability descriptors before the time limit expires; and
e) information that the enforcement manager uses to cryptographically validate the CbCS capability descriptor, if specified, as described in 5.13.6.7.13.

5.13.6.7.13.2 CbCS extension descriptor validation

The enforcement manager (see 5.13.6.7.7) shall validate the CbCS capability descriptor (see 6.26.2.3) included in the CbCS extension descriptor (see 5.13.6.7.16). If the validation fails, the enforcement manager shall interact with the secure CDB processor (see 5.13.6.7.6) in a way that causes the command containing the CbCS extension descriptor to be terminated with the sense key set to ILLEGAL REQUEST, and the additional sense code set to INVALID FIELD IN CDB.

The enforcement manager's validation of a CbCS extension descriptor shall fail if any of the following conditions occur:

1) a CbCS extension descriptor is not present on a command that table 90 (see 5.13.6.7.14) shows as requiring a CbCS capability;
2) the command is one that table 90 (see 5.13.6.7.14) shows as never being allowed if CbCS is enabled;
3) the CBCS METHOD field is set to a value that is less than the value in the minimum CbCS method CbCS parameter (see 5.13.6.7.15);
4) the CBCS METHOD field is set to a value that table 267 defines as reserved (see 6.26.2.3) or a value that the enforcement manager does not support (see 7.7.4.3.3);
5) if the CBCS METHOD field is set to CAPKEY and the integrity validation described in 5.13.6.7.13.3 fails;
6) the DESIGNATION TYPE field is set to a value that table 266 defines as reserved (see 6.26.2.3);
7) the DESIGNATION TYPE field is set to 1h (i.e., logical unit designation descriptor), and the contents of the DESIGNATION DESCRIPTOR field in which a logical unit name (see SAM-5) is indicated does not match the addressed logical unit;
8) the DESIGNATION TYPE field is set to 2h (MAM attribute descriptor) and either of the following are true:
   A) the ATTRIBUTE IDENTIFIER field in the DESIGNATION DESCRIPTOR field is set to any value other than 0401h (i.e., MEDIUM SERIAL NUMBER); or
   B) the DESIGNATION DESCRIPTOR field contents do not match the MAM attribute of the volume that is accessible via the addressed logical unit;
9) the CAPABILITY EXPIRATION TIME field is set to a non-zero value and the value in the CAPABILITY EXPIRATION TIME field is less than (i.e., prior to) the current time in the clock CbCS parameter (see 5.13.6.7.15);
10) the POLICY ACCESS TAG field is set to a non-zero value that does not match the policy access tag CbCS parameter (see 5.13.6.7.15); or
11) the command in the CDB field of the extended CDB (see 4.2.4) that contains the CbCS extension descriptor is not allowed by the PERMISSIONS BIT MASK field (see 5.13.6.7.14).

5.13.6.7.13.3 CAPKEY CbCS method capability integrity validation

If the CbCS method is CAPKEY, the enforcement manager's validation of a CbCS capability descriptor shall fail the integrity tests if any of the situations described in this subclause occur.

Before attempting to cryptographically validate the integrity of the CbCS capability descriptor, the enforcement manager shall fail the validation if any of the following conditions occur in the CbCS capability descriptor (see 6.26.2.3):

a) the KEY VERSION field specifies an invalid working key as follows:
   A) the command is not a SECURITY PROTOCOL IN command (see 6.40) or a SECURITY PROTOCOL OUT command (see 6.41) with the SECURITY PROTOCOL field set to 07h (i.e., CbCS) and the SECURITY PROTOCOL SPECIFIC field set to a value that is greater than CFFFh (i.e., the KEY VERSION field is ignored for these commands);
   B) the per-logical unit working key (see 5.13.6.7.11), if any, specified by the KEY VERSION field is invalid (see 5.13.6.7.11.2); and
   C) the target-wide working key (see 5.13.6.7.11), if any, specified by the KEY VERSION field is invalid (see 5.13.6.7.11.2);

   or
b) the INTEGRITY CHECK VALUE ALGORITHM field is set to a value that is:
   A) not one of those that table 105 (see 5.13.8) lists as being an integrity checking (i.e., AUTH) algorithm;
   B) is AUTH_COMBINED; or
   C) is a value that the enforcement manager does not support (see 7.7.4.3.3).

If no integrity checking configuration errors are found in the CbCS capability descriptor, the enforcement manager shall:

1) compute the CbCS capability key for the CbCS capability descriptor as described in 5.13.6.7.12.3; and
2) compute the expected contents of CbCS extension descriptor INTEGRITY CHECK VALUE field (see 5.13.6.7.16), using the following inputs:
   A) the integrity check value algorithm specified by the INTEGRITY CHECK VALUE ALGORITHM field (see 6.26.2.3) in the CbCS capability descriptor; and
   B) the following inputs to this integrity check value algorithm:
      a) all the bytes in the security token (see 5.13.6.7.10) for the I_T nexus on which the command was received as the string for which the integrity check value is to be computed; and
      b) the CbCS capability key computed in step 1) as the cryptographic key.

The enforcement manager shall fail the validation if the contents of CbCS extension descriptor INTEGRITY CHECK VALUE field do not match the computed expected contents of CbCS extension descriptor INTEGRITY CHECK VALUE field.

5.13.6.7.14 Association between commands and permission bits

The PERMISSIONS BIT MASK field in the CbCS capability (see 6.26.2.3) specifies which commands are allowed by the CbCS capability. When processing commands with the CbCS extension, the enforcement manager shall verify that the bits applicable to the encapsulated SCSI command are all set to one in the PERMISSIONS BIT MASK field before processing the command. The associations between commands and permission bits are defined in table 90 and table 91 for commands defined in this standard.

Table 90 — Associations between commands and permissions (part 1 of 3)

Command	PERMISSIONS BIT MASK bits [a]						
	DATA READ	DATA WRITE	PARM READ	PARM WRITE	SEC MGMT	RESRV	MGMT
ACCESS CONTROL IN	never allow [b]						
ACCESS CONTROL OUT	never allow [b]						
CHANGE ALIASES	always allow [c]						
COPY OPERATION ABORT	never allow [b]						
EXTENDED COPY(LID4)	never allow [b]						
EXTENDED COPY(LID1)	never allow [b]						
INQUIRY	always allow [c]						
LOG SELECT				1			
LOG SENSE			1				
MANAGEMENT PROTOCOL IN							1
MANAGEMENT PROTOCOL OUT							1

[a] The command in the CDB field of the extended CDB (see 4.2.4) that contains the CbCS extension descriptor shall be allowed only if all of the bits marked with a 1 in the row for that command are set in the PERMISSIONS BIT MASK field of the CbCS capability in the CbCS extension descriptor. The permissions bits represented by the empty cells in a row are ignored.

[b] If the CBCS bit is set to one in the Extended INQUIRY Data VPD page (see 7.8.7), this command shall never be allowed.

[c] This command shall always be allowed regardless of whether the CbCS extension descriptor is present and if the CbCS extension descriptor is present regardless of the value in the PERMISSIONS BIT MASK field.

Table 90 — Associations between commands and permissions (part 2 of 3)

Command	PERMISSIONS BIT MASK bits [a]						
	DATA READ	DATA WRITE	PARM READ	PARM WRITE	SEC MGMT	RESRV	MGMT
MODE SELECT(6)				1			
MODE SELECT(10)				1			
MODE SENSE(6)			1				
MODE SENSE(10)			1				
PERSISTENT RESERVE IN			1				
PERSISTENT RESERVE OUT						1	
READ ATTRIBUTE			1				
READ BUFFER					1		
READ MEDIA SERIAL NUMBER			1				
RECEIVE COPY DATA(LID4)	always allow [c]						
RECEIVE COPY DATA(LID1)	never allow [b]						
RECEIVE COPY OPERATING PARAMETERS	never allow [b]						
RECEIVE COPY FAILURE DETAILS(LID1)	never allow [b]						
RECEIVE COPY STATUS(LID4)	always allow [c]						
RECEIVE COPY STATUS(LID1)	never allow [b]						
RECEIVE ROD TOKEN INFORMATION	always allow [c]						
RECEIVE CREDENTIAL	always allow [c]						
RECEIVE DIAGNOSTIC RESULTS			1				
REMOVE I_T NEXUS	never allow [b]						
REPORT ALIASES	always allow [c]						
REPORT ALL ROD TOKENS	always allow [c]						
REPORT IDENTIFYING INFORMATION			1				
REPORT LUNS	always allow [c]						
REPORT PRIORITY			1				
REPORT SUPPORTED OPERATION CODES	always allow [c]						
REPORT SUPPORTED TASK MANAGEMENT FUNCTIONS	always allow [c]						
REPORT TARGET PORT GROUPS	always allow [c]						
REPORT TIMESTAMP			1				

[a] The command in the CDB field of the extended CDB (see 4.2.4) that contains the CbCS extension descriptor shall be allowed only if all of the bits marked with a 1 in the row for that command are set in the PERMISSIONS BIT MASK field of the CbCS capability in the CbCS extension descriptor. The permissions bits represented by the empty cells in a row are ignored.

[b] If the CBCS bit is set to one in the Extended INQUIRY Data VPD page (see 7.8.7), this command shall never be allowed.

[c] This command shall always be allowed regardless of whether the CbCS extension descriptor is present and if the CbCS extension descriptor is present regardless of the value in the PERMISSIONS BIT MASK field.

Table 90 — Associations between commands and permissions (part 3 of 3)

Command	PERMISSIONS BIT MASK bits [a]						
	DATA READ	DATA WRITE	PARM READ	PARM WRITE	SEC MGMT	RESRV	MGMT
REQUEST SENSE			1				
SECURITY PROTOCOL IN	see table 91						
SECURITY PROTOCOL OUT	see table 91						
SEND DIAGNOSTIC				1			
SET IDENTIFYING INFORMATION				1			
SET PRIORITY				1			
SET TARGET PORT GROUPS				1			
SET TIMESTAMP				1	1		
TEST UNIT READY	always allow [c]						
WRITE ATTRIBUTE				1			
WRITE BUFFER					1		

[a] The command in the CDB field of the extended CDB (see 4.2.4) that contains the CbCS extension descriptor shall be allowed only if all of the bits marked with a 1 in the row for that command are set in the PERMISSIONS BIT MASK field of the CbCS capability in the CbCS extension descriptor. The permissions bits represented by the empty cells in a row are ignored.

[b] If the CBCS bit is set to one in the Extended INQUIRY Data VPD page (see 7.8.7), this command shall never be allowed.

[c] This command shall always be allowed regardless of whether the CbCS extension descriptor is present and if the CbCS extension descriptor is present regardless of the value in the PERMISSIONS BIT MASK field.

The usage of the PERMISSIONS BIT MASK field for the SECURITY PROTOCOL IN command and the SECURITY PROTOCOL OUT command depend on the following characteristics as shown in table 91:

a) the contents of the SECURITY PROTOCOL field; and
b) the contents of the SECURITY PROTOCOL SPECIFIC field.

Table 91 — Associations between security protocol commands and permissions

Command	SECURITY PROTOCOL field	SECURITY PROTOCOL SPECIFIC field	Description
SECURITY PROTOCOL IN	00h	any	Always allowed regardless of whether the CbCS extension descriptor is present and if the CbCS extension descriptor is present regardless of the value in the PERMISSIONS BIT MASK field
SECURITY PROTOCOL IN	07h	0000h to 003Fh	
SECURITY PROTOCOL IN	07h	0040h to FFFFh	Allowed only if the CbCS extension descriptor is present and the SEC MGMT bit is set to one in the PERMISSIONS BIT MASK field
SECURITY PROTOCOL OUT	any	any	

Command standards may describe the associations between commands and permission bits for the commands that they define. The processing requirements for those associations are the same as those described in this standard.

5.13.6.7.15 CbCS parameters

5.13.6.7.15.1 Overview

CbCS parameters:

a) provide the CbCS Management Application Client class (see 5.13.6.7.4) with a means to control the operation of the Enforcement Manager class (see 5.13.6.7.7);
b) allow the Secure CDB Processor class (see 5.13.6.7.6) to receive security tokens and other CbCS information from the Enforcement Manager class; and
c) allow any application client to receive basic operational CbCS information from the Enforcement Manager class.

CbCS parameters that are not changeable indicate which CbCS features and algorithms are supported. An application client may retrieve the unchangeable CbCS parameters by using the SECURITY PROTOCOL IN command to return the Unchangeable CbCS Parameters CbCS page (see 7.7.4.3.3).

The CbCS parameters with values that change in response to various conditions are summarized in table 92.

Table 92 — Summary of changeable CbCS parameters (part 1 of 2)

Parameter	Support	Applicable CbCS page	Data direction	Capability protection	Reference
CbCS parameters that are updated automatically based on I_T nexus					
Security token	See [a]	Security Token	In	None	7.7.4.3.4
CbCS parameters that provide initial values for dynamically created logical units [b]					
Initial minimum CbCS method	Optional	Current CbCS Parameters	In	Working key	7.7.4.3.5
		Set Minimum CbCS Method [c]	Out	Working key	7.7.4.5.2
Initial policy access tag	Optional	Current CbCS Parameters	In	Working key	7.7.4.3.5
		Set Policy Access Tag [d]	Out	Working key	7.7.4.5.1

[a] Mandatory if the CAPKEY CbCS method (see 5.13.6.7.8.3) is supported.
[b] SCSI target devices that do not dynamically create logical units may not implement these CbCS parameters. Retrieving and setting these CbCS parameters is possible only if the SECURITY PROTOCOL well known logical unit (see 8.5) is implemented. SCSI target devices that dynamically create logical units but do not implement the SECURITY PROTOCOL well known logical unit shall provide a means outside the scope of this standard for managing the values of these CbCS parameters.
[c] If a Set Minimum CbCS Method CbCS page is processed by the SECURITY PROTOCOL well known logical unit, then the initial Minimum CbCS Method CbCS parameter is changed. If a Set Minimum CbCS Method CbCS page is processed by any logical unit other than the SECURITY PROTOCOL well known logical unit, then the minimum CbCS method CbCS parameter for that logical unit is changed.
[d] If a Set Policy Access Tag CbCS page is processed by the SECURITY PROTOCOL well known logical unit, then the initial policy access tag CbCS parameter is changed. If a Set Policy Access Tag CbCS page is processed by any logical unit other than the SECURITY PROTOCOL well known logical unit, then the policy access tag CbCS parameter for that logical unit is changed.

Table 92 — Summary of changeable CbCS parameters (part 2 of 2)

Parameter	Support	Applicable CbCS page	Data direction	Capability protection	Reference
CbCS parameters that affect the CbCS enforcement manager processing					
Minimum CbCS method	Mandatory	Current CbCS Parameters	In	Working key	7.7.4.3.5
		Set Minimum CbCS Method [c]	Out	Working key	7.7.4.5.2
Policy Access Tag	Mandatory	Current CbCS Parameters	In	Working key	7.7.4.3.5
		Set Policy Access Tag [d]	Out	Working key	7.7.4.5.1
Clock	Mandatory	Current CbCS Parameters	In	Working key	7.7.4.3.5
CbCS Shared keys and CbCS shared key identifiers	See [a]	see 5.13.6.7.11			

[a] Mandatory if the CAPKEY CbCS method (see 5.13.6.7.8.3) is supported.

[b] SCSI target devices that do not dynamically create logical units may not implement these CbCS parameters. Retrieving and setting these CbCS parameters is possible only if the SECURITY PROTOCOL well known logical unit (see 8.5) is implemented. SCSI target devices that dynamically create logical units but do not implement the SECURITY PROTOCOL well known logical unit shall provide a means outside the scope of this standard for managing the values of these CbCS parameters.

[c] If a Set Minimum CbCS Method CbCS page is processed by the SECURITY PROTOCOL well known logical unit, then the initial Minimum CbCS Method CbCS parameter is changed. If a Set Minimum CbCS Method CbCS page is processed by any logical unit other than the SECURITY PROTOCOL well known logical unit, then the minimum CbCS method CbCS parameter for that logical unit is changed.

[d] If a Set Policy Access Tag CbCS page is processed by the SECURITY PROTOCOL well known logical unit, then the initial policy access tag CbCS parameter is changed. If a Set Policy Access Tag CbCS page is processed by any logical unit other than the SECURITY PROTOCOL well known logical unit, then the policy access tag CbCS parameter for that logical unit is changed.

The security token CbCS parameter is described in 5.13.6.7.10.

The minimum CbCS method parameter indicates the minimum allowable CbCS method (see 5.13.6.7.8) to be used in capabilities (see 6.26.2.3) processed by an enforcement manager (see 5.13.6.7.7). The initial minimum CbCS method CbCS parameter provides an initial value for the minimum CbCS method CbCS parameter for dynamically created logical units.

The policy access tag parameter indicates the allowable contents of the POLICY ACCESS TAG field in capabilities (see 6.26.2.3) processed by an enforcement manager (see 5.13.6.7.7). The initial policy access tag CbCS parameter provides an initial value for the policy access tag CbCS parameter for dynamically created logical units.

The clock CbCS parameter indicates the time used by the enforcement manager (see 5.13.6.7.7) when evaluating the contents of the CAPABILITY EXPIRATION TIME field in a capability (see 6.26.2.3). The clock CbCS parameter is the timestamp (see 5.2) and its value is managed using the same mechanisms that are used to manage a timestamp.

The CbCS shared keys and CbCS shared key identifiers are described in 5.13.6.7.11.

5.13.6.7.16 CbCS extension descriptor format

The CbCS extension descriptor (see table 93) allows the capability-based command security technique (see 5.13.6.7) to be used with a SCSI command via the parameters defined in this subclause. Support for the CbCS extension descriptor is mandatory if the CBCS bit is set to one in the Extended INQUIRY Data VPD page (see 7.8.7). If an extended CDB (see 4.2.4) includes a CbCS extension descriptor, the CDB field may contain any CDB defined in this standard or any command standard.

Table 93 — CbCS extension descriptor format

Bit / Byte	7	6	5	4	3	2	1	0
0	EXTENSION TYPE (40h)							
1	Reserved							
2								
3								
4	CbCS capability descriptor (see 6.26.2.3)							
•••								
75								
76	(MSB)							
•••	INTEGRITY CHECK VALUE							
139								(LSB)

The EXTENSION TYPE field is defined in 4.2.4.2 and shall be set as shown in table 93 for the CbCS extension descriptor.

The CbCS capability descriptor is defined in 6.26.2.3.

The contents of INTEGRITY CHECK VALUE field depend on the contents of the CBCS METHOD field in the CbCS capability descriptor as follows:

a) if the CBCS METHOD field is not set to CAPKEY or a vendor specific value (see table 267 in 6.26.2.3), then the INTEGRITY CHECK VALUE field is reserved;
b) if the CBCS METHOD field is set to a vendor specific value, then the contents of the INTEGRITY CHECK VALUE field are vendor specific; or
c) if the CBCS METHOD field is set to CAPKEY, then the INTEGRITY CHECK VALUE field contains an integrity check value that is computed using the integrity check value algorithm specified by the INTEGRITY CHECK VALUE ALGORITHM field in the CbCS capability descriptor (see 6.26.2.3) and the following inputs to the integrity check value algorithm:
   A) all the bytes in the security token (see 5.13.6.7.10) for the I_T nexus on which the command is being sent as the string for which the integrity check value is to be computed; and
   B) the CbCS capability key from the credential (see 5.13.6.7.12) in which the CbCS capability descriptor was received by the secure CDB originator as the cryptographic key.

The enforcement manager shall validate the CbCS capability descriptor and the INTEGRITY CHECK VALUE field as described in 5.13.6.7.13.2.

5.13.7 ESP-SCSI encapsulations for parameter data

5.13.7.1 Overview

Subclause 5.13.7 defines a method for transferring encrypted and/or integrity checked parameter data in Data-In Buffers, Data-Out Buffers, variable length CDBs (see 4.2.3), and extended CDBs (see 4.2.4). This method is based on the Encapsulating Security Payload (see RFC 4303) standard developed by the IETF.

NOTE 22 - Because of the constrained usage of ESP-SCSI parameter data in Data-In Buffers and/or Data-Out Buffers, the method defined in this standard differs from the one found in RFC 4303.

5.13.7.2 ESP-SCSI required inputs

Prior to using the ESP-SCSI descriptors defined in 5.13.7, an SA shall be created (see 5.13.2.3) with SA parameters that conform to the requirements defined in 5.13.2.2 and to the following:

a) the USAGE_TYPE SA parameter shall be set to a value for which ESP-SCSI usage is defined in table 75 (see 5.13.2.2);
b) the USAGE_DATA SA parameter shall contain at least the following:
   A) the algorithm identifier and key length for the encryption algorithm (e.g., the ALGORITHM IDENTIFIER field and KEY LENGTH field from the ENCR IKEv2-SCSI cryptographic algorithm descriptor (see 7.7.3.6.2) in the IKEv2-SCSI SAUT Cryptographic Algorithms payload (see 7.7.3.5.14) negotiated by an IKEv2-SCSI SA creation protocol (see 5.13.4)); and
   B) the algorithm identifier for the integrity algorithm (e.g., the ALGORITHM IDENTIFIER field from the INTEG IKEv2-SCSI cryptographic algorithm descriptor (see 7.7.3.6.4) in the IKEv2-SCSI SAUT Cryptographic Algorithms payload (see 7.7.3.5.14) negotiated by an IKEv2-SCSI SA creation protocol (see 5.13.4));
   and
c) the KEYMAT SA parameter shall consist of the shared keys described in 5.13.4.8.6.

ESP-SCSI uses the following additional information derived from the contents of the USAGE_DATA SA parameter:

a) the encryption algorithm identifier shall indicate:
   A) the absence of encryption by being set to ENCR_NULL (see table 105 in 5.13.8);
   B) the size of the initialization vector, if any (e.g., as shown in table 549 (see 7.7.3.6.2));
   C) the size of the salt bytes, if any (e.g., as shown in table 549 (see 7.7.3.6.2)); and
   D) for combined mode encryption algorithms, the size of the integrity check value (i.e., the algorithm's MAC length as shown in table 549 (see 7.7.3.6.2));
   and
b) the integrity algorithm identifier shall indicate:
   A) the use of a combined mode encryption algorithm being set to AUTH_COMBINED (see table 105 in 5.13.8); and
   B) for non-combined mode encryption algorithms, the size of the integrity check value (e.g., as shown in table 553 (see 7.7.3.6.4)).

Each shared key in KEYMAT shall be derived from the KDF generated bits in the order shown in 5.13.4.8.6. The size of each of the shared keys in KEYMAT is determined by the negotiated encryption algorithm and integrity algorithm as described in 5.13.4.4.

5.13.7.3 ESP-SCSI data format before encryption and after decryption

Before data bytes are encrypted and after they are decrypted, they have the format shown in table 94.

Table 94 — ESP-SCSI data format before encryption and after decryption

Bit / Byte	7	6	5	4	3	2	1	0
0	UNENCRYPTED BYTES							
•••								
p-1								
p	PADDING BYTES							
•••								
j-1								
j	PAD LENGTH (j-p)							
j+1	MUST BE ZERO							

The UNENCRYPTED BYTES field contains the bytes that are to be protected via encryption or that have been decrypted.

Before encryption, the PADDING BYTES field contains zero to 255 bytes. The number of padding bytes is:

a) defined by the encryption algorithm; or
b) the number required to cause the length of all bytes prior to encryption (i.e., j+2) to be a whole multiple of the alignment (see table 549 in 7.7.3.6.2) for the encryption algorithm being used.

The contents of the padding bytes are:

a) defined by the encryption algorithm; or
b) if the encryption algorithm does not define the padding bytes contents, a series of one byte binary values starting at one and incrementing by one in each successive byte (i.e., 01h in the first padding byte, 02h in the second padding byte, etc.).

If the encryption algorithm does not place requirements on the contents of the padding bytes option (i.e., option b)), then after decryption the contents of the padding bytes shall be verified to match the series of one byte binary values described in this subclause. If this verification is not successful in a device server, the device server shall terminate the command with CHECK CONDITION status, with the sense key set to ILLEGAL REQUEST, the additional sense code set to INVALID FIELD IN PARAMETER LIST, the SKSV bit set to one, and SENSE KEY SPECIFIC field set to indicate the last byte in the encrypted data as defined in 4.5.2.4.2. If this verification is not successful in an application client, the decrypted data should be ignored.

The PAD LENGTH field is set to the number of bytes in the PADDING BYTES field.

The MUST BE ZERO field is set to zero. After decryption, the contents of the MUST BE ZERO field shall be verified to be zero. If this verification is not successful in a device server, the device server shall terminate the command with CHECK CONDITION status, with the sense key set to ILLEGAL REQUEST, the additional sense code set to INVALID FIELD IN PARAMETER LIST, the SKSV bit set to one, and SENSE KEY SPECIFIC field set to indicate the last byte in the encrypted data as defined in 4.5.2.4.2. If this verification is not successful in an application client, the decrypted data should be ignored.

5.13.7.4 ESP-SCSI outbound data descriptors

5.13.7.4.1 Overview

If ESP-SCSI is used in a variable length CDB (see 4.2.3), an extended CDB (see 4.2.4), or parameter list data that appears in a Data-Out Buffer, then the parameter list data contains one or more descriptors selected based on the criteria shown in table 95.

Table 95 — ESP-SCSI outbound data descriptors

Descriptor name	External descriptor length [a]	Initialization vector present [b]	Reference
ESP-SCSI CDB	No	No	5.13.7.4.2.1
	No	Yes	5.13.7.4.2.2
ESP-SCSI Data-Out Buffer	No	No	5.13.7.4.2.1
	No	Yes	5.13.7.4.2.2
ESP-SCSI Data-Out Buffer without length	Yes	No	5.13.7.4.3.1
	Yes	Yes	5.13.7.4.3.2

[a] Yes means the data format defined for the Data-Out Buffer parameter data includes a length for the ESP-SCSI descriptor.

[b] See the USAGE_DATA SA parameter description in 5.13.7.2 for more information about the initialization vector.

5.13.7.4.2 ESP-SCSI CDBs or Data-Out Buffer parameter lists including a descriptor length

5.13.7.4.2.1 Initialization vector absent

If the USAGE_DATA SA parameter (see 5.13.7.2) indicates an encryption algorithm whose initialization vector size is zero, then the variable length CDB (see 4.2.3), extended CDB (see 4.2.4), or Data-Out Buffer parameter list descriptor shown in table 96 contains the ESP-SCSI data.

Table 96 — ESP-SCSI CDBs or Data-Out Buffer parameter list descriptor without initialization vector

Bit / Byte	7	6	5	4	3	2	1	0
0	(MSB)							
1				DESCRIPTOR LENGTH (n-1)				(LSB)
2								
3				Reserved				
4	(MSB)							
•••				DS_SAI				
7								(LSB)
8	(MSB)							
•••				DS_SQN				
15								(LSB)
16								
•••				ENCRYPTED OR AUTHENTICATED DATA				
i-1								
i	(MSB)							
•••				INTEGRITY CHECK VALUE				
n								(LSB)

The DESCRIPTOR LENGTH field specifies the number of bytes that follow in the ESP-SCSI CDB or ESP-SCSI Data-Out Buffer parameter list descriptor.

The DS_SAI field is set to the value in the DS_SAI SA parameter (see 5.13.2.2) for the SA that is being used to prepare the ESP-SCSI CDB or ESP-SCSI Data-Out Buffer parameter list descriptor. If the DS_SAI value is not known to the device server, the device server shall terminate the command with CHECK CONDITION status, with the sense key set to ILLEGAL REQUEST, the additional sense code set to INVALID FIELD IN PARAMETER LIST, the SKSV bit set to one, and SENSE KEY SPECIFIC field set as defined in 4.5.2.4.2.

The DS_SQN field should be set to one plus the value in the application client's DS_SQN SA parameter (see 5.13.2.2) for the SA that is being used to prepare the ESP-SCSI CDB or ESP-SCSI Data-Out Buffer parameter list descriptor. Before sending the ESP-SCSI CDB or ESP-SCSI Data-Out Buffer parameter list, the application client should copy the contents of the DS_SQN field to its DS_SQN SA parameter.

The device server shall terminate the command with CHECK CONDITION status, with the sense key set to ILLEGAL REQUEST, the additional sense code set to INVALID FIELD IN PARAMETER LIST, the SKSV bit set to one, and SENSE KEY SPECIFIC field set as defined in 4.5.2.4.2 if any of the following conditions occur:

a) the DS_SQN field is set to zero;
b) the value in the DS_SQN field is less than or equal to the value in the device server's DS_SQN SA parameter; or
c) the value in the DS_SQN field is greater than 32 plus the value in the device server's DS_SQN SA parameter.

If the DS_SQN SA parameter is equal to FFFF FFFF FFFF FFFFh, the device server shall delete the SA.

The INITIALIZATION VECTOR field, if any, contains a value that is used as an input into the encryption algorithm and/or integrity algorithm specified by the SA specified by the DS_SAI field. The INITIALIZATION VECTOR field is not encrypted. The encryption algorithm and/or integrity algorithm may define additional requirements for the INITIALIZATION VECTOR field.

The ENCRYPTED OR AUTHENTICATED DATA field contains:

a) if an encryption algorithm for the SA specified by the DS_SAI field is not ENCR_NULL, encrypted data bytes for the following:
   1) the bytes in the UNENCRYPTED BYTES field (see 5.13.7.3);
   2) the bytes in the PADDING BYTES field (see 5.13.7.3);
   3) the byte that is the PAD LENGTH field (see 5.13.7.3); and
   4) the byte that is the MUST BE ZERO field (see 5.13.7.3);

   or
b) otherwise, the unencrypted data bytes.

If the integrity algorithm for the SA specified by the DS_SAI field is AUTH_COMBINED (see 5.13.7.2), then the AAD input to the encryption algorithm is composed of the following concatenated bytes:

1) the bytes in the DS_SAI field; and
2) the bytes in the DS_SQN field.

The INTEGRITY CHECK VALUE field contains a value that is computed as follows:

a) if the integrity algorithm is not AUTH_COMBINED, the integrity check value is computed using the specified integrity algorithm with the following concatenated bytes as inputs:
   1) the bytes in the DS_SAI field;
   2) the bytes in the DS_SQN field;
   3) the bytes in the INITIALIZATION VECTOR field, if any; and
   4) the bytes in the ENCRYPTED OR AUTHENTICATED DATA field after encryption, if any, has been performed;

   or
b) if the integrity algorithm is AUTH_COMBINED, the integrity check value is computed as an additional output of the specified encryption algorithm.

Upon receipt of ESP-SCSI CDB or ESP-SCSI Data-Out Buffer parameter data, the device server shall compute an integrity check value for the ESP-SCSI CDB or ESP-SCSI Data-Out Buffer parameter data as specified by the algorithms specified by the SA specified by the DS_SAI field using the inputs shown in this subclause. If the computed integrity check value does not match the value in the INTEGRITY CHECK VALUE field, the device server shall terminate the command with CHECK CONDITION status, with the sense key set to ILLEGAL REQUEST, the additional sense code set to INVALID FIELD IN PARAMETER LIST, the SKSV bit set to one, and SENSE KEY SPECIFIC field set as defined in 4.5.2.4.2.

If the command is not terminated due to a sequence number error or a mismatch between the computed integrity check value and the contents of the INTEGRITY CHECK VALUE field, then the device server shall copy the contents of the received DS_SQN field to its DS_SQN SA parameter.

5.13.7.4.2.2 Initialization vector present

If the USAGE_DATA SA parameter indicates an encryption algorithm whose initialization vector size (i.e., s) is greater than zero, then the variable length CDB (see 4.2.3), extended CDB (see 4.2.4), or Data-Out Buffer parameter data descriptor shown in table 97 contains the ESP-SCSI data.

Table 97 — ESP-SCSI CDBs or Data-Out Buffer full parameter list descriptor

Bit / Byte	7	6	5	4	3	2	1	0
0	(MSB)							
1				DESCRIPTOR LENGTH (n-1)				(LSB)
2								
3				Reserved				
4	(MSB)							
•••				DS_SAI				
7								(LSB)
8	(MSB)							
•••				DS_SQN				
15								(LSB)
16	(MSB)							
•••				INITIALIZATION VECTOR				
16+s-1								(LSB)
16+s								
•••				ENCRYPTED OR AUTHENTICATED DATA				
i-1								
i	(MSB)							
•••				INTEGRITY CHECK VALUE				
n								(LSB)

The DESCRIPTOR LENGTH field, DS_SAI field, DS_SQN field, ENCRYPTED OR AUTHENTICATED DATA field, and INTEGRITY CHECK VALUE field are defined in 5.13.7.4.2.1.

5.13.7.4.3 ESP-SCSI Data-Out Buffer parameter lists for externally specified descriptor length

5.13.7.4.3.1 Initialization vector absent

If the USAGE_DATA SA parameter (see 5.13.7.2) indicates an encryption algorithm whose initialization vector size is zero and the length of the ESP-SCSI Data-Out Buffer parameter list descriptor is specified in the same parameter list that contains the descriptor, then the Data-Out Buffer parameter list descriptor shown in table 98 contains the ESP-SCSI data.

Table 98 — ESP-SCSI Data-Out Buffer parameter list descriptor without length and initialization vector

Bit / Byte	7	6	5	4	3	2	1	0
0	Reserved							
•••								
3								
4	(MSB)	DS_SAI						
•••								
7								(LSB)
8	(MSB)	DS_SQN						
•••								
15								(LSB)
16	ENCRYPTED OR AUTHENTICATED DATA							
•••								
i-1								
i	(MSB)	INTEGRITY CHECK VALUE						
•••								
n								(LSB)

The DS_SAI field, DS_SQN field, ENCRYPTED OR AUTHENTICATED DATA field, and INTEGRITY CHECK VALUE field are defined in 5.13.7.4.2.1.

5.13.7.4.3.2 Initialization vector present

If the USAGE_DATA SA parameter indicates an encryption algorithm whose initialization vector size (i.e., s) is greater than zero and the length of the ESP-SCSI Data-Out Buffer parameter list descriptor is specified in the same parameter list that contains the descriptor, then the Data-Out Buffer parameter list descriptor shown in table 99 contains the ESP-SCSI data.

Table 99 — ESP-SCSI Data-Out Buffer parameter list descriptor without length

Bit / Byte	7	6	5	4	3	2	1	0
0	Reserved							
•••								
3								
4	(MSB)							
•••	DS_SAI							
7								(LSB)
8	(MSB)							
•••	DS_SQN							
15								(LSB)
16	(MSB)							
•••	INITIALIZATION VECTOR							
16+s-1								(LSB)
16+s	ENCRYPTED OR AUTHENTICATED DATA							
•••								
i-1								
i	(MSB)							
•••	INTEGRITY CHECK VALUE							
n								(LSB)

The DS_SAI field, DS_SQN field, INITIALIZATION VECTOR field, ENCRYPTED OR AUTHENTICATED DATA field, and INTEGRITY CHECK VALUE field are defined in 5.13.7.4.2.1.

5.13.7.5 ESP-SCSI Data-In Buffer parameter data descriptors

5.13.7.5.1 Overview

A device server shall transfer ESP-SCSI parameter data descriptors in a Data-In Buffer only in response to a request that specifies an SA using the AC_SAI SA parameter and DS_SAI SA parameter values (see 5.13.2.2). If the specified combination of AC_SAI and DS_SAI values in a command that requests the transfer of ESP-SCSI parameter data descriptors is not known to the device server, then the device server shall terminate the command with CHECK CONDITION status, with the sense key set to ILLEGAL REQUEST, the additional sense code set to INVALID FIELD IN PARAMETER LIST or to INVALID FIELD IN CDB, the SKSV bit set to one, and SENSE KEY SPECIFIC field set as defined in 4.5.2.4.2.

If ESP-SCSI is used in parameter data which appears in a Data-In Buffer, the parameter data contains one or more descriptors selected based on the criteria shown in table 100.

Table 100 — ESP-SCSI Data-In Buffer parameter data descriptors

Descriptor name	External descriptor length [a]	Initialization vector present [b]	Reference
ESP-SCSI Data-In Buffer	No	No	5.13.7.5.2.1
	No	Yes	5.13.7.5.2.2
ESP-SCSI Data-In Buffer without length	Yes	No	5.13.7.5.3.1
	Yes	Yes	5.13.7.5.3.2

[a] Yes means the data format defined for the Data-Out Buffer parameter data includes a length for the ESP-SCSI descriptor.

[b] See the USAGE_DATA SA parameter description in 5.13.7.2 for more information about the initialization vector.

If ESP-SCSI parameter data descriptors are used in a Data-In Buffer, then the outbound data (see 5.13.7.4) should include at least one ESP-SCSI descriptor using the same SA.

5.13.7.5.2 ESP-SCSI Data-In Buffer parameter data including a descriptor length

5.13.7.5.2.1 Initialization vector absent

If the USAGE_DATA SA parameter (see 5.13.7.2) indicates an encryption algorithm whose initialization vector size is zero, then the Data-In Buffer parameter data descriptor shown in table 101 contains the ESP-SCSI data.

Table 101 — ESP-SCSI Data-In Buffer parameter data descriptor without initialization vector

Bit / Byte	7	6	5	4	3	2	1	0
0	(MSB)							
1				DESCRIPTOR LENGTH (n-1)				(LSB)
2								
3				Reserved				
4	(MSB)							
•••				AC_SAI				
7								(LSB)
8	(MSB)							
•••				AC_SQN				
15								(LSB)
16								
•••				ENCRYPTED OR AUTHENTICATED DATA				
i-1								
i	(MSB)							
•••				INTEGRITY CHECK VALUE				
n								(LSB)

The DESCRIPTOR LENGTH field specifies the number of bytes that follow in the ESP-SCSI Data-In Buffer parameter data descriptor.

The AC_SAI field is set to the value in the AC_SAI SA parameter (see 5.13.2.2) for the SA that is being used to prepare the ESP-SCSI Data-In Buffer parameter data descriptor. If the AC_SAI value is not known to the application client, the ESP-SCSI data-in parameter data descriptor should be ignored.

The AC_SQN field is set to one plus the value in the device server's AC_SQN SA parameter (see 5.13.2.2) for the SA that is being used to prepare the ESP-SCSI data-on buffer parameter data descriptor. Before sending the ESP-SCSI Data-Out Buffer parameter list as part of a command that completes with GOOD status, the device server shall copy the contents of the AC_SQN field to its AC_SQN SA parameter. The device server shall not send two ESP-SCSI Data-Out Buffer parameter data descriptors that contain the same values in AC_SAI field and AC_SQN field.

If the AC_SQN SA parameter is equal to FFFF FFFF FFFF FFFFh, the device server shall delete the SA after the Data-In Buffer parameter data containing that value is sent.

The application client should ignore the ESP-SCSI data-in parameter data descriptor if any of the following occur:

a) the AC_SQN field is set to zero;
b) the value in the AC_SQN field is less than or equal to the value in the application client's AC_SQN SA parameter; or
c) the value in the AC_SQN field is greater than 32 plus the value in the application client's AC_SQN SA parameter.

The INITIALIZATION VECTOR field, if any, contains a value that is used as an input into the encryption algorithm and/or integrity algorithm specified by the SA specified by the AC_SAI field. The INITIALIZATION VECTOR field is not encrypted. The encryption algorithm and/or integrity algorithm may define additional requirements for the INITIALIZATION VECTOR field.

The ENCRYPTED OR AUTHENTICATED DATA field contains:

a) if an encryption algorithm for the SA specified by the AC_SAI field is not ENCR_NULL, encrypted data bytes for the following:
   1) the bytes in the UNENCRYPTED BYTES field (see 5.13.7.3);
   2) the bytes in the PADDING BYTES field (see 5.13.7.3);
   3) the byte that is the PAD LENGTH field (see 5.13.7.3); and
   4) the byte that is the MUST BE ZERO field (see 5.13.7.3);
   or
b) otherwise, the unencrypted data bytes.

If the integrity algorithm for the SA specified by the AC_SAI field is AUTH_COMBINED (see 5.13.7.2), then the AAD input to the encryption algorithm is composed of the following concatenated bytes:

1) the bytes in the AC_SAI field; and
2) the bytes in the AC_SQN field;

The INTEGRITY CHECK VALUE field contains a value that is computed as follows:

a) if the integrity algorithm is not AUTH_COMBINED, the integrity check value is computed using the specified integrity algorithm with the following concatenated bytes as inputs:
   1) the bytes in the AC_SAI field;
   2) the bytes in the AC_SQN field;
   3) the bytes in the INITIALIZATION VECTOR field, if any; and
   4) the bytes in the ENCRYPTED OR AUTHENTICATED DATA field after encryption, if any, has been performed;

or

b) if the integrity algorithms is AUTH_COMBINED, the integrity check value is computed as an additional output of the specified encryption algorithm.

Upon receipt of ESP-SCSI Data-In Buffer parameter data, the application client should compute an integrity check value for the ESP-SCSI parameter data as specified by the algorithms specified by the SA specified by the AC_SAI field using the inputs shown in this subclause. If the computed integrity check value does not match the value in the INTEGRITY CHECK VALUE field, the results returned by the command should be ignored.

The application client should copy the contents of the AC_SQN field to its AC_SQN SA parameter if all of the following occur:

a) the command completed with GOOD status;
b) the ESP-SCSI data-in parameter data descriptor was not ignored due to inconsistency problems with the AC_SQN field; and
c) the computed integrity check value matched the contents of the INTEGRITY CHECK VALUE field.

5.13.7.5.2.2 Initialization vector present

If the USAGE_DATA SA parameter indicates an encryption algorithm whose initialization vector size (i.e., s) is greater than zero, then the Data-In Buffer parameter data descriptor shown in table 102 contains the ESP-SCSI data.

Table 102 — ESP-SCSI Data-In Buffer full parameter data descriptor

Bit / Byte	7	6	5	4	3	2	1	0
0	(MSB)							
1				DESCRIPTOR LENGTH (n-1)				(LSB)
2								
3				Reserved				
4	(MSB)							
•••				AC_SAI				
7								(LSB)
8	(MSB)							
•••				AC_SQN				
15								(LSB)
16	(MSB)							
•••				INITIALIZATION VECTOR				
16+s-1								(LSB)
16+s								
•••				ENCRYPTED OR AUTHENTICATED DATA				
i-1								
i	(MSB)							
•••				INTEGRITY CHECK VALUE				
n								(LSB)

The DESCRIPTOR LENGTH field, AC_SAI field, AC_SQN field, ENCRYPTED OR AUTHENTICATED DATA field, and INTEGRITY CHECK VALUE field are defined in 5.13.7.5.2.1.

5.13.7.5.3 ESP-SCSI Data-In Buffer parameter data for externally specified descriptor length

5.13.7.5.3.1 Initialization vector absent

If the USAGE_DATA SA parameter (see 5.13.7.2) indicates an encryption algorithm whose initialization vector size is zero and the length of the ESP-SCSI Data-In Buffer parameter data descriptor is specified in the same parameter data that contains the descriptor, then the Data-In Buffer parameter data descriptor shown in table 103 contains the ESP-SCSI data.

Table 103 — ESP-SCSI Data-In Buffer parameter data descriptor without length and initialization vector

Bit / Byte	7	6	5	4	3	2	1	0
0	Reserved							
•••								
3								
4	(MSB)			AC_SAI				
•••								
7								(LSB)
8	(MSB)			AC_SQN				
•••								
15								(LSB)
16	ENCRYPTED OR AUTHENTICATED DATA							
•••								
i-1								
i	(MSB)			INTEGRITY CHECK VALUE				
•••								
n								(LSB)

The AC_SAI field, AC_SQN field, ENCRYPTED OR AUTHENTICATED DATA field, and INTEGRITY CHECK VALUE field are defined in 5.13.7.5.2.1.

5.13.7.5.3.2 Initialization vector present

If the USAGE_DATA SA parameter indicates an encryption algorithm whose initialization vector size (i.e., s) is greater than zero and the length of the ESP-SCSI Data-In Buffer parameter data descriptor is specified in the same parameter data that contains the descriptor, then the Data-In Buffer parameter data descriptor shown in table 104 contains the ESP-SCSI data.

Table 104 — ESP-SCSI Data-In Buffer parameter data descriptor without length

Bit / Byte	7	6	5	4	3	2	1	0
0	Reserved							
•••								
3								
4	(MSB)			AC_SAI				
•••								
7								(LSB)
8	(MSB)			AC_SQN				
•••								
15								(LSB)
16	(MSB)			INITIALIZATION VECTOR				
•••								
16+s-1								(LSB)
16+s	ENCRYPTED OR AUTHENTICATED DATA							
•••								
i-1								
i	(MSB)			INTEGRITY CHECK VALUE				
•••								
n								(LSB)

The AC_SAI field, AC_SQN field, INITIALIZATION VECTOR field, ENCRYPTED OR AUTHENTICATED DATA field, and INTEGRITY CHECK VALUE field are defined in 5.13.7.5.2.1.

5.13.8 Security algorithm codes

Table 105 lists the security algorithm codes used in security protocol parameter data.

Table 105 — Security algorithm codes (part 1 of 2)

Code	Description	Reference
Encryption algorithms		
0001 000Ch	CBC-AES-256-HMAC-SHA-1	IEEE 1619.1
0001 0010h	CCM-128-AES-256	IEEE 1619.1
0001 0014h	GCM-128-AES-256	IEEE 1619.1
0001 0016h	XTS-AES-256-HMAC-SHA-512	IEEE 1619.1
8001 000Bh [a]	ENCR_NULL	7.7.3.6.2
8001 000Ch [a]	AES-CBC	RFC 3602
8001 0010h [a]	AES-CCM with a 16-byte MAC	RFC 4309
8001 0014h [a]	AES-GCM with a 16-byte MAC	RFC 4106
8001 0400h to 8001 FFFFh	Vendor specific	
PRF and KDF algorithms [b]		
8002 0002h [a]	IKEv2-use based on SHA-1	table 551 (see 7.7.3.6.3)
8002 0004h [a]	IKEv2-use based on AES-128 in CBC mode	
8002 0005h [a]	IKEv2-use based on SHA-256	
8002 0007h [a]	IKEv2-use based on SHA-512	
8002 0400h to 8002 FFFFh	Vendor specific	
Integrity checking (i.e., AUTH) algorithms		
8003 0002h [a]	AUTH_HMAC_SHA1_96	RFC 2404
8003 000Ch [a]	AUTH_HMAC_SHA2_256_128	RFC 4868
8003 000Eh [a]	AUTH_HMAC_SHA2_512_256	RFC 4868
F003 0000h	AUTH_COMBINED	7.7.3.6.4
8003 0400h to 8003 FFFFh	Vendor specific	

[a] The lower order 16 bits of this code value are assigned to match an IANA assigned value, if any, for an equivalent IKEv2 encryption algorithm and values of 800xh in the high order 16 bits have x selected to match the IANA assigned IKEv2 transform type (e.g., 8001h – Encryption Algorithms, 8002h – PRFs and KDFs).

[b] PRFs are equivalent to the prf() functions defined in RFC 4306. KDFs are equivalent to the prf+() functions defined in RFC 4306.

[c] The low order 8 bits of this code value are assigned to match the AUTH METHOD field in the Authentication payload (see 7.7.3.5.7).

Table 105 — Security algorithm codes (part 2 of 2)

Code	Description	Reference
Diffie-Hellman algorithms		
8004 000Eh [a]	2 048-bit MODP group (finite field D-H)	RFC 3526
8004 000Fh [a]	3 072-bit MODP group (finite field D-H)	RFC 3526
8004 0010h [a]	4 096-bit MODP group (finite field D-H)	RFC 3526
8004 0013h [a]	256-bit random ECP group	RFC 4753
8004 0015h [a]	521-bit random ECP group	RFC 4753
8004 0400h to 8004 FFFFh	Vendor specific	
SA Authentication payload authentication algorithms		
00F9 0000h [c]	SA_AUTH_NONE	5.13.4.3.4 and 7.7.3.6.6
00F9 0001h [c]	RSA Digital Signature with SHA-1	RFC 4306
00F9 0002h [c]	Shared Key Message Integrity Code	RFC 4306
00F9 0009h [c]	ECDSA with SHA-256 on the P-256 curve	RFC 4754
00F9 000Bh [c]	ECDSA with SHA-512 on the P-521 curve	RFC 4754
00F9 00C9h to 00F9 00FFh	Vendor specific	
Other algorithms		
0000 0000h to 0000 FFFFh	Restricted	IANA
All other values	Reserved	

[a] The lower order 16 bits of this code value are assigned to match an IANA assigned value, if any, for an equivalent IKEv2 encryption algorithm and values of 800xh in the high order 16 bits have x selected to match the IANA assigned IKEv2 transform type (e.g., 8001h – Encryption Algorithms, 8002h – PRFs and KDFs).

[b] PRFs are equivalent to the prf() functions defined in RFC 4306. KDFs are equivalent to the prf+() functions defined in RFC 4306.

[c] The low order 8 bits of this code value are assigned to match the AUTH METHOD field in the Authentication payload (see 7.7.3.5.7).

5.14 Self-test operations

5.14.1 Self-test types

The SEND DIAGNOSTIC command (see 6.42) provides methods for an application client to request that a SCSI target device perform self test operations. This standard defines the following self-tests:

a) the default self-test (see 5.14.2);
b) the short self-test (see 5.14.3); and
c) the extended self-test (see 5.14.3).

5.14.2 Default self-test

The default self-test is mandatory for all SCSI target device types that support the SEND DIAGNOSTIC command. The operations performed for the default self-test are not defined by this standard (e.g., performing no diagnostics and returning GOOD status is a valid default self-test). An application client requests that a SCSI target device perform a default self-test by setting the SELFTEST bit to one in the SEND DIAGNOSTIC command (see 6.42).

An application client may use the DEVOFFL bit and the UNITOFFL bit in the SEND DIAGNOSTIC command to allow the device server to perform operations during a default self-test that affect conditions for one or more logical units in the SCSI target device (e.g., if the DEVOFFL bit is set to one, then the device server may clear established reservations while performing the test, and if the UNITOFFL bit is set to one, then the logical unit may alter its medium while performing the test).

While a SCSI target device is performing a default self-test, the device server shall terminate all commands, except INQUIRY commands, REPORT LUNS commands, and REQUEST SENSE commands, with CHECK CONDITION status with the sense key set to NOT READY and the additional sense code set to LOGICAL UNIT NOT READY, SELF-TEST IN PROGRESS. If the device server receives an INQUIRY command, a REPORT LUNS command, or a REQUEST SENSE command while performing a default self-test, then the device server shall process the command.

If the SCSI target device detects no errors during a default self-test, then the device server shall complete the command with GOOD status. If the SCSI target device detects an error during the test, then the device server shall terminate the command with CHECK CONDITION status with the sense key set to HARDWARE ERROR and the additional sense code set to indicate the cause of the error.

5.14.3 The short self-test and extended self-test

An application client may request that a SCSI target device perform a short self-test or the extended self-test by setting the SELFTEST bit to zero and specifying an appropriate value in the SELF-TEST CODE field in the SEND DIAGNOSTIC command (see 6.42).

The criteria for the short self-test are that the test has one or more segments and completes in two minutes or less. The criteria for the extended self-test are that the test has one or more segments and that the completion time required by the SCSI target device to complete the extended self-test is reported in the EXTENDED SELF-TEST COMPLETION MINUTES field in the Extended INQUIRY Data VPD page (see 7.8.7), the EXTENDED SELF-TEST COMPLETION TIME field in the Control mode page (see 7.5.8), or both.

The tests performed in the segments are vendor specific and may be the same for the short self-test and the extended self-test.

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) a buffer RAM test;
   B) a read/write circuitry test; and
   C) a test of the read/write heads;
b) a seek/servo segment wherein a device tests it 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.

5.14.4 Self-test modes

5.14.4.1 Self-test modes overview

A foreground mode (see 5.14.4.2) and a background mode (see 5.14.4.3) are defined for the short self-test and the extended self-test. An application client specifies the self-test mode by the value in the SELF-TEST CODE field in the SEND DIAGNOSTIC command (see 6.42).

5.14.4.2 Foreground mode

If an application client specifies a self-test to be performed in the foreground mode, the device server shall return status for the command after the self-test has been completed.

While a SCSI target device is performing a self-test in the foreground mode, the device server shall terminate all commands, except INQUIRY commands, REPORT LUNS commands, and REQUEST SENSE commands, with CHECK CONDITION status, with the sense key set to NOT READY, and the additional sense code set to LOGICAL UNIT NOT READY, SELF-TEST IN PROGRESS. If the device server receives an INQUIRY command, a REPORT LUNS command, or a REQUEST SENSE command while performing a self-test in the foreground mode, then the device server shall process the command.

If a SCSI target device is performing a self-test in the foreground mode and an error occurs during the test, then:

a) the SCSI target device shall abort the self-test; and
b) the device server is:
   A) able to update the Self-Test Results log page (see 7.3.17), then the device server shall update the Self-Test Results log page and terminate the SEND DIAGNOSTIC command with CHECK CONDITION status, with the sense key set to HARDWARE ERROR, and the additional sense code set to LOGICAL UNIT FAILED SELF-TEST. The application client may obtain additional information about the failure by reading the Self-Test Results log page; or
   B) unable to update the Self-Test Results log page, then the device server shall terminate the SEND DIAGNOSTIC command with CHECK CONDITION status, with the sense key set to HARDWARE ERROR, and the additional sense code set to LOGICAL UNIT UNABLE TO UPDATE SELF-TEST LOG.

An application client may cause a SCSI target device to abort a self-test that is being performed in the foreground mode by using a task management function (see SAM-5) (e.g., an ABORT TASK task management function, a CLEAR TASK SET task management function) or a transport specific reset (e.g., a hard reset event) (see SAM-5). In addition, a self-test being performed in the foreground mode shall be terminated by an I_T nexus loss event or a power loss expected event (see SAM-5). If a SCSI target device aborts a self-test that is being performed in the foreground mode based on the SCSI target device receiving a task management function or a transport specific event, then the device server shall update the Self-Test Results log page (see 7.3.17).

5.14.4.3 Background mode

If a device server receives a SEND DIAGNOSTIC command specifying a self-test to be performed in the background mode, then:

a) the device server shall terminate the command if the CDB is invalid; or
b) the device server shall:
   1) complete the command with GOOD status;
   2) initialize the next self-test results log parameter in the Self-Test Results log page (see 7.3.17) by setting:
      a) the SELF-TEST CODE field to the self-test code from the SEND DIAGNOSTIC command; and
      b) the SELF-TEST RESULTS field to Fh;
      and

3) begin the self-test.

An application client may request that a device server abort a self-test that is being performed in the background mode by sending a SEND DIAGNOSTIC command with the SELF-TEST CODE field set to 100b (i.e., abort background self-test function). A SCSI target device shall not abort a self-test being performed in the background mode as the result of an I_T nexus loss event (see SAM-5). A SCSI target device shall abort a self-test being performed in the background mode as the result of a power loss expected event (see SAM-5).

While the SCSI target device is performing a self-test in the background mode, the device server shall terminate with CHECK CONDITION status, with the sense key set to NOT READY and the additional sense code to LOGICAL UNIT NOT READY, SELF-TEST IN PROGRESS any received SEND DIAGNOSTIC command that meets any of the following criteria:

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

If the SCSI target device is performing a self-test in the background mode, and the device server receives any command that requires suspension of the self-test to process, except those listed in table 106, then:

a) the device server shall suspend the self-test;
b) the device server shall begin processing the command within two seconds after the CDB has been validated; and
c) after the command completes, the device server shall resume the self-test.

If the device server receives one of the commands listed in table 106, then the device server shall:

a) abort the self-test;
b) update the self-test log; and
c) begin processing the command within two seconds after the CDB has been validated.

Table 106 — Exception commands for background self-tests

Device type	Command	Reference
All device types	SEND DIAGNOSTIC (with SELF-TEST CODE field set to 100b) WRITE BUFFER with the MODE field set to any download microcode mode (see table 58 in 5.4))	6.42 6.49
Direct access block	FORMAT UNIT START STOP UNIT	SBC-3
Sequential access	ERASE FORMAT MEDIUM LOAD UNLOAD	SSC-4
Object-based storage	Any command with operation code 7Fh (i.e., all commands defined by the OSD standard)	OSD
NOTE – 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.		

5.14.4.4 Features common to foreground and background self-test modes

An application client may use a REQUEST SENSE command (see 6.39) to poll for progress indication at any time during a self-test. The device server shall provide pollable REQUEST SENSE data (see 5.10.2) with the

sense key set to NOT READY, the additional sense code set to LOGICAL UNIT NOT READY, SELF-TEST IN PROGRESS, and the PROGRESS INDICATION field set to indicate the progress of the self-test.

An application client may use the EBACKERR bit and the MRIE field in the Informational Exceptions Control mode page (see applicable command standard) to control the reporting of errors that occur during a background self-test operation.

An application client may obtain information about the 20 most recent self-tests, including the self-test in progress, if any, by reading the Self-Test Results log page (see 7.3.17). With the exception of progress indication, this is the only method for an application client to obtain information about self-tests performed in the background mode unless an error occurs during the self-test.

Table 107 summarizes:

a) when a device server returns status after receipt of a self-test command;
b) how an application client may abort a self-test;
c) how a device server processes commands that are entered into the task set while a self-test is in progress; and
d) how a self-test failure is reported.

Table 107 — Self-test mode summary

Self-test mode	When status is returned	How to abort the self-test	Processing of commands while a self-test is in progress	Self-test failure reporting
Fore-ground	After the self-test is com-plete	A task management function or reset event that causes a self-test to be aborted (see 5.14.4.2)	If the command is INQUIRY, REPORT LUNS or REQUEST SENSE, then process normally, otherwise, terminate with CHECK CONDITION sta-tus, with the sense key set to NOT READY, and the additional sense code set to LOGICAL UNIT NOT READY, SELF-TEST IN PROGRESS.	The device server terminates the SEND DIAGNOSTIC command with CHECK CONDITION status, with the sense key set to HARDWARE ERROR, and the additional sense code set to LOGICAL UNIT FAILED SELF-TEST or LOGICAL UNIT UNABLE TO UPDATE SELF-TEST LOG (see 5.14.4.2). [a]
Back-ground	After the CDB is validated	A SEND DIAG-NOSTIC command with the SELF-TEST CODE field set to 100b	Process the command, except as described in 5.14.4.3.	An application client: a) checks the Self-Test Results log page (see 7.3.17) after the PROGRESS INDICATION field returned in response to a REQUEST SENSE command indicates that the self-test is complete; or b) uses the EBACKERR bit and the MRIE field (see applicable command standard) to specify a method of indicating that a failure occurred. If a failure occurs, then an additional sense code of WARNING - BACKGROUND SELF-TEST FAILED shall be returned using the method defined in the MRIE field.

[a] The device server shall not report an error until after the Self-Test Results log page is updated.

5.15 Target port group asymmetric access states

5.15.1 Target port group access overview

Logical units may be connected to one or more service delivery subsystems via multiple target ports (see SAM-5). The access to logical units through the multiple target ports may be symmetrical (see 5.15.3) or asymmetrical (see 5.15.2).

If referrals are supported (see SBC-3), then a logical unit accessed through a target port group may have different target port group asymmetric access states based on the user data segments being accessed.

5.15.2 Asymmetric logical unit access

5.15.2.1 Introduction to asymmetric logical unit access

Asymmetric logical unit access occurs when the access characteristics of one port may differ from those of another port. SCSI target devices with target ports implemented in separate physical units may designate differing levels of access for the target ports associated with each logical unit. While commands and task management functions (see SAM-5) may be routed to a logical unit through any target port, the performance may not be optimal, and the allowable command set may be less complete than when the same commands and task management functions are routed through a different target port. In addition, some target ports may be in a state (e.g., offline) that is unique to that target port. If a failure on the path to one target port is detected, the SCSI target device may perform automatic internal reconfiguration to make a logical unit accessible from a different set of target ports or may be instructed by the application client to make a logical unit accessible from a different set of target ports.

A target port characteristic called primary target port asymmetric access state (see 5.15.2.4) defines properties of a target port and the allowable command set for a logical unit when commands and task management functions are routed through the target port maintaining that state.

A primary target port group is defined as a set of target ports that are in the same primary target port asymmetric access state at all times (i.e., a change in one target port's primary target port asymmetric access state implies an equivalent change in the primary target port asymmetric access state of all target ports in the same primary target port group). A primary target port group asymmetric access state is defined as the primary target port asymmetric access state common to the set of target ports in a primary target port group. One target port is a member of at most one primary target port group for a logical unit group (see 7.8.6.9). The grouping of target ports in a primary target port group is vendor specific.

A logical unit may have commands and task management functions routed through multiple primary target port groups. Logical units support asymmetric logical unit access if different primary target port groups may be in different primary target port group asymmetric access states. Support for asymmetric logical unit access should not affect how the device server responds to unsupported commands or how the task manager responds to unsupported task management functions.

An example of asymmetric logical unit access is a SCSI controller device with two separated controllers where all target ports on one controller are in the same primary target port asymmetric access state with respect to a logical unit and are members of the same primary target port group. Target ports on the other controller are members of another primary target port group. The behavior of each primary target port group may be different with respect to a logical unit, but all members of a single primary target port group are always in the same primary target port group asymmetric access state with respect to a logical unit.

An example of primary target port groups is shown in figure 16.

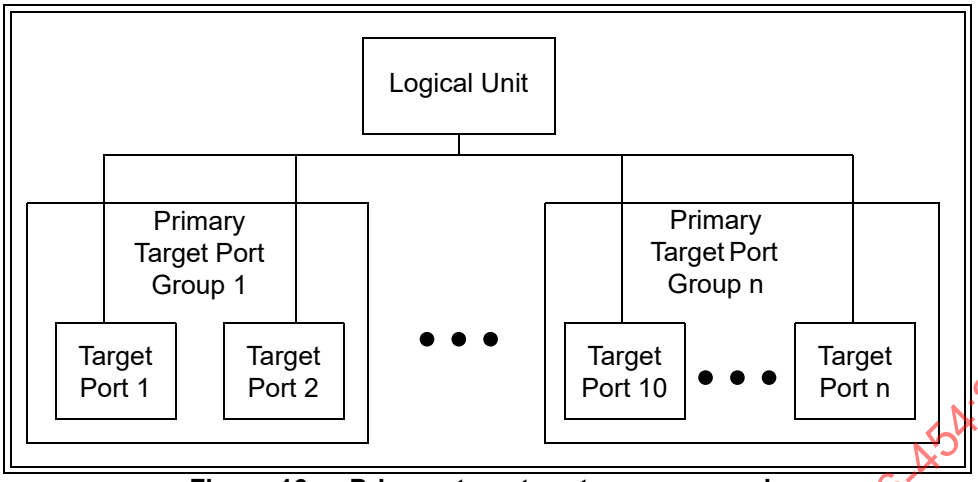


Figure 16 — Primary target port group example

Another target port characteristic called secondary target port asymmetric access state (see 5.15.2.4) indicates a condition that affects the way in which an individual target port participates in its assigned primary target port group. All target ports, if any, in one secondary target port asymmetric access state are grouped into a secondary target port group. Secondary target port groups have the following properties:

a) a target port in any secondary target port group also shall be in one primary target port group;
b) a change of secondary target port asymmetric access state for one target port shall not cause changes in the secondary target port asymmetric access state of other target ports, if any, in the same secondary target port group; and
c) a target port may be a member of zero or more secondary target port groups.

The term, target port asymmetric access state, represents both primary target port asymmetric access states and secondary target port asymmetric access states. The term, target port group, represents both primary target port groups and secondary target port groups.

5.15.2.2 Explicit and implicit asymmetric logical unit access

Asymmetric logical unit access may be managed explicitly by an application client using the REPORT TARGET PORT GROUPS (see 6.37) command and SET TARGET PORT GROUPS (see 6.45) command.

Alternatively, asymmetric logical unit access may be managed implicitly by the SCSI target device based on the type of transactions being routed through each target port and the internal configuration capabilities of the primary target port group(s) through which the logical unit may be accessed. The logical units may attempt to maintain full performance across the primary target port groups that are busiest and that show the most reliable performance, allowing other primary target port groups to select a lower performance primary target port asymmetric access state.

Implicit management of secondary target port asymmetric access states is based on the condition of an individual target port and how such conditions affect that target port's ability to participate in its assigned primary target port group.

If both explicit and implicit asymmetric logical unit access management methods are implemented, the precedence of one over the other is vendor specific.

5.15.2.3 Discovery of asymmetric logical unit access behavior

SCSI logical units with asymmetric logical unit access may be identified using the INQUIRY command. The value in the target port group support (TPGS) field (see 6.6.2) indicates whether or not the logical unit supports

asymmetric logical unit access and if so whether implicit or explicit management is supported. The target port asymmetric access states supported by a logical unit may be determined by the REPORT TARGET PORT GROUPS command parameter data (see 6.37).

5.15.2.4 Target port asymmetric access states

5.15.2.4.1 Target port asymmetric access states overview

For all SCSI target devices that report in the INQUIRY data that they support asymmetric logical unit access, all of the target ports in a primary target port group (see 5.15.2.1) shall be in the same primary target port asymmetric access state with respect to the ability to route information to a logical unit. The primary target port asymmetric access states are:

a) active/optimized;
b) active/non-optimized;
c) standby;
d) unavailable; and
e) logical block dependent.

Individual target ports may be in secondary target port groups (see 5.15.2.1) that have the following secondary target port asymmetric access states:

a) offline.

5.15.2.4.2 Active/optimized state

The active/optimized state is a primary target port asymmetric access state. While commands and task management functions are being routed through a target port in the active/optimized primary target port asymmetric access state, the device server shall function (e.g., respond to commands) as specified in the appropriate command standards. All target ports within a primary target port group should be capable of immediately accessing the logical unit.

The SCSI target device shall participate in all task management functions as defined in SAM-5 and modified by the applicable SCSI transport protocol standards.

5.15.2.4.3 Active/non-optimized state

The active/non-optimized state is a primary target port asymmetric access state. While commands and task management functions are being routed through a target port in the active/non-optimized primary target port asymmetric access state, the device server shall function as specified in the appropriate command standards.

The processing of some task management functions and commands, especially those involving data transfer or caching, may operate with lower performance than they would if the target port were in the active/optimized primary target port asymmetric access state.

The SCSI target device shall participate in all task management functions as defined in SAM-5 and modified by the applicable SCSI transport protocol standards.

5.15.2.4.4 Standby state

The standby state is a primary target port asymmetric access state. While being accessed through a target port in the standby primary target port asymmetric access state, the device server shall support those of the following commands that it supports while in the active/optimized primary target port asymmetric access state:

a) INQUIRY;
b) LOG SELECT;
c) LOG SENSE;

d) MODE SELECT;
e) MODE SENSE;
f) REPORT LUNS;
g) RECEIVE DIAGNOSTIC RESULTS;
h) SEND DIAGNOSTIC;
i) REPORT TARGET PORT GROUPS;
j) SET TARGET PORT GROUPS;
k) REQUEST SENSE;
l) PERSISTENT RESERVE IN;
m) PERSISTENT RESERVE OUT;
n) echo buffer modes of READ BUFFER; and
o) echo buffer modes of WRITE BUFFER.

The device server may support other commands while in the standby state.

The device server shall terminate commands that are not supported in the standby state with CHECK CONDITION status, with the sense key set to NOT READY, and the additional sense code set to LOGICAL UNIT NOT ACCESSIBLE, TARGET PORT IN STANDBY STATE.

The SCSI target device shall participate in all task management functions as defined in SAM-5 and modified by the applicable SCSI transport protocol standards.

5.15.2.4.5 Unavailable state

The unavailable state is a primary target port asymmetric access state. While being accessed through a target port in the unavailable primary target port asymmetric access state, the device server shall accept only a limited set of commands. The unavailable primary target port asymmetric access state is intended for situations where the target port accessibility to a logical unit may be severely constrained due to SCSI target device limitations (e.g., hardware errors). Therefore it may not be possible to transition from the unavailable state to the active/optimized state, the active/non-optimized state, or the standby state. The unavailable primary target port asymmetric access state is also intended for minimizing any disruption when using the downloading microcode modes of the WRITE BUFFER command (see 5.4).

While in the unavailable primary target port asymmetric access state, the device server shall support those of the following commands that it supports while in the active/optimized state:

a) INQUIRY (the peripheral qualifier (see 6.6.2) shall be set to 001b);
b) REPORT LUNS;
c) REPORT TARGET PORT GROUPS;
d) SET TARGET PORT GROUPS;
e) REQUEST SENSE;
f) echo buffer modes of READ BUFFER;
g) echo buffer modes of WRITE BUFFER; and
h) download microcode mode of WRITE BUFFER.

The device server may support other commands while in the unavailable state.

The device server shall terminate commands that are not supported in the unavailable state with CHECK CONDITION status, with the sense key set to NOT READY, and the additional sense code set to LOGICAL UNIT NOT ACCESSIBLE, TARGET PORT IN UNAVAILABLE STATE.

The SCSI target device is not required to participate in all task management functions (see SAM-5 and the applicable SCSI transport protocol standards).

5.15.2.4.6 Offline state

The offline state is a secondary target port asymmetric access state. Target ports in the offline secondary target port asymmetric access state are not accessible via the service delivery subsystem (e.g., during maintenance, port replacement, port disabled, hot swap, or power failures that affect only a subset of target ports). While in the offline secondary target port asymmetric state, the target port is not capable of receiving or responding to any commands or task management functions.

The offline secondary target port asymmetric access state allows a device server to report that some target ports are not capable of being accessed.

After access to the service delivery subsystem is enabled, the target port shall transition out of the offline secondary target port asymmetric access state.

5.15.2.4.7 Logical block dependent state

The logical block dependent state is a primary target port asymmetric access state. The logical block dependent state only occurs if the device server supports referrals (see 7.8.7).

The target port asymmetric access state for a user data segment shall be one of the following target port asymmetric access states:

a) active/optimized;
b) active/non-optimized;
c) transitioning; or
d) unavailable.

An application client may determine the target port asymmetric access state for user data segments by issuing a REPORT REFERRALS command (see SBC-3).

5.15.2.5 Transitions between target port asymmetric access states

The movement from one target port asymmetric access state to another is called a transition.

During a transition between target port asymmetric access states the device server shall respond to a command in one of the following ways:

a) if during the transition the logical unit is inaccessible, then the transition is performed as a single indivisible event and the device server shall respond by either returning BUSY status, or returning CHECK CONDITION status, with the sense key set to NOT READY, and the sense code set to LOGICAL UNIT NOT ACCESSIBLE, ASYMMETRIC ACCESS STATE TRANSITION; or
b) if during the transition the target ports in a primary target port group are able to access the requested logical unit, then:
   A) the device server shall support those of the following commands that it supports while in the active/optimized primary target port asymmetric access state:
      a) INQUIRY;
      b) REPORT LUNS;
      c) REPORT TARGET PORT GROUPS;
      d) REQUEST SENSE;
      e) echo buffer modes of READ BUFFER; and
      f) echo buffer modes of WRITE BUFFER;
   B) the device server may support other commands when those commands are routed through a target port that is transitioning between primary target port asymmetric access states;
   C) the device server shall terminate commands that are not supported during a transition with CHECK CONDITION status, with the sense key set to NOT READY, and the additional sense code set to LOGICAL UNIT NOT ACCESSIBLE, ASYMMETRIC ACCESS STATE TRANSITION; and

D) the SCSI target device is not required to participate in all task management functions.

If the transition was explicit to a supported target port asymmetric access state and it failed, then the device server shall terminate the command with CHECK CONDITION status, with the sense key set to HARDWARE ERROR, and the additional sense code set to SET TARGET PORT GROUPS COMMAND FAILED. If the transition was to a primary target port asymmetric access state, the primary target port group that encountered the error should complete a transition to the unavailable primary target port asymmetric access state.

If a target port asymmetric access state change occurred as a result of the failed transition, then the device server shall establish a unit attention condition for the initiator port associated with every I_T nexus other than the I_T nexus on which the SET TARGET PORT GROUPS command was received with the additional sense code set to ASYMMETRIC ACCESS STATE CHANGED.

If the transition was implicit and it failed, then the device server shall establish a unit attention condition for the initiator port associated with every I_T nexus with the additional sense code set to IMPLICIT ASYMMETRIC ACCESS STATE TRANSITION FAILED.

An implicit CLEAR TASK SET task management function may be performed following a transition failure.

Once a transition is completed, the new target port asymmetric access state may apply to some or all commands entered into the task set before the completion of the transition. The new target port asymmetric access state shall apply to all commands received by the device server after completion of a transition.

If a transition is to the offline secondary target port asymmetric access state, communication with the service delivery subsystem shall be terminated. This may result in commands being terminated and may cause command timeouts to occur on the initiator.

After an implicit target port asymmetric access state change, a device server shall establish a unit attention condition for the initiator port associated with every I_T nexus with the additional sense code set to ASYMMETRIC ACCESS STATE CHANGED.

After an explicit target port asymmetric access state change, a device server shall establish a unit attention condition with the additional sense code set to ASYMMETRIC ACCESS STATE CHANGED for the initiator port associated with every I_T nexus other than the I_T nexus on which the SET TARGET PORT GROUPS command was received.

5.15.2.6 Preference indicator

A device server may indicate one or more primary target port groups is a preferred primary target port group for accessing a logical unit by setting the PREF bit to one in the target port group descriptor (see 6.37). The preference indication is independent of the primary target port asymmetric access state.

An application client may use the PREF bit value in the target port group descriptor to influence the path selected for a logical unit (e.g., a primary target port group in the standby primary target port asymmetric access state with the PREF bit set to one may be chosen over a primary target port group in the active/optimized primary target port asymmetric access state with the PREF bit set to zero).

The value of the PREF bit for a primary target port group may change whenever a primary target port asymmetric access state changes.

5.15.2.7 Target port asymmetric access state reporting

Target port asymmetric access state information is reported in a target port group descriptor (see table 308) in the parameter data returned by the REPORT TARGET PORT GROUPS command (see 6.37). Each target port group descriptor indicates the asymmetric access state for all target ports contained in the indicated target port group (i.e., all target ports described in the target port descriptors contained in the target port group

descriptor). A primary asymmetric access state (see table 309) applies to all logical units in the same logical unit group (see 7.8.6.9).

The parameter data returned by each REPORT TARGET PORT GROUPS command is independent (e.g., the device server may return different values in the ASYMMETRIC ACCESS STATE field (see table 308) for the same target port group in the parameter data returned by REPORT TARGET PORT GROUPS commands received through different target ports).

The information returned in the target port group descriptor that contains a target port descriptor with a relative target port identifier that matches the relative target port identifier of the target port through which the REPORT TARGET PORTS GROUP command is received shall be the current information.

EXAMPLE – An application client sends two REPORT TARGET PORT GROUPS commands to the logical unit shown in figure 16. The commands are received via different target ports with target port identifiers 2 and 10. Although the command that is received via the target port with relative target port identifier 2 returns the current information for the target port group that contains the target port with relative target port identifier 2, the command may not return the current information for the target port group that contains the target port with relative target port identifier 10. The converse is true for the command that is received via the target port with relative target port identifier 10. Although the command that is received via the target port with relative target port identifier 10 returns the current information for the target port group that contains the target port with relative target port identifier 10, the command may not return the current information for the target port group that contains the target port with relative target port identifier 2.

If an application client detects different parameter data returned by REPORT TARGET PORT GROUPS commands for the same target port group (i.e., both current information and information that may not be current), then the application client should use the current information.

5.15.2.8 Implicit asymmetric logical units access management

SCSI target devices with implicit asymmetric logical units access management are capable of using mechanisms other than the SET TARGET PORT GROUPS command to set:

a) the primary target port asymmetric access state of a primary target port group; or
b) the secondary target port asymmetric access state of a target port that is a member of a primary target port group.

All logical units that report in the standard INQUIRY data (see 6.6.2) that they support asymmetric logical unit access and support implicit asymmetric logical unit access (i.e., the TPGS field is set to 01b or 11b):

a) shall implement the INQUIRY command Device Identification VPD page designator type 4h (see 7.8.6.7) and designator type 5h (see 7.8.6.8);
b) shall support the REPORT TARGET PORT GROUPS command as described in 6.37; and
c) may implement the INQUIRY command Device Identification VPD page designator type 6h (see 7.8.6.9).

Implicit logical unit access state changes between primary target port asymmetric access states may be disabled with the IALUAE bit in the Control Extension mode page (see 7.5.9).

5.15.2.9 Explicit asymmetric logical units access management

All logical units that report in the standard INQUIRY data (see 6.6.2) that they support asymmetric logical units access and support explicit asymmetric logical unit access (i.e., the TPGS field is set to 10b or 11b):

a) shall implement the INQUIRY command Device Identification VPD page designator type 4h (see 7.8.6.7) and designator type 5h (see 7.8.6.8);
b) shall support the REPORT TARGET PORT GROUPS command as described in 6.37;
c) shall support the SET TARGET PORT GROUPS command as described in 6.45; and

d) may implement the INQUIRY command Device Identification VPD page designator type 6h (see 7.8.6.9).

5.15.2.10 Behavior after power on, hard reset, logical unit reset, and I_T nexus loss

For all SCSI target devices that report in the standard INQUIRY data (see 6.6.2) that they support only explicit asymmetric logical unit access (i.e., the TPGS field is set to 10b), the target port shall preserve the primary target port asymmetric access state during any power on, hard reset, logical unit reset, and I_T nexus loss.

5.15.2.11 Behavior of target ports that are not accessible from the service delivery subsystem

If the offline secondary target port asymmetric access state is supported and a subset of the target ports in a primary target port group are not accessible via the service delivery subsystem (e.g., power failure), then those ports may be reported in a primary target port group consistent with their primary target port asymmetric access state and in the secondary target port group with the offline secondary target port asymmetric access state.

5.15.3 Symmetric logical unit access

A device server that provides symmetrical access to a logical unit may use a subset of the asymmetrical logical access features (see 5.15.2) to indicate this ability to an application client, providing an application client a common set of commands to determine how to manage target port access to a logical unit.

Symmetrical logical unit access should be represented as follows:

a) the TPGS field in the standard INQUIRY data (see 6.6.2) indicates that implicit asymmetric access is supported;
b) the REPORT TARGET PORT GROUPS command is supported; and
c) the REPORT TARGET PORT GROUPS parameter data indicates that the same state (e.g., active/optimized state) is in effect for all primary target port groups.

5.16 Third-party copies

5.16.1 General considerations for third-party copies

Third-party copy commands (see 5.16.3) cause a copy manager to perform copy operations (see 5.16.4.3) that transfer data as follows:

a) from specified areas of the medium within a logical unit to different areas within the same logical unit;
b) from one logical unit to another within a SCSI target device; or
c) from one SCSI target device to another SCSI target device.

The transfers requested by a third-party copy command are managed by a copy manager (see 5.16.2) that is contained in a logical unit (see SAM-5). The copy manager is responsible for transferring data from copy sources to copy destinations (i.e., reading from the copy sources, buffering as needed, and writing to the copy destinations).

Application clients and other copy managers request third-party copy operations by sending commands (see 5.16.3) to a copy manager for processing by that copy manager. Copy managers provide information about their capabilities to application clients in the Third-party Copy VPD page (see 7.8.17).

A copy manager is not required to support all the third-party copy features described in 5.16, but if a copy manager supports copies from one SCSI target device to another SCSI target device, then the copy manager shall support copies from one logical unit to another logical unit within the same SCSI target device.

If a logical unit contains a copy manager that supports any of the commands described in 5.16.3, then the 3PC bit shall be set to one in the standard INQUIRY data (see 6.6.2).

5.16.2 Copy manager model

A copy manager is:

a) a kind of device server that processes the commands described in 5.16.3 using foreground or background copy operations (see 5.16.4.3); and
b) a kind of application client that sends commands to other device servers and copy managers, possibly in other SCSI target devices, as necessary to perform the copy operation originated by a third-party copy command (see 5.16.4.3).

If a copy manager interacts with a copy manger in another SCSI target device, these interactions may occur over a SCSI transport protocol or over a non-SCSI transport.

Upon the successful completion of a copy operation (see 5.16.4.3), the copy manager shall have produced the results specified by the command that originated the copy operation in accordance with this standard or the applicable command standard (e.g., SBC-3 for the WRITE USING TOKEN command). To ensure interoperability, this standard or the applicable command standard may place requirements on the copy manager using terminology based on specific commands and parameter data (e.g., specific instances of the EXTENDED COPY(LID4) command). These are only functional descriptions of the required copy manager behavior. Any copy manager implementation that produces the specified results and does not violate interoperability may be used.

A copy manager may be contained in:

a) the same logical unit as the device server for a data storage device (e.g., a direct access block device (see SBC-3) or sequential-access device (see SSC-4)); or
b) a standalone SCSI target device whose only purpose is to contain one or more logical units that contain copy managers.

If the copy manager is contained in the same logical unit as the device server for a data storage device, then the following requirements shall apply:

a) the PERIPHERAL DEVICE TYPE field in the standard INQUIRY data (see 6.6.2) shall indicate the device type associated with the device server;
b) the copy manager shall have the same access to the data storage media associated with the logical unit that contains the copy manager as any other application client for the purposes of processing third-party copy commands and operations (see 5.16.3);
c) the copy manager shall be able to access all other device servers and copy managers located in the same SCSI target device as the copy manager; and
d) the copy manager may or may not be able to access device servers and copy managers located in other SCSI target devices.

If the copy manager is contained in a standalone SCSI target device whose only purpose is to contain logical units that contain copy managers, then the following requirements shall apply:

a) the PERIPHERAL DEVICE TYPE field in the standard INQUIRY data (see 6.6.2) shall indicate that the device is a processor type device (see SPC-2);
b) the device server may not implement any of the specialized processor device type commands (e.g., SEND); and

c) the copy manager shall be able to access device servers and copy managers in at least one other SCSI target device for the purposes of processing third-party copy commands and operations.

Within a SCSI target device, a copy manager shall not have access to SCSI ports that are not accessible to the device server in the same logical unit (e.g., due to restrictions imposed by asymmetric logical unit access features (see 5.15.2)), but a copy manager may have access to non-SCSI data transports. As a result:

a) if a copy manager has access to only one SCSI port, then the copy operations performed by the copy manager are limited to a single SCSI domain;
b) if a copy manager has access to multiple SCSI ports some of which are in different SCSI domains, then the copy operations performed by the copy manager are limited to the accessible SCSI domains, but may transfer data from one SCSI domain to another; or
c) if a copy manager has access to non-SCSI data transports, then the copy operations performed by the copy manager may transfer data from one SCSI domain to another SCSI domain that is not accessible to the device server associated with the copy manager.

In response to interactions with other copy managers, a copy manager may process requests for copy operations even if the copy manager does not implement an equivalent third-party copy command (see 5.16.3).

EXAMPLE – Suppose a copy manager has access to a non-SCSI transport that transfers data in message IUs instead of SCSI reads and writes. Even though no SCSI commands are performed, the copy manager is allowed to process a write command as an EXTENDED COPY(LID4) command (see 6.4) with the data to be written contained in the inline data portion of the parameter data. The copy manager is also allowed to process a read command as an EXTENDED COPY(LID4) command that generates held data (see 5.16.4.5) followed by a RECEIVE COPY DATA(LID4) command (see 6.20) to retrieve the held data. The same copy manager may indicate that it supports only the POPULATE TOKEN command (see SBC-3), WRITE USING TOKEN command (see SBC-3), and RECEIVE ROD TOKEN INFORMATION command (see 6.28 and SBC-3).

Figure 17 shows examples copy manager configurations and third-party copies they may perform.

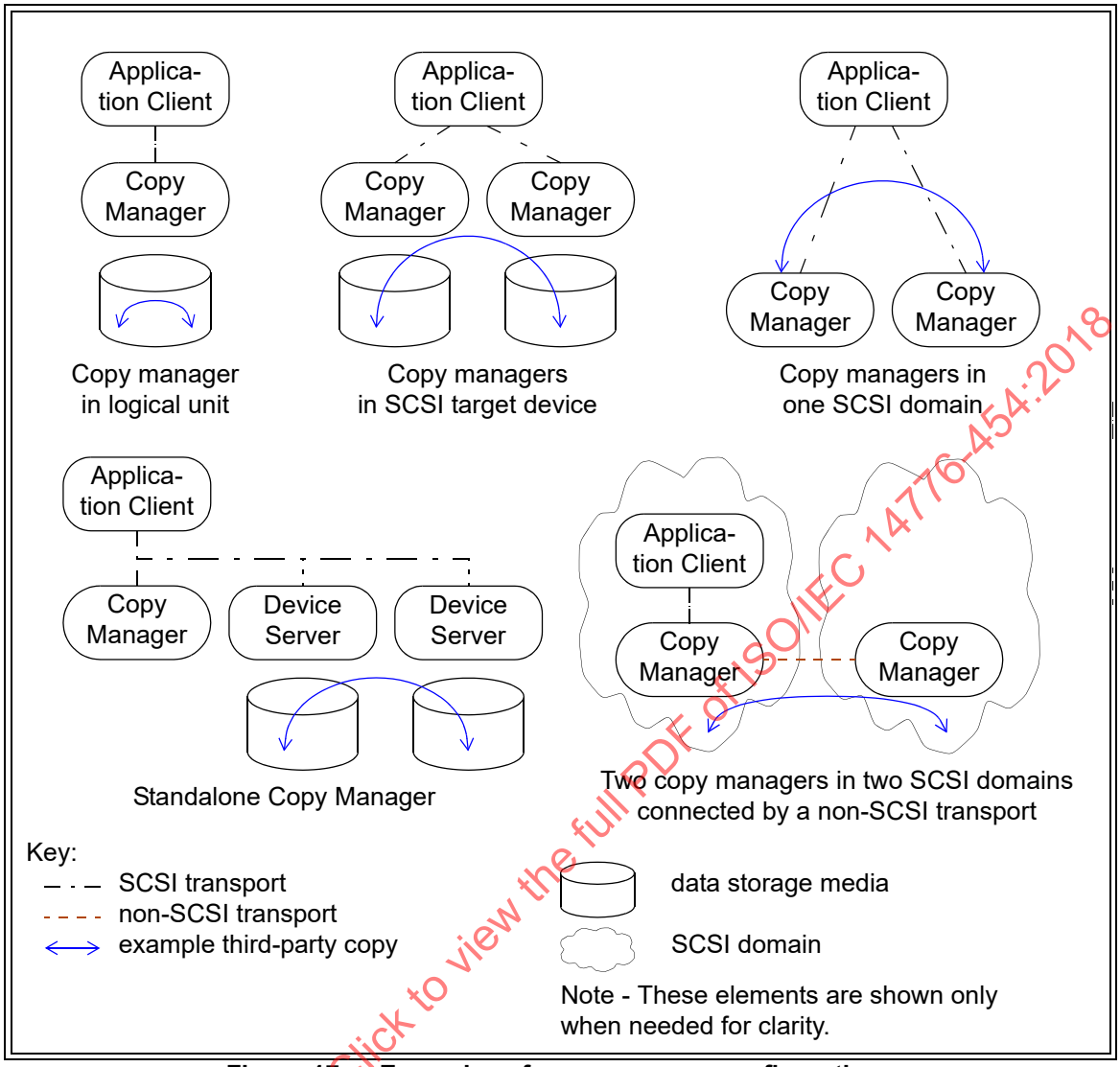


Figure 17 — Examples of copy manager configurations

5.16.3 Third-party copy commands

The commands processed by copy managers (i.e., third-party copy commands) are shown in table 108.

Table 108 — Third-party copy commands

Command	Operation code [a]	Command type	Reference
COPY OPERATION ABORT	83h/1Ch	abort	6.3
EXTENDED COPY(LID4)	83h/01h	originate	6.4
EXTENDED COPY(LID1)	83h/00h	originate	6.5
POPULATE TOKEN	83h/10h	originate	SBC-3
RECEIVE COPY DATA(LID4)	84h/06h	retrieve	6.20
RECEIVE COPY DATA(LID1)	84h/01h	retrieve	6.21
RECEIVE COPY OPERATING PARAMETERS [b]	84h/03h	retrieve	6.22
RECEIVE COPY FAILURE DETAILS(LID1)	84h/04h	retrieve	6.23
RECEIVE COPY STATUS(LID4)	84h/05h	retrieve	6.24
RECEIVE COPY STATUS(LID1)	84h/00h	retrieve	6.25
RECEIVE ROD TOKEN INFORMATION	84h/07h	retrieve	6.28
REPORT ALL ROD TOKENS	84h/08h	retrieve	6.31
WRITE USING TOKEN	83h/11h	originate	SBC-3
Reserved	all others [c]		

Key:

abort	third-party copy command that aborts a specified copy operation
originate	third-party copy command that requests the processing of a copy operation
retrieve	third-party copy command that retrieves the results (e.g., status data or held data) for a previously originated copy operation

[a] All copy manager commands are 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.

[b] New operating parameters are no longer being added to the parameter data returned by the RECEIVE COPY OPERATING PARAMETERS command. The Third-party Copy VPD page (see 7.8.17) provides the most complete information about how a copy manager operates and the functions that it supports.

[c] All service actions of operation code 83h and operation code 84h not shown in this table are reserved.

Copy manager support for third-party copy commands is optional, but support for some third-party copy commands and features may require support for other third-party copy commands as shown in table 109.

Table 109 — Mandatory copy manager command support requirements

Supported command	Optional command features supported		Copy manager support for the following commands is mandatory
	Held data [a]	R_TOKEN bit [b]	
COPY OPERATION ABORT	n/a	n/a	RECEIVE COPY STATUS(LID4)
EXTENDED COPY(LID4)	n/a	n/a	RECEIVE COPY STATUS(LID4)
EXTENDED COPY(LID4)	yes	n/a	RECEIVE COPY DATA(LID4)
EXTENDED COPY(LID1)	n/a	n/a	RECEIVE OPERATING PARAMETERS RECEIVE COPY FAILURE DETAILS(LID1) RECEIVE COPY STATUS(LID1)
EXTENDED COPY(LID1)	yes	n/a	RECEIVE COPY STATUS(LID1) RECEIVE COPY DATA(LID1)
EXTENDED COPY(LID4) or EXTENDED COPY(LID1)	n/a	yes	RECEIVE ROD TOKEN INFORMATION
POPULATE TOKEN	n/a	n/a	See SBC-3
WRITE USING TOKEN	n/a	n/a	

[a] Held data (see 5.16.4.5) refers to support for the ability to hold some or all of the data processed by a copy operation for later transfer to the application client (e.g., support for the stream →discard& application client segment descriptor (see 6.4.6.8)).

[b] R_TOKEN bit refers to support for the ability to create ROD tokens that are later returned to the application client (i.e., support for the R_TOKEN bit being set to one in the ROD CSCD descriptor (see 6.4.5.9)).

If a copy manager supports a third-party copy command in which the IMMED bit, if any, is allowed to be set to one, then the copy manager shall support the COPY OPERATION ABORT command (see 6.3).

5.16.4 Third-party copy command usage

5.16.4.1 Prior to sending a third-party copy command

Before the copy manager is instructed to transfer data, the application client requesting the data transfers shall take any necessary actions required to prepare the copy sources and copy destinations (see 5.16.1). Such preparatory actions include but are not limited to:

a) loading tapes;
b) sending media changer commands;
c) sending MODE SELECT commands, including MODE SELECT commands that:
   A) disable reporting of recovered errors by a block CSCD by setting the PER bit to zero in the Read-Write Error Recovery mode page (see SBC-3); and/or
   B) disable reporting of thin provisioning threshold events by a block CSCD by setting the LBPERE bit to zero in the Read-Write Error Recovery mode page;
d) sending persistent reservation commands; and/or
e) sending tape positioning commands.

After all preparatory actions have been completed, the third-party copy command (e.g., the EXTENDED COPY(LID4) command) should be sent to the copy manager to originate the copy operation (see 5.16.4.3).

5.16.4.2 List identifiers for third-party copy commands

A third-party copy command (see 5.16.3) may:

a) take a long time to complete;
b) have substantial command-specific data to return upon the completion of processing (e.g., held data (see 5.16.4.4)); and
c) be processed as a background operation (e.g., in response to an IMMED bit being set to one (see 5.16.4.3)).

List identifiers allow an application client to specify the copy operation (see 5.16.4.3) for which a monitoring function or management function is to be performed, as follows:

1) the application client specifies the list identifier by which a third-party copy operation is to be identified in the command or parameter list that originates the copy operation; and
2) the application client specifies the same list identifier in any command that monitors or manages the copy operation:
   A) starting from the time the copy operation is originated; and
   B) continuing until:
      a) all data associated with the copy operation has been delivered to the application client; or
      b) the application client specifies that the associated data is to be discarded.

List identifiers are specified in a LIST IDENTIFIER field whose location depends on the third-party command being processed.

Unless otherwise specified, the LIST IDENTIFIER field contains a value that uniquely identifies a copy operation among all those being processed that were received on a specific I_T nexus. If the copy manager detects a duplicate list identifier value, then the copy manager shall terminate the originating third-party copy command (see table 108 in 5.16.3) with CHECK CONDITION status, with the sense key set to ILLEGAL REQUEST, and the additional sense code set to OPERATION IN PROGRESS.

The following LIST IDENTIFIER field formats are defined by this standard:

a) an 8-bit LIST IDENTIFIER field (e.g., the RECEIVE COPY STATUS(LID1) command (see 6.24)); and
b) a 32-bit LIST IDENTIFIER field (e.g., the RECEIVE COPY DATA(LID4) command (see 6.20)).

If a copy manager supports any third-party copy commands (see 5.16.3) that contain a 32-bit LIST IDENTIFIER field, then the copy manager shall treat all list identifiers as 32-bit quantities, regardless of the field size used to specify the list identifier (e.g., a list identifier of 000000FDh shall specify the same copy operation whether it is present as FDh in an 8-bit field or as 000000FDh in a 32-bit field).

The results of specifying a list identifier from a third-party copy command that originates a copy operation (see table 108 in 5.16.3) using 32-bit list identifier values in a third-party copy command that retrieves copy operation data using an 8-bit list identifier are outside the scope of this standard.

5.16.4.3 Third-party copy commands and operations

Third-party copy commands that originate copy operations (see table 108 in 5.16.3) may support the IMMED bit to allow the application client to specify that the processing of a copy operation continue after the processing of the originating command has been completed.

If a third-party copy command supports an IMMED bit and that IMMED bit is set to one, the copy manager:

1) shall return CHECK CONDITION status if any errors are detected in the CDB;
2) shall transfer all of the parameter list, if any, to the copy manager;
3) may validate the parameter list and return CHECK CONDITION status if errors are detected;
4) shall complete the command with GOOD status; and

5) shall complete processing of all specified copy operations as a background operation (see SAM-5).

A third-party copy command definition may place additional restrictions on the use of the IMMED bit.

If the IMMED bit is not supported by a command, then the copy manager shall process that command as if the IMMED bit were set to zero.

If the IMMED bit, if any, is set to:

a) zero, then the copy manager shall not complete processing of the third-party copy command until the copy manager has completed processing of the copy operation originated by the command (i.e., the copy manager processes the copy operation in the foreground); or
b) one, then the copy manager may complete processing of the third-party copy command that originated the copy operation before completing the copy operation (i.e., the copy manager processes the copy operation in the background). Copy operations that are processed in the background shall not generate deferred errors (see SAM-5) for the errors encountered, if any, during this processing. Instead, the error information (e.g., status, sense key, additional sense code) shall be made available to the application client through use of one of the commands described in 5.16.4.4.

5.16.4.4 Monitoring progress of and retrieving results from third-party copy commands

The RECEIVE COPY STATUS(LID4) command (see 6.24) returns information about the current processing status of the copy operation (see 5.16.4.3) specified by the list identifier (see 5.16.4.2). The parameter data for the following commands contain the parameter data for the RECEIVE COPY STATUS(LID4) command as a header (i.e., the shared third-party copy status header):

a) RECEIVE COPY DATA(LID4) (see 6.24); and
b) RECEIVE ROD TOKEN INFORMATION (see 6.28).

The status information for a copy operation (see 5.16.4.3) shall be available at any time the copy operation is in progress (i.e., whenever the COPY OPERATION STATUS field is set to 10h, 11h, or 12h).

After the completion of a copy operation (see 5.16.4.3) (i.e., whenever the COPY OPERATION STATUS field is set to 01h, 02h, 03h, or 60h), the copy manager shall preserve all status information for that copy operation for a vendor specific period of time. The copy manager shall discard the status information in which the COPY OPERATION STATUS field is set to 01h, 02h, 03h, or 60h if:

a) another third-party copy command that originates a copy operation (see table 108 in 5.16.3) is received on the same I_T nexus and the list identifier matches the list identifier associated with the status information;
b) the copy manager detects a logical unit reset condition or I_T nexus loss condition (see SAM-5); or
c) the copy manager requires the resources used to preserve the status information.

The copy manager may discard the sense data in the status information in which the COPY OPERATION STATUS field is set to 01h, 02h, 03h, or 60h after that data has been returned by a RECEIVE COPY STATUS(LID4) command received on the same I_T nexus with a matching list identifier.

The copy manager shall not discard the status information for a copy operation (see 5.16.4.3) in response to:

a) an ABORT TASK task management function (see SAM-5);
b) an ABORT TASK SET task management function;
c) a CLEAR TASK SET task management function;
d) a PERSISTENT RESERVE OUT command with PREEMPT AND ABORT service action (see 6.16); or
e) a COPY OPERATION ABORT command (see 6.3).

In the parameter data for all the commands with the shared third-party copy status header, the contents of the COPY OPERATION STATUS field (see table 252 in 6.24) indicate the current processing status of the specified copy operation. Unless otherwise specified, the contents of the COPY OPERATION STATUS field are not affected by whether the copy operation is being or was performed in the foreground or background.

Although the information provided is not as complete, the following commands allow the monitoring of progress and retrieval of results for certain copy operations in a way that is compatible with SPC-3 but do not return the shared third-party copy status header:

a) RECEIVE COPY STATUS(LID1) (see 6.25);
b) RECEIVE COPY DATA(LID1) (see 6.21); and
c) RECEIVE COPY FAILURE DETAILS(LID1) (see 6.23).

5.16.4.5 Held data

A copy operation may hold some or all of the data it processes for retrieval by the application client (e.g., the processing defined for the EXTENDED COPY command stream→stream&application client segment descriptor (see 6.4.6.5)). Held data is retrieved using the RECEIVE COPY DATA(LID4) command (see 6.20) or the RECEIVE COPY DATA(LID1) command (see 6.21).

If a copy manager supports any third-party copy commands (see 5.16.3) or an EXTENDED COPY command copy function (see 6.4.6) capable of holding data for retrieval by the application client, then the copy manager:

a) shall support the RECEIVE COPY DATA(LID4) command, if any third-party copy commands are supported that originate copy operations (see table 108 in 5.16.3) with 32-bit list identifiers (see 5.16.4.2);
b) shall support the RECEIVE COPY DATA(LID1) command, if any third-party copy commands are supported that originate copy operations with 8-bit list identifiers;
c) shall support either the Third-party Copy VPD page Held Data descriptor (see 7.8.17.12) or the RECEIVE COPY OPERATING PARAMETERS command (see 6.22); and
d) should support the Third-party Copy VPD page Held Data descriptor.

After the completion of a third-party copy command that originates a copy operation (see table 108 in 5.16.3), the copy manager shall preserve all held data for a vendor specific period of time. The application client should retrieve the held data (e.g., by sending a third-party copy command that retrieves the results of a previously originated copy operation (see table 108 in 5.16.3)) as soon as possible after the completion of the copy operation to ensure that the data is not discarded by the copy manager. The copy manager shall discard the held data:

a) after all the held data for a specific copy operation has been successfully transferred to the application client;
b) if a RECEIVE COPY DATA(LID4) command (see 6.20) or a RECEIVE COPY DATA(LID1) command (see 6.21) has been received on the same I_T nexus with a matching list identifier (see 5.16.4.2), with the ALLOCATION LENGTH field set to zero;
c) if another third-party copy command that originates a copy operation is received on the same I_T nexus and the list identifier matches the list identifier associated with the held data;
d) if the copy manager detects a logical unit reset condition or I_T nexus loss condition (see SAM-5); or
e) if the copy manager requires the resources used to preserve the held data.

The copy manager shall not discard the held data in response to:

a) an ABORT TASK task management function (see SAM-5);
b) an ABORT TASK SET task management function;
c) a CLEAR TASK SET task management function;
d) a PERSISTENT RESERVE OUT command with PREEMPT AND ABORT service action (see 6.16); or
e) a COPY OPERATION ABORT command (see 6.3).

The copy manager indicates the minimum amount of held data it supports in the HELD DATA LIMIT field that is returned in:

a) the Held Data descriptor (see 7.8.17.12) in the Third-party Copy VPD page (see 7.8.17); or
b) the parameter data for a RECEIVE COPY OPERATING PARAMETERS command (see 6.22).

The HELD DATA LIMIT field indicates the length, in bytes, of the minimum amount of data the copy manager shall hold for return to the application client. If the processing of a copy operation 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 portion of the copy operation. The discarding of held data bytes shall not be considered an error.

The held data discarded (HDD) bit indicates whether held data has been discarded for the copy operation specified by a list identifier (see 5.16.4.2). If the HDD bit is set to one, held data has been discarded. If the HDD bit is set to zero, held data has not been discarded. The HDD bit is in the parameter data for:

a) the RECEIVE COPY DATA(LID4) command (see 6.20); and
b) the RECEIVE COPY STATUS(LID1) command (see 6.25).

5.16.4.6 Aborting third-party copy commands and copy operations

A task manager shall ensure that all commands and data transfers generated by a third-party copy operation have been terminated and are no longer transferring data before allowing the completion of the task management function or command (e.g., the PERSISTENT RESERVE OUT command with PREEMPT AND ABORT service action) in response to the following causes for the termination of a third-party copy command:

a) an ABORT TASK task management function (see SAM-5);
b) an ABORT TASK SET task management function (see SAM-5);
c) a CLEAR TASK SET task management function (see SAM-5);
d) a PERSISTENT RESERVE OUT command with PREEMPT AND ABORT service action (see 5.12.11.2.6); and
e) other SCSI device conditions described in SAM-5 that abort copy operations.

5.16.4.7 The COPY OPERATION ABORT command

Aborting a copy operation is not always possible with a task management function (e.g., a copy operation (see 5.16.4.3) that was originated by a third-party copy command with the IMMED bit set to one). The COPY OPERATION ABORT command (see 6.3) provides an abort capability that is equivalent to an ABORT TASK task management function for any copy operation that is able to be specified with a list identifier (see 5.16.4.2) without regard for whether the copy operation is being performed in the foreground or background (see 5.16.4.3).

Before completing the COPY OPERATION ABORT command, the copy manager shall ensure that all commands and data transfers generated by that copy operation have been terminated and are no longer transferring data.

All foreground copy operations and background copy operations associated with the specified list identifier shall be terminated with CHECK CONDITION status with the sense key set to COPY ABORTED and the additional sense code set to COMMANDS CLEARED BY DEVICE SERVER (see 5.16.4.3).

5.16.5 Responses to the conditions that result from SCSI events

The effects on copy operations that are produced by the conditions that result from SCSI events (see SAM-5) are shown in table 110.

Table 110 — Responses to the conditions that result from SCSI events

Condition that results from a SCSI event	Effects of condition on copy operations
Power on Hard reset Logical unit reset Power loss expected	All foreground and background copy operations shall be aborted as if a COPY OPERATION ABORT command (see 6.3) has been received for each copy operation.
I_T nexus loss	All foreground and background copy operations that require access to the affected I_T Nexus shall be aborted as if a COPY OPERATION ABORT command has been received for each copy operation.

5.16.6 RODs and ROD tokens

5.16.6.1 RODs and ROD related tokens overview

A ROD provides a way to represent multiple bytes of data that may or may not be addressable as the content of some media (e.g., one or more ranges of contiguous logical blocks that may be addressed as LBAs or maintained as a point in time copy (see 5.16.6.2.3) of the contents of the specified LBA ranges). Each ROD is created and maintained by a copy manager as specified by this standard.

The types of RODs are described in 5.16.6.2. The process of creating and populating a ROD is described in 5.16.6.3.

A ROD token is a token that a copy manager transfers to an application client to represent a specified ROD outside the copy manager. Application clients may use ROD tokens as follows:

a) if a valid ROD token is received by the copy manager that created it, then that copy manager is able to associate the ROD token with the original ROD and process the data represented by the ROD in specified ways (e.g., the ways specified by segment descriptors in an EXTENDED COPY command);
b) if a ROD token is received by a copy manager other than the copy manager that created it, then the receiving copy manager may be able to process the data that the ROD token represents by communicating with the copy manager that created the ROD token. The communications between the copy managers shall conform to the models described in this standard but are not required to use the commands or data transfer mechanisms described in this standard; and
c) if a ROD token is transferred from one application client to another by a means outside the scope of this standard, then the second application client may use the ROD token in any of the ways described in this subclause.

The format of a ROD token is defined in 5.16.6.4. The interval of time during which the ROD token remains valid is called the ROD token lifetime (see 5.16.6.7).

ROD management tokens (see 5.16.6.4) are used to manage (e.g., delete) ROD tokens.

5.16.6.2 ROD types

5.16.6.2.1 ROD types overview

A copy manager uses the ROD types shown in table 111.

Table 111 — ROD types

Type code ranges	Type code range applicability	Type codes defined by this standard	Description	Reference
0000 0000h	Copy manager internal ROD	0000 0000h	ROD type specified by ROD token	command definition [a]
0000 0001h to 0000 FFFFh	Reserved			
0001 0000h to FEFF FFFFh	Any device type	0001 0000h	Access upon reference	5.16.6.2.2
		0080 0000h	Point in time copy – default	5.16.6.2.3.2
		0080 0001h	Point in time copy – change vulnerable	5.16.6.2.3.3
		0080 0002h	Point in time copy – persistent	5.16.6.2.3.4
		0080 FFFFh	Point in time copy – any	5.16.6.2.3.5
		all others in this range	Reserved	
FF00 0000h to FFFF FFEFh	Device type specific			applicable command standard
FFFF FFF0h to FFFF FFFFh	Vendor specific ROD token body and ROD token extension (see 5.16.6.4 and 5.16.6.5)			

[a] Some commands (e.g., an EXTENDED COPY command with the ROD CSCD descriptor (see 6.4.5.9) in the parameter list (see 5.16.7.1)) use this ROD type code when processing a ROD token received by the copy manager. Details of such usage are specific to the command.

If a third-party copy command requests the creation of a ROD and the copy manager does not have sufficient resources to create or maintain the ROD, then the copy manager shall terminate the command with CHECK CONDITION status, with the sense key set to ILLEGAL REQUEST, and the additional sense code set to INSUFFICIENT RESOURCES TO CREATE ROD.

A copy manager shall indicate the types of RODs it supports in the Supported ROD Types third-party copy descriptor (see 7.8.17.9) in the Third-party Copy VPD page.

5.16.6.2.2 Access upon reference type RODs

Access upon reference type RODs are RODs in which the bytes are represented by addressing information (e.g., their LBAs for block devices). The copy manager that creates the ROD is not required to:

a) access the bytes until the ROD is used as a copy source or copy destination; or
b) maintain any specific data contents in association with the ROD.

An access upon reference type ROD shall remain valid as long as the addresses for the bytes remain valid (e.g., a MODE SELECT command that decreases the number of logical blocks on a block device in a way that eliminates one of the LBAs in an access upon reference type ROD causes the ROD to become invalid).

Invalidating a ROD in this way may cause it to become invalid before its specified lifetime (see 5.16.6.7) has elapsed.

5.16.6.2.3 Point in time copy RODs

5.16.6.2.3.1 Point in time copy RODs overview

Point in time copy RODs are RODs for which the data returned when the ROD is used as a copy source is the data that was present when the ROD was populated (see 5.16.6.3). The copy manager that creates the ROD shall maintain the data that populates the ROD for as long as the ROD remains valid (see 5.16.6.7). The method that the copy manager uses to maintain the data is outside the scope of this standard.

If the copy manager is unable to maintain the data that populates a point in time copy ROD, the copy manager shall invalidate this type of ROD. Invalidating a ROD in this way may cause it to become invalid before its specified lifetime (see 5.16.6.7) has elapsed.

If a third-party copy command that originates a copy operation (see table 108 in 5.16.3) specifies a point in time copy ROD as a copy destination, then the copy manager shall terminate the copy operation (see 5.16.4.3) originated by the command with CHECK CONDITION status, with the sense key set to ILLEGAL REQUEST, and the additional sense code set to INVALID FIELD IN PARAMETER LIST.

5.16.6.2.3.2 Point in time copy – default type RODs

A point in time copy – default type ROD is a point in time copy ROD (see 5.16.6.2.3.1) for which the method that maintains the data that populates the ROD is vendor specific.

5.16.6.2.3.3 Point in time copy – change vulnerable type RODs

A point in time copy – change vulnerable type ROD is a point in time copy ROD (see 5.16.6.2.3.1) for which the copy manager shall create the ROD using the method that consumes the fewest resources without regard for whether this increases the chance that the ROD may become invalid before its specified lifetime (see 5.16.6.7) has elapsed.

The copy manager may invalidate this type of ROD if the data that populated that ROD when it was created is modified (e.g., by a write command) or becomes invalid (e.g., a MODE SELECT command decreases the number of blocks on a logical blocks device in a way that eliminates one of the LBAs in the ROD).

5.16.6.2.3.4 Point in time copy – persistent type RODs

A point in time copy – persistent type ROD is a point in time copy ROD (see 5.16.6.2.3.1) for which the copy manager shall maintain the data present when the ROD was populated for the ROD's specified lifetime (see 5.16.6.7) regardless of modifications to the data that populated the ROD when it was created.

A point in time copy – persistent type ROD may become invalidated before the specified lifetime for reasons other than a modification to the data that populated the ROD when it was created (see 5.16.6.7).

5.16.6.2.3.5 Point in time copy – any type RODs

A point in time copy – any type ROD is a point in time copy ROD (see 5.16.6.2.3.1) for which the copy manager shall create the ROD using one of the following ROD types, in order of preference:

1) point in time copy – persistent (see 5.16.6.2.3.4); or
2) point in time copy – change vulnerable (see 5.16.6.2.3.3).

If a ROD token (see 5.16.6.4 and 5.16.6.5) is returned for a point in time copy – any type ROD, the value in the ROD TOKEN field shall indicate the type of ROD the copy manager created (e.g., 0080 0001h for point in time copy – change vulnerable).

5.16.6.3 Populating a ROD or ROD token

The content of a ROD is established when the ROD is populated. Details of how a ROD is populated are defined by the command or commands that cause the ROD to become populated.

After a ROD is populated, it may be:

a) used by later processing steps defined by the command that caused the ROD to be populated (i.e., used inside the copy manager that created the ROD); or
b) used to create a ROD token (see 5.16.6.4) that may be used by application clients and copy managers in the ways described in 5.16.6.1.

Commands are not required to provide a means to use a ROD inside the copy manager, but any ROD tokens that a copy manager creates during the processing of a command shall be made available for transfer to the application client using the RECEIVE ROD TOKEN INFORMATION command (see 6.28).

After the completion of a copy operation (see 5.16.4.3), the copy manager shall preserve all created ROD tokens for a vendor specific period of time. The application client should retrieve the ROD tokens using the RECEIVE ROD TOKEN INFORMATION command as soon as possible after the completion of the copy operation to ensure that the data is not discarded by the copy manager. The copy manager shall discard the parameter data for the created ROD tokens:

a) after all the ROD tokens created by a specific copy operation (see 5.16.4.3) have been transferred without errors to the application client;
b) if a RECEIVE ROD TOKEN INFORMATION command has been received on the same I_T nexus with a matching list identifier (see 5.16.4.2), with the ALLOCATION LENGTH field set to zero;
c) if another a third-party command that originates a copy operation (see table 108 in 5.16.3) is received on the same I_T nexus and the list identifier matches the list identifier associated with the ROD tokens;
d) if the copy manager detects a logical unit reset condition or I_T nexus loss condition (see SAM-5); or
e) if the copy manager requires the resources used to preserve the data.

The copy manager shall not discard the data returned by a RECEIVE ROD TOKEN INFORMATION command in response to:

a) an ABORT TASK task management function (see SAM-5); or
b) a COPY OPERATION ABORT command (see 6.3).

The format of a ROD token is defined in 5.16.6.4. The interval of time during which the ROD token remains valid is called the ROD token lifetime (see 5.16.6.7).

If a third-party copy command attempts to populate a ROD with the same data more than one time (e.g., specifying the same LBA twice), then the copy manager shall terminate the copy operation (see 5.16.4.3) originated by the command with CHECK CONDITION status, with the sense key set to ILLEGAL REQUEST, and the additional sense code set to INVALID FIELD IN PARAMETER LIST.

EXAMPLE 1 – The EXTENDED COPY(LID4) command (see 6.4) defines a CSCD descriptor that establishes an internal ROD and segment descriptors that populate that ROD. The populated ROD may be used by other segment descriptors, and may be returned as a ROD token using the methods described in this subclause.

EXAMPLE 2 – The POPULATE TOKEN command (see SBC-3) establishes and populates a ROD whose only use during the processing of the POPULATE TOKEN command is in the creation of a ROD token that is returned using the methods described in this subclause.

5.16.6.4 ROD token format

The ROD token format is shown in table 112.

Table 112 — ROD token format

Bit / Byte	7	6	5	4	3	2	1	0
	ROD token header							
0	(MSB)							
•••				ROD TYPE				
3								(LSB)
4				Reserved				
5								
6	(MSB)							
7				ROD TOKEN LENGTH (n-7)				(LSB)
	ROD token body (if any)							
8	(MSB)							
•••				COPY MANAGER ROD TOKEN IDENTIFIER				
15								(LSB)
16								
•••				CREATOR LOGICAL UNIT DESCRIPTOR				
47								
48	(MSB)							
•••				NUMBER OF BYTES REPRESENTED				
63								(LSB)
64								
•••				ROD token type specific data				
95								
	ROD token extension (if any)							
96								
•••				Device type specific data				
127								
128								
•••				ROD token type and copy manager specific data				
n								

Every ROD token shall include the ROD TYPE field and the ROD TOKEN LENGTH field. Except for the ROD TYPE field and the ROD TOKEN LENGTH field, the definition for any ROD token format may not include the other fields shown in table 112. The COPY MANAGER ROD TOKEN IDENTIFIER field, CREATOR LOGICAL UNIT DESCRIPTOR field, NUMBER OF BYTES REPRESENTED field, and ROD token type specific data bytes shall be included in the ROD token body if:

a) information about the ROD token is returned by the REPORT ALL ROD TOKENS command (see 6.31); or
b) any fields are defined in the ROD token extension.

EXAMPLE – The block device zero ROD token (see SBC-3) has a value selected from table 111 (see 5.16.6.2.1) in the ROD TYPE field, and the ROD TOKEN LENGTH field set to 01F8h. All bytes in the ROD token body and ROD token extension are reserved. This is a valid ROD token format because the block device zero ROD token is not returned by the REPORT ALL ROD TOKENS command.

Commands that manage ROD tokens (e.g., the REPORT ALL ROD TOKENS command (see 6.31)) use a ROD management token that consists of the ROD token header and the ROD token body in the ROD token being managed (i.e., the ROD token extension is omitted in ROD management tokens). The ROD TOKEN LENGTH field is not modified when a ROD management token is built from a ROD token (i.e., the ROD TOKEN LENGTH field does not indicate the length of the ROD management token).

The ROD TYPE field (see table 111 in 5.16.6.2.1) indicates the use and content of the ROD token bytes that follow the header.

The ROD TOKEN LENGTH field indicates the number of bytes that follow in the ROD token. The minimum size of a ROD token shall be 512 bytes (i.e., the minimum value in the ROD TOKEN LENGTH field is 01F8h). Bytes in unused ROD token fields shall be reserved (e.g., in a ROD token that does not include the ROD token body, the bytes assigned to the COPY MANAGER ROD TOKEN IDENTIFIER field, CREATOR LOGICAL UNIT DESCRIPTOR field, and NUMBER OF BYTES REPRESENTED field are reserved).

If present, the COPY MANAGER ROD TOKEN IDENTIFIER field contains a value that differentiates this ROD token from all other valid ROD tokens created by and known to a specific copy manager. No two ROD tokens known to a specific copy manager shall have the same value in the COPY MANAGER ROD TOKEN IDENTIFIER FIELD. After a ROD token becomes invalid, the copy manager should not reuse the copy manager ROD token identifier value from that ROD token in another ROD token for as long as possible, and shall not reuse that value until at least 50 000 ROD tokens have been created by that copy manager.

If present, the CREATOR LOGICAL UNIT DESCRIPTOR field contains an Identification Descriptor CSCD descriptor (see 6.4.5.6) for the logical unit that contains the copy manager that created the ROD token. That Identification Descriptor CSCD descriptor shall contain a designation descriptor with the DESIGNATOR TYPE field set to:

a) 2h (i.e., EUI-64-based); or
b) 3h (i.e., NAA).

If present, the NUMBER OF BYTES REPRESENTED field is set to the total number of bytes that the ROD token represents. If the number of bytes represented by the ROD token is unknown or greater than FFFF FFFF FFFF FFFF FFFF FFFF FFFF FFFEh, then the NUMBER OF BYTES REPRESENTED field shall be set to FFFF FFFF FFFF FFFF FFFF FFFF FFFF FFFFh. Command standards may define restrictions on the contents of the NUMBER OF BYTES REPRESENTED field.

If present, the ROD token type specific data contains one or more fields whose contents are based on the ROD token type.

If present, the device type specific data contains one or more fields whose contents are defined by the command standard for the device type associated with the copy manager that created the ROD token.

If present, the ROD token type and copy manager specific data contains one or more fields whose contents are dependent on the ROD token type and the copy manager that created the ROD token.

5.16.6.5 Generic ROD tokens

5.16.6.5.1 Generic ROD token format

Unless a different format is defined, ROD tokens for the ROD types other than 00h (see table 111 in 5.16.6.2.1) have the format shown in table 113.

Table 113 — Generic ROD token format

Bit / Byte	7	6	5	4	3	2	1	0
	ROD token header							
0	(MSB)			ROD TYPE				
•••								
3								(LSB)
4				Reserved				
5								
6	(MSB)			ROD TOKEN LENGTH (n-7)				
7								(LSB)
	ROD token body							
8	(MSB)			COPY MANAGER ROD TOKEN IDENTIFIER				
•••								
15								(LSB)
16				CREATOR LOGICAL UNIT DESCRIPTOR				
•••								
47								
48	(MSB)			NUMBER OF BYTES REPRESENTED				
•••								
63								(LSB)
64				Reserved				
•••								
95								
	ROD token extension							
96				Device type specific data				
•••								
127								
128				TARGET DEVICE DESCRIPTOR				
•••								
t								
t+1				EXTENDED ROD TOKEN DATA				
•••								
n								

The ROD TYPE field is defined in 5.16.6.4 and the coded values that may be used in the ROD TYPE field are defined in 5.16.6.2.1. The ROD TYPE field shall be set to one of the values shown in table 114. The ROD TYPE field shall not be set to 0080_FFFFh (i.e., point in time copy – any) (see 5.16.6.2.3.5).

Table 114 — ROD TYPE field in generic ROD

Code	Description	ROD TOKEN LENGTH field contents	TARGET DEVICE DESCRIPTOR FIELD size (in bytes)	ROD type Reference
0001 0000h	Access upon reference	01F8h	128	5.16.6.2.2
0080 0000h	Point in time copy – default	01F8h	128	5.16.6.2.3.2
0080 0001h	Point in time copy – change vulnerable	01F8h	128	5.16.6.2.3.3
0080 0002h	Point in time copy – persistent	01F8h	128	5.16.6.2.3.4
all others	see table 111 in 5.16.6.2.1			

The ROD TOKEN LENGTH field is defined in 5.16.6.4, and shall be set to the value shown in table 114 based on the contents of the ROD TYPE field.

The COPY MANAGER ROD TOKEN IDENTIFIER field, CREATOR LOGICAL UNIT DESCRIPTOR field, and NUMBER OF BYTES REPRESENTED field are defined in 5.16.6.4.

The device type specific data is specified by the command standard for the peripheral device type indicated by the CREATOR LOGICAL UNIT DESCRIPTOR field (see 5.16.6.4) (e.g., for peripheral device type 00h, see SBC-3).

The TARGET DEVICE DESCRIPTOR field contains a designation descriptor for the SCSI target device (see 7.8.6) that contains the logical unit indicated by the descriptor in CREATOR LOGICAL UNIT DESCRIPTOR field. The designation descriptor shall have the ASSOCIATION field set to 10b (i.e., SCSI target device) and the DESIGNATOR TYPE field set to:

a) 2h (i.e., EUI-64-based);
b) 3h (i.e., NAA); or
c) 8h (i.e., SCSI name string).

The EXTENDED ROD TOKEN DATA field shall contain at least 256 bits of secure random number material (see 4.4) generated when the ROD token was created, and may contain information that is used by the copy manager that created the ROD token to process the ROD token (e.g., an expiration time in a vendor specific format). The EXTENDED ROD TOKEN DATA field should not contain data that enables an entity outside the SCSI target device in which the ROD token was created to determine the data (e.g., LBAs or logical blocks) that the ROD token represents.

Those portions of the EXTENDED ROD TOKEN DATA field that do not contain defined values should be set to unpredictable random values. The contents of the EXTENDED ROD TOKEN DATA field may be defined to contain:

a) zero or more values that are based on the ROD represented by the generic ROD token (e.g., pointers, expiration time); and
b) zero or more values used by the copy manger to determine whether the generic ROD token is valid (e.g., cryptographic integrity check values, message authentication codes, digital signatures).

5.16.6.5.2 Validating generic ROD tokens

5.16.6.5.2.1 Overview of validating generic ROD tokens

A copy manager shall ensure that a generic ROD token (see 5.16.6.5.1) is valid before performing any action based on the contents of that ROD token. The determination of whether a generic ROD token is valid shall be performed only by the copy manager that created that ROD token. If the processing copy manager is in the same SCSI target device as the copy manager that created the ROD token, then the two copy managers may exchange information using a method outside the scope of this standard to accomplish the required validation.

If a copy manager is unable to validate the generic ROD token, then no data shall be transferred by the command that contains the invalid ROD token, and the copy operation (see 5.16.4.3) shall be terminated as described in 5.16.6.5.2.3.

If a command containing a generic ROD token is processed by a copy manager that is not in the same SCSI target device as the copy manager that created the ROD token, then the actions of the processing copy manager depend on the contents of the REMOTE TOKENS field in the ROD Features third-party copy descriptor (see 7.8.17.8) as follows:

a) if the REMOTE TOKENS field is set to 0h, the processing copy manager may consider the generic ROD token to be invalid without contacting another copy manager (e.g., as a result of the processing copy manager being unable to identify or contact the copy manager that created the ROD token) and terminate the command that contains the ROD token as described in 5.16.6.5.2.3; or
b) if the REMOTE TOKENS field is not set to 0h, the processing copy manager shall:
    1) use the information in the generic ROD token to locate the copy manager that created the ROD token;
    2) communicate the generic ROD token to the creating copy manager with a request to determine the validity of the ROD token (e.g., send an EXTENDED COPY command with a verify CSCD segment descriptor (see 6.4.6.9)); and
    3) if the generic ROD token is invalid, the copy manager shall terminate the copy operation (see 5.16.4.3) as described in 5.16.6.5.2.3.

The process of determining the validity of a generic ROD token involves the following checks. A generic ROD token:

a) shall be invalid if the contents the generic ROD token do not match the equivalent information stored by the copy manager for any ROD token created by the copy manager as described in this subclause;
b) should be invalid if the generic ROD token lifetime has elapsed (see 5.16.6.7); and
c) may be invalid based on vendor specific tests.

The check for whether a generic ROD token matches a generic ROD token created by the copy manager may use:

a) exact validation that detects all differences (e.g., by comparing the generic ROD token to copies of valid generic ROD tokens saved by the copy manager when those generic ROD tokens were created) to which the additional requirements stated in 5.16.6.5.2.2 do not apply; or
b) inexact validation that does not detect all differences (e.g., by comparing a hash of the generic ROD token to hashes of valid generic ROD tokens) and implementation of the additional requirements described in 5.16.6.5.2.2.

If a copy manager determines that a generic ROD token is invalid, the copy manager shall terminate the copy operation (see 5.16.4.3) as described in 5.16.6.5.2.3.

5.16.6.5.2.2 Inexact validation of generic ROD tokens

A copy manager may use the SHA-256 secure hash function to:

a) compute a hash for each generic ROD token as part of creating that generic ROD token; and
b) compare the equivalent hash for any generic ROD token to be validated to the precomputed hash values for each of the generic ROD tokens that the copy manager considers valid.

Any generic ROD token inexact validation process shall detect generic ROD tokens that were never created by the copy manager with a certainty that is greater than or equal to that of SHA-256 hash (e.g., use of the SHA-512 secure hash meets this requirement).

A copy manager that uses a secure hash function as the only means of generic ROD token validation is exposed to collision attacks against that hash function. A more secure technique is to use the secure hash as part of a keyed cryptographic integrity check (e.g., the HMAC SHA-256 message authentication code), with a secret key that is confidential to the copy manager.

Copy managers shall not use any of the following computational techniques as the only means of generic ROD token validation:

a) any form of CRC;
b) any form of arithmetic checksum;
c) an arithmetic hash that is not a secure hash; and
d) a secure hash with a known collision resistance that is less than that of SHA-256 (e.g., most secure hash functions whose result contains less than 256 bits).

5.16.6.5.2.3 Validation errors for generic ROD tokens

If the copy manager determines that a generic ROD token is invalid, the copy manager shall:

a) not perform any of the data transfers requested by the command that specified the invalid ROD token;
b) terminate the copy operation (see 5.16.4.3); and
c) return an error using the method described in 5.16.4.3 with CHECK CONDITION status with the sense key and additional sense code associated with the highest priority error detected from among those shown in table 115.

Table 115 — Generic ROD token errors sorted by reporting importance

Error detected	Sense key	Additional sense code	Priority
Internal inconsistency or corruption problem in the generic ROD token format	ILLEGAL REQUEST	INVALID TOKEN OPERATION, TOKEN CORRUPT	Highest
Generic ROD token type that is not supported by the copy manager	ILLEGAL REQUEST	INVALID TOKEN OPERATION, UNSUPPORTED TOKEN TYPE	
Remote generic ROD token [a] and the REMOTE TOKENS field [b] is set to 0h	ILLEGAL REQUEST	INVALID TOKEN OPERATION, REMOTE TOKEN USAGE NOT SUPPORTED	
Remote generic ROD token [a] with the REMOTE TOKENS field [b] not set to 0h for which the copy manager is unable to contact the copy manager that created the ROD token	COPY ABORTED	COPY TARGET DEVICE NOT REACHABLE	
Remote generic ROD token [a] with the REMOTE TOKENS field [b] not set to 0h for which the creating copy manager returns an error	sense key returned by creating copy manager	additional sense code returned by creating copy manager	
Generic ROD token with a lifetime that has expired or an inactivity timeout that has been exceeded [c]	ILLEGAL REQUEST	INVALID TOKEN OPERATION, TOKEN EXPIRED	
Generic ROD token that does not match any valid generic ROD token created by the copy manager	ILLEGAL REQUEST	INVALID TOKEN OPERATION, TOKEN UNKNOWN	
Generic ROD token that has been revoked by a system administrator [c]	ILLEGAL REQUEST	INVALID TOKEN OPERATION, TOKEN REVOKED	
Generic ROD token that has been deleted in response to an application client request [c]	ILLEGAL REQUEST	INVALID TOKEN OPERATION, TOKEN DELETED	
Generic ROD token that has an error not described elsewhere in this table	ILLEGAL REQUEST	INVALID TOKEN OPERATION, CAUSE NOT REPORTABLE	Lowest

[a] A remote generic ROD token is a generic ROD token that was created by a copy manager that is not located in the same SCSI target device as the processing copy manager.

[b] The REMOTE TOKENS field is in the ROD Features third-party copy descriptor (see 7.8.17.8).

[c] See 5.16.6.7 for details about generic ROD tokens:
a) whose lifetime that has expired;
b) whose inactivity timeout that has been exceeded;
c) that have been revoked by a system administrator; or
d) that have been deleted in response to an application client request.

5.16.6.6 ROD token usage

If the ROD token lifetime is long enough to allow all of the following steps to be processed and no other factors cause the ROD token to become invalid, then creation and use of the ROD token in multiple third-party copy commands is accomplished as follows:

1) create one or more ROD token (see 5.16.7.5.2) in an initial third-party copy command (e.g., EXTENDED COPY(LID4) command (see 6.4) or POPULATE TOKEN command (see SBC-3));
2) retrieve the ROD tokens using the RECEIVE ROD TOKEN INFORMATION command (see 6.28); and
3) one or more subsequent third-party copy commands (e.g., EXTENDED COPY(LID4) command or WRITE USING TOKEN command (see SBC-3)) may specify the ROD token in the parameter list.

Details of the how the EXTENDED COPY commands are used in the steps shown in this subclause are described in 5.16.7.6. Details of the how the POPULATE TOKEN command and WRITE USING TOKEN command are used in the steps shown in this subclause are described in SBC-3.

If a copy manager processes a third-party copy command that contains a valid ROD token in the parameter list, then the handling of access to the ROD specified by the ROD token and to the data represented by that ROD depends on the relationship of the copy manager that processes the command to the copy manager that created the ROD token as shown in table 116.

Table 116 — Copy manager relationships for processing ROD tokens

Relationship between the copy manager that processes the command and the copy manager that created the ROD token	**REMOTE TOKENS field [a] contents returned by the processing copy manager**	**Description**
The two copy managers are the same	n/a	No errors shall be returned regarding access to the valid ROD token's contents.
The two copy managers are in the same SCSI target device	n/a	The processing copy manager shall: a) communicate with the copy manager that created the ROD token to access the data that the ROD token represents; and b) not return any errors regarding access to the valid ROD token's contents.
The two copy managers are in different SCSI target devices	0h	The processing copy manager shall terminate the copy operation (see 5.16.4.3) as described in 5.16.6.5.2.3.
	not 0h	The processing copy manager: a) shall attempt to communicate with the copy manager that created the ROD token to access the data that the ROD token represents; and b) may return errors based on the inability to locate or communicate with the copy manager that created the ROD token (see 5.16.6.5.2.3).
[a] The REMOTE TOKENS field is in the ROD Features third-party copy descriptor (see 7.8.17.8).		

A copy manager indicates that it is capable of communicating with a copy manager that is not located in the same SCSI target device as itself by setting the REMOTE TOKENS field to a value other than 0h in the ROD Features third-party copy descriptor (see 7.8.17.8).

Whenever one copy manager communicates with another copy manager to access the data that the ROD token represents, the communication is modelled as the sending of EXTENDED COPY(LID4) commands (see 6.4) and RECEIVE COPY DATA(LID4) commands (see 6.20). However, any communication between copy managers that accomplishes the equivalent of this result conforms to this standard.

EXAMPLE – The process by which one copy manager obtains all the bytes in a ROD may use an EXTENDED COPY(LID4) command that contains the block→block&application client segment descriptor (see 6.4.6.4) with a null block device logical unit (see table 148 in 6.4.6.1) as the copy destination followed by a RECEIVE COPY RESULTS(LID4) command that retrieves the held data.

In addition to converting a ROD token created by another copy manger to the data that the ROD token represents, a copy manager may indicate that it is capable of communicating with another copy manager in another SCSI target device for the purpose of creating a ROD token by setting the REMOTE TOKENS field to 6h in the ROD Features third-party copy descriptor (see 7.8.17.8).

5.16.6.7 ROD token lifetime

When a ROD token is created, the copy manager that creates the ROD token assigns it a lifetime interval based on inputs to the ROD token creation process (e.g., the REQUESTED ROD TOKEN LIFETIME field in the ROD CSCD descriptor (see 6.4.5.9)) that is used as follows:

a) until the lifetime elapses, the copy manger should not make the ROD token invalid; and
b) after the lifetime elapses, the copy manager should invalidate the ROD token.

When a ROD token is created, the copy manager that creates the ROD token assigns it an inactivity timeout that is based on inputs to the ROD token creation process (e.g., the REQUESTED ROD TOKEN INACTIVITY TIMEOUT field in the ROD CSCD descriptor (see 6.4.5.9)) that is used as follows:

a) after the completion of a third-party copy command that originates a copy operation (see table 108 in 5.16.3) that specifies the ROD token, the copy manager should not make the ROD token invalid before the inactivity timeout has expired; and
b) after the inactivity timeout has expired, the copy manager should invalidate the ROD token.

A ROD token may be invalidated for reasons other than its lifetime elapsing or inactivity timeout expiring, including the following:

a) an application client request to delete a ROD token (e.g., setting the DEL_TKN bit to one in the ROD CSCD descriptor (see 6.4.5.9) of an EXTENDED COPY command);
b) a ROD token cancellation made by the copy manager in response to operating conditions (e.g., point in time copy (see 5.16.6.2.3) processing requirements, excessive writes to the represented data, resource reclamation to create a new ROD token); and
c) a system administrator request to revoke a ROD token for management reasons.

If any of the conditions described in this subclause cause a ROD token to become invalid, then the copy manager shall maintain a record of them for reporting purposes (see 5.16.6.5.2.3) for at least the lifetime of the ROD token established at the time the ROD token was created, and should maintain the record for twice the lifetime of the ROD token.

5.16.7 The EXTENDED COPY command

5.16.7.1 EXTENDED COPY parameter list

The EXTENDED COPY(LID4) command (see 6.4) and EXTENDED COPY(LID1) command (see 6.5) use a parameter list with several elements to define a flexible interface to third-party copy functions (see figure 18).

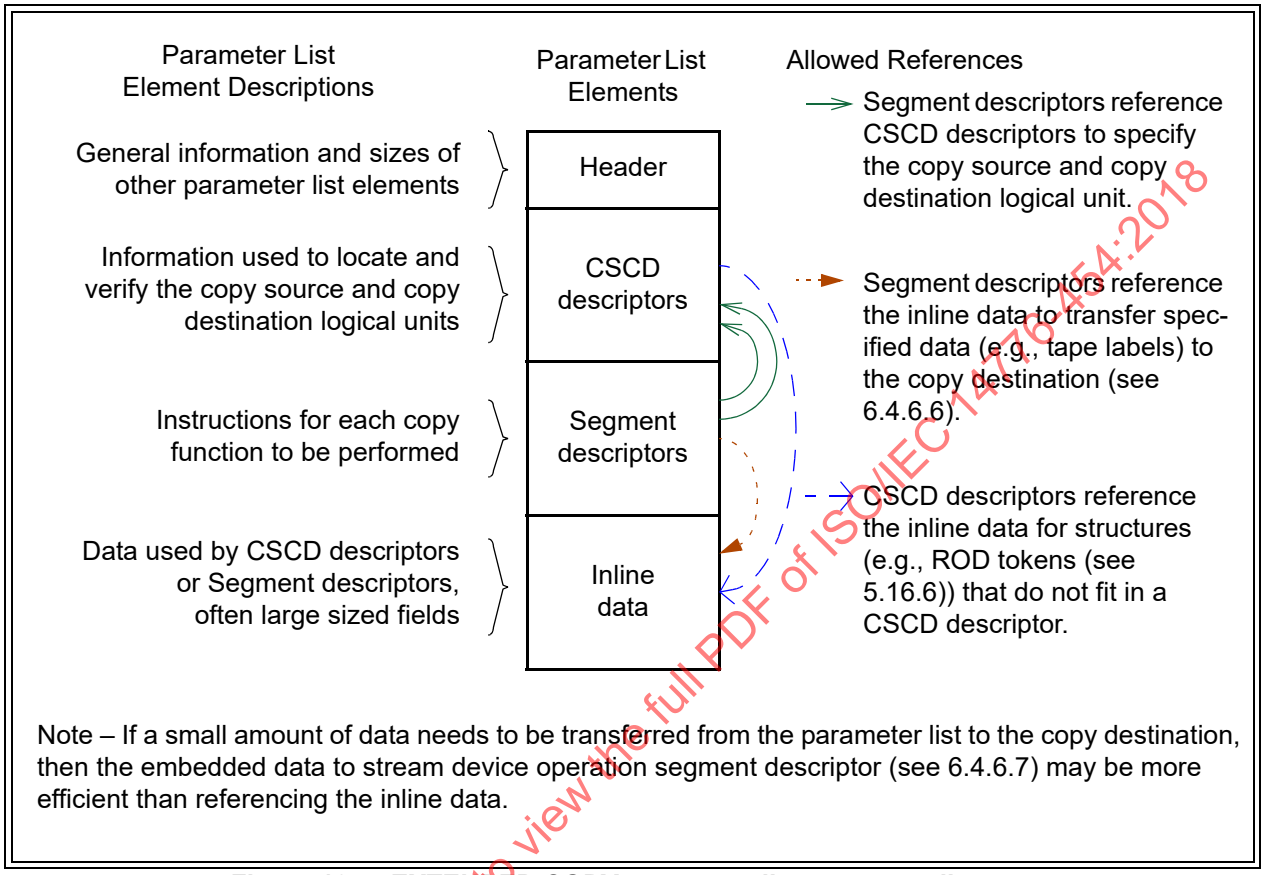


Figure 18 — EXTENDED COPY parameter list structure diagram

EXTENDED COPY parameter list elements that are not used are not included in the parameter list (e.g., the length of the inline data element is zero if no inline data is referenced).

5.16.7.2 EXTENDED COPY command processing

To process an EXTENDED COPY command, the copy manager:

1) shall determine the global processing conditions for the command (e.g., the value of the IMMED bit (see 6.4.2));
2) shall determine the first parameter list byte for each parameter list element shown in figure 18 (see 5.16.7.1) based on fields in the header;
3) may validate the contents of one or more CSCD descriptors (see 5.16.7.1) and terminate the EXTENDED COPY command if errors are detected;
4) may validate the contents of one or more segment descriptors and terminate the EXTENDED COPY command if errors are detected;
5) depending on the value of the IMMED bit in an EXTENDED COPY(LID4) command:
   A) shall return command completion if the IMMED bit is set to one; or
   B) shall not return status for the command until the copy operation (see 5.16.4.3) is completed or terminated if the IMMED bit is set to zero;

   and

6) shall process the segment descriptors in the order in which they appear in the parameter list as described in this subclause.

The copy functions performed by an EXTENDED COPY command are specified by the segment descriptors. All other elements of the EXTENDED COPY parameter list (see 5.16.7.1) are present to support the segment descriptors in their specification of the copy functions to be performed.

If a CSCD descriptor is referenced by a segment descriptor (see 5.16.7.1), the information in that CSCD descriptor should be validated at the time the referencing segment descriptor is processed. The SCSI devices participating in a SCSI domain may change between the time that processing of an EXTENDED COPY command begins and the time that processing of a specific segment descriptor begins.

The copy manager shall perform the copy functions specified by the segment descriptors (see 5.16.7.1) using a model in which commands are sent from the copy manager application client to local device servers and/or remote device servers. The specific commands sent by the copy manager to the copy sources and copy destinations while processing the segment descriptors are vendor specific. Upon successful completion of the copy operation (see 5.16.4.3), all copy sources and copy destinations that are stream devices (see SSC-4) shall be positioned at deterministic locations such that they may be repositioned to the same location by the application client with appropriate commands.

During the processing of a segment descriptor, the copy manager may be required to:

a) manage the movement of data from a copy source to a copy destination as follows:
   A) read source data by issuing data input commands to the copy source;
   B) process data in a way that may designate data as destination data (i.e., data intended for transfer to the copy destination); and
   C) write some or all of the destination data to the copy destination;
b) manage the movement of a specified portion of the embedded data or inline data to a copy destination as follows:
   A) designate the specified portion of the embedded data or inline data as destination data intended for transfer to the copy destination; and
   B) write some or all of the destination data to the copy destination;
   or
c) perform functions that are specific to the peripheral device type (e.g., writing filemarks) to a copy destination.

As part of processing a segment descriptor, the copy manager may verify the information in a CSCD descriptor's device specific fields. However, while verifying the information, the copy manager shall not send any commands that change the position of read/write media on the CSCD without repositioning the media to its original position. Any errors encountered while verifying the information shall be processed as described in 5.16.7.4.

The number of blocks to read and write, the number of bytes to process, and the nature of processing are determined by:

a) the segment descriptor type code;
b) the parameters of the segment descriptor; and
c) 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, the processing of a segment descriptor is performed as follows:

a) just enough whole block reads 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; and

c) as many whole block writes 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 copy source 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.

Segment descriptor processing requirements that are specific to one or more segment descriptor type codes are described in table 117 and the referenced subclauses.

Table 117 — Segment descriptor type specific copy manager processing requirements (part 1 of 2)

Segment descriptor type code	Reference	Description
00h (i.e., block→stream) or 0Bh (i.e., block→stream&application client)	6.4.6.2	The number of bytes processed is determined by the BLOCK DEVICE NUMBER OF BLOCKS field for the copy source (see applicable type code definition subclauses for details). [a]
02h (i.e., block→block) with DC=0 or 0Dh (i.e., block→block&application client) with DC=0	6.4.6.4	
02h (i.e., block→block) with DC=1 or 0Dh (i.e., block→block&application client) with DC=1	6.4.6.4	The number of blocks or the byte range specified shall be output to the copy destination. 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 copy source) so that the destination data (which includes any residual destination data from the previous segment) is sufficient. [a]
01h (i.e., stream→block) or 0Ch (i.e., stream→block&application client)	6.4.6.3	
09h (i.e., stream→block<o>)	6.4.6.11	
03h (i.e., stream→stream) or 0Eh (i.e., stream→stream&application client)	6.4.6.5	The number of bytes specified in the segment descriptor shall be processed. [a]
04h (i.e., inline→stream)	6.4.6.6	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 (i.e., embedded→stream)	6.4.6.7	
06h (i.e., stream→discard)	6.4.6.8	The specified number of bytes shall be removed from the source data and discarded.
07h (i.e., verify CSCD operation)	6.4.6.9	No data shall be processed and no reads or writes shall be performed on CSCDs. Residual source or destination data, if any, shall be retained or discarded as if the CAT bit were set to one.
10h (i.e., filemark→tape)	6.4.6.13	
11h (i.e., space→tape)	6.4.6.14	
12h (i.e., locate→tape)	6.4.6.15	
14h (i.e., register persistent reservation key)	6.4.6.17	
15h (i.e., Third party persistent reservations source I_T nexus)	6.4.6.18	

[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 (see 5.16.4.5).

Table 117 — Segment descriptor type specific copy manager processing requirements (part 2 of 2)

Segment descriptor type code	Reference	Description
08h (i.e., block<o>→stream)	6.4.6.10	The required blocks shall be read from the copy source, the designated byte range shall be extracted as source data, and the designated number of bytes shall be processed starting with residual source data, if any.
0Ah (i.e., block<o>→block<o>)	6.4.6.12	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 (i.e., stream→discard&application client)	6.4.6.8	The specified number of bytes shall be removed from the source data and held for retrieval by the application client.
13h (i.e., <i>tape→<i>tape)	6.4.6.16	The data movement, if any, 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 set to one.
16h (i.e., <i>block→<i>block)	6.4.6.19	
BEh (i.e., ROD←block ranges(n))	6.4.6.20	
BFh (i.e., ROD←block range(1))	6.4.6.21	

[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 (see 5.16.4.5).

The results are outside the scope of this standard if:

a) the copy source LBA range overlaps the copy destination LBA range;
b) both LBA ranges are in the same logical unit; and
c) both LBA ranges are specified in the same segment descriptor.

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 118.

Table 118 — PAD and CAT bit definitions

PAD bit in		CAT bit	Copy manager action
Source CSCD descriptor	Destination CSCD 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 CSCD ID or destination CSCD ID index in the subsequent segment descriptor does not match the corresponding CSCD ID index with which residual data was originally associated. If the CAT bit is set to one in the last segment descriptor in the parameter data (see 5.16.7.1), then any residual data shall be discarded and 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 processed as if the CAT bit is set 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 copy manager shall terminate the copy operation (see 5.16.4.3) with CHECK CONDITION status, with the sense key set to COPY ABORTED, and the additional sense code set to UNEXPECTED INEXACT SEGMENT.

[a] If the CAT bit is set to zero in the segment descriptor and the PAD bit is set to one in the destination CSCD descriptor, then the copy manager shall terminate the copy operation (see 5.16.4.3) with CHECK CONDITION status, with the sense key set to COPY ABORTED, and the additional sense code 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 (i.e., stream→block <o>) or 0Ah (i.e., 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 (i.e., block→block) with DC set to one, 0Dh (i.e., block →block& application client) with DC set to one, 01h (i.e., stream→block) or 0Ch (i.e., stream→block& application client).

Table 119 defines the PAD bit handling for segment descriptors that have either no copy source or no copy destination.

Table 119 — PAD bit processing if there is no copy source or copy destination

Segment descriptor type code	Reference	Description
04h (i.e., inline→stream)	6.4.6.6	Processing shall be as if the PAD is set to zero for the source CSCD descriptor.
05h (i.e., embedded→stream)	6.4.6.7	
06h (i.e., stream→discard) 0Fh (i.e., stream→discard&application client)	6.4.6.8	Processing shall be as if the PAD is set to zero for the destination CSCD descriptor.

5.16.7.3 EXTENDED COPY command errors detected before segment descriptor processing starts

Errors may occur during processing of an EXTENDED COPY command before the first segment descriptor is processed or before status is returned due to an IMMED bit set to one. These errors include 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;
b) set the sense key to a value that describes the exception condition (i.e., not COPY ABORTED); and
c) not return a value in the INFORMATION field (see 4.5.4).

5.16.7.4 EXTENDED COPY command errors detected during processing of segment descriptors

Errors may occur after the copy manager has begun processing segment descriptors or after status has been returned due to an IMMED bit set to one. These errors include invalid parameters in segment descriptors, invalid segment descriptors, unavailable CSCDs referenced by CSCD descriptors, inability of the copy manager to continue operating, and errors reported by a CSCD. If a copy manager command to a CSCD returns CHECK CONDITION status, the copy manager shall recover the sense data associated with the exception condition and clear any ACA condition associated with the CHECK CONDITION status.

The copy manager shall terminate the copy operation (see 5.16.4.3) with CHECK CONDITION status, with the sense key set to COPY ABORTED, and the additional sense code set to COPY TARGET DEVICE NOT REACHABLE if it is not possible to complete processing of a segment descriptor as a result of:

a) the copy manager being unable to establish communications with a CSCD;
b) the CSCD not responding to an INQUIRY command; or
c) the data returned to the copy manager in response to an INQUIRY command indicating an unsupported logical unit.

If it is not possible to complete processing of a segment descriptor as a result of the data returned in response to an INQUIRY command indicates a device type that does not match the type in the CSCD descriptor, then the copy manager shall terminate the copy operation (see 5.16.4.3) with CHECK CONDITION status, with the sense key set to COPY ABORTED, and the additional sense code set to INCORRECT COPY TARGET DEVICE TYPE.

If the copy manager has sends a command other than INQUIRY to a CSCD while processing a copy operation (see 5.16.4.3) and the CSCD either fails to respond with status or responds with status other than BUSY, TASK SET FULL, ACA ACTIVE, or RESERVATION CONFLICT, then the copy manager shall terminate the copy operation with CHECK CONDITION status, with the sense key set to COPY ABORTED, and the additional sense code set to THIRD PARTY DEVICE FAILURE.

If a CSCD completes a command from the copy manager with a status of BUSY, TASK SET FULL, ACA ACTIVE, or RESERVATION CONFLICT, then the copy manager shall either retry the command or terminate the copy operation (see 5.16.4.3) with CHECK CONDITION status, with the sense key set to COPY ABORTED, and the additional sense code set to THIRD PARTY DEVICE FAILURE.

NOTE 23 - The copy manager is assumed to employ a vendor specific retry policy that minimizes time consuming repetition of retries.

NOTE 24 - RESERVATION CONFLICT status is listed only to give the copy manager leeway in multi-port cases. The copy manager may have multiple initiator ports that are capable of reaching a CSCD, and a persistent reservation may restrict access to a single I_T nexus. The copy manager may need to try access from multiple initiator ports to find the correct I_T nexus.

If a CSCD responds to an input or output request with a GOOD status but less data than expected is transferred, then the copy manager shall terminate the copy operation (see 5.16.4.3) with CHECK CONDITION status, with the sense key set to COPY ABORTED, and the additional sense code set to COPY TARGET DEVICE DATA UNDERRUN. If an overrun is detected, then the copy manager shall terminate the copy operation with CHECK CONDITION status, with the sense key set to COPY ABORTED, and the additional sense code set to COPY TARGET DEVICE DATA OVERRUN.

After an exception condition is detected during segment descriptor processing:

a) the copy manager shall terminate the copy operation (see 5.16.4.3) with CHECK CONDITION status, with the sense key set to COPY ABORTED;
b) the copy manager shall return the segment number of the segment that was being processed at the time of the exception in the third and fourth bytes of the COMMAND-SPECIFIC INFORMATION field (see 4.5.5). The segment number is based on the relative position of the segment descriptor in the parameter list (see 5.16.7.1) (i.e., the first segment descriptor in the parameter list is assigned descriptor number zero, the second is assigned one, etc.);
c) if any data has been written to the copy destination for the segment being processed at the time the error occurred, then the copy manager shall return the residual count for the segment in the INFORMATION field (see 4.5.4) using units of:
   A) bytes if the destination CSCD descriptor contains:
      a) 03h (i.e., processor device) in the PERIPHERAL DEVICE TYPE field; or
      b) 01h (i.e., sequential-access device) in the PERIPHERAL DEVICE TYPE field and the FIXED bit is set to zero in the device type specific parameters;

   or
   B) copy destination blocks for all other cases.

   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. While computing the residual count, the copy manager shall include only the results of commands successfully completed by a copy destination (i.e., commands completed by a copy destination with GOOD status or with CHECK CONDITION status and the EOM bit set to one in the sense data). If the copy manager has used out of order transfers, then the residual count shall be based only on the contiguous transfers completed without error 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 copy destination for the segment being processed at the time the error occurred, then the copy manager shall not return a value in the INFORMATION field (see 4.5.4). Segment descriptors that do not specify a transfer count shall not have a valid residual count;
d) if the exception condition is reported by the copy source, then:
   A) if fixed format sense data (see 4.5.3) is being returned, then the first byte of the COMMAND-SPECIFIC INFORMATION field shall be set to 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 copy source. The status byte and sense data shall not be modified by the copy

manager. A zero value indicates that no status byte and sense data is being returned for the copy source; or

B) if descriptor format sense data (see 4.5.2) is being returned, then the status byte and sense data delivered to the copy manager by the copy source shall be returned in a forwarded sense data descriptor (see 4.5.2.7) with the SOURCE field set to 1h. The status byte shall not be modified by the copy manager. The sense data may be truncated by the copy manager but shall not be modified in any other way. If the sense data is truncated, then the FSDT bit in the forwarded additional sense data descriptor shall be set to one;

e) if the exception condition is reported by the copy destination, then:

A) if fixed format sense data (see 4.5.3) is being returned, then the second byte of the COMMAND-SPECIFIC INFORMATION field shall be set to 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 copy destination. The status byte and sense data shall not be modified by the copy manager. A zero value indicates that no status byte and sense data is being returned for the copy destination; or

B) if descriptor format sense data (see 4.5.2) is being returned, then the status byte and sense data delivered to the copy manager by the copy destination shall be returned in a forwarded sense data descriptor (see 4.5.2.7) with the SOURCE field set to 2h. The status byte shall not be modified by the copy manager. The sense data may be truncated by the copy manager but shall not be modified in any other way. If the sense data is truncated, then the FSDT bit in the forwarded additional sense data descriptor shall be set to one;

f) if segment processing is terminated as a result of a CSCD is unreachable or as the result of a failure in a command sent to a CSCD, then the sense key specific information shall be set as described in 4.5.2.4.5, with the FIELD POINTER field indicating the first byte of the CSCD descriptor that specifies the CSCD;

g) if, during the processing of a segment descriptor, the copy manager detects an error in the segment descriptor, then the sense key specific information shall be set as described in 4.5.2.4.5, 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 is set to an offset from the start of the parameter data. The SD bit shall be set to one to indicate the FIELD POINTER field is set to an offset from the start of the segment descriptor; and

h) if the LIST ID USAGE field is set to 00b or 10b in the parameter data (see 5.16.7.1), the copy manager shall preserve information for:

A) the RECEIVE COPY FAILURE DETAILS(LID1) command (see 6.23), if supported. The preserved information, if any, shall be discarded as described in 6.23; and

B) the:

a) RECEIVE COPY STATUS(LID4) command (see 6.24), if supported;

b) RECEIVE COPY DATA(LID4) command (see 6.20), if supported; and

c) RECEIVE ROD TOKEN INFORMATION command (see 6.28), if supported.

The preserved information, if any, shall be discarded as described in 6.24.

5.16.7.5 EXTENDED COPY considerations for RODs and ROD tokens

5.16.7.5.1 EXTENDED COPY command CSCD ROD identifiers

All ROD CSCD descriptors (see 6.4.5.9) create an identifier for the ROD that they describe. The identifier is the CSCD descriptor ID (see table 148 in 6.4.6.1) for the ROD CSCD descriptor.

A ROD identifier identifies a ROD that is one of the following:

a) specified by segment descriptors that populate the ROD (see 5.16.7.5.2) in the EXTENDED COPY parameter list (see 5.16.7.1); or

b) the conversion of a ROD token into an identifier using the following entries in the EXTENDED COPY parameter list:

A) copying the ROD token to the inline data (see 5.16.7.1) in the EXTENDED COPY parameter list; and

B) referencing the copied ROD token in a ROD CSCD descriptor in which the ROD TYPE field is set to zero.

If the R_TOKEN bit is set to one in a ROD CSCD descriptor (see 6.4.5.9) in which the ROD TYPE field is not set to zero, then the identified ROD is used to create a ROD token that is made available for transfer to the application client using the RECEIVE ROD TOKEN INFORMATION command (see 6.28) and used as described in 5.16.6.6.

After the copy manager finishes processing the copy operation (see 5.16.4.3) originated by an EXTENDED COPY command, all the ROD identifiers created by that command shall become invalid.

If a ROD identifier becomes invalid (see 5.16.6.2.2, 5.16.6.2.3, and 5.16.6.7) before processing has been completed for the copy operation (see 5.16.4.3) originated by the EXTENDED COPY command in which the ROD identifier is defined, then any attempt to use the ROD as a copy source or copy destination shall cause the copy manager shall terminate the copy operation as if an unreachable CSCD device had been encountered (see 5.16.7.4).

5.16.7.5.2 Populating an EXTENDED COPY command ROD

If the ROD TYPE field (see 6.4.5.9) is not set to zero in the ROD CSCD descriptor, the following segment descriptors are used to populate a ROD as described 5.16.6.3:

a) populate a ROD from one or more block device ranges (see 6.4.6.20); and
b) populate a ROD from one block device range (see 6.4.6.21).

If one of the segment descriptors listed in this subclause specifies the CSCD descriptor ID (see table 148 in 6.4.6.1) of the ROD CSCD descriptor as the copy destination and no errors are detected, then the specified ROD is populated with the information associated with the ROD type (see 5.16.6.2) based on the contents of the segment descriptor.

If the CSCD descriptor ID of a ROD CSCD descriptor in which the ROD TYPE field (see 6.4.5.9) is set to zero is specified as the copy destination in one of the segment descriptors listed in this subclause, then the copy manager shall terminate the copy operation (see 5.16.4.3) with CHECK CONDITION status, with the sense key set to ILLEGAL REQUEST, and the additional sense code set to INVALID FIELD IN PARAMETER LIST. The sense key specific data, if any, shall identify the CSCD descriptor ID in the segment descriptor as the field that in error.

Bytes shall be added to the ROD in the order that the segment descriptors are present in the EXTENDED COPY parameter data (see 5.16.7.1). If a single segment descriptor specifies multiple ROD byte sources, the bytes shall be added to the ROD in the order that the bytes are present in that segment descriptor.

The process of populating the ROD ends when:

a) the CSCD descriptor ID for the ROD CSCD descriptor is specified as a copy source or copy destination in a segment descriptor that is not one of those listed in this subclause; or
b) processing of the copy operation (see 5.16.4.3) originated by the EXTENDED COPY command ends.

If one of the segment descriptors listed in this subclause specifies the CSCD descriptor ID of the ROD CSCD descriptor as the copy destination after the process of populating the ROD has ended, then the copy manager shall terminate the copy operation (see 5.16.4.3) with CHECK CONDITION status, with the sense key set to COPY ABORTED, and the additional sense code set to INVALID FIELD IN PARAMETER LIST.

If the R_TOKEN bit is set to one in the ROD CSCD descriptor (see 6.4.5.9), then the copy manager shall create a ROD token (see 5.16.6) for the populated ROD and prepare the ROD token for retrieval using the RECEIVE ROD TOKEN INFORMATION command (see 6.28).

The copy manager shall terminate the copy operation (see 5.16.4.3) with CHECK CONDITION status, with the sense key set to COPY ABORTED, and the additional sense code set to INVALID FIELD IN PARAMETER LIST if:

a) the value in PERIPHERAL DEVICE TYPE field of a copy source specified in a segment descriptor does not match the value in the PERIPHERAL DEVICE TYPE field in ROD CSCD descriptor (see 6.4.5.9); or
b) characteristics (e.g., logical block length, protection information characteristics (see SBC-3)) associated with the data specified by a segment descriptor are incompatible with the characteristics associated with the data specified by a previous segment descriptor, if any.

5.16.7.6 EXTENDED COPY command use of RODs when the peripheral device type is 00h (i.e., block device)

The copy manager shall cause a ROD that was created by a copy manager associated with a logical unit whose peripheral device type is 00h to function as a block device that:

a) exists only while the copy operation (see 5.16.4.3) originated by the EXTENDED COPY command is being processed;
b) may be used by any segment descriptor (see 6.4.6.1) that operates on a block device as a:
   A) copy source; or
   B) copy destination, if allowed by the ROD type (see 5.16.6.2);
c) has the following block device characteristics:
   A) a contiguous range of logical blocks;
   B) LBAs numbered from zero to the number of logical blocks represented by the ROD (see 5.16.7.5.2) minus one;
   C) data that matches the data that is in the ROD; and
   D) characteristics that match those of the logical blocks that are represented by the ROD, including at least the following:
      a) a DISK BLOCK LENGTH field in the device type specific CSCD descriptor parameters (see 6.4.5.3) that is set to the logical block length of each logical block represented by the ROD; and
      b) protection information that matches the protection information of the logical blocks that are represented by the ROD;

   and
d) may have logical block provisioning information equivalent to the logical block provisioning information of the logical blocks that are represented by the ROD (see 5.16.6).

5.16.7.7 EXTENDED COPY command interactions with aliases

An application client may use alias values to reference SCSI target devices or SCSI target ports in the EXTENDED COPY command parameter list (see 5.16.7.1). The alias list provides a mechanism for eight-byte third party identifier fields to reference a third party device or port whose name or addressing information is longer than eight bytes.

EXAMPLE – An application may use the CHANGE ALIASES command (see 6.2) to establish an association between an alias value and a SCSI target device or target port designation. Then, the application client may send an EXTENDED COPY command containing in the parameter data an alias CSCD descriptor (see 6.4.5.7) that includes this alias value.

After the completion of the EXTENDED COPY command, an application should clear all associated alias values from the device server's alias list by sending a CHANGE ALIASES command that requests association of each alias value to a NULL DESIGNATION (see 6.2.4.2) alias format.

6 Commands for all device types

6.1 Summary of commands for all device types

The operation codes for commands that apply to all device types are listed in table 120.

Table 120 — Commands for all device types (part 1 of 2)

Command	Operation code	Support	Reference
ACCESS CONTROL IN	86h	O	8.3.2
ACCESS CONTROL OUT	87h	O	8.3.3
CHANGE ALIASES	A4h/0Bh [a]	O	6.2
COPY OPERATION ABORT	83h/1Ch [a]	O	6.3
EXTENDED COPY(LID4)	83h/01h [a]	O	6.4
EXTENDED COPY(LID1)	83h/00h [a]	O	6.5
INQUIRY	12h	M	6.6
LOG SELECT	4Ch	O	6.7
LOG SENSE	4Dh	O	6.8
MANAGEMENT PROTOCOL IN	A3h/10h [a]	O	6.9
MANAGEMENT PROTOCOL OUT	A4h/10h [a]	O	6.10
MODE SELECT(6)	15h	C	6.11
MODE SELECT(10)	55h	C	6.12
MODE SENSE(6)	1Ah	C	6.13
MODE SENSE(10)	5Ah	C	6.14
PERSISTENT RESERVE IN	5Eh	C	6.15
PERSISTENT RESERVE OUT	5Fh	C	6.16
READ ATTRIBUTE	8Ch	O	6.17
READ BUFFER	3Ch	O	6.18
READ MEDIA SERIAL NUMBER	ABh/01h [a]	C	6.19
RECEIVE COPY DATA(LID4)	84h/06h [a]	O	6.20
RECEIVE COPY DATA(LID1)	84h/01h [a]	O	6.21
RECEIVE COPY OPERATING PARAMETERS	84h/03h [a]	O	6.22
RECEIVE COPY FAILURE DETAILS(LID1)	84h/04h [a]	O	6.23
RECEIVE COPY STATUS(LID4)	84h/05h [a]	O	6.24
RECEIVE COPY STATUS(LID1)	84h/00h [a]	O	6.25
RECEIVE CREDENTIAL	7Fh/1800h [a]	O	6.26
RECEIVE DIAGNOSTIC RESULTS	1Ch	O	6.27
RECEIVE ROD TOKEN INFORMATION	84h/07h [a]	O	6.28
REMOVE I_T NEXUS	A4h/0Ch [a]	O	6.29
REPORT ALIASES	A3h/0Bh [a]	O	6.30
REPORT ALL ROD TOKENS	84h/08h [a]	O	6.31
REPORT IDENTIFYING INFORMATION	A3h/05h [a]	O	6.32

Type Key: C = Command support is defined in the applicable command standard.
M = Command support is mandatory.
O = Command support is optional.

A numeric ordered listing of operation codes is provided in F.3.

[a] 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.

[b] The following operation codes are obsolete: 16h, 17h, 18h, 39h, 3Ah, 40h, 56h, and 57h.

Table 120 — Commands for all device types (part 2 of 2)

Command	Operation code	Support	Reference
REPORT LUNS	A0h	M	6.33
REPORT PRIORITY	A3h/0Eh [a]	O	6.34
REPORT SUPPORTED OPERATION CODES	A3h/0Ch [a]	O	6.35
REPORT SUPPORTED TASK MANAGEMENT FUNCTIONS	A3h/0Dh [a]	O	6.36
REPORT TARGET PORT GROUPS	A3h/0Ah [a]	O	6.37
REPORT TIMESTAMP	A3h/0Fh [a]	O	6.38
REQUEST SENSE	03h	C	6.39
SECURITY PROTOCOL IN	A2h	O	6.40
SECURITY PROTOCOL OUT	B5h	O	6.41
SEND DIAGNOSTIC	1Dh	C	6.42
SET IDENTIFYING INFORMATION	A4h/06h [a]	O	6.43
SET PRIORITY	A4h/0Eh [a]	O	6.44
SET TARGET PORT GROUPS	A4h/0Ah [a]	O	6.45
SET TIMESTAMP	A4h/0Fh [a]	O	6.46
TEST UNIT READY	00h	M	6.47
WRITE ATTRIBUTE	8Dh	O	6.48
WRITE BUFFER	3Bh	C	6.49
Obsolete [b]			

Type Key: C = Command support is defined in the applicable command standard.
M = Command support is mandatory.
O = Command support is optional.

A numeric ordered listing of operation codes is provided in F.3.

[a] 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.

[b] The following operation codes are obsolete: 16h, 17h, 18h, 39h, 3Ah, 40h, 56h, and 57h.

6.2 CHANGE ALIASES command

6.2.1 CHANGE ALIASES command introduction

The CHANGE ALIASES command (see table 121) requests the device server to make changes to a list of associations between eight-byte alias values and SCSI target device or SCSI target port designations that the device server maintains. This command uses the MAINTENANCE OUT CDB format (see 4.2.2.3.4). A designation contains a name and optional identifier information that specifies a SCSI target device or SCSI target port (see 6.2.2). The alias list may be queried by the application client via the REPORT ALIASES command (see 6.30). If the REPORT ALIASES command is supported, the CHANGE ALIASES command shall also be supported.

Table 121 — CHANGE ALIASES command

Bit / Byte	7	6	5	4	3	2	1	0
0	OPERATION CODE (A4h)							
1	Reserved			SERVICE ACTION (0Bh)				
2	Reserved							
•••								
5								
6	(MSB)			PARAMETER LIST LENGTH				
•••								
9								(LSB)
10	Reserved							
11	CONTROL							

The OPERATION CODE field is defined in 4.2.5.1 and shall be set as shown in table 121 for the CHANGE ALIASES command.

The SERVICE ACTION field is defined in 4.2.5.2 and shall be set as shown in table 121 for the CHANGE ALIASES command.

The PARAMETER LIST LENGTH field specifies the length in bytes of the parameter data that shall be transferred from the application client to the device server. A parameter list length value of zero specifies that no data shall be transferred and no changes shall be made in the alias list.

The CONTROL byte is defined in SAM-5.

If the parameter list length results in the truncation of the header or any alias entry, the device server shall make no changes to the alias list and shall terminate the command with CHECK CONDITION status, with the sense key set to ILLEGAL REQUEST, and the additional sense code set to PARAMETER LIST LENGTH ERROR.

On successful completion of a CHANGE ALIASES command, the device server shall maintain an association of each assigned eight-byte alias value to the SCSI target device or SCSI target port designation. These associations shall be cleared by a logical unit reset or I_T nexus loss. The device server shall maintain a separate alias list for each I_T nexus.

A CHANGE ALIASES command may add, change, or remove entries from the alias list. Alias list entries not referenced in the CHANGE ALIASES parameter data shall not be changed.

If the device server has insufficient resources to make all requested changes to the alias list, then the device server shall make no changes to the alias list and shall terminate the command with CHECK CONDITION status, with the sense key set to ILLEGAL REQUEST, and the additional sense code set to INSUFFICIENT RESOURCES.

The parameter list for a CHANGE ALIASES command (see table 122) contains zero or more alias entries. If the device server processes a CHANGE ALIASES command that contains at least one alias entry while there exists any other enabled command that references an alias entry in the alias list, then the device server shall terminate the CHANGE ALIASES command with CHECK CONDITION status, with the sense key set to ILLEGAL REQUEST, and the additional sense code set to OPERATION IN PROGRESS.

Table 122 — CHANGE ALIASES parameter list

Bit / Byte	7	6	5	4	3	2	1	0
0	(MSB)							
•••				PARAMETER DATA LENGTH (n-3)				
3								(LSB)
4								
•••				Reserved				
7								
	Alias entry list							
8								
•••				Alias entry (see 6.2.2) [first]				
				⋮				
•••				Alias entry (see 6.2.2) [last]				
n								

The PARAMETER DATA LENGTH field should contain the number of bytes of alias entries, and shall be ignored by the device server.

The format of an alias entry is described in 6.2.2.

6.2.2 Alias entry format

One alias entry (see table 123) describes one alias reported via the REPORT ALIASES command (see 6.30) or to be changed via the CHANGE ALIASES command.

Table 123 — Alias entry format

Bit / Byte	7	6	5	4	3	2	1	0
0	(MSB)							
•••				ALIAS VALUE				
7								(LSB)
8				PROTOCOL IDENTIFIER				
9				Reserved				
10								
11				FORMAT CODE				
12				Reserved				
13								
14	(MSB)							
15				DESIGNATION LENGTH (n-15)				(LSB)
16								
•••				DESIGNATION				
n								

The ALIAS VALUE field is set to the numeric alias value that the device server shall associate with the SCSI target device or target port specified by the values in the PROTOCOL IDENTIFIER, FORMAT CODE and DESIGNATION fields.

The PROTOCOL IDENTIFIER field (see table 124) specifies whether the alias entry designation is independent of SCSI transport protocol and the SCSI transport protocol, if any, to which the alias entry applies.

Table 124 — Alias entry PROTOCOL IDENTIFIER field

Code	Description	Reference
00h to 0Fh	Protocol specific designation	7.6.2
10h to 7Fh	Reserved	
80h	Protocol independent designation	6.2.4
81h to FFh	Reserved	

The FORMAT CODE field contents combined with the PROTOCOL IDENTIFIER field contents defines the format of the DESIGNATION field. The subclauses that describe each PROTOCOL IDENTIFIER field usage (see table 124) define the applicable FORMAT CODE field values.

The DESIGNATION LENGTH field specifies the number of bytes of the DESIGNATION field. The DESIGNATION LENGTH value shall be a multiple of four.

The zero-padded (see 4.3.2) DESIGNATION field should designate a unique SCSI target device or target port using the following:

a) a SCSI device name or a target port name;
b) zero or more target port identifiers; and

c) zero or more SCSI transport protocol specific identifiers.

6.2.3 Alias designation validation

The device server shall not validate any designation at the time of processing either the REPORT ALIASES or CHANGE ALIASES command. Such validation shall occur only when the device server uses the alias list to resolve an alias to a designation in the context of third-party commands (e.g., EXTENDED COPY) or any other command that requires reference to the alias list.

If a designation identifies a unique SCSI target device or target port that is within a SCSI domain accessible to the device server, then the designation is considered valid.

Based on the SCSI transport protocol specific requirements for a given designation format, a designation that does not identify a unique SCSI target device or target port within the SCSI domains accessible to the device server is considered invalid.

EXAMPLE 1 – A designation may be considered invalid if the device server has no ports on the SCSI domain of the designated SCSI target device or target port.

A designation having both name and identifier information may be inconsistent if the device server is not able to access the named SCSI target device or target port through one or more of the names or identifiers in the designation. In such cases, the designation shall be processed as valid or invalid according to the SCSI transport protocol specific requirements.

EXAMPLE 2 – In FCP-4, both an N_Port and World Wide Name for a SCSI port may be given in a designation. The designation definition may require that the N_Port be that of the named port. In that case, the designation is invalid. Alternatively, the designation definition may view the N_Port as a hint for the named FC Port accessible to the device server through a different D_ID. In that case, the designation is valid and designates the named FC Port.

NOTE 25 - If only name information is provided in a designation, it is assumed that the device server has access to a mechanism for resolving names to identifiers. Access to such a service is SCSI transport protocol specific and vendor specific.

6.2.4 Alias entry protocol independent designations

6.2.4.1 Alias entry protocol independent designations overview

The protocol independent alias entry designations have the PROTOCOL IDENTIFIER field set to 80h and the FORMAT CODE field set to one of the codes shown in table 125.

Table 125 — Protocol independent alias entry FORMAT CODE field

Code	Name	Designation Length (bytes)	Designation Contents	Reference
00h	NULL DESIGNATION	0	none	6.2.4.2
01h to FFh	Reserved			

6.2.4.2 NULL DESIGNATION alias format

In response to an alias entry with the NULL DESIGNATION format, the device server shall remove the specified alias value from the alias list. Application clients should use the NULL DESIGNATION format in a CHANGE ALIASES command to remove inactive alias entries from the alias list if that alias entry is no longer needed. The NULL DESIGNATION format shall not appear in REPORT ALIASES parameter data.

6.3 COPY OPERATION ABORT command

The COPY OPERATION ABORT command (see table 126) is a third-party copy command (see 5.16.3) that requests the copy manager to abort the specified copy operation (see 5.16.4.3) as described in 5.16.4.7. Whether the copy operation is being processed in the foreground or background, the effect of the COPY OPERATION ABORT command is equivalent to an ABORT TASK task management function (see SAM-5 and 5.16.4.6).

Table 126 — COPY OPERATION ABORT command

Bit / Byte	7	6	5	4	3	2	1	0
0	OPERATION CODE (83h)							
1	Reserved			SERVICE ACTION (1Ch)				
2	(MSB)							
•••	LIST IDENTIFIER							
5								(LSB)
6								
•••	Reserved							
14								
15	CONTROL							

The OPERATION CODE field is defined in 4.2.5.1 and shall be set as shown in table 126 for the COPY OPERATION ABORT command.

The SERVICE ACTION field is defined in 4.2.5.2 and shall be set as shown in table 126 for the COPY OPERATION ABORT command.

The LIST IDENTIFIER field is defined in 5.16.4.2 and 6.4.3.2, and specifies the copy operation to be aborted. If the copy manager is not processing a copy operation with the specified list identifier, the copy manager shall terminate the COPY OPERATION ABORT command with CHECK CONDITION status, with the sense key set to ILLEGAL REQUEST, and the additional sense code set to INVALID FIELD IN CDB.

The CONTROL byte is defined in SAM-5.

6.4 EXTENDED COPY(LID4) command

6.4.1 EXTENDED COPY(LID4) command introduction

The EXTENDED COPY(LID4) command (see table 127) is a third-party copy command (see 5.16.3) that requests the copy manager to copy data from one set of copy sources (e.g., a set of source logical units) to a set of copy destinations (e.g., a set of destination logical units). The transfers requested by an EXTENDED COPY command are managed by a copy manager (see SAM-5 and 5.16.2).

The OPERATION CODE field is defined in 4.2.5.1 and shall be set as shown in table 127 for the EXTENDED COPY(LID4) command.

The SERVICE ACTION field is defined in 4.2.5.2 and shall be set as shown in table 127 for the EXTENDED COPY(LID4) command.

Table 127 — EXTENDED COPY(LID4) command

Bit / Byte	7	6	5	4	3	2	1	0
0	OPERATION CODE (83h)							
1	Reserved			SERVICE ACTION (01h)				
2	Reserved							
•••								
9								
10	(MSB)			PARAMETER LIST LENGTH				
•••								
13								(LSB)
14	Reserved							
15	CONTROL							

The PARAMETER LIST LENGTH field is defined in 4.2.5.5. A parameter list length of zero specifies that the copy manager shall not transfer any data or alter any internal state, and this shall not be considered an error. If the parameter list length causes truncation of the parameter list, then the copy manager shall transfer no data and shall terminate the EXTENDED COPY command with CHECK CONDITION status, with the sense key set to ILLEGAL REQUEST, and the additional sense code set to PARAMETER LIST LENGTH ERROR.

The CONTROL byte is defined in SAM-5.

6.4.2 EXTENDED COPY(LID4) parameter data

The format of the EXTENDED COPY(LID4) parameter list (see 5.16.7.1) is shown in table 128.

Table 128 — EXTENDED COPY(LID4) parameter list (part 1 of 2)

Bit / Byte	7	6	5	4	3	2	1	0
0	PARAMETER LIST FORMAT (01h)							
1	Reserved		STR	LIST ID USAGE		PRIORITY		
2	(MSB)			HEADER CSCD DESCRIPTOR LIST LENGTH (0020h)				
3								(LSB)
4				Reserved				
•••								
14								
15				Reserved			G_SENSE	IMMED
16	HEADER CSCD DESCRIPTOR TYPE CODE (FFh)							
17				Reserved				
18								
19								
20	(MSB)			LIST IDENTIFIER				
•••								
23								(LSB)
24				Reserved				
•••								
41								
42	(MSB)			CSCD DESCRIPTOR LIST LENGTH (n-47)				
43								(LSB)
44	(MSB)			SEGMENT DESCRIPTOR LIST LENGTH (m-n)				
45								(LSB)
46	(MSB)			INLINE DATA LENGTH (k-m)				
47								(LSB)
	CSCD descriptor list							
48	CSCD descriptor [ID 1] (see 6.4.5)							
•••								
79								
	⋮							
n-31	CSCD descriptor [ID x] (see 6.4.5)							
•••								
n								

Table 128 — EXTENDED COPY(LID4) parameter list (part 2 of 2)

Bit / Byte	7	6	5	4	3	2	1	0
	Segment descriptor list							
n+1	Segment descriptor (see 6.4.6) [first]							
•••								
	⋮							
	Segment descriptor (see 6.4.6) [last]							
•••								
m								
m+1	Inline data							
•••								
k								

The PARAMETER LIST FORMAT field (see table 129) specifies the format of the EXTENDED COPY(LID4) parameter list and shall be set as shown in table 128 for the EXTENDED COPY(LID4) command defined in this standard.

Table 129 — PARAMETER LIST FORMAT field

Code	Description
01h	The format shown in table 128
all others	Reserved

The STR bit is defined in 6.4.3.1.

The LIST ID USAGE field is defined in 6.4.3.2.

The PRIORITY field is defined in 6.4.3.3.

The HEADER CSCD DESCRIPTOR LIST LENGTH field shall be set as shown in table 128. For compatibility with SPC-3, this field specifies the length of one CSCD descriptor (i.e., a target descriptor in SPC-3 terms) in which the descriptor type is invalid. The EXTENDED COPY(LID4) command parameter list format replaces the one descriptor with the fields shown in table 128.

The good with sense data (G_SENSE) bit specifies whether the copy manager is required to include sense data with GOOD status. If the G_SENSE bit is set to zero, then the copy manager shall not include sense data with any command that completes with GOOD status. If the G_SENSE bit is set to one and the copy manager completes the EXTENDED COPY(LID4) command with GOOD status, then the copy manager shall include sense data with the GOOD status in which the sense key is set to COMPLETED, the additional sense code is set to EXTENDED COPY INFORMATION AVAILABLE, and the COMMAND-SPECIFIC INFORMATION field is set to the number of segment descriptors the copy manager has processed.

The immediate (IMMED) bit specifies whether the copy manager returns status for the EXTENDED COPY(LID4) command before the first segment descriptor is processed (see 5.16.7.2). Processing of the IMMED bit is described in 5.16.4.3.

The copy manager shall terminate the EXTENDED COPY(LID4) command with CHECK CONDITION status, with the sense key set to ILLEGAL REQUEST, and the additional sense code set to INVALID FIELD IN CDB if the IMMED bit is set to one and:

a) the G_SENSE bit is set to one; or
b) the LIST ID USAGE field (see 6.4.3.2) is set to 11b.

For interoperability with the EXTENDED COPY command defined in SPC-3, the HEADER CSCD DESCRIPTOR TYPE CODE field shall be set as shown in table 128.

The LIST IDENTIFIER field is defined in 6.4.3.2.

The CSCD DESCRIPTOR LIST LENGTH field is defined in 6.4.3.4.

The SEGMENT DESCRIPTOR LIST LENGTH field is defined in 6.4.3.5.

The INLINE DATA LENGTH field is defined in 6.4.3.6.

The CSCD descriptors are defined in 6.4.3.4 and 6.4.5.

The segment descriptors are defined in 6.4.3.5 and 6.4.6.

The inline data is defined in 6.4.3.6.

6.4.3 Shared EXTENDED COPY parameter list fields

6.4.3.1 STR bit

A sequential striped (STR) bit set to one specifies to the copy manager that the majority of the block device references in the parameter list represent sequential access of several block devices that are striped. This may be used by the copy manager to perform reads from a copy source block device at any time and in any order during processing of an EXTENDED COPY command as described in 6.4.5.3. A STR bit set to zero specifies to the copy manager that disk references, if any, may not be sequential.

6.4.3.2 LIST IDENTIFIER field and LIST ID USAGE field

The LIST IDENTIFIER field identifies the copy operation (see 5.16.4.3) originated by the EXTENDED COPY command to the copy manager as described in 5.16.4.2. The usage of the LIST IDENTIFIER field depends on the EXTENDED COPY command being processed and contents of the LIST ID USAGE field as follows:

a) if the command being processed is an EXTENDED COPY(LID4) command (see 6.4), then the LIST ID USAGE field specifies the usage of the LIST IDENTIFIER field as shown in table 130; and
b) if the command being processed is an EXTENDED COPY(LID1) command (see 6.5), then the LIST ID USAGE field specifies the usage of the LIST IDENTIFIER field as shown in table 131.

Table 130 — LIST ID USAGE field for the EXTENDED COPY(LID4) command

Value	Meaning
00b	The contents of the LIST IDENTIFIER field are defined in 5.16.4.2. The list identifier value may be used to abort (see 6.3) or to request status for a specific command sent on a specific I_T nexus (e.g., using the RECEIVE COPY STATUS(LID4) command (see 6.24)). The copy manager shall hold data, if any, for retrieval by the application client as described in 5.16.4.5.
01b	Reserved
10b	The contents of the LIST IDENTIFIER field are defined in 5.16.4.2. The list identifier value may be used to abort (see 6.3) or to request status for a specific command sent on a specific I_T nexus (e.g., using the RECEIVE COPY STATUS(LID4) command (see 6.24)). The copy manager may discard all held data (see 5.16.4.5) accessible to the application client. If the application client requests delivery of data that has been discarded as a result of the LIST ID USAGE field being set to 10b, then the copy manager shall respond as if the EXTENDED COPY(LID4) command has not been processed.
11b	If the LIST IDENTIFIER field is not set to zero, then the copy manager shall terminate the command with CHECK CONDITION status, with the sense key set to ILLEGAL REQUEST, and the additional sense code set to INVALID FIELD IN PARAMETER LIST. The copy manager shall discard all data accessible to the application client (e.g., using the RECEIVE COPY STATUS(LID4) command (see 6.24)). If the application client requests delivery of data that has been discarded as a result the LIST ID USAGE field being set to 11b then the copy manager shall respond as if the EXTENDED COPY(LID4) command has not been processed. If the parameter list contains any segment descriptors (see 6.4.6) that require data to be held for the application client (e.g., the block→block&application client segment descriptor (see 6.4.6.4)), then the copy manager shall terminate the copy operation (see 5.16.4.3) with CHECK CONDITION status, with the sense key set to ILLEGAL REQUEST, and the additional sense code set to INVALID FIELD IN PARAMETER LIST, and no segment descriptors shall be processed (see 5.16.7.3).

Table 131 — LIST ID USAGE field for the EXTENDED COPY(LID1) command

Value	SNLID bit [a]	Meaning
00b	0 or 1	The contents of the LIST IDENTIFIER field are defined in 5.16.4.2. The list identifier value may be used to abort (see 6.3) or to request status for a specific command sent on a specific I_T nexus (e.g., using the RECEIVE COPY STATUS(LID4) command (see 6.24)). The copy manager shall hold data, if any, for retrieval by the application client as described in 5.16.4.5.
01b		Reserved
10b	0 or 1	The contents of the LIST IDENTIFIER field are defined in 5.16.4.2. The list identifier value may be used to abort (see 6.3) or to request status for a specific command sent on a specific I_T nexus (e.g., using the RECEIVE COPY STATUS(LID4) command (see 6.24)). The copy manager may discard all held data (see 5.16.4.5) accessible to the application client. If the application client requests delivery of data that has been discarded as a result of the LIST ID USAGE field being set to 10b, then the copy manager shall respond as if the EXTENDED COPY(LID1) command has not been processed.
11b	0	The copy manager shall: a) terminate the command with CHECK CONDITION status with the sense key set to ILLEGAL REQUEST, and the additional sense code set to INVALID FIELD IN PARAMETER LIST; or b) process the command as if the LIST ID USAGE field is set to 10b.
	1	If the LIST IDENTIFIER field is not set to zero, the copy manager shall terminate the command with CHECK CONDITION status, with the sense key set to ILLEGAL REQUEST, and the additional sense code set to INVALID FIELD IN PARAMETER LIST. The copy manager shall discard all data accessible to the application client (e.g., using the RECEIVE COPY STATUS(LID4) command (see 6.24)). If the application client requests delivery of data that has been discarded as a result the LIST ID USAGE field being set to 11b then the copy manager shall respond as if the EXTENDED COPY(LID1) command has not been processed. If the parameter list contains any segment descriptors (see 6.4.6) that require data to be held for the application client (e.g., the block→block&application client segment descriptor (see 6.4.6.4)), then the copy manager shall terminate the copy operation (see 5.16.4.3) with CHECK CONDITION status, with the sense key set to ILLEGAL REQUEST, and the additional sense code set to INVALID FIELD IN PARAMETER LIST, and no segment descriptors shall be processed (see 5.16.7.3).

[a] See the RECEIVE COPY OPERATING PARAMETERS command (see 6.22).

During the processing of a copy operation (see 5.16.4.3) originated by the EXTENDED COPY command with the LIST ID USAGE field set to 00b or 10b, for any EXTENDED COPY command received into the task set the copy manager shall:

a) not start a copy operation; and
b) terminate the command with CHECK CONDITION status, with the sense key set to ILLEGAL REQUEST, and the additional sense code set to INVALID FIELD IN PARAMETER LIST,

if:

a) the EXTENDED COPY command that does not have a copy operation being processed includes:

A) a LIST IDENTIFIER field set to a specific value (e.g., a LIST IDENTIFIER field set to 50h); and
B) a parameter list in one format (i.e., EXTENDED COPY(LID4) command parameter list format or EXTENDED COPY(LID1) command parameter list format);

and

b) the copy operation being processed was originated by an EXTENDED COPY command that included:
A) a LIST IDENTIFIER field set to the same value (e.g., a LIST IDENTIFIER field set to 50h); and
B) a parameter list in the other format.

The copy manager may respond as if the EXTENDED COPY command had never been received, if an EXTENDED COPY command that had the LIST ID USAGE field set to 10b or 11b in its parameter list is specified by the LIST IDENTIFIER field in one of the following commands:

a) the RECEIVE COPY STATUS(LID4) command (see 6.24);
b) the RECEIVE COPY STATUS(LID1) command (see 6.25);
c) the RECEIVE COPY DATA(LID4) command (see 6.20);
d) the RECEIVE COPY DATA(LID1) command (see 6.21);
e) the RECEIVE COPY FAILURE DETAILS(LID1) command (see 6.23); and
f) the RECEIVE ROD TOKEN INFORMATION command (see 6.28).

6.4.3.3 PRIORITY field

The PRIORITY field specifies the priority of data transfers resulting from this EXTENDED COPY command relative to data transfers resulting from other commands being processed by the device server contained within the same logical unit as the copy manager. All commands other than third-party copy commands have a priority of 1h. Priority 0h is the highest priority, with increasing values in the PRIORITY field indicating lower priorities.

6.4.3.4 CSCD DESCRIPTOR LIST LENGTH field and CSCD descriptor list

The CSCD DESCRIPTOR LIST LENGTH field specifies the length in bytes of the CSCD descriptor list that follows the parameter list header (see 5.16.7.1). An EXTENDED COPY command may reference one or more CSCDs. Each CSCD is described by a CSCD descriptor (see 6.4.5).

The maximum number of CSCD descriptors permitted within a parameter list is indicated by the MAXIMUM CSCD DESCRIPTOR COUNT field in the Third-party Copy VPD page Parameter Data descriptor (see 7.8.17.5), and in the parameter data for the RECEIVE COPY OPERATING PARAMETERS command (see 6.22). If the number of CSCD descriptors exceeds the allowed number, the copy manager shall terminate the command with CHECK CONDITION status, with the sense key set to ILLEGAL REQUEST, and the additional sense code set to TOO MANY TARGET DESCRIPTORS.

6.4.3.5 SEGMENT DESCRIPTOR LIST LENGTH field and segment descriptor list

The SEGMENT DESCRIPTOR LIST LENGTH field specifies the length in bytes of the segment descriptor list that follows the CSCD descriptors (see 5.16.7.1). See 6.4.6 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 DESCRIPTOR COUNT field in the Third-party Copy VPD page Parameter Data descriptor (see 7.8.17.5), and in the parameter data for the RECEIVE COPY OPERATING PARAMETERS command (see 6.22). If the number of segment descriptors exceeds the allowed number, the copy manager shall terminate the command with CHECK CONDITION status, with the sense key set to ILLEGAL REQUEST, and the additional sense code set to TOO MANY SEGMENT DESCRIPTORS.

The maximum combined length of the CSCD descriptors and segment descriptors permitted within a parameter list is indicated by the MAXIMUM DESCRIPTOR LIST LENGTH field in the Third-party Copy VPD page Parameter Data descriptor (see 7.8.17.5), and in the parameter data for the RECEIVE COPY OPERATING PARAMETERS command (see 6.22). If the combined length of the CSCD descriptors and segment

descriptors exceeds the allowed value, then the copy manager shall terminate the command with CHECK CONDITION status, with the sense key set to ILLEGAL REQUEST, and the additional sense code set to PARAMETER LIST LENGTH ERROR.

6.4.3.6 INLINE DATA LENGTH field and inline data

The INLINE DATA LENGTH field specifies the number of bytes of inline data, following the last segment descriptor (see 5.16.7.1). A value of zero specifies that no inline data is present.

The inline data contains information that is available for:

a) reference by CSCD descriptors; or
b) transfer by the copy manager in response to segment descriptors.

6.4.4 Descriptor type codes

CSCD descriptors, the CSCD descriptor extension, and segment descriptors share a single set of code values (see table 132) that identify the type of descriptor.

Table 132 — EXTENDED COPY descriptor type codes

Descriptor type	Description	Reference
00h to BFh	Segment descriptors	6.4.6
C0h to DFh	Vendor specific descriptors	
E0h to FEh	CSCD descriptors	6.4.5
FFh [a]	CSCD descriptor extension	6.4.5.2

[a] Use of this descriptor type code is reserved except in byte 32 of a 64-byte CSCD descriptor (e.g., the IPv6 CSCD descriptor described in 7.6.3.9).

6.4.5 CSCD descriptors

6.4.5.1 CSCD descriptors introduction

The descriptor type code (see table 132) values for CSCD descriptors are shown in table 133.

Table 133 — EXTENDED COPY CSCD descriptor type codes

Descriptor type [a]	Name	Size (bytes)	Reference
E0h	Fibre Channel N_Port_Name CSCD descriptor	32	7.6.3.2
E1h	Fibre Channel N_Port_ID CSCD descriptor	32	7.6.3.3
E2h	Fibre Channel N_Port_ID With N_Port_Name Checking CSCD descriptor	32	7.6.3.4
E3h	Parallel Interface T_L CSCD descriptor	32	7.6.3.5
E4h	Identification Descriptor CSCD descriptor	32	6.4.5.6
E5h	IPv4 CSCD descriptor	32	7.6.3.8
E6h	Alias CSCD descriptor	32	6.4.5.7
E7h	RDMA CSCD descriptor	32	7.6.3.7
E8h	IEEE 1394 EUI-64 CSCD descriptor	32	7.6.3.6
E9h	SAS Serial SCSI Protocol CSCD descriptor	32	7.6.3.10
EAh	IPv6 CSCD descriptor	64	7.6.3.9
EBh	IP Copy Service CSCD descriptor	64	6.4.5.8
ECh to FDh	Reserved for CSCD descriptors		
FEh	ROD CSCD descriptor [b]	32	6.4.5.9

[a] A copy manager may not support all CSCD descriptor types; however, the copy manager shall list all supported CSCD descriptor types in the Third-party Copy VPD page Supported Descriptors descriptor (see 7.8.17.6), and in the parameter data for the RECEIVE COPY OPERATING PARAMETERS command (see 6.22).

[b] A copy manager that implements the ROD CSCD descriptor shall implement:

a) the following third-party copy descriptors in the Third-party Copy VPD page:
   A) the ROD Features third-party copy descriptor (see 7.8.17.8); and
   B) the Supported ROD Types third-party copy descriptor (see 7.8.17.9); and

b) at least one of the following segment descriptors:
   A) the Populate a ROD from one or more block device ranges (see 6.4.6.20); or
   B) the Populate a ROD from one block device range (see 6.4.6.21).

All CSCD descriptors (see table 134) are a multiple of 32 bytes in length and begin with a four-byte header containing the DESCRIPTOR TYPE CODE field that specifies the format of the descriptor. If a copy manager receives an unsupported descriptor type code in a CSCD descriptor, the copy manager shall terminate the copy operation (see 5.16.4.3) with CHECK CONDITION status, with the sense key set to ILLEGAL REQUEST, and the additional sense code set to UNSUPPORTED TARGET DESCRIPTOR TYPE CODE.

Table 134 — CSCD descriptor format

Bit / Byte	7	6	5	4	3	2	1	0
0	DESCRIPTOR TYPE CODE (E0h to FEh)							
1	LU ID TYPE		Obsolete	PERIPHERAL DEVICE TYPE				
2	(MSB)			RELATIVE INITIATOR PORT IDENTIFIER				
3								(LSB)
4				CSCD descriptor parameters				
•••								
27								
28				Device type specific parameters				
•••								
31								
32				Zero or more CSCD descriptor extensions (see 6.4.5.2)				
•••								
n								

The DESCRIPTOR TYPE CODE field is described in 6.4.4.

The LU ID TYPE field (see table 135) specifies the interpretation of the LU IDENTIFIER field in CSCD descriptors that contain a LU IDENTIFIER field.

Table 135 — LU ID TYPE field

Code	LU IDENTIFIER field contents	Support	Reference
00b	Logical Unit Number	Mandatory	SAM-5
01b	Proxy Token	Optional	8.3.1.6.2
10b to 11b	Reserved		

If a copy manager receives an unsupported value in the LU ID TYPE field, the copy manager shall terminate the copy operation (see 5.16.4.3) with CHECK CONDITION status, with the sense key set to ILLEGAL REQUEST, and the additional sense code set to INVALID FIELD IN PARAMETER LIST.

If the LU ID TYPE field specifies that the LU IDENTIFIER field contains a logical unit number, then the LU IDENTIFIER field specifies the logical unit within the SCSI device specified by other fields in the CSCD descriptor that shall be the copy source or copy destination.

If the LU ID TYPE field specifies that the LU IDENTIFIER field contains a proxy token (see 8.3.1.6.2), then the copy manager shall use the LU IDENTIFIER field contents to obtain proxy access rights to the logical unit associated with the proxy token. The logical unit number that represents the proxy access rights shall be the copy source or copy destination.

The copy manager should obtain a LUN value for addressing this logical unit by sending an ACCESS CONTROL OUT command with ASSIGN PROXY LUN service action (see 8.3.3.11) to the access controls coordinator of the SCSI target device that is specified by other fields in the CSCD descriptor. The copy manager shall use a LUN assigned on the basis of a proxy token only for those commands that are necessary

for the processing of the EXTENDED COPY command whose parameter data contains the proxy token. After the copy manager has completed EXTENDED COPY commands involving a proxy token, the copy manager should release the LUN value using an ACCESS CONTROL OUT command with RELEASE PROXY LUN service action (see 8.3.3.12).

The copy manager shall only access proxy logical units if if the LU ID TYPE field is set to 01b. If the copy manager receives a CSCD descriptor containing LU ID type 00b and a logical unit number matching a LUN value that the copy manager has obtained using an ACCESS CONTROL OUT command with ASSIGN PROXY LUN service action, then the copy manager shall terminate the copy operation (see 5.16.4.3) with CHECK CONDITION status, with the sense key set to COPY ABORTED, and the additional sense code set to COPY TARGET DEVICE NOT REACHABLE.

The PERIPHERAL DEVICE TYPE field is described in 6.6.2. The value in the PERIPHERAL DEVICE TYPE field (see table 134) specifies the format of the device type specific parameters. The device type specific parameters convey information specific to the type of device specified by the CSCD descriptor.

Table 136 lists the peripheral device type code values having formats defined for the device type specific parameters in a CSCD descriptor. Peripheral device types with code values not listed in table 136 are reserved in all PERIPHERAL DEVICE TYPE fields in the EXTENDED COPY parameter list.

Table 136 — Device type specific parameters in CSCD descriptors

Peripheral Device Type	Reference	Description	Name
00h, 05h, and 0Eh	6.4.5.3	Block devices	Block
01h	6.4.5.4	Sequential-access devices	Stream or Tape
03h	6.4.5.5	Processor devices [a]	Stream
[a] This standard defines the use of the processor device type (i.e., 03h) (see SPC-2) only for the interactions between copy managers.			

The RELATIVE INITIATOR PORT IDENTIFIER field (see 4.3.4) specifies the relative port identifier of the initiator port within the SCSI device that the copy manager shall use to access the logical unit described by the CSCD descriptor, if such access requires use of an initiator port (i.e., if the logical unit is in the same SCSI device as the copy manager, the RELATIVE INITIATOR PORT IDENTIFIER field is ignored). A RELATIVE INITIATOR PORT IDENTIFIER field set to zero specifies that the copy manager may use any initiator port or ports within the SCSI device.

CSCD descriptor extensions increase the length of a CSCD descriptor as described in 6.4.5.2.

6.4.5.2 The CSCD descriptor extension

A CSCD descriptor (see 6.4.5.1) is extended by appending one or more CSCD descriptor extensions (see table 137) to it. A CSCD descriptor extension is 32 bytes in length and begins with a one-byte header containing the descriptor type code value that identifies the descriptor as a CSCD descriptor extension (i.e., FFh).

Table 137 — CSCD descriptor extension format

Bit / Byte	7	6	5	4	3	2	1	0
0	EXTENSION DESCRIPTOR TYPE CODE (FFh)							
1	CSCD descriptor specific information							
•••								
31								

The EXTENSION DESCRIPTOR TYPE CODE field is a descriptor type code value (see 6.4.4), and shall be set to the value shown in table 137 for each CSCD descriptor extension.

The CSCD descriptor specific information contains data that extends the CSCD descriptor or CSCD descriptor extension located adjacent to and preceding this CSCD descriptor extension.

If a SOURCE CSCD DESCRIPTOR ID field or a DESTINATION CSCD DESCRIPTOR ID field in a segment descriptor (see 6.4.6.1) references a CSCD descriptor extension (i.e, a descriptor with the descriptor type code set to FFh), then the copy manager shall terminate the copy operation (see 5.16.4.3) with CHECK CONDITION status, with the sense key set to COPY ABORTED, and the additional sense code set to INVALID FIELD IN PARAMETER LIST.

6.4.5.3 Device type specific CSCD descriptor parameters for block device types

The format for the device type specific CSCD descriptor parameters for block device types (i.e., device type code values 00h, 05h, and 0Eh) is shown in table 138.

Table 138 — Device type specific CSCD 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 6.4.6.1) in the segment descriptor to determine what action should be taken if a segment of the copy does not fit exactly into an integer number of destination logical blocks (see 5.16.7.2).

The DISK BLOCK LENGTH field is set to the number of bytes in a logical block, excluding any protection information (see SBC-3), for the logical unit being addressed.

If the DISK BLOCK LENGTH field is set to zero and the PERIPHERAL DEVICE TYPE field (see 6.4.5.1) is set to 00h, then the copy manager shall determine the logical block length of the CSCD logical unit (e.g., by sending a READ CAPACITY command (see SBC-3)), and use the result wherever the use of the DISK BLOCK LENGTH field is required by this standard.

The copy manager may read ahead from copy sources of the block device type (i.e., the copy manager may perform reads from a copy source block device 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 logical blocks within the same CSCD descriptor does not differ from their order in the segment descriptor list (see 5.16.7.1).

6.4.5.4 Device type specific CSCD descriptor parameters for sequential-access device types

The format for the device type specific CSCD descriptor parameters for the sequential-access device type (i.e., device type code value 01h) operating in the implicit address mode (see SSC-4) is shown in table 139.

Table 139 — Device type specific CSCD 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 as defined in table 140.

Table 140 — Stream device transfer lengths

FIXED bit	STREAM BLOCK LENGTH field	Description
0	000000h	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.
	000001h to FFFFFFh	The copy manager shall terminate the copy operation (see 5.16.4.3) with CHECK CONDITION status, with the sense key set to ILLEGAL REQUEST, and the additional sense code set to INVALID FIELD IN PARAMETER LIST.
1	000000h	The copy manager shall terminate the copy operation (see 5.16.4.3) with CHECK CONDITION status, with the sense key set to ILLEGAL REQUEST, and the additional sense code set to INVALID FIELD IN PARAMETER LIST
	000001h to FFFFFFh	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 5.16.7.2) in the segment descriptor to determine what action should be taken if a segment of the copy does not fit exactly into an integer number of destination logical blocks (see 5.16.7.2).

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

The copy manager shall not read ahead from copy sources of the stream device type (i.e., the reads required by a segment descriptor for which the copy source is a stream device shall not be started until all writes for previous segment descriptors have completed).

6.4.5.5 Device type specific CSCD descriptor parameters for processor device types

The format for the device type specific CSCD descriptor parameters for the processor device type (i.e., device type code value 03h) (see SPC-2) is shown in table 141.

Table 141 — Device type specific CSCD 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 5.16.7.2) in the segment descriptor to determine what action should be taken if a segment of the copy does not fit exactly into an integer number of EXTENDED COPY(LID4) commands (see 6.4), EXTENDED COPY(LID1) commands (see 6.5), RECEIVE COPY DATA(LID4) commands (see 6.20), or RECEIVE COPY DATA(LID1) commands (see 6.21).

If the processor device is a copy source, the number of bytes to be transferred by an EXTENDED COPY command shall be specified by STREAM DEVICE TRANSFER LENGTH field in the segment descriptor. If the processor device is a copy destination, the number of bytes to be transferred by an EXTENDED COPY command and subsequent RECEIVE COPY DATA command shall be specified by STREAM DEVICE TRANSFER LENGTH field in the segment descriptor.

6.4.5.6 Identification Descriptor CSCD descriptor format

The CSCD descriptor format shown in table 142 requests the copy manager to locate a SCSI target device and logical unit that returns a Device Identification VPD page (see 7.8.6) containing an Identification descriptor having the specified values in the CODE SET field, ASSOCIATION field, DESIGNATOR TYPE field, IDENTIFIER LENGTH field, and DESIGNATOR field. The copy manager may use any SCSI transport protocol (see SAM-5), target port identifier (see SAM-5) and logical unit number (see SAM-5) values that result in matching VPD field values to address the logical unit. If multiple combinations of SCSI transport protocols, target port identifiers, and logical unit numbers access matching VPD field values, then the copy manager may use any combination to address the logical unit and shall try other combinations in the event that one combination becomes non-operational during the processing of an EXTENDED COPY command.

Table 142 — Identification Descriptor CSCD descriptor format

Bit / Byte	7	6	5	4	3	2	1	0
0	DESCRIPTOR TYPE CODE (E4h)							
1	LU ID TYPE		Obsolete	PERIPHERAL DEVICE TYPE				
2	(MSB)							
3	RELATIVE INITIATOR PORT IDENTIFIER							(LSB)
4	Reserved				CODE SET			
5	Reserved		ASSOCIATION		DESIGNATOR TYPE			
6	Reserved							
7	DESIGNATOR LENGTH							
8	DESIGNATOR							
•••								
27								
28	Device type specific parameters							
•••								
31								

The DESCRIPTOR TYPE CODE field, PERIPHERAL DEVICE TYPE field, RELATIVE INITIATOR PORT IDENTIFIER field, and the device type specific parameters are described in 6.4.5.1. The DESCRIPTOR TYPE CODE field shall be set as shown in table 142 for the Identification Descriptor CSCD descriptor.

The LU ID TYPE field shall be ignored in the Identification Descriptor CSCD descriptor.

The CODE SET field contains a code set enumeration (see 4.3.3) that indicates the format of the DESIGNATOR field.

The contents of the ASSOCIATION field, DESIGNATOR TYPE field, and DESIGNATOR LENGTH field are described in 7.8.6.1. The designator length shall be 20 or less.

The DESIGNATOR field is a fixed-length zero-padded (see 4.3.2) field that has the DESIGNATOR field format defined in 7.8.6.

Some combinations of code set, association, designator type, designator length and designator do not uniquely identify a logical unit to serve as a CSCD. The behavior of the copy manager if such combinations are specified is outside the scope of this standard.

6.4.5.7 Alias CSCD descriptor format

The CSCD descriptor format shown in table 143 requests the copy manager to locate a SCSI target port and logical unit using the alias list designation associated with the specified alias value. The alias list is maintained using the CHANGE ALIASES command (see 6.2).

Table 143 — Alias CSCD descriptor format

Bit / Byte	7	6	5	4	3	2	1	0
0	DESCRIPTOR TYPE CODE (E6h)							
1	LU ID TYPE		Obsolete	PERIPHERAL DEVICE TYPE				
2	(MSB)			RELATIVE INITIATOR PORT IDENTIFIER				
3								(LSB)
4				LU IDENTIFIER				
•••								
11								
12				ALIAS VALUE				
•••								
19								
20				Reserved				
•••								
27								
28				Device type specific parameters				
•••								
31								

The DESCRIPTOR TYPE CODE field, LU ID TYPE field, PERIPHERAL DEVICE TYPE field, RELATIVE INITIATOR PORT IDENTIFIER field, and the device type specific parameters are described in 6.4.5.1. The DESCRIPTOR TYPE CODE field shall be set as shown in table 143 for the Alias CSCD descriptor.

The ALIAS VALUE field specifies an alias value in the alias list as managed by the CHANGE ALIASES command (see 6.2) and maintained by the device server.

When the copy manager first processes an Alias CSCD descriptor, it shall check the value of the ALIAS VALUE field for a corresponding entry in the alias list. If the value is not in the alias list or the copy manager is unable to validate the designation (see 6.2.3) associated with the alias value, then the copy manager shall terminate the copy operation (see 5.16.4.3) because the CSCD is unreachable (see 5.16.7.4). An application client generating EXTENDED COPY commands that include Alias CSCD descriptors in the parameter list is responsible for providing a valid entry in the alias list using the CHANGE ALIASES command (see 6.2) prior to sending the EXTENDED COPY command.

6.4.5.8 IP Copy Service CSCD descriptor

The CSCD descriptor format shown in table 144 requests the copy manager to communicate with the copy service specified in the IP ADDRESS field to locate the CSCD that returns a Device Identification VPD page (see 7.8.6) containing an Identification descriptor having the specified CODE SET field, ASSOCIATION field, DESIGNATOR TYPE field, DESIGNATOR LENGTH field, and DESIGNATOR field values.

The protocol used by the copy manager to communicate with the copy service is vendor specific.

Table 144 — IP Copy Service CSCD descriptor format

Bit / Byte	7	6	5	4	3	2	1	0
0	DESCRIPTOR TYPE CODE (EBh)							
1	LU ID TYPE		Obsolete	PERIPHERAL DEVICE TYPE				
2	Reserved							IPTYPE
3	Reserved							
•••								
11								
12	(MSB)			COPY SERVICE IP ADDRESS				
•••								
27								(LSB)
28	Device type specific parameters							
31								
32	EXTENSION DESCRIPTOR TYPE CODE (FFh)							
33	Reserved							
35								
36	(MSB)			COPY SERVICE PORT NUMBER				
37								(LSB)
38	(MSB)			COPY SERVICE INTERNET PROTOCOL NUMBER				
39								(LSB)
40	Reserved					CODE SET		
41	Reserved		ASSOCIATION			DESIGNATOR TYPE		
42	Reserved							
43	DESIGNATOR LENGTH							
44	DESIGNATOR							
•••								
63								

The DESCRIPTOR TYPE CODE field, PERIPHERAL DEVICE TYPE field, RELATIVE INITIATOR PORT IDENTIFIER field, and the device type specific parameters are described in 6.4.5.1. The DESCRIPTOR TYPE CODE field shall be set as shown in table 144 for the IP Copy Service CSCD descriptor.

The LU ID TYPE field shall be ignored in the IP Copy Service CSCD descriptor.

If the IPTYPE bit is set to zero, the COPY SERVICE IP ADDRESS field shall contain a zero-padded IPv4 address (see RFC 791). If the IPTYPE bit is set to one, the COPY SERVICE IP ADDRESS field shall contain a unicast IPv6 address (see RFC 4291).

The COPY SERVICE IP ADDRESS field is set to the IP address of the copy service in the format specified by the IPTYPE bit.

The EXTENSION DESCRIPTOR TYPE CODE field is described in 6.4.5.2. If the EXTENSION DESCRIPTOR TYPE CODE field does not contain the value shown in table 144, then the copy manager shall terminate the EXTENDED COPY command with CHECK CONDITION status, with the sense key set to COPY ABORTED, and the additional sense code set to INVALID FIELD IN PARAMETER LIST.

The COPY SERVICE PORT NUMBER field shall contain the TCP port number.

The COPY SERVICE INTERNET PROTOCOL NUMBER field shall contain an Internet protocol number.

The contents of the COPY SERVICE IP ADDRESS field, the COPY SERVICE PORT NUMBER field, and the COPY SERVICE INTERNET PROTOCOL NUMBER field may be obtained from the network service descriptor with the SERVICE TYPE field set to 06h (i.e., copy service) in the Management Network Addresses VPD page (see 7.8.8) returned by the logical unit that returns a Device Identification VPD page (see 7.8.6) containing an Identification descriptor having the specified CODE SET field, ASSOCIATION field, DESIGNATOR TYPE field, DESIGNATOR LENGTH field, and DESIGNATOR field values.

The CODE SET field contains a code set enumeration (see 4.3.3) that indicates the format of the DESIGNATOR field.

The contents of the ASSOCIATION field, DESIGNATOR TYPE field, and DESIGNATOR LENGTH field are described in 7.8.6.1. The designator length shall be 20 or less.

The DESIGNATOR field is a fixed-length zero-padded (see 4.3.2) field that has the DESIGNATOR field format defined in 7.8.6.

Some combinations of code set, association, designator type, designator length and designator do not uniquely identify a logical unit to serve as a CSCD. The behavior of the copy manager if such combinations are specified is outside the scope of this standard.

6.4.5.9 ROD CSCD descriptor

The CSCD descriptor format shown in table 145 requests the copy manager to use the specified ROD as a copy source or copy destination. If the ROD represents no data when the copy manager begins processing the EXTENDED COPY command (e.g., the ROD TYPE field is not set to zero), then the copy manager shall allow the ROD to be populated as defined in 5.16.7.5.2. If a segment descriptor attempts to use an unpopulated ROD as a copy source or copy destination, then the copy manager shall terminate the copy operation (see 5.16.4.3) as if an unreachable CSCD had been encountered (see 5.16.7.4).

Table 145 — ROD CSCD descriptor format

Bit / Byte	7	6	5	4	3	2	1	0
0	DESCRIPTOR TYPE CODE (FEh)							
1	LU ID TYPE		Obsolete	PERIPHERAL DEVICE TYPE				
2	(MSB)			RELATIVE INITIATOR PORT IDENTIFIER				
3								(LSB)
4	(MSB)			ROD PRODUCER CSCD DESCRIPTOR ID				
5								(LSB)
6				Reserved				
7								
8	(MSB)			ROD TYPE				
•••								
11								(LSB)
12	(MSB)			REQUESTED ROD TOKEN LIFETIME				
•••								
15								(LSB)
16	(MSB)			REQUESTED ROD TOKEN INACTIVITY TIMEOUT				
•••								
19								(LSB)
20				Reserved				
21								
22				Reserved				R_TOKEN
23				Reserved				DEL_TKN
24	(MSB)			ROD TOKEN OFFSET				
•••								
27								(LSB)
28				Device type specific parameters				
•••								
31								

The DESCRIPTOR TYPE CODE field, PERIPHERAL DEVICE TYPE field, RELATIVE INITIATOR PORT IDENTIFIER field, and the device type specific parameters are described in 6.4.5.5. The DESCRIPTOR TYPE CODE field shall be set as shown in table 145 for the ROD CSCD descriptor.

The LU ID TYPE field shall be ignored in the ROD CSCD descriptor.

If the ROD TYPE field is set to zero, then:

a) the PERIPHERAL DEVICE TYPE field shall be ignored; and
b) the peripheral device type of the ROD CSCD descriptor shall be the peripheral device type specified by the ROD token that is specified by the ROD TOKEN OFFSET field.

The ROD PRODUCER CSCD DESCRIPTOR ID field specifies the CSCD device (see table 150 in 6.4.6.1) that contains the logical unit that contains the copy manager that:

a) created the ROD token that is specified by the ROD TOKEN OFFSET field, if the ROD TYPE field is set to zero; or
b) creates the ROD token that is created during processing of the EXTENDED COPY command, if the ROD TYPE field is not set to zero.

The ROD TYPE field (see table 111 in 5.16.6.2.1) specifies the type of ROD being populated or referenced by a ROD token.

For all ROD types (see 5.16.6.2), access to the bytes within a ROD is established by specifying:

a) the identifier of a ROD CSCD descriptor that describes the ROD for the duration of the copy operation (see 5.16.4.3) originated by an EXTENDED COPY command; or
b) that a ROD token be created that allows the copy manager that creates the ROD token creator to identify and access the bytes represented by the ROD across multiple third-party copy commands.

Methods are available to create a ROD token from a ROD identifier and create a ROD identifier from a ROD token (see 5.16.7.5.1).

Fields in the ROD CSCD descriptor and fields in the ROD Features third-party copy descriptor (see 7.8.17.8) in the Third-party Copy VPD page that affect the processing of the ROD PRODUCER CSCD DESCRIPTOR ID field are shown in table 146.

Table 146 — Inputs that affect the processing of the ROD PRODUCER CSCD DESCRIPTOR ID field

ROD PRODUCER CSCD DESCRIPTOR ID field	R_TOKEN bit	ROD TYPE field	REMOTE TOKENS field [a]	Description
n/a	1	zero	n/a	the command shall be terminated [b]
FFFFh	0	n/a	n/a	process the command
	1	not zero	n/a	process the command
F800h	0 or 1	not zero	n/a	the command shall be terminated [b]
	0	zero local [c]	n/a	process the command
		zero remote [c]	0h	remote ROD tokens not supported [d]
			4h	process the command
Others [e]	0	not zero	0h	remote ROD tokens not supported [d]
			4h	the command shall be terminated [b]
		zero	0h	remote ROD tokens not supported [d]
			4h	process the command
	1	not zero	0h	remote ROD tokens not supported [d]
			4h	remote ROD token creation not supported [f]
			6h	process the command

[a] Refers to the REMOTE TOKENS field is in the ROD Features third-party copy descriptor (see 7.8.17.8). Remote tokens values not shown in this table are n/a.

[b] The copy manager shall terminate the copy operation (see 5.16.4.3) with CHECK CONDITION status, with the sense key set to ILLEGAL REQUEST, and the additional sense code set to INVALID FIELD IN PARAMETER LIST.

[c] Processing depends on the copy manager that created the ROD token specified by the ROD TOKEN OFFSET field as follows:

a) **local:** a ROD token created by a copy manager that is contained within the same SCSI target device that is processing the copy operation originated by the EXTENDED COPY command; or

b) **remote:** a ROD token created by a copy manager that is not contained within the same SCSI target device that is processing the copy operation originated by the EXTENDED COPY command.

[d] The copy manager shall terminate the copy operation with CHECK CONDITION status, with the sense key set to ILLEGAL REQUEST, and the additional sense code set to INVALID TOKEN OPERATION, REMOTE ROD TOKEN USAGE NOT SUPPORTED.

[e] Any other CSCD descriptor ID (i.e., not FFFFh and not F800h) that specifies a logical unit not contained within the same SCSI target device as the processing copy manager.

[f] The copy manager shall terminate the copy operation with CHECK CONDITION status, with the sense key set to ILLEGAL REQUEST, and the additional sense code set to INVALID TOKEN OPERATION, REMOTE ROD TOKEN CREATION NOT SUPPORTED.

If the copy manager is unable to process the requested activities with the remote copy manager specified by the ROD PRODUCER CSCD DESCRIPTOR ID field necessary to use or create a remote ROD token, then the copy manager shall terminate the copy operation (see 5.16.4.3) as if an unreachable CSCD device had been encountered (see 5.16.7.4).

If the R_TOKEN bit is set to one, the REQUESTED ROD TOKEN LIFETIME field specifies the number of seconds the ROD token should remain valid (see 5.16.6.7).

If the REQUESTED ROD TOKEN LIFETIME field specifies a number of seconds that is less than the value in the MINIMUM TOKEN LIFETIME field in the ROD Features third-party copy descriptor (see 7.8.17.8), then the copy manager shall use the value in the MINIMUM TOKEN LIFETIME field. If the REQUESTED ROD TOKEN LIFETIME field specifies a number of seconds that is greater than the value in the MAXIMUM TOKEN LIFETIME field in the ROD Features third-party copy descriptor, then the copy manager shall terminate the copy operation (see 5.16.4.3) with CHECK CONDITION status, with the sense key set to ILLEGAL REQUEST, and the additional sense code set to INVALID FIELD IN PARAMETER LIST.

If the R_TOKEN bit is set to zero and the REQUESTED ROD TOKEN LIFETIME field is not set to zero, then the copy manager shall terminate the copy operation (see 5.16.4.3) with CHECK CONDITION status, with the sense key set to ILLEGAL REQUEST, and the additional sense code set to INVALID FIELD IN PARAMETER LIST.

If the R_TOKEN bit is set to one, the REQUESTED ROD TOKEN INACTIVITY TIMEOUT field specifies number of seconds that the copy manager should wait for the next third-party copy command that specifies the ROD token, if any, before invalidating that ROD token (see 5.16.6.7).

If the REQUESTED ROD TOKEN INACTIVITY TIMEOUT field specifies zero seconds, then the copy manager shall not invalidate the ROD token due to the lack of its active use. If the REQUESTED ROD TOKEN INACTIVITY TIMEOUT field specifies a number of seconds that is greater than the value in the MAXIMUM TOKEN INACTIVITY TIMEOUT field in the ROD Features third-party copy descriptor (see 7.8.17.8), then the copy manager shall terminate the copy operation (see 5.16.4.3) with CHECK CONDITION status, with the sense key set to ILLEGAL REQUEST, and the additional sense code set to INVALID FIELD IN PARAMETER LIST.

If the R_TOKEN bit is set to zero and the REQUESTED ROD TOKEN INACTIVITY TIMEOUT field is not set to zero, then the copy manager shall terminate the copy operation (see 5.16.4.3) with CHECK CONDITION status, with the sense key set to ILLEGAL REQUEST, and the additional sense code set to INVALID FIELD IN PARAMETER LIST.

Processing of the R_TOKEN bit depends on the value in the ROD TYPE field as follows:

a) If the ROD TYPE field is not set to zero, the R_TOKEN bit specifies whether a ROD token is to be returned for the ROD created during processing of the EXTENDED COPY command as follows:
   A) if the R_TOKEN bit is set to zero, a ROD token shall not be returned; or
   B) if the R_TOKEN bit is set to one and table 146 does not require termination of the EXTENDED COPY command with an error, then a ROD token shall be made available for retrieval using the RECEIVE ROD TOKEN INFORMATION command (see 6.28);

   or
b) if the ROD TYPE field is set to zero, the processing of the R_TOKEN bit shall be as shown in table 146.

If the LIST ID USAGE field (see 6.4.3.2) is set to a value other than 00b or 10b and the R_TOKEN bit is set to one, then the copy manager shall terminate the copy operation (see 5.16.4.3) with CHECK CONDITION status, with the sense key set to ILLEGAL REQUEST, and the additional sense code set to INVALID FIELD IN PARAMETER LIST.

The delete token (DEL_TKN) bit specifies whether the ROD token, if any, specified by the ROD CSCD descriptor should be deleted (see 5.16.6.7) when processing of the EXTENDED COPY command is complete as shown in table 147.

Table 147 — DEL_TKN bit processing

DEL_TKN bit	ROD TYPE field	ROD token created by	Description
0	n/a	n/a	Processing of the command shall not cause the ROD token, if any, to be deleted.
1	zero	processing copy manager or a copy manager in the same SCSI target device as the copy manager that created the ROD token	The copy manager should delete the ROD token after processing of the command has been completed.
		a copy manager in different SCSI target device from the copy manager that created the ROD token	The copy manager may communicate with the copy manager that created the ROD token to cause the ROD token to be deleted after processing of the command has been completed.
	not zero	n/a	The copy manager shall terminate the copy operation (see 5.16.4.3) with CHECK CONDITION status, with the sense key set to ILLEGAL REQUEST, and the additional sense code set to INVALID FIELD IN PARAMETER LIST.

If the ROD TYPE field is set to zero, the value in the ROD TOKEN OFFSET field is added to the location of the first byte of inline data in the EXTENDED COPY parameter list (see 5.16.7.1) to locate the ROD token (see 5.16.6) to be accessed via this ROD CSCD descriptor. If the ROD TYPE field is not set to zero and the ROD TOKEN OFFSET field is not set to zero, then the copy manager shall terminate the copy operation (see 5.16.4.3) with CHECK CONDITION status, with the sense key set to ILLEGAL REQUEST, and the additional sense code set to INVALID FIELD IN PARAMETER LIST.

6.4.6 Segment descriptors

6.4.6.1 Segment descriptors introduction

The descriptor type code (see table 132) values assigned to segment descriptors are shown in table 148.

Table 148 — EXTENDED COPY segment descriptor type codes (part 1 of 2)

Descriptor type code [a]	Reference	Description [b]	Name [b]
00h	6.4.6.2	Copy from block device to stream device	block→stream
01h	6.4.6.3	Copy from stream device to block device	stream→block
02h	6.4.6.4	Copy from block device to block device	block→block
03h	6.4.6.5	Copy from stream device to stream device	stream→stream
04h	6.4.6.6	Copy inline data to stream device	inline→stream
05h	6.4.6.7	Copy embedded data to stream device	embedded→stream
06h	6.4.6.8	Read from stream device and discard	stream→discard
07h	6.4.6.9	Verify CSCD	
08h	6.4.6.10	Copy block device with offset to stream device	block<o>→stream
09h	6.4.6.11	Copy stream device to block device with offset	stream→block<o>
0Ah	6.4.6.12	Copy block device with offset to block device with offset	block<o>→block<o>
0Bh	6.4.6.2	Copy from block device to stream device and hold a copy of processed data for the application client [c]	block→stream& application client
0Ch	6.4.6.3	Copy from stream device to block device and hold a copy of processed data for the application client [c]	stream→block& application client
0Dh	6.4.6.4	Copy from block device to block device and hold a copy of processed data for the application client [c]	block→block& application client
0Eh	6.4.6.5	Copy from stream device to stream device and hold a copy of processed data for the application client [c]	stream→stream& application client
0Fh	6.4.6.8	Read from stream device and hold a copy of processed data for the application client [c]	stream→discard& application client
10h	6.4.6.13	Write filemarks to sequential-access device	filemark→tape

[a] A copy manager may not support all segment descriptor types. However, the copy manager shall list all supported segment descriptor types in the Third-party Copy VPD page Supported Descriptors descriptor (see 7.8.17.6), and in the parameter data for the RECEIVE COPY OPERATING PARAMETERS command (see 6.22).

[b] Block devices are those with peripheral device type codes 00h (i.e, direct access block), 05h (i.e., CD/DVD), and 0Eh (i.e., simplified direct-access). Stream devices are those devices with peripheral device type codes 01h (i.e., sequential-access) and 03h (i.e., processor). Sequential-access stream (i.e., tape) devices are those with peripheral device type code 01h. See 6.6.2 for peripheral device type code definitions.

[c] The application client uses the RECEIVE COPY DATA(LID4) command (see 6.20) or RECEIVE COPY DATA(LID1) command (see 6.21) to retrieve data held for it by the copy manager (see 5.16.4.5).

Table 148 — EXTENDED COPY segment descriptor type codes (part 2 of 2)

Descriptor type code [a]	Reference	Description [b]	Name [b]
11h	6.4.6.14	Space records or filemarks on sequential-access device	space→tape
12h	6.4.6.15	Locate on sequential-access device	locate→tape
13h	6.4.6.16	Tape device image copy	<i>tape→<i>tape
14h	6.4.6.17	Register persistent reservation key	
15h	6.4.6.18	Third party persistent reservations source I_T nexus	
16h	6.4.6.19	Block device image copy	<i>block→<i>block
17h to BDh		Reserved	
BEh	6.4.6.20	Populate ROD from one or more block ranges	ROD←block ranges(n)
BFh	6.4.6.21	Populate ROD from one block range	ROD←block range(1)

[a] A copy manager may not support all segment descriptor types. However, the copy manager shall list all supported segment descriptor types in the Third-party Copy VPD page Supported Descriptors descriptor (see 7.8.17.6), and in the parameter data for the RECEIVE COPY OPERATING PARAMETERS command (see 6.22).

[b] Block devices are those with peripheral device type codes 00h (i.e, direct access block), 05h (i.e., CD/DVD), and 0Eh (i.e., simplified direct-access). Stream devices are those devices with peripheral device type codes 01h (i.e., sequential-access) and 03h (i.e., processor). Sequential-access stream (i.e., tape) devices are those with peripheral device type code 01h. See 6.6.2 for peripheral device type code definitions.

[c] The application client uses the RECEIVE COPY DATA(LID4) command (see 6.20) or RECEIVE COPY DATA(LID1) command (see 6.21) to retrieve data held for it by the copy manager (see 5.16.4.5).

Segment descriptors (see table 149) begin with an eight byte header.

Table 149 — Segment descriptor header

Bit / Byte	7	6	5	4	3	2	1	0
0	DESCRIPTOR TYPE CODE (00h to 3Fh)							
1	Reserved						DC	CAT
2	(MSB)			DESCRIPTOR LENGTH (n-3)				
3								(LSB)
4	(MSB)			SOURCE CSCD DESCRIPTOR ID				
5								(LSB)
6	(MSB)			DESTINATION CSCD DESCRIPTOR ID				
7								(LSB)
8	Segment descriptor parameters							
•••								
n								

The DESCRIPTOR TYPE CODE field is described in 6.4.4. If a copy manager receives an unsupported descriptor type code in a segment descriptor, the copy manager shall terminate the copy operation (see 5.16.4.3) with CHECK CONDITION status, with the sense key set to ILLEGAL REQUEST, and the additional sense code 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 (see 6.4.6.4). The DC bit shall be ignored for all other segment descriptors.

The CAT bit is described in 5.16.7.2.

The DESCRIPTOR LENGTH field is set to the length in bytes of the fields that follow the DESCRIPTOR LENGTH field in the segment descriptor.

The SOURCE CSCD DESCRIPTOR ID field (see table 150) specifies the copy source.

The DESTINATION CSCD DESCRIPTOR ID field (see table 150) specifies the copy destination.

Table 150 — CSCD descriptor ID values

Code	Description
0000h [a]	The copy source or copy destination is specified by the contents of the CSCD descriptor whose location in the EXTENDED COPY command parameter list (see 5.16.7.1) is computed as follows: 16 + (code <Σψμβολ>× 32) where code is 0000h to 07FFh as shown in this table
0001h to 07FFh [b]	
C000h	The copy source or copy destination is a null logical unit [b] whose peripheral device type is 00h (i.e., block)
C001h	The copy source or copy destination is a null logical unit [b] whose peripheral device type is 01h (i.e., stream)
F800h [c]	The copy source or copy destination is the logical unit specified by the ROD token specified in the ROD CSCD descriptor (see 6.4.5.9) that has this value in its ROD PRODUCER CSCD DESCRIPTOR ID field.
FFFFh	The copy source or copy destination is the logical unit that contains the copy manager (see SAM-5) that is processing the EXTENDED COPY command (i.e., the logical unit to which the EXTENDED COPY command was sent)
all others	Reserved

[a] If a SOURCE CSCD DESCRIPTOR ID field or a DESTINATION CSCD DESCRIPTOR ID field is set to this value, the copy manager may terminate the copy operation (see 5.16.4.3) with CHECK CONDITION status (see 6.4.2).

[b] A null logical unit is a logical unit that has a specified peripheral device type to which the copy manager is not allowed to send any SCSI commands. If the processing of a segment descriptor requires sending a SCSI command to a source device or destination device specified to be a null logical unit, then the copy manager shall terminate the copy operation (see 5.16.4.3) as if an unreachable CSCD had been encountered (see 5.16.7.4). Null logical units are useful for processing the residual data from previous segment descriptors without affecting any media (e.g., a segment descriptor of type 06h stream device to discard with the SOURCE CSCD DESCRIPTOR ID field set to C002h, the BYTE COUNT field set to zero, the CAT bit set to zero, and the PAD bit set to one may be used to discard all residual data).

[c] If this code appears in any field other than the ROD PRODUCER CSCD DESCRIPTOR ID field in the ROD CSCD descriptor, then the copy manager shall terminate the copy operation (see 5.16.4.3) as if an unreachable CSCD had been encountered (see 5.16.7.4).

If a segment descriptor format does not require a SOURCE CSCD DESCRIPTOR ID field or a DESTINATION CSCD DESCRIPTOR ID field, then that field is reserved.

If the CSCD specified by a SOURCE CSCD DESCRIPTOR ID field or a DESTINATION CSCD DESCRIPTOR ID field is not accessible to the copy manager, then the copy manager shall terminate the copy operation (see 5.16.4.3) with CHECK CONDITION status, with the sense key set to COPY ABORTED, and the additional sense code set to UNREACHABLE COPY TARGET.

6.4.6.2 Block device to stream device functions

The segment descriptor format shown in table 151 is used by the copy functions that move data from a block device to a stream device.

Table 151 — Block device to stream device segment descriptor

Bit / Byte	7	6	5	4	3	2	1	0
0	DESCRIPTOR TYPE CODE (00h or 0Bh)							
1	Reserved							CAT
2	(MSB)			DESCRIPTOR LENGTH (0014h)				
3								(LSB)
4	(MSB)			SOURCE CSCD DESCRIPTOR ID				
5								(LSB)
6	(MSB)			DESTINATION CSCD DESCRIPTOR ID				
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 6.4.4 and 6.4.6.1.

For descriptor type code 00h (i.e., block→stream) or descriptor type code 0Bh (i.e., block→stream&application client), the copy manager shall copy the data from the copy source block device specified by the SOURCE CSCD DESCRIPTOR ID field to the copy destination stream device specified by the DESTINATION CSCD DESCRIPTOR ID field using the logical blocks starting at the location specified by the BLOCK DEVICE LOGICAL BLOCK ADDRESS field. As many logical blocks shall be read as necessary to process (see 5.16.7.2) a number of bytes equal to the contents of the DISK BLOCK LENGTH field in the CSCD 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 (i.e., 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 copy operation (see 5.16.4.3) originated by the EXTENDED COPY command as described in 5.16.4.5.

The CAT bit is described in 5.16.7.2.

The DESCRIPTOR LENGTH field is described in 6.4.6.1, and shall be set as shown in table 151 for descriptor type code 00h (i.e., block→stream) and descriptor type code 0Bh (i.e., block→stream&application client).

The SOURCE CSCD DESCRIPTOR ID field and DESTINATION CSCD DESCRIPTOR ID field are described in 6.4.6.1.

The STREAM DEVICE TRANSFER LENGTH field specifies the amount of data to be written by each write command sent to the copy destination stream device. See 6.4.5.4 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 CSCD descriptor parameters for the copy destination sequential-access device type.

The BLOCK DEVICE NUMBER OF BLOCKS field specifies the length, in copy source logical blocks, of data to be processed (see 5.16.7.2) in the segment. A value of zero shall not be considered 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 copy source block device for this segment.

6.4.6.3 Stream device to block device functions

The segment descriptor format shown in table 152 is used by the copy functions that move data from a stream device to a block device.

Table 152 — Stream device to block device segment descriptor

Bit / Byte	7	6	5	4	3	2	1	0
0	DESCRIPTOR TYPE CODE (01h or 0Ch)							
1	Reserved							CAT
2	(MSB)			DESCRIPTOR LENGTH (0014h)				
3								(LSB)
4	(MSB)			SOURCE CSCD DESCRIPTOR ID				
5								(LSB)
6	(MSB)			DESTINATION CSCD DESCRIPTOR ID				
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 6.4.4 and 6.4.6.1.

For descriptor type code 01h (i.e., stream→block) or descriptor type code 0Ch (i.e., stream→block&application client), the copy manager shall copy the data from the copy source stream device specified by the SOURCE CSCD DESCRIPTOR ID field to the copy destination block device specified by the DESTINATION CSCD DESCRIPTOR ID 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 specified by the BLOCK DEVICE LOGICAL BLOCK ADDRESS field and continuing for the number of logical blocks specified in the BLOCK DEVICE NUMBER OF BLOCKS field.

For descriptor type code 0Ch (i.e., 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 copy operation (see 5.16.4.3) originated by the EXTENDED COPY command as described in 5.16.4.5.

The CAT bit is described in 5.16.7.2.

The DESCRIPTOR LENGTH field is described in 6.4.6.1, and shall be set as shown in table 152 for descriptor type code 01h (i.e., stream→block) and descriptor type code 0Ch (i.e., stream→block&application client).

The SOURCE CSCD DESCRIPTOR ID field and DESTINATION CSCD DESCRIPTOR ID field are described in 6.4.6.1.

The STREAM DEVICE TRANSFER LENGTH field specifies the amount of data to be read from the copy source stream device by each read command. See 6.4.5.4 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 CSCD descriptor parameters for the copy source sequential-access device type.

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

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

6.4.6.4 Block device to block device functions

The segment descriptor format shown in table 153 is used by the copy functions that move data from a block device to a block device.

Table 153 — 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 CSCD DESCRIPTOR ID				
5								(LSB)
6	(MSB)			DESTINATION CSCD DESCRIPTOR ID				
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 6.4.4 and 6.4.6.1.

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

If the destination count (DC) bit is set to zero, then:

a) as many logical blocks shall be read as necessary to process (see 5.16.7.2) a number of bytes equal to the contents of the DISK BLOCK LENGTH field in the CSCD descriptor for the copy source device times the contents of the BLOCK DEVICE NUMBER OF BLOCKS field; and
b) 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 is set to one, then:

a) the number of logical blocks specified by the BLOCK DEVICE NUMBER OF BLOCKS field shall be written to the copy destination block device;
b) as many bytes shall be processed (see 5.16.7.2) as necessary for these writes to be performed; and
c) as many logical blocks shall be read as necessary to supply the data to be processed.

For descriptor type code 0Dh (i.e., 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 copy operation (see 5.16.4.3) originated by the EXTENDED COPY command as described in 5.16.4.5.

The CAT bit is described in 5.16.7.2.

The DC bit specifies whether the BLOCK DEVICE NUMBER OF BLOCKS field refers to the copy source or copy destination. A DC bit set to zero specifies that the BLOCK DEVICE NUMBER OF BLOCKS field refers to the copy source. A DC bit set to one specifies that the BLOCK DEVICE NUMBER OF BLOCKS field refers to the copy destination.

The DESCRIPTOR LENGTH field shall is described in 6.4.6.16.3.7.1, and shall be set as shown in table 153 for descriptor type code 02h (i.e., block→block) and descriptor type code 0Dh (i.e., block→block&application client).

The SOURCE CSCD DESCRIPTOR ID field and DESTINATION CSCD DESCRIPTOR ID field are described in 6.4.6.1.

If the DC bit is set to zero, the BLOCK DEVICE NUMBER OF BLOCKS field specifies the number of logical blocks to be processed. If the DC bit is set to one, the BLOCK DEVICE NUMBER OF BLOCKS field specifies the number of logical blocks to be written to the copy destination.

If the DC bit is set to zero, a BLOCK DEVICE NUMBER OF BLOCKS field set to zero specifies that:

a) no source logical blocks shall be read and no source data shall be processed;
b) any residual destination data from a previous segment shall be written if possible to the destination in whole logical block transfers; and
c) any residual data shall be processed as described in 5.16.7.2.

If the DC bit set to one, a BLOCK DEVICE NUMBER OF BLOCKS field set to zero specifies that:

a) no destination logical blocks shall be written; and
b) the only processing to be performed is that any residual source data or destination data from the previous segment shall be processed as residual data as described in 5.16.7.2.

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

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

6.4.6.5 Stream device to stream device functions

The segment descriptor format shown in table 154 is used by the copy functions that move data from a stream device to a stream device.

Table 154 — 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)			DESCRIPTOR LENGTH (0010h)				
3								(LSB)
4	(MSB)			SOURCE CSCD DESCRIPTOR ID				
5								(LSB)
6	(MSB)			DESTINATION CSCD DESCRIPTOR ID				
7								(LSB)
8	Reserved							
9	(MSB)			SOURCE STREAM DEVICE TRANSFER LENGTH				
10								
11								(LSB)
12	Reserved							
13	(MSB)			DESTINATION STREAM DEVICE TRANSFER LENGTH				
14								
15								(LSB)
16	(MSB)			BYTE COUNT				
•••								
19								(LSB)

The DESCRIPTOR TYPE CODE field is described in 6.4.4 and 6.4.6.1.

For descriptor type code 03h (i.e., stream→stream) or descriptor type code 0Eh (i.e., stream→stream&application client), the copy manager shall copy the data from the copy source stream device specified by the SOURCE CSCD DESCRIPTOR ID field to the destination copy stream device specified by the DESTINATION CSCD DESCRIPTOR ID field. Data shall be read from the copy source stream device starting at the current position of the copy source stream device. Data shall be written to the copy destination stream device starting at the current position of the copy destination stream device. The BYTE COUNT field defines the number of bytes to be processed (see 5.16.7.2) by the copy manager. The copy manager shall perform reads as necessary to supply the source data, and as many writes as possible using the destination data.

For descriptor type code 0Eh (i.e., 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 copy operation (see 5.16.4.3) originated by the EXTENDED COPY command as described in 5.16.4.5.

The CAT bit is described in 5.16.7.2.

The DESCRIPTOR LENGTH field is described in 6.4.6.1, and shall be set as shown in table 154 for descriptor type code 03h (i.e., stream→stream) and descriptor type code 0Eh (i.e., stream→stream&application client).

The SOURCE CSCD DESCRIPTOR ID field and DESTINATION CSCD DESCRIPTOR ID field are described in 6.4.6.1.

The SOURCE STREAM DEVICE TRANSFER LENGTH field specifies the amount of data to be read from the copy source stream device by each read command. See 6.4.5.4 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 CSCD descriptor parameters for the copy source sequential-access device type.

The DESTINATION STREAM DEVICE TRANSFER LENGTH field specifies the amount of data to be written to the copy destination stream device by each write command. See 6.4.5.4 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 CSCD descriptor parameters for the copy destination sequential-access device type.

The BYTE COUNT field specifies the number of bytes that shall be processed (see 5.16.7.2) for this segment descriptor. A value of zero shall not be considered an error, and specifies that no source data shall be read and no source data shall be processed. However, a value of zero specifies that any residual destination data from a previous segment shall be written if possible to the copy destination in whole-block transfers, and any residual data shall be processed as described in 5.16.7.2.

6.4.6.6 Inline data to stream device function

The segment descriptor format shown in table 155 requests the copy manager to write inline data from the EXTENDED COPY parameter list to a stream device.

Table 155 — 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 CSCD DESCRIPTOR ID				
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 6.4.4 and 6.4.6.1, and shall be set as shown in table 155 for the inline data to stream device segment descriptor.

Descriptor type code 04h (i.e., inline→stream) requests the copy manager to write inline data to a copy destination stream device. The inline data shall be read from the inline data in the EXTENDED COPY parameter list (see 5.16.7.1). The data shall be written to the copy destination stream device specified by the DESTINATION CSCD DESCRIPTOR ID 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 5.16.7.2), and shall be processed as residual source data.

The CAT bit is described in 5.16.7.2.

The DESCRIPTOR LENGTH field is described in 6.4.6.1, and shall be set as shown in table 155 for descriptor type code 04h (i.e., inline→stream).

The DESTINATION CSCD DESCRIPTOR ID field is described in 6.4.6.1.

The STREAM DEVICE TRANSFER LENGTH field specifies the amount of data to be written to the copy destination stream device by each write command. See 6.4.5.4 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 CSCD descriptor parameters for the copy 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 5.16.7.1) to locate the first byte of inline data to be written to the copy destination 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 copy destination 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 6.4.3.6), then the copy manager shall terminate the copy operation (see 5.16.4.3) with CHECK CONDITION status, with the sense key set to COPY ABORTED, and the additional sense code set to INLINE DATA LENGTH EXCEEDED.

6.4.6.7 Embedded data to stream device function

The segment descriptor format shown in table 156 requests the copy manager to write embedded data from the segment descriptor to a stream device.

Table 156 — 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-3)				
3								(LSB)
4	Reserved							
5	Reserved							
6	(MSB)			DESTINATION CSCD DESCRIPTOR ID				
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 6.4.4 and 6.4.6.1, and shall be set as shown in table 156 for the embedded data to stream device segment descriptor.

Descriptor type code 05h (i.e., embedded→stream) requests the copy manager to write embedded data from the segment descriptor to a copy destination stream device. The embedded data shall be read from the segment descriptor. The data shall be written to the copy destination stream device specified by the DESTINATION CSCD DESCRIPTOR ID field starting at the current position of the copy destination 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 5.16.7.2), and shall be processed as residual source data.

The CAT bit is described in 5.16.7.2.

The DESCRIPTOR LENGTH field is described in 6.4.6.1. The value in the DESCRIPTOR LENGTH field shall be a multiple of 4.

The DESTINATION CSCD DESCRIPTOR ID field is described in 6.4.6.1.

The STREAM DEVICE TRANSFER LENGTH field specifies the amount of data to be written to the copy destination stream device by each write command. See 6.4.5.4 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 CSCD descriptor parameters for the copy 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 copy destination stream device. A value of zero shall not be considered an error. If the value in the EMBEDDED DATA NUMBER OF BYTES field is greater than the value in the DESCRIPTOR LENGTH field minus 12, then the copy manager shall terminate the copy operation (see 5.16.4.3) with CHECK CONDITION status, with the sense key set to ILLEGAL REQUEST, and the additional sense code set to INVALID FIELD IN PARAMETER LIST.

The EMBEDDED DATA field is a zero-padded (see 4.3.2) field whose length is a multiple of 4 that specifies the embedded data to be copied to the copy destination stream device.

6.4.6.8 Stream device to discard functions

The segment descriptor format shown in table 157 requests the copy manager to read data from a stream device and not transfer it to any copy destination.

Table 157 — 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 CSCD DESCRIPTOR ID				
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 6.4.4 and 6.4.6.1.

For descriptor type code 06h (i.e., stream→discard) or descriptor type code 0Fh (i.e., stream→discard&application client), the copy manager shall read data as necessary from the copy source stream device specified by the SOURCE CSCD DESCRIPTOR ID field starting at the current position of the copy source stream device. The number of bytes specified 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 (i.e., stream→discard) the removed data shall be discarded and not written to any copy destination.

For descriptor type code 0Fh (i.e., stream→discard&application client) the removed data shall be held for delivery to the application client upon completion of the copy operation (see 5.16.4.3) originated by the EXTENDED COPY command as described in 5.16.4.5.

The CAT bit is described in 5.16.7.2.

The DESCRIPTOR LENGTH field is described in 6.4.6.1, shall be set as shown in table 157 for descriptor type code 0Fh (i.e., stream→discard&application client) and descriptor type code 0Fh (i.e., stream→discard&application client).

The DESTINATION CSCD DESCRIPTOR ID field is described in 6.4.6.1.

The SOURCE STREAM DEVICE TRANSFER LENGTH field specifies the amount of data to be read from the copy source stream device on each read command. See 6.4.5.4 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 CSCD descriptor parameters for the copy source sequential-access device type.

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

6.4.6.9 Verify CSCD function

The segment descriptor format shown in table 158 requests the copy manager to verify the accessibility of a CSCD.

Table 158 — Verify CSCD 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 CSCD DESCRIPTOR ID				
5								(LSB)
6				Reserved				
7								
8	Reserved							TUR
9	Reserved							
•••								
11								

The DESCRIPTOR TYPE CODE field is described in 6.4.4 and 6.4.6.1, and shall be set as shown in table 158 for the verify CSCD segment descriptor.

Descriptor type code 07h requests the copy manager to verify the accessibility of the CSCD specified by the SOURCE CSCD DESCRIPTOR ID field.

The DESCRIPTOR LENGTH field is described in 6.4.6.1, and shall be set as shown in table 158 for the verify CSCD segment descriptor.

The SOURCE CSCD DESCRIPTOR ID field is described in 6.4.6.1.

Support for a value of one in the test unit ready (TUR) bit is optional. If setting the TUR bit to one is supported and the TUR bit is set to one, then a TEST UNIT READY command (see 6.47) shall be used to determine the readiness of the CSCD. If setting the TUR to one is not supported and the TUR bit is set to one, the copy manager shall terminate the copy operation (see 5.16.4.3) with CHECK CONDITION status, with the sense key set to COPY ABORTED, and the additional sense code set to INVALID FIELD IN PARAMETER LIST. The sense key specific information shall be set as described in 5.16.7.4. If the TUR bit is set to zero, then the accessibility should be verified without disturbing established unit attention conditions or ACA conditions (e.g., using the INQUIRY command (see 6.6)).

If the SOURCE CSCD DESCRIPTOR ID field specifies a ROD CSCD descriptor (see 6.4.5.9), the copy manager shall ignore the TUR bit and shall process the Verify CSCD segment descriptor based on the contents of ROD TYPE field as follows:

a) if the ROD TYPE field is set to zero, the copy manager shall verify the accessibility as follows:
   A) if the ROD token was created by the copy manager that is processing the Verify CSCD segment descriptor, then the ROD token shall be validated as described in 5.16.6.5.2;
   B) if the ROD token was created by a copy manager in the same SCSI target device as the copy manager that is processing the Verify CSCD segment descriptor, then the creating copy manager shall be requested to validate the ROD token as described in 5.16.6.5.2; and
   C) if the ROD token was created by a copy manager in a SCSI target device other than the SCSI target device that contains the copy manager that is processing the Verify CSCD segment descriptor, then the validation of the ROD token depends on the contents of the REMOTE TOKENS field in the ROD Features third-party copy descriptor (see 7.8.17.8) as follows:
      a) if the REMOTE TOKENS field is set to 0h, the copy manager shall terminate the copy operation (see 5.16.4.3) with CHECK CONDITION status with the sense key set to ILLEGAL REQUEST and the additional sense code set to INVALID TOKEN OPERATION, REMOTE TOKEN USAGE NOT SUPPORTED; or
      b) if the REMOTE TOKENS field is not set to 0h, the processing copy manager shall request the copy manager that created the ROD token to validate it;

   or

b) if the ROD TYPE field is not set to zero, the processing copy manager shall ignore the Verify CSCD segment descriptor. This shall not be considered an error.

6.4.6.10 Block device with offset to stream device function

The segment descriptor format shown in table 159 requests the copy manager to move data from a block device with a byte offset to a stream device.

Table 159 — Block device with offset to stream device segment descriptor

Bit / Byte	7	6	5	4	3	2	1	0
0	DESCRIPTOR TYPE CODE (08h)							
1	Reserved							CAT
2	(MSB)			DESCRIPTOR LENGTH (0018h)				
3								(LSB)
4	(MSB)			SOURCE CSCD DESCRIPTOR ID				
5								(LSB)
6	(MSB)			DESTINATION CSCD DESCRIPTOR ID				
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 6.4.4 and 6.4.6.1, and shall be set as shown in table 159 for the block device with offset to stream device segment descriptor.

Descriptor type code 08h (i.e., block<o>→stream) requests the copy manager to copy the data from the copy source block device specified by the SOURCE CSCD DESCRIPTOR ID field to the copy destination stream device specified by the DESTINATION CSCD DESCRIPTOR ID field using data starting at the location specified by the BLOCK DEVICE BYTE OFFSET field in the logical block specified 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 copy destination stream device starting at the current position of the media.

The CAT bit is described in 5.16.7.2.

The DESCRIPTOR LENGTH field is described in 6.4.6.1, and shall be set as shown in table 159 for the block device with offset to stream device segment descriptor.

The SOURCE CSCD DESCRIPTOR ID field and DESTINATION CSCD DESCRIPTOR ID field are described in 6.4.6.1.

The STREAM DEVICE TRANSFER LENGTH field specifies the amount of data to be written by each write command sent to the copy destination stream device. See 6.4.5.4 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 CSCD descriptor parameters for the copy destination sequential-access device type.

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

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

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

6.4.6.11 Stream device to block device with offset function

The segment descriptor format shown in table 160 requests the copy manager to move data from a stream device to a block device with a byte offset.

Table 160 — Stream device with offset to block device segment descriptor

Bit / Byte	7	6	5	4	3	2	1	0
0	DESCRIPTOR TYPE CODE (09h)							
1	Reserved							CAT
2	(MSB)			DESCRIPTOR LENGTH (0018h)				
3								(LSB)
4	(MSB)			SOURCE CSCD DESCRIPTOR ID				
5								(LSB)
6	(MSB)			DESTINATION CSCD DESCRIPTOR ID				
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 6.4.4 and 6.4.6.1, and shall be set as shown in table 160 for the stream device with offset to block device segment descriptor.

Descriptor type code 09h (i.e., stream→block<o>) requests the copy manager to copy the data from the copy source stream device specified by the SOURCE CSCD DESCRIPTOR ID field to the copy destination block device specified by the DESTINATION CSCD DESCRIPTOR ID field using the stream data starting at the current position of the copy source stream device. The data shall be written starting at the location specified by the BLOCK DEVICE BYTE OFFSET field in the logical block specified 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 copy destination block device before the starting offset shall be preserved. The content on the ending logical block on the copy destination block device beyond the end of the transfer shall be preserved. The copy manager may implement this function by reading the starting and ending logical blocks, modifying a portion of the logical blocks as required, and writing the full logical blocks to the copy destination block device.

The CAT bit is described in 5.16.7.2.

The DESCRIPTOR LENGTH field is described in 6.4.6.1, and shall be set as shown in table 160 for the stream device with offset to block device segment descriptor.

The SOURCE CSCD DESCRIPTOR ID field and DESTINATION CSCD DESCRIPTOR ID field are described in 6.4.6.1.

The STREAM DEVICE TRANSFER LENGTH field specifies the amount of data to be read from the copy source stream device by each read command. See 6.4.5.4 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 CSCD descriptor parameters for the copy source sequential-access device type.

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

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

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

6.4.6.12 Block device with offset to block device with offset function

The segment descriptor format shown in table 161 requests the copy manager to move data from a block device with a byte offset to a block device with a byte offset.

Table 161 — 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 CSCD DESCRIPTOR ID				
5								(LSB)
6	(MSB)			DESTINATION CSCD DESCRIPTOR ID				
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 6.4.4 and 6.4.6.1, and shall be set as shown in table 161 for the block device with offset to block device with offset segment descriptor.

Descriptor type code 0Ah (i.e., block<o>→block<o>) requests the copy manager to copy the data from the copy source block device specified by the SOURCE CSCD DESCRIPTOR ID field to the copy destination block device specified by the DESTINATION CSCD DESCRIPTOR ID field using data starting at the location specified by the source BLOCK DEVICE BYTE OFFSET field in the logical block specified 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 to the copy destination block device starting at the location specified by the DESTINATION BLOCK DEVICE BYTE OFFSET field in the logical block specified by the DESTINATION BLOCK DEVICE LOGICAL BLOCK ADDRESS field.

The content of the starting logical block on the copy destination block device before the starting offset shall be preserved. The content on the ending logical block on the copy destination block device 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 logical blocks as required, and writing the full logical blocks on the copy destination block device.

The CAT bit is described in 5.16.7.2.

The DESCRIPTOR LENGTH field is described in 6.4.6.1, and shall be set as shown in table 161 for the block device with offset to block device with offset segment descriptor.

The SOURCE CSCD DESCRIPTOR ID field and DESTINATION CSCD DESCRIPTOR ID field are described in 6.4.6.1.

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

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

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

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

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

6.4.6.13 Write filemarks function

The segment descriptor format shown in table 162 requests the copy manager to write filemarks on the destination tape device.

Table 162 — Write filemarks 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 CSCD DESCRIPTOR ID				
7								(LSB)
8	Reserved						Obsolete	W_IMMED
9	(MSB)			FILEMARK COUNT				
10								
11								(LSB)

The DESCRIPTOR TYPE CODE field is described in 6.4.4 and 6.4.6.1, and shall be set as shown in table 162 for the write filemarks segment descriptor.

Descriptor type code 10h (i.e., filemark→tape) requests the copy manager to write filemarks to the copy destination tape device specified by the DESTINATION CSCD DESCRIPTOR ID field starting at the current position of the copy destination tape device. If the PERIPHERAL DEVICE TYPE field in the CSCD descriptor specified by the DESTINATION CSCD DESCRIPTOR ID field does not contain 01h, then the copy manager shall terminate the copy operation (see 5.16.4.3) with CHECK CONDITION status, with the sense key set to COPY ABORTED, and the additional sense code set to INVALID OPERATION FOR COPY SOURCE OR DESTINATION.

The DESCRIPTOR LENGTH field is described in 6.4.6.1, and shall be set as shown in table 162 for the write filemarks segment descriptor.

The DESTINATION CSCD DESCRIPTOR ID field is described in 6.4.6.1.

If the write immediate (W_IMMED) bit in the segment descriptor is set to one, the copy manager shall send a WRITE FILEMARKS command to the copy destination tape device with the IMMED bit set to one. If the W_IMMED bit is set to zero, the copy manager shall send a WRITE FILEMARKS command to the copy destination tape device with the IMMED bit set to zero.

The FILEMARK COUNT field contents in the WRITE FILEMARKS command sent to the copy destination tape device shall be set to the value in the FILEMARK COUNT field in the segment descriptor.

6.4.6.14 Space function

The segment descriptor format shown in table 163 requests the copy manager to send a SPACE command (see SSC-4) to the destination tape device.

Table 163 — Space 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 CSCD DESCRIPTOR ID				
7								(LSB)
8	Reserved					CODE		
9	(MSB)			COUNT				
10								
11								(LSB)

The DESCRIPTOR TYPE CODE field is described in 6.4.4 and 6.4.6.1, and shall be set as shown in table 163 for the space segment descriptor.

Descriptor type code 11h (i.e., space→tape) requests the copy manager to send a SPACE command to the copy destination tape device specified by the DESTINATION CSCD DESCRIPTOR ID field. If the PERIPHERAL DEVICE TYPE field in the CSCD descriptor specified by the DESTINATION CSCD DESCRIPTOR ID field does not contain 01h, then the copy manager shall terminate the copy operation (see 5.16.4.3) with CHECK CONDITION status, with the sense key set to COPY ABORTED, and the additional sense code set to INVALID OPERATION FOR COPY SOURCE OR DESTINATION.

The DESCRIPTOR LENGTH field is described in 6.4.6.1, and shall be set as shown in table 163 for the space segment descriptor.

The DESTINATION CSCD DESCRIPTOR ID field is described in 6.4.6.1.

The CODE field and COUNT field contents in the SPACE command sent to the copy destination tape device shall be set to the values in the CODE field and COUNT field in the segment descriptor. The PARAMETER LENGTH

field, if any, shall be set to zero (i.e., implicit mode) or 16 (i.e., explicit mode). All other fields in the SPACE command sent to the copy destination tape device that affect the positioning of the tape shall be set to zero.

6.4.6.15 Locate function

The segment descriptor format shown in table 164 requests the copy manager to send a LOCATE command (see SSC-4) to the destination tape device.

Table 164 — Locate 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								
6	(MSB)			DESTINATION CSCD DESCRIPTOR ID				
7								(LSB)
8	(MSB)			LOGICAL OBJECT IDENTIFIER				
•••								
11								(LSB)

The DESCRIPTOR TYPE CODE field is described in 6.4.4 and 6.4.6.1, and shall be set as shown in table 164 for the locate segment descriptor.

Descriptor type code 12h (i.e., locate→tape) requests the copy manager to send a LOCATE command to the copy destination tape device specified by the DESTINATION CSCD DESCRIPTOR ID field. If the PERIPHERAL DEVICE TYPE field in the CSCD descriptor specified by the DESTINATION CSCD DESCRIPTOR ID field does not contain 01h, then the copy manager shall terminate the copy operation (see 5.16.4.3) with CHECK CONDITION status, with the sense key set to COPY ABORTED, and the additional sense code set to INVALID OPERATION FOR COPY SOURCE OR DESTINATION.

The DESCRIPTOR LENGTH field is described in 6.4.6.1, and shall be set as shown in table 164 for the locate segment descriptor.

The DESTINATION CSCD DESCRIPTOR ID field is described in 6.4.6.1.

The LOGICAL OBJECT IDENTIFIER field contents in the LOCATE command sent to the copy destination tape device shall be set to the value in the LOGICAL OBJECT IDENTIFIER field in the segment descriptor. All other fields in the LOCATE command sent to the copy destination tape device that affect the positioning of the tape shall be set to zero.

NOTE 26 - The restrictions described in this subclause for the LOCATE command limit the function to locating logical block identifiers in the current tape partition.

6.4.6.16 Tape device image copy function

The segment descriptor format shown in table 165 requests the copy manager to perform an image copy from the copy source tape device to the copy destination tape device.

Table 165 — 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 CSCD DESCRIPTOR ID				
5								(LSB)
6	(MSB)			DESTINATION CSCD DESCRIPTOR ID				
7								(LSB)
8	(MSB)			COUNT				
•••								
11								(LSB)

The DESCRIPTOR TYPE CODE field is described in 6.4.4 and 6.4.6.1 and shall be set as shown in table 165 for the tape device image copy segment descriptor.

Descriptor type code 13h (i.e., <i>tape→<i>tape) requests the copy manager to create a compatible image of the copy source specified by the SOURCE CSCD DESCRIPTOR ID field on the copy destination specified by the DESTINATION CSCD DESCRIPTOR ID field beginning at the current positions of the copy source and the copy destination. If the PERIPHERAL DEVICE TYPE field in the CSCD descriptor specified by the SOURCE CSCD DESCRIPTOR ID field or the DESTINATION CSCD DESCRIPTOR ID field does not contain 01h, then the copy manager shall terminate the copy operation (see 5.16.4.3) with CHECK CONDITION status, with the sense key set to COPY ABORTED, and the additional sense code set to INVALID OPERATION FOR COPY SOURCE OR DESTINATION.

The DESCRIPTOR LENGTH field is described in 6.4.6.1 and shall be set as shown in table 165 for descriptor type code 13h (i.e., <i>tape→<i>tape).

The SOURCE CSCD DESCRIPTOR ID field and DESTINATION CSCD DESCRIPTOR ID field are described in 6.4.6.1.

A COUNT field set to zero specifies that the tape image copy function shall not terminate due to any number of consecutive filemarks. Other error or exception conditions (e.g., early-warning, end-of-partition on the copy destination) may cause the copy manager to terminate the copy operation (see 5.16.4.3) prior to completion. If this occurs, the residue shall not be calculated and the INFORMATION field in the sense data shall be set to zero.

A COUNT field not set to zero specifies that the tape image copy function shall be terminated if the specified number of consecutive filemarks are copied.

The tape image copy operation terminates when:

a) the copy source encounters an end-of-partition as defined by the copy source;
b) the copy source encounters an end-of-data as defined by the copy source (i.e., BLANK CHECK sense key); or

c) the copy manager has copied the number of consecutive filemarks specified in the COUNT field from the copy source to the copy destination.

6.4.6.17 Register persistent reservation key function

The segment descriptor format shown in table 166 requests the copy manager to register an I_T nexus using the reservation key (see 5.12.7) specified by the RESERVATION KEY field with the logical unit specified by the DESTINATION CSCD DESCRIPTOR ID field.

Table 166 — Register persistent reservation 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 CSCD DESCRIPTOR ID				
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 6.4.4 and 6.4.6.1, and shall be set as shown in table 166 for the register persistent reservation key segment descriptor.

Descriptor type code 14h requests the copy manager to register an I_T nexus using the reservation key specified by the RESERVATION KEY field with the logical unit specified by the DESTINATION CSCD DESCRIPTOR ID field using a PERSISTENT RESERVE OUT command with a REGISTER service action (see 6.16.2).

The DESCRIPTOR LENGTH field is described in 6.4.6.1, and shall be set as shown in table 166 for the register persistent reservation key segment descriptor.

The DESTINATION CSCD DESCRIPTOR ID field is described in 6.4.6.1.

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

The application client sending an EXTENDED COPY command that contains a register persistent reservation key segment descriptor may be required to remove the reservation key held by the copy manager as described in 5.12.11 prior to sending the EXTENDED COPY command.

6.4.6.18 Third party persistent reservations source I_T nexus function

The segment descriptor format shown in table 167 requests the copy manager to send a PERSISTENT RESERVE OUT command with REGISTER AND MOVE service action (see 5.12.8) with the specified I_T nexus after all other segment descriptors have been processed. If an error is detected any time after receiving a third party persistent source reservation I_T nexus segment descriptor, then the PERSISTENT RESERVE OUT command REGISTER AND MOVE service action shall be processed before the copy operation (see 5.16.4.3) originated by the EXTENDED COPY command is completed.

This segment descriptor should be placed at or near the beginning of the list of segment descriptors to assure the copy manager processes the PERSISTENT RESERVE OUT command with REGISTER AND MOVE service action in the event of an error that terminates the processing of segment descriptors. If an error is detected in a segment descriptor and third party persistent reservations source I_T nexus segment descriptor has not been processed, then the copy manager shall not send a PERSISTENT RESERVE OUT command with REGISTER AND MOVE service action.

Placing more than one source third party persistent reservations source I_T nexus segment descriptor in the list of descriptors is not an error. All source third party persistent reservations source I_T nexus segment descriptors known to the copy manager shall be processed after all other segment descriptors have been processed.

Table 167 — Third party persistent reservations source I_T nexus segment descriptor

Bit Byte	7	6	5	4	3	2	1	0
0	DESCRIPTOR TYPE CODE (15h)							
1	Reserved							
2	(MSB)			DESCRIPTOR LENGTH (n-3)				
3								(LSB)
4	Reserved							
5	Reserved							
6	(MSB)			DESTINATION CSCD DESCRIPTOR ID				
7								(LSB)
8	(MSB)			RESERVATION KEY				
•••								
15								(LSB)
16	(MSB)			SERVICE ACTION RESERVATION KEY				
•••								
23								(LSB)
24	Reserved							
25	Reserved						UNREG	APTPL
26	(MSB)			RELATIVE TARGET PORT IDENTIFIER				
27								(LSB)
28	(MSB)			TRANSPORTID LENGTH (n-31)				
•••								
31								(LSB)
32	TransportID							
•••								
n								

The DESCRIPTOR TYPE CODE field is described in 6.4.4 and 6.4.6.1, and shall be set as shown in table 167 for the third party persistent reservations source I_T nexus segment descriptor.

Descriptor type code 15h requests the copy manager to send PERSISTENT RESERVE OUT command with REGISTER AND MOVE service action (see 6.16) to the target port specified by the DESTINATION CSCD DESCRIPTOR ID field.

The DESCRIPTOR LENGTH field is described in 6.4.6.1. The value in the DESCRIPTOR LENGTH field shall be a multiple of 4.

The DESTINATION CSCD DESCRIPTOR ID field is described in 6.4.6.1.

If the PERIPHERAL DEVICE TYPE field in the CSCD descriptor specified by the DESTINATION CSCD DESCRIPTOR ID field does not contain 01h, then the copy manager shall terminate the copy operation (see 5.16.4.3) with CHECK CONDITION status, with the sense key set to COPY ABORTED, and the additional sense code set to INVALID OPERATION FOR COPY SOURCE OR DESTINATION.

Bytes eight to n of the third party persistent reservations source I_T nexus segment descriptor shall be sent as the parameter list (see 6.16.4) for the PERSISTENT RESERVE OUT command with REGISTER AND MOVE service action.

For a description of the RESERVATION KEY field, SERVICE ACTION RESERVATION KEY field, UNREG bit, APTPL bit, RELATIVE TARGET PORT IDENTIFIER field, TRANSPORTID LENGTH field, and TransportID, see 6.16.4.

6.4.6.19 Block device image copy function

The segment descriptor format shown in table 168 requests the copy manager to perform an image copy from the copy source block device to the copy destination block device.

Table 168 — Block device image copy segment descriptor

Bit / Byte	7	6	5	4	3	2	1	0
0	DESCRIPTOR TYPE CODE (16h)							
1	Reserved							
2	(MSB)			DESCRIPTOR LENGTH (0018h)				
3								(LSB)
4	(MSB)			SOURCE CSCD DESCRIPTOR ID				
5								(LSB)
6	(MSB)			DESTINATION CSCD DESCRIPTOR ID				
7								(LSB)
8	(MSB)			STARTING SOURCE LOGICAL BLOCK ADDRESS				
•••								
15								(LSB)
16	(MSB)			STARTING DESTINATION LOGICAL BLOCK ADDRESS				
•••								
23								(LSB)
24	(MSB)			NUMBER OF LOGICAL BLOCKS				
•••								
27								(LSB)

The DESCRIPTOR TYPE CODE field is described in 6.4.4 and 6.4.6.1 and shall be set as shown in table 168 for the block device image copy segment descriptor.

Descriptor type code 16h (i.e., <i>block→<i>block) requests the copy manager to copy logical blocks from the copy source block device specified by the SOURCE CSCD DESCRIPTOR ID field to the copy destination block device specified by the DESTINATION CSCD DESCRIPTOR ID field while preserving the characteristics (e.g., logical block length protection information) associated with each logical block.

The copy manager shall terminate the copy operation (see 5.16.4.3) with CHECK CONDITION status, with the sense key set to COPY ABORTED, and the additional sense code set to INCORRECT COPY TARGET DEVICE TYPE if:

a) the DISK BLOCK LENGTH field in the device type specific CSCD descriptor parameters (see 6.4.5.3) for the copy source is not equal to DISK BLOCK LENGTH field in the device type specific CSCD descriptor parameters for the copy destination; or
b) the protection information characteristics for the copy source are not the same as the protection information characteristics for the copy destination.

While copying logical blocks from the copy source to the copy destination, the copy manager shall preserve the protection information.

While copying logical blocks from the copy source to the copy destination, the copy manager should preserve the logical block provisioning information, if any. If the copy manager detects differences between the logical block provisioning characteristics for the copy source and the logical block provisioning characteristics for the copy destination that prevent the preservation of logical block provisioning information (see SBC-3), then the copy manager may terminate the copy operation (see 5.16.4.3) with CHECK CONDITION status, with the sense key set to COPY ABORTED, and the additional sense code set to INCORRECT COPY TARGET DEVICE TYPE.

The DESCRIPTOR LENGTH field is described in 6.4.6.1, and shall be set as shown in table 168 for descriptor type code 16h (i.e., <i>block→<i>block).

The SOURCE CSCD DESCRIPTOR ID field and DESTINATION CSCD DESCRIPTOR ID field are described in 6.4.6.1.

The STARTING SOURCE LOGICAL BLOCK ADDRESS field specifies the first logical block to read from the copy source block device.

The STARTING DESTINATION LOGICAL BLOCK ADDRESS field specifies the first logical block to write on the copy destination block device.

The NUMBER OF LOGICAL BLOCKS field specifies the number of logical blocks to copy. If the NUMBER OF LOGICAL BLOCKS field is set to zero, the copy manager shall use FFFF FFFF FFFF FFFFh as the number of logical blocks to copy.

The copy manager shall copy logical blocks from the copy source to the copy destination beginning at the specified logical block addresses until:

a) the specified number of logical blocks are copied;
b) the maximum logical block address on the copy source is reached; or
c) the maximum logical block address on the copy destination is reached.

The <i>block→<i>block copy function shall be considered a success regardless of which limit on the number of logical blocks copied caused the function to end.

6.4.6.20 Populate a ROD from one or more block ranges function

The segment descriptor format shown in table 169 requests the copy manager to add one or more ranges from the copy source block device to the end of the ROD that is specified as the copy destination. The copy manager shall terminate the copy operation (see 5.16.4.3) with CHECK CONDITION status, with the sense key set to ILLEGAL REQUEST, and the additional sense code set to INVALID FIELD IN PARAMETER LIST if:

a) the PERIPHERAL DEVICE TYPE field in the specified copy source CSCD descriptor is not set to 00h (i.e., block device);
b) the PERIPHERAL DEVICE TYPE field in the specified copy destination CSCD descriptor is not set to 00h (i.e., block device);
c) the SOURCE CSCD DESCRIPTOR ID field is not set to FFFFh (i.e., the logical unit that contains the copy manager);
d) the copy destination CSCD descriptor is not a ROD CSCD descriptor (see 6.4.5.9);
e) the ROD TYPE field in the copy destination CSCD descriptor is set to zero; or
f) the same LBA is specified in more than one range descriptor (see 5.16.6.3).

Table 169 — Populate a ROD from one or more block ranges segment descriptor

Bit / Byte	7	6	5	4	3	2	1	0
0	DESCRIPTOR TYPE CODE (BEh)							
1	Reserved							
2	(MSB)			DESCRIPTOR LENGTH (n-3)				
3								(LSB)
4	(MSB)			SOURCE CSCD DESCRIPTOR ID				
5								(LSB)
6	(MSB)			DESTINATION CSCD DESCRIPTOR ID				
7								(LSB)
8	Reserved							
•••								
12								
13	RANGE DESCRIPTOR TYPE							
14	(MSB)			RANGE DESCRIPTORS LENGTH (n-15)				
15								(LSB)
	Range descriptors							
16	Range descriptor [first]							
•••								
	⋮							
	Range descriptor [last]							
•••								
n								

The DESCRIPTOR TYPE CODE field is described in 6.4.4 and 6.4.6.1 and shall be set as shown in table 169 for the Populate a ROD from one or more block ranges segment descriptor.

Descriptor type code BEh (i.e., ROD←block ranges(n)) requests the copy manager to add the ranges specified in the segment descriptor from the copy source specified by the SOURCE CSCD DESCRIPTOR ID field to the end of the ROD specified by the DESTINATION CSCD DESCRIPTOR ID field.

The DESCRIPTOR LENGTH field is described in 6.4.6.1.

The SOURCE CSCD DESCRIPTOR ID field and DESTINATION CSCD DESCRIPTOR ID field are described in 6.4.6.1.

The RANGE DESCRIPTOR TYPE field (see table 170) specifies the format of all range descriptors.

Table 170 — RANGE DESCRIPTOR TYPE field

Code	Description	Reference
01h	Four gibi-block range descriptor	table 171
all others	Reserved	

The RANGE DESCRIPTORS LENGTH field specifies the number of bytes of range descriptors that follow.

If the RANGE DESCRIPTOR TYPE field is set to 01h, each range descriptor (see table 171) has the format specified by the RANGE DESCRIPTOR TYPE field.

Table 171 — Populate a ROD four gibi-block range descriptor format

Bit / Byte	7	6	5	4	3	2	1	0
0	(MSB)							
•••				LOGICAL BLOCK ADDRESS				
7								(LSB)
8	(MSB)							
•••				NUMBER OF LOGICAL BLOCKS				
11								(LSB)
12								
•••				Reserved				
15								

The LOGICAL BLOCK ADDRESS field specifies the first LBA from the copy source to be added to the copy destination ROD.

The NUMBER OF LOGICAL BLOCKS field specifies the number of consecutive LBAs from the copy source to be added to the copy destination ROD.

6.4.6.21 Populate a ROD from one block range function

The segment descriptor format shown in table 172 requests the copy manager to add one range from the copy source block device to the end of the ROD that is specified as the copy destination. The copy manager shall terminate the copy operation (see 5.16.4.3) with CHECK CONDITION status, with the sense key set to ILLEGAL REQUEST, and the additional sense code set to INVALID FIELD IN PARAMETER LIST if:

a) the PERIPHERAL DEVICE TYPE field in the specified copy source CSCD descriptor is not set to 00h (i.e., block device);
b) the PERIPHERAL DEVICE TYPE field in the specified copy destination CSCD descriptor is not set to 00h (i.e., block device);
c) the SOURCE CSCD DESCRIPTOR ID field is not set to FFFFh (i.e., the logical unit that contains the copy manager);
d) the copy destination CSCD descriptor is not a ROD CSCD descriptor (see 6.4.5.9); or
e) the ROD TYPE field in the copy destination CSCD descriptor is set to zero.

Table 172 — Populate a ROD from one block range segment descriptor

Bit / Byte	7	6	5	4	3	2	1	0
0	DESCRIPTOR TYPE CODE (BFh)							
1	Reserved							
2	(MSB)			DESCRIPTOR LENGTH (0010h)				
3								(LSB)
4	(MSB)			SOURCE CSCD DESCRIPTOR ID				
5								(LSB)
6	(MSB)			DESTINATION CSCD DESCRIPTOR ID				
7								(LSB)
8	(MSB)			LOGICAL BLOCK ADDRESS				
•••								
15								(LSB)
16	(MSB)			NUMBER OF LOGICAL BLOCKS				
•••								
19								(LSB)

The DESCRIPTOR TYPE CODE field is described in 6.4.4 and 6.4.6.1 and shall be set as shown in table 172 for the Populate a ROD from one block range segment descriptor.

Descriptor type code BFh (i.e., ROD←block range(1)) requests the copy manager to add the range specified in the segment descriptor from the copy source specified by the SOURCE CSCD DESCRIPTOR ID field to the end of the ROD specified by the DESTINATION CSCD DESCRIPTOR ID field.

The DESCRIPTOR LENGTH field is described in 6.4.6.1, and shall be set as shown in table 172 for descriptor type code BEh (i.e., ROD←block range(1)).

The SOURCE CSCD DESCRIPTOR ID field and DESTINATION CSCD DESCRIPTOR ID field are described in 6.4.6.1.

The LOGICAL BLOCK ADDRESS field specifies the first LBA from the copy source to be added to the copy destination ROD.

The NUMBER OF LOGICAL BLOCKS field specifies the number of consecutive LBAs from the copy source to be added to the copy destination ROD.

6.5 EXTENDED COPY(LID1) command

The EXTENDED COPY(LID1) command (see table 173) is an SPC-3 compatible third-party copy command (see 5.16.3) that requests the copy manager to copy data from one set of copy sources (e.g., a set of source logical units) to a set of copy destinations (e.g., a set of destination logical units).

Table 173 — EXTENDED COPY(LID1) command

Bit / Byte	7	6	5	4	3	2	1	0
0	OPERATION CODE (83h)							
1	Reserved			SERVICE ACTION (00h)				
2	Reserved							
•••								
9								
10	(MSB)			PARAMETER LIST LENGTH				
•••								
13								(LSB)
14	Reserved							
15	CONTROL							

The OPERATION CODE field is defined in 4.2.5.1 and shall be set as shown in table 173 for the EXTENDED COPY(LID1) command.

The SERVICE ACTION field is defined in 4.2.5.2 and shall be set as shown in table 173 for the EXTENDED COPY(LID1) command.

The PARAMETER LIST LENGTH field is defined in 6.4.1.

The CONTROL byte is defined in SAM-5.

The format of the EXTENDED COPY(LID1) parameter list is shown in table 174.

Table 174 — EXTENDED COPY(LID1) parameter list

Bit / Byte	7	6	5	4	3	2	1	0
0	LIST IDENTIFIER							
1	Reserved		STR	LIST ID USAGE		PRIORITY		
2	(MSB)			CSCD DESCRIPTOR LIST LENGTH (n-15)				
3								(LSB)
4				Reserved				
•••								
7								
8	(MSB)			SEGMENT DESCRIPTOR LIST LENGTH (m-n)				
11								(LSB)
12	(MSB)			INLINE DATA LENGTH (k-m)				
15								(LSB)
	CSCD descriptor list							
16				CSCD descriptor (see 6.4.5) [ID 1]				
•••								
47								
	⋮							
n-31				CSCD descriptor (see 6.4.5) [ID x]				
•••								
n								
	Segment descriptor list							
n+1				Segment descriptor (see 6.4.6) [first]				
•••								
n+1+l								
	⋮							
				Segment descriptor (see 6.4.6) [last]				
•••								
m								
m+1				Inline data				
•••								
k								

The LIST IDENTIFIER field is defined in 6.4.3.2.

The STR bit is defined in 6.4.3.1.

The LIST ID USAGE field is defined in 6.4.3.2.

The PRIORITY field is defined in 6.4.3.3.

The CSCD DESCRIPTOR LIST LENGTH field is defined in 6.4.3.4.

The SEGMENT DESCRIPTOR LIST LENGTH field is defined in 6.4.3.5.

The INLINE DATA LENGTH field is defined in 6.4.3.6.

The CSCD descriptors are defined in 6.4.3.4 and 6.4.5.

The segment descriptors are defined in 6.4.3.5 and 6.4.6.

The inline data is defined in 6.4.3.6.

6.6 INQUIRY command

6.6.1 INQUIRY command introduction

The INQUIRY command (see table 175) requests the device server to return information regarding the logical unit and SCSI target device.

Table 175 — INQUIRY command

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

The OPERATION CODE field is defined in 4.2.5.1 and shall be set as shown in table 175 for the INQUIRY command.

An enable vital product data (EVPD) bit set to one specifies that the device server shall return the vital product data specified by the PAGE CODE field (see 7.8). If the device server does not implement the requested vital product data page, then the device server shall terminate the command with CHECK CONDITION status, with the sense key set to ILLEGAL REQUEST, and the additional sense code set to INVALID FIELD IN CDB.

An EVPD bit set to zero specifies that the device server shall return the standard INQUIRY data (see 6.6.2). If the PAGE CODE field is not set to zero and the EVPD bit is set to zero, then the device server shall terminate the command with CHECK CONDITION status, with the sense key set to ILLEGAL REQUEST, and the additional sense code set to INVALID FIELD IN CDB.

If the EVPD bit is set to one, the PAGE CODE field specifies which page of vital product data information the device server shall return (see 7.8).

The ALLOCATION LENGTH field is defined in 4.2.5.6.

The CONTROL byte is defined in SAM-5.

In response to an INQUIRY command received by an incorrect logical unit, the SCSI target device shall return the INQUIRY data with the peripheral qualifier set to the value defined in 6.6.2. The device server shall terminate an INQUIRY command with CHECK CONDITION status only if the device server is unable to return the requested INQUIRY data.

If an INQUIRY command is received from an initiator port for which the device server has a pending unit attention condition (i.e., before the device server reports CHECK CONDITION status), then the device server shall perform the INQUIRY command and shall not clear the unit attention condition (see SAM-5).

The device server should be able to process the INQUIRY command even when an error occurs that prohibits normal command completion.

INQUIRY data (i.e., standard INQUIRY data (see 6.6.2) and all VPD pages (see 7.8)) should be returned even though the device server is not ready for other commands. Standard INQUIRY data, the Extended INQUIRY Data VPD page (see 7.8.7), and the Device Identification VPD page (see 7.8.6) should be available without

incurring any media access delays. If reporting INQUIRY data requires a delay (e.g., the device server stores some of the standard INQUIRY data or VPD data on the media), then the device server may return ASCII spaces (20h) in ASCII fields and zeros in other fields until the data is available from the media.

INQUIRY data may change as the SCSI target device and its logical units perform their initialization sequence.

EXAMPLE – Logical units may provide a minimum command set from nonvolatile memory until they load the final firmware from the media. After the firmware has been loaded, more options may be supported and therefore different INQUIRY data may be returned.

If INQUIRY data changes for any reason, the device server shall establish a unit attention condition for the initiator port associated with every I_T nexus (see SAM-5), with the additional sense code set to INQUIRY DATA HAS CHANGED.

NOTE 27 - The INQUIRY command may be used by an application client after a hard reset or power on condition to determine the device types for system configuration.

6.6.2 Standard INQUIRY data

The standard INQUIRY data (see table 176) shall contain at least 36 bytes.

Table 176 — Standard INQUIRY data format (part 1 of 2)

Bit / Byte	7	6	5	4	3	2	1	0
0	PERIPHERAL QUALIFIER			PERIPHERAL DEVICE TYPE				
1	RMB	LU_CONG	Reserved					
2	VERSION							
3	Reserved	Reserved	NORMACA	HISUP	RESPONSE DATA FORMAT (2h)			
4	ADDITIONAL LENGTH (n-4)							
5	SCCS	ACC	TPGS		3PC	Reserved		PROTECT
6	Obsolete	ENCSERV	VS	MULTIP	Obsolete	Reserved	Reserved	ADDR16 [a]
7	Obsolete	Reserved	WBUS16 [a]	SYNC [a]	Obsolete	Reserved	CMDQUE	VS
8	(MSB)							
•••	T10 VENDOR IDENTIFICATION							
15								(LSB)
16	(MSB)							
•••	PRODUCT IDENTIFICATION							
31								(LSB)
32	(MSB)							
•••	PRODUCT REVISION LEVEL							
35								(LSB)
36	Vendor specific							
•••								
55								
56	Reserved				CLOCKING [a]		QAS [a]	IUS [a]
57	Reserved							

Table 176 — Standard INQUIRY data format (part 2 of 2)

Bit / Byte	7	6	5	4	3	2	1	0
58	(MSB)			VERSION DESCRIPTOR 1				
59								(LSB)
	⋮							
72	(MSB)			VERSION DESCRIPTOR 8				
73								(LSB)
74	Reserved							
•••								
95								
	Vendor specific parameters							
96	Vendor specific							
•••								
n								

[a] The meanings of these fields are specific to SPI-5 (see 6.6.3). For SCSI transport protocols other than the SCSI Parallel Interface, these fields are reserved.

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

The PERIPHERAL QUALIFIER field is defined in table 177.

Table 177 — PERIPHERAL QUALIFIER field

Qualifier	Description
000b	A peripheral device having the indicated peripheral device type is connected to this logical unit. If the device server is unable to determine whether or not a peripheral device is connected, then the device server also shall use this peripheral qualifier. This peripheral qualifier does not indicate that the peripheral device connected to the logical unit is ready for access.
001b	A peripheral device having the indicated peripheral device type is not connected to this logical unit. However, the device server is capable of supporting the indicated peripheral device type on this logical unit.
010b	Reserved
011b	The device server is not capable of supporting a peripheral device on this logical unit. For this peripheral qualifier the peripheral device type shall be set to 1Fh. All other peripheral device type values are reserved for this peripheral qualifier.
100b to 111b	Vendor specific

The peripheral device type is defined in table 178.

Table 178 — Peripheral device type

Code	Reference [a]	Description
00h	SBC-3	Direct access block device (e.g., magnetic disk)
01h	SSC-4	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-6	CD/DVD device
06h		Obsolete
07h	SBC	Optical memory device (e.g., some optical disks)
08h	SMC-3	Media changer device (e.g., jukeboxes)
09h		Obsolete
0Ah to 0Bh		Obsolete
0Ch	SCC-2	Storage array controller device (e.g., RAID)
0Dh	SES-2	Enclosure services device
0Eh	RBC	Simplified direct-access device (e.g., magnetic disk)
0Fh	OCRW	Optical card reader/writer device
10h		Reserved
11h	OSD-2	Object-based Storage Device
12h	ADC-3	Automation/Drive Interface
13h	clause 9	Security manager device
14h	ZBC	Host managed zoned block device
15h to 1Dh		Reserved
1Eh	clause 8	Well known logical unit [b]
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] All well known logical units use the same peripheral device type code.

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

A logical unit conglomerate (LU_CONG) bit set to zero indicates that the logical unit is not part of a logical unit conglomerate (see SAM-5). A LU_CONG bit set to one indicates that the logical unit is part of a logical unit conglomerate.

The VERSION field indicates the implemented version of this standard and is defined in table 179.

Table 179 — VERSION field

Code	Description
00h	The device server does not claim conformance to any standard.
01h to 02h	Obsolete
03h	The device server complies to INCITS 301-1997 (a withdrawn standard).
04h	The device server complies to INCITS 351-2001 (SPC-2).
05h	The device server complies to INCITS 408-2005 (SPC-3).
06h	The device server complies to this standard.
07h	Reserved
08h to 0Ch	Obsolete
0Dh to 3Fh	Reserved
40h to 44h	Obsolete
45h to 47h	Reserved
48h to 4Ch	Obsolete
4Dh to 7Fh	Reserved
80h to 84h	Obsolete
85h to 87h	Reserved
88h to 8Ch	Obsolete
8Dh to FFh	Reserved

The Normal ACA Supported (NORMACA) bit set to one indicates that the device server supports a NACA bit set to one in the CDB CONTROL byte and supports the ACA task attribute (see SAM-5). A NORMACA bit set to zero indicates that the device server does not support a NACA bit set to one and does not support the ACA task attribute.

A historical support (HISUP) bit set to zero indicates the SCSI target device does not use the LUN structures described in SAM-5. A HISUP bit set to one indicates the SCSI target device uses the LUN structures described in SAM-5.

The RESPONSE DATA FORMAT field (see table 180) indicates the format of the standard INQUIRY data and shall be set as shown in table 176.

Table 180 — RESPONSE DATA FORMAT field

Code	Description
00h to 01h	Obsolete
02h	The response data format complies to this standard.
all other values	Reserved

The ADDITIONAL LENGTH field indicates the length in bytes of the remaining standard INQUIRY data. The contents of the ADDITIONAL LENGTH field are not altered based on the allocation length (see 4.2.5.6).

An SCC Supported (SCCS) bit set to one indicates that the SCSI target device contains an embedded storage array controller component that is addressable through this logical unit. See SCC-2 for details about storage array controller devices. An SCCS bit set to zero indicates that no embedded storage array controller component is addressable through this logical unit.

An Access Controls Coordinator (ACC) bit set to one indicates that the SCSI target device contains an access controls coordinator that is addressable through this logical unit. An ACC bit set to zero indicates that no

access controls coordinator is addressable through this logical unit. If the SCSI target device contains an access controls coordinator that is addressable through any logical unit other than the ACCESS CONTROLS well known logical unit (see 8.3), then the ACC bit shall be set to one for LUN 0.

The contents of the target port group support (TPGS) field (see table 181) indicate the support for asymmetric logical unit access (see 5.15).

Table 181 — TPGS field

Code	Description
00b	The logical unit does not support asymmetric logical unit access or supports a form of asymmetric access that is vendor specific. Neither the REPORT TARGET GROUPS command nor the SET TARGET PORT GROUPS command is supported.
01b	The logical unit supports only implicit asymmetric logical unit access (see 5.15.2.8). The logical unit is capable of changing target port asymmetric access states without a SET TARGET PORT GROUPS command. The REPORT TARGET PORT GROUPS command is supported and the SET TARGET PORT GROUPS command is not supported.
10b	The logical unit supports only explicit asymmetric logical unit access (see 5.15.2.9). The logical unit only changes target port asymmetric access states as requested with the SET TARGET PORT GROUPS command. Both the REPORT TARGET PORT GROUPS command and the SET TARGET PORT GROUPS command are supported.
11b	The logical unit supports both explicit and implicit asymmetric logical unit access. Both the REPORT TARGET PORT GROUPS command and the SET TARGET PORT GROUPS commands are supported.

A third-party copy (3PC) bit set to one indicates that the logical unit supports third-party copy commands (see 5.16.3). A 3PC bit set to zero indicates that the logical unit does not support third-party copy commands.

A PROTECT bit set to zero indicates that the logical unit does not support protection information. A PROTECT bit set to one indicates that the logical unit supports:

a) type 1 protection, type 2 protection, or type 3 protection (see SBC-3); or
b) logical block protection (see SSC-4).

More information about the type of protection the logical unit supports is available in the SPT field (see 7.8.7).

An Enclosure Services (ENCSERV) bit set to one indicates that the SCSI target device contains an embedded enclosure services component that is addressable through this logical unit. See SES-3 for details about enclosure services. An ENCSERV bit set to zero indicates that no embedded enclosure services component is addressable through this logical unit.

A multiple SCSI port (MULTIP) bit set to one indicates that the logical unit is in a SCSI target device with multiple SCSI target ports (see SAM-5). A MULTIP bit set to zero indicates that the logical unit is in a SCSI target device with a single SCSI target port.

The CMDQUE bit shall be set to one indicating that the logical unit supports the command management model defined in SAM-5.

The T10 VENDOR IDENTIFICATION field contains eight bytes of left-aligned ASCII data (see 4.3.1) identifying the manufacturer of the logical unit. The T10 vendor identification shall be one assigned by INCITS. A list of assigned T10 vendor identifications is in Annex G and on the T10 web site (http://www.t10.org).

NOTE 28 - The T10 web site (http://www.t10.org) provides a convenient means to request an identification code.

The PRODUCT IDENTIFICATION field contains 16 bytes of left-aligned ASCII data (see 4.3.1) that identifies the product and is defined by the manufacturer.

The PRODUCT REVISION LEVEL field contains four bytes of left-aligned ASCII data that identifies the product revision and is defined by the manufacturer.

The VERSION DESCRIPTOR fields provide for identifying up to eight standards to which the SCSI target device and/or logical unit claim conformance. The value in each VERSION DESCRIPTOR field shall be selected from table 182. All version descriptor values not listed in table 182 are reserved. Technical Committee T10 of INCITS maintains an electronic copy of the information in table 182 on its world wide web site (http://www.t10.org/). In the event that the T10 world wide web site is no longer active, the information may be accessible on the INCITS world wide web site (http://www.incits.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 transport standard if any, followed by the SCSI transport protocol standard, followed by the appropriate SPC-x version, followed by the device type command set, followed by a secondary command set if any.

Table 182 — Version descriptor values (part 1 of 12)

Standard	Version Descriptor Value
ACS-2 (no version claimed)	1761h
ACS-2 INCITS 482-2013	1762h
ACS-3 (no version claimed)	1765h
ADC (no version claimed)	03C0h
ADC INCITS 403-2005	03D7h
ADC T10/1558-D revision 7	03D6h
ADC T10/1558-D revision 6	03D5h
ADC-2 (no version claimed)	04A0h
ADC-2 INCITS 441-2008	04ACh
ADC-2 T10/1741-D revision 7	04A7h
ADC-2 T10/1741-D revision 8	04AAh
ADC-3 (no version claimed)	0500h
ADC-3 INCITS 497-2012	050Ah
ADC-3 T10/1895-D revision 04	0502h
ADC-3 T10/1895-D revision 05	0504h
ADC-3 T10/1895-D revision 05a	0506h
ADC-4 (no version claimed)	0640h
ADP (no version claimed)	09C0h
ADT (no version claimed)	09E0h
ADT INCITS 406-2005	09FDh
ADT T10/1557-D revision 14	09FAh
ADT T10/1557-D revision 11	09F9h
ADT-2 (no version claimed)	0A20h
ADT-2 INCITS 472-2011	0A2Bh
ADT-2 T10/1742-D revision 06	0A22h
ADT-2 T10/1742-D revision 08	0A27h
A numeric ordered listing of the version descriptor value assignments is provided in F.9.	

Table 182 — Version descriptor values (part 2 of 12)

Standard	Version Descriptor Value
ADT-2 T10/1742-D revision 09	0A28h
ADT-3 (no version claimed)	0A60h
ATA/ATAPI-6 (no version claimed)	15E0h
ATA/ATAPI-6 INCITS 361-2002	15FDh
ATA/ATAPI-7 (no version claimed)	1600h
ATA/ATAPI-7 ISO/IEC 24739	161Eh
ATA/ATAPI-7 INCITS 397-2005	161Ch
ATA/ATAPI-7 T13/1532-D revision 3	1602h
ATA/ATAPI-8 ATA8-AAM (no version claimed)	1620h
ATA/ATAPI-8 ATA8-AAM INCITS 451-2008	1628h
ATA/ATAPI-8 ATA8-APT Parallel Transport (no version claimed)	1621h
ATA/ATAPI-8 ATA8-AST Serial Transport (no version claimed)	1622h
ATA/ATAPI-8 ATA8-ACS ATA/ATAPI Command Set (no version claimed)	1623h
ATA/ATAPI-8 ATA8-ACS INCITS 452-2009 w/ Amendment 1	162Ah
BCC (no version claimed)	0380h
EPI (no version claimed)	0B20h
EPI INCITS TR-23 1999	0B3Ch
EPI T10/1134 revision 16	0B3Bh
Fast-20 (no version claimed)	0AC0h
Fast-20 INCITS 277-1996	0ADCh
Fast-20 T10/1071 revision 06	0ADBh
FC 10GFC (no version claimed)	0EA0h
FC 10GFC ISO/IEC 14165-116	0EA3h
FC 10GFC INCITS 364-2003	0EA2h
FC 10GFC ISO/IEC 14165-116 with AM1	0EA5h
FC 10GFC INCITS 364-2003 with AM1 INCITS 364/AM1-2007	0EA6h
FC-AL (no version claimed)	0D40h
FC-AL INCITS 272-1996	0D5Ch
FC-AL-2 (no version claimed)	0D60h
FC-AL-2 ISO/IEC 14165-122 with AM1 & AM2	0D65h
FC-AL-2 INCITS 332-1999 with AM1-2003 & AM2-2006	0D63h
FC-AL-2 INCITS 332-1999 with Amnd 2 AM2-2006	0D64h
FC-AL-2 INCITS 332-1999 with Amnd 1 AM1-2003	0D7Dh
FC-AL-2 INCITS 332-1999	0D7Ch
FC-AL-2 T11/1133-D revision 7.0	0D61h
FC-DA (no version claimed)	12E0h
FC-DA ISO/IEC 14165-341	12E9h
FC-DA INCITS TR-36 2004	12E8h
FC-DA T11/1513-DT revision 3.1	12E2h
FC-DA-2 (no version claimed)	12C0h
A numeric ordered listing of the version descriptor value assignments is provided in F.9.	

Table 182 — Version descriptor values (part 3 of 12)

Standard	Version Descriptor Value
FC-DA-2 INCITS TR-49 2012	12C9h
FC-DA-2 T11/1870DT revision 1.04	12C3h
FC-DA-2 T11/1870DT revision 1.06	12C5h
FC-FLA (no version claimed)	1320h
FC-FLA INCITS TR-20 1998	133Ch
FC-FLA T11/1235 revision 7	133Bh
FC-FS (no version claimed)	0DA0h
FC-FS ISO/IEC 14165-251	0DBDh
FC-FS INCITS 373-2003	0DBCh
FC-FS T11/1331-D revision 1.2	0DB7h
FC-FS T11/1331-D revision 1.7	0DB8h
FC-FS-2 (no version claimed)	0E00h
FC-FS-2 INCITS 242-2007 with AM1 INCITS 242/AM1-2007	0E03h
FC-FS-2 INCITS 242-2007	0E02h
FC-FS-3 (no version claimed)	0EE0h
FC-FS-3 INCITS 470-2011	0EEBh
FC-FS-3 T11/1861-D revision 0.9	0EE2h
FC-FS-3 T11/1861-D revision 1.0	0EE7h
FC-FS-3 T11/1861-D revision 1.10	0EE9h
FC-FS-4 (no version claimed)	0F60h
FC-LS (no version claimed)	0E20h
FC-LS INCITS 433-2007	0E29h
FC-LS T11/1620-D revision 1.62	0E21h
FC-LS-2 (no version claimed)	0F00h
FC-LS-2 INCITS 477-2011	0F07h
FC-LS-2 T11/2103-D revision 2.11	0F03h
FC-LS-2 T11/2103-D revision 2.21	0F05h
FC-LS-3 (no version claimed)	0F80h
FCP (no version claimed)	08C0h
FCP INCITS 269-1996	08DCh
FCP T10/0993-D revision 12	08DBh
FC-PH (no version claimed)	0D20h
FC-PH INCITS 230-1994	0D3Bh
FC-PH INCITS 230-1994 with Amnd 1 INCITS 230/AM1-1996	0D3Ch
FC-PH-3 (no version claimed)	0D80h
FC-PH-3 INCITS 303-1998	0D9Ch
FC-PI (no version claimed)	0DC0h
FC-PI INCITS 352-2002	0DDCh
FC-PI-2 (no version claimed)	0DE0h
FC-PI-2 INCITS 404-2006	0DE4h
A numeric ordered listing of the version descriptor value assignments is provided in F.9.	

Table 182 — Version descriptor values (part 4 of 12)

Standard	Version Descriptor Value
FC-PI-2 T11/1506-D revision 5.0	0DE2h
FC-PI-3 (no version claimed)	0E60h
FC-PI-3 INCITS 460-2011	0E6Eh
FC-PI-3 T11/1625-D revision 2.0	0E62h
FC-PI-3 T11/1625-D revision 2.1	0E68h
FC-PI-3 T11/1625-D revision 4.0	0E6Ah
FC-PI-4 (no version claimed)	0E80h
FC-PI-4 INCITS 450-2009	0E88h
FC-PI-4 T11/1647-D revision 8.0	0E82h
FC-PI-5 (no version claimed)	0F20h
FC-PI-5 INCITS 479-2011	0F2Eh
FC-PI-5 T11/2118-D revision 2.00	0F27h
FC-PI-5 T11/2118-D revision 3.00	0F28h
FC-PI-5 T11/2118-D revision 6.00	0F2Ah
FC-PI-5 T11/2118-D revision 6.10	0F2Bh
FC-PI-6 (no version claimed)	0F40h
FC-PLDA (no version claimed)	1340h
FC-PLDA INCITS TR-19 1998	135Ch
FC-PLDA T11/1162 revision 2.1	135Bh
FCP-2 (no version claimed)	0900h
FCP-2 INCITS 350-2003	0917h
FCP-2 T10/1144-D revision 8	0918h
FCP-2 T10/1144-D revision 4	0901h
FCP-2 T10/1144-D revision 7	0915h
FCP-2 T10/1144-D revision 7a	0916h
FCP-3 (no version claimed)	0A00h
FCP-3 ISO/IEC 14776-223	0A1Ch
FCP-3 INCITS 416-2006	0A11h
FCP-3 T10/1560-D revision 4	0A0Fh
FCP-3 T10/1560-D revision 3f	0A07h
FCP-4 (no version claimed)	0A40h
FCP-4 INCITS 481-2012	0A46h
FCP-4 T10/1828-D revision 01	0A42h
FCP-4 T10/1828-D revision 02	0A44h
FCP-4 T10/1828-D revision 02b	0A45h
FC-SCM (no version claimed)	12A0h
FC-SCM INCITS TR-47 2012	12AAh
FC-SCM T11/1824DT revision 1.0	12A3h
FC-SCM T11/1824DT revision 1.1	12A5h
FC-SCM T11/1824DT revision 1.4	12A7h
A numeric ordered listing of the version descriptor value assignments is provided in F.9.	

Table 182 — Version descriptor values (part 5 of 12)

Standard	Version Descriptor Value
FC-SP (no version claimed)	0E40h
FC-SP INCITS 426-2007	0E45h
FC-SP T11/1570-D revision 1.6	0E42h
FC-SP-2 (no version claimed)	0EC0h
FC-Tape (no version claimed)	1300h
FC-Tape INCITS 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
IEEE 1667 (no version claimed)	FFC0h
IEEE 1667-2006	FFC1h
IEEE 1667-2009	FFC2h
iSCSI (no version claimed)	0960h
iSCSI (versions as described via RFC 7144)	0961h to 097Fh
MMC (no version claimed)	0140h
MMC INCITS 304-1997	015Ch
MMC T10/1048-D revision 10a	015Bh
MMC-2 (no version claimed)	0240h
MMC-2 INCITS 333-2000	025Ch
MMC-2 T10/1228-D revision 11a	025Bh
MMC-2 T10/1228-D revision 11	0255h
MMC-3 (no version claimed)	02A0h
MMC-3 INCITS 360-2002	02B8h
MMC-3 T10/1363-D revision 10g	02B6h
MMC-3 T10/1363-D revision 9	02B5h
MMC-4 (no version claimed)	03A0h
MMC-4 INCITS 401-2005	03BFh
MMC-4 T10/1545-D revision 3	03BDh
MMC-4 T10/1545-D revision 3d	03BEh
MMC-4 T10/1545-D revision 5	03B0h
MMC-4 T10/1545-D revision 5a	03B1h
MMC-5 (no version claimed)	0420h
MMC-5 T10/1675-D revision 04	0432h
MMC-5 INCITS 430-2007	0434h
MMC-5 T10/1675-D revision 03	042Fh
MMC-5 T10/1675-D revision 03b	0431h
A numeric ordered listing of the version descriptor value assignments is provided in F.9.	

Table 182 — Version descriptor values (part 6 of 12)

Standard	Version Descriptor Value
MMC-6 (no version claimed)	04E0h
MMC-6 INCITS 468-2010 + MMC-6/AM1 INCITS 468-2010/AM 1	04E7h
MMC-6 INCITS 468-2010	04E6h
MMC-6 T10/1836-D revision 02b	04E3h
MMC-6 T10/1836-D revision 02g	04E5h
OCRW (no version claimed)	0280h
OCRW ISO/IEC 14776-381	029Eh
OSD (no version claimed)	0340h
OSD INCITS 400-2004	0356h
OSD T10/1355-D revision 10	0355h
OSD T10/1355-D revision 0	0341h
OSD T10/1355-D revision 7a	0342h
OSD T10/1355-D revision 8	0343h
OSD T10/1355-D revision 9	0344h
OSD-2 (no version claimed)	0440h
OSD-2 INCITS 458-2011	0448h
OSD-2 T10/1729-D revision 4	0444h
OSD-2 T10/1729-D revision 5	0446h
OSD-3 (no version claimed)	0560h
PQI (no version claimed)	2200h
PQI T10/BSR INCITS 490 revision 6	2204h
PQI T10/BSR INCITS 490 revision 7	2206h
PQI-2 (no version claimed)	2240h
RBC (no version claimed)	0220h
RBC INCITS 330-2000	023Ch
RBC T10/1240-D revision 10a	0238h
SAM (no version claimed)	0020h
SAM INCITS 270-1996	003Ch
SAM T10/0994-D revision 18	003Bh
SAM-2 (no version claimed)	0040h
SAM-2 ISO/IEC 14776-412	005Eh
SAM-2 INCITS 366-2003	005Ch
SAM-2 T10/1157-D revision 24	0055h
SAM-2 T10/1157-D revision 23	0054h
SAM-3 (no version claimed)	0060h
SAM-3 INCITS 402-2005	0077h
SAM-3 T10/1561-D revision 14	0076h
SAM-3 T10/1561-D revision 7	0062h
SAM-3 T10/1561-D revision 13	0075h
SAM-4 (no version claimed)	0080h
A numeric ordered listing of the version descriptor value assignments is provided in F.9.	

Table 182 — Version descriptor values (part 7 of 12)

Standard	Version Descriptor Value
SAM-4 ISO/IEC 14776-414	0092h
SAM-4 INCITS 447-2008	0090h
SAM-4 T10/1683-D revision 13	0087h
SAM-4 T10/1683-D revision 14	008Bh
SAM-5 (no version claimed)	00A0h
SAM-5 T10/2104-D revision 4	00A2h
SAS (no version claimed)	0BE0h
SAS INCITS 376-2003	0BFDh
SAS T10/1562-D revision 05	0BFCh
SAS T10/1562-D revision 01	0BE1h
SAS T10/1562-D revision 03	0BF5h
SAS T10/1562-D revision 04	0BFAh
SAS T10/1562-D revision 04	0BFBh
SAS-1.1 (no version claimed)	0C00h
SAS-1.1 ISO/IEC 14776-151	0C12h
SAS-1.1 INCITS 417-2006	0C11h
SAS-1.1 T10/1601-D revision 9	0C07h
SAS-1.1 T10/1601-D revision 10	0C0Fh
SAS-2 (no version claimed)	0C20h
SAS-2 INCITS 457-2010	0C2Ah
SAS-2 T10/1760-D revision 14	0C23h
SAS-2 T10/1760-D revision 15	0C27h
SAS-2 T10/1760-D revision 16	0C28h
SAS-2.1 (no version claimed)	0C40h
SAS-2.1 INCITS 478-2011	0C4Eh
SAS-2.1 INCITS 478-2011 w/ Amnd 1 INCITS 478/AM1-2014	0C4Fh
SAS-2.1 T10/2125-D revision 04	0C48h
SAS-2.1 T10/2125-D revision 06	0C4Ah
SAS-2.1 T10/2125-D revision 07	0C4Bh
SAS-3 (no version claimed)	0C60h
SAS-3 T10/BSR INCITS 519 revision 05a	0C63h
SAS-3 T10/BSR INCITS 519 revision 06	0C65h
SAS-4 (no version claimed)	0C80h
SAT (no version claimed)	1EA0h
SAT INCITS 431-2007	1EADh
SAT T10/1711-D revision 9	1EABh
SAT T10/1711-D revision 8	1EA7h
SAT-2 (no version claimed)	1EC0h
SAT-2 INCITS 465-2010	1ECAh
SAT-2 T10/1826-D revision 06	1EC4h
A numeric ordered listing of the version descriptor value assignments is provided in F.9.	

Table 182 — Version descriptor values (part 8 of 12)

Standard	Version Descriptor Value
SAT-2 T10/1826-D revision 09	1EC8h
SAT-3 (no version claimed)	1EE0h
SAT-3 T10/BSR INCITS 517 revision 4	1EE2h
SAT-3 T10/BSR INCITS 517 revision 7	1EE4h
SAT-4 (no version claimed)	1F00h
SBC (no version claimed)	0180h
SBC INCITS 306-1998	019Ch
SBC T10/0996-D revision 08c	019Bh
SBC-2 (no version claimed)	0320h
SBC-2 ISO/IEC 14776-322	033Eh
SBC-2 INCITS 405-2005	033Dh
SBC-2 T10/1417-D revision 16	033Bh
SBC-2 T10/1417-D revision 5a	0322h
SBC-2 T10/1417-D revision 15	0324h
SBC-3 (no version claimed)	04C0h
SBC-3 T10/BSR INCITS 514 revision 35	04C3h
SBC-3 T10/BSR INCITS 514 revision 36	04C5h
SBC-4 (no version claimed)	0600h
SBP-2 (no version claimed)	08E0h
SBP-2 INCITS 325-1998	08FCh
SBP-2 T10/1155-D revision 04	08FBh
SBP-3 (no version claimed)	0980h
SBP-3 INCITS 375-2004	099Ch
SBP-3 T10/1467-D revision 5	099Bh
SBP-3 T10/1467-D revision 1f	0982h
SBP-3 T10/1467-D revision 3	0994h
SBP-3 T10/1467-D revision 4	099Ah
SCC (no version claimed)	0160h
SCC INCITS 276-1997	017Ch
SCC T10/1047-D revision 06c	017Bh
SCC-2 (no version claimed)	01E0h
SCC-2 INCITS 318-1998	01FCh
SCC-2 T10/1125-D revision 04	01FBh
SES (no version claimed)	01C0h
SES INCITS 305-1998	01DCh
SES T10/1212-D revision 08b	01DBh
SES INCITS 305-1998 w/ Amendment INCITS 305/AM1-2000	01DEh
SES T10/1212 revision 08b w/ Amendment INCITS 305/AM1-2000	01DDh
SES-2 (no version claimed)	03E0h
SES-2 ISO/IEC 14776-372	03F2h
A numeric ordered listing of the version descriptor value assignments is provided in F.9.	

Table 182 — Version descriptor values (part 9 of 12)

Standard	Version Descriptor Value
SES-2 INCITS 448-2008	03F0h
SES-2 T10/1559-D revision 16	03E1h
SES-2 T10/1559-D revision 19	03E7h
SES-2 T10/1559-D revision 20	03EBh
SES-3 (no version claimed)	0580h
SFSC (no version claimed)	05E0h
SIP (no version claimed)	08A0h
SIP INCITS 292-1997	08BCh
SIP T10/0856-D revision 10	08BBh
SMC (no version claimed)	01A0h
SMC ISO/IEC 14776-351	01BEh
SMC INCITS 314-1998	01BCh
SMC T10/0999-D revision 10a	01BBh
SMC-2 (no version claimed)	02E0h
SMC-2 INCITS 382-2004	02FEh
SMC-2 T10/1383-D revision 7	02FDh
SMC-2 T10/1383-D revision 5	02F5h
SMC-2 T10/1383-D revision 6	02FCh
SMC-3 (no version claimed)	0480h
SMC-3 INCITS 484-2012	0486h
SMC-3 T10/1730-D revision 15	0482h
SMC-3 T10/1730-D revision 16	0484h
SOP (no version claimed)	21E0h
SOP T10/BSR INCITS 489 revision 4	21E4h
SOP T10/BSR INCITS 489 revision 5	21E6h
SOP-2 (no version claimed)	2220h
SPC (no version claimed)	0120h
SPC INCITS 301-1997	013Ch
SPC T10/0995-D revision 11a	013Bh
SPC-2 (no version claimed)	0260h
SPC-2 ISO/IEC 14776-452	0278h
SPC-2 INCITS 351-2001	0277h
SPC-2 T10/1236-D revision 20	0276h
SPC-2 T10/1236-D revision 12	0267h
SPC-2 T10/1236-D revision 18	0269h
SPC-2 T10/1236-D revision 19	0275h
SPC-3 (no version claimed)	0300h
SPC-3 ISO/IEC 14776-453	0316h
SPC-3 INCITS 408-2005	0314h
SPC-3 T10/1416-D revision 7	0301h
A numeric ordered listing of the version descriptor value assignments is provided in F.9.	

Table 182 — Version descriptor values (part 10 of 12)

Standard	Version Descriptor Value
SPC-3 T10/1416-D revision 21	0307h
SPC-3 T10/1416-D revision 22	030Fh
SPC-3 T10/1416-D revision 23	0312h
SPC-4 (no version claimed)	0460h
SPC-4 T10/BSR INCITS 513 revision 16	0461h
SPC-4 T10/BSR INCITS 513 revision 18	0462h
SPC-4 T10/BSR INCITS 513 revision 23	0463h
SPC-4 T10/BSR INCITS 513 revision 36	0466h
SPC-4 T10/BSR INCITS 513 revision 37	0468h
SPC-4 T10/BSR INCITS 513 revision 37a	0469h
SPC-5 (no version claimed)	05C0h
SPI (no version claimed)	0AA0h
SPI INCITS 253-1995	0ABAh
SPI T10/0855-D revision 15a	0AB9h
SPI INCITS 253-1995 with SPI Amnd INCITS 253/AM1-1998	0ABCh
SPI T10/0855-D revision 15a with SPI Amnd revision 3a	0ABBh
SPI-2 (no version claimed)	0AE0h
SPI-2 INCITS 302-1999	0AFCh
SPI-2 T10/1142-D revision 20b	0AFBh
SPI-3 (no version claimed)	0B00h
SPI-3 INCITS 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
SPI-4 INCITS 362-2002	0B56h
SPI-4 T10/1365-D revision 7	0B54h
SPI-4 T10/1365-D revision 9	0B55h
SPI-4 T10/1365-D revision 10	0B59h
SPI-5 (no version claimed)	0B60h
SPI-5 INCITS 367-2003	0B7Ch
SPI-5 T10/1525-D revision 6	0B7Bh
SPI-5 T10/1525-D revision 3	0B79h
SPI-5 T10/1525-D revision 5	0B7Ah
SPL (no version claimed)	20A0h
SPL ISO/IEC 14776-261:2012	20AAh
SPL INCITS 476-2011	20A7h
SPL INCITS 476-2011 + SPL AM1 INCITS 476/AM1 2012	20A8h
SPL T10/2124-D revision 6a	20A3h
SPL T10/2124-D revision 7	20A5h
A numeric ordered listing of the version descriptor value assignments is provided in F.9.	

Table 182 — Version descriptor values (part 11 of 12)

Standard	Version Descriptor Value
SPL-2 (no version claimed)	20C0h
SPL-2 INCITS 505-2013	20C8h
SPL-2 T10/BSR INCITS 505 revision 4	20C2h
SPL-2 T10/BSR INCITS 505 revision 5	20C4h
SPL-3 (no version claimed)	20E0h
SPL-3 T10/BSR INCITS 492 revision 6	20E4h
SPL-3 T10/BSR INCITS 492 revision 7	20E6h
SPL-4 (no version claimed)	2100h
SRP (no version claimed)	0940h
SRP INCITS 365-2002	095Ch
SRP T10/1415-D revision 16a	0955h
SRP T10/1415-D revision 10	0954h
SSA-PH2 (no version claimed)	1360h
SSA-PH2 INCITS 293-1996	137Ch
SSA-PH2 T10.1/1145-D revision 09c	137Bh
SSA-PH3 (no version claimed)	1380h
SSA-PH3 INCITS 307-1998	139Ch
SSA-PH3 T10.1/1146-D revision 05b	139Bh
SSA-S2P (no version claimed)	0880h
SSA-S2P INCITS 294-1996	089Ch
SSA-S2P T10.1/1121-D revision 07b	089Bh
SSA-S3P (no version claimed)	0860h
SSA-S3P INCITS 309-1998	087Ch
SSA-S3P T10.1/1051-D revision 05b	087Bh
SSA-TL1 (no version claimed)	0840h
SSA-TL1 INCITS 295-1996	085Ch
SSA-TL1 T10.1/0989-D revision 10b	085Bh
SSA-TL2 (no version claimed)	0820h
SSA-TL2 INCITS 308-1998	083Ch
SSA-TL2 T10.1/1147-D revision 05b	083Bh
SSC (no version claimed)	0200h
SSC INCITS 335-2000	021Ch
SSC T10/0997-D revision 22	0207h
SSC T10/0997-D revision 17	0201h
SSC-2 (no version claimed)	0360h
SSC-2 INCITS 380-2003	037Dh
SSC-2 T10/1434-D revision 9	0375h
SSC-2 T10/1434-D revision 7	0374h
SSC-3 (no version claimed)	0400h
SSC-3 INCITS 467-2011	0409h
A numeric ordered listing of the version descriptor value assignments is provided in F.9.	

Table 182 — Version descriptor values (part 12 of 12)

Standard	Version Descriptor Value
SSC-3 T10/1611-D revision 04a	0403h
SSC-3 T10/1611-D revision 05	0407h
SSC-4 (no version claimed)	0520h
SSC-4 INCITS 516-2013	0527h
SSC-4 T10/BSR INCITS 516 revision 2	0523h
SSC-4 T10/BSR INCITS 516 revision 3	0525h
SSC-5 (no version claimed)	05A0h
SST (no version claimed)	0920h
SST T10/1380-D revision 8b	0935h
UAS (no version claimed)	1740h
UAS T10/2095-D revision 02	1743h
UAS T10/2095-D revision 04	1747h
UAS INCITS 471-2010	1748h
UAS-2 (no version claimed)	1780h
Universal Serial Bus Specification, Revision 1.1	1728h
Universal Serial Bus Specification, Revision 2.0	1729h
USB Mass Storage Class Bulk-Only Transport, Revision 1.0	1730h
ZBC (no version claimed)	0620h
ZBC BSR INCITS 536 revision 02	0622h
Version Descriptor Not Supported or No Standard Identified	0000h
Reserved	All others
A numeric ordered listing of the version descriptor value assignments is provided in F.9.	

6.6.3 SCSI Parallel Interface specific INQUIRY data

As shown in table 176, portions of bytes 6 and 7 and all of byte 56 of the standard INQUIRY data shall be used only by SCSI target devices that implement the SCSI Parallel Interface. For details on how the SPI-specific fields relate to the SCSI Parallel Interface see SPI-n (where n is 2 or greater). Table 183 shows the SPI-specific standard INQUIRY fields.

Table 183 — SPI-specific standard INQUIRY bits

Bit / Byte	7	6	5	4	3	2	1	0
6	n/a [a]							ADDR16
7	n/a [a]		WBUS16	SYNC	n/a [a]	Obsolete	n/a [a]	
	⋮							
56	Reserved				CLOCKING		QAS	IUS

[a] The meanings of these bits and fields are not specific to SPI-5. See table 176 for their definitions.

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

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

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

Table 184 defines the relationships between the ADDR16 bit and the WBUS16 bit.

Table 184 — 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 185.

Table 185 — CLOCKING field

Code	Description
00b	Indicates the target port supports only ST (see SPI-5)
01b	Indicates the target port supports only DT (see SPI-5)
10b	Reserved
11b	Indicates the target port supports ST and DT

A quick arbitration and selection supported (QAS) bit set to one indicates that the target port supports quick arbitration and selection (see SPI-5). A QAS bit set to zero indicates that the target port does not support quick arbitration and selection.

An information units supported (IUS) bit set to one indicates that the SCSI target device supports information unit transfers (see SPI-5). A IUS bit set to zero indicates that the SCSI target device does not support information unit transfers.

6.7 LOG SELECT command

6.7.1 Introduction

The LOG SELECT command (see table 186) requests the device server to manage the specified statistical information maintained by the SCSI target device about the SCSI target 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 (see 7.3) are defined to manage the log data. The LOG SELECT command provides a method for sending zero or more log pages in the command's parameter list.

Table 186 — LOG SELECT command

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

The OPERATION CODE field is defined in 4.2.5.1 and shall be set as shown in table 186 for the LOG SELECT command.

The parameter code reset (PCR) bit specifies whether the device server shall set parameters to their vendor specific default values (e.g., zero) as described in table 189.

The save parameters (SP) bit specifies whether the device server shall save parameters to non-volatile memory as described in table 189.

The page control (PC) field specifies which values the device server shall process for bounded data counter log parameters (see 7.3.2.2.2.2) and unbounded data counter log parameters (see 7.3.2.2.2.3) in response to a LOG SELECT command as described in table 187. The PC field shall be ignored for ASCII format list log parameters (see 7.3.2.2.2.4) and binary format list log parameters (see 7.3.2.2.2.5).

Table 187 — Page control (PC) field

Value	Description
00b	Threshold values [a]
01b	Cumulative values [a]
10b	Default threshold values
11b	Default cumulative values

[a] In order of preference, the threshold values and cumulative values for data counter parameters are:
1) the current values if there has been an update to a cumulative parameter value (e.g., by a LOG SELECT command or by a device specific event) in the specified page or pages since the last logical unit reset occurred;
2) the saved values, if saved parameters are implemented, current values have been saved, and an update has not occurred since the last logical unit reset; and
3) the vendor specific default values, if saved values are not available or not implemented.

When evaluated together, the combination of the values in the PCR bit, the SP bit, and the PC field specify the actions that a SCSI target device performs while processing a LOG SELECT command.

If the PARAMETER LIST LENGTH field is set to zero, the PAGE CODE field and SUBPAGE CODE field specify the log page or log pages to which the other CDB fields apply (see 6.7.2).

The device server shall terminate the command with CHECK CONDITION status, with the sense key set to ILLEGAL REQUEST, and the additional sense code set to INVALID FIELD IN CDB, if:

a) the PARAMETER LIST LENGTH field is set to a value other than zero, and:
   A) the PAGE CODE field is set to a value other than zero; or
   B) the SUBPAGE CODE field is set to a value other than zero.

The PARAMETER LIST LENGTH field specifies the length in bytes of the parameter list that shall be located in the Data-Out Buffer.

If the PARAMETER LIST LENGTH field is set to a value other than zero, the actions that a SCSI target device performs after receiving a LOG SELECT command are determined by the values in the PCR bit, the SP bit, and the PC field as described in table 351 (see 7.3.2.1).

If the PARAMETER LIST LENGTH field is set to zero, no log pages shall be transferred. This condition shall not be considered an error. The LOG SELECT command shall be processed as described in 6.7.2.

The CONTROL byte is defined in SAM-5.

6.7.2 Processing LOG SELECT when the parameter list length is zero

If the PARAMETER LIST LENGTH field is set to zero (i.e., when there is no parameter data being sent with a LOG SELECT command), then the SCSI target device responds by processing the log parameter values as described in this subclause.

The PAGE CODE field and SUBPAGE CODE field (see table 188) specify the log page or log pages to which the other CDB fields apply (see table 189).

Table 188 — PAGE CODE field and SUBPAGE CODE field

PAGE CODE field	SUBPAGE CODE field	Description
00h	00h	All log parameters in all log pages [a]
00h to 3Fh	01h to FEh	All log parameters in the log page specified by the page code and subpage code
00h to 3Fh	FFh	All log parameters in the log pages specified by page code and all subpage codes
01h to 3Fh	00h	All log parameters in the log page specified by the page code

[a] This is equivalent to the LOG SELECT command operation specified by SPC-3.

Table 189 defines the meaning of the combinations of values for the PCR bit, the SP bit, and the PC field.

Table 189 — PCR bit, SP bit, and PC field meanings when parameter list length is zero (part 1 of 3)

PCR bit	SP bit	PC field	Description
0b	0b	0xb	This is not an error. The device server shall make no changes to any log parameter values and shall not save any values to non-volatile media.
0b	1b	00b	The device server shall make no changes to any log parameter values and shall process the saving of current parameter values as follows: a) if the values are current threshold data counter parameters, then: A) if the device server implements saving of the current threshold values, then the device server shall save all current threshold values to non-volatile media; or B) if the device server does not implement saving of the current threshold values, then the device server shall terminate the command [a]. or b) if the values are current list parameters, then: A) if the device server implements saving of the current list parameters, then the device server shall save all current list parameters to non-volatile media; or B) if the device server does not implement saving of the current list parameters, then the device server shall terminate the command [a].

[a] The device server shall terminate the command with CHECK CONDITION status, with the sense key set to ILLEGAL REQUEST, and the additional sense code set to INVALID FIELD IN CDB.

[b] Vendor specific default threshold values and vendor specific default cumulative values may be zero.

Table 189 — PCR bit, SP bit, and PC field meanings when parameter list length is zero (part 2 of 3)

PCR bit	SP bit	PC field	Description
0b	1b	01b	The device server shall make no change to any log parameter values and shall process the saving of current parameter values as follows: a) if the values are current cumulative data counter parameters, then: A) if the device server implements saving of the current cumulative values, then the device server shall save all current cumulative values to non-volatile media; or B) if the device server does not implement saving of the current cumulative values, then the device server shall terminate the command [a]. or b) if the values are current list parameters, then: A) if the device server implements saving of the current list parameters, then the device server shall save all current list parameters to non-volatile media; or B) if the device server does not implement saving of the current list parameters, then the device server shall terminate the command [a].
0b	xb	10b	The device server shall set all current threshold values to the vendor specific default threshold values [b] and shall not save any values to non-volatile media.
0b	xb	11b	The device server shall set all current cumulative values to the vendor specific default cumulative values [b] and shall not save any values to non-volatile media.
1b	0b	xxb	The device server shall: 1) set all current threshold values to the vendor specific default threshold values [b]; 2) set all current cumulative values to the vendor specific default cumulative values [b]; 3) set all list parameters to their vendor specific default values; and 4) not save any values to non-volatile media.
1b	1b	00b	The device server shall process the saving of current threshold values as follows: a) if the device server implements saving of the current threshold values, then the device server shall: 1) save all current threshold values to non-volatile media; Note: Device servers compliant with SPC-3 may save the log parameter values to non-volatile media after setting the log parameter values instead of before setting the log parameter values. 2) set all current threshold values to the vendor specific default threshold values [b]; 3) set all current cumulative values to the vendor specific default cumulative values [b], and 4) set all list parameters to their vendor specific default values. or b) if the device server does not implement saving of the current threshold values, then the device server shall terminate the command [a].

[a] The device server shall terminate the command with CHECK CONDITION status, with the sense key set to ILLEGAL REQUEST, and the additional sense code set to INVALID FIELD IN CDB.

[b] Vendor specific default threshold values and vendor specific default cumulative values may be zero.

Table 189 — PCR bit, SP bit, and PC field meanings when parameter list length is zero (part 3 of 3)

PCR bit	SP bit	PC field	Description
1b	1b	01b	The device server shall process the saving of current cumulative values as follows: a) if the device server implements saving of the current cumulative values, then the device server shall: 1) save all current cumulative values to non-volatile media; Note: Device servers compliant with SPC-3 may save the log parameter values to non-volatile media after setting the log parameter values instead of before setting the log parameter values. 2) set all current threshold values to the vendor specific default threshold values [b]; 3) set all current cumulative values to the vendor specific default cumulative values [b], and 4) set all list parameters to their vendor specific default values; or b) if the device server does not implement saving of the current cumulative values, then the device server shall terminate the command [a].
1b	1b	1xb	The device server shall: 1) set all current threshold values to the vendor specific default threshold values [b]; 2) set all current cumulative values to the vendor specific default cumulative values [b]; 3) set all list parameters to their vendor specific default values; and 4) not save any values to non-volatile media.

[a] The device server shall terminate the command with CHECK CONDITION status, with the sense key set to ILLEGAL REQUEST, and the additional sense code set to INVALID FIELD IN CDB.

[b] Vendor specific default threshold values and vendor specific default cumulative values may be zero.

The current cumulative values may be updated by the device server as defined for the specific log page or by the application client using the LOG SELECT command. The current threshold values may only be modified by the application client via the LOG SELECT command.

NOTE 29 - Log pages or log parameters that are not available may become available at some later time (e.g., after the logical unit has become ready).

6.8 LOG SENSE command

The LOG SENSE command (see table 190) requests the device server to return statistical or other operational information maintained by the SCSI target device about the SCSI target device or its logical units. It is a complementary command to the LOG SELECT command.

Table 190 — LOG SENSE command

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

The OPERATION CODE field is defined in 4.2.5.1 and shall be set as shown in table 190 for the LOG SENSE command.

An save parameters (SP) bit set to zero specifies the device server shall perform the specified LOG SENSE command and shall not save any log parameters.

An SP bit set to one specifies that the device server shall perform the specified LOG SENSE command and shall save all log parameters identified as saveable by the DS bit (see 7.3) to a nonvolatile, vendor specific location. If the SP bit is set to one and the logical unit does not implement saving log parameters, the device server shall terminate the command with CHECK CONDITION status, with the sense key set to ILLEGAL REQUEST, and the additional sense code set to INVALID FIELD IN CDB.

The PC field shall be ignored for ASCII format list log parameters (see 7.3.2.2.2.4) and binary format list log parameters (see 7.3.2.2.2.5).

For bounded data counter log parameters (see 7.3.2.2.2.2) and unbounded data counter log parameters (see 7.3.2.2.2.3), the page control (PC) field (see table 187 in 6.7.1) specifies which log parameter values are to be returned by a device server in response to a LOG SENSE command.

For ASCII format list log parameters (see 7.3.2.2.2.4) and binary format list log parameters (see 7.3.2.2.2.5), the PC field shall be ignored. If the parameters specified by the PAGE CODE field and SUBPAGE CODE field in the CDB are list parameters, then the parameter values returned by a device server in response to a LOG SENSE command are determined as follows in order of preference:

1) the current list log parameter values, if there has been an update to a list log parameter value (e.g., by a LOG SELECT command or by a device specific event) in the specified page or pages since the last logical unit reset occurred;
2) the saved list log parameter values, if saved log parameters are implemented and an update has not occurred since the last logical unit reset; and
3) the vendor specific default list log parameter values, if saved values are not available or not implemented and an update has not occurred since the last logical unit reset.

The PAGE CODE field and SUBPAGE CODE field specify which log page of data is being requested (see 7.3). If the log page specified by the page code and subpage code combination is reserved or not implemented, then the device server shall terminate the command with CHECK CONDITION status, with the sense key set to ILLEGAL REQUEST, and 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 logical unit, whichever is less. If the PARAMETER POINTER field is set to:

a) 0000h, then the device server shall begin the transfer of the parameter data with the log parameter having the lowest parameter code that is implemented by the device server for the specified log page;
b) a value specifying the parameter code of a log parameter implemented by the device server for the specified log page, then the device server shall begin the transfer of the parameter data with the log parameter having the specified code;
c) a value that does not specify a parameter code implemented by the device server for the specified log page but is less than the largest parameter code implemented by the device server, then the device server shall begin the transfer of the parameter data with the log parameter implemented by the device server that has the next largest parameter code after the specified value; or
d) a value greater than the largest parameter code implemented by the device server for the specified log page, then the device server shall terminate the command with CHECK CONDITION status, with the sense key set to ILLEGAL REQUEST, and the additional sense code set to INVALID FIELD IN CDB.

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

The ALLOCATION LENGTH field is defined in 4.2.5.6.

The CONTROL byte is defined in SAM-5.

6.9 MANAGEMENT PROTOCOL IN command

6.9.1 MANAGEMENT PROTOCOL IN command description

The MANAGEMENT PROTOCOL IN command (see table 191) requests the device server to return management protocol information (see 6.9.2) or the results of one or more MANAGEMENT PROTOCOL OUT commands (see 6.10).

Table 191 — MANAGEMENT PROTOCOL IN command

Bit / Byte	7	6	5	4	3	2	1	0
0	OPERATION CODE (A3h)							
1	Reserved			SERVICE ACTION (10h)				
2	MANAGEMENT PROTOCOL							
3	MANAGEMENT PROTOCOL SPECIFIC1							
•••								
5								
6	(MSB)							
•••	ALLOCATION LENGTH							
9								(LSB)
10	MANAGEMENT PROTOCOL SPECIFIC2							
11	CONTROL							

The OPERATION CODE field is defined in 4.2.5.1 and shall be set as shown in table 191 for the MANAGEMENT PROTOCOL IN command.

The MANAGEMENT PROTOCOL field (see table 192) specifies which management protocol is being used.

Table 192 — MANAGEMENT PROTOCOL field in MANAGEMENT PROTOCOL IN command

Code	Description	Reference
00h	Management protocol information	6.9.2
01h to 2Fh	Reserved	
30h to 35h	Defined by the SNIA	3.1.169
36h to EFh	Reserved	
F0h to FFh	Vendor Specific	

The contents of the MANAGEMENT PROTOCOL SPECIFIC1 field and MANAGEMENT PROTOCOL SPECIFIC2 field depend on the protocol specified by the MANAGEMENT PROTOCOL field (see table 192).

The ALLOCATION LENGTH field is defined in 4.2.5.6.

The CONTROL byte is defined in SAM-5.

Indications of data overrun or underrun and the mechanism, if any, for processing retries depend on the protocol specified by the MANAGEMENT PROTOCOL field (see table 192).

Any association between a previous MANAGEMENT PROTOCOL OUT command and the data transferred by a MANAGEMENT PROTOCOL IN command depends on the protocol specified by the MANAGEMENT PROTOCOL field (see table 192). If the device server has no data to transfer (e.g., the results for any previous

MANAGEMENT PROTOCOL OUT commands are not yet available), then the device server may transfer data indicating it has no other data to transfer.

The format of the data transferred depends on the protocol specified by the MANAGEMENT PROTOCOL field (see table 192).

The device server shall retain data resulting from a MANAGEMENT PROTOCOL OUT command, if any, until one of the following events is processed:

a) transfer of the data via a MANAGEMENT PROTOCOL IN command from the same I_T_L nexus as defined by the protocol specified by the MANAGEMENT PROTOCOL field (see table 192);
b) logical unit reset (see SAM-5); or
c) I_T nexus loss (see SAM-5) associated with the I_T nexus that sent the MANAGEMENT PROTOCOL OUT command.

6.9.2 Management protocol information description

6.9.2.1 Overview

The management protocol information management protocol (i.e., the MANAGEMENT PROTOCOL field set to 00h in a MANAGEMENT PROTOCOL IN command) returns management protocol related information from the logical unit. A MANAGEMENT PROTOCOL IN command in which the MANAGEMENT PROTOCOL field is set to 00h is not associated with any previous MANAGEMENT PROTOCOL OUT command and shall be processed without regard for whether a MANAGEMENT PROTOCOL OUT command has been processed.

If the MANAGEMENT PROTOCOL IN command is supported, the MANAGEMENT PROTOCOL value of 00h shall be supported as defined in this standard.

6.9.2.2 CDB description

If the MANAGEMENT PROTOCOL field is set to 00h in a MANAGEMENT PROTOCOL IN command, the MANAGEMENT PROTOCOL SPECIFIC1 field is reserved and the MANAGEMENT PROTOCOL SPECIFIC2 field (see table 193) contains a single numeric value as defined in 3.6.

Table 193 — MANAGEMENT PROTOCOL SPECIFIC2 field for MANAGEMENT PROTOCOL IN protocol 00h

Code	Description	Support	Reference
00h	Supported management protocol list	Mandatory	6.9.2.3
01h to FFh	Reserved		

All other CDB fields for MANAGEMENT PROTOCOL IN command shall meet the requirements stated in 6.9.1.

Each time a MANAGEMENT PROTOCOL IN command with the MANAGEMENT PROTOCOL field set to 00h is received, the device server shall transfer the data defined 6.9.2.3 starting with byte 0.

6.9.2.3 Supported management protocols list description

If the MANAGEMENT PROTOCOL field is set to 00h and the MANAGEMENT PROTOCOL SPECIFIC2 field is set to 00h in a MANAGEMENT PROTOCOL IN command, then the parameter data shall have the format shown in table 194.

Table 194 — Supported management protocols MANAGEMENT PROTOCOL IN parameter data

Bit / Byte	7	6	5	4	3	2	1	0
0	Reserved							
•••								
5								
6	(MSB)			SUPPORTED MANAGEMENT PROTOCOL LIST LENGTH (n-7)				
7								(LSB)
Supported management protocol list								
8	SUPPORTED MANAGEMENT PROTOCOL (00h) [first]							
	⋮							
n	SUPPORTED MANAGEMENT PROTOCOL [last]							

The SUPPORTED MANAGEMENT PROTOCOL LIST LENGTH field indicates the total length, in bytes, of the supported management protocol list that follows.

Each SUPPORTED MANAGEMENT PROTOCOL field in the supported management protocols list shall contain one of the management protocol values supported by the logical unit. The values shall be listed in ascending order starting with 00h.

6.10 MANAGEMENT PROTOCOL OUT command

The MANAGEMENT PROTOCOL OUT command (see table 195) is used to send data to the logical unit. The data sent specifies one or more operations to be performed by the logical unit. The format and function of the operations depends on the contents of the MANAGEMENT PROTOCOL field (see table 195). Depending on the protocol specified by the MANAGEMENT PROTOCOL field, the application client may use the MANAGEMENT PROTOCOL IN command (see 6.9) to retrieve data derived from these operations.

Table 195 — MANAGEMENT PROTOCOL OUT command

Bit / Byte	7	6	5	4	3	2	1	0
0	OPERATION CODE (A4h)							
1	Reserved			SERVICE ACTION (10h)				
2	MANAGEMENT PROTOCOL							
3	MANAGEMENT PROTOCOL SPECIFIC1							
•••								
5								
6	(MSB)							
•••	PARAMETER LIST LENGTH							
9								(LSB)
10	MANAGEMENT PROTOCOL SPECIFIC2							
11	CONTROL							

The OPERATION CODE field is defined in 4.2.5.1 and shall be set as shown in table 195 for the MANAGEMENT PROTOCOL OUT command.

The MANAGEMENT PROTOCOL field (see table 196) specifies which management protocol is being used.

Table 196 — MANAGEMENT PROTOCOL field in MANAGEMENT PROTOCOL OUT command

Code	Description	Reference
00h to 2Fh	Reserved	3.1.169
30h to 35h	Defined by the SNIA	
36h to EFh	Reserved	
F0h to FFh	Vendor Specific	

The contents of the MANAGEMENT PROTOCOL SPECIFIC1 field and MANAGEMENT PROTOCOL SPECIFIC2 field depend on the protocol specified by the MANAGEMENT PROTOCOL field (see table 196).

The PARAMETER LIST LENGTH field is defined in 4.2.5.5.

The CONTROL byte is defined in SAM-5.

Any association between a MANAGEMENT PROTOCOL OUT command and a subsequent MANAGEMENT PROTOCOL IN command depends on the protocol specified by the MANAGEMENT PROTOCOL field (see table 196). Each management protocol shall define whether:

a) the device server shall complete the command with GOOD status as soon as it determines the data has been correctly received. An indication that the data has been processed is obtained by sending a MANAGEMENT PROTOCOL IN command and processing the parameter data that is returned; or

b) the device server shall complete the command with GOOD status only after the data has been successfully processed and an associated MANAGEMENT PROTOCOL IN command is not required.

The format of the data transferred depends on the protocol specified by the MANAGEMENT PROTOCOL field (see table 196).

6.11 MODE SELECT(6) command

The MODE SELECT(6) command (see table 197) requests the device server to set the specified medium, logical unit, or peripheral device parameters. 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 mode pages, page lengths, and other parameters.

NOTE 30 - Implementations should migrate from the MODE SELECT(6) command to the MODE SELECT(10) command.

Table 197 — 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								
4	PARAMETER LIST LENGTH							
5	CONTROL							

The OPERATION CODE field is defined in 4.2.5.1 and shall be set as shown in table 197 for the MODE SELECT(6) command.

If an application client sends a MODE SELECT command that changes any parameters applying to other I_T nexuses, then the device server shall establish a unit attention condition (see SAM-5) for the initiator port associated with every I_T nexus except the I_T nexus on which the MODE SELECT command was received, with the additional sense code set to MODE PARAMETERS CHANGED.

A page format (PF) bit set to zero specifies that all parameters after the block descriptors are vendor specific. A PF bit set to one specifies that the MODE SELECT parameters following the header and block descriptor(s) are structured as pages of related parameters (see 7.5).

A save pages (SP) bit set to zero specifies that the device server shall perform the specified MODE SELECT operation and shall not save any mode pages. If the device server supports only saved mode pages (i.e., the device server does not support a distinction between current mode pages and saved mode pages) and the SP bit is set to zero, then the device server shall terminate the command with CHECK CONDITION status, with the sense key set to ILLEGAL REQUEST, and the additional sense code set to INVALID FIELD IN CDB. An SP bit set to one specifies that the device server shall perform the specified MODE SELECT operation and shall save to a nonvolatile vendor specific location all the saveable mode pages including any sent in the parameter list. Mode pages that are saved are specified by the parameter saveable (PS) bit in each mode page (see 7.5). If the PS bit is set to one in the MODE SENSE data, then the mode page shall be saveable by issuing a MODE SELECT command with the SP bit set to one. If the logical unit does not implement saved mode pages and the SP bit is set to one, then the device server shall terminate the command with CHECK CONDITION status, with the sense key 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 specifies that the Data-Out Buffer shall be empty. This condition shall not be considered an error.

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

The CONTROL byte is defined in SAM-5.

The mode parameter list for the MODE SELECT command and MODE SENSE command is defined in 7.5. 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 unique for each device-type may be found in the applicable command standards.

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 in response to any of 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 the current value of the mode parameter;
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 mode page;
d) if the application client sends an 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 and the device server checks reserved fields.

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, then the device server handles the condition by either:

a) rounding the parameter to an acceptable value and terminating the command as described in 5.8; or
b) terminating the command with CHECK CONDITION status, with the sense key set to ILLEGAL REQUEST, and the additional sense code set 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 31 - 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.

6.12 MODE SELECT(10) command

The MODE SELECT(10) command (see table 198) requests the device server to set the specified medium, logical unit, or peripheral device parameters. See the MODE SELECT(6) command (6.11) 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, page lengths, and other parameters. Device servers that implement the MODE SELECT(10) command shall also implement the MODE SENSE(10) command.

Table 198 — 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							
•••								
6								
7	(MSB)			PARAMETER LIST LENGTH				
8								(LSB)
9	CONTROL							

The OPERATION CODE field is defined in 4.2.5.1 and shall be set as shown in table 198 for the MODE SELECT(10) command.

The CONTROL byte is defined in SAM-5.

See the MODE SELECT(6) command (6.11) for a description of the other fields and operation of this command.

6.13 MODE SENSE(6) command

6.13.1 MODE SENSE(6) command introduction

The MODE SENSE(6) command (see table 199) requests the device server to return the specified medium, logical unit, or peripheral device parameters. 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.

NOTE 32 - Implementations should migrate from the MODE SENSE(6) command to the MODE SENSE(10) command.

Table 199 — 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	SUBPAGE CODE							
4	ALLOCATION LENGTH							
5	CONTROL							

The OPERATION CODE field is defined in 4.2.5.1 and shall be set as shown in table 199 for the MODE SENSE(6) command.

A disable block descriptors (DBD) bit set to zero specifies that the device server may return zero or more block descriptors in the MODE SENSE parameter data (see 7.5). A DBD bit set to one specifies that the device server shall not return any block descriptors in the MODE SENSE parameter data.

The page control (PC) field specifies the type of mode parameter values to be returned in the mode pages. The PC field is defined in table 200.

Table 200 — Page control (PC) field

Code	Type of parameter	Reference
00b	Current values	6.13.2
01b	Changeable values	6.13.3
10b	Default values	6.13.4
11b	Saved values	6.13.5

The PC field only affects the mode parameters within the mode pages. The PS bit, SPF bit, PAGE CODE field, SUBPAGE CODE field, and PAGE LENGTH field should return current values (i.e., as if PC is set to 00b). The mode parameter header and mode parameter block descriptor should return current values.

Some SCSI target devices may not distinguish between current mode parameters 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 SP bit in the MODE SELECT command (see 6.11).

The PAGE CODE field and SUBPAGE CODE field (see table 201) specify which mode pages and subpages to return.

Table 201 — Mode page code usage in MODE SENSE commands for all devices

Page Code	Subpage Code	Description
00h	vendor specific	Vendor specific
01h to 3Eh	00h	See specific device types (i.e., page_0 mode page format (see 7.5.7))
	01h to DFh	See specific device types (i.e., sub_page mode page format (see 7.5.7))
	E0h to FEh	Vendor specific (i.e., sub_page mode page format (see 7.5.7))
	FFh	Return all supported subpages for the specified device specific mode page in the page_0 mode page format for subpage 00h and in the sub_page mode page format for subpages 01h to FEh
3Fh	00h	Return all subpage 00h mode pages in page_0 mode page format
	01h to FEh	Reserved
	FFh	Return all subpages for all mode pages in the page_0 mode page format for subpage 00h and in the sub_page mode page format for subpages 01h to FEh

The ALLOCATION LENGTH field is defined in 4.2.5.6.

The CONTROL byte is defined in SAM-5.

An application client may request any one or all of the supported mode pages from the device server. If the device server receives a MODE SENSE command with a page code value or subpage code value that is not implemented by the logical unit, then the device server shall terminate the command with CHECK CONDITION status, with the sense key set to ILLEGAL REQUEST, and the additional sense code set to INVALID FIELD IN CDB.

If an application client requests all supported mode pages, the device server shall return the supported pages in ascending page code order beginning with mode page 01h. If mode page 00h is implemented, the device server shall return mode page 00h after all other mode pages have been returned.

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 commands and MODE SENSE commands is defined in 7.5. 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.

6.13.2 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 mode parameters were restored to their saved values (see 6.11); or

c) the default values of the mode parameters if a MODE SELECT command has not successfully completed since the mode parameters were restored to their default values (see 6.11).

6.13.3 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 bits in the fields of the mode parameters that are changeable all shall be set to one and the bits in the fields of the mode parameters that are non-changeable (i.e., defined by the logical unit) all shall be set to zero.

If the logical unit does not implement changeable parameters mode pages and the device server receives a MODE SENSE command with 01b in the PC field, then the device server shall terminate the command with CHECK CONDITION status, with the sense key 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 results in an error condition (see 6.11).

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.

6.13.4 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 logical unit is not ready.

6.13.5 Saved values

A PC field value of 11b requests that the device server return the saved values of the mode parameters. Mode parameters not supported by the logical unit shall be set to zero. If saved values are not implemented, the device server shall terminate the command with CHECK CONDITION status, with 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 such that they are retained while the device is powered down. All saveable mode pages should be considered saved if a MODE SELECT command issued with the SP bit set to one has completed with a GOOD status or after the successful completion of a FORMAT UNIT command.

6.13.6 Initial responses

After a logical unit reset, the device server shall respond in the following manner:

a) if default values are requested, return the default values;
b) if saved values are requested, return 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, the device server shall terminate the command with CHECK CONDITION status, with the sense key set to NOT READY. If saved parameters are not implemented, respond as defined in 6.13.5; or
c) if current values are requested and the current values have been received as part of a MODE SELECT command, then the current values shall be returned. If the current values have not been received as part of a MODE SELECT command, the device server shall return:

A) the saved values, if saving is implemented and saved values are available; or
B) the default values.

6.14 MODE SENSE(10) command

The MODE SENSE(10) command (see table 202) requests the device server to return the specified medium, logical unit, or peripheral device parameters. 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 202 — 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	SUBPAGE CODE							
4	Reserved							
•••								
6								
7	(MSB)		ALLOCATION LENGTH					
8								(LSB)
9	CONTROL							

The OPERATION CODE field is defined in 4.2.5.1 and shall be set as shown in table 202 for the MODE SENSE(10) command.

If the Long LBA Accepted (LLBAA) bit is set to one, the device server is allowed to return parameter data with the LONGLBA bit equal to one (see 7.5.5). If LLBAA bit is set to zero, the LONGLBA bit shall be zero in the parameter data returned by the device server.

The CONTROL byte is defined in SAM-5.

See the MODE SENSE(6) command (6.13) for a description of the other fields and operation of this command.

6.15 PERSISTENT RESERVE IN command

6.15.1 PERSISTENT RESERVE IN command introduction

The PERSISTENT RESERVE IN command (see table 203) requests the device server to return information about persistent reservations and reservation keys (i.e., registrations) that are active within a device server. This command is used in conjunction with the PERSISTENT RESERVE OUT command (see 6.16).

Table 203 — PERSISTENT RESERVE IN command

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

The OPERATION CODE field is defined in 4.2.5.1 and shall be set as shown in table 203 for the PERSISTENT RESERVE IN command.

The SERVICE ACTION field is defined in 4.2.5.2. The service action codes for the PERSISTENT RESERVE IN command are defined in table 204.

Table 204 — PERSISTENT RESERVE IN service action codes

Code	Name	Description	Reference
00h	READ KEYS	Returns all registered reservation keys (i.e., registrations) as described in 5.12.6.2	6.15.2
01h	READ RESERVATION	Returns the current persistent reservations as described in 5.12.6.3	6.15.3
02h	REPORT CAPABILITIES	Returns capability information	6.15.4
03h	READ FULL STATUS	Returns complete information about all registrations and the persistent reservations, if any	6.15.5
04h to 1Fh	Reserved	Reserved	

The ALLOCATION LENGTH field is defined in 4.2.5.6. The PERSISTENT RESERVE IN parameter data includes a length field that indicates the number of parameter data bytes that follow in the parameter data. The allocation length should be set to a value large enough to include the length field for the specified service action.

The CONTROL byte is defined in SAM-5.

6.15.2 READ KEYS service action

The READ KEYS service action requests the device server to return a parameter list containing a header and a list of each currently registered I_T nexus' reservation key. If multiple I_T nexuses 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 5.12.6.2.

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 205.

Table 205 — PERSISTENT RESERVE IN parameter data for READ KEYS

Bit / Byte	7	6	5	4	3	2	1	0
0	(MSB)							
•••				PRGENERATION				
3								(LSB)
4	(MSB)							
•••				ADDITIONAL LENGTH (n-7)				
7								(LSB)
	Reservation key list							
8	(MSB)							
•••				Reservation key [first]				
15								(LSB)
				⋮				
n-7	(MSB)							
•••				Reservation key [last]				
n								(LSB)

The Persistent Reservations Generation (PRGENERATION) field shall contain the value of a 32-bit wrapping counter that the device server shall update (e.g., increment) during the processing of any PERSISTENT RESERVE OUT command as described in table 216 (see 6.16.2). The PRgeneration value shall not be updated by a PERSISTENT RESERVE IN command or by a PERSISTENT RESERVE OUT command that is terminated due to an error or reservation conflict. Regardless of the APTPL bit value the PRgeneration value shall be set to zero by a power on.

The ADDITIONAL LENGTH field indicates the number of bytes in the Reservation key list. The contents of the ADDITIONAL LENGTH field are not altered based on the allocation length (see 4.2.5.6).

The reservation key list contains the eight byte reservation keys for all I_T nexuses that have been registered (see 5.12.7).

6.15.3 READ RESERVATION service action

6.15.3.1 READ RESERVATION service action operation

The READ RESERVATION service action requests the device server to return parameter data that contains a header (see table 206) or the persistent reservation, if any, that is present in the device server (see table 207).

For more information on READ RESERVATION see 5.12.6.3.

If no persistent reservation is held, 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 206.

Table 206 — Format of PERSISTENT RESERVE IN parameter data for READ RESERVATION with no reservation held

Bit / Byte	7	6	5	4	3	2	1	0
0	(MSB)							
•••				PRGENERATION				
3								(LSB)
4	(MSB)							
•••				ADDITIONAL LENGTH (00000000h)				
7								(LSB)

The PRGENERATION field shall be as defined for the PERSISTENT RESERVE IN command with READ KEYS service action parameter data (see 6.15.2).

The ADDITIONAL LENGTH field shall be set to zero (see table 206), indicating that no persistent reservation is held.

If a persistent reservation is held, 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 207.

Table 207 — Format of PERSISTENT RESERVE IN parameter data for READ RESERVATION with a reservation held

Bit / Byte	7	6	5	4	3	2	1	0
0	(MSB)							
•••				PRGENERATION				
3								(LSB)
4	(MSB)							
•••				ADDITIONAL LENGTH (00000010h)				
7								(LSB)
8	(MSB)							
•••				RESERVATION KEY				
15								(LSB)
16								
•••				Obsolete				
19								
20				Reserved				
21	SCOPE				TYPE			
22								
23				Obsolete				

The PRGENERATION field shall be as defined for the PERSISTENT RESERVE IN command with READ KEYS service action parameter data.

The ADDITIONAL LENGTH field indicates the number of bytes that follow and shall be set as shown in table 207. The contents of the ADDITIONAL LENGTH field are not altered based on the allocation length (see 4.2.5.6).

The RESERVATION KEY field shall indicate the reservation key under which the persistent reservation is held (see 5.12.10).

The SCOPE field shall be set to LU_SCOPE (see 6.15.3.2).

The TYPE field shall indicate the persistent reservation type (see 6.15.3.3) specified in the PERSISTENT RESERVE OUT command that created the persistent reservation.

The obsolete fields in bytes 16 to 19, byte 22, and byte 23 were defined in SPC-2.

6.15.3.2 Persistent reservations scope

The SCOPE field (see table 208) shall be set to LU_SCOPE, specifying that the persistent reservation applies to the entire logical unit.

Table 208 — Persistent reservation SCOPE field

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

The LU_SCOPE scope shall be implemented by all device servers that implement PERSISTENT RESERVE OUT.

6.15.3.3 Persistent reservations type

The TYPE field (see table 209) specifies the characteristics of the persistent reservation being established for all logical blocks within the logical unit. Table 66 (see 5.12.1) defines the persistent reservation types under which each command defined in this standard is processed. Each other command standard defines the persistent reservation types under which each command defined in that command standard is processed.

Table 209 — Persistent reservation TYPE field

Code	Name	Description
0h		Obsolete
1h	Write Exclusive	**Access Restrictions**: Some commands (e.g., media-access write commands) are only allowed for the persistent reservation holder (see 5.12.10). **Persistent Reservation Holder:** There is only one persistent reservation holder.
2h		Obsolete
3h	Exclusive Access	**Access Restrictions**: Some commands (e.g., media-access commands) are only allowed for the persistent reservation holder (see 5.12.10). **Persistent Reservation Holder:** There is only one persistent reservation holder.
4h		Obsolete
5h	Write Exclusive – Registrants Only	**Access Restrictions**: Some commands (e.g., media-access write commands) are only allowed for registered I_T nexuses. **Persistent Reservation Holder:** There is only one persistent reservation holder (see 5.12.10).
6h	Exclusive Access – Registrants Only	**Access Restrictions**: Some commands (e.g., media-access commands) are only allowed for registered I_T nexuses. **Persistent Reservation Holder:** There is only one persistent reservation holder (see 5.12.10).
7h	Write Exclusive – All Registrants	**Access Restrictions**: Some commands (e.g., media-access write commands) are only allowed for registered I_T nexuses. **Persistent Reservation Holder:** Each registered I_T nexus is a persistent reservation holder (see 5.12.10).
8h	Exclusive Access – All Registrants	**Access Restrictions**: Some commands (e.g., media-access commands) are only allowed for registered I_T nexuses. **Persistent Reservation Holder:** Each registered I_T nexus is a persistent reservation holder (see 5.12.10).
9h to Fh	Reserved	

6.15.4 REPORT CAPABILITIES service action

The REPORT CAPABILITIES service action requests the device server to return information on persistent reservation features.

The format for the parameter data provided in response to a PERSISTENT RESERVE IN command with the REPORT CAPABILITIES service action is shown in table 210.

Table 210 — PERSISTENT RESERVE IN parameter data for REPORT CAPABILITIES

Bit / Byte	7	6	5	4	3	2	1	0
0	(MSB)							
1		LENGTH (0008h)						(LSB)
2	RLR_C	Reserved		CRH	SIP_C	ATP_C	Reserved	PTPL_C
3	TMV	ALLOW COMMANDS			Reserved			PTPL_A
4		PERSISTENT RESERVATION TYPE MASK						
5								
6		Reserved						
7								

The LENGTH field indicates the length in bytes of the parameter data and shall be set as shown in table 210 in the parameter data for a PERSISTENT RESERVE IN command with the REPORT CAPABILITIES service action. The contents of the LENGTH field are not altered based on the allocation length (see 4.2.5.6).

A replace lost reservation capable (RLR_C) bit set to one indicates that the device server supports the REPLACE LOST RESERVATION service action in the PERSISTENT RESERVE OUT command. An RLR_C bit set to zero indicates that the device server does not support the REPLACE LOST RESERVATION service action in the PERSISTENT RESERVE OUT command. If the RLR_C bit is set to zero then the device server shall not terminate any commands with CHECK CONDITION status with the sense key set to DATA PROTECT and the additional sense code set to PERSISTENT RESERVATION INFORMATION LOST (see 5.12.5.4).

A compatible reservation handling (CRH) bit set to one indicates that the device server supports the exceptions to the SPC-2 RESERVE commands and RELEASE commands described in 5.12.3. A CRH bit set to zero indicates that RESERVE commands and RELEASE commands are processed as defined in SPC-2.

A specify initiator ports capable (SIP_C) bit set to one indicates that the device server supports the SPEC_I_PT bit in the PERSISTENT RESERVE OUT command parameter data (see 6.16.3). An SIP_C bit set to zero indicates that the device server does not support the SPEC_I_PT bit in the PERSISTENT RESERVE OUT command parameter data.

An all target ports capable (ATP_C) bit set to one indicates that the device server supports the ALL_TG_PT bit in the PERSISTENT RESERVE OUT command parameter data. An ATP_C bit set to zero indicates that the device server does not support the ALL_TG_PT bit in the PERSISTENT RESERVE OUT command parameter data.

A persist through power loss capable (PTPL_C) bit set to one indicates that the device server supports the persist through power loss capability (see 5.12.5) for persistent reservations and the APTPL bit in the PERSISTENT RESERVE OUT command parameter data. An PTPL_C bit set to zero indicates that the device server does not support the persist through power loss capability.

A type mask valid (TMV) bit set to one indicates that the PERSISTENT RESERVATION TYPE MASK field contains a bit map indicating which persistent reservation types are supported by the device server. A TMV bit set to zero indicates that the PERSISTENT RESERVATION TYPE MASK field shall be ignored.

The ALLOW COMMANDS field (see table 211) indicates whether certain commands are allowed through certain types of persistent reservations.

Table 211 — ALLOW COMMANDS field (part 1 of 2)

Code	Description
000b	No information is provided about whether certain commands are allowed through certain types of persistent reservations.

001b	The device server allows the TEST UNIT READY command (see table 66 in 5.12.1) through Write Exclusive type reservations and Exclusive Access type reservations.
	The device server does not provide information about whether the following commands are allowed through Write Exclusive type reservations:
	a) MODE SENSE;
	b) READ ATTRIBUTE;
	c) READ BUFFER;
	d) RECEIVE COPY DATA(LID1);
	e) RECEIVE COPY OPERATING PARAMETERS;
	f) RECEIVE COPY FAILURE DETAILS(LID1);
	g) RECEIVE COPY STATUS(LID1);
	h) RECEIVE DIAGNOSTIC RESULTS;
	i) REPORT SUPPORTED OPERATION CODES;
	j) REPORT SUPPORTED TASK MANAGEMENT FUNCTIONS; and
	k) READ DEFECT DATA (see SBC-3).

010b	The device server allows the TEST UNIT READY command through Write Exclusive type reservations and Exclusive Access type reservations. The device server does not allow: a) the following commands through Write Exclusive type reservations: A) MODE SENSE; B) READ ATTRIBUTE; C) READ BUFFER; D) RECEIVE DIAGNOSTIC RESULTS; E) REPORT SUPPORTED OPERATION CODES; F) REPORT SUPPORTED TASK MANAGEMENT FUNCTIONS; and G) READ DEFECT DATA (see SBC-3); and b) the following commands through Write Exclusive type reservations or Exclusive Access type reservations: A) RECEIVE COPY DATA(LID1); B) RECEIVE COPY OPERATING PARAMETERS; C) RECEIVE COPY FAILURE DETAILS(LID1); and D) RECEIVE COPY STATUS(LID1).

Table 211 — ALLOW COMMANDS field (part 2 of 2)

Code	Description
011b	The device server allows: a) the TEST UNIT READY command through Write Exclusive type reservations and Exclusive Access type reservations; and b) the following commands through Write Exclusive type reservations: A) MODE SENSE; B) READ ATTRIBUTE; C) READ BUFFER; D) RECEIVE DIAGNOSTIC RESULTS; E) REPORT SUPPORTED OPERATION CODES; F) REPORT SUPPORTED TASK MANAGEMENT FUNCTIONS; and G) READ DEFECT DATA (see SBC-3). The device server does not allow the following commands through Write Exclusive type reservations or Exclusive Access type reservations: a) RECEIVE COPY DATA(LID1); b) RECEIVE COPY OPERATING PARAMETERS; c) RECEIVE COPY FAILURE DETAILS(LID1); and d) RECEIVE COPY STATUS(LID1).
100b	The device server allows: a) the following commands through Write Exclusive and Exclusive Access persistent reservations: A) TEST UNIT READY; B) RECEIVE COPY DATA(LID1); C) RECEIVE COPY OPERATING PARAMETERS; D) RECEIVE COPY FAILURE DETAILS(LID1); and E) RECEIVE COPY STATUS(LID1); and b) the following commands through Write Exclusive persistent reservations: A) MODE SENSE; B) READ ATTRIBUTE; C) READ BUFFER; D) RECEIVE DIAGNOSTIC RESULTS; and E) REPORT SUPPORTED OPERATION CODES; and F) REPORT SUPPORTED TASK MANAGEMENT FUNCTIONS; G) READ DEFECT DATA (see SBC-3).
101b	The device server allows: a) the following commands through Write Exclusive and Exclusive Access persistent reservations: A) TEST UNIT READY; B) RECEIVE COPY DATA(LID1); C) RECEIVE COPY OPERATING PARAMETERS; D) RECEIVE COPY FAILURE DETAILS(LID1); E) RECEIVE COPY STATUS(LID1); F) REPORT SUPPORTED OPERATION CODES; and G) REPORT SUPPORTED TASK MANAGEMENT FUNCTIONS; and b) the following commands through Write Exclusive persistent reservations: A) MODE SENSE; B) READ ATTRIBUTE; C) READ BUFFER; D) RECEIVE DIAGNOSTIC RESULTS; and E) READ DEFECT DATA (see SBC-3).
all others	Reserved

A Persist Through Power Loss Activated (PTPL_A) bit set to one indicates that the persist through power loss capability is activated (see 5.12.5). A PTPL_A bit set to zero indicates that the persist through power loss capability is not activated.

The PERSISTENT RESERVATION TYPE MASK field (see table 212) contains a bit map that indicates the persistent reservation types that are supported by the device server.

Table 212 — Persistent Reservation Type Mask format

Bit / Byte	7	6	5	4	3	2	1	0
4	WR_EX_AR	EX_AC_RO	WR_EX_RO	Reserved	EX_AC	Reserved	WR_EX	Reserved
5	Reserved							EX_AC_AR

A Write Exclusive – All Registrants (WR_EX_AR) bit set to one indicates that the device server supports the Write Exclusive – All Registrants persistent reservation type. An WR_EX_AR bit set to zero indicates that the device server does not support the Write Exclusive – All Registrants persistent reservation type.

An Exclusive Access – Registrants Only (EX_AC_RO) bit set to one indicates that the device server supports the Exclusive Access – Registrants Only persistent reservation type. An EX_AC_RO bit set to zero indicates that the device server does not support the Exclusive Access – Registrants Only persistent reservation type.

A Write Exclusive – Registrants Only (WR_EX_RO) bit set to one indicates that the device server supports the Write Exclusive – Registrants Only persistent reservation type. An WR_EX_RO bit set to zero indicates that the device server does not support the Write Exclusive – Registrants Only persistent reservation type.

An Exclusive Access (EX_AC) bit set to one indicates that the device server supports the Exclusive Access persistent reservation type. An EX_AC bit set to zero indicates that the device server does not support the Exclusive Access persistent reservation type.

A Write Exclusive (WR_EX) bit set to one indicates that the device server supports the Write Exclusive persistent reservation type. An WR_EX bit set to zero indicates that the device server does not support the Write Exclusive persistent reservation type.

An Exclusive Access – All Registrants (EX_AC_AR) bit set to one indicates that the device server supports the Exclusive Access – All Registrants persistent reservation type. An EX_AC_AR bit set to zero indicates that the device server does not support the Exclusive Access – All Registrants persistent reservation type.

6.15.5 READ FULL STATUS service action

The READ FULL STATUS service action requests the device server to return a parameter list describing the registration status and persistent reservation status of each currently registered I_T nexus for the logical unit.

For more information on READ FULL STATUS see 5.12.6.4.

The format for the parameter data provided in response to a PERSISTENT RESERVE IN command with the READ FULL STATUS service action is shown in table 213.

Table 213 — PERSISTENT RESERVE IN parameter data for READ FULL STATUS

Bit / Byte	7	6	5	4	3	2	1	0
0	(MSB)							
•••				PRGENERATION				
3								(LSB)
4	(MSB)							
•••				ADDITIONAL LENGTH (n-7)				
7								(LSB)
	Full status descriptors							
8								
•••				Full status descriptor (see table 214) [first]				
				⋮				
•••				Full status descriptor (see table 214) [last]				
n								

The PRGENERATION field shall be as defined for the PERSISTENT RESERVE IN command with READ KEYS service action parameter data (see 6.15.2).

The ADDITIONAL LENGTH field indicates the number of bytes that follow in the full status descriptors. The contents of the ADDITIONAL LENGTH field are not altered based on the allocation length (see 4.2.5.6).

The format of the full status descriptors is shown in table 214. Each full status descriptor describes one or more registered I_T nexuses. The device server shall return persistent reservations status information for every registered I_T nexus.

Table 214 — PERSISTENT RESERVE IN full status descriptor format

Bit / Byte	7	6	5	4	3	2	1	0
0	(MSB)							
•••				RESERVATION KEY				
7								(LSB)
8								
•••				Reserved				
11								
12				Reserved			ALL_TG_PT	R_HOLDER
13	SCOPE				TYPE			
14								
•••				Reserved				
17								
18	(MSB)							
19				RELATIVE TARGET PORT IDENTIFIER				(LSB)
20	(MSB)							
•••				ADDITIONAL DESCRIPTOR LENGTH (n-23)				
23								(LSB)
24								
•••				TransportID				
n								

The RESERVATION KEY field contains the reservation key.

A Reservation Holder (R_HOLDER) bit set to one indicates that all I_T nexuses described by this full status descriptor are registered and are persistent reservation holders (see 5.12.10). An R_HOLDER bit set to zero indicates that all I_T nexuses described by this full status descriptor are registered but are not persistent reservation holders.

An All Target Ports (ALL_TG_PT) bit set to zero indicates that this full status descriptor represents a single I_T nexus. An ALL_TG_PT bit set to one indicates that:

a) this full status descriptor represents all the I_T nexuses that are associated with both:
   A) the initiator port specified by the TransportID; and
   B) every target port in the SCSI target device;
b) all the I_T nexuses are registered with the same reservation key; and
c) all the I_T nexuses are either reservation holders or not reservation holders as indicated by the R_HOLDER bit.

The device server is not required to return a full status descriptor with the ALL_TG_PT bit set to one. Instead, it may return separate full status descriptors for each I_T nexus.

If the R_HOLDER bit is set to one (i.e., if the I_T nexus described by this full status descriptor is a reservation holder), then the SCOPE field and the TYPE field are as defined in the READ RESERVATION service action parameter data (see 6.15.3). If the R_HOLDER bit is set to zero, the contents of the SCOPE field and the TYPE field are not defined by this standard.

If the ALL_TG_PT bit is set to zero, the RELATIVE TARGET PORT IDENTIFIER field (see 4.3.4) is set to the relative port identifier of the target port that is part of the I_T nexus described by this full status descriptor. If the ALL_TG_PT bit is set to one, the contents of the RELATIVE TARGET PORT IDENTIFIER field are not defined by this standard.

The ADDITIONAL DESCRIPTOR LENGTH field indicates the number of bytes that follow in the descriptor (i.e., the size of the TransportID).

The TransportID specifies a TransportID (see 7.6.4) identifying the initiator port that is part of the I_T nexus or I_T nexuses described by this full status descriptor.

6.16 PERSISTENT RESERVE OUT command

6.16.1 PERSISTENT RESERVE OUT command introduction

The PERSISTENT RESERVE OUT command (see table 215) requests the device server to perform a service action that adds, removes, or manages a persistent reservation for the logical unit for the exclusive or shared use of one or more I_T nexuses.

I_T nexuses performing PERSISTENT RESERVE OUT service actions are identified by a registered reservation key provided by the application client. An application client may use the PERSISTENT RESERVE IN command to obtain the reservation key, if any, for the I_T nexus holding a persistent reservation and may use the PERSISTENT RESERVE OUT command to preempt that persistent reservation.

Table 215 — 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								
5	(MSB)							
•••	PARAMETER LIST LENGTH							
8								(LSB)
9	CONTROL							

The OPERATION CODE field is defined in 4.2.5.1 and shall be set as shown in table 215 for the PERSISTENT RESERVE OUT command.

The SERVICE ACTION field is defined in 4.2.5.2 and 6.16.2.

If a PERSISTENT RESERVE OUT command is attempted, but there are insufficient device server resources to complete the operation, the device server shall terminate the command with CHECK CONDITION status, with the sense key set to ILLEGAL REQUEST, and the additional sense code 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 SCOPE field and TYPE field are defined in 6.15.3.2 and 6.15.3.3. If a SCOPE field specifies a

scope that is not implemented, the device server shall terminate the command with CHECK CONDITION status, with the sense key set to ILLEGAL REQUEST, and the additional sense code 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 LENGTH field specifies the number of bytes of parameter data for the PERSISTENT RESERVE OUT command.

The parameter list shall be 24 bytes in length and the PARAMETER LIST LENGTH field shall contain 24 (i.e., 18h), if the following conditions are true:

a) the SPEC_I_PT bit (see 6.16.3) is set to zero; and
b) the service action is not REGISTER AND MOVE.

If the SPEC_I_PT bit is set to zero, the service action is not REGISTER AND MOVE, and the parameter list length is not 24, then the device server shall terminate the command with CHECK CONDITION status, with the sense key set to ILLEGAL REQUEST, and the additional sense code set to PARAMETER LIST LENGTH ERROR.

If the parameter list length is larger than the device server is able to process, the device server should terminate the command with CHECK CONDITION status, with the sense key set to ILLEGAL REQUEST, and the additional sense code set to PARAMETER LIST LENGTH ERROR.

The CONTROL byte is defined in SAM-5.

6.16.2 PERSISTENT RESERVE OUT service actions and parameter list formats

As part of processing the PERSISTENT RESERVE OUT service actions, the device server shall increment the PRgeneration value as defined in table 216.

The PERSISTENT RESERVE OUT command service actions are defined in table 216.

Table 216 — PERSISTENT RESERVE OUT service action codes

Code	Name	Description	PRGENERATION field incremented	Parameter list format
00h	REGISTER	Register a reservation key with the device server (see 5.12.7) or unregister a reservation key (see 5.12.11.2.3).	Yes	Basic (see 6.16.3)
01h	RESERVE	Creates a persistent reservation having a specified SCOPE and TYPE (see 5.12.9). The SCOPE and TYPE of a persistent reservation are defined in 6.15.3.2 and 6.15.3.3.	No	Basic (see 6.16.3)
02h	RELEASE	Releases the selected persistent reservation (see 5.12.11.2.2).	No	Basic (see 6.16.3)
03h	CLEAR	Clears all reservation keys (i.e., registrations) and all persistent reservations (see 5.12.11.2.7).	Yes	Basic (see 6.16.3)
04h	PREEMPT	Preempts persistent reservations and/or removes registrations (see 5.12.11.2.4).	Yes	Basic (see 6.16.3)
05h	PREEMPT AND ABORT	Preempts persistent reservations and/or removes registrations and aborts all commands for all preempted I_T nexuses (see 5.12.11.2.4 and 5.12.11.2.6).	Yes	Basic (see 6.16.3)
06h	REGISTER AND IGNORE EXISTING KEY	Register a reservation key with the device server (see 5.12.7) or unregister a reservation key (see 5.12.11.2.3).	Yes	Basic (see 6.16.3)
07h	REGISTER AND MOVE	Register a reservation key for another I_T nexus with the device server and move an existing persistent reservation to that I_T nexus (see 5.12.8)	Yes	Register and move (see 6.16.4)
08h	REPLACE LOST RESERVATION	Replace lost persistent reservation information (see 5.12.11.3).	No [a]	Basic (see 6.16.3)
09h to 1Fh	Reserved			

[a] The processing of a REPLACE LOST RESERVATION service action does not increment the PRGENERATION field. Instead, the PRGENERATION field is set to zero (see 5.12.11.3).

Table 217 summarizes which fields are set by the application client and interpreted by the device server for each service action and scope value.

Table 217 — PERSISTENT RESERVE OUT service actions and valid parameters (part 1 of 2)

Service action	Allowed SCOPE	Parameters (part 1 of 2)				
		Parameter list format	TYPE	RESERVATION KEY	SERVICE ACTION RESERVATION KEY	APTPL
REGISTER	ignored	Basic (see 6.16.3)	ignored	valid	valid	valid
REGISTER AND IGNORE EXISTING KEY	ignored	Basic (see 6.16.3)	ignored	ignored	valid	valid
RESERVE	LU_SCOPE	Basic (see 6.16.3)	valid	valid	ignored	ignored
RELEASE	LU_SCOPE	Basic (see 6.16.3)	valid	valid	ignored	ignored
CLEAR	ignored	Basic (see 6.16.3)	ignored	valid	ignored	ignored
PREEMPT	LU_SCOPE	Basic (see 6.16.3)	valid	valid	valid	ignored
PREEMPT AND ABORT	LU_SCOPE	Basic (see 6.16.3)	valid	valid	valid	ignored
REGISTER AND MOVE	LU_SCOPE	Register and move (see 6.16.4)	valid	valid	valid	valid
REPLACE LOST RESERVATION	LU_SCOPE	Basic (see 6.16.3)	valid	valid	valid	ignored

Table 217 — PERSISTENT RESERVE OUT service actions and valid parameters (part 2 of 2)

Service action	Allowed SCOPE	Parameters (part 2 of 2)			
		Parameter list format	ALL_TG_PT	SPEC_I_PT	UNREG
REGISTER	ignored	Basic (see 6.16.3)	valid	valid	n/a
REGISTER AND IGNORE EXISTING KEY	ignored	Basic (see 6.16.3)	valid	invalid	n/a
RESERVE	LU_SCOPE	Basic (see 6.16.3)	ignored	invalid	n/a
RELEASE	LU_SCOPE	Basic (see 6.16.3)	ignored	invalid	n/a
CLEAR	ignored	Basic (see 6.16.3)	ignored	invalid	n/a
PREEMPT	LU_SCOPE	Basic (see 6.16.3)	ignored	invalid	n/a
PREEMPT AND ABORT	LU_SCOPE	Basic (see 6.16.3)	ignored	invalid	n/a
REGISTER AND MOVE	LU_SCOPE	Register and move (see 6.16.4)	n/a	n/a	valid
REPLACE LOST RESERVATION	LU_SCOPE	Basic (see 6.16.3)	invalid	invalid	n/a

6.16.3 Basic PERSISTENT RESERVE OUT parameter list

The parameter list format shown in table 218 shall be used by the PERSISTENT RESERVE OUT command with all service actions except the REGISTER AND MOVE service action. All fields shall be sent, even if the field is not required for the specified service action and scope values.

Table 218 — 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								
•••				Obsolete				
19								
20	Reserved				SPEC_I_PT	ALL_TG_PT	Reserved	APTPL
21	Reserved							
22								
23				Obsolete				
24								
•••				Additional parameter data				
n								

The obsolete fields in bytes 16 to 19, byte 22 and byte 23 were defined in SPC-2.

The RESERVATION KEY field contains an eight byte value provided by the application client to the device server to identify the I_T nexus 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 I_T nexus from which the command was received, except for:

a) the REGISTER AND IGNORE EXISTING KEY service action where the RESERVATION KEY field shall be ignored;
b) the REGISTER service action for an unregistered I_T nexus where the RESERVATION KEY field shall contain zero; and
c) the REPLACE LOST RESERVATION service action where the RESERVATION KEY field shall contain zero.

Except as noted in this subclause, if a PERSISTENT RESERVE OUT command specifies a RESERVATION KEY field other than the reservation key registered for the I_T nexus, then the device server shall complete the command with RESERVATION CONFLICT status. Except as noted in this subclause, the reservation key of the I_T nexus shall be verified to be correct regardless of the SERVICE ACTION and SCOPE field values.

The SERVICE ACTION RESERVATION KEY field contains information associated with the following service actions:

a) REGISTER;
b) REGISTER AND IGNORE EXISTING KEY;
c) PREEMPT;

d) PREEMPT AND ABORT; and
e) REPLACE LOST RESERVATION.

The SERVICE ACTION RESERVATION KEY field is ignored for the following service actions:

a) RESERVE;
b) RELEASE; and
c) CLEAR.

For the REGISTER service action and REGISTER AND IGNORE EXISTING KEY service action, the SERVICE ACTION RESERVATION KEY field contains:

a) the new reservation key to be registered in place of the registered reservation key; or
b) zero to unregister the registered reservation key.

For the PREEMPT service action and PREEMPT AND ABORT service action, the SERVICE ACTION RESERVATION KEY field contains the reservation key of:

a) the registrations to be removed; and
b) if the SERVICE ACTION RESERVATION KEY field identifies a persistent reservation holder (see 5.12.10), then the persistent reservations to be preempted.

For the REPLACE LOST RESERVATION service action, the SERVICE ACTION RESERVATION KEY field contains the new reservation key to be registered

If the Specify Initiator Ports (SPEC_I_PT) bit is set to zero, the device server shall apply the registration only to the I_T nexus that sent the PERSISTENT RESERVE OUT command. If the SPEC_I_PT bit is set to one for any service action except the REGISTER service action, then the device server shall terminate the command with CHECK CONDITION status, with the sense key set to ILLEGAL REQUEST, and the additional sense code set to INVALID FIELD IN PARAMETER LIST. If the SPEC_I_PT bit is set to one for the REGISTER service action, then the additional parameter data (see table 219) shall include a list of transport IDs and the device server shall also apply the registration to the I_T nexus for each initiator port specified by a TransportID. If a registration fails for any initiator port (e.g., if the logical unit does not have enough resources available to hold the registration information), then no registrations shall be made and the command shall be terminated with CHECK CONDITION status.

Table 219 — PERSISTENT RESERVE OUT specify initiator ports additional parameter data

Bit / Byte	7	6	5	4	3	2	1	0
24	TRANSPORTID PARAMETER DATA LENGTH (n-27)							
•••								
27								
	TransportIDs list							
28	TransportID [first]							
•••								
	⋮							
•••	TransportID [last]							
n								

The TRANSPORTID PARAMETER DATA LENGTH field specifies the number of bytes of TransportIDs that follow.

If:

a) the value in the PARAMETER LIST LENGTH field in the CDB does not include all of the additional parameter list bytes specified by the TRANSPORTID PARAMETER DATA LENGTH field; or
b) the value in the TRANSPORTID PARAMETER DATA LENGTH field results in the truncation of a TransportID,

then the device server:

a) shall terminate the command with CHECK CONDITION status, with the sense key set to ILLEGAL REQUEST; and
b) should set the additional sense code to PARAMETER LIST LENGTH ERROR.

The format of a TransportID is defined in 7.6.4.

The All Target Ports (ALL_TG_PT) bit is valid only for the REGISTER service action and the REGISTER AND IGNORE EXISTING KEY service action, and shall be ignored for all other service actions. If the device server receives a REGISTER service action or a REGISTER AND IGNORE EXISTING KEY service action with the ALL_TG_PT bit set to one, then the device server shall create the specified registration on all target ports in the SCSI target device known to the device server (i.e., as if the same registration request had been received individually through each target port). If the device server receives a REGISTER service action or a REGISTER AND IGNORE EXISTING KEY service action with the ALL_TG_PT bit set to zero, then the device server shall apply the registration only to the target port through which the PERSISTENT RESERVE OUT command was received. If a device server that does not support an ALL_TG_PT bit set to one receives that value in a REGISTER service action or a REGISTER AND IGNORE EXISTING KEY service action, then the device server shall terminate the command with CHECK CONDITION status, with the sense key set to ILLEGAL REQUEST, and the additional sense code set to INVALID FIELD IN PARAMETER LIST.

The Activate Persist Through Power Loss (APTPL) bit is:

a) valid only for the REGISTER service action and REGISTER AND IGNORE EXISTING KEY service action; and
b) shall be ignored for all other service actions.

If a device server that does not support an APTPL bit set to one detects that value in a REGISTER service action or a REGISTER AND IGNORE EXISTING KEY service action, then the device server shall terminate the command with CHECK CONDITION status, with the sense key set to ILLEGAL REQUEST, and the additional sense code set to INVALID FIELD IN PARAMETER LIST.

If the last valid APTPL bit value processed by the device server is zero, the loss of power in the SCSI target device shall release the persistent reservation for the logical unit and remove all registered reservation keys (see 5.12.7). If the last valid APTPL bit value processed by the device server is one, the logical unit shall retain any persistent reservation(s) that may be present and all reservation keys (i.e., registrations) for all I_T nexuses even if power is lost (see 5.12.5).

6.16.4 Parameter list for the PERSISTENT RESERVE OUT command with REGISTER AND MOVE service action

The parameter list format shown in table 220 shall be used by the PERSISTENT RESERVE OUT command with REGISTER AND MOVE service action.

Table 220 — PERSISTENT RESERVE OUT with REGISTER AND MOVE service action 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				Reserved				
17				Reserved			UNREG	APTPL
18	(MSB)							
19				RELATIVE TARGET PORT IDENTIFIER				(LSB)
20	(MSB)							
•••				TRANSPORTID LENGTH (n-23)				
23								(LSB)
24								
•••				TransportID				
n								

The RESERVATION KEY field contains an eight byte value provided by the application client to the device server to identify the I_T nexus 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 I_T nexus from which the command was received. If a PERSISTENT RESERVE OUT command specifies a RESERVATION KEY field other than the reservation key registered for the I_T nexus, then the device server shall complete the command with RESERVATION CONFLICT status.

The SERVICE ACTION RESERVATION KEY field contains the reservation key to be registered to the specified I_T nexus.

If a device server that does not support an Activate Persist Through Power Loss (APTPL) bit set to one detects that value in a REGISTER AND MOVE service action, then the device server shall terminate the command with CHECK CONDITION status, with the sense key set to ILLEGAL REQUEST, and the additional sense code set to INVALID FIELD IN PARAMETER LIST.

If the last valid APTPL bit value processed by the device server is zero, the loss of power in the SCSI target device shall release the persistent reservation for the logical unit and remove all registered reservation keys (see 5.6.5). If the last valid APTPL bit value processed by the device server is one, the logical unit shall retain any persistent reservation(s) that may be present and all reservation keys (i.e., registrations) for all I_T nexuses even if power is lost (see 5.12.5).

An unregister (UNREG) bit set to zero specifies that the device server shall not unregister the I_T nexus on which the PERSISTENT RESERVE OUT command REGISTER AND MOVE service action was received. An

UNREG bit set to one specifies that the device server shall unregister the I_T nexus on which the PERSISTENT RESERVE OUT command REGISTER AND MOVE service action was received.

The RELATIVE TARGET PORT IDENTIFIER field (see 4.3.4) specifies the relative port identifier of the target port in the I_T nexus to which the persistent reservation is to be moved.

The TRANSPORTID LENGTH field specifies the number of bytes of the TransportID that follow, shall be a minimum of 24 bytes, and shall be a multiple of four.

The TransportID specifies the initiator port in the I_T nexus to which the persistent reservation is to be moved. The format of the TransportID is defined in 7.6.4.

If:

a) the value in the PARAMETER LIST LENGTH field in the CDB does not include all of the parameter list bytes specified by the TRANSPORTID LENGTH field; or
b) the value in the TRANSPORTID LENGTH field results in the truncation of a TransportID;

then the device server:

a) shall terminate the command with CHECK CONDITION status, with the sense key set to ILLEGAL REQUEST; and
b) should set the additional sense code to PARAMETER LIST LENGTH ERROR.

6.17 READ ATTRIBUTE command

6.17.1 READ ATTRIBUTE command introduction

The READ ATTRIBUTE command (see table 221) requests the device server to read attribute information from medium auxiliary memory.

Table 221 — READ ATTRIBUTE command

Bit / Byte	7	6	5	4	3	2	1	0
0	OPERATION CODE (8Ch)							
1	Reserved			SERVICE ACTION				
2	Restricted (see SMC-3)							
•••								
4								
5	LOGICAL VOLUME NUMBER							
6	Reserved							
7	PARTITION NUMBER							
8	(MSB) FIRST ATTRIBUTE IDENTIFIER							
9								(LSB)
10	(MSB) ALLOCATION LENGTH							
•••								
13								(LSB)
14	Reserved							CACHE
15	CONTROL							

The OPERATION CODE field is defined in 4.2.5.1 and shall be set as shown in table 221 for the READ ATTRIBUTE command.

The SERVICE ACTION field is defined in 4.2.5.2. The service action codes defined for the READ ATTRIBUTE command are shown in table 222.

Table 222 — READ ATTRIBUTE service action codes

Code	Name	Description	Reference
00h	ATTRIBUTE VALUES	Return attribute values.	6.17.2
01h	ATTRIBUTE LIST	Return a list of available attribute identifiers, identifiers that are in the read only state or in the read/write state (see 5.7).	6.17.3
02h	LOGICAL VOLUME LIST	Return a list of known logical volume numbers.	6.17.4
03h	PARTITION LIST	Return a list of known partition numbers.	6.17.5
04h	Restricted		SMC-3
05h	SUPPORTED ATTRIBUTES	Return a list of supported attribute identifiers, identifiers that are in the read only state, in the read/write state, or in the nonexistent state (see 5.7).	6.17.6
06h to 1Fh	Reserved		

The LOGICAL VOLUME NUMBER field specifies a logical volume (e.g., the medium auxiliary memory storage for one side of a double sided medium) within the medium auxiliary memory. The number of logical volumes of the medium auxiliary memory shall equal the number of logical volumes of the attached medium. If the medium only has a single logical volume, then its logical volume number shall be zero.

The PARTITION NUMBER field specifies a partition (see SSC-4) within a logical volume. The number of partitions of the medium auxiliary memory shall equal the number of partitions of the attached medium. If the medium only has a single partition, then its partition number shall be zero.

If the combination of logical volume number and partition number is not valid, the device server shall terminate the command with CHECK CONDITION status, with the sense key set to ILLEGAL REQUEST, and the additional sense code set to INVALID FIELD IN CDB.

The FIRST ATTRIBUTE IDENTIFIER field specifies the attribute identifier of the first attribute to be returned. If the specified attribute is in the unsupported state or nonexistent state (see 5.7), the device server shall terminate the READ ATTRIBUTE command with CHECK CONDITION status, with the sense key set to ILLEGAL REQUEST, and the additional sense code set to INVALID FIELD IN CDB.

The ALLOCATION LENGTH field is defined in 4.2.5.6. that the device server shall

Caching of either a subset or the complete set of attribute information from the most recently mounted medium may be supported as cached attribute information. If cached attribute information is supported the device server shall:

a) support the CACHE bit being set to one; and
b) clear cached attribute information at the start of a medium load.

A CACHE bit set to one specifies that the device server may return cached attribute information. A CACHE bit set to zero specifies that the device server shall not return cached attribute information.

If medium auxiliary memory is not accessible, the device server shall return the status defined in table 223.

Table 223 — Status to be returned if medium auxiliary memory is not accessible

CACHE bit	Medium state	Cached attribute information	Status	Sense Key	Additional sense code
0b	present	ignored	CHECK CONDITION	NOT READY	MEDIUM NOT PRESENT
	not present	ignored	CHECK CONDITION	MEDIUM ERROR	LOGICAL UNIT NOT READY, AUXILIARY MEMORY NOT ACCESSIBLE
1b	present	available	GOOD	NO SENSE	NO ADDITIONAL SENSE INFORMATION
		not available	CHECK CONDITION	NOT READY	MEDIUM NOT PRESENT
	not present	available	GOOD	NO SENSE	NO ADDITIONAL SENSE INFORMATION
		not available	CHECK CONDITION	MEDIUM ERROR	LOGICAL UNIT NOT READY, AUXILIARY MEMORY NOT ACCESSIBLE

If the medium auxiliary memory is not operational, the device server shall terminate the READ ATTRIBUTE command with CHECK CONDITION status, with the sense key set to MEDIUM ERROR, and the additional sense code set to AUXILIARY MEMORY READ ERROR.

If a medium is mounted and the medium auxiliary memory is accessible, the CACHE bit shall be ignored.

The CONTROL byte is defined in SAM-5.

The format of parameter data returned by the READ ATTRIBUTE command depends on the service action specified.

6.17.2 ATTRIBUTE VALUES service action

The READ ATTRIBUTE command with ATTRIBUTE VALUES service action returns parameter data containing the attributes that are in the read only state or read/write state (see 5.7) specified by the PARTITION NUMBER field, LOGICAL VOLUME NUMBER field, and FIRST ATTRIBUTE IDENTIFIER field in the CDB. The parameter data shall contain the requested attributes in ascending numerical order by attribute identifier value and in the format shown in table 224.

Table 224 — READ ATTRIBUTE with ATTRIBUTE VALUES service action parameter data format

Bit / Byte	7	6	5	4	3	2	1	0
0	(MSB)							
•••	AVAILABLE DATA (n-3)							
3								(LSB)
	Attribute(s)							
4								
•••	Attribute (see 7.4.1) [first]							
	⋮							
•••	Attribute (see 7.4.1) [last]							
n								

The AVAILABLE DATA field shall contain the number of bytes of attribute information in the parameter data. The contents of the AVAILABLE DATA field are not altered based on the allocation length (see 4.2.5.6).

The format of the attributes is described in 7.4.1.

6.17.3 ATTRIBUTE LIST service action

The READ ATTRIBUTE command with ATTRIBUTE LIST service action returns parameter data containing the attribute identifiers for the attributes that are in the read only state or in the read/write state (see 5.7) in the specified partition number and volume number. The contents of FIRST ATTRIBUTE IDENTIFIER field in the CDB shall be ignored. The parameter data shall contain the requested attribute identifiers in ascending numerical order by attribute identifier value and in the format shown in table 225.

Table 225 — READ ATTRIBUTE with ATTRIBUTE LIST service action parameter data format

Bit / Byte	7	6	5	4	3	2	1	0
0	(MSB)							
•••				AVAILABLE DATA (n-3)				
3								(LSB)
	Attribute identifiers							
4	(MSB)							
5				ATTRIBUTE IDENTIFIER [first]				(LSB)
				⋮				
n-1	(MSB)							
n				ATTRIBUTE IDENTIFIER [last]				(LSB)

The AVAILABLE DATA field shall contain the number of bytes of attribute identifiers in the parameter data. The contents of the AVAILABLE DATA field are not altered based on the allocation length (see 4.2.5.6).

An ATTRIBUTE IDENTIFIER field is returned for each attribute that is in the read only state or in the read/write state (see 5.7) in the specified partition number and volume number. See 7.4.2 for a description of the attribute identifier values.

6.17.4 LOGICAL VOLUME LIST service action

The READ ATTRIBUTE command with LOGICAL VOLUME LIST service action returns parameter data (see table 226) identifying the supported number of logical volumes. The contents of LOGICAL VOLUME NUMBER field, PARTITION NUMBER field, and FIRST ATTRIBUTE IDENTIFIER field in the CDB shall be ignored.

Table 226 — READ ATTRIBUTE with LOGICAL VOLUME LIST service action parameter data format

Bit / Byte	7	6	5	4	3	2	1	0
0	(MSB)							
1				AVAILABLE DATA (0002h)				(LSB)
2	FIRST LOGICAL VOLUME NUMBER							
3	NUMBER OF LOGICAL VOLUMES AVAILABLE							

The AVAILABLE DATA field shall set as shown in table 226 for the parameter data returned by a READ ATTRIBUTE command with LOGICAL VOLUME LIST service action. The contents of the AVAILABLE DATA field are not altered based on the allocation length (see 4.2.5.6).

The FIRST LOGICAL VOLUME NUMBER field indicates the first volume available. Logical volume numbering should start at zero.

The NUMBER OF LOGICAL VOLUMES AVAILABLE field indicates the number of volumes available.

6.17.5 PARTITION LIST service action

The READ ATTRIBUTE command with PARTITION LIST service action returns parameter data (see table 227) identifying the number of partitions supported in the specified logical volume number. The contents of PARTITION NUMBER field and FIRST ATTRIBUTE IDENTIFIER field in the CDB shall be ignored.

Table 227 — READ ATTRIBUTE with PARTITION LIST service action parameter data format

Bit / Byte	7	6	5	4	3	2	1	0
0	(MSB)			AVAILABLE DATA (0002h)				
1								(LSB)
2	FIRST PARTITION NUMBER							
3	NUMBER OF PARTITIONS AVAILABLE							

The AVAILABLE DATA field shall set as shown in table 227 for the parameter data returned by a READ ATTRIBUTE command with PARTITION LIST service action. The contents of the AVAILABLE DATA field are not altered based on the allocation length (see 4.2.5.6).

The FIRST PARTITION NUMBER field indicates the first partition available on the specified logical volume number. Partition numbering should start at zero.

The NUMBER OF PARTITIONS AVAILABLE field indicates the number of partitions available on the specified logical volume number.

6.17.6 SUPPORTED ATTRIBUTES service action

The READ ATTRIBUTE command with SUPPORTED ATTRIBUTES service action returns parameter data containing the attribute identifiers for the attributes that are supported by the device server in the specified partition number and volume number. The contents of FIRST ATTRIBUTE IDENTIFIER field in the CDB shall be ignored. The parameter data shall contain the requested attribute identifiers in ascending numerical order by attribute identifier value and in the format shown in table 228.

Table 228 — READ ATTRIBUTE with SUPPORTED ATTRIBUTES service action parameter data format

Bit / Byte	7	6	5	4	3	2	1	0
0	(MSB)			AVAILABLE DATA (n-3)				
•••								
3								(LSB)
	Attribute identifiers							
4	(MSB)			ATTRIBUTE IDENTIFIER [first]				
5								(LSB)
	⋮							
n-1	(MSB)			ATTRIBUTE IDENTIFIER [last]				
n								(LSB)

The AVAILABLE DATA field shall contain the number of bytes of attribute identifiers in the parameter data. The contents of the AVAILABLE DATA field are not altered based on the allocation length (see 4.2.5.6).

For each attribute that is in the read only state, in the read/write state, or in the nonexistent state (see 5.7) in the specified partition number and volume number:

a) an attribute identifier (see 7.4.2) is returned for device type attributes (see 7.4.2.2), medium type attributes (see 7.4.2.3), and host type attributes defined in the applicable command standards; and
b) if vendor specific host type attributes (see 7.4.2.1) are supported, then an attribute identifier is returned for the first supported vendor specific host type attribute and attribute identifiers may be returned for other supported vendor specific host type attributes.

6.18 READ BUFFER command

6.18.1 READ BUFFER command introduction

The READ BUFFER command (see table 229) is used in conjunction with the WRITE BUFFER command for:

a) testing logical unit buffer memory;
b) testing the integrity of the service delivery subsystem;
c) downloading microcode (see 5.4); and
d) retrieving error history.

Table 229 — 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 OPERATION CODE field is defined in 4.2.5.1 and shall be set as shown in table 229 for the READ BUFFER command.

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 230.

Table 230 — READ BUFFER MODE field

Code	Description	References
00h	Obsolete	
01h	Vendor specific	6.18.2
02h	Data	6.18.3
03h	Descriptor	6.18.4
04h to 09h	Reserved	
0Ah	Read data from echo buffer	6.18.5
0Bh	Echo buffer descriptor	6.18.6
0Ch to 19h	Reserved	
1Ah	Obsolete	
1Bh	Reserved	
1Ch	Error history	5.5 and 6.18.7
1Dh to 1Fh	Reserved	

The MODE field may be processed as specifying a service action by the REPORT SUPPORTED OPERATION CODES command (see 6.35).

If the MODE field is not set to 01h (i.e., vendor specific), the ALLOCATION LENGTH field is defined in 4.2.5.6.

The CONTROL byte is defined in SAM-5.

6.18.2 Vendor specific mode (01h)

In this mode, the meanings of the BUFFER ID field, BUFFER OFFSET field, and ALLOCATION LENGTH field are not specified by this standard.

6.18.3 Data mode (02h)

In this mode, the device server returns parameter data that contains logical unit buffer data. The BUFFER ID field specifies a buffer within the logical unit from which data shall be transferred. The manufacturer assigns buffer ID codes to buffers within the logical unit. Buffer ID zero shall be supported. If more than one buffer is supported, then additional buffer ID codes shall be assigned contiguously, beginning with one. Buffer ID code assignments for the READ BUFFER command with data mode shall be the same as for the WRITE BUFFER command with data mode (see 6.49.3). If an unsupported buffer ID code is selected, the device server shall terminate the command with CHECK CONDITION status, with the sense key set to ILLEGAL REQUEST, and the additional sense code set to INVALID FIELD IN CDB.

The BUFFER OFFSET field specifies 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 6.18.4). If the device server is unable to accept the specified buffer offset, the device server shall terminate the command with CHECK CONDITION status, with the sense key set to ILLEGAL REQUEST, and the additional sense code set to INVALID FIELD IN CDB.

NOTE 33 - The buffer is shared by all application clients and I_T nexuses. If one application client writes the buffer, a second application client writes the buffer, and then the first application client reads the buffer, then the read may or may not retrieve the data from the second application client.

6.18.4 Descriptor mode (03h)

In this mode, the device server returns parameter data that contains a maximum of four bytes of READ BUFFER descriptor information. The BUFFER OFFSET field is reserved in this mode. The allocation length should be set to four or greater. The READ BUFFER descriptor is defined as shown in table 231.

Table 231 — READ BUFFER descriptor

Bit / Byte	7	6	5	4	3	2	1	0
0	OFFSET BOUNDARY							
1	(MSB)							
2	BUFFER CAPACITY							
3								(LSB)

For READ BUFFER commands, the OFFSET BOUNDARY field (see table 232) applies to the following modes:

a) data (i.e., 02h) (see 6.18.3); and
b) error history (i.e., 1Ch) (see 6.18.7).

For WRITE BUFFER commands, the OFFSET BOUNDARY field (see table 232) applies to the following modes:

a) data (i.e., 02h) (see 6.49.3);
b) download microcode with offsets and activate (i.e., 06h) (see 6.49.6);
c) download microcode with offsets, save, and activate (i.e., 07h) (see 6.49.7);
d) download microcode with offsets, select activation events, save, and defer activate (i.e., 0Dh) (see 6.49.9); and
e) download microcode with offsets, save, and defer activate (i.e., 0Eh) (see 6.49.10).

For data mode (i.e., 02h), the boundary alignment indicated by the OFFSET BOUNDARY field applies only to the buffer specified by the BUFFER ID field. For modes other than data to which the OFFSET BOUNDARY field applies, the boundary alignment applies regardless of the buffer specified by the BUFFER ID field.

Table 232 — OFFSET BOUNDARY field

Code	Description
00h to FEh	Multiples of 2^{code} (e.g., 00h means multiples of 1 byte or no offset restrictions, 01h means multiples of 2 bytes or even offsets, 02h means multiples of 4 bytes)
FFh	000000h is the only supported buffer offset

The BUFFER CAPACITY field indicates the maximum size in bytes of the buffer specified by the BUFFER ID field for:

a) the READ BUFFER command with data mode (i.e., 02h); and
b) the WRITE BUFFER command with data mode (i.e., 02h).

6.18.5 Read data from echo buffer mode (0Ah)

In this mode, the device server returns parameter data that contains the echo buffer (i.e., parameter list data sent to the device server by the most recent WRITE BUFFER command with the MODE field set to write data to echo buffer (see 6.49.8) received on the same I_T nexus). The device server shall return the same number of bytes of data as was received during the processing of the prior WRITE BUFFER command with the MODE field set to write data to echo buffer, limited by the allocation length (see 4.2.5.6).

The BUFFER ID field and BUFFER OFFSET field are ignored in this mode.

If the device server has not received a WRITE BUFFER command with the mode set to write data to echo buffer received on this I_T nexus since the last logical unit reset condition (see SAM-5) and completed that command without an error, then the device server shall terminate the READ BUFFER command with CHECK CONDITION status, with the sense key set to ILLEGAL REQUEST, and the additional sense code set to COMMAND SEQUENCE ERROR. If the data in the echo buffer has been overwritten as the result of a WRITE BUFFER command received on another I_T nexus and the device server supports error reporting on echo buffer overwrites (i.e., the EBOS bit is set to one in the echo buffer descriptor (see 6.18.6)), then the device server shall terminate the READ BUFFER command with CHECK CONDITION status, with the sense key set to ABORTED COMMAND, and the additional sense code set to ECHO BUFFER OVERWRITTEN.

After a WRITE BUFFER command with the mode set to write data to echo buffer has completed without an error, the application client may send multiple READ BUFFER commands with the mode set to read data from echo buffer in order to read the echo buffer data multiple times.

6.18.6 Echo buffer descriptor mode (0Bh)

In this mode, the device server returns parameter data that contains a maximum of four bytes of READ BUFFER descriptor information for the echo buffer. If there is an echo buffer is not implemented, the device server shall return all zeros in the READ BUFFER descriptor. The BUFFER ID field and BUFFER OFFSET field are reserved in this mode. The allocation length should be set to four or greater. The echo buffer descriptor is defined as shown in table 233.

Table 233 — 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)

If the echo buffer is implemented, the echo buffer descriptor shall be implemented.

An echo buffer overwritten supported (EBOS) bit set to one indicates either:

a) the device server 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 I_T nexus, or
b) the device server ensures echo buffer data returned to each I_T nexus is the same as that previously written by that I_T nexus.

An EBOS bit set to zero specifies that the echo buffer may be overwritten by any intervening command received on any I_T nexus.

The BUFFER CAPACITY field shall contain the size of the echo buffer in bytes aligned to a four-byte boundary. The maximum echo buffer size is 4 096 bytes.

A READ BUFFER command with the mode set to echo buffer descriptor may be used to determine the echo buffer capacity and supported features before a WRITE BUFFER command with the mode set to write data to echo buffer (see 6.49.8) is sent.

6.18.7 Error history mode (1Ch)

6.18.7.1 Error history overview

This mode is used to manage and retrieve error history (see 5.5).

If the device server is unable to process a READ BUFFER command with the MODE field set to 1Ch, the device server shall terminate the READ BUFFER command with CHECK CONDITION status, with the sense key set to ILLEGAL REQUEST, and the additional sense code set to COMMAND SEQUENCE ERROR.

The BUFFER ID field (see table 234) specifies the action that the device server shall perform, and the parameter data, if any, that the device server shall return.

Table 234 — Error history BUFFER ID field

Code	Description	Buffer offset	Error history I_T nexus constrained	Reference
00h	Return error history directory	0000h	Yes	6.18.7.2
01h	Return error history directory and create new error history snapshot	0000h	Yes	6.18.7.2
02h	Return error history directory and establish new error history I_T nexus	0000h	No	6.18.7.2
03h	Return error history directory, establish new error history I_T nexus, and create new error history snapshot	0000h	No	6.18.7.2
04h to 0Fh	Reserved		Yes	
10h to EFh	Return error history	0000h to FFFFh	Yes	6.18.7.3
F0h to FDh	Reserved		Yes	
FEh	Clear error history I_T nexus	Ignored	Yes	6.18.7.4
FFh	Clear error history I_T nexus and release error history snapshot	Ignored	Yes	6.18.7.5

The device server shall terminate the READ BUFFER command with CHECK CONDITION status with the sense key set to ILLEGAL REQUEST and the additional sense code set to OPERATION IN PROGRESS if the device server receives a READ BUFFER command:

a) with the MODE field set to 1Ch;
b) with the BUFFER ID field set to a value that table 234 shows as constrained by error history I_T nexus;
c) if an error history I_T nexus exists and the command is received from an I_T nexus that is different than that I_T nexus; and
d) an error history snapshot exists.

The BUFFER OFFSET field specifies the byte offset from the start of the buffer specified by the BUFFER ID field from which the device server shall return data. The application client should conform to the offset boundary requirements indicated in the READ BUFFER descriptor (see 6.18.4). If the buffer offset is not one of those shown in table 234 or the device server is unable to accept the specified buffer offset, then the device server shall terminate the READ BUFFER command with CHECK CONDITION status, with the sense key set to ILLEGAL REQUEST, and the additional sense code set to INVALID FIELD IN CDB.

6.18.7.2 Error history directory

Whenever allowed by established error history I_T nexus constraints (see 6.18.7.1), if any, all error history directory device server actions return an error history directory (see table 236). Some error history directory device server actions also discard the existing error history snapshot and create a new error history snapshot (see table 235).

Table 235 — Summary of error history directory device server actions

BUFFER ID field	Establish new error history I_T nexus	Error history snapshot	
		Preserved (if exists)	Created
00h	No [a]	Yes	No [b]
01h	No [a]	No	Yes
02h	Yes	Yes	No [b]
03h	Yes	No	Yes

[a] If no error history I_T nexus is established, a new one is established.
[b] If no error history snapshot exists, a new one is created.

The error history directory is defined in table 236.

Table 236 — Error history directory

Bit / Byte	7	6	5	4	3	2	1	0
0	(MSB)							
•••				T10 VENDOR IDENTIFICATION				
7								(LSB)
8	VERSION							
9	Reserved				EHS_RETRIEVED	EHS_SOURCE		CLR_SUP
10								
•••				Reserved				
29								
30	(MSB)							
31				DIRECTORY LENGTH (n-31)				(LSB)
	Error history directory list							
32								
•••				Error history directory entry (see table 239) [first]				
39								
	⋮							
n-7								
•••				Error history directory entry (see table 239) [last]				
n								

The T10 VENDOR IDENTIFICATION field contains eight bytes of left-aligned ASCII data (see 4.3.1) identifying the manufacturer of the logical unit. The T10 vendor identification shall be one assigned by INCITS. A list of assigned T10 vendor identifications is in Annex G and on the T10 web site (http://www.t10.org).

NOTE 34 - The T10 VENDOR IDENTIFICATION field may contain a different value than the VENDOR IDENTIFICATION field in the standard INQUIRY data (see 6.6.2) (e.g., this field may indicate a disk drive component manufacturer while the standard INQUIRY data indicates the original equipment manufacturer).

The VERSION field indicates the version and format of the vendor specific error history. The VERSION field is assigned by the manufacturer indicated in the T10 VENDOR IDENTIFICATION field.

The error history retrieved (EHS_RETRIEVED) field (see table 237) indicates whether a clear error history device server action has been requested for the error history snapshot. EHS_RETRIEVED field shall be set to 00b or 10b when the error history snapshot is created.

Table 237 — EHS_RETRIEVED field

Code	Description
00b	No information
01b	The error history I_T nexus has requested buffer ID FEh (i.e., clear error history I_T nexus) or buffer ID FFh (i.e., clear error history I_T nexus and release snapshot) for the current error history snapshot.
10b	An error history I_T nexus has not requested buffer ID FEh (i.e., clear error history I_T nexus) or buffer ID FFh (i.e., clear error history I_T nexus and release snapshot) for the current error history snapshot.
11b	Reserved

The error history source (EHS_SOURCE) field (see table 238) indicates the source of the error history snapshot.

Table 238 — EHS_SOURCE field

Code	Description
00b	The error history snapshot was created by the device server and was not created due to processing a READ BUFFER command.
01b	Error history snapshot was created due to processing of the current READ BUFFER command
10b	Error history snapshot was created due to processing of a previous READ BUFFER command
11b	Reserved

A clear support (CLR_SUP) bit set to one indicates that the CLR bit is supported in the WRITE BUFFER command download error history mode (see 6.49.12). A CLR_SUP bit set to zero indicates that the CLR bit is not supported.

The DIRECTORY LENGTH field indicates the number of bytes that follow in the error history directory list. This value shall not be altered even if the allocation length is not sufficient to transfer the entire error history directory list.

The error history directory list contains an error history directory entry (see table 239) for each supported buffer ID in the range of 00h to EFh. The first entry shall be for buffer ID 00h and the entries shall be in order of ascending buffer IDs. The supported buffer IDs are not required to be contiguous. There shall not be any entries for buffer IDs greater than or equal to F0h.

Table 239 — Error history directory entry

Bit / Byte	7	6	5	4	3	2	1	0
0	SUPPORTED BUFFER ID							
1	Reserved							
•••								
3								
4	(MSB)							
•••	MAXIMUM AVAILABLE LENGTH							
7								(LSB)

The SUPPORTED BUFFER ID field indicates the error history buffer ID associated with this entry.

The MAXIMUM AVAILABLE LENGTH field indicates the maximum number of data bytes contained in the buffer indicated by the SUPPORTED BUFFER ID field. The actual number of bytes available for transfer may be smaller.

6.18.7.3 Error history data buffer

Unless an error is encountered, the device server shall return parameter data that contains error history in a vendor specific format from the error history snapshot from the specified buffer at the specified buffer offset.

If the device server receives a READ BUFFER command with the MODE field set to 1Ch from the established error history I_T nexus and the BUFFER ID field is set to a value that the error history directory (see 6.18.7.2) shows as not supported, then the device server shall terminate the command with CHECK CONDITION status with the sense key set to ILLEGAL REQUEST and the additional sense code set to INVALID FIELD IN CDB.

If the value in the BUFFER OFFSET field is not supported, the device server shall terminate the command with CHECK CONDITION status with the sense key set to ILLEGAL REQUEST and the additional sense code set to INVALID FIELD IN CDB.

The amount of error history in the specified buffer shall be less than or equal to the number of bytes indicated by the MAXIMUM AVAILABLE LENGTH field in the error history directory (see 6.18.7.2).

6.18.7.4 Clear error history I_T nexus

If the BUFFER ID field is set to FEh, the device server shall:

a) clear the error history I_T nexus, if any; and
b) not transfer any data.

6.18.7.5 Clear error history I_T nexus and release snapshot

If the BUFFER ID field is set to FFh, the device server shall:

a) clear the error history I_T nexus, if any,
b) release the error history snapshot, if any; and
c) not transfer any data.

6.19 READ MEDIA SERIAL NUMBER command

The READ MEDIA SERIAL NUMBER command (see table 240) requests the device server to return the current media serial number. This command uses the SERVICE ACTION IN(12) CDB format (see 4.2.2.3.5).

Table 240 — READ MEDIA SERIAL NUMBER command

Bit / Byte	7	6	5	4	3	2	1	0
0	OPERATION CODE (ABh)							
1	Reserved			SERVICE ACTION (01h)				
2	Reserved							
•••								
5								
6	(MSB)			ALLOCATION LENGTH				
•••								
9								(LSB)
10	Reserved							
11	CONTROL							

The OPERATION CODE field is defined in 4.2.5.1 and shall be set as shown in table 240 for the READ MEDIA SERIAL NUMBER command.

The SERVICE ACTION field is defined in 4.2.5.2 and shall be set as shown in table 240 for the READ MEDIA SERIAL NUMBER command.

The ALLOCATION LENGTH field is defined in 4.2.5.6.

The CONTROL byte is defined in SAM-5.

The READ MEDIA SERIAL NUMBER parameter data format is shown in table 241.

Table 241 — READ MEDIA SERIAL NUMBER parameter data format

Bit / Byte	7	6	5	4	3	2	1	0
0	(MSB)			MEDIA SERIAL NUMBER LENGTH ((4×n)-4)				
•••								
3								(LSB)
4	MEDIA SERIAL NUMBER							
•••								
(4×n)-1								

The MEDIA SERIAL NUMBER LENGTH field shall contain the number of bytes in the MEDIA SERIAL NUMBER field. The media serial number length shall be a multiple of four. The contents of the MEDIA SERIAL NUMBER LENGTH field are not altered based on the allocation length (see 4.2.5.6).

The MEDIA SERIAL NUMBER field shall contain the vendor specific serial number of the media currently installed. If the number of bytes in the vendor specific serial number is not a multiple of four, then up to three bytes containing zero shall be appended to the highest numbered bytes of the MEDIA SERIAL NUMBER field.

If the media serial number is not available (e.g., the currently installed media has no valid media serial number), then the MEDIA SERIAL NUMBER LENGTH field shall be set to zero.

If the media serial number is not accessible as a result of there is no media present, the device server shall terminate the command with CHECK CONDITION status, with the sense key set to NOT READY, and the additional sense code set to MEDIUM NOT PRESENT.

6.20 RECEIVE COPY DATA(LID4) command

The RECEIVE COPY DATA(LID4) command (see table 242) is a third-party copy command (see 5.16.3) that requests the copy manager to return the held data (see 5.16.4.5), if any, for the copy operation (see 5.16.4.3) specified by the LIST IDENTIFIER field (see 5.16.4.2) in the CDB.

Table 242 — RECEIVE COPY DATA(LID4) command

Bit / Byte	7	6	5	4	3	2	1	0
0	OPERATION CODE (84h)							
1	Reserved			SERVICE ACTION (06h)				
2	(MSB)							
•••				LIST IDENTIFIER				
5								(LSB)
6								
•••				Reserved				
9								
10	(MSB)							
•••				ALLOCATION LENGTH				
13								(LSB)
14	Reserved							
15	CONTROL							

The OPERATION CODE field is defined in 4.2.5.1 and shall be set as shown in table 242 for the RECEIVE COPY DATA(LID4) command.

The SERVICE ACTION field is defined in 4.2.5.2 and shall be set as shown in table 242 for the RECEIVE COPY DATA(LID4) command.

The LIST IDENTIFIER field is defined in 5.16.4.2 and 6.4.3.2, and specifies the copy operation (see 5.16.4.3) about which information is to be transferred. The receive command shall return information from the copy operation that was received on the same I_T nexus with a list identifier that matches the list identifier specified in the RECEIVE COPY RESULTS command's CDB. If no copy operation known to the copy manager has a matching list identifier, the copy manager shall terminate the RECEIVE COPY DATA(LID4) command with CHECK CONDITION status, with the sense key set to ILLEGAL REQUEST, and the additional sense code set to INVALID FIELD IN CDB.

The ALLOCATION LENGTH field is defined in 4.2.5.6. The actual length of the parameter data is available in the AVAILABLE DATA field in the parameter data.

The CONTROL byte is defined in SAM-5.

Table 243 shows the format of the parameter data for the RECEIVE COPY DATA(LID4) command.

Table 243 — Parameter data for the RECEIVE COPY DATA(LID4) command

Bit / Byte	7	6	5	4	3	2	1	0
0	(MSB)							
•••				AVAILABLE DATA (n-3)				
3								(LSB)
4	Reserved				RESPONSE TO SERVICE ACTION			
5	HDD			COPY OPERATION STATUS				
6	(MSB)							
7				OPERATION COUNTER				(LSB)
8	(MSB)							
•••				ESTIMATED STATUS UPDATE DELAY				
11								(LSB)
12				EXTENDED COPY COMPLETION STATUS				
13				LENGTH OF THE SENSE DATA FIELD (x-31)				
14				SENSE DATA LENGTH				
15				TRANSFER COUNT UNITS				
16	(MSB)							
•••				TRANSFER COUNT				
23								(LSB)
24	(MSB)							
25				SEGMENTS PROCESSED				(LSB)
26								
•••				Reserved				
31								
32								
•••				SENSE DATA				
x								
x+1	(MSB)							
•••				HELD DATA LENGTH (n-(x+4))				
x+4								(LSB)
x+5								
•••				HELD DATA				
n								

The copy manager shall not discard the data returned by a RECEIVE COPY DATA(LID4) command in response to an ABORT TASK task management function (see SAM-5) or a COPY OPERATION ABORT command (see 6.3).

The AVAILABLE DATA field, RESPONSE TO SERVICE ACTION field, COPY OPERATION STATUS field, OPERATION COUNTER field, ESTIMATED STATUS UPDATE DELAY field, EXTENDED COPY COMPLETION STATUS field, LENGTH OF THE SENSE DATA FIELD field, SENSE DATA LENGTH field, TRANSFER COUNT UNITS field, TRANSFER COUNT field, SEGMENTS PROCESSED field, and SENSE DATA field are defined in 6.24.

The held data discarded (HDD) bit indicates whether held data has been discarded as described in 5.16.4.5.

If the COPY OPERATION STATUS field is set to a value that table 252 (see 6.24) describes as meaning the operation completed without errors, then the HELD DATA LENGTH field indicates the number of bytes that follow in the HELD DATA field. If the COPY OPERATION STATUS field is not set to a value that table 252 describes as meaning the operation completed without errors, then the HELD DATA LENGTH field shall be set to zero.

The HELD DATA field contains the held data (see 5.16.4.5) for the copy operation (see 5.16.4.3) specified by the list identifier in the CDB (see 5.16.4.2). Unless the copy manager's held data limit (see 5.16.4.5) is exceeded, the first byte held (i.e., the oldest byte held) in response to the copy operation (e.g., the first segment descriptor in the EXTENDED COPY parameter list prescribing the holding of data) is returned in the first byte in the HELD DATA field. The last byte held (i.e., the newest byte held) in response to the copy operation (e.g., the last segment descriptor in the EXTENDED COPY parameter list prescribing the holding of data) is returned in the last byte in the HELD DATA field.

6.21 RECEIVE COPY DATA(LID1) command

The RECEIVE COPY DATA(LID1) command (see table 244) is an SPC-3 compatible third-party copy command (see 5.16.3) that requests the copy manager to return the held data (see 5.16.4.5), if any, for the copy operation (see 5.16.4.3) specified by the LIST IDENTIFIER field (see 5.16.4.2) in the CDB.

Table 244 — RECEIVE COPY DATA(LID1) command

Bit / Byte	7	6	5	4	3	2	1	0
0	OPERATION CODE (84h)							
1	Reserved			SERVICE ACTION (01h)				
2	LIST IDENTIFIER							
3	Reserved							
•••								
9								
10	(MSB)							
•••	ALLOCATION LENGTH							
13								(LSB)
14	Reserved							
15	CONTROL							

The OPERATION CODE field is defined in 4.2.5.1 and shall be set as shown in table 244 for the RECEIVE COPY DATA(LID1) command.

The SERVICE ACTION field is defined in 4.2.5.2 and shall be set as shown in table 244 for the RECEIVE COPY DATA(LID1) command.

The LIST IDENTIFIER field and ALLOCATION LENGTH field are defined in 6.20.

The copy manager shall terminate the RECEIVE COPY DATA(LID1) command with CHECK CONDITION status, with the sense key set to ILLEGAL REQUEST, and the additional sense code set to INVALID FIELD IN CDB if:

a) no copy operation known to the copy manager has a list identifier that matches the LIST IDENTIFIER field in the CDB; or
b) if the LIST IDENTIFIER field in the CDB specifies a copy operation that still is being processed by the copy manager.

The CONTROL byte is defined in SAM-5.

Table 245 shows the format of the parameter data for the RECEIVE COPY DATA(LID1) command.

Table 245 — Parameter data for the RECEIVE COPY DATA(LID1) command

Bit / Byte	7	6	5	4	3	2	1	0
0	(MSB)							
•••				AVAILABLE DATA (n-3)				
3								(LSB)
4								
•••				HELD DATA				
n								

The AVAILABLE DATA field shall contain the number of bytes of held data that follow. The contents of the AVAILABLE DATA field are not altered based on the allocation length (see 4.2.5.6).

The HELD DATA field is defined in 6.20.

6.22 RECEIVE COPY OPERATING PARAMETERS command

The RECEIVE COPY OPERATING PARAMETERS command (see table 246) is an SPC-3 compatible third-party copy command (see 5.16.3) that requests the copy manager to return the operating parameters for the EXTENDED COPY command. The Third-party Copy VPD page (see 7.8.17) provides the same information as the RECEIVE COPY OPERATING PARAMETERS command and additional copy manager support information.

Table 246 — RECEIVE COPY OPERATING PARAMETERS command

Bit / Byte	7	6	5	4	3	2	1	0
0	OPERATION CODE (84h)							
1	Reserved			SERVICE ACTION (03h)				
2								
•••				Reserved				
9								
10	(MSB)							
•••				ALLOCATION LENGTH				
13								(LSB)
14	Reserved							
15	CONTROL							

The OPERATION CODE field is defined in 4.2.5.1 and shall be set as shown in table 246 for the RECEIVE COPY OPERATING PARAMETERS command.

The SERVICE ACTION field is defined in 4.2.5.2 and shall be set as shown in table 246 for the RECEIVE COPY OPERATING PARAMETERS command.

The CONTROL byte is defined in SAM-5.

Table 247 shows the format of the parameter data for the RECEIVE COPY OPERATING PARAMETERS command. If the Third-party Copy VPD page (see 7.8.17) is supported, the values in the RECEIVE COPY OPERATING PARAMETERS parameter data fields shall be the same as the values in the equivalent fields in the Third-party Copy VPD page.

Table 247 — Parameter data for the RECEIVE COPY OPERATING PARAMETERS command (part 1 of 2)

Bit / Byte	7	6	5	4	3	2	1	0
0	(MSB)							
•••				AVAILABLE DATA (n-3)				
3								(LSB)
4				Reserved				SNLID
5								
6				Reserved				
7								
8	(MSB)							
9				MAXIMUM CSCD DESCRIPTOR COUNT				(LSB)
10	(MSB)							
11				MAXIMUM SEGMENT DESCRIPTOR COUNT				(LSB)
12	(MSB)							
•••				MAXIMUM DESCRIPTOR LIST LENGTH				
15								(LSB)
16	(MSB)							
•••				MAXIMUM SEGMENT LENGTH				
19								(LSB)
20	(MSB)							
•••				MAXIMUM INLINE DATA LENGTH				
23								(LSB)
24	(MSB)							
•••				HELD DATA LIMIT				
27								(LSB)
28	(MSB)							
•••				MAXIMUM STREAM DEVICE TRANSFER SIZE				
31								(LSB)
32				Reserved				
33								
34	(MSB)							
35				TOTAL CONCURRENT COPIES				(LSB)
36				MAXIMUM CONCURRENT COPIES				
37				DATA SEGMENT GRANULARITY (log 2)				
38				INLINE DATA GRANULARITY (log 2)				
39				HELD DATA GRANULARITY (log 2)				

Table 247 — Parameter data for the RECEIVE COPY OPERATING PARAMETERS command (part 2 of 2)

Bit / Byte	7	6	5	4	3	2	1	0
40	Reserved							
41								
42								
43	IMPLEMENTED DESCRIPTOR LIST LENGTH (n-43)							
44	List of implemented descriptor type codes (ordered)							
•••								
n								

The AVAILABLE DATA field shall contain the number of bytes following the AVAILABLE DATA field in the parameter data. The contents of the AVAILABLE DATA field are not altered based on the allocation length (see 4.2.5.6).

A supports no list identifier (SNLID) bit set to one indicates the copy manager supports an EXTENDED COPY command parameter list in which the LIST ID USAGE field is set to 11b and the LIST IDENTIFIER field is set to zero as described in 6.4.3.2. A SNLID bit set to zero indicates the copy manager does not support an EXTENDED COPY command parameter list (see 5.16.7.1) in which the LIST ID USAGE field is set to 11b.

The MAXIMUM CSCD DESCRIPTOR COUNT field indicates the maximum number of CSCD descriptors that the copy manager allows in an EXTENDED COPY command parameter list (see 5.16.7.1).

The MAXIMUM SEGMENT COUNT field indicates the maximum number of segment descriptors that the copy manager allows in an EXTENDED COPY command parameter list (see 5.16.7.1).

The MAXIMUM DESCRIPTOR LIST LENGTH field indicates the maximum length, in bytes, of the CSCD descriptor list and segment descriptor list that the copy manager allows in an EXTENDED COPY command parameter list (see 5.16.7.1).

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 5.16.7.2) 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 (see 6.4.3.6) that the copy manager supports in an EXTENDED COPY parameter list (see 5.16.7.1). The MAXIMUM INLINE DATA LENGTH field shall be set to zero if the copy manager does not support descriptor type code 04h (see 6.4.6.6).

The HELD DATA LIMIT field indicates the length, in bytes, of the minimum amount of data the copy manager shall hold for return to the application client as described in 5.16.4.5.

If the SNLID bit is set to one, the TOTAL CONCURRENT COPIES field indicates the maximum number of EXTENDED COPY commands with the LIST ID USAGE field set to 00b, 10b, or 11b that are supported for concurrent processing by the copy manager. If the SNLID bit is set to zero, the TOTAL CONCURRENT COPIES field shall be set to zero.

The MAXIMUM CONCURRENT COPIES field indicates the maximum number of EXTENDED COPY commands with the LIST ID USAGE field set to 00b or 10b that are supported for concurrent processing by the copy manager. If the SNLID bit is set to one, the contents of the TOTAL CONCURRENT COPIES field shall be greater than or equal to the contents of the MAXIMUM CONCURRENT COPIES field.

Each EXTENDED COPY command with the LIST ID USAGE field set to 00b or 10b reduces the number of EXTENDED COPY commands with the LIST ID USAGE field set to 11b that may be processed concurrently. However, the converse may not be true (e.g., when the number of outstanding EXTENDED COPY commands with the LIST ID USAGE field set to 11b exceeds the difference between the total concurrent copies and maximum concurrent copies).

The DATA SEGMENT GRANULARITY field indicates the length of the smallest data block that the 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 5.16.7.2) 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 6.4.6.6). 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 5.16.7.2) 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, then the copy manager shall terminate the copy operation (see 5.16.4.3) with CHECK CONDITION status, with the sense key set to COPY ABORTED, and the additional sense code set to COPY SEGMENT GRANULARITY VIOLATION.

The HELD DATA GRANULARITY field indicates the length of the smallest block of held data (see 5.16.4.5) that the copy manager shall transfer to the application client in response to a RECEIVE COPY DATA(LID4) command (see 6.20) or a RECEIVE COPY DATA(LID1) command (see 6.21). The amount of data held by the copy manager in response to any one function (e.g., one segment descriptor (see 6.4.6)) in a copy operation (see 5.16.4.3) 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 indicates 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 CSCD DESCRIPTOR TYPE CODE value (see 6.4.5) 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.

6.23 RECEIVE COPY FAILURE DETAILS(LID1) command

The RECEIVE COPY FAILURE DETAILS(LID1) command (see table 248) is an SPC-3 compatible third-party copy command (see 5.16.3) that requests the copy manager to return details of the segment processing failure, if any, that caused termination of the EXTENDED COPY(LID1) command specified by the LIST IDENTIFIER field in the CDB.

Table 248 — RECEIVE COPY FAILURE DETAILS(LID1) command

Bit / Byte	7	6	5	4	3	2	1	0
0	OPERATION CODE (84h)							
1	Reserved			SERVICE ACTION (04h)				
2	LIST IDENTIFIER							
3	Reserved							
•••								
9								
10	(MSB)			ALLOCATION LENGTH				
•••								
13								(LSB)
14	Reserved							
15	CONTROL							

The OPERATION CODE field is defined in 4.2.5.1 and shall be set as shown in table 248 for the RECEIVE COPY FAILURE DETAILS(LID1) command.

The SERVICE ACTION field is defined in 4.2.5.2 and shall be set as shown in table 248 for the RECEIVE COPY FAILURE DETAILS(LID1) command.

The LIST IDENTIFIER field and ALLOCATION LENGTH field are defined in 6.20.

The copy manager shall terminate the RECEIVE COPY FAILURE DETAILS(LID1) command with CHECK CONDITION status, with the sense key set to ILLEGAL REQUEST, and the additional sense code set to INVALID FIELD IN CDB if:

a) no EXTENDED COPY(LID1) command known to the copy manager has a list identifier that matches the LIST IDENTIFIER field in the CDB; or
b) if the LIST IDENTIFIER field in the CDB specifies an EXTENDED COPY(LID1) command that still is being processed by the copy manager.

The CONTROL byte is defined in SAM-5.

Table 249 shows the format of the parameter data for the RECEIVE COPY FAILURE DETAILS(LID1) command.

Table 249 — Parameter data for the RECEIVE COPY FAILURE DETAILS(LID1) command

Bit / Byte	7	6	5	4	3	2	1	0
0	(MSB)							
•••				AVAILABLE DATA (n-3)				
3								(LSB)
4								
•••				Reserved				
55								
56				EXTENDED COPY COMMAND STATUS				
57				Reserved				
58	(MSB)							
59				SENSE DATA LENGTH (n-59)				(LSB)
60								
•••				SENSE DATA				
n								

If processing of an EXTENDED COPY(LID1) command is aborted and processing of a segment descriptor is incomplete, then 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 CSCDs (e.g., stream devices) have been positioned by incomplete processing.

The application client should issue a RECEIVE COPY FAILURE DETAILS(LID1) command as soon as possible after the failure of the EXTENDED COPY(LID1) command to ensure 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(LID1) command have been successfully transferred to the application client;
b) if a RECEIVE COPY FAILURE DETAILS(LID1) command has been received on the same I_T nexus with a matching list identifier, with the ALLOCATION LENGTH field set to zero;
c) if another EXTENDED COPY(LID1) command is received on the same I_T nexus using the same list identifier;
d) if the copy manager detects a logical unit reset condition or I_T nexus loss condition (see SAM-5); or
e) if 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 that follow. 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 transferred. The contents of the AVAILABLE DATA field are not altered based on the allocation length (see 4.2.5.6).

The COPY COMMAND STATUS field contains the status value that was returned for the EXTENDED COPY(LID1) command specified 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(LID1) command indicated by the list identifier with CHECK CONDITION status.

6.24 RECEIVE COPY STATUS(LID4) command

The RECEIVE COPY STATUS(LID4) command (see table 250) is a third-party copy command (see 5.16.3) that requests the copy manager to return the status information (see 5.16.4.4) for the copy operation (see 5.16.4.3) specified by the LIST IDENTIFIER field (see 5.16.4.2) in the CDB.

Table 250 — RECEIVE COPY STATUS(LID4) command

Bit / Byte	7	6	5	4	3	2	1	0
0	OPERATION CODE (84h)							
1	Reserved			SERVICE ACTION (05h)				
2	(MSB)							
•••				LIST IDENTIFIER				
5								(LSB)
6								
•••				Reserved				
9								
10	(MSB)							
•••				ALLOCATION LENGTH				
13								(LSB)
14	Reserved							
15	CONTROL							

The OPERATION CODE field is defined in 4.2.5.1 and shall be set as shown in table 250 for the RECEIVE COPY STATUS(LID4) command.

The SERVICE ACTION field is defined in 4.2.5.2 and shall be set as shown in table 250 for the RECEIVE COPY STATUS(LID4) command.

The LIST IDENTIFIER field and ALLOCATION LENGTH field are defined in 6.20.

If no copy operation known to the copy manager has a list identifier that matches the contents of the LIST IDENTIFIER field, then the copy manager shall terminate the RECEIVE COPY STATUS(LID4) command with CHECK CONDITION status, with the sense key set to ILLEGAL REQUEST, and the additional sense code set to INVALID FIELD IN CDB.

The CONTROL byte is defined in SAM-5.

Table 251 shows the format of the parameter data for the RECEIVE COPY STATUS(LID4) command.

Table 251 — Parameter data for the RECEIVE COPY STATUS(LID4) command

Bit / Byte	7	6	5	4	3	2	1	0
0	(MSB)							
•••				AVAILABLE DATA (n-3)				
3								(LSB)
4	Reserved			RESPONSE TO SERVICE ACTION				
5	Reserved			COPY OPERATION STATUS				
6	(MSB)							
7				OPERATION COUNTER				(LSB)
8	(MSB)							
•••				ESTIMATED STATUS UPDATE DELAY				
11								(LSB)
12	EXTENDED COPY COMPLETION STATUS							
13	LENGTH OF THE SENSE DATA FIELD (X-31)							
14	SENSE DATA LENGTH							
15	TRANSFER COUNT UNITS							
16	(MSB)							
•••				TRANSFER COUNT				
23								(LSB)
24	(MSB)							
25				SEGMENTS PROCESSED				(LSB)
26								
•••				Reserved				
31								
32								
•••				SENSE DATA				
x								

The AVAILABLE DATA field shall contain the number of bytes that follow in the parameter data. The contents of the AVAILABLE DATA field are not altered based on the allocation length (see 4.2.5.6).

The RESPONSE TO SERVICE ACTION field indicates the value in the SERVICE ACTION field of the third-party copy command (see 5.16.3) specified by the LIST IDENTIFIER field in the CDB.

The COPY OPERATION STATUS field indicates the status of the copy operation (see 5.16.4.3) specified by the LIST IDENTIFIER field in the CDB as defined in table 252.

Table 252 — COPY OPERATION STATUS field

Code	Meaning	Operation completed without errors
01h	Operation completed without errors	Yes
02h	Operation completed with errors	No
03h [a]	Operation completed without errors but with partial ROD token usage (e.g., the residual is negative (see SBC-3))	Yes
04h	Operation completed without errors but with residual data [b]	Yes
10h	Operation in progress, foreground or background unknown [c]	No
11h	Operation in progress in foreground (see 5.16.4.3)	No
12h	Operation in progress in background (see 5.16.4.3)	No
60h	Operation terminated (e.g., by the preemption of a persistent reservation (see 5.12.11.2.6) or by a COPY OPERATION ABORT command (see 6.3))	No
all others	Reserved	

[a] If the third-party copy command that originated the copy operation is an EXTENDED COPY command, then the COPY OPERATION STATUS field shall never be set to 03h.

[b] The copy manager has determined that the application client should verify that all requested data transfers have been performed.

[c] Codes 11h and 12h should be used instead of this code whenever possible.

The OPERATION COUNTER field contains a wrapping counter of the number of SCSI commands, or equivalent, that the copy manager has sent to a copy source or copy destination as part of processing the copy operation (see 5.16.4.3) specified by the LIST IDENTIFIER field in the CDB, and for which the copy manager has received one of the following completion status values:

a) GOOD;
b) CONDITION MET; or
c) CHECK CONDITION with the sense key set to RECOVERED ERROR.

The ESTIMATED STATUS UPDATE DELAY field indicates the number of milliseconds that the copy manager recommends that the application client wait before sending another RECEIVE COPY STATUS(LID4) command on the same I_T nexus with the same list identifier. If a RECEIVE COPY STATUS(LID4) command is received sooner than the indicated time, the copy manager may return the same parameter data that was returned in response to the previous RECEIVE COPY STATUS(LID4) command. If the COPY OPERATION STATUS field is set to 01h, 02h, 03h, 04h, or 60h, then the ESTIMATED STATUS UPDATE DELAY field shall be set to FFFF FFFEh. If the ESTIMATED STATUS UPDATE DELAY field is set to FFFF FFFFh, then the copy manager is unable to recommend a delay interval.

The EXTENDED COPY COMPLETION STATUS field indicates the status code (see SAM-5), if any, established for the completed copy operation (see 5.16.4.3) specified by the LIST IDENTIFIER field and is affected by the contents of the COPY OPERATION STATUS field as shown in table 253. If the IMMED bit, if any, (see 5.16.4.3) is set to one in the third-party copy command that originated the copy operation, then the contents of the EXTENDED COPY COMPLETION STATUS field may be different than the status returned by the originating third-party copy command.

Table 253 — EXTENDED COPY COMPLETION STATUS field contents based on COPY OPERATION STATUS field

COPY OPERATION STATUS field contents	EXTENDED COPY COMPLETION STATUS field contents
10h, 11h, or 12h	The EXTENDED COPY COMPLETION STATUS field is reserved.
01h, 02h, 03h, or 04h	The EXTENDED COPY COMPLETION STATUS field indicates the status code (see SAM-5) established for the completed copy operation specified by the LIST IDENTIFIER field.
60h	The EXTENDED COPY COMPLETION STATUS field shall be set to TASK ABORTED (see SAM-5).

The LENGTH OF THE SENSE DATA FIELD field indicates the number of bytes in the SENSE DATA field. The LENGTH OF THE SENSE DATA FIELD field shall be set to zero or a multiple of four.

If the COPY OPERATION STATUS field is set to 10h, 11h, 12h, or 60h, then the SENSE DATA LENGTH field shall be set to zero. If the COPY OPERATION STATUS field is set to 01h, 02h, 03h, or 04h, then the SENSE DATA LENGTH field indicates the number of bytes in the SENSE DATA field that contain sense data. The value in the SENSE DATA LENGTH field shall be less than or equal to the value in the LENGTH OF THE SENSE DATA FIELD field (e.g., the SENSE DATA field may contain pad bytes that are counted in the LENGTH OF THE SENSE DATA FIELD field but not in the SENSE DATA LENGTH field).

The TRANSFER COUNT UNITS field indicates the units for the TRANSFER COUNT field as shown in table 254.

Table 254 — COPY STATUS TRANSFER COUNT UNITS field

Code	Meaning [a]	Multiplier to convert TRANSFER COUNT field to bytes
00h	Bytes	1
01h	Kibibytes	2^{10} (i.e., 1 024)
02h	Mebibytes	2^{20}
03h	Gibibytes	2^{30}
04h	Tebibytes	2^{40}
05h	Pebibytes	2^{50}
06h	Exbibytes	2^{60}
F1h	n/a	logical block length of copy destination
all others	Reserved	

[a] See 3.5.2.

The TRANSFER COUNT field indicates the amount of data written to a copy destination for the copy operation (see 5.16.4.3) specified by the LIST IDENTIFIER field prior to the time at which the copy manager processed the RECEIVE COPY STATUS(LID4) command. If data has been written to the copy destination in an order other than what the command specified (e.g., if concurrent processing of multiple segment descriptors has resulted

in some data being written out of order), then the TRANSFER COUNT field shall indicate only those bytes that have been written in strict order conformance with what the command specified.

The SEGMENTS PROCESSED field indicates the number of segments the copy manager has processed for the copy operation (see 5.16.4.3) specified by the LIST IDENTIFIER field including the segment currently being processed. The SEGMENTS PROCESSED field shall be set to zero if:

a) the copy manager has not yet begun processing segment descriptors; or
b) the RESPONSE TO SERVICE ACTION field indicates a third-party copy command that does not support segment descriptors.

6.25 RECEIVE COPY STATUS(LID1) command

The RECEIVE COPY STATUS(LID1) command (see table 255) is an SPC-3 compatible third-party copy command (see 5.16.3) that requests the copy manager to return the status information for the copy operation (see 5.16.4.3) specified by the LIST IDENTIFIER field (see 5.16.4.2) in the CDB.

Table 255 — RECEIVE COPY STATUS(LID1) command

Bit / Byte	7	6	5	4	3	2	1	0
0	OPERATION CODE (84h)							
1	Reserved			SERVICE ACTION (00h)				
2	LIST IDENTIFIER							
3	Reserved							
•••								
9								
10	(MSB)							
•••	ALLOCATION LENGTH							
13								(LSB)
14	Reserved							
15	CONTROL							

The OPERATION CODE field is defined in 4.2.5.1 and shall be set as shown in table 255 for the RECEIVE COPY STATUS(LID1) command.

The SERVICE ACTION field is defined in 4.2.5.2 and shall be set as shown in table 255 for the RECEIVE COPY STATUS(LID1) command.

The LIST IDENTIFIER field and ALLOCATION LENGTH field are defined in 6.20.

If no copy operation known to the copy manager has a list identifier that matches the contents of the LIST IDENTIFIER field, then the copy manager shall terminate the RECEIVE COPY STATUS(LID1) command with CHECK CONDITION status, with the sense key set to ILLEGAL REQUEST, and the additional sense code set to INVALID FIELD IN CDB.

The CONTROL byte is defined in SAM-5.

Table 256 shows the format of the parameter data for the RECEIVE COPY STATUS(LID1) command.

Table 256 — Parameter data for the RECEIVE COPY STATUS(LID1) command

Bit / Byte	7	6	5	4	3	2	1	0
0	(MSB)							
•••				AVAILABLE DATA (00000008h)				
3								(LSB)
4	HDD			COPY COMMAND STATUS				
5	(MSB)							
6				SEGMENTS PROCESSED				(LSB)
7				TRANSFER COUNT UNITS				
8	(MSB)							
•••				TRANSFER COUNT				
11								(LSB)

After the completion of a copy operation (see 5.16.4.3), the copy manager shall preserve all data returned by the RECEIVE COPY STATUS(LID1) command for a vendor specific period of time. The copy manager shall discard the RECEIVE COPY STATUS(LID1) command parameter data if:

a) a RECEIVE COPY STATUS(LID1) command is received on the same I_T nexus with a matching list identifier;
b) another third-party copy command that originates a copy operation (see table 108 in 5.16.3) is received on the same I_T nexus and the list identifier matches the list identifier associated with the data preserved for the RECEIVE COPY STATUS(LID1) command;
c) the copy manager detects a logical unit reset condition or I_T nexus loss condition (see SAM-5); or
d) the copy manager requires the resources used to preserve the data.

The AVAILABLE DATA field shall contain the number of bytes that follow in the parameter data, and shall be set as shown in table 256 for the RECEIVE COPY STATUS(LID1) command. The contents of the AVAILABLE DATA field are not altered based on the allocation length (see 4.2.5.6).

The held data discarded (HDD) bit indicates whether held data has been discarded as described in 5.16.4.5.

The COPY COMMAND STATUS field is set to the current status of the third-party copy command (see 5.16.3) specified by the LIST IDENTIFIER field in the CDB as defined in table 257.

Table 257 — COPY COMMAND STATUS field

Code	Meaning
00h	Operation in progress
01h	Operation completed without errors
02h	Operation completed with errors
03h to 7Fh	Reserved

The SEGMENTS PROCESSED field, TRANSFER COUNT UNITS field, and TRANSFER COUNT field are defined in 6.24.

6.26 RECEIVE CREDENTIAL command

6.26.1 RECEIVE CREDENTIAL command description

6.26.1.1 Overview

The RECEIVE CREDENTIAL command (see table 258) requests a security manager device server (e.g., a CbCS management device server (see 5.13.6.7.3)) to return a credential to a secure CDB originator (see 5.13.6.2) (e.g., for use in the CbCS extension descriptor (see 5.13.6.7.16) of a CDB).

Table 258 — RECEIVE CREDENTIAL command

Bit / Byte	7	6	5	4	3	2	1	0
0	OPERATION CODE (7Fh)							
1	CONTROL							
2	Reserved							
•••								
6								
7	ADDITIONAL CDB LENGTH (n-7)							
8	(MSB)			SERVICE ACTION (1800h)				
9								(LSB)
10	(MSB)			ALLOCATION LENGTH				
11								(LSB)
12	(MSB)			AC_SAI				
•••								
15								(LSB)
16	ENCRYPTED REQUEST DESCRIPTOR							
•••								
n								

The OPERATION CODE field is defined in 4.2.5.1 and shall be set as shown in table 258 for the RECEIVE CREDENTIAL command.

The ADDITIONAL CDB LENGTH field is defined in 4.2.3.

The SERVICE ACTION field is defined in 4.2.5.2 and shall be set as shown in table 258 for the RECEIVE CREDENTIAL command.

The ALLOCATION LENGTH field is defined in 4.2.5.6.

The AC_SAI field is set to the value of the AC_SAI SA parameter (see 5.13.2.2) for the SA to be used to encrypt the parameter data as described in 6.26.2.1.

The ENCRYPTED REQUEST DESCRIPTOR field shall contain an ESP-SCSI CDB descriptor (see 5.13.7.4). Before encryption and after decryption, the UNENCRYPTED BYTES field (see 5.13.7.3) that are used to compute the ENCRYPTED OR AUTHENTICATED DATA field (see 5.13.7.4) contents shall have the format shown in table 259.

Table 259 — RECEIVE CREDENTIAL command unencrypted bytes format

Bit / Byte	7	6	5	4	3	2	1	0
0	(MSB)			CREDENTIAL REQUEST TYPE				
1								(LSB)
2				CREDENTIAL REQUEST DESCRIPTOR				
•••								
n								

The CREDENTIAL REQUEST TYPE field (see table 260) specifies type of credential being requested and the format of the CREDENTIAL REQUEST DESCRIPTOR field.

Table 260 — CREDENTIAL REQUEST TYPE field

Code	Description	Reference
0001h	CbCS logical unit	6.26.1.2
0002h	CbCS logical unit and volume	6.26.1.3
all other codes	Reserved	

The CREDENTIAL REQUEST DESCRIPTOR field specifies information that describes the requested credential (see table 260).

The CONTROL byte is defined in SAM-5.

If return of the requested credential is not permitted, the device server shall terminate the command with CHECK CONDITION status, with the sense key set to ILLEGAL REQUEST, and the additional sense code set to ACCESS DENIED - NO ACCESS RIGHTS.

The DS_SAI field in the ENCRYPTED REQUEST DESCRIPTOR field is set to the value of the DS_SAI SA parameter (see 5.13.2.2) for the SA to be used to encrypt the unencrypted bytes and the parameter data as described.

If the device server is not maintaining an SA with an AC_SAI SA parameter that matches the AC_SAI field contents and a DS_SAI SA parameter that matches the DS_SAI field contents, then the device server shall terminate the command with CHECK CONDITION status, with the sense key set to ILLEGAL REQUEST, and the additional sense code set to INVALID FIELD IN CDB.

If the device server is maintaining the SA specified by the AC_SAI field and the DS_SAI field, then the SA shall be verified for use by this RECEIVE CREDENTIAL command as follows:

a) the USAGE_TYPE SA parameter (see 5.13.2.2) shall be verified to be equal to 82h (i.e., CbCS authentication and credential encryption; and
b) the USAGE_DATA SA parameter (see 5.13.2.2) shall be verified not to contain an ALGORITHM IDENTIFIER field (see 7.7.3.6) that is set to ENCR_NULL based on the contents the IKEv2-SCSI SAUT Cryptographic Algorithm payload (see 7.7.3.5.14) for the ENCR algorithm type (see 7.7.3.6.2) during creation of the SA (see 5.13.2.3).

If any of these SA verifications fails, the device server shall terminate the command with CHECK CONDITION status, with the sense key set to ILLEGAL REQUEST, and the additional sense code set to INVALID FIELD IN CDB.

6.26.1.2 CbCS logical unit credential request descriptor

If the CREDENTIAL REQUEST TYPE field is set to 0001h (i.e., CbCS logical unit), the format of the CREDENTIAL REQUEST DESCRIPTOR field is as shown in table 261.

Table 261 — CbCS logical unit credential request descriptor format

Bit / Byte	7	6	5	4	3	2	1	0
0	DESIGNATION DESCRIPTOR							
•••								
19								

The format of the DESIGNATION DESCRIPTOR field is defined in table 590 (see 7.8.6.1). The device server shall terminate the command with CHECK CONDITION status, with the sense key set to ILLEGAL REQUEST, and the additional sense code set to INVALID FIELD IN CDB if any of the fields in the DESIGNATION DESCRIPTOR field are set as follows:

a) the DESIGNATOR TYPE field is set to any value other than 3h (i.e., NAA);
b) the ASSOCIATION field is set to any value other than 00b (i.e., logical unit) or 10b (i.e., SCSI target device); or
c) the DESIGNATOR LENGTH field is set to a value that is larger than 16.

6.26.1.3 CbCS logical unit and volume credential request descriptor

If the CREDENTIAL REQUEST TYPE field is set to 0002h (i.e., CbCS logical unit and volume), the format of the CREDENTIAL REQUEST DESCRIPTOR field is as shown in table 262.

Table 262 — CbCS logical unit and volume credential request descriptor format

Bit / Byte	7	6	5	4	3	2	1	0
0	DESIGNATION DESCRIPTOR							
•••								
19								
20	MAM ATTRIBUTE							
•••								
56								

The format of the DESIGNATION DESCRIPTOR field is defined in table 590 (see 7.8.6.1). The device server shall terminate the command with CHECK CONDITION status, with the sense key set to ILLEGAL REQUEST, and the additional sense code set to INVALID FIELD IN CDB if any of the fields in the DESIGNATION DESCRIPTOR field are set as follows:

a) the DESIGNATOR TYPE field is set to any value other than 3h (i.e., NAA);
b) the ASSOCIATION field is set to any value other than 00b (i.e., logical unit) or 10b (i.e., SCSI target device); or
c) the DESIGNATOR LENGTH field is set to a value that is larger than 16.

The format of the MAM ATTRIBUTE field is defined in table 440 (see 7.4.1). If the ATTRIBUTE IDENTIFIER field in the MAM ATTRIBUTE field is set to any value other than 0401h (i.e., MEDIUM SERIAL NUMBER), then the device server shall terminate the command with CHECK CONDITION status, with the sense key set to ILLEGAL REQUEST, and the additional sense code set to INVALID FIELD IN CDB.

6.26.2 RECEIVE CREDENTIAL parameter data

6.26.2.1 RECEIVE CREDENTIAL parameter data encryption

The RECEIVE CREDENTIAL parameter data shall be one of the ESP-SCSI Data-In Buffer descriptors shown in table 100 (see 5.13.7.5.1). The SA specified by the AC_SAI field and the DS_SAI field in the CDB shall be used to construct the ESP-SCSI Data-In Buffer descriptor as described in 5.13.7.5.

Before processing the parameter data, the application client should validate and decrypt the ESP-SCSI Data-In Buffer descriptor as described in 5.13.7.5. If any errors are detected by the validation and decryption processing, the parameter data should be ignored.

6.26.2.2 RECEIVE CREDENTIAL decrypted parameter data

Before encryption and after decryption, the UNENCRYPTED BYTES field (see 5.13.7.3) that are used to compute the ENCRYPTED OR AUTHENTICATED DATA field (see 5.13.7.5) contents shall contain a CbCS credential (see table 263).

Table 263 — CbCS credential format

Bit / Byte	7	6	5	4	3	2	1	0
0	Reserved				CREDENTIAL FORMAT (1h)			
1	Reserved							
2	(MSB)			CREDENTIAL LENGTH (n-3)				
3								(LSB)
4	(MSB)			CAPABILITY LENGTH (k-5)				
5								(LSB)
6				CbCS capability descriptor (see 6.26.2.3)				
•••								
k								
k+1	(MSB)			CAPABILITY KEY LENGTH (n-(k-4))				
•••								
k+4								(LSB)
k+5				CAPABILITY KEY				
•••								
n								

The CREDENTIAL FORMAT field (see table 264) indicates the format of the credential.

Table 264 — Credential format values

Value	Description
0h	Reserved
1h	The format defined by this standard
2h to Fh	Reserved

The CREDENTIAL LENGTH field indicates the number of bytes that follow in the credential including the capability length, the CbCS capability descriptor, the capability key length, and the capability key.

The CAPABILITY LENGTH field indicates the number of bytes that follow in the capability.

The contents of the CbCS capability descriptor are defined in 6.26.2.3.

The CAPABILITY KEY LENGTH field indicates the number of bytes that follow in the capability key.

The CAPABILITY KEY field contains an integrity check value that is computed and used as described in 5.13.6.7.12.

6.26.2.3 CbCS capability descriptor

6.26.2.3.1 Overview

A CbCS capability descriptor (see table 265) specifies the commands that are allowed by the CbCS extension descriptor in which it appears.

Table 265 — CbCS capability descriptor format

Bit / Byte	7	6	5	4	3	2	1	0
0	DESIGNATION TYPE				KEY VERSION			
1	CBCS METHOD							
2	(MSB)							
•••				CAPABILITY EXPIRATION TIME				
7								(LSB)
8	(MSB)							
•••				INTEGRITY CHECK VALUE ALGORITHM				
11								(LSB)
12								
•••				PERMISSIONS BIT MASK				
15								
16	(MSB)							
•••				POLICY ACCESS TAG				
19								(LSB)
20								
•••				DESIGNATION DESCRIPTOR				
57								
58								
•••				DISCRIMINATOR				
71								

The DESIGNATION TYPE field (see table 266) specifies the format of the Designation descriptor.

Table 266 — DESIGNATION TYPE field

Code	Description	DESIGNATION DESCRIPTOR field format reference
0h	Reserved	
1h	Logical unit designation descriptor	6.26.2.3.2
2h	MAM attribute designation descriptor	6.26.2.3.3
3h to Fh	Reserved	

The KEY VERSION field specifies which working key (see 5.13.6.7.11), is being used to compute the capability key (see 5.13.6.7.12) for this CbCS capability.

The CBCS METHOD field (see table 267) specifies the CbCS method used by this CbCS capability.

Table 267 — CBCS METHOD field

Code	CbCS method	Reference
00h	BASIC	5.13.6.7.8.2
01h	CAPKEY	5.13.6.7.8.3
02h to EFh	Reserved	
F0h to FEh	Vendor specific	
FFh	Reserved	

The CAPABILITY EXPIRATION TIME field specifies expiration time of this CbCS capability as the number of milliseconds that have elapsed since midnight, 1 January 1970 UT. If the CAPABILITY EXPIRATION TIME field is set to zero, this CbCS capability does not have an expiration time.

If the CAPABILITY EXPIRATION TIME field is not set to zero, then:

a) the clock maintained by the CbCS management device server (see 5.13.6.7.3) should be synchronized with the clock maintained by the enforcement manager (see 5.13.6.7.7). The method for synchronizing the clocks is outside the scope of this standard. However, the protocol should be implemented in a secure manner (e.g., it should not be possible for an adversary to set the clock in the SCSI device or in the secure CDB processor backwards to enable the reuse of expired CbCS credentials). The value in the enforcement manager's clock is available in the Current CbCS Parameters CbCS page (see 7.7.4.3.5) to assist in this synchronization;
b) the CbCS management device server should set the CAPABILITY EXPIRATION TIME field to a value that is at least an order of magnitude larger than the allowed deviation between the clocks.

The INTEGRITY CHECK VALUE ALGORITHM field specifies the algorithm used to compute the capability key and other integrity check values for this CbCS capability. The value in the INTEGRITY CHECK VALUE ALGORITHM field is selected from the codes that the Unchangeable CbCS Parameters CbCS page (see 7.7.4.3.3) lists as supported integrity check value algorithms.

The PERMISSIONS BIT MASK field (see table 268) specifies the permissions allowed by this CbCS capability. More than one permissions bit may be set. The relationship between commands and bits in the PERMISSIONS BIT MASK field is defined in for the commands defined by this standard and in the command standard that defines commands for a specific device type.

Table 268 — PERMISSIONS BIT MASK field format

Bit / Byte	7	6	5	4	3	2	1	0
0	DATA READ	DATA WRITE	PARM READ	PARM WRITE	SEC MGMT	RESRV	MGMT	PHY ACC
1	Reserved							
2								
3	Restricted (see applicable command standard)							

A DATA READ bit set to zero indicates a command has no read permission for user data and protection information, if any. A DATA READ bit set to one indicates a command has permission to read user data and protection information, if any.

A DATA WRITE bit set to zero indicates a command has no write permission for user data and protection information, if any. A DATA WRITE bit set to one indicates a command has permission to write user data and protection information, if any.

A parameter data read (PARM READ) bit set to zero indicates a command has no parameter data read permission. A PARM READ bit set to one indicates a command has permission to read parameter data.

A parameter data write (PARM WRITE) bit set to zero indicates a command has no parameter data write permission. A PARM WRITE bit set to one indicates a command has permission to write parameter data.

A security management (SEC MGMT) bit set to zero indicates a command has no security management permission. A SEC MGMT bit set to one indicates a command has security management permission.

A reservation (RESRV) bit set to zero indicates a command has no persistent reservation permission. A RESRV bit set to one indicates a command has permission to make or modify persistent reservations.

A management (MGMT) bit set to zero indicates a command has no storage management permission. A MGMT bit set to one indicates a command has storage management permission. Storage management is outside the scope of this standard.

A physical access (PHY ACC) bit set to zero indicates a command has no permission to affect physical access to the logical unit or volume (see SSC-4 or SMC-3). A PHY ACC bit set to one indicates a command has permission to affect physical access to the logical unit or volume.

If the POLICY ACCESS TAG field is set to a value other than zero, the policy access tag attribute of the logical unit (see 5.13.6.7.15) is compared to the POLICY ACCESS TAG field contents as part of validating the CbCS capability (see 5.13.6.7.13.2). If the POLICY ACCESS TAG field is set to zero, then no comparison is made.

The DESIGNATION DESCRIPTOR field is used during the validation of the CbCS capability (see 5.13.6.7.13.2) to ensure that the command is being addressed to the correct logical unit or volume (see SSC-4 or SMC-3). The format of the DESIGNATION DESCRIPTOR field is defined by the value in the DESIGNATION TYPE field as described in table 266.

If the CREDENTIAL REQUEST TYPE field in a RECEIVE CREDENTIAL command is set to 0001h (i.e., CbCS logical unit), then the DESIGNATION DESCRIPTOR field shall contain a logical unit designation descriptor that matches the DESIGNATION DESCRIPTOR field (see 6.26.1.2) in the CREDENTIAL REQUEST DESCRIPTOR field in the CDB. If the CREDENTIAL REQUEST TYPE field in a RECEIVE CREDENTIAL command is set to 0002h (i.e., CbCS logical unit and volume), then the DESIGNATION DESCRIPTOR field shall contain a MAM attribute designation descriptor that matches the MAM ATTRIBUTE field (see 6.26.1.3) in the CREDENTIAL REQUEST DESCRIPTOR field in the CDB.

The DISCRIMINATOR field provides uniqueness to the CbCS capability descriptor and may be used to limit the delegation or prevent leakage of the CbCS capability to other application clients. The CbCS management device server (see 5.13.6.7.3) shall not return the same CbCS capability descriptor to two secure CDB originators.

The enforcement manager (see 5.13.6.7.7) shall validate each CbCS capability descriptor received as described in 5.13.6.7.13.2.

6.26.2.3.2 Logical unit designation descriptor format

If the DESIGNATION TYPE field is set to 0001h (i.e., logical unit designation descriptor), then the format of the DESIGNATION DESCRIPTOR field is as shown in table 269.

Table 269 — Logical unit designation descriptor format

Bit / Byte	7	6	5	4	3	2	1	0
0	DESIGNATION DESCRIPTOR							
•••								
19								
20	Reserved							
•••								
37								

The format of the DESIGNATION DESCRIPTOR field is defined in table 590 (see 7.8.6.1) with the following additional requirements:

a) the DESIGNATOR TYPE field shall contain 3h (i.e., NAA);
b) the ASSOCIATION field shall be set to 00b (i.e., logical unit); and
c) the DESIGNATOR LENGTH field shall be set to a value that is smaller than 17.

6.26.2.3.3 Volume designation descriptors

If the DESIGNATION TYPE field is set to 0002h (i.e., volume unit designation descriptor), then the format of the DESIGNATION DESCRIPTOR field is as shown in table 270.

Table 270 — Volume designation descriptor format

Bit / Byte	7	6	5	4	3	2	1	0
0	MAM ATTRIBUTE							
•••								
36								
37	Reserved							

The format of the MAM ATTRIBUTE field is defined in table 440 (see 7.4.1) with the following additional requirements:

a) the MAM ATTRIBUTE field shall contain 0401h (i.e., MEDIUM SERIAL NUMBER); and
b) the ATTRIBUTE LENGTH field shall contain 0020h.

6.27 RECEIVE DIAGNOSTIC RESULTS command

The RECEIVE DIAGNOSTIC RESULTS command (see table 271) requests the device server to return data based on the most recent SEND DIAGNOSTIC command (see 6.42) or a diagnostic page specified by the PAGE CODE field.

Table 271 — 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							

The OPERATION CODE field is defined in 4.2.5.1 and shall be set as shown in table 271 for the RECEIVE DIAGNOSTIC RESULTS command.

A page code valid (PCV) bit set to zero specifies that the device server return parameter data based on the most recent SEND DIAGNOSTIC command (e.g., the diagnostic page with the same page code as that specified in the most recent SEND DIAGNOSTIC command). The response to a RECEIVE DIAGNOSTIC RESULTS command with the PCV bit set to zero is vendor specific if:

a) the most recent SEND DIAGNOSTIC command was not a SEND DIAGNOSTIC command defining parameter data to return;
b) a RECEIVE DIAGNOSTIC RESULTS command with a PCV bit set to one has been processed since the last SEND DIAGNOSTIC command was processed; or
c) no SEND DIAGNOSTIC command with the PF bit set to one defining parameter data to return has been processed since power on, hard reset, or logical unit reset.

A page code valid (PCV) bit set to one specifies that the device server return the diagnostic page specified in the PAGE CODE field. Page code values are defined in 7.2 or in another command standard.

NOTE 35 - Logical units compliant with SPC-2 may transfer more than one diagnostic page in the parameter data if the PCV bit is set to zero and the previous SEND DIAGNOSTIC command sent more than one diagnostic page in the parameter list.

NOTE 36 - To ensure that the diagnostic command information is not destroyed by a command sent from another I_T nexus, the logical unit should be reserved.

The ALLOCATION LENGTH field is defined in 4.2.5.6.

The CONTROL byte is defined in SAM-5.

See 7.2 for RECEIVE DIAGNOSTIC RESULTS diagnostic page format definitions.

6.28 RECEIVE ROD TOKEN INFORMATION command

The RECEIVE ROD TOKEN INFORMATION command (see table 272) is a third-party copy command (see 5.16.3) that requests the copy manager to return all the ROD tokens (see 5.16.6.1) created during the copy operation (see 5.16.4.3) specified by the LIST IDENTIFIER field (see 5.16.4.2) in the CDB.

Table 272 — RECEIVE ROD TOKEN INFORMATION command

Bit / Byte	7	6	5	4	3	2	1	0
0	OPERATION CODE (84h)							
1	Reserved			SERVICE ACTION (07h)				
2	(MSB)							
•••	LIST IDENTIFIER							
5								(LSB)
6								
•••	Reserved							
9								
10	(MSB)							
•••	ALLOCATION LENGTH							
13								(LSB)
14	Reserved							
15	CONTROL							

The OPERATION CODE field is defined in 4.2.5.1 and shall be set as shown in table 272 for the RECEIVE ROD TOKEN INFORMATION command.

The SERVICE ACTION field is defined in 4.2.5.2 and shall be set as shown in table 272 for the RECEIVE ROD TOKEN INFORMATION command.

The LIST IDENTIFIER field and ALLOCATION LENGTH field are defined in 6.20.

If no copy operation known to the copy manager has a list identifier that matches the contents of the LIST IDENTIFIER field, then the copy manager shall terminate the RECEIVE ROD TOKEN INFORMATION command with CHECK CONDITION status, with the sense key set to ILLEGAL REQUEST, and the additional sense code set to INVALID FIELD IN CDB.

The CONTROL byte is defined in SAM-5.

Table 273 shows the format of the parameter data for the RECEIVE ROD TOKEN INFORMATION command.

Table 273 — Parameter data for the RECEIVE ROD TOKEN INFORMATION command

Bit / Byte	7	6	5	4	3	2	1	0
0	(MSB)							
•••				AVAILABLE DATA (n-3)				
3								(LSB)
4		Reserved			RESPONSE TO SERVICE ACTION			
5	Reserved			COPY OPERATION STATUS				
6	(MSB)							
7				OPERATION COUNTER				(LSB)
8	(MSB)							
•••				ESTIMATED STATUS UPDATE DELAY				
11								(LSB)
12				EXTENDED COPY COMPLETION STATUS				
13				LENGTH OF THE SENSE DATA FIELD (x-31)				
14				SENSE DATA LENGTH				
15				TRANSFER COUNT UNITS				
16	(MSB)							
•••				TRANSFER COUNT				
23								(LSB)
24	(MSB)							
25				SEGMENTS PROCESSED				(LSB)
26								
•••				Reserved				
31								
32								
•••				SENSE DATA				
x								
x+1								
•••				ROD TOKEN DESCRIPTORS LENGTH (n-(x+4))				
x+4								
				ROD token descriptors				
x+5								
•••				ROD token descriptor (see table 274) [first]				
				⋮				
•••				ROD token descriptor (see table 274) [last]				
n								

The AVAILABLE DATA field, RESPONSE TO SERVICE ACTION field, COPY OPERATION STATUS field, OPERATION COUNTER field, ESTIMATED STATUS UPDATE DELAY field, EXTENDED COPY COMPLETION STATUS field, LENGTH OF THE SENSE DATA FIELD field, SENSE DATA LENGTH field, TRANSFER COUNT UNITS field, TRANSFER COUNT field, SEGMENTS PROCESSED field, and SENSE DATA field are defined in 6.24.

If the COPY OPERATION STATUS field is set to a value that table 252 (see 6.24) describes as meaning the operation completed without errors, then the ROD TOKEN DESCRIPTORS LENGTH field indicates the number of bytes that follow in ROD token descriptors. If the COPY OPERATION STATUS field is not set to a value that table 252 (see 6.24) describes as meaning the operation completed without errors, then the ROD TOKEN DESCRIPTORS LENGTH field shall be set to zero.

If the COPY OPERATION STATUS field is set to 01h or 03h, then each ROD token descriptor (see table 274) shall contain one ROD token created by the copy operation whose service action is indicated by the RESPONSE TO SERVICE ACTION field.

Table 274 — ROD token descriptor format

Bit / Byte	7	6	5	4	3	2	1	0
0	(MSB)							
1				ID FOR CREATING ROD CSCD DESCRIPTOR				(LSB)
2								
•••				ROD TOKEN				
n								

If the RESPONSE TO SERVICE ACTION field is set to 00h or 01h (i.e., if the ROD tokens being returned were created by an EXTENDED COPY command), then the ID FOR CREATING ROD CSCD DESCRIPTOR field indicates the CSCD descriptor ID for the ROD CSCD descriptor in which the R_TOKEN bit was set to one resulting in the creation of the ROD token contained in the ROD TOKEN field (i.e., the value any segment descriptor that caused the ROD to be populated had in its DESTINATION CSCD DESCRIPTOR ID field). If the RESPONSE TO SERVICE ACTION field does not contain 00h or 01h (i.e., if the ROD tokens being returned were created by a command other than the EXTENDED COPY command), then the ID FOR CREATING ROD CSCD DESCRIPTOR field is reserved.

The ROD TOKEN field contains a ROD token (see 5.16.6.4) created by the copy manager.

6.29 REMOVE I_T NEXUS command

The REMOVE I_T NEXUS command (see table 275) requests the device server to establish an I_T nexus loss (see SAM-5) for the logical unit containing the device server for each specified I_T nexus. This command uses the MAINTENANCE OUT CDB format (see 4.2.2.3.4).

Table 275 — REMOVE I_T NEXUS command

Bit / Byte	7	6	5	4	3	2	1	0
0	OPERATION CODE (A4h)							
1	Reserved			SERVICE ACTION (0Ch)				
2	Reserved							
•••								
5								
6	(MSB)			PARAMETER LIST LENGTH				
•••								
9								(LSB)
10	Reserved							
11	CONTROL							

The OPERATION CODE field is defined in 4.2.5.1 and shall be set as shown in table 275 for the REMOVE I_T NEXUS command.

The SERVICE ACTION field is defined in 4.2.5.2 and shall be set as shown in table 275 for the REMOVE I_T NEXUS command.

The PARAMETER LIST LENGTH field specifies the length in bytes of the parameter data that shall be transferred from the application client to the device server. A parameter list length value of zero specifies that no data shall be transferred and no I_T nexuses shall be removed.

The CONTROL byte is defined in SAM-5.

If the parameter list length results in the truncation of the header or any I_T nexus descriptor, then the device server:

a) shall not remove any I_T nexuses; and
b) shall terminate the command with CHECK CONDITION status, with the sense key set to ILLEGAL REQUEST, and the additional sense code set to PARAMETER LIST LENGTH ERROR.

Table 276 shows the format of the REMOVE I_T NEXUS parameter list.

Table 276 — REMOVE I_T NEXUS parameter list format

Bit / Byte	7	6	5	4	3	2	1	0
0	Reserved							
1								
2	(MSB)			I_T NEXUS DESCRIPTOR LIST LENGTH (n-3)				
3								(LSB)
	I_T nexus descriptor list							
4	I_T nexus descriptor [first]							
•••								
	⋮							
	I_T nexus descriptor [last]							
•••								
n								

The I_T NEXUS DESCRIPTOR LIST LENGTH field specifies the length in bytes of the I_T nexus descriptor list.

The I_T nexus descriptor list contains one or more I_T nexus descriptors (see table 277). The I_T nexus descriptors may appear in the I_T nexus descriptor list in any order. The device server shall process the I_T nexus descriptors in order. The device server shall ignore any I_T nexus descriptor that describes an I_T nexus not known to the logical unit.

The device server shall not remove any I_T nexuses and shall terminate the command with CHECK CONDITION status, with the sense key set to ILLEGAL REQUEST, and the additional sense code set to INVALID FIELD IN PARAMETER LIST if any of the I_T nexus descriptors describe:

a) the same I_T nexus as the one through which the REMOVE I_T NEXUS command is received; or
b) an I_T nexus that does not support the I_T NEXUS RESET task management function.

Table 277 — I_T nexus descriptor

Bit / Byte	7	6	5	4	3	2	1	0
0	(MSB)			RELATIVE TARGET PORT IDENTIFIER				
1								(LSB)
2	(MSB)			TRANSPORTID LENGTH (n-3)				
3								(LSB)
4	TRANSPORTID							
•••								
n								

The RELATIVE TARGET PORT IDENTIFIER field (see 4.3.4) specifies the relative target port identifier of the target port of the I_T nexus to be reset.

The TRANSPORTID LENGTH field specifies the length in bytes of the TRANSPORTID field.

The TRANSPORTID field specifies a TransportID (see 7.6.4) identifying the initiator port of the I_T nexus to be reset.

6.30 REPORT ALIASES command

The REPORT ALIASES command (see table 278) requests the device server to return the alias list. This command uses the MAINTENANCE IN CDB format (see 4.2.2.3.3). The alias list is managed using the CHANGE ALIASES command (see 6.2). If the CHANGE ALIASES command is supported, the REPORT ALIASES command shall also be supported.

Table 278 — REPORT ALIASES command

Bit / Byte	7	6	5	4	3	2	1	0
0	OPERATION CODE (A3h)							
1	Reserved			SERVICE ACTION (0Bh)				
2	Reserved							
•••								
5								
6	(MSB)			ALLOCATION LENGTH				
•••								
9								(LSB)
10	Reserved							
11	CONTROL							

The OPERATION CODE field is defined in 4.2.5.1 and shall be set as shown in table 278 for the REPORT ALIASES command.

The SERVICE ACTION field is defined in 4.2.5.2 and shall be set as shown in table 278 for the REPORT ALIASES command.

The ALLOCATION LENGTH field is defined in 4.2.5.6.

The CONTROL byte is defined in SAM-5.

The parameter data for a REPORT ALIASES command (see table 279) contains zero or more alias entries.

Table 279 — REPORT ALIASES parameter data

Bit / Byte	7	6	5	4	3	2	1	0
0	(MSB)							
•••				ADDITIONAL LENGTH (n-3)				
3								(LSB)
4				Reserved				
5				Reserved				
6	(MSB)							
7				NUMBER OF ALIASES				(LSB)
				Alias entry list				
8								
•••				Alias entry (see 6.2.2) [first]				
				⋮				
•••				Alias entry (see 6.2.2) [last]				
n								

The ADDITIONAL LENGTH field indicates the number of bytes in the remaining parameter data. The contents of the ADDITIONAL LENGTH field are not altered based on the allocation length (see 4.2.5.6).

The NUMBER OF ALIASES field indicates the number of alias entries in the alias list and shall not be changed if the CDB contains an insufficient allocation length.

The parameter data shall include one alias entry for each alias in the alias list. The format of an alias entry is described in 6.2.2.

6.31 REPORT ALL ROD TOKENS command

The REPORT ALL ROD TOKENS command (see table 280) is a third-party copy command (see 5.16.3) that requests the copy manager to return a ROD management token (see 5.16.6.4) for each ROD token that was created by and is known to the copy manager.

Table 280 — REPORT ALL ROD TOKENS command

Bit / Byte	7	6	5	4	3	2	1	0
0	OPERATION CODE (84h)							
1	Reserved			SERVICE ACTION (08h)				
2	Reserved							
•••								
9								
10	(MSB)			ALLOCATION LENGTH				
•••								
13								(LSB)
14	Reserved							
15	CONTROL							

The OPERATION CODE field is defined in 4.2.5.1 and shall be set as shown in table 280 for the REPORT ALL ROD TOKENS command.

The SERVICE ACTION field is defined in 4.2.5.2 and shall be set as shown in table 280 for the REPORT ALL ROD TOKENS command.

The ALLOCATION LENGTH field is defined in 4.2.5.6.

The CONTROL byte is defined in SAM-5.

In response to the REPORT ALL ROD TOKENS command, the copy manager shall return one or more ROD management tokens. Each ROD management token shall represent a ROD token that:

a) has fields defined in the ROD token body;
b) was created by the copy manager that is processing the REPORT ALL ROD TOKENS command; and
c) is known to that copy manager.

The format of the REPORT ALL ROD TOKENS parameter data is shown in table 281.

Table 281 — Parameter data for the REPORT ALL ROD TOKENS command

Bit / Byte	7	6	5	4	3	2	1	0
0	(MSB)							
•••	AVAILABLE DATA (n-3)							
3								(LSB)
4								
•••	Reserved							
7								
	ROD management token list							
8								
•••	ROD management token [first]							
103								
	⋮							
n-95								
•••	ROD management token [last]							
n								

The AVAILABLE DATA field shall contain the number of bytes that follow in the parameter data. The contents of the AVAILABLE DATA field are not altered based on the allocation length (see 4.2.5.6).

Each ROD management token is a 96-byte token that contains the fields described in 5.16.6.4, and represents a ROD token that meets the criteria described in this subclause.

6.32 REPORT IDENTIFYING INFORMATION command

6.32.1 REPORT IDENTIFYING INFORMATION command overview

The REPORT IDENTIFYING INFORMATION command (see table 282) requests the device server to return identifying information (see 5.6). This command uses the MAINTENANCE IN CDB format (see 4.2.2.3.3). The REPORT IDENTIFYING INFORMATION command is an extension to the REPORT PERIPHERAL DEVICE/COMPONENT DEVICE IDENTIFIER service action of the MAINTENANCE IN command defined in SCC-2.

The device server shall return the same identifying information regardless of the I_T nexus being used to transfer the identifying information.

Processing a REPORT IDENTIFYING INFORMATION command may require the enabling of a nonvolatile memory within the logical unit. If the nonvolatile memory is not ready, the device server shall terminate the command with CHECK CONDITION status, and not wait for the nonvolatile memory to become ready. The sense key shall be set to NOT READY and the additional sense code shall be set as described in table 336 (see 6.47). This information should allow the application client to determine the action required to cause the device server to become ready.

Table 282 — REPORT IDENTIFYING INFORMATION command

Bit / Byte	7	6	5	4	3	2	1	0
0	OPERATION CODE (A3h)							
1	Reserved			SERVICE ACTION (05h)				
2	Reserved							
3								
4	Restricted (see SCC-2)							
5								
6	(MSB)			ALLOCATION LENGTH				
•••								
9								(LSB)
10	IDENTIFYING INFORMATION TYPE							Reserved
11	CONTROL							

The OPERATION CODE field is defined in 4.2.5.1 and shall be set as shown in table 282 for the REPORT IDENTIFYING INFORMATION command.

The SERVICE ACTION field is defined in 4.2.5.2 and shall be set as shown in table 282 for the REPORT IDENTIFYING INFORMATION command.

The ALLOCATION LENGTH field is defined in 4.2.5.6.

The IDENTIFYING INFORMATION TYPE field (see table 283) specifies the type of identifying information to be returned. If the specified identifying information type is not implemented by the device server, the device server shall terminate the command with CHECK CONDITION status with the sense key set to ILLEGAL REQUEST, and the additional sense code set to INVALID FIELD IN CDB.

Table 283 — IDENTIFYING INFORMATION TYPE field

Code	Description	Reference
0000000b	Peripheral device identifying information (see 5.6).	6.32.2
0000010b	Peripheral device text identifying information (see 5.6).	6.32.2
1111110b	Identifying information supported – The parameter data contains a list of supported identifying information types and the maximum length of each.	6.32.3
xxxxxx1b	Restricted	SCC-2
All other	Reserved	

The CONTROL byte is defined in SAM-5.

6.32.2 Peripheral device identifying information parameter data

The peripheral device identifying information parameter data format (see table 284) is used if the IDENTIFYING INFORMATION TYPE field is set to 0000000b or 0000010b.

Table 284 — Peripheral device identifying information parameter data

Bit / Byte	7	6	5	4	3	2	1	0
0	Reserved							
1								
2	(MSB)	IDENTIFYING INFORMATION LENGTH (n-3)						
3								(LSB)
4	IDENTIFYING INFORMATION							
•••								
n								

The IDENTIFYING INFORMATION LENGTH field indicates the length in bytes of the IDENTIFYING INFORMATION field. The contents of the IDENTIFYING INFORMATION LENGTH field are not altered based on the allocation length (see 4.2.5.6).

The IDENTIFYING INFORMATION field contains the peripheral device identifying information that has the specified identifying information type (see 6.32.1).

6.32.3 Identifying information supported parameter data

The identifying information supported parameter data format (see table 285) is used if the IDENTIFYING INFORMATION TYPE field is set to 1111110b.

Table 285 — Identifying information supported parameter data

Bit / Byte	7	6	5	4	3	2	1	0
0	Reserved							
1								
2	(MSB)	IDENTIFYING INFORMATION LENGTH (n-3)						
3								(LSB)
	Identifying information descriptor list							
4	Identifying information descriptor [first] (see table 286)							
•••								
7								
	⋮							
n-3	Identifying information descriptor [last] (see table 286)							
•••								
n								

The IDENTIFYING INFORMATION LENGTH field indicates the length in bytes of the identifying information descriptor list. The contents of the IDENTIFYING INFORMATION LENGTH field are not altered based on the allocation length (see 4.2.5.6).

The identifying information descriptor list contains an identifying information descriptor (see table 286) for each identifying information type supported by the device server. The identifying information descriptors shall be sorted in increasing order based on the value in the IDENTIFYING INFORMATION TYPE field.

Table 286 — Identifying information descriptor

Bit / Byte	7	6	5	4	3	2	1	0
0	IDENTIFYING INFORMATION TYPE							Reserved
1	Reserved							
2	(MSB)			MAXIMUM IDENTIFYING INFORMATION LENGTH				
3								(LSB)

The IDENTIFYING INFORMATION TYPE field indicates the identifying information type (see table 283) associated with the descriptor.

The MAXIMUM IDENTIFYING INFORMATION LENGTH field indicates the maximum number of bytes supported for the identifying information type indicated the value in the IDENTIFYING INFORMATION TYPE field.

6.33 REPORT LUNS command

The REPORT LUNS command (see table 287) requests the device server to return the peripheral device logical unit inventory accessible to the I_T nexus. The logical unit inventory returned to the application client is a list that shall include the logical unit numbers of all logical units having a PERIPHERAL QUALIFIER value of 000b (see 6.6.2) based on the SELECT REPORT field. Logical unit numbers for logical units with PERIPHERAL QUALIFIER values other than 000b and 011b may be included in the logical unit inventory. Logical unit numbers for logical units with a PERIPHERAL QUALIFIER value of 011b shall not be included in the logical unit inventory.

Table 287 — REPORT LUNS command

Bit / Byte	7	6	5	4	3	2	1	0
0	OPERATION CODE (A0h)							
1	Reserved							
2	SELECT REPORT							
3	Reserved							
•••								
5								
6	(MSB)			ALLOCATION LENGTH				
•••								
9								(LSB)
10	Reserved							
11	CONTROL							

The OPERATION CODE field is defined in 4.2.5.1 and shall be set as shown in table 287 for the REPORT LUNS command.

The SELECT REPORT field (see table 288) specifies the types of logical unit addresses that shall be reported.

Table 288 — SELECT REPORT field

Code	Description
00h	The list shall contain the logical units accessible to the I_T nexus with the following addressing methods (see SAM-5): a) simple logical unit addressing method; b) logical unit addressing method; c) peripheral device addressing method; d) flat space addressing method; e) extended flat space addressing method; and f) long extended flat space addressing method. If there are no logical units to report, the LUN LIST LENGTH field shall be set to zero.
01h	The list shall contain only well known logical units, if any. If there are no well known logical units, the LUN LIST LENGTH field shall be zero.
02h	The list shall contain all logical units accessible to the I_T nexus.
10h	If the device server processing the command is in LUN 0 or the REPORT LUNS well-known logical unit, then the list shall contain only administrative logical units (see SAM-5). The LUN LIST LENGTH field shall be set to zero if the device server processing the command is not in: a) LUN 0; or b) the REPORT LUNS well-known logical unit. If there are no logical units to report, the LUN LIST LENGTH field shall be set to zero.
11h	If the device server processing the command is in LUN 0 or the REPORT LUNS well-known logical unit, then the list shall contain only: a) administrative logical units (see SAM-5); b) logical units with the logical unit addressing method at level 1; and c) logical units with single level LUN structure with the following addressing methods (see SAM-5): A) peripheral device addressing method; B) flat space addressing method; C) extended flat space addressing method; and D) long extended flat space addressing method. The LUN LIST LENGTH field shall be set to zero if the device server processing the command is not in: a) LUN 0; or b) the REPORT LUNS well-known logical unit. If there are no logical units to report, the LUN LIST LENGTH field shall be set to zero.
12h	If the device server processing the command is in an administrative logical unit, the list shall contain: a) the logical unit processing the command; and b) subsidiary logical units that are contained in the same logical unit conglomerate that contains the logical unit processing the command. The LUN LIST LENGTH field shall be set to zero if the device server processing the command is not in an administrative logical unit. If there are no logical units to report, the LUN LIST LENGTH field shall be set to zero.
F8h to FFh	Vendor specific
all others	Reserved

The ALLOCATION LENGTH field is defined in 4.2.5.6.

The CONTROL byte is defined in SAM-5.

The REPORT LUNS command shall return CHECK CONDITION status only if the device server is unable to return the requested report of the logical unit inventory.

If a REPORT LUNS command is received from an I_T nexus with a pending unit attention condition (i.e., before the device server reports CHECK CONDITION status), then the device server shall perform the REPORT LUNS command (see SAM-5).

The REPORT LUNS parameter data should be returned even though the device server is not ready for other commands. The report of the logical unit inventory should be available without incurring any media access delays. If the device server is not ready with the logical unit inventory or if the inventory list is null for the requesting I_T nexus and the SELECT REPORT field set to 02h, then the device server shall provide a default logical unit inventory that contains at least LUN 0 or the REPORT LUNS well known logical unit (see 8.2). A non-empty peripheral device logical unit inventory that does not contain either LUN 0 or the REPORT LUNS well known logical unit is valid.

If a REPORT LUNS command is received for a logical unit that the SCSI target device does not support and the device server is not capable of returning the logical unit inventory, then the device server shall terminate the command with CHECK CONDITION status, with the sense key set to ILLEGAL REQUEST, and the additional sense code set to LOGICAL UNIT NOT SUPPORTED.

The device server shall report those devices in the logical unit inventory using the format shown in table 289.

Table 289 — REPORT LUNS parameter data format

Bit / Byte	7	6	5	4	3	2	1	0
0	(MSB)							
•••	LUN LIST LENGTH (n-7)							
3								(LSB)
4	Reserved							
•••								
7								
	LUN list							
8	LUN [first]							
•••								
15								
	⋮							
n-7	LUN [last]							
•••								
n								

The LUN LIST LENGTH field shall contain the length in bytes of the LUN list. The LUN list length is the number of logical unit numbers in the logical unit inventory multiplied by eight. The contents of the LUN LIST LENGTH field are not altered based on the allocation length (see 4.2.5.6).

6.34 REPORT PRIORITY command

The REPORT PRIORITY command (see table 290) requests the device server to return the command priorities (see SAM-5) that have been assigned to one or more I_T nexuses associated with the logical unit (i.e., I_T_L nexuses). This command uses the MAINTENANCE IN CDB format (see 4.2.2.3.3).

Table 290 — REPORT PRIORITY command

Bit / Byte	7	6	5	4	3	2	1	0
0	OPERATION CODE (A3h)							
1	Reserved				SERVICE ACTION (0Eh)			
2	PRIORITY REPORTED			Reserved				
3	Reserved							
•••								
5								
6	(MSB)							
•••	ALLOCATION LENGTH							
9								(LSB)
10	Reserved							
11	CONTROL							

The OPERATION CODE field is defined in 4.2.5.1 and shall be set as shown in table 290 for the REPORT PRIORTY command.

The SERVICE ACTION field is defined in 4.2.5.2 and shall be set as shown in table 290 for the REPORT PRIORTY command.

The PRIORITY REPORTED field (see table 291) specifies the information to be returned in the parameter data.

Table 291 — PRIORITY REPORTED field

Code	Description
00b	Only the priority for the I_T nexus on which the command was received shall be reported in the REPORT PRIORITY parameter data.
01b	The priority for each I_T nexus that is not set to the initial command priority shall be reported in the REPORT PRIORITY parameter data.
10b to 11b	Reserved

The ALLOCATION LENGTH field is defined in 4.2.5.6. The allocation length should be at least four.

The CONTROL byte is defined in SAM-5.

The format of the parameter data for the REPORT PRIORITY command is shown in table 292.

Table 292 — REPORT PRIORITY parameter data format

Bit / Byte	7	6	5	4	3	2	1	0
0	(MSB)							
•••				PRIORITY PARAMETER DATA LENGTH (n-3)				
3								(LSB)
	Priority descriptors							
4								
•••				Priority descriptor (see table 293) [first]				
				⋮				
•••				Priority descriptor (see table 293) [last]				
n								

The PRIORITY PARAMETER DATA LENGTH field indicates the number of bytes of parameter data that follow. The contents of the PRIORITY PARAMETER DATA LENGTH field are not altered based on the allocation length (see 4.2.5.6).

Each priority descriptor (see table 293) contains priority information for a single I_T_L nexus.

Table 293 — Priority descriptor format

Bit / Byte	7	6	5	4	3	2	1	0
0				Reserved	CURRENT PRIORITY			
1				Reserved				
2	(MSB)							
3				RELATIVE TARGET PORT IDENTIFIER				(LSB)
4				Reserved				
5				Reserved				
6	(MSB)							
7				ADDITIONAL DESCRIPTOR LENGTH (n-7)				(LSB)
8								
•••				TransportID				
n								

The CURRENT PRIORITY field indicates the priority assigned to the I_T_L nexus represented by this descriptor. If the PRIORITY REPORTED field in this command is set to 00b and the priority for the I_T_L nexus associated with this command is set to the initial command priority, then the CURRENT PRIORITY field shall be set to zero. The priority assigned to an I_T_L nexus may be used as a command priority for commands received via that I_T_L nexus (see SAM-5).

The RELATIVE TARGET PORT IDENTIFIER field (see 4.3.4) indicates the relative port identifier of the target port that is part of the I_T_L nexus to which the current priority applies.

The ADDITIONAL DESCRIPTOR LENGTH field indicates the number of bytes that follow in the descriptor (i.e., the size of the TransportID).

The TransportID specifies a TransportID (see 7.6.4) identifying the initiator port that is part of the I_T_L nexus to which the current priority applies.

6.35 REPORT SUPPORTED OPERATION CODES command

6.35.1 REPORT SUPPORTED OPERATION CODES command introduction

The REPORT SUPPORTED OPERATION CODES command (see table 294) requests the device server to return information on commands the addressed logical unit supports. This command uses the MAINTENANCE IN CDB format (see 4.2.2.3.3). An application client may request a list of all operation codes and service actions supported by the logical unit or the command support data for a specific command.

Table 294 — REPORT SUPPORTED OPERATION CODES command

Bit / Byte	7	6	5	4	3	2	1	0
0	OPERATION CODE (A3h)							
1	Reserved			SERVICE ACTION (0Ch)				
2	RCTD	Reserved				REPORTING OPTIONS		
3	REQUESTED OPERATION CODE							
4	(MSB)	REQUESTED SERVICE ACTION						
5								(LSB)
6	(MSB)	ALLOCATION LENGTH						
...								
9								(LSB)
10	Reserved							
11	CONTROL							

The OPERATION CODE field is defined in 4.2.5.1 and shall be set as shown in table 294 for the REPORT SUPPORTED OPERATION CODES command.

The SERVICE ACTION field is defined in 4.2.5.2 and shall be set as shown in table 294 for the REPORT SUPPORTED OPERATION CODES command.

A return command timeouts descriptor (RCTD) bit set to one specifies that the command timeouts descriptor (see 6.35.4) shall be included in each command descriptor (see 6.35.2) that is returned or in the one_command parameter data (see 6.35.3) that is returned. A RCTD bit set to zero specifies that the command timeouts descriptor shall not be returned.

The REPORTING OPTIONS field (see table 295) specifies the information to be returned in the parameter data.

Table 295 — REPORT SUPPORTED OPERATION CODES REPORTING OPTIONS field

Code	Description	Parameter data reference
000b	A list of all operation codes and service actions [a] supported by the logical unit shall be returned in the all_commands parameter data format. The REQUESTED OPERATION CODE field and REQUESTED SERVICE ACTION field shall be ignored.	6.35.2
001b	The command support data for the operation code specified in the REQUESTED OPERATION CODE field shall be returned in the one_command parameter data format. The REQUESTED SERVICE ACTION field shall be ignored. If the REQUESTED OPERATION CODE field specifies an operation code for which the device server implements service actions [a], then the device server shall terminate the command with CHECK CONDITION status, with the sense key set to ILLEGAL REQUEST, and the additional sense code set to INVALID FIELD IN CDB.	6.35.3
010b	The command support data for the operation code and service action [a] specified in the REQUESTED OPERATION CODE field and REQUESTED SERVICE ACTION field shall be returned in the one_command parameter data format. If the REQUESTED OPERATION CODE field specifies an operation code for which the device server does not implement service actions [a], then the device server shall terminate the command with CHECK CONDITION status, with the sense key set to ILLEGAL REQUEST, and the additional sense code set to INVALID FIELD IN CDB.	6.35.3
011b	The command support data for the operation code and service action [a] specified in the REQUESTED OPERATION CODE field and REQUESTED SERVICE ACTION field shall be returned in the one_command parameter data format. If: a) the operation code specified by the REQUESTED OPERATION CODE field specifies an operation code for which the device server does not implement service actions, the REQUESTED SERVICE ACTION field is set to 00h, and the command is supported; or b) the operation code specified by the REQUESTED OPERATION CODE field specifies an operation code for which the device server implements service actions and the value in the REQUESTED SERVICE ACTION field is supported, then the command support data shall indicate that the command is supported (i.e., the SUPPORT field (see table 298) is set to 011b or 101b). Otherwise, the command support data shall indicate that the command is not supported (i.e., the SUPPORT field is set to 001b).	6.35.3
100b to 111b	Reserved	

[a] The device server should also process the following fields in the following commands as specifying service actions (i.e., the specified field should be processed as a SERVICE ACTION field):
a) the MODE field in the READ BUFFER command (see 6.18); and
b) the MODE field in the WRITE BUFFER command (see 6.49).

The REQUESTED OPERATION CODE field specifies the operation code of the command to be returned in the one_command parameter data format (see 6.35.3).

The REQUESTED SERVICE ACTION field specifies the service action of the command to be returned in the one_command parameter data format.

The ALLOCATION LENGTH field is defined in 4.2.5.6.

The CONTROL byte is defined in SAM-5.

6.35.2 All_commands parameter data format

The REPORT SUPPORTED OPERATION CODES all_commands parameter data format (see table 296) begins with a four-byte header that contains the length in bytes of the parameter data followed by a list of supported commands. Each command descriptor contains information about a single supported command CDB (i.e., one operation code and service action combination, or one non-service-action operation code). The list of command descriptors shall contain all commands supported by the logical unit.

Table 296 — All_commands parameter data

Bit / Byte	7	6	5	4	3	2	1	0
0	(MSB)							
•••				COMMAND DATA LENGTH (n-3)				
3								(LSB)
	Command descriptors							
4								
•••				Command descriptor (see table 297) [first]				
				⋮				
•••				Command descriptor (see table 297) [last]				
n								

The COMMAND DATA LENGTH field indicates the length in bytes of the command descriptor list. The contents of the COMMAND DATA LENGTH field are not altered based on the allocation length (see 4.2.5.6).

Each command descriptor (see table 297) contains information about a single supported command CDB.

Table 297 — Command descriptor format

Bit / Byte	7	6	5	4	3	2	1	0
0	OPERATION CODE							
1	Reserved							
2	(MSB)							
3	SERVICE ACTION							(LSB)
4	Reserved							
5	Reserved						CTDP	SERVACTV
6	(MSB)							
7	CDB LENGTH							(LSB)
8								
•••	Command timeouts descriptor, if any (see 6.35.4)							
19								

The OPERATION CODE field indicates the operation code of a command supported by the logical unit.

The SERVICE ACTION field indicates a supported service action of the supported operation code indicated by the OPERATION CODE field. If the operation code indicated in the OPERATION CODE field does not have any service actions, the SERVICE ACTION field shall be set to 00h.

A command timeouts descriptor present (CTDP) bit set to one indicates that the command timeouts descriptor (see 6.35.4) is included in this command descriptor. A CTDP bit set to zero indicates that the command timeouts descriptor is not included in this command descriptor.

A service action valid (SERVACTV) bit set to zero indicates the operation code indicated by the OPERATION CODE field does not have service actions and the SERVICE ACTION field contents are reserved. A SERVACTV bit set to one indicates the operation code indicated by the OPERATION CODE field has service actions and the contents of the SERVICE ACTION field are valid.

The CDB LENGTH field indicates the length of the command CDB in bytes for the operation code indicated in the OPERATION CODE field, and if the SERVACTV bit is set to one the service action indicated by the SERVICE ACTION field.

The command timeouts descriptor is described in 6.35.4.

6.35.3 One_command parameter data format

The REPORT SUPPORTED OPERATION CODES one_command parameter data format (see table 298) contains information about the CDB and a usage map for bits in the CDB for the command specified by the REPORTING OPTIONS field, REQUESTED OPERATION CODE field, and REQUESTED SERVICE ACTION field in the REPORT SUPPORTED OPERATION CODES CDB.

Table 298 — One_command parameter data

Bit / Byte	7	6	5	4	3	2	1	0
0	Reserved							
1	CTDP	Reserved				SUPPORT		
2	(MSB)							
3	CDB SIZE (n-3)							(LSB)
4	CDB USAGE DATA							
•••								
n								
n+1	Command timeouts descriptor, if any (see 6.35.4)							
•••								
n+12								

A command timeouts descriptor present (CTDP) bit set to one indicates that the command timeouts descriptor (see 6.35.4) is included in the parameter data. A CTDP bit set to zero indicates that the command timeouts descriptor is not included in the parameter data.

The SUPPORT field is defined in table 299.

Table 299 — SUPPORT values

Support	Description
000b	Data about the requested SCSI command is not currently available. No data after byte one is valid. A subsequent request for command support data may be successful.
001b	The device server does not support the requested command. Data after byte one is undefined.
010b	Reserved
011b	The device server supports the requested command in conformance with a SCSI standard. The parameter data format conforms to the definition in table 298.
100b	Reserved
101b	The device server supports the requested command in a vendor specific manner. The parameter data format conforms to the definition in table 298.
110b to 111b	Reserved

The CDB SIZE field indicates the size of the CDB USAGE DATA field in the parameter data, and the number of bytes in the CDB for command being queried (i.e., the command specified by the REPORTING OPTIONS field, REQUESTED OPERATION CODE field, and REQUESTED SERVICE ACTION field in the REPORT SUPPORTED OPERATION CODES CDB).

The CDB USAGE DATA field contains information about the CDB for the command being queried. The first byte of the CDB USAGE DATA field shall contain the operation code for the command being queried. If the command being queried contains a service action, then that service action code shall be placed in the CDB USAGE DATA field in the same location as the SERVICE ACTION field of the command CDB. All other bytes of the CDB USAGE DATA field shall contain a usage map for bits in the CDB for the command being queried.

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

For example, the CDB usage bit map for the REPORT SUPPORTED OPERATION CODES command is: A3h, 0Ch, 87h, FFh, FFh, FFh, FFh, FFh, FFh, FFh, 00h, 07h. This example assumes that the logical unit only supports the low-order three bits of the CDB CONTROL byte. The first byte contains the operation code, and the second byte contains three reserved bits and the service action. The remaining bytes contain the usage map.

The command timeouts descriptor is described in 6.35.4.

6.35.4 Command timeouts descriptor

6.35.4.1 Overview

The command timeouts descriptor (see table 300) contains timeout information for commands supported by the logical unit based on the time from the start of processing for the command to its reported completion.

Values contained in the command timeouts descriptor do not include times that are outside the control of the device server (e.g., prior commands with the IMMED bit set to one in the CDB, concurrent commands from the same or different I_T nexuses, manual unloads, power on self tests, prior aborted commands, commands that

force cache synchronization, delays in the service delivery subsystem, time in the task set before the start of processing).

For commands that cause a change in power condition (see 5.11), values contained in the command timeouts descriptor do not include the power condition transition time (e.g., the time to spinup rotating media).

Values contained in the command timeouts descriptor should not be used to compare products.

Table 300 — Command timeouts descriptor format

Bit / Byte	7	6	5	4	3	2	1	0
0	(MSB)			DESCRIPTOR LENGTH (000Ah)				
1								(LSB)
2	Reserved							
3	COMMAND SPECIFIC							
4	(MSB)			NOMINAL COMMAND PROCESSING TIMEOUT				
•••								
7								(LSB)
8	(MSB)			RECOMMENDED COMMAND TIMEOUT				
•••								
11								(LSB)

The DESCRIPTOR LENGTH field indicates the number of bytes that follow in the command timeouts descriptor. The contents of the DESCRIPTOR LENGTH field are not altered based on the allocation length (see 4.2.5.6).

The COMMAND SPECIFIC field contains timeout information that is defined for a specific command (e.g., the WRITE BUFFER command (see 6.35.4.2)). If no command specific timeout information is defined by this or the applicable command standard the COMMAND SPECIFIC field is reserved.

A non-zero value in the NOMINAL COMMAND PROCESSING TIMEOUT field indicates the minimum amount of time in seconds the application client should wait prior to querying for the progress of the command identified by the parameter data that contains this command timeouts descriptor. A value of zero in the NOMINAL COMMAND PROCESSING TIMEOUT field indicates that no timeout is indicated.

NOTE 37 - The value contained in the NOMINAL COMMAND PROCESSING TIMEOUT field may include time required for typical device error recovery procedures expected to occur on a regular basis.

A non-zero value in the RECOMMENDED COMMAND TIMEOUT field specifies the recommended time in seconds the application client should wait prior to timing out the command identified by the parameter data that contains this command timeouts descriptor. A value of zero in the RECOMMENDED COMMAND TIMEOUT field indicates that no time is indicated.

The device server should set the recommended command timeout to a value greater than or equal to the nominal command processing timeout.

6.35.4.2 WRITE BUFFER command timeouts descriptor COMMAND SPECIFIC field usage

For the WRITE BUFFER command (see 6.49), the COMMAND SPECIFIC field usage is reserved for all modes except the following:

a) download microcode mode (04h);

b) download microcode and save mode (05h);
c) download microcode with offsets mode (06h);
d) download microcode with offsets and save mode (07h);
e) download microcode with offsets, select activation events, save, and defer activate mode (0Dh) only if the microcode is activated by an event other than the activate deferred microcode mode;
f) download microcode with offsets, save, and defer activate mode (0Eh) only if the microcode is activated by an event other than an activate deferred microcode mode; and
g) activate deferred microcode mode (0Fh).

If the command timeouts descriptor describes one of the WRITE BUFFER modes listed in this subclause, then the COMMAND SPECIFIC field indicates the maximum time, in one second increments, that access to the SCSI device is limited or not possible through any SCSI ports associated with a logical unit that processes a WRITE BUFFER command that specifies one of the named modes. A value of zero in the COMMAND SPECIFIC field indicates that no maximum time is indicated.

6.36 REPORT SUPPORTED TASK MANAGEMENT FUNCTIONS command

The REPORT SUPPORTED TASK MANAGEMENT FUNCTIONS command (see table 301) requests the device server to return on task management functions (see SAM-5) the addressed logical unit supports. This command uses the MAINTENANCE IN CDB format (see 4.2.2.3.3).

Table 301 — REPORT SUPPORTED TASK MANAGEMENT FUNCTIONS command

Bit / Byte	7	6	5	4	3	2	1	0
0	OPERATION CODE (A3h)							
1	Reserved			SERVICE ACTION (0Dh)				
2	REPD	Reserved						
3	Reserved							
•••								
5								
6	(MSB)	ALLOCATION LENGTH						
•••								
9								(LSB)
10	Reserved							
11	CONTROL							

The OPERATION CODE field is defined in 4.2.5.1 and shall be set as shown in table 301 for the REPORT SUPPORTED TASK MANAGEMENT FUNCTIONS command.

The SERVICE ACTION field is defined in 4.2.5.2 and shall be set as shown in table 301 for the REPORT SUPPORTED TASK MANAGEMENT FUNCTIONS command.

A return extended parameter data (REPD) bit set to one specifies that the task management timeout information shall be included in the parameter data that is returned. A REPD bit set to zero specifies that the task management timeout information shall not be returned.

The ALLOCATION LENGTH field is defined in 4.2.5.6. The allocation length should be at least four.

The CONTROL byte is defined in SAM-5.

The format of the parameter data for the REPORT SUPPORTED TASK MANAGEMENT FUNCTIONS command depends on the value of the REPD bit as follows:

a) if the REPD bit is set to zero, the REPORT SUPPORTED TASK MANAGEMENT FUNCTION parameter data returned is shown in table 302; and
b) if the REPD bit is set to one, the REPORT SUPPORTED TASK MANAGEMENT FUNCTION parameter data returned is shown in table 303.

The REPORT SUPPORTED TASK MANAGEMENT FUNCTIONS basic parameter data format is shown in table 302.

Table 302 — REPORT SUPPORTED TASK MANAGEMENT FUNCTIONS basic parameter data

Bit / Byte	7	6	5	4	3	2	1	0
0	ATS	ATSS	CACAS	CTSS	LURS	QTS	Obsolete	Obsolete
1	Reserved					QAES	QTSS	ITNRS
2	Reserved							
3	REPORT SUPPORTED TASK MANAGEMENT FUNCTIONS ADDITIONAL DATA LENGTH (00h)							

The REPORT SUPPORTED TASK MANAGEMENT FUNCTIONS extended parameter data format is shown in table 303.

Table 303 — REPORT SUPPORTED TASK MANAGEMENT FUNCTIONS extended parameter data

Bit / Byte	7	6	5	4	3	2	1	0
0	ATS	ATSS	CACAS	CTSS	LURS	QTS	Obsolete	Obsolete
1	Reserved					QAES	QTSS	ITNRS
2	Reserved							
3	REPORT SUPPORTED TASK MANAGEMENT FUNCTIONS ADDITIONAL DATA LENGTH (0Ch)							
4	Reserved							TMFTMOV
5	Reserved							
6	ATTS	ATSTS	CACATS	CTSTS	LURTS	QTTS	Reserved	Reserved
7	Reserved					QAETS	QTSTS	ITNRTS
8	(MSB)							
•••	TASK MANAGEMENT FUNCTIONS LONG TIMEOUT							
11								(LSB)
12	(MSB)							
•••	TASK MANAGEMENT FUNCTIONS SHORT TIMEOUT							
15								(LSB)

An ABORT TASK supported (ATS) bit set to one indicates the ABORT TASK task management function (see SAM-5) is supported by the logical unit. An ATS bit set to zero indicates the ABORT TASK task management function is not supported.

An ABORT TASK SET supported (ATSS) bit set to one indicates the ABORT TASK SET task management function (see SAM-5) is supported by the logical unit. An ATSS bit set to zero indicates the ABORT TASK SET task management function is not supported.

A CLEAR ACA supported (CACAS) bit set to one indicates the CLEAR ACA task management function (see SAM-5) is supported by the logical unit. A CACAS bit set to zero indicates the CLEAR ACA task management function is not supported.

A CLEAR TASK SET supported (CTSS) bit set to one indicates the CLEAR TASK SET task management function (see SAM-5) is supported by the logical unit. A CTSS bit set to zero indicates the CLEAR TASK SET task management function is not supported.

A LOGICAL UNIT RESET supported (LURS) bit set to one indicates the LOGICAL UNIT RESET task management function (see SAM-5) is supported by the logical unit. An LURS bit set to zero indicates the LOGICAL UNIT RESET task management function is not supported.

A QUERY TASK supported (QTS) bit set to one indicates the QUERY TASK task management function (see SAM-5) is supported by the logical unit. A QTS bit set to zero indicates the QUERY TASK task management function is not supported.

A QUERY ASYNCHRONOUS EVENT supported (QAES) bit set to one indicates the QUERY ASYNCHRONOUS EVENT task management function (see SAM-5) is supported by the logical unit. A QAES bit set to zero indicates the QUERY ASYNCHRONOUS EVENT task management function is not supported.

A QUERY TASK SET supported (QTSS) bit set to one indicates the QUERY TASK SET task management function (see SAM-5) is supported by the logical unit. A QTSS bit set to zero indicates the QUERY TASK SET task management function is not supported.

An I_T NEXUS RESET supported (ITNRS) bit set to one indicates the I_T NEXUS RESET task management function (see SAM-5) is supported by the logical unit. An ITNRS bit set to zero indicates the I_T NEXUS RESET task management function is not supported.

The REPORT SUPPORTED TASK MANAGEMENT FUNCTIONS ADDITIONAL DATA LENGTH field indicates the number of bytes that follow in the parameter data. The contents of the REPORT SUPPORTED TASK MANAGEMENT FUNCTIONS ADDITIONAL DATA LENGTH field are not altered based on the allocation length (see 4.2.5.6).

A task management function timeouts valid (TMFTMOV) bit set to one indicates the contents of the TASK MANAGEMENT FUNCTIONS SHORT TIMEOUT field and TASK MANAGEMENT FUNCTIONS LONG TIMEOUT field are valid. A TMFTMOV bit set to zero indicates the contents of the TASK MANAGEMENT FUNCTIONS SHORT TIMEOUT field and TASK MANAGEMENT FUNCTIONS LONG TIMEOUT field are not valid and should be ignored.

An ABORT TASK timeout selector (ATTS) bit set to one indicates that the value in the TASK MANAGEMENT FUNCTIONS SHORT TIMEOUT field applies to the ABORT TASK task management function. An ATTS bit set to zero indicates that the value in the TASK MANAGEMENT FUNCTIONS LONG TIMEOUT field applies to the ABORT TASK task management function.

An ABORT TASK SET timeout selector (ATSTS) bit set to one indicates that the value in the TASK MANAGEMENT FUNCTIONS SHORT TIMEOUT field applies to the ABORT TASK SET task management function. An ATSTS bit set to zero indicates that the value in the TASK MANAGEMENT FUNCTIONS LONG TIMEOUT field applies to the ABORT TASK SET task management function.

A CLEAR ACA timeout selector (CACATS) bit set to one indicates that the value in the TASK MANAGEMENT FUNCTIONS SHORT TIMEOUT field applies to the CLEAR ACA task management function. A CACATS bit set to zero indicates that the value in the TASK MANAGEMENT FUNCTIONS LONG TIMEOUT field applies to the CLEAR ACA task management function.

A CLEAR TASK SET timeout selector (CTSTS) bit set to one indicates that the value in the TASK MANAGEMENT FUNCTIONS SHORT TIMEOUT field applies to the CLEAR TASK SET task management function. A CTSTS bit set to zero indicates that the value in the TASK MANAGEMENT FUNCTIONS LONG TIMEOUT field applies to the CLEAR TASK SET task management function.

A LOGICAL UNIT RESET timeout selector (LURTS) bit set to one indicates that the value in the TASK MANAGEMENT FUNCTIONS SHORT TIMEOUT field applies to the LOGICAL UNIT RESET task management function. A LURTS bit set to zero indicates that the value in the TASK MANAGEMENT FUNCTIONS LONG TIMEOUT field applies to the LOGICAL UNIT RESET task management function.

A QUERY TASK timeout selector (QTTS) bit set to one indicates that the value in the TASK MANAGEMENT FUNCTIONS SHORT TIMEOUT field applies to the QUERY TASK task management function. A QTTS bit set to zero indicates that the value in the TASK MANAGEMENT FUNCTIONS LONG TIMEOUT field applies to the QUERY TASK task management function.

A QUERY ASYNCHRONOUS EVENT timeout selector (QAETS) bit set to one indicates that the value in the TASK MANAGEMENT FUNCTIONS SHORT TIMEOUT field applies to the QUERY ASYNCHRONOUS EVENT task management function. A QAETS bit set to zero indicates that the value in the TASK MANAGEMENT FUNCTIONS LONG TIMEOUT field applies to the QUERY ASYNCHRONOUS EVENT task management function.

A QUERY TASK SET timeout selector (QTSTS) bit set to one indicates that the value in the TASK MANAGEMENT FUNCTIONS SHORT TIMEOUT field applies to the QUERY TASK SET task management function. A QTSTS bit set to zero indicates that the value in the TASK MANAGEMENT FUNCTIONS LONG TIMEOUT field applies to the QUERY TASK SET task management function.

An I_T NEXUS RESET timeout selector (ITNRTS) bit set to one indicates that the value in the TASK MANAGEMENT FUNCTIONS SHORT TIMEOUT field applies to the I_T NEXUS RESET task management function. An ITNRTS bit set to zero indicates that the value in the TASK MANAGEMENT FUNCTIONS LONG TIMEOUT field applies to the I_T NEXUS RESET task management function.

If the TMFTMOV bit is set to one and the TASK MANAGEMENT FUNCTIONS LONG TIMEOUT field is not set to zero, then the contents of the TASK MANAGEMENT FUNCTIONS LONG TIMEOUT field indicate the recommended time in 100 millisecond increments that the application client should wait prior to timing out a task management function for which the applicable selector bit is set to zero. If the TMFTMOV bit is set to zero or the TASK MANAGEMENT FUNCTIONS LONG TIMEOUT field is set to zero, then the recommended timeout is unspecified for any task management function for which the applicable selector bit is set to zero.

If the TMFTMOV bit is set to one and the TASK MANAGEMENT FUNCTIONS SHORT TIMEOUT field is not set to zero, then the contents of the TASK MANAGEMENT FUNCTIONS SHORT TIMEOUT field indicate the recommended time in 100 millisecond increments that the application client should wait prior to timing out a task management function for which the applicable selector bit is set to one. If the TMFTMOV bit is set to zero or the TASK MANAGEMENT FUNCTIONS SHORT TIMEOUT field is set to zero, then the recommended timeout is unspecified for any task management function for which the applicable selector bit is set to one.

6.37 REPORT TARGET PORT GROUPS command

The REPORT TARGET PORT GROUPS command (see table 304) requests the device server to return target port group information. This command uses the MAINTENANCE IN CDB format (see 4.2.2.3.3). This command shall be supported by logical units that report in the standard INQUIRY data (see 6.6.2) that they support asymmetric logical unit access (i.e., return a non-zero value in the TPGS field).

Table 304 — REPORT TARGET PORT GROUPS command

Bit / Byte	7	6	5	4	3	2	1	0
0	OPERATION CODE (A3h)							
1	PARAMETER DATA FORMAT			SERVICE ACTION (0Ah)				
2	Reserved							
•••								
5								
6	(MSB)			ALLOCATION LENGTH				
•••								
9								(LSB)
10	Reserved							
11	CONTROL							

The OPERATION CODE field is defined in 4.2.5.1 and shall be set as shown in table 304 for the REPORT TARGET PORT GROUPS command.

The SERVICE ACTION field is defined in 4.2.5.2 and shall be set as shown in table 304 for the REPORT TARGET PORT GROUPS command.

The PARAMETER DATA FORMAT field (see table 305) specifies the requested format for the parameter data returned by the REPORT TARGET PORT GROUPS command. The device server may ignore the PARAMETER DATA FORMAT field. If the device server ignores the PARAMETER DATA FORMAT field, the device server shall return the parameter data format defined in table 306.

Table 305 — PARAMETER DATA FORMAT field

Code	Description	Reference
000b	Length only header parameter data format	table 306
001b	Extended header parameter data format	table 307
010b to 111b	Reserved	

The ALLOCATION LENGTH field is defined in 4.2.5.6.

The CONTROL byte is defined in SAM-5.

Returning REPORT TARGET PORT GROUPS parameter data may require the enabling of a nonvolatile memory. If the nonvolatile memory is not ready, the device server shall terminate the command with CHECK CONDITION status, rather than wait for the nonvolatile memory to become ready. The sense key shall be set to NOT READY and the additional sense code shall be set as described in table 336 (see 6.47).

The length only header format of the parameter data for the REPORT TARGET PORT GROUPS command is shown in table 306.

Table 306 — Length only header parameter data format

Bit / Byte	7	6	5	4	3	2	1	0
0	(MSB)							
•••	RETURN DATA LENGTH (n-3)							
3								(LSB)
	Target port group descriptor list							
4								
•••	Target port group descriptor (see table 308) [first]							
	⋮							
•••	Target port group descriptor (see table 308) [last]							
n								

The RETURN DATA LENGTH field indicates the length in bytes of the list of target port group descriptors. The contents of the RETURN DATA LENGTH field are not altered based on the allocation length (see 4.2.5.6).

There shall be one target port group descriptor (see table 308) for each target port group.

The extended header format of the parameter data for the REPORT TARGET PORT GROUPS command is shown in table 307.

Table 307 — Extended header parameter data format

Bit / Byte	7	6	5	4	3	2	1	0
0	(MSB)							
•••	RETURN DATA LENGTH (n-3)							
3								(LSB)
4	Reserved	RTPG_FMT (001b)			Reserved			
5	IMPLICIT TRANSITION TIME							
6	Reserved							
7								
	Target port group descriptor list							
8								
•••	Target port group descriptor (see table 308) [first]							
	⋮							
•••	Target port group descriptor (see table 308) [last]							
n								

The RETURN DATA LENGTH field indicates the length in bytes of the list of target port groups. The contents of the RETURN DATA LENGTH field are not altered based on the allocation length (see 4.2.5.6).

The report target port groups format (RTPG_FMT) field indicates the returned parameter data format and shall be set as shown in table 307 for the extended header parameter data format.

The IMPLICIT TRANSITION TIME field indicates the minimum amount of time in seconds the application client should wait prior to timing out an implicit state transition (see 5.15.2.2). A value of zero indicates that the implicit transition time is not specified.

There shall be one target port group descriptor (see table 308) for each target port group.

Table 308 — Target port group descriptor format

Bit / Byte	7	6	5	4	3	2	1	0
0	PREF	RTPG_FMT (000b)			ASYMMETRIC ACCESS STATE			
1	T_SUP	O_SUP	Reserved	LBD_SUP	U_SUP	S_SUP	AN_SUP	AO_SUP
2	(MSB)	TARGET PORT GROUP						
3								(LSB)
4	Reserved							
5	STATUS CODE							
6	Vendor specific							
7	TARGET PORT COUNT							
	Target port descriptor list							
8	Target port descriptor (see table 311) [first]							
•••								
11								
	⋮							
n-3	Target port descriptor (see table 311) [last]							
•••								
n								

A preferred target port (PREF) bit set to one indicates that the primary target port group is a preferred primary target port group for accessing the addressed logical unit (see 5.15.2.6). A PREF bit set to zero indicates the primary target port group is not a preferred primary target port group.

The RTPG_FMT field indicates the returned parameter data format and shall be set as shown in table 308 for the target port group descriptor format.

The ASYMMETRIC ACCESS STATE field (see table 309) contains the target port group's target port asymmetric access state (see 5.15.2.7).

Table 309 — ASYMMETRIC ACCESS STATE field

Code	State	Type (see 5.15.2.1)	Reference
0h	Active/optimized	Primary	5.15.2.4.2
1h	Active/non-optimized	Primary	5.15.2.4.3
2h	Standby	Primary	5.15.2.4.4
3h	Unavailable	Primary	5.15.2.4.5
4h	Logical block dependent	Primary	5.15.2.4.7
5h to Dh	Reserved		
Eh	Offline	Secondary	5.15.2.4.6
Fh	Transitioning between states	Primary	5.15.2.5

If any of the T_SUP bit, O_SUP bit, LBD_SUP bit, U_SUP bit, S_SUP bit, AN_SUP bit, or AO_SUP bit are set to one, then the T_SUP bit, O_SUP bit, LBD_SUP bit, U_SUP bit, S_SUP bit, AN_SUP bit, and AO_SUP bit are as defined in this standard. If the T_SUP bit, O_SUP bit, U_SUP bit, S_SUP bit, AN_SUP bit, and AO_SUP bit are all set to zero, then which target port asymmetric access states are supported is vendor specific.

A transitioning supported (T_SUP) bit set to one indicates that the device server supports returning the ASYMMETRIC ACCESS STATE field set to Fh (i.e., transitioning between states). A T_SUP bit set to zero indicates that the device server does not return an ASYMMETRIC ACCESS STATE field set to Fh.

An offline supported (O_SUP) bit set to one indicates that the offline secondary target port asymmetric access state is supported. A O_SUP bit set to zero indicates that the offline secondary target port asymmetric access state is not supported.

A logical block dependent (LBD_SUP) bit set to one indicates that the logical block dependent primary target port asymmetric access state is supported. An LBD_SUP bit set to zero indicates that the logical block dependent primary target port asymmetric access state is not supported.

An unavailable supported (U_SUP) bit set to one indicates that the unavailable primary target port asymmetric access state is supported. A U_SUP bit set to zero indicates that the unavailable primary target port asymmetric access state is not supported.

A standby supported (S_SUP) bit set to one indicates that the standby primary target port asymmetric access state is supported. An S_SUP bit set to zero indicates that the standby primary target port asymmetric access state is not supported.

An active/non-optimized supported (AN_SUP) bit set to one indicates that the active/non-optimized primary target port asymmetric access state is supported. An AN_SUP bit set to zero indicates that the active/non-optimized primary target port asymmetric access state is not supported.

An active/optimized supported (AO_SUP) bit set to one indicates that the active/optimized primary target port asymmetric access state is supported. An AO_SUP bit set to zero indicates that the active/optimized primary target port asymmetric access state is not supported.

The TARGET PORT GROUP field contains an identification of the target port group (see 5.15) described by this target port group descriptor. Target port group information is also returned in the Device Identification VPD page (see 7.8.6).

The STATUS CODE field (see table 310) indicates why a target port group may be in a specific target port asymmetric access state.

Table 310 — STATUS CODE field

Code	Description
00h	No status available.
01h	The target port asymmetric access state altered by SET TARGET PORT GROUPS command.
02h	The target port asymmetric access state altered by implicit asymmetrical logical unit access behavior.
03h to FFh	Reserved

The TARGET PORT COUNT field indicates the number of target ports that are in that target port group and the number of target port descriptors in the target port group descriptor. Every target port group shall contain at least one target port. The target port group descriptor shall include one target port descriptor for each target port in the target port group.

The format of each target port descriptor is shown in table 311.

Table 311 — Target port descriptor format

Bit / Byte	7	6	5	4	3	2	1	0
0	Reserved							
1								
2	(MSB)	RELATIVE TARGET PORT IDENTIFIER						
3								(LSB)

The RELATIVE TARGET PORT IDENTIFIER field (see 4.3.4) indicates a relative port identifier of a target port in the target port group.

6.38 REPORT TIMESTAMP command

The REPORT TIMESTAMP command (see table 312) requests the device server to return the current value of a device clock (see 5.2). This command uses the MAINTENANCE IN CDB format (see 4.2.2.3.3).

Table 312 — REPORT TIMESTAMP command

Bit / Byte	7	6	5	4	3	2	1	0
0	OPERATION CODE (A3h)							
1	Reserved			SERVICE ACTION (0Fh)				
2	Reserved							
•••								
5								
6	(MSB)			ALLOCATION LENGTH				
•••								
9								(LSB)
10	Reserved							
11	CONTROL							

The OPERATION CODE field is defined in 4.2.5.1 and shall be set as shown in table 312 for the REPORT TIMESTAMP command.

The SERVICE ACTION field is defined in 4.2.5.2 and shall be set as shown in table 312 for the REPORT TIMESTAMP command.

The ALLOCATION LENGTH field is defined in 4.2.5.6.

The CONTROL byte is defined in SAM-5.

The format for the parameter data for the REPORT TIMESTAMP command is shown in table 313.

Table 313 — REPORT TIMESTAMP parameter data format

Bit / Byte	7	6	5	4	3	2	1	0
0	(MSB)			TIMESTAMP PARAMETER DATA LENGTH (000Ah)				
1								(LSB)
2	Reserved					TIMESTAMP ORIGIN		
3	Reserved							
4	(MSB)			TIMESTAMP				
•••								
9								(LSB)
10	Reserved							
11	Reserved							

The TIMESTAMP PARAMETER DATA LENGTH field indicates the number of bytes of parameter data that follow. The contents of the TIMESTAMP PARAMETER DATA LENGTH field are not altered based on the allocation length (see 4.2.5.6).

The TIMESTAMP ORIGIN field indicates the most recent event that initialized the returned device clock using the values shown in table 57 (see 5.2).

The TIMESTAMP field contains the current value of a device clock (see 5.2).

6.39 REQUEST SENSE command

The REQUEST SENSE command (see table 314) requests the device server to return parameter data that contains sense data (see 4.5).

Table 314 — REQUEST SENSE command

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

The OPERATION CODE field is defined in 4.2.5.1 and shall be set as shown in table 314 for the REQUEST SENSE command.

The descriptor format (DESC) bit (see table 315) specifies which sense data format the device server shall return in the parameter data.

Table 315 — DESC bit

Code	Descriptor format sense data supported?	Description
0b	yes or no	The device server shall return fixed format sense data (see 4.5.3) in the parameter data.
1b	yes	The device server shall return descriptor format sense data (see 4.5.2) in the parameter data.
	no	The device server shall return no parameter data and shall terminate the REQUEST SENSE command with CHECK CONDITION status with the sense key set to ILLEGAL REQUEST and the additional sense code set to INVALID FIELD IN CDB.
Note: Device servers that are compliant with SPC-3 are capable of ignoring a DESC bit that is set to one.		

The ALLOCATION LENGTH field is defined in 4.2.5.6. Application clients should request 252 bytes of sense data to ensure they retrieve all the sense data. If fewer than 252 bytes are requested, sense data may be lost since the REQUEST SENSE command with any allocation length clears the sense data.

The CONTROL byte is defined in SAM-5.

Sense data shall be available and cleared under the conditions defined in SAM-5.

Upon completion of the REQUEST SENSE command, the logical unit shall be in the same power condition (see 5.11) that was active before the REQUEST SENSE command was received. A REQUEST SENSE command shall not reset any power condition timers.

The device server shall return CHECK CONDITION status for a REQUEST SENSE command only to report exception conditions specific to the REQUEST SENSE command itself. Examples of conditions that cause a REQUEST SENSE command to return CHECK CONDITION status are:

a) an invalid field value is detected in the CDB;
b) the device server does not support the REQUEST SENSE command (see 4.2.1);
c) the initiator port that sent the REQUEST SENSE command is pending enrollment for access controls (see 8.3.1.7) and no other sense data is available to return;
d) an unrecovered error is detected by a SCSI target port; or
e) a malfunction prevents return of the sense data.

If a REQUEST SENSE command is terminated with CHECK CONDITION status, then:

a) any parameter data that is transferred is invalid (i.e., the parameter data does not contain sense data); and
b) if the REQUEST SENSE command was received on an I_T nexus with a pending unit attention condition (i.e., before the device server reports CHECK CONDITION status), then the device server shall not clear the pending unit attention condition (see SAM-5).

The REQUEST SENSE command shall be capable of being processed to completion with GOOD status in any power condition (see 5.11). However, some implementations may require a level of power consumption that is higher than associated with the current power condition in any other circumstance. Details of how much power processing a REQUEST SENSE command requires is outside the scope of this standard.

Except as described elsewhere in this subclause, the device server shall process a REQUEST SENSE command as follows:

1) return applicable sense data in the parameter data as follows:
   1) if the logical unit reports a peripheral qualifier of 011b or 001b in its standard INQUIRY data (see 6.6.2), then return parameter data containing sense data with the sense key set to ILLEGAL REQUEST and the additional sense code shall be set to LOGICAL UNIT NOT SUPPORTED;
   2) if the logical unit reports a peripheral qualifier of 000b in its standard INQUIRY data because it has a peripheral device connected but is not ready for access, then return parameter data containing sense data appropriate to the condition that is making the logical unit not operational;
   3) if the REQUEST SENSE command was received on an I_T nexus with a pending unit attention condition, then return parameter data containing sense data for the unit attention and clear the unit attention condition as described in SAM-5;
   4) if a recovered error occurs during the processing of the REQUEST SENSE command, then return parameter data containing sense data with the sense key set to RECOVERED ERROR;
   5) if deferred error sense data (see 4.5.7) is available, then return parameter data containing sense data for the deferred error;
   6) return pollable sense data (see 5.10.2), if any; or
   7) return parameter data containing sense data with the sense key set to NO SENSE and the additional sense code set to NO ADDITIONAL SENSE INFORMATION;

   and
2) complete the REQUEST SENSE command with GOOD status.

Device servers shall return at least 18 bytes of parameter data in response to a REQUEST SENSE command if the allocation length is 18 or greater and the DESC bit is set to zero. 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.

6.40 SECURITY PROTOCOL IN command

The SECURITY PROTOCOL IN command (see table 316) requests the device server to return security protocol information (see 7.7.1) or the results of one or more SECURITY PROTOCOL OUT commands (see 6.41).

Table 316 — SECURITY PROTOCOL IN command

Bit / Byte	7	6	5	4	3	2	1	0
0	OPERATION CODE (A2h)							
1	SECURITY PROTOCOL							
2	SECURITY PROTOCOL SPECIFIC							
3								
4	INC_512	Reserved						
5	Reserved							
6	(MSB)	ALLOCATION LENGTH						
•••								
9								(LSB)
10	Reserved							
11	CONTROL							

The OPERATION CODE field is defined in 4.2.5.1 and shall be set as shown in table 316 for the SECURITY PROTOCOL IN command.

The SECURITY PROTOCOL field (see table 317) specifies which security protocol is being used.

Table 317 — SECURITY PROTOCOL field in SECURITY PROTOCOL IN command

Code	Description	Reference
00h	Security protocol information	7.7.1
01h to 06h	Defined by TCG	3.1.179
07h	CbCS	7.7.4
08h to 1Fh	Reserved	
20h	Tape Data Encryption	SSC-4
21h	Data Encryption Configuration	ADC-3
22h to 3Fh	Reserved	
40h	SA Creation Capabilities	7.7.2
41h	IKEv2-SCSI	7.7.3
42h to EAh	Reserved	
EBh	Defined by SCSA	3.1.148
ECh	JEDEC Universal Flash Storage	UFS
EDh	SDcard TrustedFlash Security Systems Specification 1.1.3	3.1.149
EEh	Authentication in Host Attachments of Transient Storage Devices	IEEE 1667
EFh	ATA Device Server Password	SAT-3
F0h to FFh	Vendor Specific	

The contents of the SECURITY PROTOCOL SPECIFIC field are defined by the protocol specified by the SECURITY PROTOCOL field (see table 317).

A 512 increment (INC_512) bit set to one specifies that the ALLOCATION LENGTH field (see 4.2.5.6) expresses the maximum number of bytes available to receive data in increments of 512 bytes (e.g., a value of one means 512 bytes, two means 1 024 bytes, etc.). Pad bytes may or may not be appended to meet this length. Pad bytes shall have a value of 00h. An INC_512 bit set to zero specifies that the ALLOCATION LENGTH field expresses the maximum number of bytes available to receive data in increments of one byte.

Indications of data overrun or underrun and the mechanism, if any, for processing retries are defined by the protocol specified by the SECURITY PROTOCOL field (see table 317).

The CONTROL byte is defined in SAM-5.

Any association between a previous SECURITY PROTOCOL OUT command and the data transferred by a SECURITY PROTOCOL IN command depends on the protocol specified by the SECURITY PROTOCOL field (see table 317). If the device server has no data to transfer (e.g., the results for any previous SECURITY PROTOCOL OUT commands are not yet available), then the device server may transfer data indicating it has no other data to transfer.

The format of the data transferred depends on the protocol specified by the SECURITY PROTOCOL field (see table 317).

The device server shall retain data resulting from a SECURITY PROTOCOL OUT command, if any, until one of the following events is processed:

a) transfer of the data via a SECURITY PROTOCOL IN command from the same I_T_L nexus as defined by the protocol specified by the SECURITY PROTOCOL field (see table 317);
b) logical unit reset (see SAM-5); or
c) I_T nexus loss (see SAM-5) associated with the I_T nexus that sent the SECURITY PROTOCOL OUT command.

6.41 SECURITY PROTOCOL OUT command

The SECURITY PROTOCOL OUT command (see table 318) requests the device server to process the specified parameter list using the specified security protocol. Depending on the protocol specified by the SECURITY PROTOCOL field, the application client may use the SECURITY PROTOCOL IN command (see 6.40) to retrieve data that results from the processing of one or more SECURITY PROTOCOL OUT commands.

Table 318 — SECURITY PROTOCOL OUT command

Bit / Byte	7	6	5	4	3	2	1	0
0	OPERATION CODE (B5h)							
1	SECURITY PROTOCOL							
2	SECURITY PROTOCOL SPECIFIC							
3								
4	INC_512	Reserved						
5	Reserved							
6	(MSB)							
•••	TRANSFER LENGTH							
9								(LSB)
10	Reserved							
11	CONTROL							

The OPERATION CODE field is defined in 4.2.5.1 and shall be set as shown in table 318 for the SECURITY PROTOCOL OUT command.

The SECURITY PROTOCOL field (see table 319) specifies which security protocol is being used.

Table 319 — SECURITY PROTOCOL field in SECURITY PROTOCOL OUT command

Code	Description	Reference
00h	Reserved	
01h to 06h	Defined by TCG	3.1.179
07h	CbCS	7.7.4
08h to 1Fh	Reserved	
20h	Tape Data Encryption	SSC-4
21h	Data Encryption Configuration	ADC-3
22h to 40h	Reserved	
41h	IKEv2-SCSI	7.7.3
42h to EAh	Reserved	
EBh	Defined by SCSA	3.1.148
ECh	JEDEC Universal Flash Storage	UFS
EDh	SDcard TrustedFlash Security Systems Specification 1.1.3	3.1.149
EEh	Authentication in Host Attachments of Transient Storage Devices	IEEE 1667
EFh	ATA Device Server Password	SAT-3
F0h to FFh	Vendor Specific	

The contents of the SECURITY PROTOCOL SPECIFIC field are defined by the protocol specified by the SECURITY PROTOCOL field (see table 319).

A 512 increment (INC_512) bit set to one specifies that the TRANSFER LENGTH field (see 4.2.5.4) expresses the number of bytes to be transferred in increments of 512 bytes (e.g., a value of one means 512 bytes, two means 1 024 bytes, etc.). Pad bytes shall be appended as needed to meet this requirement. Pad bytes shall have a value of 00h. A INC_512 bit set to zero specifies that the TRANSFER LENGTH field indicates the number of bytes to be transferred.

The CONTROL byte is defined in SAM-5.

Any association between a SECURITY PROTOCOL OUT command and a subsequent SECURITY PROTOCOL IN command is defined by the protocol specified by the SECURITY PROTOCOL field (see table 319). Each protocol shall define whether:

a) the device server shall complete the command with GOOD status as soon as it determines the data has been correctly received. An indication that the data has been processed is obtained by sending a SECURITY PROTOCOL IN command and receiving the results in the associated data transfer; or
b) the device server shall complete the command with GOOD status only after the data has been successfully processed and an associated SECURITY PROTOCOL IN command is not required.

The format of the data transferred depends on the protocol specified by the SECURITY PROTOCOL field (see table 319).

6.42 SEND DIAGNOSTIC command

The SEND DIAGNOSTIC command (see table 320) requests the device server to perform diagnostic operations on the SCSI target device, on the logical unit, or on both. Logical units that support this command shall support, at a minimum, the default self-test feature (i.e., the SELFTEST bit equal to one and a parameter list length of zero).

Table 320 — 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							

The OPERATION CODE field is defined in 4.2.5.1 and shall be set as shown in table 320 for the SEND DIAGNOSTIC command.

The SELF-TEST CODE field is defined in table 321 and further described in table 322.

Table 321 — SELF-TEST CODE field

Code	Name	Description
000b		This value is used if an operation not described in this table is being requested (see table 322).
001b	Background short self-test	The device server shall start its short self-test (see 5.14.3) in the background mode (see 5.14.4.3).
010b	Background extended self-test	The device server shall start its extended self-test (see 5.14.3) in the background mode (see 5.14.4.3).
011b	Reserved	
100b	Abort background self-test	If the SCSI target device is performing a self-test in the background mode (see 5.14.4.3), then the device server shall abort the self-test. If the target device is performing a self-test in the foreground mode, then the device server shall process the command as described in 5.14.4.2. [a] If the target device is not performing a self-test, then the device server shall terminate the command with CHECK CONDITION status, with the sense key set to ILLEGAL REQUEST, and the additional sense code set to INVALID FIELD IN CDB.
101b	Foreground short self-test	The device server shall start its short self-test (see 5.14.3) in the foreground mode (see 5.14.4.2).
110b	Foreground extended self-test	The device server shall start its extended self-test (see 5.14.3) in the foreground mode (see 5.14.4.2).
111b	Reserved	

[a] Device servers compliant with SPC-3 may terminate the command with CHECK CONDITION status with the sense key set to ILLEGAL REQUEST and the additional sense code set to INVALID FIELD IN CDB.

The page format (PF) bit (see table 322) specifies the format of any parameter list sent by the application client for a SEND DIAGNOSTIC command and may specify the format of the parameter data, if any, returned by the device server in response to a subsequent RECEIVE DIAGNOSTIC RESULTS command (see 6.27).

NOTE 38 - Logical units compliant with SPC-2 may transfer more than one diagnostic page in the SEND DIAGNOSTIC command's parameter list and by doing so may request that more than one diagnostic page be sent in the RECEIVE DIAGNOSTIC RESULTS command's parameter data.

The self-test (SELFTEST) bit (see table 322) specifies whether the device server shall perform the default self-test (see 5.14.2).

A SCSI target device offline (DEVOFFL) bit set to one specifies that the device server may perform a default self-test (see 5.14.2) that affects any logical unit in the SCSI target device (e.g., by alteration of reservations, log parameters, or sense data). A DEVOFFL bit set to zero specifies that, after the device server has completed a default self-test specified in the SEND DIAGNOSTIC command, no logical unit shall exhibit any effects resulting from the device server's processing the SEND DIAGNOSTIC command that are detectable by any application client. If the SELFTEST bit is set to zero, the device server shall ignore the DEVOFFL bit.

A unit offline (UNITOFFL) bit set to one specifies that the device server may perform a default self-test (see 5.14.2) that affects 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). A UNITOFFL bit set to zero specifies that, after the device server has completed a default self-test specified in the SEND DIAGNOSTIC command, the user accessible medium shall exhibit no effects resulting from the device server's processing the SEND DIAGNOSTIC command that are detectable by any application client. If the SELFTEST bit is set to zero, the device server shall ignore the UNITOFFL bit.

The PARAMETER LIST LENGTH field (see table 322) specifies the length in bytes of the parameter list that shall be transferred from the application client Data-Out Buffer to the device server. A parameter list length of zero specifies that no data shall be transferred. This condition shall not be considered an error.

The CONTROL byte is defined in SAM-5.

Table 322 defines:

a) the combinations of values for the SELF-TEST CODE field, the PF bit, if supported, the SELFTEST bit, and the PARAMETER LIST LENGTH field; and
b) under which conditions the DEVOFFL bit and the UNITOFFL bit, if supported, may be used and under which conditions these bits are ignored.

Table 322 — The meanings of the SELF-TEST CODE field, the PF bit, the SELFTEST bit, and the PARAMETER LIST LENGTH field (part 1 of 4)

SELF-TEST CODE field	PF bit	SELF TEST bit	PARAMETER LIST LENGTH field	Description [a, b]
000b	0	0	0000h	The device server shall complete the command with GOOD status.
000b	0	0	>0000h	The device server shall transfer the amount of vendor specific parameter list specified by the PARAMETER LIST LENGTH field. If the parameter list is valid and requests that the device server perform a diagnostic operation [c], then: a) if the operation succeeds, the device server shall complete the command with GOOD status; b) if the operation fails, then the device server shall terminate the command with CHECK CONDITION status with the sense key set to HARDWARE ERROR and the additional sense code set to the appropriate value to describe the failure; and c) if parameter data is required to be returned by the device server for the operation, then a subsequent RECEIVE DIAGNOSTIC RESULTS command (see 6.27) determines the device server's response. If the parameter list is valid and does not request the device server to perform a diagnostic operation, then the device server processes the vendor specific data. A subsequent RECEIVE DIAGNOSTIC RESULTS command may be required for an application client to determine the device server's response. If the parameter list is not valid, the device server shall terminate the command with CHECK CONDITION status with the sense key set to ILLEGAL REQUEST and the additional sense code set to INVALID FIELD IN PARAMETER LIST.

[a] Each description assumes that all other bits and fields in the CDB that are not defined for a particular case are valid. If, in any case, some other bit or field in the CDB is not valid, the device server shall terminate the command with CHECK CONDITION status with the sense key set to ILLEGAL REQUEST and the additional sense code set to INVALID FIELD IN CDB.

[b] The device server shall ignore the DEVOFFL bit and the UNITOFFL bit in all cases where their use is not included in the description.

[c] Before beginning any self-test, the device server shall:
a) stop all running power condition timers;
b) not stop any process that results in a background function occuring (e.g., not stop any timers or counters associated with background functions); and
c) after completing this command, the device server shall reinitialize and restart all enabled power condition timers.

Table 322 — The meanings of the SELF-TEST CODE field, the PF bit, the SELFTEST bit, and the PARAMETER LIST LENGTH field (part 2 of 4)

SELF-TEST CODE field	PF bit	SELF TEST bit	PARAMETER LIST LENGTH field	Description [a, b]
000b	0	1	0000h	The device server shall perform its default self-test, and if supported, use the values in the DEVOFFL bit and the UNITOFFL bit for processing the self-test. If the self-test succeeds, the device server shall complete the command with GOOD status. If the self-test fails, the device server shall terminate the command with CHECK CONDITION status with the sense key set to HARDWARE ERROR and the additional sense code set to the appropriate value to describe the failure.
000b	0	1	>0000h	The device server shall terminate the command with CHECK CONDITION status with the sense key set to ILLEGAL REQUEST and the additional sense code set to INVALID FIELD IN CDB.
000b	1	0	0000h	The device server shall: a) complete the command with GOOD status; and b) if requested by a subsequent RECEIVE DIAGNOSTIC RESULTS command, then: A) if the PF bit is supported by the device server, return a single diagnostic page (see 7.2) as specified by the command; or B) if the PF bit is not supported by the device server, return vendor specific parameter data as specified by the command.

[a] Each description assumes that all other bits and fields in the CDB that are not defined for a particular case are valid. If, in any case, some other bit or field in the CDB is not valid, the device server shall terminate the command with CHECK CONDITION status with the sense key set to ILLEGAL REQUEST and the additional sense code set to INVALID FIELD IN CDB.

[b] The device server shall ignore the DEVOFFL bit and the UNITOFFL bit in all cases where their use is not included in the description.

[c] Before beginning any self-test, the device server shall:
a) stop all running power condition timers;
b) not stop any process that results in a background function occuring (e.g., not stop any timers or counters associated with background functions); and
c) after completing this command, the device server shall reinitialize and restart all enabled power condition timers.

Table 322 — The meanings of the SELF-TEST CODE field, the PF bit, the SELFTEST bit, and the PARAMETER LIST LENGTH field (part 3 of 4)

SELF-TEST CODE field	PF bit	SELF TEST bit	PARAMETER LIST LENGTH field	Description [a, b]
000b	1	0	>0000h	The device server shall transfer the amount of parameter list specified by the PARAMETER LIST LENGTH field. If the PF bit is supported by the device server, then: a) the parameter list shall be a single diagnostic page; and b) if the specified parameter list length results in the truncation of the diagnostic page (e.g., the parameter list length is less than the page length indicated by the diagnostic page), then the device server shall terminate the command with CHECK CONDITION status, with the sense key set to ILLEGAL REQUEST, and the additional sense code set to INVALID FIELD IN CDB. If the PF bit is not supported by the device server, the parameter list is vendor specific. If the parameter list is valid and requests that the device server perform a diagnostic operation [c], then: a) if the operation succeeds, the device server shall complete the command with GOOD status; b) if the operation fails, the device server shall terminate the command with CHECK CONDITION status with the sense key set to HARDWARE ERROR and the additional sense code set to the appropriate value to describe the failure; and c) if parameter data is required to be returned by the device server for the operation, then a subsequent RECEIVE DIAGNOSTIC RESULTS command (see 6.27) determines the device server's response. If the parameter list is valid and does not request the device server to perform a diagnostic operation, then the device server processes the data as defined in 7.2. A subsequent RECEIVE DIAGNOSTIC RESULTS command may be required for an application client to determine the device server's response. If the parameter list is not valid, the device server shall terminate the command with CHECK CONDITION status with the sense key set to ILLEGAL REQUEST and the additional sense code set to INVALID FIELD IN PARAMETER LIST.

[a] Each description assumes that all other bits and fields in the CDB that are not defined for a particular case are valid. If, in any case, some other bit or field in the CDB is not valid, the device server shall terminate the command with CHECK CONDITION status with the sense key set to ILLEGAL REQUEST and the additional sense code set to INVALID FIELD IN CDB.

[b] The device server shall ignore the DEVOFFL bit and the UNITOFFL bit in all cases where their use is not included in the description.

[c] Before beginning any self-test, the device server shall:
a) stop all running power condition timers;
b) not stop any process that results in a background function occuring (e.g., not stop any timers or counters associated with background functions); and
c) after completing this command, the device server shall reinitialize and restart all enabled power condition timers.

Table 322 — The meanings of the SELF-TEST CODE field, the PF bit, the SELFTEST bit, and the PARAMETER LIST LENGTH field (part 4 of 4)

SELF-TEST CODE field	PF bit	SELF TEST bit	PARAMETER LIST LENGTH field	Description [a, b]
000b	1	1	n/a	The device server shall terminate the command with CHECK CONDITION status with the sense key set to ILLEGAL REQUEST and the additional sense code set to INVALID FIELD IN CDB.
>000b	0	0	0000h	The device server shall perform the operation defined in table 321. [c]
>000b	0	0	>0000h	The device server shall terminate the command with CHECK CONDITION status with the sense key set to ILLEGAL REQUEST and the additional sense code set to INVALID FIELD IN CDB.
>000b	0	1	n/a	
>000b	1	0	0000h	The device server shall perform the operation defined in table 321. [c] If the device server supports the PF bit, the device server shall return a single diagnostic page as specified by the subsequent RECEIVE DIAGNOSTIC RESULTS command, if any. If the device server does not support the PF bit, the device server shall return vendor specific data for a subsequent RECEIVE DIAGNOSTIC RESULTS command, if any.
>000b	1	0	>0000h	The device server shall terminate the command with CHECK CONDITION status with the sense key set to ILLEGAL REQUEST and the additional sense code set to INVALID FIELD IN CDB.
>000b	1	1	n/a	

[a] Each description assumes that all other bits and fields in the CDB that are not defined for a particular case are valid. If, in any case, some other bit or field in the CDB is not valid, the device server shall terminate the command with CHECK CONDITION status with the sense key set to ILLEGAL REQUEST and the additional sense code set to INVALID FIELD IN CDB.

[b] The device server shall ignore the DEVOFFL bit and the UNITOFFL bit in all cases where their use is not included in the description.

[c] Before beginning any self-test, the device server shall:

a) stop all running power condition timers;

b) not stop any process that results in a background function occuring (e.g., not stop any timers or counters associated with background functions); and

c) after completing this command, the device server shall reinitialize and restart all enabled power condition timers.

6.43 SET IDENTIFYING INFORMATION command

The SET IDENTIFYING INFORMATION command (see table 323) requests the device server to set identifying information (see 5.6) in the logical unit to the value specified in the SET IDENTIFYING INFORMATION parameter list. This command uses the MAINTENANCE OUT CDB format (see 4.2.2.3.4). The SET IDENTIFYING INFORMATION command is an extension to the SET PERIPHERAL DEVICE/COMPONENT DEVICE IDENTIFIER service action of the MAINTENANCE OUT command defined in SCC-2.

Processing a SET IDENTIFYING INFORMATION command may require the enabling of a nonvolatile memory within the logical unit. If the nonvolatile memory is not ready, the device server shall terminate the command with CHECK CONDITION status, and not wait for the nonvolatile memory to become ready. The sense key shall be set to NOT READY and the additional sense code shall be set as described in table 336 (see 6.47). This information should allow the application client to determine the action required to cause the device server to become ready.

On successful completion of a SET IDENTIFYING INFORMATION command that changes identifying information saved by the logical unit, the device server shall establish a unit attention condition (see SAM-5) for the initiator port associated with every I_T nexus except the I_T nexus on which the SET IDENTIFYING INFORMATION command was received, with the additional sense code set to DEVICE IDENTIFIER CHANGED.

Table 323 — SET IDENTIFYING INFORMATION command

Bit / Byte	7	6	5	4	3	2	1	0
0	OPERATION CODE (A4h)							
1	Reserved			SERVICE ACTION (06h)				
2	Reserved							
3								
4	Restricted (see SCC-2)							
5								
6	(MSB)			PARAMETER LIST LENGTH				
•••								
9								(LSB)
10	IDENTIFYING INFORMATION TYPE							Reserved
11	CONTROL							

The OPERATION CODE field is defined in 4.2.5.1 and shall be set as shown in table 323 for the SET IDENTIFYING INFORMATION command.

The SERVICE ACTION field is defined in 4.2.5.2 and shall be set as shown in table 323 for the SET IDENTIFYING INFORMATION command.

The PARAMETER LIST LENGTH field specifies the length in bytes of the identifying information that shall be transferred from the application client to the device server. A parameter list length of zero specifies that no data shall be transferred, and that subsequent REPORT IDENTIFYING INFORMATION commands shall return the IDENTIFYING INFORMATION LENGTH field set to zero for the specified identifying information type.

The IDENTIFYING INFORMATION TYPE field (see table 324) specifies the identifying information type to be set. If the specified identifying information type is not implemented by the device server, then the device server shall

terminate the command with CHECK CONDITION status with the sense key set to ILLEGAL REQUEST, and the additional sense code set to INVALID FIELD IN CDB.

Table 324 — IDENTIFYING INFORMATION TYPE field

Code	Description
0000000b	Peripheral device identifying information (see 5.6). If the PARAMETER LIST LENGTH field is set to a value greater than the maximum length of the peripheral device identifying information supported by the device server (see 5.6), then the device server shall terminate the command with CHECK CONDITION status with the sense key set to ILLEGAL REQUEST and the additional sense code set to INVALID FIELD IN CDB.
0000010b	Peripheral device text identifying information (see 5.6). If the PARAMETER LIST LENGTH field is set to a value greater than the maximum length of the peripheral device text identifying information (see 5.6) supported by the device server, then the device server shall terminate the command with CHECK CONDITION status with the sense key set to ILLEGAL REQUEST and the additional sense code set to INVALID FIELD IN CDB. If the IDENTIFYING INFORMATION field does not contain a null-terminated (see 4.3.2) UTF-8 format string, then the device server shall terminate the command with CHECK CONDITION status with the sense key set to ILLEGAL REQUEST and the additional sense code set to INVALID FIELD IN PARAMETER LIST.
xxxxxx1b	Restricted (see SCC-2)
All others	Reserved

The CONTROL byte is defined in SAM-5.

The SET IDENTIFYING INFORMATION parameter list (see table 325) contains the identifying information to be set by the device server.

Table 325 — SET IDENTIFYING INFORMATION parameter list

Bit / Byte	7	6	5	4	3	2	1	0
0	IDENTIFYING INFORMATION							
•••								
n								

The IDENTIFYING INFORMATION field specifies the identifying information to be set for the specified identifying information type (see 5.6).

Upon successful completion of a SET IDENTIFYING INFORMATION command, the identifying information that is saved by the logical unit shall persist through logical unit resets, hard resets, power loss, I_T nexus losses, media format operations, and media replacement.

6.44 SET PRIORITY command

The SET PRIORITY command (see table 326) requests the device server to set the command priority (see SAM-5) for commands received on the specified I_T nexus.. This command uses the MAINTENANCE OUT CDB format (see 4.2.2.3.4). The command priority set by this command shall remain in effect until one of the following occurs:

a) another SET PRIORITY command is received;
b) hard reset;
c) logical unit reset; or
d) power on.

The command priority set by this command shall not be affected by an I_T nexus loss.

Table 326 — SET PRIORITY command

Bit / Byte	7	6	5	4	3	2	1	0
0	OPERATION CODE (A4h)							
1	Reserved			SERVICE ACTION (0Eh)				
2	I_T_L NEXUS TO SET		Reserved					
3	Reserved							
•••								
5								
6	(MSB)							
•••	PARAMETER LIST LENGTH							
9								(LSB)
10	Reserved							
11	CONTROL							

The OPERATION CODE field is defined in 4.2.5.1 and shall be set as shown in table 326 for the SET PRIORITY command.

The SERVICE ACTION field is defined in 4.2.5.2 and shall be set as shown in table 326 for the SET PRIORITY command.

The I_T_L NEXUS TO SET field (see table 327) specifies the I_T_L nexus and the location of the priority value to be assigned to that I_T_L nexus.

Table 327 — I_T_L NEXUS TO SET field

Code	Description
00b	The priority for the I_T_L nexus associated with this command shall be set to the value contained in the PRIORITY TO SET field in the SET PRIORITY parameter list (see table 328). All fields in the SET PRIORITY parameter list except the PRIORITY TO SET field shall be ignored. If the parameter list length is zero, the device server shall terminate the command with CHECK CONDITION status, with the sense key set to ILLEGAL REQUEST, and the additional sense code set to PARAMETER LIST LENGTH ERROR.
01b	The priority for the I_T_L nexus specified by the logical unit that is processing this command, the RELATIVE TARGET PORT IDENTIFIER field, and the TransportID in the SET PRIORITY parameter list (see table 328) shall be set to the value specified by the PRIORITY TO SET field in the SET PRIORITY parameter list. If the parameter list length results in the truncation of the RELATIVE TARGET PORT IDENTIFIER field, the ADDITIONAL DESCRIPTOR LENGTH field, or the TransportID, then the device server shall terminate the command with CHECK CONDITION status, with the sense key set to ILLEGAL REQUEST, and the additional sense code set to PARAMETER LIST LENGTH ERROR. On successful completion of a SET PRIORITY command the device server shall establish a unit attention condition (see SAM-5) for the initiator port associated with the I_T nexus specified by the TransportID and the RELATIVE TARGET PORT IDENTIFIER field, with the additional sense code set to PRIORITY CHANGED.
10b	The priority value specified in the INITIAL COMMAND PRIORITY field of the Control Extension mode page (see 7.5.9) shall be used for all I_T_L nexuses associated with the logical unit that is processing this command regardless of any prior priority. The contents of the SET PRIORITY parameter list shall be ignored. On successful completion of a SET PRIORITY command the device server shall establish a unit attention condition (see SAM-5) for the initiator port associated with every other I_T_L nexus, with the additional sense code set to PRIORITY CHANGED.
11b	Reserved

The PARAMETER LIST LENGTH field specifies the length in bytes of the SET PRIORITY parameter list (see table 328) that shall be contained in the Data-Out Buffer. A parameter list length of zero specifies that the Data-Out Buffer shall be empty. This condition shall not be considered an error.

The CONTROL byte is defined in SAM-5.

The format of the parameter data for the SET PRIORITY command is shown in table 328.

Table 328 — SET PRIORITY parameter list format

Bit / Byte	7	6	5	4	3	2	1	0
0				Reserved	PRIORITY TO SET			
1				Reserved				
2	(MSB)			RELATIVE TARGET PORT IDENTIFIER				
3								(LSB)
4				Reserved				
5				Reserved				
6	(MSB)			ADDITIONAL LENGTH (n-7)				
7								(LSB)
8				TransportID				
•••								
n								

The PRIORITY TO SET field specifies the priority to be assigned to the I_T_L nexus specified by the I_T_L NEXUS TO SET field in the CDB. The value in the PRIORITY TO SET field shall be returned in subsequent REPORT PRIORITY commands (see 6.34) until one of the conditions described in this subclause occurs. A priority to set value of zero specifies the I_T_L nexus specified by the I_T_L NEXUS TO SET field shall be set to the value specified in the INITIAL COMMAND PRIORITY field of the Control Extension mode page (see 7.5.9). The contents of the I_T_L NEXUS TO SET field may specify that the PRIORITY TO SET field shall be ignored.

The RELATIVE TARGET PORT IDENTIFIER field (see 4.3.4) specifies the relative port identifier of the target port that is part of the I_T_L nexus for which the priority is to be set. The contents of the I_T_L NEXUS TO SET field may specify that the RELATIVE TARGET PORT IDENTIFIER field shall be ignored.

The ADDITIONAL LENGTH field specifies the number of bytes that follow in the SET PRIORITY parameter list (i.e., the size of the TransportID).

The TransportID specifies a TransportID (see 7.6.4) identifying the initiator port that is part of the I_T_L nexus for which the priority is to be set. The contents of the I_T_L NEXUS TO SET field may specify that the TRANSPORTID field shall be ignored.

6.45 SET TARGET PORT GROUPS command

The SET TARGET PORT GROUPS command (see table 329) requests the device server to set the primary target port asymmetric access state of all of the target ports in the specified primary target port groups and/or the secondary target port asymmetric access state of the specified target ports. This command uses the MAINTENANCE OUT CDB format (see 4.2.2.3.4). See 5.15 for details regarding the transition between target port asymmetric access states. This command is mandatory for all logical units that report in the standard

INQUIRY data (see 6.6.2) that they support explicit asymmetric logical units access (i.e., the TPGS field is set to either 10b or 11b).

Table 329 — SET TARGET PORT GROUPS command

Bit / Byte	7	6	5	4	3	2	1	0
0	OPERATION CODE (A4h)							
1	Reserved			SERVICE ACTION (0Ah)				
2	Reserved							
•••								
5								
6	(MSB)			PARAMETER LIST LENGTH				
•••								
9								(LSB)
10	Reserved							
11	CONTROL							

The OPERATION CODE field is defined in 4.2.5.1 and shall be set as shown in table 329 for the SET TARGET PORT GROUPS command.

The SERVICE ACTION field is defined in 4.2.5.2 and shall be set as shown in table 329 for the SET TARGET PORT GROUPS command.

The PARAMETER LIST LENGTH field specifies the length in bytes of the target port group management parameters that shall be transferred from the application client to the device server. A parameter list length of zero specifies that no data shall be transferred, and that no change shall be made in the target port asymmetric access state of any target port groups or target ports. If the specfied parameter list length is not supported, the device server shall terminate the command with CHECK CONDITION status, with the sense key set to ILLEGAL REQUEST, and the additional sense code set to INVALID FIELD IN CDB.

The CONTROL byte is defined in SAM-5.

The allowable values to which target port asymmetric access states may be set is vendor specific and should be reported in the REPORT TARGET PORT GROUP parameter data (see 6.37).

Primary target port groups that are not specified in a parameter list may change primary target port asymmetric access states as a result of the SET TARGET PORT GROUPS command. This shall not be considered an implicit target port asymmetric access state change.

If a SET TARGET PORT GROUPS command attempts to establish an invalid combination of target port asymmetric access states or attempts to establish an unsupported target port asymmetric access state, then the device server shall terminate the command with CHECK CONDITION status, with the sense key set to ILLEGAL REQUEST, and the additional sense code set to INVALID FIELD IN PARAMETER LIST.

How a SET TARGET PORT GROUPS command is completed with success depends upon which of the following conditions apply:

a) if the transition is treated as a single indivisible event (see 5.15.2.5), the SET TARGET PORT GROUPS command shall not complete until the transition to the requested state has completed; or
b) if the transition is not treated as a single indivisible event (i.e., the device server supports other commands (see 5.15.2.5) when those commands are routed through a target port that is transitioning between target port asymmetric access states), then the SET TARGET PORT GROUPS command may complete before the transition into the requested state has completed.

How a SET TARGET PORT GROUPS command is terminated with an error depends upon which of the following conditions apply:

a) if the processing of a SET TARGET PORT GROUPS command requires the enabling of a nonvolatile memory and the nonvolatile memory is not ready, then the device server shall terminate the command with CHECK CONDITION status, rather than wait for the logical unit to become ready. The sense key shall be set to NOT READY and the additional sense code shall be set as described in table 336 (see 6.47); or
b) if a failure occurred before the transition was completed, the device server shall terminate the command with CHECK CONDITION status, with the sense key set to HARDWARE ERROR, and the additional sense code set to SET TARGET PORT GROUPS COMMAND FAILED.

If two SET TARGET PORT GROUPS commands are processed concurrently, the target port asymmetric access state change behavior is vendor specific. A SCSI target device should not process multiple SET TARGET PORT GROUPS concurrently.

The SET TARGET PORT GROUPS parameter data format is shown in table 330.

Table 330 — SET TARGET PORT GROUPS parameter list format

Bit / Byte	7	6	5	4	3	2	1	0
0	Reserved							
•••								
3								
	Set target port group descriptor list							
4	Set target port group descriptor (see table 331) [first]							
•••								
7								
	⋮							
n-3	Set target port group descriptor (see table 331) [last]							
•••								
n								

The format of the set target port group descriptor is defined in table 331.

Table 331 — Set target port group descriptor parameter list

Bit / Byte	7	6	5	4	3	2	1	0
0	Reserved				ASYMMETRIC ACCESS STATE			
1	Reserved							
2	(MSB) TARGET PORT GROUP OR TARGET PORT							
3								(LSB)

If the ASYMMETRIC ACCESS STATE field (see table 332) specifies a primary target port asymmetric access state, then all the target ports in the specified target port group shall transition to the specified state (see 5.15.2.5). If

the ASYMMETRIC ACCESS STATE field specifies a secondary target port asymmetric access state, then the specified target port shall transition to the specified state.

Table 332 — ASYMMETRIC ACCESS STATE field

Value	State (see 5.15.2.4)	Type (see 5.15.2.1)
0h	Active/optimized	Primary
1h	Active/non-optimized	Primary
2h	Standby	Primary
3h	Unavailable	Primary
4h	Illegal Request [a]	
5h to Dh	Reserved	
Eh	Offline	Secondary
Fh	Illegal Request [a]	

[a] If the ASYMMETRIC ACCESS STATE field in any target port group descriptor contains 04h or Fh, then the device server shall terminate the command with CHECK CONDITION status, with the sense key set to ILLEGAL REQUEST, and the additional sense code set to INVALID FIELD IN PARAMETER LIST.

If the ASYMMETRIC ACCESS STATE field (see table 332) specifies a primary target port asymmetric access state, then the TARGET PORT GROUP OR TARGET PORT field specifies a primary target port group for which the primary target port asymmetric access state shall be changed. If the ASYMMETRIC ACCESS STATE field specifies a secondary target port asymmetric access state, then the TARGET PORT GROUP OR TARGET PORT field specifies the relative target port identifier of the target port for which the secondary target port asymmetric access state shall be changed.

6.46 SET TIMESTAMP command

The SET TIMESTAMP command (see table 333) requests the device server to initialize a device clock (see 5.2) if the SCSIP bit is set to one in the Control Extension mode page (see 7.5.9). If the SCSIP bit is set to zero, the device server shall terminate the SET TIMESTAMP command with CHECK CONDITION status, with the sense key set to ILLEGAL REQUEST, and the additional sense code set to INVALID FIELD IN CDB. This command uses the MAINTENANCE OUT CDB format (see 4.2.2.3.4).

Table 333 — SET TIMESTAMP command

Bit / Byte	7	6	5	4	3	2	1	0
0	OPERATION CODE (A4h)							
1	Reserved			SERVICE ACTION (0Fh)				
2	Reserved							
•••								
5								
6	(MSB)			PARAMETER LIST LENGTH				
•••								
9								(LSB)
10	Reserved							
11	CONTROL							

The OPERATION CODE field is defined in 4.2.5.1 and shall be set as shown in table 333 for the SET TIMESTAMP command.

The SERVICE ACTION field is defined in 4.2.5.2 and shall be set as shown in table 333 for the SET TIMESTAMP command.

The PARAMETER LIST LENGTH field specifies the length in bytes of the SET TIMESTAMP parameters that shall be transferred from the application client to the device server. A parameter list length of zero specifies that no data shall be transferred, and that no change shall be made to a device clock.

The CONTROL byte is defined in SAM-5.

The format for the parameter list for the SET TIMESTAMP command is shown in table 334.

Table 334 — SET TIMESTAMP parameter list format

Bit / Byte	7	6	5	4	3	2	1	0
0	Reserved							
•••								
3								
4	(MSB)			TIMESTAMP				
•••								
9								(LSB)
10	Reserved							
11								

The TIMESTAMP field specifies the value to which a device clock shall be initialized (see 5.2). The timestamp should be the number of milliseconds that have elapsed since midnight, 1 January 1970 UT. If the most significant byte in the TIMESTAMP field is greater than F0h, the device server shall terminate the command with CHECK CONDITION status, with the sense key set to ILLEGAL REQUEST, and the additional sense code set to INVALID FIELD IN PARAMETER LIST.

On successful completion of a SET TIMESTAMP command the device server shall establish a unit attention condition for the initiator port associated with every I_T nexus except the I_T nexus on which the SET TIMESTAMP command was received (see SAM-5), with the additional sense code set to TIMESTAMP CHANGED.

6.47 TEST UNIT READY command

The TEST UNIT READY command (see table 335) requests the device server to indicate whether the logical unit is ready. The device server shall not initiate a self-test as a result of processing this command. If the logical unit is able to accept an appropriate medium-access command without returning CHECK CONDITION status, then the device server shall complete this command with GOOD status. If the logical unit is unable to become operational or is in a state such that an application client action (e.g., a START UNIT command) is required to make the logical unit ready, then the device server shall terminate this command with CHECK CONDITION status, with the sense key set to NOT READY.

Table 335 — TEST UNIT READY command

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

The OPERATION CODE field is defined in 4.2.5.1 and shall be set as shown in table 335 for the TEST UNIT READY command.

The CONTROL byte is defined in SAM-5.

Table 336 defines the suggested CHECK CONDITION status responses to the TEST UNIT READY command. Other conditions (e.g., deferred errors, reservations, or target port asymmetric access state changes) may result in other responses (e.g., GOOD status, CHECK CONDITION status, BUSY status, or RESERVATION CONFLICT status, each with or without other sense key and additional sense code values).

Table 336 — Preferred TEST UNIT READY responses

Status	Sense Key	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
CHECK CONDITION	NOT READY	LOGICAL UNIT NOT READY, SANITIZE IN PROGRESS

6.48 WRITE ATTRIBUTE command

The WRITE ATTRIBUTE command (see table 337) requests the device server to write the specified attributes to medium auxiliary memory. Device servers that implement the WRITE ATTRIBUTE command shall also implement the READ ATTRIBUTE command (see 6.17). Application clients should issue READ ATTRIBUTE commands prior to using this command to discover device server support for medium auxiliary memory.

Table 337 — WRITE ATTRIBUTE command

Bit / Byte	7	6	5	4	3	2	1	0
0	OPERATION CODE (8Dh)							
1	Reserved							WTC
2	Restricted (see SMC-3)							
3								
4								
5	LOGICAL VOLUME NUMBER							
6	Reserved							
7	PARTITION NUMBER							
8	Reserved							
9								
10	(MSB) PARAMETER LIST LENGTH							
•••								
13								(LSB)
14	Reserved							
15	CONTROL							

The OPERATION CODE field is defined in 4.2.5.1 and shall be set as shown in table 337 for the WRITE ATTRIBUTE command.

The write-through cache (WTC) bit set to one specifies the attributes in the parameter list shall be synchronized with the medium auxiliary memory during the processing of the WRITE ATTRIBUTE command and GOOD status shall not be returned until the attributes have been synchronized with the medium auxiliary memory. The WTC bit is set to zero specifies no requirement related to the attributes in the parameter list being synchronized with the medium auxiliary memory during the processing of the WRITE ATTRIBUTE command.

The LOGICAL VOLUME NUMBER field specifies a logical volume (e.g., the medium auxiliary memory storage for one side of a double sided medium) within the medium auxiliary memory. The number of logical volumes of the medium auxiliary memory shall equal that of the attached medium. If the medium only has a single logical volume, then its logical volume number shall be zero.

The PARTITION NUMBER field specifies a partition (see SSC-4) within a logical volume. The number of partitions of the medium auxiliary memory shall equal that of the attached medium. If the medium only has a single partition, then its partition number shall be zero.

If the combination of logical volume number and partition number is not valid, the device server shall terminate the command with CHECK CONDITION status, with the sense key 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 parameter list contained in the Data-Out Buffer. A parameter list length of zero specifies that no parameter list is present; this shall not be considered

an error. If the parameter list length results in the truncation of an attribute, the WRITE ATTRIBUTE command shall be terminated with CHECK CONDITION status, with the sense key set to ILLEGAL REQUEST, and the additional sense code set to PARAMETER LIST LENGTH ERROR.

The CONTROL byte is defined in SAM-5.

The parameter list shall have the format shown in table 338. Attributes should be sent in ascending numerical order. If the attributes are not in order, then no attributes shall be changed and the device server shall terminate the WRITE ATTRIBUTE command with CHECK CONDITION status, with the sense key set to ILLEGAL REQUEST, and the additional sense code set to INVALID FIELD IN PARAMETER LIST.

Table 338 — WRITE ATTRIBUTE parameter list format

Bit / Byte	7	6	5	4	3	2	1	0
0	(MSB)							
•••	PARAMETER DATA LENGTH (n-3)							
3								(LSB)
	Attribute(s)							
4								
•••	Attribute (see 7.4.1) [first]							
	⋮							
•••	Attribute (see 7.4.1) [last]							
n								

The PARAMETER DATA LENGTH field should contain the number of bytes of attribute data and shall be ignored by the device server.

The format of the attributes is described in 7.4.1.

If there is not enough space to write the attributes to the medium auxiliary memory, then no attributes shall be changed and the device server shall terminate the WRITE ATTRIBUTE command with CHECK CONDITION status, with the sense key set to ILLEGAL REQUEST, and the additional sense code set to AUXILIARY MEMORY OUT OF SPACE.

If the medium auxiliary memory is not accessible as a result of there being no medium present, then no attributes shall be changed and the device server shall terminate the WRITE ATTRIBUTE command with CHECK CONDITION status, with the sense key set to NOT READY, and the additional sense code set to MEDIUM NOT PRESENT.

If the medium is present but the medium auxiliary memory is not accessible, then no attributes shall be changed and the device server shall terminate the WRITE ATTRIBUTE command with CHECK CONDITION status, with the sense key set to MEDIUM ERROR, and the additional sense code set to LOGICAL UNIT NOT READY, AUXILIARY MEMORY NOT ACCESSIBLE.

If the medium auxiliary memory is not operational (e.g., bad checksum), the device server shall terminate the WRITE ATTRIBUTE command with CHECK CONDITION status, with the sense key set to MEDIUM ERROR, and the additional sense code set to AUXILIARY MEMORY WRITE ERROR.

If the WRITE ATTRIBUTE command parameter list contains an attribute with an ATTRIBUTE LENGTH field (see 7.4.1) set to zero, then one of the following actions shall occur:

a) if the attribute state is unsupported or read only (see 5.7), then no attributes shall be changed and the device server shall terminate the WRITE ATTRIBUTE command with CHECK CONDITION status, with the sense key set to ILLEGAL REQUEST, and the additional sense code set to INVALID FIELD IN PARAMETER LIST;
b) if the attribute state is read/write, the attribute shall be changed to the nonexistent state. This attribute shall not be returned in response to a READ ATTRIBUTE command and shall not be included in the parameter data returned by a READ ATTRIBUTE command with ATTRIBUTE LIST service action; or
c) if the attribute state is nonexistent, the attribute in the WRITE ATTRIBUTE command parameter list shall be ignored; this shall not be considered an error.

No attributes shall be changed, the WRITE ATTRIBUTE command shall be terminated with CHECK CONDITION status, with the sense key set to ILLEGAL REQUEST, and the additional sense code set to INVALID FIELD IN PARAMETER LIST if the parameter list contains any of the following:

a) an attempt to change an attribute in the read only state (see 5.7);
b) an attribute with incorrect ATTRIBUTE LENGTH field (see 7.4.1) contents; or
c) an attribute with unsupported ATTRIBUTE VALUE field (see 7.4.1) contents.

6.49 WRITE BUFFER command

6.49.1 WRITE BUFFER command introduction

The WRITE BUFFER command (see table 339) is used in conjunction with the READ BUFFER command for:

a) testing logical unit buffer memory;
b) testing the integrity of the service delivery subsystem;
c) downloading microcode (see 5.4); and
d) downloading application client error history (see 5.5).

Table 339 — WRITE BUFFER command

Bit / Byte	7	6	5	4	3	2	1	0
0	OPERATION CODE (3Bh)							
1	MODE SPECIFIC			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 if the data mode or the combined header and data mode is specified.

The OPERATION CODE field is defined in 4.2.5.1 and shall be set as shown in table 339 for the WRITE BUFFER command.

The usage of the MODE SPECIFIC field depends on the value in the MODE field.

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 340.

Table 340 — WRITE BUFFER MODE field

Code	Description	Reference
00h	Obsolete	
01h	Vendor specific	6.49.2
02h	Data	6.49.3
03h	Reserved	
04h	Download microcode and activate	5.4 and 6.49.4
05h	Download microcode, save, and activate	5.4 and 6.49.5
06h	Download microcode with offsets and activate	5.4 and 6.49.6
07h	Download microcode with offsets, save, and activate	5.4 and 6.49.7
08h to 09h	Reserved	
0Ah	Write data to echo buffer	6.49.8
0Bh to 0Ch	Reserved	
0Dh	Download microcode with offsets, select activation events, save, and defer activate	5.4 and 6.49.9
0Eh	Download microcode with offsets, save, and defer activate	5.4 and 6.49.10
0Fh	Activate deferred microcode	5.4 and 6.49.11
10h to 19h	Reserved	
1Ah	Obsolete	
1Bh	Obsolete	
1Ch	Download application client error history	5.5 and 6.49.12
1Dh to 1Fh	Reserved	

The MODE field may be processed as specifying a service action by the REPORT SUPPORTED OPERATION CODES command (see 6.35).

The CONTROL byte is defined in SAM-5.

6.49.2 Vendor specific mode (01h)

In this mode, the MODE SPECIFIC field is reserved.

The meaning of the BUFFER ID field, BUFFER OFFSET field, and PARAMETER LIST LENGTH field are not specified by this standard.

6.49.3 Data mode (02h)

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 manufacturer assigns buffer ID codes to buffers within the logical unit. Buffer ID zero shall be supported. If more than one buffer is supported, then additional buffer ID codes shall be assigned contiguously, beginning with one. If an unsupported buffer ID code is selected, the device server shall terminate the command with CHECK CONDITION status, with the sense key set to ILLEGAL REQUEST, and the additional sense code set to INVALID FIELD IN CDB.

The MODE SPECIFIC field is reserved.

The BUFFER OFFSET field specifies the location in the buffer to which the data is written. The application client should conform to the offset boundary requirements returned in the READ BUFFER descriptor (see 6.18.4). If the device server is unable to process the specified buffer offset, the device server shall terminate the command with CHECK CONDITION status, with the sense key set to ILLEGAL REQUEST, and the additional sense code set to 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 specify a parameter list length plus the buffer offset that does not exceed the capacity of the specified buffer. The capacity of the buffer is indicated by the BUFFER CAPACITY field in the READ BUFFER descriptor (see 6.18.4). If the BUFFER OFFSET field and PARAMETER LIST LENGTH field specify a transfer in excess of the buffer capacity, then the device server shall terminate the command with CHECK CONDITION status, with the sense key set to ILLEGAL REQUEST, and the additional sense code set to INVALID FIELD IN CDB.

6.49.4 Download microcode and activate mode (04h)

In this mode, microcode shall be transferred to the device server and activated (see 5.4).

The MODE SPECIFIC field is reserved.

The BUFFER ID field, BUFFER OFFSET field, and PARAMETER LIST LENGTH field are vendor specific.

6.49.5 Download microcode, save, and activate mode (05h)

In this mode, microcode shall be transferred to the device server, saved to nonvolatile storage, and activated (see 5.4) based on the setting of the ACTIVATE MICROCODE field in the Extended INQUIRY VPD page (see 7.8.7).

The MODE SPECIFIC field is reserved.

The BUFFER ID field, BUFFER OFFSET field, and PARAMETER LIST LENGTH field are vendor specific.

6.49.6 Download microcode with offsets and activate mode (06h)

In this mode, microcode shall be transferred to the device server using one or more WRITE BUFFER commands and activated (see 5.4).

The MODE SPECIFIC field is reserved.

The BUFFER ID field specifies a buffer within the logical unit. The manufacturer 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, then additional buffer ID codes shall be assigned contiguously, beginning with one. If an unsupported buffer ID code is specified, the device server shall terminate the command with CHECK

CONDITION status, with the sense key set to ILLEGAL REQUEST, and the additional sense code set to INVALID FIELD IN CDB.

The BUFFER OFFSET field specifies the location in the buffer to which the microcode is written. The application client shall send only commands that conform to the offset boundary requirements returned in the READ BUFFER descriptor (see 6.18.4). If the device server is unable to process the specified buffer offset, the device server shall terminate the command with CHECK CONDITION status, with the sense key set to ILLEGAL REQUEST, and the additional sense code set to 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 specify a parameter list length plus the buffer offset that does not exceed the capacity of the specified buffer. If the BUFFER OFFSET field and PARAMETER LIST LENGTH field specify a transfer in excess of the buffer capacity, then the device server shall terminate the command with CHECK CONDITION status, with the sense key set to ILLEGAL REQUEST, and the additional sense code set to INVALID FIELD IN CDB.

6.49.7 Download microcode with offsets, save, and activate mode (07h)

In this mode, microcode shall be transferred to the device server using one or more WRITE BUFFER commands, saved to nonvolatile storage, and activated (see 5.4) based on the setting of the ACTIVATE MICROCODE field in the Extended INQUIRY VPD page (see 7.8.7).

The BUFFER ID field, BUFFER OFFSET field, and PARAMETER LIST LENGTH field are defined in the download microcode with offsets mode (see 6.49.6).

6.49.8 Write data to echo buffer mode (0Ah)

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 device server as it would for a write operation. Data shall be aligned on four-byte boundaries.

The BUFFER ID and BUFFER OFFSET fields shall be ignored in this mode.

NOTE 39 - It is recommended that the logical unit assign echo buffers on a per I_T nexus basis to limit the number of exception conditions that may occur if there is more than one I_T nexus present.

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 the data 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 specify a parameter list length that does not exceed the capacity of the echo buffer. The capacity of the echo buffer is indicated by the BUFFER CAPACITY field in the READ BUFFER echo buffer descriptor (see 6.18.6). If the PARAMETER LIST LENGTH field specifies a transfer in excess of the buffer capacity, the device server shall terminate the command with CHECK CONDITION status, with the sense key set to ILLEGAL REQUEST, and the additional sense code set to INVALID FIELD IN CDB.

6.49.9 Download microcode with offsets, select activation, save, and defer activate mode (0Dh)

In this mode, microcode shall be transferred to the device server using one or more WRITE BUFFER commands, saved to nonvolatile storage, and considered deferred (see 5.4). The deferred microcode shall be activated and no longer considered deferred if a WRITE BUFFER command with the activate deferred microcode mode (0Fh) is processed (see 6.49.11).

The MODE SPECIFIC field (see table 341) specifies additional events that shall be used to activate the deferred microcode.

Table 341 — MODE SPECIFIC field

Bit	7	6	5	...
	PO_ACT	HR_ACT	VSE_ACT	...

If the power on activate (PO_ACT) bit is set to one, then deferred microcode shall be activated and no longer considered deferred if a power on occurs. If the PO_ACT bit is set to zero, then deferred microcode shall not be activated if a power on occurs.

If the hard reset activate (HR_ACT) bit is set to one, then deferred microcode shall be activated and no longer considered deferred if a hard reset occurs. If the HR_ACT bit is set to zero, then deferred microcode shall not be activated if a hard reset occurs.

If the vendor specific event activate (VSE_ACT) bit is set to one, then deferred microcode shall be activated and no longer considered deferred if a vendor specific event occurs. If the VSE_ACT bit is set to zero, then deferred microcode shall not be activated if a vendor specific event occurs.

The supported activation events shall be reported in the POA_SUP bit, HRA_SUP bit, and VSA_SUP bit in the Extended INQUIRY VPD page (see 7.8.7). If the MODE SPECIFIC field specifies an activation event that is not supported (e.g., if the PO_ACT bit is set to one and the POA_SUP bit is set to zero), then the device server shall terminate the command CHECK CONDITION status with the sense key set to ILLEGAL REQUEST and the additional sense code set to INVALID FIELD IN CDB.

The BUFFER ID field, BUFFER OFFSET field, and PARAMETER LIST LENGTH field are defined in the download microcode with offsets mode (see 6.49.6).

6.49.10 Download microcode with offsets, save, and defer activate mode (0Eh)

In this mode, microcode shall be transferred to the device server using one or more WRITE BUFFER commands, saved to nonvolatile storage, and considered deferred (see 5.4).

The deferred microcode shall be activated and no longer considered deferred if any one of the following occurs:

a) a power on;
b) a hard reset;
c) a START STOP UNIT command is processed (see SBC-3);
d) a FORMAT UNIT command is processed (see SBC-3); or
e) a WRITE BUFFER command with the activate deferred microcode mode (0Fh) is processed (see 6.49.11).

The MODE SPECIFIC field, BUFFER ID field, BUFFER OFFSET field, and PARAMETER LIST LENGTH field are defined in the download microcode with offsets mode (see 6.49.6).

6.49.11 Activate deferred microcode mode (0Fh)

In this mode, deferred microcode, if any, that has been saved using one of the modes list in this subclause shall be activated and no longer considered deferred (see 5.4). The modes that save deferred microcode are:

a) the download microcode with offsets, select activation events, save, and defer activate mode (0Dh) (see 6.49.9); and

b) the download microcode with offsets, save, and defer activate mode (0Eh) (see 6.49.10).

The MODE SPECIFIC field is reserved.

The the BUFFER ID field, the BUFFER OFFSET field, and PARAMETER LIST LENGTH field shall be ignored in this mode.

If there is no deferred microcode that has been saved using one of the modes list in this subclause, the device server shall terminate the WRITE BUFFER command with CHECK CONDITION status, with the sense key set to ILLEGAL REQUEST, and the additional sense code set to COMMAND SEQUENCE ERROR.

6.49.12 Download application client error history mode (1Ch)

In this mode the device server transfers application client error history from the application client and stores it in the error history (see 5.5). The format of the application client error history parameter list is defined in table 342.

The MODE SPECIFIC field is reserved.

The BUFFER ID field and BUFFER OFFSET field shall be ignored in this mode.

Upon successful completion of a WRITE BUFFER command, the information contained in the application client error history parameter list shall be appended to the application client error history in a format determined by the logical unit.

The PARAMETER LIST LENGTH field specifies the length in bytes of the application client error history parameter list that shall be transferred from the application client to the device server. If the PARAMETER LIST LENGTH field specifies a transfer that exceeds the error history capacity, the device server shall terminate the command with CHECK CONDITION status, with the sense key set to ILLEGAL REQUEST, and the additional sense code set to INVALID FIELD IN CDB.

The device server shall not return an error based on the contents of any of the field values defined in table 342 except:

a) the CLR bit;
b) the ERROR LOCATION LENGTH field; and
c) the APPLICATION CLIENT ERROR HISTORY LENGTH field.

Table 342 — Application client error history parameter list format

Bit / Byte	7	6	5	4	3	2	1	0
0	(MSB)							
•••				T10 VENDOR IDENTIFICATION				
7								(LSB)
8	(MSB)							
9				ERROR TYPE				(LSB)
10				Reserved				CLR
11				Reserved				
12	(MSB)							
•••				TIMESTAMP				
17								(LSB)
18				Reserved				
19								
20			Reserved			CODE SET		
21				ERROR LOCATION FORMAT				
22	(MSB)							
23				ERROR LOCATION LENGTH (m-25)				(LSB)
24	(MSB)							
25				APPLICATION CLIENT ERROR HISTORY LENGTH (n-m)				(LSB)
26	(MSB)							
•••				ERROR LOCATION				
m								(LSB)
m+1								
•••				APPLICATION CLIENT ERROR HISTORY				
n								

The T10 VENDOR IDENTIFICATION field contains eight bytes of left-aligned ASCII data (see 4.3.1) identifying the manufacturer providing the application client error history. The T10 vendor identification shall be one assigned by INCITS. A list of assigned T10 vendor identifications is in Annex G and on the T10 web site (http://www.t10.org).

The ERROR TYPE field (see table 343) specifies the error detected by the application client.

Table 343 — ERROR TYPE field

Code	Description
0000h	No error specified by the application client.
0001h	An unknown error was detected by the application client.
0002h	The application client detected corrupted data.
0003h	The application client detected a permanent error.
0004h	The application client detected a service response of SERVICE DELIVERY OR TARGET FAILURE (see SAM-5).
0005h to 7FFFh	Reserved
8000h to FFFFh	Vendor specific

If the CLR_SUP bit is set to one in the error history directory parameter data (see 6.18.7.2), a CLR bit set to one specifies that the device server shall:

a) clear the portions of the error history that the device server allows to be cleared; and
b) ignore any application client error history specified in the parameter list.

If the CLR_SUP bit is set to one in the error history directory parameter data, a CLR bit set to zero specifies that the device server shall:

a) not clear the error history; and
b) process all application client error history specified in the parameter list.

If the CLR_SUP bit is set to zero in the error history directory parameter data, the device server shall ignore the CLR bit.

The TIMESTAMP field should contain:

a) a time based on the timestamp reported by the REPORT TIMESTAMP command, if the device server supports a device clock (see 5.2);
b) the number of milliseconds that have elapsed since midnight, 1 January 1970 UT; or
c) zero, if the application client is not able to determine the UT of the log entry.

The CODE SET field contains a code set enumeration (see 4.3.3) that indicates the format of the APPLICATION CLIENT ERROR HISTORY field.

The ERROR LOCATION FORMAT field (see table 344) specifies the format of the ERROR LOCATION field.

Table 344 — ERROR LOCATION FORMAT field

Code	Description
00h	No error history location specified by the application client.
01h	For block devices (see SBC-3 and RBC), the ERROR LOCATION field specifies the logical block address associated the specified application client error history. For other peripheral device types, this code is reserved.
02h to 7Fh	Reserved
80h to FFh	Vendor specific

The ERROR LOCATION LENGTH field specifies the length of the ERROR LOCATION field. The ERROR LOCATION LENGTH field value shall be a multiple of four. An ERROR LOCATION LENGTH field set to zero specifies that there is no error location information.

The APPLICATION CLIENT ERROR HISTORY LENGTH field specifies the length of the APPLICATION CLIENT ERROR HISTORY field. The APPLICATION CLIENT ERROR HISTORY LENGTH field value shall be a multiple of four. An APPLICATION CLIENT ERROR HISTORY field set to zero specifies that there is no vendor specific information.

The ERROR LOCATION field specifies the location at which the application client detected the error, in the format specified by the ERROR LOCATION FORMAT field.

The APPLICATION CLIENT ERROR HISTORY field specifies vendor specific application client error history (see 5.5.1).

7 Parameters for all device types

7.1 Overview

Parameters for all device types are defined in this clause as follows:

a) diagnostic parameters are defined in 7.2;
b) log parameters are defined in 7.3;
c) medium auxiliary memory attributes are defined in 7.4;
d) mode parameters are defined in 7.5;
e) protocol specific parameters are defined in 7.6;
f) security parameters are defined in 7.7; and
g) vital product data parameters are defined in 7.8.

7.2 Diagnostic parameters

7.2.1 Summary of diagnostic page codes

The page code assignments for diagnostic pages are summarized in table 345.

Table 345 — Summary of diagnostic page codes

Diagnostic page description	Page code	Reference
Defined by SES-3 for: a) standalone enclosure services devices (i.e., logical units with the PERIPHERAL DEVICE TYPE field set to 0Dh in standard INQUIRY data (see 6.6.2)); and b) attached enclosure services devices (i.e., logical units with the ENCSERV bit set to one in standard INQUIRY data).	01h to 2Fh	SES-3
Protocol Specific	3Fh	7.2.3
Supported Diagnostic Pages	00h	7.2.4
Restricted (see applicable command standard)	40h to 7Fh	
Vendor specific [a]		
Reserved	All other codes	
A numeric ordered listing of diagnostic page codes is provided in F.4.		
[a] The following page codes are vendor specific: 80h to FFh.		

7.2.2 Diagnostic page format for all device types

This subclause describes the diagnostic page structure and the diagnostic pages that are applicable to all SCSI devices. Diagnostic pages specific to each device type are described in the command standard that applies to that device type.

A SEND DIAGNOSTIC command (see 6.42) with a PF bit set to one specifies that the SEND DIAGNOSTIC parameter list consists of a single diagnostic page and that the data returned by the subsequent RECEIVE DIAGNOSTIC RESULTS command (see 6.27) that has the PCV bit set to zero. If the PF bit is supported in the SEND DIAGNOSTIC command (see 6.42), a SEND DIAGNOSTIC command with a PF bit set to one specifies that the subsequent RECEIVE DIAGNOSTIC RESULTS command shall use the diagnostic page format

defined in table 346. A RECEIVE DIAGNOSTIC RESULTS command with a PCV bit set to one specifies that the device server return a diagnostic page using the format defined in table 346.

Table 346 — Diagnostic page format

Bit / Byte	7	6	5	4	3	2	1	0
0	PAGE CODE							
1	Page code specific							
2	(MSB)			PAGE LENGTH (n-3)				
3								(LSB)
4				Diagnostic parameters				
•••								
n								

Each diagnostic page defines:

a) a function or operation that the device server shall perform as a result of a SEND DIAGNOSTIC command; or
b) the information being returned as a result of a RECEIVE DIAGNOSTIC RESULTS command with the PCV bit equal to one.

The diagnostic parameters contain data that is formatted according to the page code specified.

The PAGE CODE field (see 7.2.1) identifies the diagnostic page.

The PAGE LENGTH field indicates the number of bytes that follow in the diagnostic parameters. If the application client sends a SEND DIAGNOSTIC command with a parameter list containing a PAGE LENGTH field that results in the truncation of any parameter, then the device server shall terminate the command with CHECK CONDITION status, with the sense key set to ILLEGAL REQUEST, and the additional sense code set to INVALID FIELD IN PARAMETER LIST.

The diagnostic parameters are defined for each diagnostic page code. The diagnostic parameters within a diagnostic page may be defined differently in a SEND DIAGNOSTIC command than in a RECEIVE DIAGNOSTIC RESULTS command.

7.2.3 Protocol Specific diagnostic page

The Protocol Specific diagnostic page (see table 347) provides access to SCSI transport protocol specific diagnostic parameters.

Table 347 — Protocol Specific diagnostic page

Bit / Byte	7	6	5	4	3	2	1	0
0	PAGE CODE (3Fh)							
1	Reserved				PROTOCOL IDENTIFIER			
2	(MSB)			PAGE LENGTH (n-3)				
3								(LSB)
4	SCSI transport protocol specific diagnostic parameters							
•••								
n								

The PAGE CODE field is described in 7.2.2, and shall be set to 3Fh to indicate a Protocol Specific diagnostic page follows.

The PROTOCOL IDENTIFIER field contains one of the values shown in table 479 (see 7.6.1) to identify the SCSI transport protocol standard that defines the SCSI transport protocol specific diagnostic parameters.

The PAGE LENGTH field specifies the length in bytes of the following supported page list.

The SCSI transport protocol specific diagnostic parameters are defined by the SCSI transport protocol standard that corresponds to the value in the PROTOCOL IDENTIFIER field.

7.2.4 Supported Diagnostic Pages diagnostic page

The Supported Diagnostic Pages diagnostic page (see table 348) returns the list of diagnostic pages implemented by the device server. This diagnostic page shall be implemented if the device server implements the diagnostic page format option of the SEND DIAGNOSTIC command and RECEIVE DIAGNOSTIC RESULTS commands.

Table 348 — Supported Diagnostic Pages diagnostic page

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 diagnostic 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, with the sense key set to ILLEGAL REQUEST, and the additional sense code set to INVALID FIELD IN PARAMETER LIST.

This diagnostic page specifies that the device server shall make available the list of all supported diagnostic pages to be returned by a subsequent RECEIVE DIAGNOSTIC RESULTS command.

The definition of this diagnostic page for the RECEIVE DIAGNOSTIC RESULTS command includes the list of diagnostic pages supported by the device server.

The PAGE CODE field is described in 7.2.2, and shall be set to 00h to indicate a Supported Diagnostics Pages diagnostic page follows.

The PAGE LENGTH field indicates the length in bytes of the following supported page list.

The SUPPORTED PAGE LIST field shall contain a list of all diagnostic page codes, one per byte, implemented by the device server in ascending order beginning with page code 00h.

7.3 Log parameters

7.3.1 Summary of log page codes

The page code assignments for log pages are summarized in table 349.

Table 349 — Summary of log page codes

Log page name	Page code	Subpage code	Reference
Application Client	0Fh	00h	7.3.4
Buffer Over-Run/Under-Run	01h	00h	7.3.5
Cache Memory Statistics	19h	20h	7.3.6
General Statistics and Performance	19h	00h	7.3.7
Group Statistics and Performance (1 to 31)	19h	01h to 1Fh	7.3.8
Informational Exceptions	2Fh	00h	7.3.9
Last *n* Deferred Errors or Asynchronous Events	0Bh	00h	7.3.10
Last *n* Error Events	07h	00h	7.3.11
Non-Medium Error	06h	00h	7.3.12
Power Condition Transitions	1Ah	00h	7.3.13
Protocol Specific Port [a]	18h	00h to FEh	7.3.14
Read Error Counters	03h	00h	7.3.15
Read Reverse Error Counters	04h	00h	7.3.16
Self-Test Results	10h	00h	7.3.17
Start-Stop Cycle Counter	0Eh	00h	7.3.18
Supported Log Pages	00h	00h	7.3.19
Supported Log Pages and Subpages	00h	FFh	7.3.20
Supported Subpages	01h to 3Fh	FFh	7.3.21
Temperature	0Dh	00h	7.3.22
Verify Error Counters	05h	00h	7.3.23
Write Error Counters	02h	00h	7.3.24
Restricted (see applicable protocol standard)	08h to 0Ah	00h to FEh	
	0Ch	00h to FEh	
	11h to 17h	00h to FEh	
	1Bh to 2Eh	00h to FEh	
Vendor specific [b]			
Reserved	All other codes		

A numeric ordered listing of log pages codes and subpage codes is provided in F.5.

[a] Each SCSI transport protocol standard may define a different name for these log pages.
[b] Page codes 30h to 3Eh each with subpage codes 00h to FEh are vendor specific.

7.3.2 Log page structure and log parameter structure for all device types

7.3.2.1 Log page structure

This subclause describes the log page structure that is applicable to all SCSI devices. Log pages specific to each device type are described in the command standard that applies to that device type. The LOG SELECT command (see 6.7) supports the ability to send zero or more log pages. The LOG SENSE command (see 6.8) returns the log page specified by the combination of the PAGE CODE field and SUBPAGE CODE field in the CDB.

Each log page begins with a four-byte page header followed by zero or more variable length log parameters defined for that log page. The log page format is shown in table 350.

Table 350 — Log page format

Bit / Byte	7	6	5	4	3	2	1	0
0	DS	SPF	PAGE CODE					
1	SUBPAGE CODE							
2	(MSB)							
3	PAGE LENGTH (n-3)							(LSB)
	Log parameter(s)							
4	Log parameter (see 7.3.2.2) [first] (Length x)							
•••								
x+3								
	⋮							
n-y+1	Log parameter (see 7.3.2.2) [last] (Length y)							
•••								
n								

For the LOG SENSE command (see 6.8), the DS bit indicates whether log parameters in this log page are saved if the SP bit is set to one in the CDB. If the DS bit is set to zero, the log parameters are saved if the SP bit is set to one. If the DS bit is set to one, the log parameters are not saved. For the LOG SELECT command (see 6.7), the disable save (DS) bit operates in conjunction with the PCR bit, the SP bit, the PC field, and the PARAMETER LIST LENGTH field in the CDB.

If the subpage format (SPF) bit is set to zero, the SUBPAGE CODE field shall contain 00h. If the SPF bit is set to one, the SUBPAGE CODE field shall contain a value between 01h and FFh.

The PAGE CODE field indicates the number of the log page (see 7.3.1) that is being transferred.

The SUBPAGE CODE field indicates the subpage number of the log page (see 7.3.1) that is being transferred.

If an application client specifies values in the PAGE CODE field and SUBPAGE CODE field for a log page that is reserved or not implemented by the device server, then the device server shall terminate the LOG SELECT command with CHECK CONDITION status, with the sense key set to ILLEGAL REQUEST, and the additional sense code set to INVALID FIELD IN PARAMETER LIST.

If the PARAMETER LIST LENGTH field in a LOG SELECT command contains zero, the meanings for the PCR bit, SP bit, and PC field are defined in 6.7.2.

If the PARAMETER LIST LENGTH field in a LOG SELECT command contains a non-zero value (i.e., if a parameter list is being sent with the LOG SELECT command), then table 351 defines the meaning for the combinations of values for:

a) the PCR bit, the SP bit, and the PC field in the LOG SELECT command (see 6.7.1);
b) the DS bit in the log page header (see table 350); and
c) the PARAMETER CODE field and the FORMAT AND LINKING field in each log parameter (see 7.3.2.2.1).

Table 351 — LOG SELECT PCR bit, SP bit, and DS bit meanings when parameter list length is not zero

PCR bit	SP bit	DS bit	Description
0b	0b	xb	The device server shall set the specified values [a] to the values in the parameter list and shall not save any values to non-volatile media.
0b	1b	0b	The device server shall set the specified values [a] to the values in the parameter list and shall process the optional saving of log parameter values as follows: a) if default data counter values are specified (see table 187 in 6.7.1), no values shall be saved; b) if values other than default data counter values are specified and the device server implements saving of the specified values [a], then the device server shall save the specified values [a] in the parameter list to non-volatile media; or c) if values other than default values are specified and the device server does not implement saving of one or more of the specified values [a], then the device server shall terminate the command with CHECK CONDITION status, with the sense key set to ILLEGAL REQUEST, and the additional sense code set to INVALID FIELD IN PARAMETER LIST.
0b	1b	1b	The device server shall set the specified values [a] to the values in the parameter list and shall not save any values in the specified log page to non-volatile media.
1b	xb	xb	The device server shall terminate the command with CHECK CONDITION status, with the sense key set to ILLEGAL REQUEST, and the additional sense code set to INVALID FIELD IN CDB.

[a] The specified parameters are determined by the PARAMETER CODE field contents (see 7.3.2.2.1) in the LOG SELECT parameter data as well as by the PC field contents (see table 187 in 6.7.1) in the LOG SELECT CDB.

The PAGE LENGTH field indicates the length in bytes of the log parameters that follow. If the application client sends a log page length that results in the truncation of any parameter, the device server shall terminate the command with CHECK CONDITION status, with the sense key set to ILLEGAL REQUEST, and the additional sense code set to INVALID FIELD IN PARAMETER LIST.

7.3.2.2 Log parameter structure

7.3.2.2.1 Introduction

Most log pages contain one or more data structures called log parameters (see table 352). 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 or binary description related to a particular event.

Each log parameter begins with a four-byte parameter header followed by one or more bytes of parameter value data.

Table 352 — Log parameter

Bit / Byte	7	6	5	4	3	2	1	0
0	(MSB)			PARAMETER CODE				
1								(LSB)
2	Parameter control byte (see 7.3.2.2.2)							
	DU	Obsolete	TSD	ETC	TMC		FORMAT AND LINKING	
3	PARAMETER LENGTH (n-3)							
4	(MSB)			PARAMETER VALUE				
•••								
n								(LSB)

The PARAMETER CODE field identifies the log parameter being transferred. The device server shall return the log parameters in a log page in ascending order based on the value in their PARAMETER CODE field.

If an application client specifies a value in the PARAMETER CODE field in the LOG SELECT command parameter data that is reserved or not implemented by the logical unit, then the device server shall terminate the command with CHECK CONDITION status, with the sense key set to ILLEGAL REQUEST, and the additional sense code set to INVALID FIELD IN PARAMETER LIST.

The DU bit, TSD bit, ETC bit, TMC field, and FORMAT AND LINKING field are collectively referred to as the parameter control byte. The bits and fields in the parameter control byte are described in 7.3.2.2.2.

The PARAMETER LENGTH field specifies the length in bytes of the PARAMETER VALUE field. If the application client specifies a parameter length that results in the truncation of the PARAMETER VALUE field, then the device server shall terminate the command with CHECK CONDITION status, with the sense key set to ILLEGAL REQUEST, and the additional sense code set to INVALID FIELD IN PARAMETER LIST.

If the application client sends a value in a PARAMETER VALUE field that is outside the range supported by the logical unit, and rounding is implemented for that parameter, then the device server may:

a) round to an acceptable value and terminate the command as described in 5.8; or
b) terminate the command with CHECK CONDITION status, with the sense key set to ILLEGAL REQUEST, and the additional sense code set to INVALID FIELD IN PARAMETER LIST.

If the parameter data for one LOG SELECT command contains more than one log page and the log pages are not in ascending order by page code value then subpage code value, then the device server shall terminate the command with CHECK CONDITION status, with the sense key set to ILLEGAL REQUEST, and the additional sense code set to INVALID FIELD IN PARAMETER LIST.

If the parameter data for one LOG SELECT command contains more than one log parameter in any one log page and the log parameters are not in ascending order by parameter code value, then the device server shall

terminate the command with CHECK CONDITION status, with the sense key set to ILLEGAL REQUEST, and the additional sense code set to INVALID FIELD IN PARAMETER LIST.

Application clients should send LOG SENSE commands prior to sending LOG SELECT commands to determine supported log pages and page lengths.

The SCSI target device may provide independent sets of log parameters for each logical unit or for each combination of logical units and I_T nexuses. If the SCSI target device does not support independent sets of log parameters and any log parameters are changed that affect other I_T nexuses, then the device server shall establish a unit attention condition (see SAM-5) for the initiator port associated with every I_T nexus except the I_T nexus on which the LOG SELECT command was received, with the additional sense code set to LOG PARAMETERS CHANGED.

7.3.2.2.2 Parameter control byte

7.3.2.2.2.1 Introduction

The bits and fields in the parameter control byte are described in 7.3.2.2.2.

For cumulative log parameter values, indicated by the PC field (see table 187 in 6.7.1) of the LOG SELECT command and LOG SENSE command, the disable update (DU) bit is defined as follows:

a) DU set to zero indicates that the device server shall update the log parameter value to reflect all events that should be noted by that parameter; or
b) DU set to one 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 - While updating cumulative log parameter values, a device server may use volatile memory to hold these values until a LOG SELECT command or LOG SENSE command is received with an SP bit set to one or a vendor specific event occurs. As a result the updated cumulative log parameter values may be lost if a power cycle occurs.

If the PC field (see table 187 in 6.7.1) indicates that threshold values or default values are being processed, then the device server shall:

a) set the DU bit to zero, if a LOG SENSE command is being processed; and
b) ignore the DU bit, if a LOG SELECT command is being processed.

Regardless of the value in the PC field, the device server shall process ASCII format list log parameters (see 7.3.2.2.2.4) and binary format list log parameters (see 7.3.2.2.2.5) by:

a) setting the DU bit to zero, if a LOG SENSE command is being processed; and
b) ignoring the DU bit, if a LOG SELECT command is being processed.

A target save disable (TSD) bit set to zero indicates that the logical unit implicitly saves the log parameter at vendor specific intervals. This implicit saving operation shall be done frequently enough to ensure that the cumulative parameter values retain statistical significance (i.e., across power cycles). A TSD bit set to one indicates that either the logical unit does not implicitly save the log parameter or implicit saving of the log parameter has been disabled individually by an application client setting the TSD bit to one. An application client may disable the implicit saving for all log parameters without changing any TSD bits using the GLTSD bit in the Control mode page (see 7.5.8).

An enable threshold comparison (ETC) bit set to one indicates that a comparison to the threshold value is performed whenever the cumulative value is updated. An ETC bit set to 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 353) defines the basis for comparison of the cumulative value and threshold value. The TMC field is valid only if the ETC bit is set to one. The value of the TMC field is the same for cumulative parameters and threshold parameters.

Table 353 — Threshold met criteria (TMC) field

Code	Basis for comparison
00b	Every update of the cumulative value
01b	Cumulative value equal to threshold value
10b	Cumulative value not equal to threshold value
11b	Cumulative value greater than threshold value

If the ETC bit is set to one and the result of the comparison is true, the device server shall establish a unit attention condition (see SAM-5) for the initiator port associated with every I_T nexus, with the additional sense code set to THRESHOLD CONDITION MET.

The FORMAT AND LINKING field (see table 354) indicates the type of log parameter.

Table 354 — FORMAT AND LINKING field

Code	Log parameter type	Reference
00b	Bounded data counter	7.3.2.2.2
01b	ASCII format list	7.3.2.2.4
10b	Bounded data counter or unbounded data counter	7.3.2.2.2 or 7.3.2.2.3
11b	Binary format list	7.3.2.2.5

7.3.2.2.2.2 Parameter control byte values for bounded data counter parameters

The device server shall return parameter control byte values associated with LOG SENSE commands and process parameter control byte values associated with LOG SELECT commands as shown in table 355 for any log parameter that is defined to be a bounded data counter log parameter.

Table 355 — Parameter control byte values for bounded data counter parameters

Field or bit	Value associated with		Description
	LOG SENSE commands	LOG SELECT commands	
DU	0 or 1	0 or 1	If the DU bit is set to zero, the device server shall update the log parameter value to reflect all events that should be noted by that parameter. If the DU bit is set to one, 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.
TSD	0 or 1	0 or 1	If the TSD bit is set to zero, the device server shall save the log parameter to its medium at vendor specific intervals. If the TSD bit is set to one, the device server shall save the log parameter to its medium.
ETC	0 or 1	0 or 1	If the ETC bit is set to one, a comparison to the threshold value is performed whenever the cumulative value is updated. If the ETC bit is set to zero, a comparison shall not be performed.
TMC	any	any	The TMC field (see table 353 in 7.3.2.2.2.1) defines the basis for comparison of the cumulative and threshold values. The TMC field is valid only if the ETC bit is set to one.
FORMAT AND LINKING	00b or 10b	00b or 10b	The log parameter is a data counter (see table 354 in 7.3.2.2.2.1) and the handling of a parameter that reaches its maximum value is described in this subclause

If a LOG SELECT command contains a bounded data counter log parameter in which the parameter control byte values differ from those shown in table 355, then the device server shall terminate the command with CHECK CONDITION status, with the sense key set to ILLEGAL REQUEST, and the additional sense code set to INVALID FIELD IN PARAMETER LIST.

Each bounded data counter log parameter contains one saturating counter that is:

a) associated with one or more events; and
b) incremented whenever one of these events occurs.

In a bounded data counter log parameter, if the counter has associated with it a vendor specific maximum value, then upon reaching this maximum value, the data counter shall not be incremented (i.e., its value does not wrap).

In a bounded data counter log parameter, if the counter reaches its maximum value (i.e., saturates), the device server shall:

a) set the DU bit to one;
b) handle other bounded data counter log parameters in the log page based on the contents of the FORMAT AND LINKING field in each other log parameter as follows:
   A) if the FORMAT AND LINKING field is set to 00b, then that other log parameter shall stop incrementing until reinitialized by a LOG SELECT command; or

B) if the FORMAT AND LINKING field is set to 10b, then that other log parameter shall not stop incrementing, but may be reinitialized by a LOG SELECT command.

and

c) not alter the handling of other log parameters in the log page that are:
   A) unbounded data counter log parameters (see 7.3.2.2.2.3);
   B) ASCII format list log parameters (see 7.3.2.2.2.4); and
   C) binary format list log parameters (see 7.3.2.2.2.5).

The processing of a command shall not be altered as a result of the counter in a bounded data counter log parameter reaching its maximum value (i.e., saturates). If the RLEC bit is set to one in the Control mode page (see 7.5.8) and the processing of a command encounters no exception conditions other than the counter in a bounded data counter log parameter reaching its maximum value, then the device server shall terminate the command with CHECK CONDITION status, with the sense key set to RECOVERED ERROR, and the additional sense code set to LOG COUNTER AT MAXIMUM.

7.3.2.2.2.3 Parameter control byte values for unbounded data counter parameters

The device server shall return parameter control byte values associated with LOG SENSE commands and process parameter control byte values associated with LOG SELECT commands as shown in table 356 for any log parameter that is defined to be an unbounded data counter log parameter.

Table 356 — Parameter control byte values for unbounded data counter parameters

Field or bit	Value associated with		Description
	LOG SENSE commands	LOG SELECT commands	
DU	0 or 1	0 or 1	If the DU bit is set to zero, the device server shall update the log parameter value or values to reflect all events that should be noted by that parameter. If the DU bit is set to one, the device server shall not update the log parameter value or values except in response to a LOG SELECT command that specifies a new value for the parameter.
TSD	0 or 1	0 or 1	If the TSD bit is set to zero, the device server shall save the log parameter to its medium at vendor specific intervals. If the TSD bit is set to one, the device server shall save the log parameter to its medium.
ETC	0	0	Threshold comparisons are not performed for unbounded data counter parameters
TMC	00b	ignored	Threshold comparisons are not performed for unbounded data counter parameters
FORMAT AND LINKING	10b	10b	The log parameter is a data counter for which saturation of another log parameter does not affect the incrementing of this log parameter (see table 354 in 7.3.2.2.2.1).

If a LOG SELECT command contains an unbounded data counter log parameter in which the parameter control byte values differ from those shown in table 356, then the device server shall terminate the command with CHECK CONDITION status, with the sense key set to ILLEGAL REQUEST, and the additional sense code set to INVALID FIELD IN PARAMETER LIST.

Each unbounded data counter log parameter contains one or more saturating counters or wrapping counters. The description of each counter field in the log parameter defines when the device server modifies the contents of the counter that is transferred in that field.

Changes in an unbounded data counter (e.g., a counter reaching saturation or another maximum value) shall not affect the handling of other log parameters in the log page. The processing of a command and the status returned by that command shall not be altered as a result of a counter in an unbounded data counter log parameter saturating or reaching its maximum value.

The device server shall not change the value in the DU bit in an unbounded data counter log parameter unless requested to do so by a LOG SELECT command.

7.3.2.2.2.4 Parameter control byte values for ASCII format list log parameters

The device server shall return parameter control byte values associated with LOG SENSE commands and process parameter control byte values associated with LOG SELECT commands as shown in table 357 for any log parameter that is defined to be an ASCII format (see 4.3.1) list log parameter.

Table 357 — Parameter control byte values for ASCII format list log parameters

Field or bit	Value associated with		Description
	LOG SENSE commands	LOG SELECT commands	
DU	0	ignored	The DU bit is not defined for list parameters.
TSD	0 or 1	0 or 1	If the TSD bit is set to zero, the device server shall save the log parameter to its medium at vendor specific intervals. If the TSD bit is set to one, the device server shall save the log parameter to its medium.
ETC	0	0	Threshold comparisons are not performed for list parameters.
TMC	00b	ignored	Threshold comparisons are not performed for list parameters.
FORMAT AND LINKING	01b	01b	The log parameter is an ASCII format list parameter (see table 354 in 7.3.2.2.2.1).

If a LOG SELECT command contains an ASCII format list log parameter in which the parameter control byte values differ from those shown in table 357, then the device server shall terminate the command with CHECK CONDITION status, with the sense key set to ILLEGAL REQUEST, and the additional sense code set to INVALID FIELD IN PARAMETER LIST.

7.3.2.2.2.5 Parameter control byte values for binary format list log parameters

The device server shall return parameter control byte values associated with LOG SENSE commands and process parameter control byte values associated with LOG SELECT commands as shown in table 358 for any log parameter that is defined to be a binary format list log parameter.

Table 358 — Parameter control byte values for binary format list log parameters

Field or bit	Value associated with		Description
	LOG SENSE commands	LOG SELECT commands	
DU	0	ignored	The DU bit is not defined for list parameters.
TSD	0 or 1	0 or 1	If the TSD bit is set to zero, the device server shall save the log parameter to its medium at vendor specific intervals. If the TSD bit is set to one, the device server shall save the log parameter to its medium.
ETC	0	0	Threshold comparisons are not performed for list parameters.
TMC	00b	ignored	Threshold comparisons are not performed for list parameters.
FORMAT AND LINKING	11b	11b	The log parameter is an binary format list parameter (see table 354 in 7.3.2.2.2.1).

If a LOG SELECT command contains a binary format list log parameter in which the parameter control byte values differ from those shown in table 358, then the device server shall terminate the command with CHECK CONDITION status, with the sense key set to ILLEGAL REQUEST, and the additional sense code set to INVALID FIELD IN PARAMETER LIST.

7.3.3 Resetting and setting log parameters

In a LOG SELECT command, an application client may specify that:

a) all the parameters in a log page or pages are to be reset (i.e., the PCR bit is set to one and the PARAMETER LIST LENGTH field is set to zero); or
b) individual parameters in log page are to be changed to specified new values (i.e., the PCR bit is set to zero and the PARAMETER LIST LENGTH field is not set to zero).

The device server handling of these requests to reset or change the cumulative value of a log parameter depends on the log parameter that is being reset or changed. The keywords that describe how the device server handles these requests are defined in table 359.

Table 359 — Keywords for resetting or changing log parameter cumulative values

Keyword	Device server handling when	
	PCR bit is set to one [a]	**PCR bit is set to zero** [b]
Always	Reset the log parameter cumulative value.	Change the log parameter cumulative value.
Reset Only	Reset the log parameter cumulative value.	If any changes are requested in the PARAMETER VALUE field of the log parameter cumulative value, then: a) terminate the command with CHECK CONDITION status, with the sense key set to ILLEGAL REQUEST, and the additional sense code set to INVALID FIELD IN PARAMETER LIST; and b) do not make any requested changes in any field in any log parameter cumulative value in any log page.
Never	Do not reset the log parameter cumulative value. Table 189 (see 6.7.1) describes error conditions that may apply.	

[a] If the PCR bit is set to one and the PARAMETER LIST LENGTH field is not set to zero, the device server shall terminate the LOG SELECT command (see table 351 in 7.3.2.1).

[b] If the PCR bit is set to zero and the PARAMETER LIST LENGTH field is set to zero, no log parameters are changed (see 6.7.2).

7.3.4 Application Client log page

7.3.4.1 Overview

Using the format shown in table 361, the Application Client log page provides a place for application clients to store information. The parameter codes for the Application Client log page are listed in table 360.

Table 360 — Application Client log page parameter codes

Parameter code	Description	Resettable or Changeable [a]	Reference	Support
0000h to 003Fh	General Usage Application Client	Always	7.3.4.2	Mandatory
0040h to 0FFFh				Optional
all others	Reserved			

[a] The keywords in this column – Always, Reset Only, and Never – are defined in 7.3.3.

The Application Client log page has the format shown in table 361.

Table 361 — Application Client log page

Bit / Byte	7	6	5	4	3	2	1	0
0	DS	SPF (0b)	PAGE CODE (0Fh)					
1	SUBPAGE CODE (00h)							
2	(MSB)	PAGE LENGTH (n-3)						
3								(LSB)
	Application client log parameters							
4 •••	Application client log parameter (see 7.3.4.2) [first]							
	⋮							
••• n	Application client log parameter (see 7.3.4.2) [last]							

The DS bit, SPF bit, PAGE CODE field, SUBPAGE CODE field, and PAGE LENGTH field are described in 7.3.2. The SPF bit, PAGE CODE field, and SUBPAGE CODE field shall be set as shown in table 361 for the Application Client log page.

Each application client log parameter contains the information described in 7.3.4.2.

7.3.4.2 General Usage Application Client log parameter

The General Usage Application Client log parameter has the format shown in table 362. This information may be used to describe the system configuration and system problems, but the specific definition of the data is application client specific.

Table 362 — General Usage Application Client log parameter

Bit / Byte	7	6	5	4	3	2	1	0
0	(MSB)	PARAMETER CODE (see table 360)						
1								(LSB)
2	Parameter control byte – binary format list log parameter (see 7.3.2.2.2.5)							
	DU	Obsolete	TSD	ETC	TMC		FORMAT AND LINKING	
3	PARAMETER LENGTH (FCh)							
4 ••• 255	GENERAL USAGE PARAMETER BYTES							

The PARAMETER CODE field is described in 7.3.2.2.1, and shall be set as shown in table 360 for the General Usage Application Client log parameter.

The DU bit, TSD bit, ETC bit, TMC field, and FORMAT AND LINKING field are described in 7.3.2.2.2.1. These fields shall be set as described for a binary format list log parameter (see 7.3.2.2.2.5) for the General Usage Application Client log parameter.

The contents of the GENERAL USAGE PARAMETER BYTES field represent data sent to the device server in a previous LOG SELECT command. If a previous LOG SELECT command has not occurred, the contents of the GENERAL USAGE PARAMETER BYTES field are vendor specific.

7.3.5 Buffer Over-Run/Under-Run log page

7.3.5.1 Overview

Using the format shown in table 364, the Buffer Over-Run/Under-Run log page defines bounded data counters that record the number of buffer over-runs or under-runs detected by the device server. The parameter codes for the Buffer Over-Run/Under-Run log page are listed in table 363.

Table 363 — Buffer Over-Run/Under-Run log page parameter codes (part 1 of 2)

Parameter code [a]	Incremented once per:			Resettable or Changeable [b]	Reference	Support
	Over- or Under-run	Experienced by (or during)	Problem detected			
0000h	Under-run	Undefined	Undefined	Reset Only	7.3.5.2	At least one [c]
0001h	Over-run					
0020h	Under-run	A command				
0021h	Over-run					
0040h	Under-run	An I_T nexus				
0041h	Over-run					
0080h	Under-run	A unit of time [d]				
0081h	Over-run					
0002h	Under-run	Undefined	Service delivery subsystem busy	Reset Only	7.3.5.2	At least one [c]
0003h	Over-run					
0022h	Under-run	A command				
0023h	Over-run					
0042h	Under-run	An I_T nexus				
0043h	Over-run					
0082h	Under-run	A unit of time [d]				
0083h	Over-run					

[a] See SPC-2 for a description of how these parameter codes are derived.

[b] The keywords in this column – Always, Reset Only, and Never – are defined in 7.3.3.

[c] If the Buffer Over-Run/Under-Run log page is supported, at least one of the parameter codes listed in this table shall be supported.

[d] The size of the unit of time is vendor specific.

Table 363 — Buffer Over-Run/Under-Run log page parameter codes (part 2 of 2)

Parameter code [a]	Incremented once per:			Resettable or Changeable [b]	Reference	Support
	Over- or Under-run	Experienced by (or during)	Problem detected			
0004h	Under-run	Undefined	Transfer rate too slow	Reset Only	7.3.5.2	At least one [c]
0005h	Over-run					
0024h	Under-run	A command				
0025h	Over-run					
0044h	Under-run	An I_T nexus				
0045h	Over-run					
0084h	Under-run	A unit of time [d]				
0085h	Over-run					
all others	Reserved					

[a] See SPC-2 for a description of how these parameter codes are derived.

[b] The keywords in this column – Always, Reset Only, and Never – are defined in 7.3.3.

[c] If the Buffer Over-Run/Under-Run log page is supported, at least one of the parameter codes listed in this table shall be supported.

[d] The size of the unit of time is vendor specific.

A buffer over-run or under-run may occur if a SCSI initiator device does not transfer data to or from the logical unit's buffer fast enough to keep up with reading or writing the media. A buffer over-run condition may occur during a read operation if 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 if a buffer empty condition prevents continued transfer of data to the media from the buffer.

The Buffer Over-Run/Under-Run log page has the format shown in table 364.

Table 364 — Buffer Over-Run/Under-Run log page

Bit / Byte	7	6	5	4	3	2	1	0
0	DS	SPF (0b)		PAGE CODE (01h)				
1				SUBPAGE CODE (00h)				
2	(MSB)			PAGE LENGTH (n-3)				
3								(LSB)
	Buffer over-run/under-run log parameters							
4 •••	Buffer Over-run/Under-run log parameter (see 7.3.5.2) [first]							
	⋮							
••• n	Buffer Over-run/Under-run log parameter (see 7.3.5.2) [last]							

The DS bit, SPF bit, PAGE CODE field, SUBPAGE CODE field, and PAGE LENGTH field are described in 7.3.2. The SPF bit, PAGE CODE field, and SUBPAGE CODE field shall be set as shown in table 364 for the Buffer Over-Run/Under-Run log page.

Each Buffer Over-run/Under-run log parameter contains the information described in 7.3.5.2.

7.3.5.2 Buffer Over-run/Under-run log parameter

The Buffer Over-run/Under-run log parameter has the format shown in table 365.

Table 365 — Buffer Over-run/Under-run log parameter

Bit / Byte	7	6	5	4	3	2	1	0
0	(MSB)							
1	PARAMETER CODE (see table 363)							(LSB)
2	Parameter control byte – bounded data counter log parameter (see 7.3.2.2.2.2)							
	DU	Obsolete	TSD	ETC	TMC		FORMAT AND LINKING	
3	PARAMETER LENGTH (n-3)							
4	(MSB)							
•••	OVER-RUN/UNDER-RUN COUNTER							
n								(LSB)

The PARAMETER CODE field is described in 7.3.2.2.1, and shall be set as shown in table 363 for the Buffer Over-run/Under-run log parameter log parameter.

The DU bit, TSD bit, ETC bit, TMC field, and FORMAT AND LINKING field are described in 7.3.2.2.2.1. These fields shall be set as described for a bounded data counter log parameter (see 7.3.2.2.2.2) for the Buffer Over-run/Under-run log parameter log parameter.

The PARAMETER LENGTH field is described in 7.3.2.2.1.

The OVER-RUN/UNDER-RUN COUNTER field contains the value for the counter described by the contents of the PARAMETER CODE field.

Each counter contains the total number of times buffer over-run or under-run conditions have occurred since the last time the counter was reset. The counter shall be incremented for each occurrence of a buffer under-run or over-run condition and may be incremented more than once for multiple occurrences during the processing of a single command.

7.3.6 Cache Memory Statistics log page

7.3.6.1 Overview

Using the format shown in table 368, the Cache Memory Statistics log page contains statistics and performance results identified by the parameter codes listed in table 366.

Table 366 — Cache Memory Statistics log page parameter codes

Parameter code	Description	Resettable or Changeable [a]	Reference	Support
0001h	Read Cache Memory Hits	Reset Only	7.3.6.2	At least one [b]
0002h	Reads To Cache Memory	Reset Only	7.3.6.3	
0003h	Write Cache Memory Hits	Reset Only	7.3.6.4	
0004h	Writes From Cache Memory	Reset Only	7.3.6.5	
0005h	Time From Last Hard Reset	Never	7.3.6.6	
0006h	Time Interval	Never	7.3.6.7	
all others	Reserved			

[a] The keywords in this column – Always, Reset Only, and Never – are defined in 7.3.3.

[b] If the Cache Memory Statistics log page is supported, at least one of the parameter codes listed in this table shall be supported.

The Cache Memory Statistics log page provides the following statistics and performance results associated with the addressed logical unit:

a) Cache Memory Statistics log parameters:
   A) number of read cache memory hits;
   B) number of reads to cache memory;
   C) number of write cache memory hits; and
   D) number of writes from cache memory;
b) Hard Reset log parameter:
   A) time from last hard reset (see SAM-5); and
c) Time Interval log parameter:
   A) time interval.

In the Cache Memory Statistics log page, read commands and write commands are those shown in table 367.

Table 367 — Cache Memory Statistics log page commands

Read commands [a]	Write commands [a]
READ(10) READ(12) READ(16) READ(32) PRE-FETCH(10) PRE-FETCH(16)	SYNCHRONIZE CACHE(10) SYNCHRONIZE CACHE(16) WRITE(10) WRITE(12) WRITE(16) WRITE(32) WRITE AND VERIFY(10) WRITE AND VERIFY(12) WRITE AND VERIFY(16) WRITE AND VERIFY(32)

[a] All commands in this column are defined in SBC-3.

The Cache Memory Statistics log page has the format shown in table 368.

Table 368 — Cache Memory Statistics log page

Bit / Byte	7	6	5	4	3	2	1	0
0	DS	SPF (1b)	PAGE CODE (19h)					
1	SUBPAGE CODE (20h)							
2	(MSB)	PAGE LENGTH (n-3)						
3								(LSB)
	Cache memory statistics log parameters							
4 •••	Cache memory statistics log parameter (see table 366) [first]							
	⋮							
••• n	Cache memory statistics log parameter (see table 366) [last]							

The DS bit, SPF bit, PAGE CODE field, SUBPAGE CODE field, and PAGE LENGTH field are described in 7.3.2. The SPF bit, PAGE CODE field, and SUBPAGE CODE field shall be set as shown in table 368 for the Cache Memory Statistics log page.

The contents of each cache memory statistics log parameter depends on the value in its PARAMETER CODE field (see table 366).

7.3.6.2 Read Cache Memory Hits log parameter

The Read Cache Memory Hits log parameter has the format shown in table 369.

Table 369 — Read Cache Memory Hits log parameter

Bit / Byte	7	6	5	4	3	2	1	0
0	(MSB)	PARAMETER CODE (0001h)						
1								(LSB)
2	Parameter control byte – bounded data counter log parameter (see 7.3.2.2.2.2)							
	DU	Obsolete	TSD	ETC	TMC		FORMAT AND LINKING	
3	PARAMETER LENGTH (08h)							
4 •••	(MSB)	NUMBER OF READ CACHE MEMORY HITS						
11								(LSB)

The PARAMETER CODE field is described in 7.3.2.2.1, and shall be set as shown in table 369 for the Read Cache Memory Hits log parameter.

The DU bit, TSD bit, ETC bit, TMC field, and FORMAT AND LINKING field are described in 7.3.2.2.2.1. These fields shall be set as described for a bounded data counter log parameter (see 7.3.2.2.2.2) for the Read Cache Memory Hits log parameter.

The PARAMETER LENGTH field is described in 7.3.2.2.1, and shall be set as shown in table 369 for the Read Cache Memory Hits log parameter.

The NUMBER OF READ CACHE MEMORY HITS field indicates the number of read commands (see table 367 in 7.3.6.1) received on an I_T nexus that resulted in:

a) user data being read from cache memory; and
b) no user data read from the medium before the read command being processed is completed.

The NUMBER OF READ CACHE MEMORY HITS field shall not be modified as a result of any read command that contains an FUA bit (see SBC-3) set to one or an FUA_NV bit (see SBC-2) set to one.

The contents of the NUMBER OF READ CACHE MEMORY HITS field shall be set to zero as part of processing a hard reset condition (see SAM-5).

7.3.6.3 Reads To Cache Memory log parameter

The Reads To Cache Memory log parameter has the format shown in table 370.

Table 370 — Reads To Cache Memory log parameter

Bit / Byte	7	6	5	4	3	2	1	0
0	(MSB)							
1	PARAMETER CODE (0002h)							(LSB)
2	Parameter control byte – bounded data counter log parameter (see 7.3.2.2.2)							
	DU	Obsolete	TSD	ETC	TMC		FORMAT AND LINKING	
3	PARAMETER LENGTH (08h)							
4	(MSB)							
•••	NUMBER OF READS TO CACHE MEMORY							
11								(LSB)

The PARAMETER CODE field is described in 7.3.2.2.1, and shall be set as shown in table 370 for the Reads To Cache Memory log parameter.

The DU bit, TSD bit, ETC bit, TMC field, and FORMAT AND LINKING field are described in 7.3.2.2.2.1. These fields shall be set as described for a bounded data counter log parameter (see 7.3.2.2.2.2) for the Reads To Cache Memory log parameter.

The PARAMETER LENGTH field is described in 7.3.2.2.1, and shall be set as shown in table 370 for the Reads To Cache Memory log parameter.

The NUMBER OF READS TO CACHE MEMORY field indicates the number of read commands (see table 367 in 7.3.6.1) that move user data from the medium to cache memory.

The NUMBER OF READS TO CACHE MEMORY field shall not be modified as a result of any read command that contains an FUA bit (see SBC-3) set to one or an FUA_NV bit (see SBC-2) set to one.

The contents of the NUMBER OF READS TO CACHE MEMORY field shall be set to zero as part of processing a hard reset condition (see SAM-5).

7.3.6.4 Write Cache Memory Hits log parameter

The Write Cache Memory Hits log parameter has the format shown in table 371.

Table 371 — Write Cache Memory Hits log parameter

Bit / Byte	7	6	5	4	3	2	1	0
0	(MSB)							
1	PARAMETER CODE (0003h)							(LSB)
2	Parameter control byte – bounded data counter log parameter (see 7.3.2.2.2.2)							
	DU	Obsolete	TSD	ETC	TMC		FORMAT AND LINKING	
3	PARAMETER LENGTH (08h)							
4	(MSB)							
•••	NUMBER OF WRITES FROM CACHE MEMORY							
11								(LSB)

The PARAMETER CODE field is described in 7.3.2.2.1, and shall be set as shown in table 371 for the Write Cache Memory Hits log parameter.

The DU bit, TSD bit, ETC bit, TMC field, and FORMAT AND LINKING field are described in 7.3.2.2.2.1. These fields shall be set as described for a bounded data counter log parameter (see 7.3.2.2.2.2) for the Writes From Cache Memory log parameter.

The PARAMETER LENGTH field is described in 7.3.2.2.1, and shall be set as shown in table 371 for the Write Cache Memory Hits log parameter.

The NUMBER OF WRITES FROM CACHE MEMORY field indicates the number of write commands (see table 367 in 7.3.6.1) received on an I_T nexus that resulted in:

a) user data being written to cache memory; and
b) no user data written to the medium before the write command being processed is completed.

The NUMBER OF WRITES FROM CACHE MEMORY field shall not be modified as a result of any write command that contains an FUA bit (see SBC-3) set to one or an FUA_NV bit (see SBC-2) set to one.

The contents of the NUMBER OF WRITES FROM CACHE MEMORY field shall be set to zero as part of processing a hard reset condition (see SAM-5).

7.3.6.5 Writes From Cache Memory log parameter

The Writes From Cache Memory log parameter has the format shown in table 372.

Table 372 — Writes From Cache Memory log parameter

Bit / Byte	7	6	5	4	3	2	1	0
0	(MSB)							
1	PARAMETER CODE (0004h)							(LSB)
2	Parameter control byte – bounded data counter log parameter (see 7.3.2.2.2.2)							
	DU	Obsolete	TSD	ETC	TMC		FORMAT AND LINKING	
3	PARAMETER LENGTH (08h)							
4	(MSB)							
•••	NUMBER OF WRITES FROM CACHE MEMORY							
11								(LSB)

The PARAMETER CODE field is described in 7.3.2.2.1, and shall be set as shown in table 372 for the Writes From Cache Memory log parameter.

The DU bit, TSD bit, ETC bit, TMC field, and FORMAT AND LINKING field are described in 7.3.2.2.2.1. These fields shall be set as described for a bounded data counter log parameter (see 7.3.2.2.2.2) for the Writes From Cache Memory log parameter.

The PARAMETER LENGTH field is described in 7.3.2.2.1, and shall be set as shown in table 372 for the Writes From Cache Memory log parameter.

The NUMBER OF WRITES FROM CACHE MEMORY field indicates the number of write commands (see table 367 in 7.3.6.1) that move user data from cache memory to the medium.

The NUMBER OF WRITES FROM CACHE MEMORY field shall not be modified as a result of any write command that contains an FUA bit (see SBC-3) set to one or an FUA_NV bit (see SBC-2) set to one.

The contents of the NUMBER OF WRITES FROM CACHE MEMORY field shall be set to zero as part of processing a hard reset condition (see SAM-5).

7.3.6.6 Time From Last Hard Reset log parameter

The Time From Last Hard Reset log parameter has the format shown in table 373.

Table 373 — Time From Last Hard Reset log parameter

Bit / Byte	7	6	5	4	3	2	1	0
0	(MSB)							
1	PARAMETER CODE (0005h)							(LSB)
2	Parameter control byte – bounded data counter log parameter (see 7.3.2.2.2.2)							
	DU	Obsolete	TSD	ETC	TMC		FORMAT AND LINKING	
3	PARAMETER LENGTH (08h)							
4	(MSB)							
•••	LAST HARD RESET INTERVALS							
11								(LSB)

The PARAMETER CODE field is described in 7.3.2.2.1, and shall be set as shown in table 373 for the Time From Last Hard Reset log parameter.

The DU bit, TSD bit, ETC bit, TMC field, and FORMAT AND LINKING field are described in 7.3.2.2.2.1. These fields shall be set as described for a bounded data counter log parameter (see 7.3.2.2.2.2) for the Time From Last Hard Reset log parameter.

The PARAMETER LENGTH field is described in 7.3.2.2.1, and shall be set as shown in table 373 for the Time From Last Hard Reset log parameter.

The LAST HARD RESET INTERVALS field indicates the number of time intervals that have occurred since a hard reset was processed by the logical unit.

The time since a hard reset was processed by the logical unit is calculated as follows:

time = (time intervals since last hard reset × time interval)

where:

time intervals since last hard reset is the contents of the LAST HARD RESET INTERVALS field; and

time interval is the value represented in the time interval descriptor of the Time Interval log parameter (see table 374 in 7.3.6.7).

7.3.6.7 Time Interval log parameter

The Time Interval log parameter has the format shown in table 374.

Table 374 — Time Interval log parameter

Bit / Byte	7	6	5	4	3	2	1	0
0	(MSB)							
1	PARAMETER CODE							(LSB)
2	Parameter control byte – binary format list log parameter (see 7.3.2.2.2.5)							
	DU	Obsolete	TSD	ETC	TMC		FORMAT AND LINKING	
3	PARAMETER LENGTH (08h)							
4	Time interval descriptor (see table 375)							
•••								
11								

The PARAMETER CODE field is described in 7.3.2.2.1. A PARAMETER CODE field set to:

a) 0003h identifies the log parameter being transferred as the Time Interval log parameter in the General Statistics and Performance log page (see 7.3.7); or
b) 0006h identifies the log parameter being transferred as the Time Interval log parameter in the Cache Memory Statistics log page (see 7.3.6).

The DU bit, TSD bit, ETC bit, TMC field, and FORMAT AND LINKING field are described in 7.3.2.2.2.1. These fields shall be set as described for a binary format list log parameter (see 7.3.2.2.2.5) for the log parameters described in this subclause.

The PARAMETER LENGTH field is described in 7.3.2.2.1, and shall be set as shown in table 374 for the log parameters described in this subclause.

The time interval descriptor (see table 375) contains the time interval in seconds used in various time interval fields in:

a) the Time From Last Hard Reset log parameter (see 7.3.6.6);
b) the General Access Statistics and Performance log parameter (see 7.3.7.2);
c) the Force Unit Access Statistics and Performance log parameter (see 7.3.7.4);
d) the Group n Statistics and Performance log parameter (see 7.3.8.2); and
e) the Group n Force Unit Access Statistics and Performance log parameter (see 7.3.8.3).

Table 375 — Time interval descriptor

Bit / Byte	7	6	5	4	3	2	1	0
0	(MSB)							
•••	EXPONENT							
3								(LSB)
4	(MSB)							
•••	INTEGER							
7								(LSB)

The EXPONENT field contains the negative power of 10 exponent to multiply with the INTEGER field (e.g., a value of 9 represents 10^{-9})

When multiplied by the exponent, the INTEGER field contains the value that represents one time interval (e.g., a value of 5 in the INTEGER field and a value of 9 in the EXPONENT field represents a time interval of 5×10^{-9} seconds or 5 nanoseconds).

7.3.7 General Statistics and Performance log pages

7.3.7.1 Overview

Using the format shown in table 378, the General Statistics and Performance log page collects statistics and performance information for all read CDBs and write CDBs based on the parameter codes listed in table 376.

Table 376 — General Statistics and Performance log page parameter codes

Parameter code	Description	Resettable or Changeable [a]	Reference	Support
0001h	General Access Statistics and Performance	Reset Only	7.3.7.2	Mandatory
0002h	Idle Time	Reset Only	7.3.7.3	Mandatory
0003h	Time Interval	Never	7.3.6.7	Mandatory
0004h	Force Unit Access Statistics and Performance	Reset Only	7.3.7.4	Optional
all others	Reserved			

[a] The keywords in this column – Always, Reset Only, and Never – are defined in 7.3.3.

The General Statistics and Performance log page provides the following statistics and performance results associated to the addressed logical unit:

a) Statistics and Performance log parameters:
   A) number of read commands;
   B) number of write commands;
   C) number of read logical blocks transferred by a target port;
   D) number of write logical blocks transferred by a target port;
   E) read command processing time;
   F) write command processing time;
   G) sum of the command weights of the read commands plus write commands;
   H) sum of the weighted command time of the read commands plus write commands;
b) Idle Time log parameter:
   A) idle time;
c) Time Interval log parameter:
   A) time interval;
   and
d) Force Unit Access Statistics and Performance log parameters:
   A) number of read commands with the FUA bit (see SBC-3) set to one;
   B) number of write commands with the FUA bit set to one;
   C) number of read commands with the FUA_NV bit (see SBC-2) set to one;
   D) number of write commands with the FUA_NV bit set to one; and
   E) read command with the FUA bit set to one processing intervals;
   F) write command with the FUA bit set to one processing intervals;
   G) read command with the FUA_NV bit set to one processing intervals; and
   H) write command with the FUA_NV bit set to one processing intervals.

In the General Statistics and Performance log page and the Group Statistics and Performance (n) log pages (see 7.3.8), read commands and write commands are those shown in table 377.

Table 377 — Statistics and Performance log pages commands

Read commands [a]	Write commands [a]
POPULATE TOKEN READ(10) READ(12) READ(16) READ(32)	WRITE USING TOKEN UNMAP WRITE(10) WRITE(12) WRITE(16) WRITE(32) WRITE AND VERIFY(10) WRITE AND VERIFY(12) WRITE AND VERIFY(16) WRITE AND VERIFY(32)
[a] See SBC-3.	

The General Statistics and Performance log page has the format shown in table 378.

Table 378 — General Statistics and Performance log page

Bit Byte	7	6	5	4	3	2	1	0
0	DS	SPF (0b)	PAGE CODE (19h)					
1	SUBPAGE CODE (00h)							
2	(MSB)	PAGE LENGTH (n-3)						
3								(LSB)
	General statistics and performance log parameters							
4	General statistics and performance log parameter (see table 376) [first]							
•••								
	⋮							
	General statistics and performance log parameter (see table 376) [last]							
•••								
n								

The DS bit, SPF bit, PAGE CODE field, SUBPAGE CODE field, and PAGE LENGTH field are described in 7.3.2. The SPF bit, PAGE CODE field, and SUBPAGE CODE field shall be set as shown in table 378 for the General Statistics and Performance log page.

The contents of each general statistics and performance log parameter depends on the value in its PARAMETER CODE field (see table 376).

7.3.7.2 General Access Statistics and Performance log parameter

The General Access Statistics and Performance log parameter has the format shown in table 379.

Table 379 — General Access Statistics and Performance log parameter

Bit / Byte	7	6	5	4	3	2	1	0
0	(MSB)			PARAMETER CODE (0001h)				
1								(LSB)
2	Parameter control byte – unbounded data counter log parameter (see 7.3.2.2.2.3)							
	DU	Obsolete	TSD	ETC	TMC		FORMAT AND LINKING	
3	PARAMETER LENGTH (40h)							
4	(MSB)			NUMBER OF READ COMMANDS				
•••								
11								(LSB)
12	(MSB)			NUMBER OF WRITE COMMANDS				
•••								
19								(LSB)
20	(MSB)			NUMBER OF LOGICAL BLOCKS RECEIVED				
•••								
27								(LSB)
28	(MSB)			NUMBER OF LOGICAL BLOCKS TRANSMITTED				
•••								
35								(LSB)
36	(MSB)			READ COMMAND PROCESSING INTERVALS				
•••								
43								(LSB)
44	(MSB)			WRITE COMMAND PROCESSING INTERVALS				
•••								
51								(LSB)
52	(MSB)			WEIGHTED NUMBER OF READ COMMANDS PLUS WRITE COMMANDS				
•••								
59								(LSB)
60	(MSB)			WEIGHTED READ COMMAND PROCESSING PLUS WRITE COMMAND PROCESSING				
•••								
67								(LSB)

The PARAMETER CODE field is described in 7.3.2.2.1, and shall be set as shown in table 379 for the General Access Statistics and Performance log parameter.

The DU bit, TSD bit, ETC bit, TMC field, and FORMAT AND LINKING field are described in 7.3.2.2.2.1. These fields shall be set as described for an unbounded data counter log parameter (see 7.3.2.2.2.3) for the General Access Statistics and Performance log parameter.

The PARAMETER LENGTH field is described in 7.3.2.2.1, and shall be set as shown in table 379 for the General Access Statistics and Performance log parameter.

The NUMBER OF READ COMMANDS field indicates the number of read commands (see table 377 in 7.3.7.1) received by the logical unit.

The NUMBER OF WRITE COMMANDS field indicates the number of write commands (see table 377 in 7.3.7.1) received by the logical unit.

The NUMBER OF LOGICAL BLOCKS RECEIVED field indicates the number of logical blocks received by any SCSI target port for the logical unit as a result of write commands (see table 377 in 7.3.7.1).

The NUMBER OF LOGICAL BLOCKS TRANSMITTED field indicates the number of logical blocks transmitted by any SCSI target port for the logical unit as a result of read commands (see table 377 in 7.3.7.1).

The READ COMMAND PROCESSING INTERVALS field indicates the cumulative number of time intervals (see 7.3.6.7) spent by the logical unit processing read commands (see table 377 in 7.3.7.1).

The WRITE COMMAND PROCESSING INTERVALS field indicates the cumulative number of time intervals (see 7.3.6.7) spent by the logical unit processing write commands (see table 377 in 7.3.7.1).

If command priority is supported (see SAM-5), then the WEIGHTED NUMBER OF READ COMMANDS PLUS WRITE COMMANDS field indicates the cumulative command weight of the read commands and write commands (see table 377 in 7.3.7.1) processed by the logical unit.

Command weight is calculated as follows:

command weight = (360 360 / command priority)

where:

command priority is as defined in SAM-5. However, if the computed command priority is zero, then the command priority shall be set to seven (i.e., a mid-range command priority value).

If command priority is not supported, then the WEIGHTED NUMBER OF READ COMMANDS PLUS WRITE COMMANDS field shall be set to zero.

If command priority is supported (see SAM-5), then the WEIGHTED READ COMMAND PROCESSING PLUS WRITE COMMAND PROCESSING field indicates the cumulative weighted command time of the time intervals (see 7.3.6.7) spent processing read commands and write commands (see table 377 in 7.3.7.1) by the logical unit.

Weighted command time is calculated as follows:

weighted command time = (time increments processing the command × time interval)
× (360 360 / command priority)

where:

time increments processing a command is the number of time intervals from the time the task manager places the command into a task set until the device server sends a SCSI transport protocol service response for the command;

time interval is the value represented in the TIME INTERVAL DESCRIPTOR field of the Time Interval log parameter (see 7.3.6.7), and

command priority is as defined in SAM-5. However, if the computed command priority is zero, then the command priority time shall be set to seven (i.e., a mid-range command priority value).

If command priority is not supported, then the WEIGHTED READ COMMAND PROCESSING PLUS WRITE COMMAND PROCESSING field shall be set to zero.

7.3.7.3 Idle Time log parameter

The Idle Time log parameter has the format shown in table 380.

Table 380 — Idle Time log parameter

Bit / Byte	7	6	5	4	3	2	1	0
0	(MSB)							
1	PARAMETER CODE (0002h)							(LSB)
2	Parameter control byte – unbounded data counter log parameter (see 7.3.2.2.2.3)							
	DU	Obsolete	TSD	ETC	TMC		FORMAT AND LINKING	
3	PARAMETER LENGTH (08h)							
4	(MSB)							
•••	IDLE TIME INTERVALS							
11								(LSB)

The PARAMETER CODE field is described in 7.3.2.2.1, and shall be set as shown in table 380 for the Idle Time log parameter.

The DU bit, TSD bit, ETC bit, TMC field, and FORMAT AND LINKING field are described in 7.3.2.2.2.1. These fields shall be set as described for an unbounded data counter log parameter (see 7.3.2.2.2.3) for the Idle Time log parameter.

The PARAMETER LENGTH field is described in 7.3.2.2.1, and shall be set as shown in table 380 for the Idle Time log parameter.

The IDLE TIME INTERVALS field indicates the cumulative number of idle times spent while there are no commands in the task set and there are no commands being processed by the logical unit.

Idle time is calculated as follows:

idle time = (time increments not processing commands × time interval)

where:

time increments not processing commands is the number of time intervals while there are no commands in the task set and the device server has sent a SCSI transport protocol service response for all commands being processed (i.e., there are no commands to be processed or being processed); and

time interval is the value represented in the time interval descriptor of the Time Interval log parameter (see 7.3.6.7).

7.3.7.4 Force Unit Access Statistics and Performance log parameter

The Force Unit Access Statistics and Performance log parameter has the format shown in table 381.

Table 381 — Force Unit Access Statistics and Performance log parameter

Bit / Byte	7	6	5	4	3	2	1	0
0	(MSB)							
1	PARAMETER CODE (0004h)							(LSB)
2	Parameter control byte – unbounded data counter log parameter (see 7.3.2.2.2.3)							
	DU	Obsolete	TSD	ETC	TMC		FORMAT AND LINKING	
3	PARAMETER LENGTH (40h)							
4	(MSB)							
•••	NUMBER OF READ FUA COMMANDS							
11								(LSB)
12	(MSB)							
•••	NUMBER OF WRITE FUA COMMANDS							
19								(LSB)
20	(MSB)							
•••	NUMBER OF READ FUA_NV COMMANDS							
27								(LSB)
28	(MSB)							
•••	NUMBER OF WRITE FUA_NV COMMANDS							
35								(LSB)
36	(MSB)							
•••	READ FUA COMMAND PROCESSING INTERVALS							
43								(LSB)
44	(MSB)							
•••	WRITE FUA COMMAND PROCESSING INTERVALS							
51								(LSB)
52	(MSB)							
•••	READ FUA_NV COMMAND PROCESSING INTERVALS							
59								(LSB)
60	(MSB)							
•••	WRITE FUA_NV COMMAND PROCESSING INTERVALS							
67								(LSB)

The PARAMETER CODE field is described in 7.3.2.2.1, and shall be set as shown in table 381 for the Force Unit Access Statistics and Performance log parameter.

The DU bit, TSD bit, ETC bit, TMC field, and FORMAT AND LINKING field are described in 7.3.2.2.2.1. These fields shall be set as described for an unbounded data counter log parameter (see 7.3.2.2.2.3) for the Force Unit Access Statistics and Performance log parameter.

The PARAMETER LENGTH field is described in 7.3.2.2.1, and shall be set as shown in table 381 for the Force Unit Access Statistics and Performance log parameter.

The NUMBER OF READ FUA COMMANDS field indicates the number of read commands (see table 377 in 7.3.7.1) with the FUA bit (see SBC-3) set to one received by the logical unit.

The NUMBER OF WRITE FUA COMMANDS field indicates the number of write commands (see table 377 in 7.3.7.1) with the FUA bit (see SBC-3) set to one received by the logical unit.

The NUMBER OF READ FUA_NV COMMANDS field indicates the number of read commands (see table 377 in 7.3.7.1) with the FUA_NV bit (see SBC-2) set to one received by the logical unit.

The NUMBER OF WRITE FUA_NV COMMANDS field indicates the number of write commands (see table 377 in 7.3.7.1) with the FUA_NV bit (see SBC-2) set to one received by the logical unit.

The READ FUA COMMAND PROCESSING INTERVALS field indicates the cumulative number of time intervals (see 7.3.6.7) spent by the logical unit processing read commands (see table 377 in 7.3.7.1) with the FUA bit (see SBC-3) set to one.

The WRITE FUA COMMAND PROCESSING INTERVALS field indicates the cumulative number of time intervals (see 7.3.6.7) spent by the logical unit processing write commands (see table 377 in 7.3.7.1) with the FUA bit (see SBC-3) set to one.

The READ FUA_NV COMMAND PROCESSING INTERVALS field indicates the cumulative number of time intervals (see 7.3.6.7) spent by the logical unit processing read commands (see table 377 in 7.3.7.1) with the FUA_NV bit (see SBC-2) set to one.

The WRITE FUA_NV COMMAND PROCESSING INTERVALS field indicates the cumulative number of time intervals (see 7.3.6.7) spent by the logical unit processing write commands (see table 377 in 7.3.7.1) with the FUA_NV bit (see SBC-2) set to one.

7.3.8 Group Statistics and Performance (n) log pages

7.3.8.1 Overview

Using the format shown in table 383, Group Statistics and Performance (n) log pages collect statistics and performance information for the group number specified in a read CDB or a write CDB based on the parameter codes listed in table 382.

Table 382 — Group Statistics and Performance log page parameter codes

Parameter code	Description	Resettable or Changeable [a]	Reference	Support
0001h	Group n Statistics and Performance	Reset Only	7.3.8.2	Mandatory
0004h	Group n Force Unit Access Statistics and Performance	Reset Only	7.3.8.3	Optional
all others	Reserved			

[a] The keywords in this column – Always, Reset Only, and Never – are defined in 7.3.3.

The Group Statistics and Performance (n) log pages provide the following statistics and performance results associated to the addressed logical unit and the GROUP NUMBER field:

a) Statistics and Performance log parameters:
   A) number of read commands;
   B) number of write commands;
   C) number of read logical blocks transferred by a target port;
   D) number of write logical blocks transferred by a target port;
   E) read command processing time;
   F) write command processing time;
   and

b) Force Unit Access Statistics and Performance log parameters:
   A) number of read commands with the FUA bit (see SBC-3) set to one;
   B) number of write commands with the FUA bit set to one;
   C) number of read commands with the FUA_NV bit (see SBC-2) set to one;
   D) number of write commands with the FUA_NV bit set to one;
   E) read command with the FUA bit set to one processing intervals;
   F) write command with the FUA bit set to one processing intervals;
   G) read command with the FUA_NV bit set to one processing intervals; and
   H) write command with the FUA_NV bit set to one processing intervals.

In the Group Statistics and Performance (n) log pages, read commands and write commands are those shown in table 377 (see 7.3.7.1).

The Group Statistics and Performance (n) log pages provide logging of statistics and performance of read and write operations based on group numbers. There are 31 Group Statistics and Performance (n) log pages one for each group number. The statistics and performance information associated with each group number is collected in the corresponding Group Statistics and Performance (n) log page (e.g., operations associated with group number 16 are logged in the Group Statistics and Performance (16) log page).

Each Group Statistics and Performance (n) log page has the format shown in table 383.

Table 383 — Group Statistics and Performance (n) log page

Bit / Byte	7	6	5	4	3	2	1	0
0	DS	SPF (1b)	PAGE CODE (19h)					
1	SUBPAGE CODE (01h to 1Fh) (see table 384) [a]							
2	(MSB)	PAGE LENGTH (n-3)						
3								(LSB)
	Group statistics and performance log parameters							
4 •••	General statistics and performance log parameter (see table 382) [first]							
	⋮							
••• n	General statistics and performance log parameter (see table 382) [last]							

[a] The log parameter associated with the specific group number as specified by the value of n is collected in the corresponding log parameter (e.g., the count of read commands with the GROUP NUMBER field set to 9 is logged in the GROUP N NUMBER OF READ COMMANDS field in the Group n Statistics and Performance log parameter of the Group Statistics and Performance (9) log page).

The DS bit, SPF bit, PAGE CODE field, SUBPAGE CODE field, and PAGE LENGTH field are described in 7.3.2. The SPF bit, PAGE CODE field, and SUBPAGE CODE field shall be set as shown in table 383 for the Group Statistics and Performance (n) log pages.

The SUBPAGE CODE field (see table 384) associates the Group Statistics and Performance (n) log page being transferred with the contents of the GROUP NUMBER field in a read command CDB or write command CDB (see table 377 in 7.3.7.1 and SBC-3).

Table 384 — Group Statistics and Performance (n) subpage codes

Subpage code	Log page name [a]	Group number [b]
01h	Group Statistics and Performance (1)	00001b
02h	Group Statistics and Performance (2)	00010b
03h	Group Statistics and Performance (3)	00011b
04h	Group Statistics and Performance (4)	00100b
05h	Group Statistics and Performance (5)	00101b
06h	Group Statistics and Performance (6)	00110b
07h	Group Statistics and Performance (7)	00111b
08h	Group Statistics and Performance (8)	01000b
09h	Group Statistics and Performance (9)	01001b
0Ah	Group Statistics and Performance (10)	01010b
0Bh	Group Statistics and Performance (11)	01011b
0Ch	Group Statistics and Performance (12)	01100b
0Dh	Group Statistics and Performance (13)	01101b
0Eh	Group Statistics and Performance (14)	01110b
0Fh	Group Statistics and Performance (15)	01111b
10h	Group Statistics and Performance (16)	10000b
11h	Group Statistics and Performance (17)	10001b
12h	Group Statistics and Performance (18)	10010b
13h	Group Statistics and Performance (19)	10011b
14h	Group Statistics and Performance (20)	10100b
15h	Group Statistics and Performance (21)	10101b
16h	Group Statistics and Performance (22)	10110b
17h	Group Statistics and Performance (23)	10111b
18h	Group Statistics and Performance (24)	11000b
19h	Group Statistics and Performance (25)	11001b
1Ah	Group Statistics and Performance (26)	11010b
1Bh	Group Statistics and Performance (27)	11011b
1Ch	Group Statistics and Performance (28)	11100b
1Dh	Group Statistics and Performance (29)	11101b
1Eh	Group Statistics and Performance (30)	11110b
1Fh	Group Statistics and Performance (31)	11111b

[a] The statistics and performance information associated with a group number is collected in the corresponding Group Statistics and Performance (n) log page (e.g., operations associated with group number 10000b are logged in the Group Statistics and Performance (16) log page).

[b] The GROUP NUMBER field is from the read command CDB or the write command CDB (see table 377 in 7.3.7.1 and SBC-3).

The contents of each group statistics and performance log parameter depends on the value in its PARAMETER CODE field (see table 382).

7.3.8.2 Group n Statistics and Performance log parameter

The Group n Statistics and Performance log parameter has the format shown in table 385.

Table 385 — Group n Statistics and Performance log parameter

Bit / Byte	7	6	5	4	3	2	1	0
0	(MSB)			PARAMETER CODE (0001h)				
1								(LSB)
2	Parameter control byte – unbounded data counter log parameter (see 7.3.2.2.2.3)							
	DU	Obsolete	TSD	ETC	TMC		FORMAT AND LINKING	
3	PARAMETER LENGTH (30h)							
4	(MSB)			GROUP N NUMBER OF READ COMMANDS				
•••								
11								(LSB)
12	(MSB)			GROUP N NUMBER OF WRITE COMMANDS				
•••								
19								(LSB)
20	(MSB)			GROUP N NUMBER OF LOGICAL BLOCKS RECEIVED				
•••								
27								(LSB)
28	(MSB)			GROUP N NUMBER OF LOGICAL BLOCKS TRANSMITTED				
•••								
35								(LSB)
36	(MSB)			GROUP N READ COMMAND PROCESSING INTERVALS				
•••								
43								(LSB)
44	(MSB)			GROUP N WRITE COMMAND PROCESSING INTERVALS				
•••								
51								(LSB)

The PARAMETER CODE field is described in 7.3.2.2.1, and shall be set as shown in table 385 for the Group n Statistics and Performance log parameter.

The DU bit, TSD bit, ETC bit, TMC field, and FORMAT AND LINKING field are described in 7.3.2.2.2.1. These fields shall be set as described for an unbounded data counter log parameter (see 7.3.2.2.2.3) for the Group n Statistics and Performance log parameter.

The PARAMETER LENGTH field is described in 7.3.2.2.1, and shall be set as shown in table 385 for the Group n Statistics and Performance log parameter.

The GROUP N NUMBER OF READ COMMANDS field indicates the number of read commands (see table 377 in 7.3.7.1) received by the logical unit.

The GROUP N NUMBER OF WRITE COMMANDS field indicates the number of write commands (see table 377 in 7.3.7.1) received by the logical unit.

The GROUP N NUMBER OF LOGICAL BLOCKS RECEIVED field indicates the number of logical blocks received by any SCSI target port for the logical unit as a result of write commands (see table 377 in 7.3.7.1).

The GROUP N NUMBER OF LOGICAL BLOCKS TRANSMITTED field indicates the number of logical blocks transmitted by any SCSI target port for the logical unit as a result of read commands (see table 377 in 7.3.7.1).

The GROUP N READ COMMAND PROCESSING INTERVALS field indicates the cumulative number of time intervals spent by the logical unit processing read commands (see table 377 in 7.3.7.1). Time intervals are defined in the Time Interval log parameter (see 7.3.6.7) as returned in the General Statistics and Performance log page (see 7.3.7).

The GROUP N WRITE COMMAND PROCESSING INTERVALS field indicates the cumulative number of time intervals spent by the logical unit processing write commands (see table 377 in 7.3.7.1). Time intervals are defined in the Time Interval log parameter (see 7.3.6.7) as returned in the General Statistics and Performance log page (see 7.3.7).

7.3.8.3 Group n Force Unit Access Statistics and Performance log parameter

The Group n Force Unit Access Statistics and Performance log parameter has the format shown in table 386.

Table 386 — Group n Force Unit Access Statistics and Performance log parameter

Bit / Byte	7	6	5	4	3	2	1	0
0	(MSB)							
1				PARAMETER CODE (0004h)				(LSB)
2	Parameter control byte – unbounded data counter log parameter (see 7.3.2.2.3)							
	DU	Obsolete	TSD	ETC	TMC		FORMAT AND LINKING	
3	PARAMETER LENGTH (40h)							
4	(MSB)							
•••				GROUP N NUMBER OF READ FUA COMMANDS				
11								(LSB)
12	(MSB)							
•••				GROUP N NUMBER OF WRITE FUA COMMANDS				
19								(LSB)
20	(MSB)							
•••				GROUP N NUMBER OF READ FUA_NV COMMANDS				
27								(LSB)
28	(MSB)							
•••				GROUP N NUMBER OF WRITE FUA_NV COMMANDS				
35								(LSB)
36	(MSB)							
•••				GROUP N READ FUA COMMAND PROCESSING INTERVALS				
43								(LSB)
44	(MSB)							
•••				GROUP N WRITE FUA COMMAND PROCESSING INTERVALS				
51								(LSB)
52	(MSB)							
•••				GROUP N READ FUA_NV COMMAND PROCESSING INTERVALS				
59								(LSB)
60	(MSB)							
•••				GROUP N WRITE FUA_NV COMMAND PROCESSING INTERVALS				
67								(LSB)

The PARAMETER CODE field is described in 7.3.2.2.1, and shall be set as shown in table 386 for the Group n Force Unit Access Statistics and Performance log parameter.

The DU bit, TSD bit, ETC bit, TMC field, and FORMAT AND LINKING field are described in 7.3.2.2.2.1. These fields shall be set as described for an unbounded data counter log parameter (see 7.3.2.2.2.3) for the Group n Force Unit Access Statistics and Performance log parameter.

The PARAMETER LENGTH field is described in 7.3.2.2.1, and shall be set as shown in table 386 for the Group n Force Unit Access Statistics and Performance log parameter.

The GROUP N NUMBER OF READ FUA COMMANDS field indicates the number of read commands (see table 377 in 7.3.7.1) with the FUA bit (see SBC-3) set to one received by the logical unit.

The GROUP N NUMBER OF WRITE FUA COMMANDS field indicates the number of write commands (see table 377 in 7.3.7.1) with the FUA bit (see SBC-3) set to one received by the logical unit.

The GROUP N NUMBER OF READ FUA_NV COMMANDS field indicates the number of read commands (see table 377 in 7.3.7.1) with the FUA_NV bit (see SBC-2) set to one received by the logical unit.

The GROUP N NUMBER OF WRITE FUA_NV COMMANDS field indicates the number of write commands (see table 377 in 7.3.7.1) with the FUA_NV bit (see SBC-2) set to one received by the logical unit.

The GROUP N READ FUA COMMAND PROCESSING INTERVALS field indicates the cumulative number of time intervals spent by the logical unit processing read commands (see table 377 in 7.3.7.1) with the FUA bit (see SBC-3) set to one. Time intervals are defined in the Time Interval log parameter (see 7.3.6.7) as returned in the General Statistics and Performance log page (see 7.3.7).

The GROUP N WRITE FUA COMMAND PROCESSING INTERVALS field indicates the cumulative number of time intervals spent by the logical unit processing write commands (see table 377 in 7.3.7.1) with the FUA bit (see SBC-3) set to one. Time intervals are defined in the Time Interval log parameter (see 7.3.6.7) as returned in the General Statistics and Performance log page (see 7.3.7).

The GROUP N READ FUA_NV COMMAND PROCESSING INTERVALS field indicates the cumulative number of time intervals spent by the logical unit processing read commands (see table 377 in 7.3.7.1) with the FUA_NV bit (see SBC-2) set to one. Time intervals are defined in the Time Interval log parameter (see 7.3.6.7) as returned in the General Statistics and Performance log page (see 7.3.7).

The GROUP N WRITE FUA_NV COMMAND PROCESSING INTERVALS field indicates the cumulative number of time intervals spent by the logical unit processing write commands (see table 377 in 7.3.7.1) with the FUA_NV bit (see SBC-2) set to one. Time intervals are defined in the Time Interval log parameter (see 7.3.6.7) as returned in the General Statistics and Performance log page (see 7.3.7).

7.3.9 Informational Exceptions log page

7.3.9.1 Overview

Using the format shown in table 388, the Informational Exceptions log page returns details about informational exceptions identified by the parameter codes listed in table 387.

Table 387 — Informational Exceptions log page parameter codes

Parameter code	Description	Resettable or Changeable [a]	Reference	Support
0000h	Informational Exceptions General	Reset Only	7.3.9.2	Mandatory
0001h to FFFFh	Vendor specific			

[a] The keywords in this column – Always, Reset Only, and Never – are defined in 7.3.3.

The Informational Exceptions log page has the format shown in table 388.

Table 388 — Informational Exceptions log page

Bit / Byte	7	6	5	4	3	2	1	0
0	DS	SPF (0b)	PAGE CODE (2Fh)					
1	SUBPAGE CODE (00h)							
2	(MSB)	PAGE LENGTH (n-3)						
3								(LSB)
	Informational exceptions log parameters							
4	Informational exceptions log parameter (see table 387) [first]							
•••								
	⋮							
•••	Informational exceptions log parameter (see table 387) [last]							
n								

The DS bit, SPF bit, PAGE CODE field, SUBPAGE CODE field, and PAGE LENGTH field are described in 7.3.2. The SPF bit, PAGE CODE field, and SUBPAGE CODE field shall be set as shown in table 388 for the Informational Exceptions log page.

The contents of each informational exceptions log parameter depends on the value in its PARAMETER CODE field (see table 387).

7.3.9.2 Informational Exceptions General log parameter

The Informational Exceptions General log parameter has the format shown in table 389.

Table 389 — Informational Exceptions General log parameter

Bit / Byte	7	6	5	4	3	2	1	0
0	(MSB)			PARAMETER CODE (0000h)				
1								(LSB)
2	Parameter control byte – binary format list log parameter (see 7.3.2.2.2.5)							
	DU	Obsolete	TSD	ETC	TMC		FORMAT AND LINKING	
3	PARAMETER LENGTH (n-3)							
4	INFORMATIONAL EXCEPTION ADDITIONAL SENSE CODE							
5	INFORMATIONAL EXCEPTION ADDITIONAL SENSE CODE QUALIFIER							
6	MOST RECENT TEMPERATURE READING							
7	Vendor specific							
•••								
n								

The PARAMETER CODE field is described in 7.3.2.2.1, and shall be set as shown in table 389 for the Informational Exceptions General log parameter.

The DU bit, TSD bit, ETC bit, TMC field, and FORMAT AND LINKING field are described in 7.3.2.2.2.1. These fields shall be set as described for a binary format list log parameter (see 7.3.2.2.2.5) for the Informational Exceptions General log parameter.

The PARAMETER LENGTH field is described in 7.3.2.2.1. The parameter length shall be at least 04h.

If the INFORMATIONAL EXCEPTION ADDITIONAL SENSE CODE field is set to zero, no informational exception condition exists and contents of the INFORMATIONAL EXCEPTION ADDITIONAL SENSE CODE QUALIFIER field are unspecified. If the INFORMATIONAL EXCEPTION ADDITIONAL SENSE CODE field is set to a value other than zero, an informational exception condition exists that has an additional sense code indicated by INFORMATIONAL EXCEPTION ADDITIONAL SENSE CODE field and an ADDITIONAL SENSE CODE QUALIFIER field indicated by the INFORMATIONAL EXCEPTION ADDITIONAL SENSE CODE QUALIFIER field.

The MOST RECENT TEMPERATURE READING field indicates the temperature in degrees Celsius of the SCSI target device at the time the LOG SENSE command is performed. 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 as a result of a sensor failure or other condition, then 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.

7.3.10 Last *n* Deferred Errors or Asynchronous Events log page

7.3.10.1 Overview

Using the format shown in table 391, the Last *n* Deferred Errors or Asynchronous Events log page provides one or more Deferred Error or Asynchronous Event log parameters (see 7.3.10.2). The number of Deferred Error or Asynchronous Event log parameters supported (i.e., *n*) is vendor specific. The parameter codes for the Last *n* Deferred Errors or Asynchronous Events log page are listed in table 390.

Table 390 — Last *n* Deferred Errors or Asynchronous Events log page parameter codes

Parameter code	Description	Resettable or Changeable [a]	Reference	Support
0000h	Deferred Error or Asynchronous Event	Reset Only	7.3.10.2	Mandatory
0001h to FFFFh				Optional

[a] The keywords in this column – Always, Reset Only, and Never – are defined in 7.3.3.

The Last *n* Deferred Errors or Asynchronous Events log page has the format shown in table 391.

Table 391 — Last *n* Deferred Errors or Asynchronous Events log page

Bit / Byte	7	6	5	4	3	2	1	0
0	DS	SPF (0b)	PAGE CODE (0Bh)					
1	SUBPAGE CODE (00h)							
2	(MSB)	PAGE LENGTH (n-3)						
3								(LSB)
	Deferred error or asynchronous event log parameters							
4 •••	Deferred error or asynchronous event log parameter (see 7.3.10.2) [first]							
	⋮							
••• n	Deferred error or asynchronous event log parameter (see 7.3.10.2) [last]							

The DS bit, SPF bit, PAGE CODE field, SUBPAGE CODE field, and PAGE LENGTH field are described in 7.3.2. The SPF bit, PAGE CODE field, and SUBPAGE CODE field shall be set as shown in table 391 for the Last *n* Deferred Errors or Asynchronous Events log page.

The contents of each deferred error or asynchronous event log parameter are described in 7.3.10.2. The device server shall assign parameter codes to log parameters as follows:

a) if the vendor specific number of supported deferred error or asynchronous event log parameters has not been exceeded, then the parameter code in each log parameter shall indicate 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 at a more recent time; or
b) if the vendor specific number of supported deferred error or asynchronous event log parameters has been exceeded, then:

A) the log parameter with the oldest data shall be overwritten with the newest data (i.e., after the highest supported parameter code is used, reporting wraps so that the next deferred error or asynchronous event is reported in the log parameter with parameter code zero); and
B) if the RLEC bit is set to one in the Control mode page (see 7.5.8) and a LOG SELECT command that transfers the Last *n* Deferred Errors or Asynchronous Events log page completes without error, except for the parameter code being at its maximum value, then the device server shall terminate the command with CHECK CONDITION status, with the sense key set to RECOVERED ERROR, and the additional sense code set to LOG LIST CODES EXHAUSTED.

7.3.10.2 Deferred Error or Asynchronous Event log parameters

Each Deferred Error or Asynchronous Event log parameter has the format shown in table 392.

Table 392 — Deferred Error or Asynchronous Event log parameter

Bit / Byte	7	6	5	4	3	2	1	0
0	(MSB)							
1	PARAMETER CODE (see table 390)							(LSB)
2	Parameter control byte – binary format list log parameter (see 7.3.2.2.2.5)							
	DU	Obsolete	TSD	ETC	TMC		FORMAT AND LINKING	
3	PARAMETER LENGTH (n-3)							
4	SENSE DATA (see 4.5)							
•••								
n								

The PARAMETER CODE field is described in 7.3.2.2.1, and shall be set as shown in table 390 for the Deferred Error or Asynchronous Event log parameter.

The DU bit, TSD bit, ETC bit, TMC field, and FORMAT AND LINKING field are described in 7.3.2.2.2.1. These fields shall be set as described for a binary format list log parameter (see 7.3.2.2.2.5) for the Deferred Error or Asynchronous Event log parameter.

The PARAMETER LENGTH field is described in 7.3.2.2.1.

The SENSE DATA field contains the sense data (see 4.5) for a deferred error or asynchronous event that has occurred.

7.3.11 Last *n* Error Events log page

7.3.11.1 Overview

Using the format shown in table 394, the Last *n* Error Events log page provides one or more Error Event log parameters (see 7.3.11.2). The number of these Error Event log parameters supported (i.e., *n*) is vendor specific. The parameter codes for the Last *n* Error Events log page are listed in table 393.

Table 393 — Last *n* Error Events log page parameter codes

Parameter code	Description	Resettable or Changeable [a]	Reference	Support
0000h	Error Event	Reset Only	7.3.11.2	Mandatory
0001h to FFFFh				Optional

[a] The keywords in this column – Always, Reset Only, and Never – are defined in 7.3.3.

The Last *n* Error Events log page has the format shown in table 394.

Table 394 — Last *n* Error Events log page

Bit / Byte	7	6	5	4	3	2	1	0
0	DS	SPF (0b)	PAGE CODE (07h)					
1	SUBPAGE CODE (00h)							
2	(MSB)	PAGE LENGTH (n-3)						
3								(LSB)
	Error event log parameters							
4	Error event log parameter (see 7.3.11.2) [first]							
•••								
	⋮							
•••	Error event log parameter (see 7.3.11.2) [last]							
n								

The DS bit, SPF bit, PAGE CODE field, SUBPAGE CODE field, and PAGE LENGTH field are described in 7.3.2. The SPF bit, PAGE CODE field, and SUBPAGE CODE field shall be set as shown in table 394 for the Last *n* Error Events log page.

The contents of each error event log parameter is described are 7.3.11.2. The device server shall assign parameter codes to log parameters as follows:

a) if the vendor specific number of supported error event log parameters has not been exceeded, then the parameter code in each log parameter shall indicate the relative time at which the error event occurred. A higher parameter code indicates that the error event occurred later in time; or
b) if the vendor specific number of supported error event log parameters has been exceeded, then:
   A) the log parameter with the oldest data shall be overwritten with the newest data (i.e., after the highest supported parameter code is used, reporting wraps so that the next error event is reported in the log parameter with parameter code zero); and
   B) if the RLEC bit is set to one in the Control mode page (see 7.5.8) and a LOG SELECT command that transfers the Last *n* Error Events log page completes without error, except for the parameter

code being at its maximum value, then the device server shall terminate the command with CHECK CONDITION status, with the sense key set to RECOVERED ERROR, and the additional sense code set to LOG LIST CODES EXHAUSTED.

7.3.11.2 Error Event log parameters

Each Error Event log parameter has the format shown in table 395.

Table 395 — Error Event log parameter

Bit / Byte	7	6	5	4	3	2	1	0
0	(MSB)							
1	PARAMETER CODE (see table 393)							(LSB)
2	Parameter control byte – ASCII format list log parameter (see 7.3.2.2.2.4)							
	DU	Obsolete	TSD	ETC	TMC		FORMAT AND LINKING	
3	PARAMETER LENGTH (n-3)							
4	ERROR EVENT DATA							
•••								
n								

The PARAMETER CODE field is described in 7.3.2.2.1, and shall be set as shown in table 393 for the Error Event log parameter.

The DU bit, TSD bit, ETC bit, TMC field, and FORMAT AND LINKING field are described in 7.3.2.2.2.1. These fields shall be set as described for a ASCII format list log parameter (see 7.3.2.2.2.4) for the Error Event log parameter.

The PARAMETER LENGTH field is described in 7.3.2.2.1.

The ERROR EVENT DATA field contains ASCII data (see 4.3.1) that may describe the error event. The contents of the ERROR EVENT DATA field are not defined by this standard.

7.3.12 Non-Medium Error log page

7.3.12.1 Overview

Using the format shown in table 397, the Non-Medium Error log page provides for counting the occurrences of recoverable error events other than read (see 7.3.15), read reverse (see 7.3.16), verify (see 7.3.23), or write (see 7.3.24) failures. No discrimination among the various types of events is provided. Vendor specific discrimination may be provided through the vendor specific parameter codes. The parameter codes for the Non-Medium Error log page are listed in table 396.

Table 396 — Non-Medium Error log page parameter codes

Parameter code	Description	Resettable or Changeable [a]	Reference	Support
0000h	Non-Medium Error Count	Reset Only	7.3.12.2	Mandatory
8000h to FFFFh	Vendor specific error counts			
all others	Reserved			

[a] The keywords in this column – Always, Reset Only, and Never – are defined in 7.3.3.

The Non-Medium Error log page has the format shown in table 397.

Table 397 — Non-Medium Error log page

Bit / Byte	7	6	5	4	3	2	1	0
0	DS	SPF (0b)	PAGE CODE (06h)					
1	SUBPAGE CODE (00h)							
2	(MSB)	PAGE LENGTH (n-3)						
3								(LSB)
	Non-medium error log parameters							
4	Non-medium error log parameter (see table 396) [first]							
•••								
	⋮							
	Non-medium error log parameter (see table 396) [last]							
•••								
n								

The DS bit, SPF bit, PAGE CODE field, SUBPAGE CODE field, and PAGE LENGTH field are described in 7.3.2. The SPF bit, PAGE CODE field, and SUBPAGE CODE field shall be set as shown in table 397 for the Non-Medium Error log page.

The contents of each non-medium error log parameter depends on the value in its PARAMETER CODE field (see table 396).

7.3.12.2 Non-Medium Error Count log parameter

The Non-Medium Error Count log parameter has the format shown in table 398.

Table 398 — Non-Medium Error Count log parameter

Bit / Byte	7	6	5	4	3	2	1	0
0	(MSB)	PARAMETER CODE (0000h)						
1								(LSB)
2	Parameter control byte – bounded data counter log parameter (see 7.3.2.2.2.2)							
	DU	Obsolete	TSD	ETC	TMC		FORMAT AND LINKING	
3	PARAMETER LENGTH (n-3)							
4	(MSB)	NON-MEDIUM ERROR COUNT						
•••								
n								(LSB)

The PARAMETER CODE field is described in 7.3.2.2.1, and shall be set as shown in table 398 for the Non-Medium Error Count log parameter.

The DU bit, TSD bit, ETC bit, TMC field, and FORMAT AND LINKING field are described in 7.3.2.2.2.1. These fields shall be set as described for a bounded data counter log parameter (see 7.3.2.2.2.2) for the Non-Medium Error Count log parameter.

The PARAMETER LENGTH field is described in 7.3.2.2.1.

The NON-MEDIUM ERROR COUNT field indicates the number of recoverable error events other than read, read reverse, verify, or write failures.

7.3.13 Power Condition Transitions log page

7.3.13.1 Overview

Using the format shown in table 400, the Power Condition Transitions log page provides a count of the occurrences of power condition transition events using the parameter codes listed in table 399.

Table 399 — Power Conditions Transitions log page parameter codes

Parameter code	Description	Resettable or Changeable [a]	Reference	Support
0001h	Accumulated Transitions to active	Never	7.3.13.2	Mandatory
0002h	Accumulated Transitions to idle_a			At least one [b]
0003h	Accumulated Transitions to idle_b			
0004h	Accumulated Transitions to idle_c			
0008h	Accumulated Transitions to standby_z			
0009h	Accumulated Transitions to standby_y			
all others	Reserved			

[a] The keywords in this column – Always, Reset Only, and Never – are defined in 7.3.3.

[b] If the Power Conditions Transitions log page is supported, at least one of these parameter codes shall be supported.

The Power Conditions Transitions log page has the format shown in table 400.

Table 400 — Power Condition Transitions log page

Bit / Byte	7	6	5	4	3	2	1	0
0	DS	SPF (0b)	PAGE CODE (1Ah)					
1	SUBPAGE CODE (00h)							
2	(MSB)	PAGE LENGTH (n-3)						
3								(LSB)
	Power condition transitions log parameters							
4 •••	Power condition transitions log parameter (see table 399) [first]							
	⋮							
••• n	Power condition transitions log parameter (see table 399) [last]							

The DS bit, SPF bit, PAGE CODE field, SUBPAGE CODE field, and PAGE LENGTH field are described in 7.3.2. The SPF bit, PAGE CODE field, and SUBPAGE CODE field shall be set as shown in table 400 for the Power Conditions Transitions log page.

The contents of each power condition transitions log parameter depends on the value in its PARAMETER CODE field (see table 399).

7.3.13.2 Accumulated Transitions log parameter

The Accumulated Transitions log parameter has the format shown in table 401.

Table 401 — Accumulated Transitions log parameter

Bit / Byte	7	6	5	4	3	2	1	0
0	(MSB)							
1				PARAMETER CODE (see table 399)				(LSB)
2	Parameter control byte – binary format list log parameter (see 7.3.2.2.2.5)							
	DU	Obsolete	TSD	ETC	TMC		FORMAT AND LINKING	
3	PARAMETER LENGTH (04h)							
4	(MSB)							
•••				ACCUMULATED TRANSITIONS TO				
7								(LSB)

The PARAMETER CODE field is described in 7.3.2.2.1, and shall be set as shown in table 399 for the Accumulated Transitions log parameter.

The DU bit, TSD bit, ETC bit, TMC field, and FORMAT AND LINKING field are described in 7.3.2.2.2.1. These fields shall be set as described for a binary format list log parameter (see 7.3.2.2.2.5) for the Accumulated Transitions log parameter.

The PARAMETER LENGTH field is described in 7.3.2.2.1, and shall be set as shown in table 401 for the Accumulated Transitions log parameter.

The ACCUMULATED TRANSITIONS TO field contains a saturating counter that is incremented by one at a time defined by the contents of the PARAMETER CODE field as described in table 402. The time in the transition at which the count is incremented is vendor specific.

Table 402 — Accumulated Transitions parameter codes and saturating counters

Parameter code	Saturating counter incremented while the device server transitions from any other power condition to this power condition
0001h	active power condition (see 5.11.4)
0002h	idle_a power condition (see 5.11.5)
0003h	idle_b power condition (see 5.11.5)
0004h	idle_c power condition (see 5.11.5)
0008h	standby_z power condition (see 5.11.6)
0009h	standby_y power condition (see 5.11.6)

All values in the ACCUMULATED TRANSITIONS TO field are accumulated from the time the SCSI target device is manufactured.

7.3.14 Protocol Specific Port log page

7.3.14.1 Overview

Using the format shown in table 403, the Protocol Specific Port log page provides SCSI transport protocol specific parameters that are associated with the SCSI target ports in the SCSI target device. This log page may be implemented in any logical unit, including the TARGET LOG PAGES well-known logical unit (see 8.4).

Protocol Specific Port log pages do not identify the information being logged using the PARAMETER CODE field in each log parameter. Instead, the SUBPAGE CODE field in the log page header (see table 403) serves as the indicator of what logged information is present. The PARAMETER CODE field identifies the SCSI Target Port to which the logged information applies.

The Protocol Specific Port log page has the format shown in table 403.

Table 403 — Protocol Specific Port log page

Bit / Byte	7	6	5	4	3	2	1	0
0	DS	SPF	PAGE CODE (18h)					
1	SUBPAGE CODE (00h to FEh)							
2	(MSB)		PAGE LENGTH (n-3)					
3								(LSB)
	Protocol specific port log parameters							
4								
•••	Protocol specific port log parameter [first]							
			⋮					
•••	Protocol specific port log parameter [last]							
n								

The DS bit, SPF bit, PAGE CODE field, SUBPAGE CODE field, and PAGE LENGTH field are described in 7.3.2. The SPF bit, PAGE CODE field, and SUBPAGE CODE field shall be set as shown in table 403 for the Protocol Specific Port log page.

The contents of each protocol specific port log parameter is defined by the corresponding SCSI transport protocol standard.

7.3.14.2 Generic protocol specific port log parameter

The generic format of a protocol specific port log parameter is shown in table 404.

Table 404 — Generic protocol specific port log parameter

Bit / Byte	7	6	5	4	3	2	1	0
0	(MSB)							
1	PARAMETER CODE							(LSB)
2	Parameter control byte – binary format list log parameter (see 7.3.2.2.2.5)							
	DU	Obsolete	TSD	ETC	TMC		FORMAT AND LINKING	
3	PARAMETER LENGTH (n-3)							
4	Reserved				PROTOCOL IDENTIFIER			
5	SCSI transport protocol specific							
•••								
n								

For the Generic protocol specific port log parameter, the PARAMETER CODE field is described in 7.3.2.2.1, and contains the relative target port identifier of the target port for which the parameter data applies.

The DU bit, TSD bit, ETC bit, TMC field, and FORMAT AND LINKING field are described in 7.3.2.2.2.1. These fields shall be set as described for a binary format list log parameter (see 7.3.2.2.2.5) for the Generic protocol specific port log parameter.

The PARAMETER LENGTH field is described in 7.3.2.2.1.

The PROTOCOL IDENTIFIER field contains one of the values shown in table 479 (see 7.6.1) to identify the SCSI transport protocol standard that defines the SCSI transport protocol specific data in this log parameter.

The SCSI transport protocol specific data is defined by the corresponding SCSI transport protocol standard.

7.3.15 Read Error Counters log page

7.3.15.1 Overview

The command standard that applies to the device type defines the operations that result in events that are reported in this log page.

Using the format shown in table 406, the Read Error Counters log page contains log parameters that report bounded data counters for detected events (e.g., total bytes processed) identified by the parameter codes listed in table 405.

Table 405 — Read Error Counters log page parameter codes

Parameter code	Description	Resettable or Changeable [a]	Reference	Support
0000h	Errors corrected without substantial delay [c]	Reset Only	7.3.15.2	At least one [b]
0001h	Errors corrected with possible delays [c]			
0002h	Total (e.g., rewrites or rereads) [c]			
0003h	Total errors corrected [c]			
0004h	Total times correction algorithm processed [c]			
0005h	Total bytes processed [c]			
0006h	Total uncorrected errors [c]			
8000h to FFFFh	Vendor specific			
all others	Reserved			

[a] The keywords in this column – Always, Reset Only, and Never – are defined in 7.3.3.

[b] If the Read Error Counters log page is supported, at least one of the parameter codes listed in this table shall be supported.

[c] The conditions that result in this error counter being modified are vendor specific. This counter should not be used to compare products because the products may define errors differently.

The Read Error Counters log page has the format shown in table 406.

Table 406 — Read Error Counters log page

Bit / Byte	7	6	5	4	3	2	1	0
0	DS	SPF (0b)	PAGE CODE (03h)					
1	SUBPAGE CODE (00h)							
2	(MSB)	PAGE LENGTH (n-3)						
3								(LSB)
	Read Error Counter log parameters							
4	Read Error Counter log parameter (see 7.3.15.2) [first]							
•••								
	⋮							
	Read Error Counter log parameter (see 7.3.15.2) [last]							
•••								
n								

The DS bit, SPF bit, PAGE CODE field, SUBPAGE CODE field, and PAGE LENGTH field are described in 7.3.2. The SPF bit, PAGE CODE field, and SUBPAGE CODE field shall be set as shown in table 406 for the Read Error Counters log page.

Each Read Error Counter log parameter contains the information described in 7.3.15.2.

7.3.15.2 Read Error Counter log parameter

The Read Error Counter log parameter has the format shown in table 407.

Table 407 — Read Error Counter log parameter

Bit / Byte	7	6	5	4	3	2	1	0
0	(MSB)	PARAMETER CODE (see table 405)						
1								(LSB)
2	Parameter control byte – bounded data counter log parameter (see 7.3.2.2.2.2)							
	DU	Obsolete	TSD	ETC	TMC		FORMAT AND LINKING	
3	PARAMETER LENGTH (n-3)							
4	(MSB)	READ ERROR COUNTER						
•••								
n								(LSB)

The PARAMETER CODE field is described in 7.3.2.2.1, and shall be set as shown in table 405 for the Read Error Counter log parameter.

The DU bit, TSD bit, ETC bit, TMC field, and FORMAT AND LINKING field are described in 7.3.2.2.2.1. These fields shall be set as described for a bounded data counter log parameter (see 7.3.2.2.2.2) for the Read Error Counter log parameter.

The PARAMETER LENGTH field is described in 7.3.2.2.1.

The READ ERROR COUNTER field contains the value for the counter described by the contents of the PARAMETER CODE field.

7.3.16 Read Reverse Error Counters log page

7.3.16.1 Overview

The command standard that applies to the device type defines the operations that result in events that are reported in this log page.

Using the format shown in table 409, the Read Reverse Error Counters log page contains log parameters that report bounded data counters for detected events (e.g., total bytes processed) identified by the parameter codes listed in table 408.

Table 408 — Read Reverse Error Counters log page parameter codes

Parameter code	Description	Resettable or Changeable [a]	Reference	Support
0000h	Errors corrected without substantial delay [c]	Reset Only	7.3.16.2	At least one [b]
0001h	Errors corrected with possible delays [c]			
0002h	Total (e.g., rewrites or rereads) [c]			
0003h	Total errors corrected [c]			
0004h	Total times correction algorithm processed [c]			
0005h	Total bytes processed [c]			
0006h	Total uncorrected errors [c]			
8000h to FFFFh	Vendor specific			
all others	Reserved			

[a] The keywords in this column – Always, Reset Only, and Never – are defined in 7.3.3.

[b] If the Read Reverse Error Counters log page is supported, at least one of the parameter codes listed in this table shall be supported.

[c] The conditions that result in this error counter being modified are vendor specific. This counter should not be used to compare products because the products may define errors differently.

The Read Reverse Error Counters log page has the format shown in table 409.

Table 409 — Read Reverse Error Counters log page

Bit / Byte	7	6	5	4	3	2	1	0
0	DS	SPF (0b)	PAGE CODE (04h)					
1	SUBPAGE CODE (00h)							
2	(MSB)							
3	PAGE LENGTH (n-3)							(LSB)
	Read reverse error counter log parameters							
4	Read Reverse Error Counter log parameter (see 7.3.16.2) [first]							
•••								
	⋮							
	Read Reverse Error Counter log parameter (see 7.3.16.2) [last]							
•••								
n								

The DS bit, SPF bit, PAGE CODE field, SUBPAGE CODE field, and PAGE LENGTH field are described in 7.3.2. The SPF bit, PAGE CODE field, and SUBPAGE CODE field shall be set as shown in table 409 for the Read Reverse Error Counters log page.

Each Read Reverse Error Counter log parameter contains the information described in 7.3.16.2.

7.3.16.2 Read Reverse Error Counter log parameter

The Read Reverse Error Counter log parameter has the format shown in table 410.

Table 410 — Read Reverse Error Counter log parameter

Bit / Byte	7	6	5	4	3	2	1	0
0	(MSB)							
1	PARAMETER CODE (see table 408)							(LSB)
2	Parameter control byte – bounded data counter log parameter (see 7.3.2.2.2.2)							
	DU	Obsolete	TSD	ETC	TMC		FORMAT AND LINKING	
3	PARAMETER LENGTH (n-3)							
4	(MSB)							
•••	READ REVERSE ERROR COUNTER							
n								(LSB)

The PARAMETER CODE field is described in 7.3.2.2.1, and shall be set as shown in table 408 for the Read Reverse Error Counter log parameter.

The DU bit, TSD bit, ETC bit, TMC field, and FORMAT AND LINKING field are described in 7.3.2.2.2.1. These fields shall be set as described for a bounded data counter log parameter (see 7.3.2.2.2.2) for the Read Reverse Error Counter log parameter.

The PARAMETER LENGTH field is described in 7.3.2.2.1.

The READ REVERSE ERROR COUNTER field contains the value for the counter described by the contents of the PARAMETER CODE field.

7.3.17 Self-Test Results log page

7.3.17.1 Overview

Using the format shown in table 412, the Self-Test Results log page reports the results from the 20 most recent self-tests (see 5.14). The parameter codes for the Self-Test Results log page are listed in table 411.

Table 411 — Self-Test Results log page parameter codes

Parameter code	Description	Resettable or Changeable [a]	Reference	Support
0001h to 0014h	Self-Test Results	Reset Only	7.3.17.2	Mandatory
all others	Reserved			

[a] The keywords in this column – Always, Reset Only, and Never – are defined in 7.3.3.

The Self-Test Results log page has the format shown in table 412.

Table 412 — Self-Test Results log page

Bit / Byte	7	6	5	4	3	2	1	0
0	DS	SPF (0b)	PAGE CODE (10h)					
1	SUBPAGE CODE (00h)							
2	(MSB)	PAGE LENGTH (0190h)						
3								(LSB)
	Self-test results log parameters							
4	Self-test results log parameter [first]							
•••								
23								
	⋮							
384	Self-test results log parameter [twentieth]							
•••								
403								

The DS bit, SPF bit, PAGE CODE field, SUBPAGE CODE field, and PAGE LENGTH field are described in 7.3.2. The SPF bit, PAGE CODE field, SUBPAGE CODE field, and PAGE LENGTH field shall be set as shown in table 412 for the Self-Test Results log page.

The PARAMETER CODE field indicates the time at which the self-test has been performed with respect to other self-tests (i.e., 0001h indicates the most recent self-test, 0002h indicates the second most recent self-test, etc.). If fewer than 20 self-tests have been performed, then:

a) unused self-test log parameters shall have parameter code values higher than those of any used self-test log parameter; and

b) each unused self-test log parameter entry shall have the format shown in table 413.

Table 413 — Unused Self-Test Results log parameter

Bit / Byte	7	6	5	4	3	2	1	0
0	(MSB)							
1	PARAMETER CODE (see table 411)							(LSB)
2	Parameter control byte – binary format list log parameter (see 7.3.2.2.2.5)							
	DU	Obsolete	TSD	ETC	TMC		FORMAT AND LINKING	
3	PARAMETER LENGTH (10h)							
4	ZERO							
•••								
19								

The PARAMETER CODE field is described in 7.3.2.2.1, and shall be set as shown in table 411 for each Unused Self-Test Results log parameter.

The DU bit, TSD bit, ETC bit, TMC field, and FORMAT AND LINKING field are described in 7.3.2.2.2.1. These fields shall be set as described for a binary format list log parameter (see 7.3.2.2.2.5) for each Unused Self-Test Results log parameter.

The PARAMETER LENGTH field is described in 7.3.2.2.1, and shall be set as shown in table 413 for each Unused Self-Test Results log parameter.

The ZERO field is set to zero.

7.3.17.2 Self-Test Results log parameters

Each Self-Test Results log parameter has the format shown in table 414.

Table 414 — Self-Test Results log parameter

Bit / Byte	7	6	5	4	3	2	1	0
0	(MSB)			PARAMETER CODE (see table 411)				
1								(LSB)
2	Parameter control byte – binary format list log parameter (see 7.3.2.2.2.5)							
	DU	Obsolete	TSD	ETC	TMC		FORMAT AND LINKING	
3	PARAMETER LENGTH (10h)							
4	SELF-TEST CODE			Reserved	SELF-TEST RESULTS			
5	SELF-TEST NUMBER							
6	(MSB)			ACCUMULATED POWER ON HOURS				
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 is described in 7.3.2.2.1, and shall be set as shown in table 411 for the Self-Test Results log parameter.

The DU bit, TSD bit, ETC bit, TMC field, and FORMAT AND LINKING field are described in 7.3.2.2.2.1. These fields shall be set as described for a binary format list log parameter (see 7.3.2.2.2.5) for the Self-Test Results log parameter.

The PARAMETER LENGTH field is described in 7.3.2.2.1, and shall be set as shown in table 414 for the Self-Test Results log parameter.

The SELF-TEST CODE field indicates the value in the SELF-TEST CODE field of the SEND DIAGNOSTIC command (see 6.42) that initiated this self-test.

Table 415 defines the content of the SELF-TEST RESULTS field.

Table 415 — SELF-TEST RESULTS field

Code	Description
0h	The self-test completed without error.
1h	The background self-test was aborted by the application client using a SEND DIAGNOSTIC command (see 6.42) with the SELF-TEST CODE field set to 100b (i.e., abort background self-test).
2h	The self-test routine was aborted by an application client using a method other than a SEND DIAGNOSTIC command with the SELF-TEST CODE field set to 100b (e.g., by a task management function, or by issuing an exception command as defined in 5.14.4).
3h	An unknown error occurred while the device server was processing 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 and which test is indicated by the contents of the SELF-TEST NUMBER field.
8h to 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 one field.

If the specific segment in which the failure occurred is not able to be identified, the SELF-TEST NUMBER field shall contain 00h.

The ACCUMULATED POWER ON HOURS field indicates the elapsed hours of powered on operation since manufacture at the time the self-test was completed. If the self-test is still in progress, the ACCUMULATED POWER ON HOURS field shall be set to zero. If the number of hours is greater than FFFFh, the ACCUMULATED POWER ON HOURS field shall be set to FFFFh.

The ADDRESS OF FIRST FAILURE field indicates 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 has no implication about the quality of any other logical block on the logical unit (e.g., 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 FFFF FFFF FFFF FFFFh 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 field, ADDITIONAL SENSE CODE field, and ADDITIONAL SENSE CODE QUALIFIER field 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 4.5).

7.3.18 Start-Stop Cycle Counter log page

7.3.18.1 Overview

Using the format shown in table 417, the Start-Stop Cycle Counter log page provides information about manufacturing dates and cycle counts since date of manufacture using the parameter codes listed in table 416.

Table 416 — Start-Stop Cycle Counter log page parameter codes

Parameter code	Description	Resettable or Changeable [a]	Reference	Support
0001h	Date of Manufacture	Never	7.3.18.2	At least one [b]
0002h	Accounting Date	Always	7.3.18.3	
0003h	Specified Cycle Count Over Device Lifetime	Never	7.3.18.4	
0004h	Accumulated Start-Stop Cycles	Never	7.3.18.5	
0005h	Specified Load-Unload Count Over Device Lifetime	Never	7.3.18.6	
0006h	Accumulated Load-Unload Cycles	Never	7.3.18.7	
all others	Reserved			

[a] The keywords in this column – Always, Reset Only, and Never – are defined in 7.3.3.

[b] If the Start-Stop Cycle Counter log page is supported, at least one of the parameter codes listed in this table shall be supported.

The Start-Stop Cycle Counter log page has the format shown in table 417.

Table 417 — Start-Stop Cycle Counter log page

Bit / Byte	7	6	5	4	3	2	1	0
0	DS	SPF (0b)	PAGE CODE (0Eh)					
1	SUBPAGE CODE (00h)							
2	(MSB)							
3	PAGE LENGTH (n-3)							(LSB)
	Start stop cycle log parameters							
4	Start stop cycle log parameter (see table 416) [first]							
...								
	⋮							
...	Start stop cycle log parameter (see table 416) [last]							
n								

The DS bit, SPF bit, PAGE CODE field, SUBPAGE CODE field, and PAGE LENGTH field are described in 7.3.2. The SPF bit, PAGE CODE field, and SUBPAGE CODE field shall be set as shown in table 417 for the Start-Stop Cycle Counter log page.

7.3.18.2 Date of Manufacture log parameter

The Date of Manufacture log parameter has the format shown in table 418.

Table 418 — Date of Manufacture log parameter

Bit / Byte	7	6	5	4	3	2	1	0
0	(MSB)			PARAMETER CODE (0001h)				
1								(LSB)
2	Parameter control byte – ASCII format list log parameter (see 7.3.2.2.2.4)							
	DU	Obsolete	TSD	ETC	TMC		FORMAT AND LINKING	
3	PARAMETER LENGTH (06h)							
4	(MSB)			YEAR OF MANUFACTURE				
•••								
7								(LSB)
8	(MSB)			WEEK OF MANUFACTURE				
9								(LSB)

The PARAMETER CODE field is described in 7.3.2.2.1, and shall be set as shown in table 418 for the Date of Manufacture log parameter.

The DU bit, TSD bit, ETC bit, TMC field, and FORMAT AND LINKING field are described in 7.3.2.2.2.1. These fields shall be set as described for a ASCII format list log parameter (see 7.3.2.2.2.4) for the Date of Manufacture log parameter.

The PARAMETER LENGTH field is described in 7.3.2.2.1, and shall be set as shown in table 418 for the Date of Manufacture log parameter.

The YEAR OF MANUFACTURE field indicates the year in which the SCSI target device was manufactured and contains four numeric ASCII characters (e.g., 30h for zero and 39h for nine).

The WEEK OF MANUFACTURE field indicates the week of the year in which the SCSI target device was manufactured and contains two numeric ASCII characters.

7.3.18.3 Accounting Date log parameter

The Accounting Date log parameter has the format shown in table 419.

Table 419 — Accounting Date log parameter

Bit / Byte	7	6	5	4	3	2	1	0
0	(MSB)			PARAMETER CODE (0002h)				
1								(LSB)
2	Parameter control byte – ASCII format list log parameter (see 7.3.2.2.2.4)							
	DU	Obsolete	TSD	ETC	TMC		FORMAT AND LINKING	
3	PARAMETER LENGTH (06h)							
4	(MSB)			ACCOUNTING DATE YEAR				
•••								
7								(LSB)
8	(MSB)			ACCOUNTING DATE WEEK				
9								(LSB)

The PARAMETER CODE field is described in 7.3.2.2.1, and shall be set as shown in table 419 for the Accounting Date log parameter.

The DU bit, TSD bit, ETC bit, TMC field, and FORMAT AND LINKING field are described in 7.3.2.2.2.1. These fields shall be set as described for a ASCII format list log parameter (see 7.3.2.2.2.4) for the Accounting Date log parameter.

The PARAMETER LENGTH field is described in 7.3.2.2.1, and shall be set as shown in table 419 for the Accounting Date log parameter.

The ACCOUNTING DATE YEAR field indicates the year in which the SCSI target device was placed in service and contains four numeric ASCII characters (e.g., 30h for zero and 39h for nine).

The ACCOUNTING DATE WEEK field indicates the week of the year in which the SCSI target device was placed in service and contains two numeric ASCII characters.

A LOG SELECT command may be used to change the value of the ACCOUNTING DATE YEAR field and ACCOUNTING DATE WEEK field. If the Accounting Date log parameter is not yet set, then the value placed:

a) in the ACCOUNTING DATE YEAR field shall be four ASCII space characters (i.e., 20h); and
b) in the ACCOUNTING DATE WEEK field shall be two ASCII space characters (i.e., 20h).

The ACCOUNTING DATE YEAR field and ACCOUNTING DATE WEEK field shall not be checked for validity by the device server.

7.3.18.4 Specified Cycle Count Over Device Lifetime log parameter

The Specified Cycle Count Over Device Lifetime log parameter has the format shown in table 420.

Table 420 — Specified Cycle Count Over Device Lifetime log parameter

Bit / Byte	7	6	5	4	3	2	1	0
0	(MSB)							
1	PARAMETER CODE (0003h)							(LSB)
2	Parameter control byte – binary format list log parameter (see 7.3.2.2.2.5)							
	DU	Obsolete	TSD	ETC	TMC		FORMAT AND LINKING	
3	PARAMETER LENGTH (04h)							
4	(MSB)							
•••	SPECIFIED CYCLE COUNT OVER DEVICE LIFETIME							
7								(LSB)

The PARAMETER CODE field is described in 7.3.2.2.1, and shall be set as shown in table 420 for the Specified Cycle Count Over Device Lifetime log parameter.

The DU bit, TSD bit, ETC bit, TMC field, and FORMAT AND LINKING field are described in 7.3.2.2.2.1. These fields shall be set as described for a binary format list log parameter (see 7.3.2.2.2.5) for the Specified Cycle Count Over Device Lifetime log parameter.

The PARAMETER LENGTH field is described in 7.3.2.2.1, and shall be set as shown in table 420 for the Specified Cycle Count Over Device Lifetime log parameter.

The contents of the SPECIFIED CYCLE COUNT OVER DEVICE LIFETIME field indicate the number of stop-start cycles that may be performed over the lifetime of the SCSI target device without degrading the SCSI target device's operation or reliability outside the limits specified by the manufacturer of the SCSI target device.

7.3.18.5 Accumulated Start-Stop Cycles log parameter

The Accumulated Start-Stop Cycles log parameter has the format shown in table 421.

Table 421 — Accumulated Start-Stop Cycles log parameter

Bit / Byte	7	6	5	4	3	2	1	0
0	(MSB)							
1	PARAMETER CODE (0004h)							(LSB)
2	Parameter control byte – binary format list log parameter (see 7.3.2.2.2.5)							
	DU	Obsolete	TSD	ETC	TMC		FORMAT AND LINKING	
3	PARAMETER LENGTH (04h)							
4	(MSB)							
•••	ACCUMULATED START-STOP CYCLES							
7								(LSB)

The PARAMETER CODE field is described in 7.3.2.2.1, and shall be set as shown in table 421 for the Accumulated Start-Stop Cycles log parameter.

The DU bit, TSD bit, ETC bit, TMC field, and FORMAT AND LINKING field are described in 7.3.2.2.2.1. These fields shall be set as described for a binary format list log parameter (see 7.3.2.2.2.5) for the Accumulated Start-Stop Cycles log parameter.

The PARAMETER LENGTH field is described in 7.3.2.2.1, and shall be set as shown in table 421 for the Accumulated Start-Stop Cycles log parameter.

The contents of the ACCUMULATED START-STOP CYCLES field indicate the number of stop-start cycles the SCSI target device has detected since its date of manufacture. This saturating counter is incremented by one for each complete cycle. The time in the cycle at which the counter is incremented is vendor specific.

For rotating magnetic storage devices (see SBC-3), a single start-stop cycle is defined as an operational cycle that:

a) begins with the disk spindle at rest;
b) continues while the disk accelerates to its normal operational rotational rate;
c) continues during the entire period the disk is rotating;
d) continues as the disk decelerates toward a resting state; and
e) 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.

7.3.18.6 Specified Load-Unload Count Over Device Lifetime log parameter

The Specified Load-Unload Count Over Device Lifetime log parameter has the format shown in table 422.

Table 422 — Specified Load-Unload Count Over Device Lifetime log parameter

Bit / Byte	7	6	5	4	3	2	1	0
0	(MSB)							
1	PARAMETER CODE (0005h)							(LSB)
2	Parameter control byte – binary format list log parameter (see 7.3.2.2.2.5)							
	DU	Obsolete	TSD	ETC	TMC		FORMAT AND LINKING	
3	PARAMETER LENGTH (04h)							
4	(MSB)							
•••	SPECIFIED LOAD-UNLOAD COUNT OVER DEVICE LIFETIME							
7								(LSB)

The PARAMETER CODE field is described in 7.3.2.2.1, and shall be set as shown in table 422 for the Specified Load-Unload Count Over Device Lifetime log parameter.

The DU bit, TSD bit, ETC bit, TMC field, and FORMAT AND LINKING field are described in 7.3.2.2.2.1. These fields shall be set as described for a binary format list log parameter (see 7.3.2.2.2.5) for the Specified Load-Unload Count Over Device Lifetime log parameter.

The PARAMETER LENGTH field is described in 7.3.2.2.1, and shall be set as shown in table 422 for the Specified Load-Unload Count Over Device Lifetime log parameter.

The contents of the SPECIFIED LOAD-UNLOAD COUNT OVER DEVICE LIFETIME field indicate the number of load-unload cycles that may be performed over the lifetime of the SCSI target device without degrading the SCSI target device's operation or reliability outside the limits specified by the manufacturer of the SCSI target device.

7.3.18.7 Accumulated Load-Unload Cycles log parameter

The Accumulated Load-Unload Cycles log parameter has the format shown in table 423.

Table 423 — Accumulated Load-Unload Cycles log parameter

Bit / Byte	7	6	5	4	3	2	1	0
0	(MSB)							
1				PARAMETER CODE (0006h)				(LSB)
2	Parameter control byte – binary format list log parameter (see 7.3.2.2.2.5)							
	DU	Obsolete	TSD	ETC	TMC		FORMAT AND LINKING	
3	PARAMETER LENGTH (04h)							
4	(MSB)							
•••				ACCUMULATED LOAD-UNLOAD CYCLES				
7								(LSB)

The PARAMETER CODE field is described in 7.3.2.2.1, and shall be set as shown in table 423 for the Accumulated Load-Unload Cycles log parameter.

The DU bit, TSD bit, ETC bit, TMC field, and FORMAT AND LINKING field are described in 7.3.2.2.2.1. These fields shall be set as described for a binary format list log parameter (see 7.3.2.2.2.5) for the Accumulated Load-Unload Cycles log parameter.

The PARAMETER LENGTH field is described in 7.3.2.2.1, and shall be set as shown in table 423 for the Accumulated Load-Unload Cycles log parameter.

The contents of the ACCUMULATED LOAD-UNLOAD CYCLES field indicate the number of load-unload cycles the SCSI target device has detected since its date of manufacture. This saturating counter is incremented by one for each complete cycle. The time in the cycle at which the counter is incremented is vendor specific.

For rotating magnetic storage devices (see SBC-3), a single load-unload cycle is defined as an operational cycle that:

a) begins with the heads unloaded from the medium;
b) continues while the heads are loaded onto the spinning medium; and
c) ends when the heads are unloaded from the medium.

The Accumulated Load-Unload Cycles log parameter is not applicable to rotating magnetic storage devices without unloadable heads.

7.3.19 Supported Log Pages log page

For the LOG SENSE command, the Supported Log Pages log page (see table 424) returns the list of log pages implemented by the logical unit. Logical units that implement the LOG SENSE command shall implement this log page. This log page is not defined for the LOG SELECT command.

Table 424 — Supported Log Pages log page

Bit / Byte	7	6	5	4	3	2	1	0
0	DS (0b)	SPF (0b)	PAGE CODE (00h)					
1	SUBPAGE CODE (00h)							
2	(MSB)	PAGE LENGTH (n-3)						
3								(LSB)
	Supported pages list							
4	Supported page descriptor (see table 425) [first]							
	⋮							
n	Supported page descriptor (see table 425) [last]							

The DS bit, SPF bit, PAGE CODE field, SUBPAGE CODE field, and PAGE LENGTH field are described in 7.3.2. The DS bit, SPF bit, PAGE CODE field, and SUBPAGE CODE field shall be set as shown in table 424 for the Supported Log Pages log page.

The supported page descriptors shall contain a list of all log page codes (see table 425) with a subpage code of zero implemented by the logical unit in ascending order beginning with page code 00h.

Table 425 — Supported page descriptor

Bit / Byte	7	6	5	4	3	2	1	0
0	Reserved		PAGE CODE					

The PAGE CODE field indicates the number of a supported log page.

7.3.20 Supported Log Pages and Subpages log page

For the LOG SENSE command, the Supported Log Pages and Subpages log page (see table 426) returns the list of log pages and subpages implemented by the logical unit. If log subpages are supported, this page shall be supported. This log page is not defined for the LOG SELECT command.

Table 426 — Supported Log Pages and Subpages log page

Bit / Byte	7	6	5	4	3	2	1	0
0	DS (0b)	SPF (1b)	PAGE CODE (00h)					
1	SUBPAGE CODE (FFh)							
2	(MSB)	PAGE LENGTH (n-3)						
3								(LSB)
	Supported page/subpage descriptors							
4	Supported page/subpage descriptor (see table 427) [first]							
5								
	⋮							
n-1	Supported page/subpage descriptor (see table 427) [last]							
n								

The DS bit, SPF bit, PAGE CODE field, SUBPAGE CODE field, and PAGE LENGTH field are described in 7.3.2. The DS bit, SPF bit, PAGE CODE field, and SUBPAGE CODE field shall be set as shown in table 426 for the Supported Log Pages and Subpages log page.

The supported page/subpage descriptors (see table 427) shall be in ascending order sorted by page code then subpage code.

Table 427 — Supported page/subpage descriptor

Bit / Byte	7	6	5	4	3	2	1	0
0	Reserved		PAGE CODE					
1	SUBPAGE CODE							

The PAGE CODE field indicates the number of a supported log page.

The SUBPAGE CODE field indicates the supported subpage number of a supported log page.

7.3.21 Supported Subpages log page

For the LOG SENSE command, the Supported Subpages log page (see table 428) returns the list of all subpage codes (i.e., 00h to FFh) that are implemented by the logical unit for a specified page code. If log subpages are supported, this page shall be supported. This log page is not defined for the LOG SELECT command.

Table 428 — Supported Subpages log page

Bit / Byte	7	6	5	4	3	2	1	0
0	DS (0b)	SPF (1b)	PAGE CODE					
1	SUBPAGE CODE (FFh)							
2	(MSB)							
3	PAGE LENGTH (n-3)							(LSB)
	Supported subpage descriptors							
4	Supported subpage descriptor (see table 429) [first]							
5								
	⋮							
n-1	Supported subpage descriptor (see table 429) [last]							
n								

The DS bit, SPF bit, SUBPAGE CODE field, and PAGE LENGTH field are described in 7.3.2. The DS bit, SPF bit, and SUBPAGE CODE field shall be set as shown in table 428 for the Supported Subpages log page.

The PAGE CODE field (see 7.3.2) indicates the log page for which log page and subpage codes are being returned.

The supported subpage descriptors (see table 429) shall be in ascending order sorted by page code then subpage code, and shall include a descriptor with subpage code 00h for any implemented log page in which the SPF bit is set to zero.

Table 429 — Supported subpage descriptor

Bit / Byte	7	6	5	4	3	2	1	0
0	Reserved		PAGE CODE					
1	SUBPAGE CODE							

The PAGE CODE field indicates the number of a supported log page.

The page code is the same in the page header (see table 428) and in each supported subpage descriptor (see table 429).

The SUBPAGE CODE field indicates the supported subpage number of a supported log page.

7.3.22 Temperature log page

7.3.22.1 Overview

Using the format shown in table 431, the Temperature log page provides information about the current operating temperature of the SCSI Target Device using the parameter codes listed in table 430.

Table 430 — Temperature log page parameter codes

Parameter code	Description	Resettable or Changeable [a]	Reference	Support
0000h	Temperature	Never	7.3.22.2	Mandatory
0001h	Reference Temperature	Never	7.3.22.3	Optional
all others	Reserved			

[a] The keywords in this column – Always, Reset Only, and Never – are defined in 7.3.3.

The Temperature log page has the format shown in table 431.

Table 431 — Temperature log page

Bit / Byte	7	6	5	4	3	2	1	0
0	DS	SPF (0b)	PAGE CODE (0Dh)					
1	SUBPAGE CODE (00h)							
2	(MSB)	PAGE LENGTH (n-3)						
3								(LSB)
	Temperature log parameters							
4	Temperature log parameter (see table 430) [first]							
•••								
	⋮							
•••	Temperature log parameter (see table 430) [last]							
n								

The DS bit, SPF bit, PAGE CODE field, SUBPAGE CODE field, and PAGE LENGTH field are described in 7.3.2. The SPF bit, PAGE CODE field, and SUBPAGE CODE field shall be set as shown in table 431 for the Temperature log page.

The contents of each temperature log parameter depends on the value in its PARAMETER CODE field (see table 430).

7.3.22.2 Temperature log parameter

The Temperature log parameter has the format shown in table 432.

Table 432 — Temperature log parameter

Bit / Byte	7	6	5	4	3	2	1	0
0	(MSB)							
1	PARAMETER CODE (0000h)							(LSB)
2	Parameter control byte – binary format list log parameter (see 7.3.2.2.2.5)							
	DU	Obsolete	TSD	ETC	TMC		FORMAT AND LINKING	
3	PARAMETER LENGTH (02h)							
4	Reserved							
5	TEMPERATURE							

The PARAMETER CODE field is described in 7.3.2.2.1, and shall be set as shown in table 432 for the Temperature log parameter.

The DU bit, TSD bit, ETC bit, TMC field, and FORMAT AND LINKING field are described in 7.3.2.2.2.1. These fields shall be set as described for a binary format list log parameter (see 7.3.2.2.2.5) for the Temperature log parameter.

The PARAMETER LENGTH field is described in 7.3.2.2.1, and shall be set as shown in table 432 for the Temperature log parameter.

The TEMPERATURE field indicates the temperature of the SCSI target device in degrees Celsius at the time the LOG SENSE command is performed. Temperatures equal to or less than zero degrees Celsius shall cause the TEMPERATURE field to be set to zero. If the device server is unable to detect a valid temperature because of a sensor failure or other condition, then the TEMPERATURE field shall be set to FFh. The temperature should be reported with an accuracy of plus or minus three Celsius degrees while the SCSI target device is operating at a steady state within its environmental limits.

7.3.22.3 Reference Temperature log parameter

The Reference Temperature log parameter has the format shown in table 433.

Table 433 — Reference Temperature log parameter

Bit / Byte	7	6	5	4	3	2	1	0
0	(MSB)							
1	PARAMETER CODE (0001h)							(LSB)
2	Parameter control byte – binary format list log parameter (see 7.3.2.2.2.5)							
	DU	Obsolete	TSD	ETC	TMC		FORMAT AND LINKING	
3	PARAMETER LENGTH (02h)							
4	Reserved							
5	REFERENCE TEMPERATURE							

The PARAMETER CODE field is described in 7.3.2.2.1, and shall be set as shown in table 433 for the Reference Temperature log parameter.

The DU bit, TSD bit, ETC bit, TMC field, and FORMAT AND LINKING field are described in 7.3.2.2.2.1. These fields shall be set as described for a binary format list log parameter (see 7.3.2.2.2.5) for the Reference Temperature log parameter.

The PARAMETER LENGTH field is described in 7.3.2.2.1, and shall be set as shown in table 433 for the Reference Temperature log parameter.

The REFERENCE TEMPERATURE field indicates the maximum reported sensor temperature in degrees Celsius at which the SCSI target device is capable of operating continuously without degrading the SCSI target device's operation or reliability beyond manufacturer accepted limits. If the device server is unable to return a reference temperature and the optional Reference Temperature log parameter is included in the Temperature log page, then REFERENCE TEMPERATURE field is set to FFh.

The reference temperature may change for vendor specific reasons.

7.3.23 Verify Error Counters log page

7.3.23.1 Overview

The command standard that applies to the device type defines the operations that result in events that are reported in this log page.

Using the format shown in table 435, the Verify Error Counters log page contains log parameters that report bounded data counters for detected events (e.g., total bytes processed) identified by the parameter codes listed in table 434.

Table 434 — Verify Error Counters log page parameter codes

Parameter code	Description	Resettable or Changeable [a]	Reference	Support
0000h	Errors corrected without substantial delay [c]	Reset Only	7.3.23.2	At least one [b]
0001h	Errors corrected with possible delays [c]			
0002h	Total (e.g., rewrites or rereads) [c]			
0003h	Total errors corrected [c]			
0004h	Total times correction algorithm processed [c]			
0005h	Total bytes processed [c]			
0006h	Total uncorrected errors [c]			
8000h to FFFFh	Vendor specific			
all others	Reserved			

[a] The keywords in this column – Always, Reset Only, and Never – are defined in 7.3.3.

[b] If the Verify Error Counters log page is supported, at least one of the parameter codes listed in this table shall be supported.

[c] The conditions that result in this error counter being modified are vendor specific. This counter should not be used to compare products because the products may define errors differently.

The Verify Error Counters log page has the format shown in table 435.

Table 435 — Verify Error Counters log page

Bit / Byte	7	6	5	4	3	2	1	0
0	DS	SPF (0b)	PAGE CODE (05h)					
1	SUBPAGE CODE (00h)							
2	(MSB)	PAGE LENGTH (n-3)						
3								(LSB)
	Verify error counter log parameters							
4	Verify Error Counter log parameter (see 7.3.23.2) [first]							
•••								
	⋮							
	Verify Error Counter log parameter (see 7.3.23.2) [last]							
•••								
n								

The DS bit, SPF bit, PAGE CODE field, SUBPAGE CODE field, and PAGE LENGTH field are described in 7.3.2. The SPF bit, PAGE CODE field, and SUBPAGE CODE field shall be set as shown in table 435 for the Verify Error Counters log page.

Each Verify Error Counter log parameter contains the information described in 7.3.23.2.

7.3.23.2 Verify Error Counter log parameter

The Verify Error Counter log parameter has the format shown in table 436.

Table 436 — Verify Error Counter log parameter

Bit / Byte	7	6	5	4	3	2	1	0
0	(MSB)	PARAMETER CODE (see table 434)						
1								(LSB)
2	Parameter control byte – bounded data counter log parameter (see 7.3.2.2.2.2)							
	DU	Obsolete	TSD	ETC	TMC		FORMAT AND LINKING	
3	PARAMETER LENGTH (n-3)							
4	(MSB)	VERIFY ERROR COUNTER						
•••								
n								(LSB)

The PARAMETER CODE field is described in 7.3.2.2.1, and shall be set as shown in table 434 for the Verify Error Counter log parameter.

The DU bit, TSD bit, ETC bit, TMC field, and FORMAT AND LINKING field are described in 7.3.2.2.2.1. These fields shall be set as described for a bounded data counter log parameter (see 7.3.2.2.2.2) for the Verify Error Counter log parameter.

The PARAMETER LENGTH field is described in 7.3.2.2.1.

The VERIFY ERROR COUNTER field contains the value for the counter described by the contents of the PARAMETER CODE field.

7.3.24 Write Error Counters log page

7.3.24.1 Overview

The command standard that applies to the device type defines the operations that result in events that are reported in this log page.

Using the format shown in table 438, the Write Error Counters log page contains log parameters that report bounded data counters for detected events (e.g., total bytes processed) identified by the parameter codes listed in table 437.

Table 437 — Write Error Counters log page parameter codes

Parameter code	Description	Resettable or Changeable [a]	Reference	Support
0000h	Errors corrected without substantial delay [c]	Reset Only	7.3.24.2	At least one [b]
0001h	Errors corrected with possible delays [c]			
0002h	Total (e.g., rewrites or rereads) [c]			
0003h	Total errors corrected [c]			
0004h	Total times correction algorithm processed [c]			
0005h	Total bytes processed [c]			
0006h	Total uncorrected errors [c]			
8000h to FFFFh	Vendor specific			
all others	Reserved			

[a] The keywords in this column – Always, Reset Only, and Never – are defined in 7.3.3.

[b] If the Write Error Counters log page is supported, at least one of the parameter codes listed in this table shall be supported.

[c] The conditions that result in this error counter being modified are vendor specific. This counter should not be used to compare products because the products may define errors differently.

The Write Error Counters log page has the format shown in table 438.

Table 438 — Write Error Counters log page

Bit / Byte	7	6	5	4	3	2	1	0
0	DS	SPF (0b)	PAGE CODE (02h)					
1	SUBPAGE CODE (00h)							
2	(MSB)	PAGE LENGTH (n-3)						
3								(LSB)
	Write error counter log parameters							
4	Write Error Counter log parameter (see 7.3.24.2) [first]							
•••								
	⋮							
	Write Error Counter log parameter (see 7.3.24.2) [last]							
•••								
n								

The DS bit, SPF bit, PAGE CODE field, SUBPAGE CODE field, and PAGE LENGTH field are described in 7.3.2. The SPF bit, PAGE CODE field, and SUBPAGE CODE field shall be set as shown in table 438 for the Write Error Counters log page.

Each Write Error Counter log parameter contains the information described in 7.3.24.2.

7.3.24.2 Write Error Counter log parameter

The Write Error Counter log parameter has the format shown in table 439.

Table 439 — Write Error Counter log parameter

Bit / Byte	7	6	5	4	3	2	1	0
0	(MSB)	PARAMETER CODE (see table 437)						
1								(LSB)
2	Parameter control byte – bounded data counter log parameter (see 7.3.2.2.2.2)							
	DU	Obsolete	TSD	ETC	TMC		FORMAT AND LINKING	
3	PARAMETER LENGTH (n-3)							
4	(MSB)	WRITE ERROR COUNTER						
•••								
n								(LSB)

The PARAMETER CODE field is described in 7.3.2.2.1, and shall be set as shown in table 437 for the Write Error Counter log parameter.

The DU bit, TSD bit, ETC bit, TMC field, and FORMAT AND LINKING field are described in 7.3.2.2.2.1. These fields shall be set as described for a bounded data counter log parameter (see 7.3.2.2.2.2) for the Write Error Counter log parameter.

The PARAMETER LENGTH field is described in 7.3.2.2.1.

The WRITE ERROR COUNTER field contains the value for the counter described by the contents of the PARAMETER CODE field.

7.4 Medium auxiliary memory attributes

7.4.1 Attribute format

Each medium auxiliary memory attribute shall be communicated between the application client and device server in the format shown in table 440. This format shall be used in the parameter data for the WRITE ATTRIBUTE command (see 6.48) and the READ ATTRIBUTE command (see 6.17). The attribute format in this standard implies nothing about the physical representation of an attribute in the medium auxiliary memory.

Table 440 — MAM ATTRIBUTE format

Bit / Byte	7	6	5	4	3	2	1	0
0	(MSB)			ATTRIBUTE IDENTIFIER				
1								(LSB)
2	READ ONLY			Reserved			FORMAT	
3	(MSB)			ATTRIBUTE LENGTH (n-4)				
4								(LSB)
5				ATTRIBUTE VALUE				
•••								
n								

The ATTRIBUTE IDENTIFIER field contains a code value identifying the attribute (see 7.4.2).

The READ ONLY bit indicates whether the attribute is in the read only state (see 5.7). If the READ ONLY bit is set to one, the attribute is in the read only state. If the READ ONLY bit is set to zero, the attribute is not in the read only state.

The FORMAT field (see table 441) specifies the format of the data in the ATTRIBUTE VALUE field.

Table 441 — MAM attribute FORMAT field

Format	Name	Description
00b	BINARY	The ATTRIBUTE VALUE field contains binary data.
01b	ASCII	The ATTRIBUTE VALUE field contains left-aligned ASCII data (see 4.3.1).
10b	TEXT	The attribute contains textual data. The character set is as described in the TEXT LOCALIZATION IDENTIFIER attribute (see 7.4.2.4.6).
11b		Reserved

The ATTRIBUTE LENGTH field specifies the length in bytes of the ATTRIBUTE VALUE field.

The ATTRIBUTE VALUE field contains the:

a) current value of the attribute for the READ ATTRIBUTE command (see 6.17); or
b) intended value of the attribute for the WRITE ATTRIBUTE command (see 6.48).

7.4.2 Attribute identifier values

7.4.2.1 Attribute identifier values overview

The values in the ATTRIBUTE IDENTIFIER field (see 7.4.1) are assigned according to the attribute type (see 5.7) and whether the attribute is standard or vendor specific (see table 442).

Table 442 — MAM attribute identifier range assignments

Attribute Identifiers	Attribute Type	Standardized	Reference
0000h to 03FFh	Device	Yes	7.4.2.2
0400h to 07FFh	Medium	Yes	7.4.2.3
0800h to 0BFFh	Host	Yes	7.4.2.4
0C00h to 0FFFh	Device	Vendor specific	
1000h to 13FFh	Medium	Vendor specific	
1400h to 17FFh	Host	Vendor specific	
1800h to FFFFh	Reserved		

Device servers may process a WRITE ATTRIBUTE command containing standardized host type attribute identifier values (i.e., 0800h-0BFFh) or vendor specific host type attribute identifier values (i.e., 1400h-17FFh). Standardized host type attribute identifier values may be verified as described in 7.4.2.4.

7.4.2.2 Device type attributes

Device type attributes (see table 443) shall be maintained and updated by the device server while the medium and associated medium auxiliary memory are present. All supported device type attributes shall have a status of read only or unavailable (see 5.7).

Table 443 — Device type attributes (part 1 of 2)

Attribute Identifier	Name	Attribute Length (in bytes)	Format [a]	Reference
0000h	REMAINING CAPACITY IN PARTITION	8	BINARY	7.4.2.2.1
0001h	MAXIMUM CAPACITY IN PARTITION	8	BINARY	7.4.2.2.1
0002h	Restricted (see SSC-4)			
0003h	LOAD COUNT	8	BINARY	7.4.2.2.2
0004h	MAM SPACE REMAINING	8	BINARY	7.4.2.2.3
0005h to 0006h	Restricted (see SSC-4)			
0007h	INITIALIZATION COUNT	2	BINARY	7.4.2.2.4
0008h	VOLUME IDENTIFIER	32	ASCII	7.4.2.2.5
0009h	Restricted (see SSC-4)			
000Ah to 0209h	Reserved			

[a] See table 441 in 7.4.1.

Table 443 — Device type attributes (part 2 of 2)

Attribute Identifier	Name	Attribute Length (in bytes)	Format [a]	Reference
020Ah	DEVICE VENDOR/SERIAL NUMBER AT LAST LOAD	40	ASCII	7.4.2.2.6
020Bh	DEVICE VENDOR/SERIAL NUMBER AT LOAD-1	40	ASCII	7.4.2.2.6
020Ch	DEVICE VENDOR/SERIAL NUMBER AT LOAD-2	40	ASCII	7.4.2.2.6
020Dh	DEVICE VENDOR/SERIAL NUMBER AT LOAD-3	40	ASCII	7.4.2.2.6
020Eh to 021Fh	Reserved			
0220h	TOTAL MEBIBYTES WRITTEN IN MEDIUM LIFE	8	BINARY	7.4.2.2.7
0221h	TOTAL MEBIBYTES READ IN MEDIUM LIFE	8	BINARY	7.4.2.2.7
0222h	TOTAL MEBIBYTES WRITTEN IN CURRENT/LAST LOAD	8	BINARY	7.4.2.2.8
0223h	TOTAL MEBIBYTES READ IN CURRENT/LAST LOAD	8	BINARY	7.4.2.2.8
0224h	LOGICAL POSITION OF FIRST ENCRYPTED BLOCK	8	BINARY	7.4.2.2.9
0225h	LOGICAL POSITION OF FIRST UNENCRYPTED BLOCK AFTER THE FIRST ENCRYPTED BLOCK	8	BINARY	7.4.2.2.10
0226h to 033Fh	Reserved			
0340h	MEDIUM USAGE HISTORY	90	BINARY	7.4.2.2.11
0341h	PARTITION USAGE HISTORY	60	BINARY	7.4.2.2.12
0342h to 03FFh	Reserved			

[a] See table 441 in 7.4.1.

7.4.2.2.1 REMAINING CAPACITY IN PARTITION and MAXIMUM CAPACITY IN PARTITION

These attributes are native capacities (i.e., assuming no data compression for the specified medium partition) expressed in units of mebibytes.

7.4.2.2.2 LOAD COUNT

This attribute is a saturating counter that indicates how many times this medium has been fully loaded. This attribute should not be reset to zero by any action of the device server.

7.4.2.2.3 MAM SPACE REMAINING

This attribute indicates the space currently available in the medium auxiliary memory. The total medium auxiliary memory capacity is reported in the MAM CAPACITY attribute (see 7.4.2.3.4).

NOTE 41 - It is not always possible to utilize all of the available space in a given medium auxiliary memory implementation. Depending on the internal organization of the memory and the software that controls it, fragmentation issues have the potential to result in conditions where cause certain attribute sizes to not be fully accommodated as the medium auxiliary memory nears its maximum capacity.

7.4.2.2.4 INITIALIZATION COUNT

This attribute is a saturating counter that indicates the number of times that a device server has logically formatted the medium. This value is cumulative over the life of the medium and shall not be reset to zero.

7.4.2.2.5 VOLUME IDENTIFIER

This attribute indicates the current volume identifier (see SMC-3) of the medium. If the device server supports this attribute but does not have access to the volume identifier, the device server shall report this attribute with an attribute length value of zero.

7.4.2.2.6 DEVICE VENDOR/SERIAL NUMBER AT LAST LOAD, DEVICE VENDOR/SERIAL NUMBER AT LOAD –1, DEVICE VENDOR/SERIAL NUMBER AT LOAD –2 and DEVICE VENDOR/SERIAL NUMBER AT LOAD –3

These attributes give a history of the last four device servers in which the medium has been loaded. The format of these attributes is shown in table 444.

Table 444 — DEVICE VENDOR/SERIAL NUMBER attribute format

Bit / Byte	7	6	5	4	3	2	1	0
0	(MSB)							
•••				T10 VENDOR IDENTIFICATION				
7								(LSB)
8	(MSB)							
•••				PRODUCT SERIAL NUMBER				
39								(LSB)

The T10 VENDOR IDENTIFICATION field shall contain the T10 vendor identification bytes that the device server loading the medium returns in the Standard INQUIRY data (see 6.6.2) parameter data for an INQUIRY command (see 6.6).

The PRODUCT SERIAL NUMBER field contains ASCII data (see 4.3.1) that is a manufacturer assigned serial number. If the product serial number is not available, the PRODUCT SERIAL NUMBER field shall contain ASCII spaces (20h).

7.4.2.2.7 TOTAL MEBIBYTES WRITTEN IN MEDIUM LIFE and TOTAL MEBIBYTES READ IN MEDIUM LIFE

These attributes indicate the total number of data mebibytes that are transferred to or from the medium, after any data compression has been applied, over the entire medium life. These values are cumulative and shall not be reset to zero.

7.4.2.2.8 TOTAL MEBIBYTES WRITTEN IN CURRENT/LAST LOAD and TOTAL MEBIBYTES READ IN CURRENT/LAST LOAD

These attributes indicate the total number of data mebibytes that are transferred to or from the medium, after any data compression has been applied, during:

a) the current load if the medium is currently loaded; or
b) the last load if the medium is currently unloaded.

The device server should reset these attributes to zero when the medium is loaded.

7.4.2.2.9 LOGICAL POSITION OF FIRST ENCRYPTED BLOCK

This attribute indicates the first encrypted logical block, if any, in the partition specified by the PARTITION NUMBER field in the CDB, and shall be set to:

a) FFFF FFFF FFFF FFFFh if there is no encrypted logical block in the partition specified by the PARTITION NUMBER field in the CDB;
b) FFFF FFFF FFFF FFFEh if it is unknown whether there are any encrypted logical blocks in the partition specified by the PARTITION NUMBER field in the CDB; or
c) the address of the first logical block in the partition specified by the PARTITION NUMBER field in the CDB that contains encrypted data.

7.4.2.2.10 LOGICAL POSITION OF FIRST UNENCRYPTED BLOCK AFTER THE FIRST ENCRYPTED BLOCK

If this attribute is supported, the LOGICAL POSITION OF FIRST ENCRYPTED BLOCK (see 7.4.2.2.9) attribute shall be supported. This attribute indicates the first unencrypted logical block, if any, in the partition specified by the PARTITION NUMBER field in the CDB after one or more encrypted logical blocks, and shall be set to:

a) FFFF FFFF FFFF FFFFh if there is:
   A) no encrypted logical block in the partition specified by the PARTITION NUMBER field in the CDB (i.e., if the value for the LOGICAL POSITION OF FIRST ENCRYPTED BLOCK attribute (see 7.4.2.2.9) is set to FFFF FFFF FFFF FFFFh); or
   B) no unencrypted logical block in the partition specified by the PARTITION NUMBER field in the CDB after the address specified in the LOGICAL POSITION OF FIRST ENCRYPTED BLOCK attribute;
b) FFFF FFFF FFFF FFFEh if it is unknown whether:
   A) there are any encrypted logical blocks in the partition specified by the PARTITION NUMBER field in the CDB (i.e., the LOGICAL POSITION OF FIRST ENCRYPTED BLOCK attribute (see 7.4.2.2.9) is set to FFFF FFFF FFFF FFFEh); or
   B) any logical block in the partition specified by the PARTITION NUMBER field in the CDB after the first encrypted logical block contains unencrypted data;

   or
c) the address of the first logical block in the partition specified by the PARTITION NUMBER field in the CDB that contains unencrypted data and is located after the address specified in the LOGICAL POSITION OF FIRST ENCRYPTED BLOCK attribute.

7.4.2.2.11 MEDIUM USAGE HISTORY

This attribute provides saturating counters (see table 445) for the entire medium. The value in each field is the sum for all partitions. If a field is not used, it should be set to zero.

Table 445 — MEDIUM USAGE HISTORY attribute format (part 1 of 2)

Bit / Byte	7	6	5	4	3	2	1	0
0	(MSB)							
•••				CURRENT AMOUNT OF DATA WRITTEN				
5								(LSB)
6	(MSB)							
•••				CURRENT WRITE RETRIES COUNT				
11								(LSB)

Table 445 — MEDIUM USAGE HISTORY attribute format (part 2 of 2)

Bit / Byte	7	6	5	4	3	2	1	0
12	(MSB)							
•••				CURRENT AMOUNT OF DATA READ				
17								(LSB)
18	(MSB)							
•••				CURRENT READ RETRIES COUNT				
23								(LSB)
24	(MSB)							
•••				PREVIOUS AMOUNT OF DATA WRITTEN				
29								(LSB)
30	(MSB)							
•••				PREVIOUS WRITE RETRIES COUNT				
35								(LSB)
36	(MSB)							
•••				PREVIOUS AMOUNT OF DATA READ				
41								(LSB)
42	(MSB)							
•••				PREVIOUS READ RETRIES COUNT				
47								(LSB)
48	(MSB)							
•••				TOTAL AMOUNT OF DATA WRITTEN				
53								(LSB)
54	(MSB)							
•••				TOTAL WRITE RETRIES COUNT				
59								(LSB)
60	(MSB)							
•••				TOTAL AMOUNT OF DATA READ				
65								(LSB)
66	(MSB)							
•••				TOTAL READ RETRIES COUNT				
71								(LSB)
72	(MSB)							
•••				LOAD COUNT				
77								(LSB)
78	(MSB)							
•••				TOTAL CHANGE PARTITION COUNT				
83								(LSB)
84	(MSB)							
•••				TOTAL PARTITION INITIALIZE COUNT				
89								(LSB)

The CURRENT AMOUNT OF DATA WRITTEN field indicates the amount of data written to the medium during this load of the medium. This value is expressed in mebibytes (see 3.5.2).

The CURRENT WRITE RETRIES COUNT field indicates the total number of times a write retry occurred during this load of the medium.[2)]

The CURRENT AMOUNT OF DATA READ field indicates the amount of data read from the medium during this load of the medium. This value is expressed in mebibytes (see 3.5.2).

The CURRENT READ RETRIES COUNT field indicates the number of times a read retry occurred during this load of the medium.[2)]

The PREVIOUS AMOUNT OF DATA WRITTEN field indicates the amount of data written to the medium during the previous load of the medium. This value is expressed in mebibytes (see 3.5.2).

The PREVIOUS WRITE RETRIES COUNT field indicates the total number of times a write retry occurred during the previous load of the medium.[2)]

The PREVIOUS AMOUNT OF DATA READ field indicates the amount of data read from the medium during the previous load of the medium. This value is expressed in mebibytes (see 3.5.2).

The PREVIOUS READ RETRIES COUNT field indicates the number of times a read retry occurred during the previous load of the medium.[2)]

The TOTAL AMOUNT OF DATA WRITTEN field indicates the amount of data written to the medium since the last time the medium was formatted. This value is expressed in mebibytes (see 3.5.2).

The TOTAL WRITE RETRIES COUNT field indicates the total number of times a write retry occurred since the last time the medium was formatted.[2)]

The TOTAL AMOUNT OF DATA READ field indicates the amount of data read from the medium since the last time the medium was formatted. This value is expressed in mebibytes (see 3.5.2).

The TOTAL READ RETRIES COUNT field indicates the number of times a read retry occurred since the last time the medium was formatted.[2)]

The LOAD COUNT field indicates the number of loads since the last time the medium was formatted. This count accumulates over the life of the medium but it is reset to zero after a medium format.

The TOTAL CHANGE PARTITION COUNT field indicates the number of times that switches between partitions have been performed on the medium. This count accumulates over the life of the medium but it is reset to zero after a medium format.

The TOTAL PARTITION INITIALIZE COUNT field indicates number of times that any of the partitions on the medium have been erased. This count accumulates over the life of the medium but it is reset to zero after a medium format.

2) The definition of a retry as counted by this attribute field is outside the scope of this standard. This count should not be used to compare products because the products may define errors differently.

7.4.2.2.12 PARTITION USAGE HISTORY

This attribute provides saturating counters (see table 446) for the partition specified by the PARTITION NUMBER field in the CDB. If a field is not used, it should be set to zero.

Table 446 — PARTITION USAGE HISTORY attribute format (part 1 of 2)

Bit / Byte	7	6	5	4	3	2	1	0
0	(MSB)							
•••				CURRENT AMOUNT OF DATA WRITTEN				
3								(LSB)
4	(MSB)							
•••				CURRENT WRITE RETRIES COUNT				
7								(LSB)
8	(MSB)							
•••				CURRENT AMOUNT OF DATA READ				
11								(LSB)
12	(MSB)							
•••				CURRENT READ RETRIES COUNT				
15								(LSB)
16	(MSB)							
•••				PREVIOUS AMOUNT OF DATA WRITTEN				
19								(LSB)
20	(MSB)							
•••				PREVIOUS WRITE RETRIES COUNT				
23								(LSB)
24	(MSB)							
•••				PREVIOUS AMOUNT OF DATA READ				
27								(LSB)
28	(MSB)							
•••				PREVIOUS READ RETRIES COUNT				
31								(LSB)
32	(MSB)							
•••				TOTAL AMOUNT OF DATA WRITTEN				
35								(LSB)
36	(MSB)							
•••				TOTAL WRITE RETRIES COUNT				
39								(LSB)
40	(MSB)							
•••				TOTAL AMOUNT OF DATA READ				
43								(LSB)
44	(MSB)							
•••				TOTAL READ RETRIES COUNT				
47								(LSB)
48	(MSB)							
•••				LOAD COUNT				
51								(LSB)

Table 446 — PARTITION USAGE HISTORY attribute format (part 2 of 2)

Bit / Byte	7	6	5	4	3	2	1	0
52	(MSB)							
•••				CHANGE PARTITION COUNT				
55								(LSB)
56	(MSB)							
•••				PARTITION INITIALIZE COUNT				
59								(LSB)

The CURRENT AMOUNT OF DATA WRITTEN field indicates the amount of data written to the medium in the partition specified by the PARTITION NUMBER field in the CDB during this load of the medium. This value is expressed in mebibytes (see 3.5.2).

The CURRENT WRITE RETRIES COUNT field indicates the total number of times a write retry occurred in the partition specified by the PARTITION NUMBER field in the CDB during this load of the medium.[2)]

The CURRENT AMOUNT OF DATA READ field indicates the amount of data read from the medium in the partition specified by the PARTITION NUMBER field in the CDB during this load of the medium. This value is expressed mebibytes (see 3.5.2).

The CURRENT READ RETRIES COUNT field indicates the number of times a read retry occurred in the partition specified by the PARTITION NUMBER field in the CDB during this load of the medium.[2)]

The PREVIOUS AMOUNT OF DATA WRITTEN field indicates the amount of data written to the medium in the partition specified by the PARTITION NUMBER field in the CDB during the previous load of the medium. This value is expressed in mebibytes (see 3.5.2).

The PREVIOUS WRITE RETRIES COUNT field indicates the total number of times a write retry occurred in the partition specified by the PARTITION NUMBER field in the CDB during the previous load of the medium.[2)]

The PREVIOUS AMOUNT OF DATA READ field indicates the amount of data read from the medium in the partition specified by the PARTITION NUMBER field in the CDB during the previous load of the medium. This value is expressed in mebibytes (see 3.5.2).

The PREVIOUS READ RETRIES COUNT field indicates the number of times a read retry occurred in the partition specified by the PARTITION NUMBER field in the CDB during the previous load of the medium.[2)]

The TOTAL AMOUNT OF DATA WRITTEN field indicates the amount of data written to the medium in the partition specified by the PARTITION NUMBER field in the CDB since the last time the medium was formatted. This value is expressed in mebibytes (see 3.5.2).

The TOTAL WRITE RETRIES COUNT field indicates the total number of times a write retry occurred in the partition specified by the PARTITION NUMBER field in the CDB since the last time the medium was formatted.[2)]

The TOTAL AMOUNT OF DATA READ field indicates the amount of data read from the medium in the partition specified by the PARTITION NUMBER field in the CDB since the last time the medium was formatted. This value is expressed in mebibytes (see 3.5.2).

The TOTAL READ RETRIES COUNT field indicates the number of times a read retry occurred in the partition specified by the PARTITION NUMBER field in the CDB since the last time the medium was formatted.[2)]

The LOAD COUNT field indicates the number of loads in the partition specified by the PARTITION NUMBER field in the CDB since the last time the medium was formatted. This count accumulates over the life of the medium but it is reset to zero after a medium format.

The TOTAL CHANGE PARTITION COUNT field indicates the number of times that switches to the partition specified by the PARTITION NUMBER field in the CDB have been performed on the medium. This count accumulates over the life of the medium but it is reset to zero after a medium format.

The TOTAL PARTITION INITIALIZE COUNT field indicates number of times that the partition specified by the PARTITION NUMBER field in the CDB has been initialized. This count accumulates over the life of the medium but it is reset to zero after a medium format.

7.4.2.3 Medium type attributes

Medium type attributes (see table 447) are stored in the medium auxiliary memory by the manufacturer. The device server shall not alter medium type attributes. All supported medium type attributes shall have a status of read only or unavailable (see 5.7).

Table 447 — Medium type attributes

Attribute Identifier	Name	Attribute Length (in bytes)	Format [a]	Reference
0400h	MEDIUM MANUFACTURER	8	ASCII	7.4.2.3.1
0401h	MEDIUM SERIAL NUMBER	32	ASCII	7.4.2.3.2
0402h to 0405h	Restricted (see SSC-4)			
0406h	MEDIUM MANUFACTURE DATE	8	ASCII	7.4.2.3.3
0407h	MAM CAPACITY	8	BINARY	7.4.2.3.4
0408h	MEDIUM TYPE	1	BINARY	7.4.2.3.5
0409h	MEDIUM TYPE INFORMATION	2	BINARY	7.4.2.3.5
040Ah	NUMERIC MEDIUM SERIAL NUMBER	unspecified	unspecified	7.4.2.3.6
040Bh to 07FFh	Reserved			

[a] See table 441 in 7.4.1.

7.4.2.3.1 MEDIUM MANUFACTURER

This attribute contains eight bytes of left-aligned ASCII data (see 4.3.1) identifying the manufacturer of the media. The medium manufacturer shall be a T10 vendor identification assigned by INCITS. A list of assigned T10 vendor identifications is in Annex G and on the T10 web site (http://www.T10.org).

NOTE 42 - The T10 web site (http://www.t10.org) provides a convenient means to request an identification code.

7.4.2.3.2 MEDIUM SERIAL NUMBER

This attribute contains the manufacturer's serial number for the medium.

7.4.2.3.3 MEDIUM MANUFACTURE DATE

This attribute contains the date of manufacture of the medium. The format is YYYYMMDD (i.e., four numeric ASCII characters for the year followed by two numeric ASCII characters for the month followed by two numeric ASCII characters for the day with no intervening spaces).

7.4.2.3.4 MAM CAPACITY

This attribute is the total capacity of the medium auxiliary memory, in bytes.

7.4.2.3.5 MEDIUM TYPE and MEDIUM TYPE INFORMATION

These attributes contain information about non-data media and other types of media. The MEDIUM TYPE INFORMATION attribute is interpreted according to the type of medium indicated by the MEDIUM TYPE (see table 448).

Table 448 — MEDIUM TYPE and MEDIUM TYPE INFORMATION attributes

MEDIUM TYPE	Description	MEDIUM TYPE INFORMATION
00h	Data medium	Reserved
01h	Cleaning medium	Maximum number of cleaning cycles permitted
02h to 7Fh	Reserved	Reserved
80h	Write-once medium	Reserved
81h to FFh	Reserved	Reserved

7.4.2.3.6 NUMERIC MEDIUM SERIAL NUMBER

This attribute contains the manufacturer's serial number for the medium in a vendor specific format.

7.4.2.4 Host type attributes

Application clients may use the WRITE ATTRIBUTE command (see 6.48) and READ ATTRIBUTE command (see 6.17) to maintain the attributes shown in table 449. All supported host type attributes shall have a status of nonexistent or read/write (see 5.7).

Table 449 — Host type attributes

Attribute Identifier	Name	Attribute Length (in bytes)	Format [a]	Reference
0800h	APPLICATION VENDOR	8	ASCII	7.4.2.4.1
0801h	APPLICATION NAME	32	ASCII	7.4.2.4.2
0802h	APPLICATION VERSION	8	ASCII	7.4.2.4.3
0803h	USER MEDIUM TEXT LABEL	160	TEXT	7.4.2.4.4
0804h	DATE AND TIME LAST WRITTEN	12	ASCII	7.4.2.4.5
0805h	TEXT LOCALIZATION IDENTIFIER	1	BINARY	7.4.2.4.6
0806h	BARCODE	32	ASCII	7.4.2.4.7
0807h	OWNING HOST TEXTUAL NAME	80	TEXT	7.4.2.4.8
0808h	MEDIA POOL	160	TEXT	7.4.2.4.9
0809h	PARTITION USER TEXT LABEL	16	ASCII	7.4.2.4.10
080Ah	LOAD/UNLOAD AT PARTITION	1	BINARY	7.4.2.4.11
080Bh	APPLICATION FORMAT VERSION	16	ASCII	7.4.2.4.12
080Ch	Restricted (see SSC-4)			
080Dh to BFFh	Reserved			

[a] See table 441 in 7.4.1.

7.4.2.4.1 APPLICATION VENDOR

This attribute identifies the manufacturer of the application client (e.g., class driver or backup program) associated with use of the MAM. The attribute value should be a T10 vendor identification assigned by INCITS. A list of assigned T10 vendor identifications is in Annex G and on the T10 web site (http://www.T10.org). This attribute may provide inaccurate information if the application client does not update it.

NOTE 43 - The T10 web site (http://www.t10.org) provides a convenient means to request an identification code.

7.4.2.4.2 APPLICATION NAME

This attribute contains the name of the application client.

7.4.2.4.3 APPLICATION VERSION

This attribute contains the version of the application client.

7.4.2.4.4 USER MEDIUM TEXT LABEL

This attribute contains a user assigned label for the volume.

7.4.2.4.5 DATE & TIME LAST WRITTEN

This attribute contains the time at which the application client last wrote to the medium auxiliary memory. The format is YYYYMMDDHHMM (i.e., four numeric ASCII characters for the year followed by two numeric ASCII characters for the month followed by two numeric ASCII characters for the day followed by two numeric ASCII characters between 00 and 24 for the hour followed by two numeric ASCII characters for the minute with no intervening spaces).

7.4.2.4.6 TEXT LOCALIZATION IDENTIFIER

This attribute defines the character set (see table 450) used for attributes with a TEXT format (see 7.4.1).

Table 450 — TEXT LOCALIZATION IDENTIFIER attribute values

Value	Meaning
00h	No code specified (ASCII)
01h	ISO/IEC 8859-1 (Europe, Latin America)
02h	ISO/IEC 8859-2 (Eastern Europe)
03h	ISO/IEC 8859-3 (SE Europe/miscellaneous)
04h	ISO/IEC 8859-4 (Scandinavia/Baltic)
05h	ISO/IEC 8859-5 (Cyrillic)
06h	ISO/IEC 8859-6 (Arabic)
07h	ISO/IEC 8859-7 (Greek)
08h	ISO/IEC 8859-8 (Hebrew)
09h	ISO/IEC 8859-9 (Latin 5)
0Ah	ISO/IEC 8859-10 (Latin 6)
0Bh to 7Fh	Reserved
80h	ISO/IEC 10646 (UTF-16BE)
81h	ISO/IEC 10646 (UTF-8)
82h to FFh	Reserved

7.4.2.4.7 BARCODE

This attribute contains the barcode associated with the medium described by the MAM.

7.4.2.4.8 OWNING HOST TEXTUAL NAME

This attribute indicates the host from which that USER MEDIUM TEXT LABEL (see 7.4.2.4.4) originates.

7.4.2.4.9 MEDIA POOL

This attribute indicates the media pool to which this medium belongs.

7.4.2.4.10 PARTITION USER TEXT LABEL

This attribute is a user assigned identifier for the partition specified by the PARTITION NUMBER field in the CDB.

7.4.2.4.11 LOAD/UNLOAD AT PARTITION

This attribute indicates whether the media is capable of being loaded or unloaded at the partition specified by the PARTITION NUMBER field in the CDB. If loads and unloads are enabled for the specified partition, the value of this attribute shall be one. If loads and unloads are not enabled for the specified partition, the value of this attribute shall be zero. All attribute values other than zero and one are reserved. If LOAD/UNLOAD AT

PARTITION is disabled, then loads and unloads are performed at the beginning of the media instead of at the specified partition. If this attribute is in the nonexistent state (see 5.7), then the default action shall be to load and unload at the beginning of media.

7.4.2.4.12 APPLICATION FORMAT VERSION

This attribute indicates the version of the format being used by the application that set this attribute.

7.5 Mode parameters

7.5.1 Summary of mode page codes

The page code assignments for mode pages are summarized in table 451.

Table 451 — Summary of mode page codes

Mode page name	Page code	Subpage code	Reference
Control	0Ah	n/a [a]	7.5.8
Control Extension	0Ah	01h	7.5.9
Disconnect-Reconnect	02h	n/a [a]	7.5.10
Extended	15h	01h to FEh	7.5.11
Extended Device-Type Specific	16h	01h to FEh	7.5.12
Power Condition	1Ah	n/a [a]	7.5.13
Power Consumption	1Ah	01h	7.5.14
Protocol Specific Logical Unit	18h	n/a [a]	7.5.15
Protocol Specific Port	19h	n/a [a]	7.5.16
Restricted (see applicable protocol standard)	18h	01h to FEh	
	19h	01h to FEh	
Restricted (see applicable command standard)	01h	00h to FEh	
	03h to 08h	00h to FEh	
	0Ah	F0h to FEh	
	0Bh to 14h	00h to FEh	
	1Ah	F0h to FEh	
	1Bh to 1Fh	00h to FEh	
	20h to 3Eh	00h to FEh	
Return all pages	3Fh [b]	00h [b]	6.13 and 6.14
Return all pages and subpages	3Fh [b]	FFh [b]	6.13 and 6.14
Return all subpages	00h to 3Eh [b]	FFh [b]	6.13 and 6.14
Obsolete [c]			
Vendor specific [d]			
Reserved	All other codes		

A numeric ordered listing of mode page and subpage codes is provided in F.6.

[a] Applicable only in the MODE SENSE command (see table 201 in 6.13).
[b] The use of these page codes and subpage codes is described in table 201 (see 6.13).
[c] The following page codes are obsolete: 09h.
[d] Page code 00h is vendor specific and does not require a page format or a subpage code.

7.5.2 Mode page policies

Logical units shall share mode parameter header and block descriptor values across all I_T nexuses. I_T nexus loss shall not affect mode parameter header, block descriptor, and mode page values.

Each logical unit shall maintain current values and saved values of each mode page based on any of the policies listed in table 452. The mode page policy used for each mode page may be reported in the Mode Page Policy VPD page (see 7.8.9).

Table 452 — Mode page policies

Mode page policy	Number of mode page copies
Shared	One copy of the mode page that is shared by all I_T nexuses.
Per target port	A separate copy of the mode page for each target port with each copy shared by all initiator ports.
Per I_T nexus	A separate copy of the mode page for each I_T nexus

After a logical unit reset, each mode parameter header, block descriptor, and mode page shall revert to saved values if supported or default values if saved values are not supported.

7.5.3 Mode parameters overview

This subclause describes the mode parameter headers, block descriptors, and mode pages used with MODE SELECT command (see 6.11 and 6.12) and MODE SENSE command (see 6.13 and 6.14) that are applicable to all SCSI devices. Subpages are identical to mode pages except that they include a SUBPAGE CODE field that further differentiates the mode page contents. Mode pages specific to each device type are described in the command standard that applies to that device type.

7.5.4 Mode parameter list format

The mode parameter list shown in table 453 contains a header, followed by zero or more block descriptors, followed by zero or more variable length mode pages. Parameter lists are defined for each device type.

Table 453 — Mode parameter list

Bit Byte	7	6	5	4	3	2	1	0
	Mode parameter header							
	Block descriptor(s)							
	Mode page(s) or vendor specific (e.g., page code set to zero)							

7.5.5 Mode parameter header formats

The mode parameter header that is used by the MODE SELECT(6) command (see 6.11) and the MODE SENSE(6) command (see 6.13) is defined in table 454.

Table 454 — 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 mode parameter header that is used by the MODE SELECT(10) command (see 6.12) and the MODE SENSE(10) command (see 6.14) is defined in table 455.

Table 455 — 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 indicates the number of bytes that follow in the mode parameter list. When using the MODE SELECT command, the MODE DATA LENGTH field is reserved.

NOTE 44 - Logical units that support more than 256 bytes of block descriptors and mode pages should implement ten-byte mode commands. The MODE DATA LENGTH field in the six-byte CDB header limits the transferred 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 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 for definition of this field.

If the Long LBA (LONGLBA) bit is set to zero, the mode parameter block descriptor(s), if any, are each eight bytes long and have the format described in 7.5.6.1. If the LONGLBA bit is set to one, the mode parameter block descriptor(s), if any, are each 16 bytes long and have a format described in a command standard.

The BLOCK DESCRIPTOR LENGTH field indicates 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 the length of mode pages or vendor specific parameters (e.g., page code set to zero), 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.

7.5.6 Mode parameter block descriptor formats

7.5.6.1 General block descriptor format

If the LONGLBA bit is set to zero (see 7.5.5), the mode parameter block descriptor format for all device types except direct access block devices (see SBC-3) is shown in table 456.

Table 456 — General mode parameter block descriptor

Bit / Byte	7	6	5	4	3	2	1	0
0	DENSITY CODE							
1	(MSB)			NUMBER OF BLOCKS				
•••								
3								(LSB)
4	Reserved							
5	(MSB)			BLOCK LENGTH				
•••								
7								(LSB)

Block descriptors specify some of the medium characteristics for all or part of a logical unit. Each block descriptor, if any, 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). Whenever any block descriptor values are changed, the device server shall establish a unit attention condition (see SAM-5) for the initiator port associated with every I_T nexus except the I_T nexus on which the MODE SELECT command (see 6.11) was received, with the additional sense code set to MODE PARAMETERS CHANGED. Command standards may place additional requirements on the general mode parameter block descriptor. Requirements in the command standards that conflict with requirements defined in this subclause shall take precedence over the requirements defined in this subclause.

The DENSITY CODE field is unique for each device type. Refer to the mode parameters subclause of the specific device type command standard for definition of this field.

The NUMBER OF BLOCKS field specifies the number of logical blocks on the medium to which the DENSITY CODE field and BLOCK LENGTH field 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 number of logical blocks on the medium exceeds the maximum value that may be specified in the NUMBER OF BLOCKS field, then a value of FF FFFFh indicates that all of the remaining logical blocks of the logical unit shall have the medium characteristics specified.

NOTE 45 - There may be implicit association between parameters defined in the mode pages and block descriptors. In this case, the device server may change parameters not explicitly sent with the MODE SELECT command. A subsequent MODE SENSE command may be used to detect these changes.

NOTE 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-4).

7.5.7 Mode page and subpage formats and page codes

The page_0 mode page format is defined in table 457.

Table 457 — Page_0 mode page format

Bit / Byte	7	6	5	4	3	2	1	0
0	PS	SPF (0b)	PAGE CODE					
1	PAGE LENGTH (n-1)							
2	Mode parameters							
•••								
n								

The sub_page mode page format is defined in table 458.

Table 458 — Sub_page mode page format

Bit / Byte	7	6	5	4	3	2	1	0
0	PS	SPF (1b)	PAGE CODE					
1	SUBPAGE CODE							
2	(MSB) PAGE LENGTH (n-3)							
3								(LSB)
4	Mode parameters							
•••								
n								

Each mode page contains a PS bit, an SPF bit, 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 for the specific device type. Each mode page with a SPF bit set to one contains a SUBPAGE CODE field.

A SubPage Format (SPF) bit set to zero indicates that the page_0 mode page format is being used. A SPF bit set to one indicates that the sub_page mode page format is being used.

When using the MODE SENSE command, a parameters saveable (PS) bit set to one indicates that the mode page may be saved by the logical unit in a nonvolatile, vendor specific location. A PS bit set to zero indicates that the device server is not able to save the supported parameters. When using the MODE SELECT command, the PS bit is reserved.

The PAGE CODE field and SUBPAGE CODE field (see 7.5.1) identify the format and parameters defined for that mode page. Page codes are defined as applying to all device types or as applying to a specific device type. The page codes that apply to a specific device type are defined in the command standard for that device type. The applicability of each subpage code matches that of the page code with which the subpage code is associated.

The device server shall use the page_0 mode page format only for mode pages associated with subpage code 00h (i.e., mode pages for which table 201 (see 6.13.1) requires the use of page_0 mode page format).

When using the MODE SENSE command, if page code 00h (i.e., vendor specific mode page) is implemented, the device server shall return that mode page last in response to a request to return all mode pages (e.g., a MODE SENSE command (see 6.13 or 6.14) with the PAGE CODE field set to 3Fh). When using the MODE SELECT command, this mode page should be sent last.

The PAGE LENGTH field specifies the length in bytes of the mode parameters that follow. If the parameter list in a MODE SELECT command (see 6.11 or 6.12) does not set the PAGE LENGTH field to the value that is returned for the mode page by a MODE SENSE command, then the device server shall terminate the MODE SELECT command with CHECK CONDITION status, with the sense key set to ILLEGAL REQUEST, and the additional sense code set to INVALID FIELD IN PARAMETER LIST. The logical unit may implement a mode page that is less than the full mode page length defined, provided no field is truncated and the PAGE LENGTH field correctly specifies the actual length implemented.

The mode parameters for each mode page are defined in the following subclauses or in the mode parameters subclause in the command standard for the specific device type. Mode parameters not implemented by the logical unit shall be set to zero.

7.5.8 Control mode page

The Control mode page (see table 459) provides controls over SCSI features that are applicable to all device types (e.g., task set management and error logging). If a field in this mode page is changed while there is a command already in the task set, then whether the old or new value of the field applies to that command is outside the scope of this standard. The mode page policy (see 7.5.2) for this mode page shall be shared or per I_T nexus.

Table 459 — Control mode page

Bit / Byte	7	6	5	4	3	2	1	0
0	PS	SPF (0b)	PAGE CODE (0Ah)					
1	PAGE LENGTH (0Ah)							
2	TST			TMF_ONLY	DPICZ	D_SENSE	GLTSD	RLEC
3	QUEUE ALGORITHM MODIFIER				NUAR	QERR		Obsolete
4	VS	RAC	UA_INTLCK_CTRL		SWP	Obsolete		
5	ATO	TAS	ATMPE	RWWP	Reserved	AUTOLOAD MODE		
6	Obsolete							
7								
8	(MSB)	BUSY TIMEOUT PERIOD						
9								(LSB)
10	(MSB)	EXTENDED SELF-TEST COMPLETION TIME						
11								(LSB)

The PS bit, SPF bit, PAGE CODE field, and PAGE LENGTH field are described in 7.5.7.

The SPF bit, PAGE CODE field, and PAGE LENGTH field shall be set as shown in table 459 for the Control mode page.

A task set type (TST) field (see table 460) specifies the type of task set in the logical unit.

Table 460 — Task set type (TST) field

Code	Description
000b	The logical unit maintains one task set for all I_T nexuses
001b	The logical unit maintains separate task sets for each I_T nexus
010b to 111b	Reserved

Regardless of the mode page policy (see 7.5.2) for the Control mode page, the shared mode page policy shall be applied to the TST field. If the most recent MODE SELECT changes the setting of this field, then the device server shall establish a unit attention condition (see SAM-5) for the initiator port associated with every I_T nexus except the I_T nexus on which the MODE SELECT command was received, with the additional sense code set to MODE PARAMETERS CHANGED.

The allow task management functions only (TMF_ONLY) bit set to zero specifies that the device server shall process commands with the ACA task attribute received on the faulted I_T nexus while an ACA condition is established (see SAM-5). A TMF_ONLY bit set to one specifies that the device server shall complete all commands received on the faulted I_T nexus with an ACA ACTIVE status while an ACA condition is established.

A disable protection information check if protect field is zero (DPICZ) bit set to zero indicates that checking of protection information bytes is enabled. A DPICZ bit set to one indicates that checking of protection information is disabled on commands with:

a) the RDPROTECT field (see SBC-3) set to zero;
b) the VRPROTECT field (see SBC-3) set to zero; or
c) the ORPROTECT field (see SBC-3) set to zero.

A descriptor format sense data (D_SENSE) bit set to zero specifies that the device server shall use fixed format sense data (see 4.5.3) when including sense data in the same I_T_L_Q nexus transaction as the status. A D_SENSE bit set to one specifies that the device server shall use descriptor format sense data (see 4.5.2) when including sense data in the same I_T_L_Q nexus transaction as the status, except as defined in 4.5.1. If an application client enables reporting of deferrals in sense data (see SBC-3), then the application client should be able to handle up to 252 bytes of sense data.

A global logging target save disable (GLTSD) bit set to zero specifies that the logical unit implicitly saves, at vendor specific intervals, each log parameter in which the TSD bit (see 7.3) is set to zero. A GLTSD bit set to one specifies that the logical unit shall not implicitly save any log parameters.

A report log exception condition (RLEC) bit set to one specifies that the device server shall report log exception conditions as described in 7.3. A RLEC bit set to zero specifies that the device server shall not report log exception conditions.

The QUEUE ALGORITHM MODIFIER field (see table 461) specifies restrictions on the algorithm used for reordering commands having the SIMPLE task attribute (see SAM-5).

Table 461 — QUEUE ALGORITHM MODIFIER field

Code	Description
0h	**Restricted reordering:** The device server shall order the processing sequence of commands having the SIMPLE task attribute such that data integrity is maintained for that I_T nexus (i.e., if the transmission of new SCSI transport protocol requests is halted at any time, the final value of all data observable on the medium shall be the same as if all the commands had been processed with the ORDERED task attribute).
1h	**Unrestricted reordering allowed:** The device server may reorder the processing sequence of commands having the SIMPLE task attribute in any manner. Any data integrity exposures related to command sequence order shall be explicitly handled by the application client through the selection of appropriate commands and task attributes.
2h to 7h	Reserved
8h to Fh	Vendor specific

A no unit attention on release (NUAR) bit set to one specifies that the device server shall not establish a unit attention condition as described in 5.12.11.2.2. A NUAR bit set to zero specifies that the device server shall establish a unit attention condition as described in 5.12.11.2.2.

The queue error management (QERR) field (see table 462) specifies how the device server shall handle other commands as the result of one command being terminated with CHECK CONDITION status (see SAM-5). The task set type (see the TST field definition in this subclause) defines which other commands are affected. If the TST field equals 000b, then all commands from all I_T nexuses are affected. If the TST field equals 001b, then only commands from the same I_T nexus as the command that is terminated with CHECK CONDITION status are affected.

Table 462 — Queue error management (QERR) field

Code	Definition
00b	If an ACA condition is established, the affected commands in the task set shall resume after the ACA condition is cleared (see SAM-5). Otherwise, all commands other than the command terminated with CHECK CONDITION status shall be processed as if no error occurred.
01b	All the affected commands in the task set shall be aborted when the CHECK CONDITION status is returned. If the TAS bit is set to zero, the device server shall establish a unit attention condition (see SAM-5) for the initiator port associated with every I_T nexus that had commands aborted except for the I_T nexus on which the command was terminated with CHECK CONDITION status, with the additional sense code set to COMMANDS CLEARED BY ANOTHER INITIATOR. If the TAS bit is set to one, all affected commands in the task set for I_T nexuses other than the I_T nexus on which the command was terminated with CHECK CONDITION status shall be completed with TASK ABORTED status and no unit attention shall be established. For the I_T nexus on which the command was terminated with CHECK CONDITION status, no status shall be returned for the commands that are aborted.
10b	Reserved
11b	Affected commands in the task set belonging to the I_T nexus on which a command was terminated with CHECK CONDITION status shall be aborted as part of processing the command termination.

The report a check (RAC) bit provides control of reporting long busy conditions or CHECK CONDITION status. A RAC bit set to one specifies that the device server should return CHECK CONDITION status rather than returning BUSY status if the reason for returning BUSY status may persist for a longer time than that specified

by the BUSY TIMEOUT PERIOD field. A RAC bit set to zero specifies that the device server may return BUSY status regardless of the length of time the reason for returning BUSY status may persist.

The unit attention interlocks control (UA_INTLCK_CTRL) field (see table 463) controls the clearing of unit attention conditions reported in the same I_T_L_Q nexus transaction as a CHECK CONDITION status and whether returning a status of BUSY, TASK SET FULL or RESERVATION CONFLICT results in the establishment of a unit attention condition (see SAM-5).

Table 463 — Unit attention interlocks control (UA_INTLCK_CTRL) field

Code	Definition
00b	The logical unit shall clear any unit attention condition reported in the same I_T_L_Q nexus transaction as a CHECK CONDITION status and shall not establish a unit attention condition for a command that is completed with BUSY status, TASK SET FULL status, or RESERVATION CONFLICT status.
01b	Reserved
10b	The logical unit shall not clear any unit attention condition reported in the same I_T_L_Q nexus transaction as a CHECK CONDITION status and shall not establish a unit attention condition for a command that is completed with BUSY status, TASK SET FULL status, or RESERVATION CONFLICT status.
11b	The logical unit shall not clear any unit attention condition reported in the same I_T_L_Q nexus transaction as a CHECK CONDITION status and shall establish a unit attention condition for the initiator port associated with the I_T nexus on which the BUSY status, TASK SET FULL status, or RESERVATION CONFLICT status is being returned. Depending on the status, the additional sense code shall be set to PREVIOUS BUSY STATUS, PREVIOUS TASK SET FULL STATUS, or PREVIOUS RESERVATION CONFLICT STATUS. Until it is cleared by a REQUEST SENSE command, a unit attention condition shall be established only once for a BUSY status, TASK SET FULL status, or RESERVATION CONFLICT status regardless to the number of commands completed with one of those status codes.
NOTE – The requirements for REQUEST SENSE processing of unit attention interlocks are defined in SAM -5.	

A software write protect (SWP) bit set to one specifies that the logical unit shall inhibit writing to the medium. If the SWP bit is changed from zero to one, any cached or buffered write data shall be written to the medium before enforcing the write protected condition. If the SWP bit is set to one, the device server shall terminate all commands that require writes to the medium with CHECK CONDITION status, with the sense key set to DATA PROTECT, and the additional sense code set to WRITE PROTECTED. If the SWP bit is set to zero, specifies logical unit may allow writing to the medium depending on other write inhibit mechanisms implemented by the logical unit.

If the device type's command standard defines a write protect (WP) bit in the DEVICE-SPECIFIC PARAMETER field in the mode parameter header, then:

a) if the SWP bit is set to one, the WP bit shall be set to one for subsequent MODE SENSE commands; and
b) if the SWP bit is set to zero, the value of the WP bit is device type specific.

For a list of commands affected by the SWP bit and details of the WP bit see the command standard for the specific device type.

An application tag owner (ATO) bit set to zero specifies that the device server may modify the contents of the LOGICAL BLOCK APPLICATION TAG field and, depending on the protection type, may modify the contents of the LOGICAL BLOCK REFERENCE TAG field (see SBC-3). If the ATO bit is set to one the device server shall not modify the LOGICAL BLOCK APPLICATION TAG field and, depending on the protection type, shall not modify the contents of the LOGICAL BLOCK REFERENCE TAG field.

A task aborted status (TAS) bit set to zero specifies that aborted commands shall be terminated by the device server without any response to the application client. A TAS bit set to one specifies that commands aborted by the actions of an I_T nexus other than the I_T nexus on which the command was received shall be completed with TASK ABORTED status (see SAM-5).

An application tag mode page enabled (ATMPE) bit set to zero specifies that use of the Application Tag mode page (see SBC-3) is disabled and the contents of logical block application tags are not defined by the Application Tag mode page. An ATMPE bit set to one specifies that use of the Application Tag mode page is enabled.

If:

a) the ATMPE bit is set to one;
b) the ATO bit is set to one;
c) the value in the DPICZ bit allows protection information checking for the specified command; and
d) the APP_CHK bit is set to one in the Extended INQUIRY VPD page (see 7.8.7),

then knowledge of the value of the Application Tag shall come from the values in the Application Tag mode page as specified by the DPICZ bit.

A reject write without protection (RWWP) bit set to zero specifies that the device server shall process write commands that are specified to include user data without protection information (e.g., a WRITE(10) command with the WRPROTECT field set to 000b (see SBC-3)). A RWWP bit set to one specifies that the device server in a logical unit that has been formatted with protection information shall terminate with CHECK CONDITION status with the sense key set to ILLEGAL REQUEST and the additional sense code set to INVALID FIELD IN CDB any write command that is specified to include user data without protection information.

The AUTOLOAD MODE field specifies the action to be taken by a removable medium device server at the time when a medium is inserted. For devices other than removable medium devices, this field is reserved. Table 464 shows the usage of the AUTOLOAD MODE field.

Table 464 — AUTOLOAD MODE field

Code	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 to 111b	Reserved

The BUSY TIMEOUT PERIOD field specifies the maximum time, in 100 milliseconds increments, that the application client allows for the device server to return BUSY status for unanticipated conditions that are not a routine part of commands from the application client. This value may be rounded down as defined in 5.8. A 0000h value in this field is undefined by this standard. An FFFFh value in this field is defined as an unlimited time.

The EXTENDED SELF-TEST COMPLETION TIME field specifies advisory data that is the time in seconds that the device server requires to complete an extended self-test provided the device server is not interrupted by subsequent commands and no errors occur during processing 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 the processing of the self-test. Device servers supporting SELF-TEST CODE field values other than 000b for the SEND DIAGNOSTIC command (see 6.42) shall support the EXTENDED SELF-TEST COMPLETION TIME field. The EXTENDED SELF-TEST COMPLETION TIME field is not changeable. A value of FFFFh indicates that the extended self-test takes 65 535 seconds or longer. If the value is FFFFh, then refer to the EXTENDED SELF-TEST COMPLETION MINUTES field in the Extended INQUIRY Data VPD page (see 7.8.7).

7.5.9 Control Extension mode page

The Control Extension mode page (see table 465) provides controls over SCSI features that are applicable to all device types. The mode page policy (see 7.5.2) for this mode page shall be shared. If a field in this mode page is changed while there is a command already in the task set, then whether the old or new value of the field applies to that command is outside the scope of this standard.

Table 465 — Control Extension mode page

Bit / Byte	7	6	5	4	3	2	1	0
0	PS	SPF (1b)	PAGE CODE (0Ah)					
1	SUBPAGE CODE (01h)							
2	(MSB)							
3	PAGE LENGTH (001Ch)							(LSB)
4	Reserved					TCMOS	SCSIP	IALUAE
5	Reserved				INITIAL COMMAND PRIORITY			
6	MAXIMUM SENSE DATA LENGTH							
7								
•••	Reserved							
31								

The PS bit, SPF bit, PAGE CODE field, SUBPAGE CODE field, and PAGE LENGTH field are described in 7.5.7.

The SPF bit, PAGE CODE field, SUBPAGE CODE field, and PAGE LENGTH field shall be set as shown in table 465 for the Control Extension mode page.

A timestamp changeable by methods outside this standard (TCMOS) bit set to one specifies that a device clock (see 5.2) may be initialized by methods outside the scope of this standard. A TCMOS bit set to zero specifies that a device clock shall not be initialized by any method except the ones defined by this standard.

A SCSI precedence (SCSIP) bit set to one specifies that device clock initialization (see 5.2) specified by a SET TIMESTAMP command (see 6.46) shall take precedence over methods outside the scope of this standard. A SCSIP bit set to zero specifies that methods outside this standard may initialize a device clock and that the SET TIMESTAMP command shall be terminated as described in 6.46.

An implicit asymmetric logical unit access enabled (IALUAE) bit set to one specifies that implicitly managed transitions between primary target port asymmetric access states (see 5.15.2) are allowed. An IALUAE bit set to zero specifies that implicitly managed transitions between primary target port asymmetric access states are disallowed and indicates that implicitly managed transitions between primary target port asymmetric access states are disallowed or not supported.

The INITIAL COMMAND PRIORITY field specifies the priority that may be used as the command priority (see SAM-5) for commands received by the logical unit on any I_T nexus (i.e., on any I_T_L nexus) where a priority has not been modified by a SET PRIORITY command (see 6.44). If a MODE SELECT command specifies an initial command priority value that is different than the current initial command priority, then the device server shall set any priorities that have not be set with a SET PRIORITY command to a value different than the new initial command priority value to the new priority. The device server shall establish a unit attention condition for the initiator port associated with every I_T_L nexus that receives a new priority, with the additional sense code set to PRIORITY CHANGED.

The MAXIMUM SENSE DATA LENGTH field specifies the maximum number of bytes of sense data the device server shall include in the same I_T_L_Q nexus transaction as the status. A MAXIMUM SENSE DATA LENGTH field

set to zero specifies that there is no limit. The device server shall not include more sense data bytes in the same I_T_L_Q nexus transaction as the status than the smaller of the length indicated by:

a) the MAXIMUM SENSE DATA LENGTH field; and
b) the MAXIMUM SUPPORTED SENSE DATA LENGTH field in the Extended INQUIRY VPD page (see 7.8.7).

7.5.10 Disconnect-Reconnect mode page

NOTE 47 - The name for this mode page, disconnect-reconnect, comes from the SCSI parallel interface.

The Disconnect-Reconnect mode page (see table 466) provides the application client the means to tune the performance of a service delivery subsystem. The mode page policy (see 7.5.2) for this mode page shall be shared or per target port. If the SCSI target device contains more than one target port, the mode page policy should be per target port.

The Disconnect-Reconnect mode page controls parameters that affect one or more target ports. The parameters that may be implemented are defined in the SCSI transport protocol standard for the target port. The MLUS bit (see 7.8.9) shall be set to one in the mode page policy descriptor for this mode page.

The parameters for a target port affect its behavior regardless of which initiator port is forming an I_T nexus with the target port. The parameters may be accessed by MODE SENSE (see 6.13) and MODE SELECT (see 6.11) commands directed to any logical unit accessible through the target port. If a parameter value is changed, all the device servers for all logical units accessible through the target port shall establish a unit attention condition for the initiator port associated with every I_T nexus that includes the target port except the I_T nexus on which the MODE SELECT command was received, with the additional sense code set to MODE PARAMETERS CHANGED.

If a parameter that is not appropriate for the specific SCSI transport protocol implemented by the target port is non-zero, then the device server shall terminate the command with CHECK CONDITION status, with the sense key set to ILLEGAL REQUEST, and the additional sense code set to INVALID FIELD IN PARAMETER LIST.

An interconnect tenancy is a period of time during which a given pair of SCSI ports (i.e., an initiator port and a target port) are accessing the interconnect layer to communicate with each other (e.g., on arbitrated interconnects, a tenancy typically begins when a SCSI port successfully arbitrates for the interconnect and ends when the SCSI port releases the interconnect for use by other devices). Data and other information transfers take place during interconnect tenancies.

Table 466 — Disconnect-Reconnect mode page

Bit / Byte	7	6	5	4	3	2	1	0
0	PS	SPF (0b)	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 PS bit, SPF bit, PAGE CODE field, and PAGE LENGTH field are described in 7.5.7.

The SPF bit, PAGE CODE field, and PAGE LENGTH field shall be set as shown in table 466 for the Disconnect-Reconnect mode page.

The BUFFER FULL RATIO field specifies to the target port how full the buffer should be during read operations prior to requesting an interconnect tenancy. Target ports that do not implement the requested ratio should round down to the nearest implemented ratio as defined in 5.8.

The BUFFER EMPTY RATIO field specifies to the target port how empty the buffer should be during write operations prior to requesting an interconnect tenancy. Target ports that do not implement the requested ratio should round down to the nearest implemented ratio as defined in 5.8.

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 port determines when to request an interconnect tenancy consistent with the disconnect time limit parameter. These parameters are advisory to the target port.

EXAMPLE – Consider a target port 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 target port should request an interconnect tenancy whenever two or more buffers are full.

The BUS INACTIVITY LIMIT field specifies the maximum time that the target port is permitted to maintain an interconnect tenancy without data or information transfer. If the bus inactivity limit is exceeded, then the target port shall conclude the interconnect tenancy, within the restrictions placed on it by the applicable SCSI transport 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 5.8. A value of zero specifies that there is no bus inactivity limit. Different SCSI transport protocols define different units of measure for the bus inactivity limit.

The DISCONNECT TIME LIMIT field specifies the minimum time that the target port shall wait between interconnect tenancies. This value may be rounded as defined in 5.8. A value of zero specifies that there is no disconnect time limit. Different SCSI transport protocols define different units of measure for the disconnect time limit.

The CONNECT TIME LIMIT field specifies the maximum duration of a single interconnect tenancy. If the connect time limit is exceeded, then the target port shall conclude the interconnect tenancy, within the restrictions placed on it by the applicable SCSI transport 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 5.8. A value of zero specifies that there is no connect time limit. Different SCSI transport protocols define different units of measure for the connect time limit.

The MAXIMUM BURST SIZE field specifies the maximum amount of data that the target port shall transfer during a single data transfer operation. This value is expressed in increments of 512 bytes (i.e., a value of one means 512 bytes, two means 1 024 bytes, etc.). The relationship, if any, between data transfer operations and interconnect tenancies is defined in the individual SCSI transport protocol standards. A value of zero specifies there is no limit on the amount of data transferred per data transfer operation.

In terms of the SCSI transport protocol services (see SAM-5), the device server shall limit the Request Byte Count argument to the **Receive Data-Out** protocol service and the **Send Data-In** protocol service to the amount specified in the MAXIMUM BURST SIZE field.

The enable modify data pointers (EMDP) bit specifies whether or not the target port may transfer data out of order. If the EMDP bit is set to zero, the target port shall not transfer data out of order. If the EMDP bit is set to one, the target port is allowed to transfer data out of order.

The FAIR ARBITRATION field specifies whether the target port should use fair or unfair arbitration when requesting an interconnect tenancy. The field may be used to specify different fairness methods as defined in the individual SCSI transport protocol standards.

A disconnect immediate (DIMM) bit set to zero specifies that the target port may transfer data for a command during the same interconnect tenancy in which the SCSI target device receives the command. Whether or not the target port does so may depend upon the target port's internal algorithms, the rules of the applicable SCSI transport protocol, and settings of the other parameters in this mode page. A disconnect immediate (DIMM) bit set to one specifies that the target port shall not transfer data for a command during the same interconnect tenancy in which the SCSI target device receives the command.

The data transfer disconnect control (DTDC) field (see table 467) 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 467 — Data transfer disconnect control (DTDC) field

Code	Description
000b	Data transfer disconnect control is not used. Interconnect tenancies are controlled by other fields in this mode page.
001b	All data for a command shall be transferred within a single interconnect tenancy.

Table 467 — Data transfer disconnect control (DTDC) field

Code	Description
010b	Reserved
011b	All data and the response for a command shall be transferred within a single interconnect tenancy.
100b to 111b	Reserved

The FIRST BURST SIZE field specifies the maximum amount of data that may be transferred to the target port for a command along with the command (i.e., the first burst). This value is expressed in increments of 512 bytes (i.e., a value of one means 512 bytes, two means 1 024 bytes, etc.). The meaning of a value of zero is SCSI transport protocol specific. SCSI transport protocols supporting this field shall provide an additional mechanism to enable and disable the first burst function.

In terms of the SCSI transport protocol services (see SAM-5), the **Receive Data-Out** protocol service shall transfer the first FIRST BURST SIZE amount of data during the first burst.

7.5.11 Extended mode page

The Extended mode page (see table 468) provides a means to specify subpages that are defined for all device types. Subpage code 00h is reserved. All Extended mode pages use the sub_page mode page format.

Table 468 — Extended mode page

Bit Byte	7	6	5	4	3	2	1	0
0	PS	SPF (1b)	PAGE CODE (15h)					
1	SUBPAGE CODE							
2	(MSB)							
3	PAGE LENGTH (n-3)							(LSB)
4	Subpage specific mode parameters							
•••								
n								

The PS bit, SPF bit, PAGE CODE field, and PAGE LENGTH field are described in 7.5.7.

The SPF bit and PAGE CODE field shall be set as shown in table 468 for the Extended mode page.

7.5.12 Extended Device-Type Specific mode page

The Extended Device-Type Specific mode page (see table 469) provides a means to specify subpages that are defined differently for each device type. Subpage code 00h is reserved in the MODE SENSE command

(see 6.13.1). All Extended Device-Type Specific mode pages use the sub_page mode page format.

Table 469 — Extended Device-Type Specific mode page

Bit / Byte	7	6	5	4	3	2	1	0
0	PS	SPF (1b)	PAGE CODE (16h)					
1	SUBPAGE CODE							
2	(MSB)		PAGE LENGTH (n-3)					
3								(LSB)
4			Subpage specific mode parameters					
•••								
n								

The PS bit, SPF bit, PAGE CODE field, and PAGE LENGTH field are described in 7.5.7.

The SPF bit and PAGE CODE field shall be set as shown in table 469 for the Extended Device-Type Specific mode page.

7.5.13 Power Condition mode page

The Power Condition mode page provides an application client with a method to control the power condition of a logical unit (see 5.11).

The mode page policy (see 7.5.2) for this mode page shall be shared.

The logical unit shall use the values in the Power Condition mode page to control its power condition after a power on or a hard reset until a START STOP UNIT command (see SBC-3) setting a power condition is received.

Table 470 defines the Power Condition mode page.

Table 470 — Power Condition mode page

Bit / Byte	7	6	5	4	3	2	1	0
0	PS	SPF (0b)	PAGE CODE (1Ah)					
1	PAGE LENGTH (26h)							
2	PM_BG_PRECEDENCE		Reserved					STANDBY_Y
3	Reserved				IDLE_C	IDLE_B	IDLE_A	STANDBY_Z
4	(MSB)							
•••	IDLE_A CONDITION TIMER							
7								(LSB)
8	(MSB)							
•••	STANDBY_Z CONDITION TIMER							
11								(LSB)
12	(MSB)							
•••	IDLE_B CONDITION TIMER							
15								(LSB)
16	(MSB)							
•••	IDLE_C CONDITION TIMER							
19								(LSB)
20	(MSB)							
•••	STANDBY_Y CONDITION TIMER							
23								(LSB)
24								
•••	Reserved							
38								
39	CCF IDLE		CCF STANDBY		CCF STOPPED		Reserved	

The PS bit, SPF bit, PAGE CODE field, and PAGE LENGTH field are described in 7.5.7.

The SPF bit, PAGE CODE field, and PAGE LENGTH field shall be set as shown in table 470 for the Power Condition mode page.

The PM_BG_PRECEDENCE field (see table 471) specifies the interactions between background functions and power management.

Table 471 — PM_BG_PRECEDENCE field

Code	Description
00b	Vendor specific
01b	Performing background functions take precedence over maintaining low power conditions as follows: a) if the logical unit is in a low power condition as the result of a power condition timer associated with that condition expiring, then: 1) the logical unit shall change from that power condition, if necessary, to the power condition required to perform the background function, if: a) a timer associated with a background scan operation expires, and that function is enabled (see SBC-3); or b) an event occurs to initiate a device specific background function, and that function is enabled (see 5.3); 2) the logical unit shall perform the background function(s) based on the definitions in this standard and other command standards (e.g., if the device server receives a command while performing a background function, then the logical unit shall suspend the function to process the command); 3) if more than one condition is met to initiate a background function, then: a) all initiated background functions shall be performed; and b) the order of performing the functions is vendor specific; and 4) after all initiated background functions have been completed, the device server shall check to see if any power condition timers have expired. If any power condition timer has expired, then the logical unit shall change to the power condition associated with the highest priority timer that has expired; or b) if the logical unit is performing a background function, and a power condition timer expires, then the logical unit shall perform all initiated background functions before the logical unit changes to a power condition associated with a timer that has expired.
10b	Maintaining low power conditions take precedence over performing background functions as follows: a) if the logical unit is in a low power condition, then the logical unit shall not change from that power condition to perform a background function; b) the device server may perform any initiated and enabled background function based on the definitions in this standard or other command standards, if all of the following are true: A) a condition is met to initiate a background function; B) that background function is enabled; C) the logical unit changes to a power condition in which the background function may be performed (e.g., the device server processes a medium access command causing the logical unit to change its power condition to continue processing that command); and D) all outstanding application client requests have been completed; or c) if the logical unit is performing a background function, and a power condition timer expires that causes a change to a power condition in which the logical unit is unable to continue performing the background function, then the logical unit shall: A) suspend the background function; and B) change to the power condition associated with the timer that expired.
11b	Reserved

The behavior of the idle condition timer and standby condition timer controlled by this mode page is defined in the power condition overview (see 5.11.1) and the power condition state machine (see 5.11.8).

If the STANDBY_Y bit is set to one, the standby_y condition timer is enabled. If the STANDBY_Y bit is set to zero, the device server shall ignore the standby_y condition timer.

If the IDLE_C bit is set to one, the idle_c condition timer is enabled. If the IDLE_C bit is set to zero, the device server shall ignore the idle_c condition timer.

If the IDLE_B bit is set to one, the idle_b condition timer is enabled. If the IDLE_B bit is set to zero, the device server shall ignore the idle_b condition timer.

If the IDLE_A bit is set to one, the idle_a condition timer is enabled. If the IDLE_A bit is set to zero, the device server shall ignore the idle_a condition timer.

If the STANDBY_Z bit is set to one, the standby_z condition timer is enabled. If the STANDBY_Z bit is set to zero, the device server shall ignore the standby_z condition timer.

If any of the power condition enable bits (e.g., the IDLE_C bit or the STANDBY_Y bit) are set to zero and are not changeable (see 6.13.3), then the device server does not implement the power condition timer associated with that enable bit (see table 65 in 5.11.8.1).

The IDLE_A CONDITION TIMER field specifies the initial value, in 100 millisecond increments, for the idle_a power condition timer (see 5.11.8.1). This value may be rounded up or down to the nearest implemented time as described in 5.8.

The STANDBY_Z CONDITION TIMER field specifies the initial value, in 100 millisecond increments, for the standby_z power condition timer (see 5.11.8.1). This value may be rounded up or down to the nearest implemented time as described in 5.8.

The IDLE_B CONDITION TIMER field specifies the initial value, in 100 millisecond increments, for the idle_b power condition timer (see 5.11.8.1). This value may be rounded up or down to the nearest implemented time as described in 5.8.

The IDLE_C CONDITION TIMER field specifies the initial value, in 100 millisecond increments, for the idle_c power condition timer (see 5.11.8.1). This value may be rounded up or down to the nearest implemented time as described in 5.8.

The STANDBY_Y CONDITION TIMER field specifies the initial value, in 100 millisecond increments, for the standby_y power condition timer (see 5.11.8.1). This value may be rounded up or down to the nearest implemented time as described in 5.8.

The CHECK CONDITION if from idle_c (CCF IDLE) field is defined in table 472.

Table 472 — CCF IDLE field

Code	Description
00b	Restricted [a]
01b	If the transition was from an idle_c power condition, returning CHECK CONDITION status is disabled. [b]
10b	If the transition was from an idle_c power condition, returning CHECK CONDITION status is enabled. [b]
11b	Reserved

[a] See SAS-2 for command processing in the Active_Wait state and Idle_Wait state.

[b] For direct-access block devices see the Active_Wait state in SBC-3 for the definition of command processing in that state. For devices that are not direct-access block devices, see the Active_Wait state in this standard (i.e., see 5.11.8.6) for the definition of command processing in that state.

The CHECK CONDITION if from standby (CCF STANDBY) field is defined in table 473.

Table 473 — CCF STANDBY field

Code	Description
00b	Restricted [a]
01b	If the transition was from a standby power condition, returning CHECK CONDITION status is disabled. [b]
10b	If the transition was from a standby power condition, returning CHECK CONDITION status is enabled. [b]
11b	Reserved

[a] See SAS-2 for command processing in the Active_Wait state and Idle_Wait state.
[b] For direct-access block devices see the Active_Wait state and the Idle_Wait state in SBC-3 for the definition of command processing in those states. For devices that are not direct-access block devices, see the Active_Wait state in this standard (i.e., see 5.11.8.6) for the definition of command processing in that state.

The CHECK CONDITION if from stopped (CCF STOPPED) field is defined in table 474.

Table 474 — CCF STOPPED field

Code	Description
00b	Restricted [a]
01b	If the transition was from a stopped power condition, returning CHECK CONDITION status is disabled. [b]
10b	If the transition was from a stopped power condition, returning CHECK CONDITION status is enabled. [b]
11b	Reserved

[a] See SAS-2 for command processing in the Active_Wait state and Idle_Wait state.
[b] For direct-access block devices see the Active_Wait state, the Idle_Wait state description and the Standby_Wait state in SBC-3 for the definition of command processing in those states.

7.5.14 Power Consumption mode page

The Power Consumption mode page (see table 475) provides a method to select a maximum power consumption level while in the active power condition (see 5.11.4) based on the contents of the power consumption descriptors in the Power Consumption VPD page (see 7.8.11) as described in 5.11.2. The mode page policy (see 7.5.2) for this mode page shall be shared.

Table 475 — Power Consumption mode page

Bit / Byte	7	6	5	4	3	2	1	0
0	PS	SPF (1b)	PAGE CODE (1Ah)					
1	SUBPAGE CODE (01h)							
2	(MSB)							
3	PAGE LENGTH (000Ch)							(LSB)
4	Reserved							
•••								
6								
7	POWER CONSUMPTION IDENTIFIER							
8	Reserved							
•••								
15								

The PS bit, SPF bit, PAGE CODE field, SUBPAGE CODE field, and PAGE LENGTH field are described in 7.5.7.

The SPF bit, PAGE CODE field, SUBPAGE CODE field, and PAGE LENGTH field shall be set as shown in table 475 for the Power Consumption mode page.

The POWER CONSUMPTION IDENTIFIER field specifies the power consumption identifier from one of the power consumption descriptors in the Power Consumption VPD page (see 7.8.11) that the device server is to use as described in 5.11.2. If none of the power consumption descriptors in the Power Consumption VPD page contain the value in the POWER CONSUMPTION IDENTIFIER field, then the device server shall terminate the MODE SELECT command with CHECK CONDITION status, with the sense key set to ILLEGAL REQUEST, and the additional sense code set to INVALID FIELD IN PARAMETER LIST.

7.5.15 Protocol Specific Logical Unit mode page

The Protocol Specific Logical Unit mode page (see table 476) provides protocol specific controls that are associated with a logical unit.

During an I_T_L nexus, the Protocol Specific Logical Unit mode page controls parameters that affect both:

a) one or more target ports; and
b) the logical unit.

The parameters that may be implemented are defined in the SCSI transport protocol standard for the target port. The mode page policy (see 7.5.2) for this mode page shall be shared or per target port and should be per target port.

The parameters for a target port and logical unit affect their behavior regardless of which initiator port is forming an I_T_L nexus with the target port and logical unit. If a parameter value is changed, the device server shall establish a unit attention condition for the initiator port associated with every I_T nexus except the I_T

nexus on which the MODE SELECT command was received, with the additional sense code set to MODE PARAMETERS CHANGED.

Table 476 — Protocol Specific Logical Unit mode page

Bit / Byte	7	6	5	4	3	2	1	0
0	PS	SPF (0b)	PAGE CODE (18h)					
1	PAGE LENGTH (n-1)							
2	Protocol specific mode parameters				PROTOCOL IDENTIFIER			
3	Protocol specific mode parameters							
•••								
n								

The PS bit, SPF bit, PAGE CODE field, and PAGE LENGTH field are described in 7.5.7.

The SPF bit and PAGE CODE field shall be set as shown in table 476 for the Protocol Specific Logical Unit mode page.

The value in the PROTOCOL IDENTIFIER field (see 7.6.1) defines the SCSI transport protocol to which the mode page applies. For a MODE SENSE command (see 6.13), the device server shall set the PROTOCOL IDENTIFIER field to one of the values shown in table 479 (see 7.6.1) to indicate the SCSI transport protocol used by the target port through which the MODE SENSE command is being processed. For a MODE SELECT command (see 6.11), the application client should set the PROTOCOL IDENTIFIER field to the same value that is returned in a MODE SENSE command for that SCSI target port. If a device server receives a mode page containing a transport protocol identifier value other than the one used by the target port on which the MODE SELECT command was received, then the device server shall terminate the command with CHECK CONDITION status, with the sense key set to ILLEGAL REQUEST, and the additional sense code set to INVALID FIELD IN PARAMETER LIST.

7.5.16 Protocol Specific Port mode page

The Protocol Specific Port mode page provides protocol specific controls that are associated with a SCSI port. The page_0 mode page format (see table 477) is used if a MODE SENSE command (see 6.13.1) contains zero in the SUBPAGE CODE field, and sub_page mode page format (see table 478) is used for subpages 01h through FEh. See the SCSI transport protocol standard for definition of the protocol specific mode parameters.

The Protocol Specific Port mode page controls parameters that affect one or more target ports. The parameters that may be implemented are defined in the SCSI transport protocol standard for the target port. The mode page policy (see 7.5.2) for this mode page shall be shared or per target port. If the SCSI target device contains more than one target port, the mode page policy should be per target port.

The parameters for a target port affect its behavior regardless of which initiator port is forming an I_T nexus with the target port. The MLUS bit (see 7.8.9) shall be set to one in the mode page policy descriptor for this mode page.

The parameters may be accessed by MODE SENSE (see 6.13) and MODE SELECT (see 6.11) commands directed to any logical unit accessible through the target port. If a parameter value is changed, the device servers for all logical units accessible through the target port shall establish a unit attention condition for the

initiator port associated with every I_T nexus except the I_T nexus on which the MODE SELECT command was received, with the additional sense code set to MODE PARAMETERS CHANGED.

Table 477 — Page_0 mode page format Protocol Specific Port mode page

Bit / Byte	7	6	5	4	3	2	1	0
0	PS	SPF (0b)	PAGE CODE (19h)					
1	PAGE LENGTH (n-1)							
2	Protocol specific mode parameters				PROTOCOL IDENTIFIER			
3	Protocol specific mode parameters							
•••								
n								

Table 478 — Sub_page mode page format Protocol Specific Port mode page

Bit / Byte	7	6	5	4	3	2	1	0
0	PS	SPF (1b)	PAGE CODE (19h)					
1	SUBPAGE CODE							
2	(MSB)							
3	PAGE LENGTH (n-3)							(LSB)
4	Reserved							
5	Protocol specific mode parameters				PROTOCOL IDENTIFIER			
6	Protocol specific mode parameters							
•••								
n								

The PS bit, SPF bit, PAGE CODE field, SUBPAGE CODE field, and PAGE LENGTH field are described in 7.5.7.

The SPF bit and PAGE CODE field shall be set as shown in table 477 or table 478 for the Protocol Specific Port mode page.

The value in the PROTOCOL IDENTIFIER field (see 7.6.1) defines the SCSI transport protocol to which the mode page applies. For a MODE SENSE command, the device server shall set the PROTOCOL IDENTIFIER field to one of the values shown in table 479 (see 7.6.1) to indicate the SCSI transport protocol used by the target port through which the MODE SENSE command is being processed. For a MODE SELECT command, the application client should set the PROTOCOL IDENTIFIER field to the same value that is returned in a MODE SENSE command for that SCSI target port. If a device server receives a mode page containing a transport protocol identifier value other than the one used by the target port on which the MODE SELECT command was received, then the device server shall terminate the command with CHECK CONDITION status, with the sense key set to ILLEGAL REQUEST, and the additional sense code set to INVALID FIELD IN PARAMETER LIST.

7.6 Protocol specific parameters

7.6.1 Protocol specific parameters introduction

Some commands include protocol specific information in their command definitions. This subclause describes those protocol specific parameters.

Protocol specific parameters may include a PROTOCOL IDENTIFIER field (see table 479) as a reference for the SCSI transport protocol to which the protocol specific parameter applies.

Table 479 — PROTOCOL IDENTIFIER field values

Protocol Identifier	Description	Protocol Standard
0h	Fibre Channel Protocol for SCSI	FCP-4
1h	SCSI Parallel Interface	SPI-5
2h	Serial Storage Architecture SCSI-3 Protocol	SSA-S3P
3h	Serial Bus Protocol for IEEE 1394	SBP-3
4h	SCSI RDMA Protocol	SRP
5h	Internet SCSI (iSCSI)	iSCSI
6h	SAS Serial SCSI Protocol	SPL-3
7h	Automation/Drive Interface Transport Protocol	ADT-2
8h	AT Attachment Interface	ACS-2
9h	USB Attached SCSI	UAS
Ah	SCSI over PCI Express	SOP
Bh to Eh	Reserved	
Fh	No specific protocol	

7.6.2 Alias entry protocol specific designations

7.6.2.1 Introduction to alias entry protocol specific designations

The alias entry formats (see 6.2.2) used by specific SCSI transport protocols in the CHANGE ALIASES command (see 6.2) and REPORT ALIASES command (see 6.30) are based on the SCSI transport protocol specified in the PROTOCOL IDENTIFIER field (see 7.6.1). These alias entry formats are defined in 7.6.2.

7.6.2.2 Fibre Channel specific alias entry formats

7.6.2.2.1 Summary of Fibre Channel specific alias entry formats

The alias entry formats for the Fibre Channel protocol are summarized in table 480.

Table 480 — Fibre Channel alias entry format codes

Format Code	Description	Designation Length (bytes)	Reference
00h	World Wide Port Name	8	7.6.2.2.2

Table 480 — Fibre Channel alias entry format codes

Format Code	Description	Designation Length (bytes)	Reference
01h	World Wide Port Name with N_Port checking	12	7.6.2.2.3
02h to FFh	Reserved		

7.6.2.2.2 Fibre Channel world wide port name alias entry format

The format of a Fibre Channel world wide port name alias entry is shown in table 481.

Table 481 — Fibre Channel world wide port name alias entry format

Bit / Byte	7	6	5	4	3	2	1	0
0	(MSB)							
•••				ALIAS VALUE				
7								(LSB)
8	PROTOCOL IDENTIFIER (00h)							
9								
10				Reserved				
11	FORMAT CODE (00h)							
12								
13				Reserved				
14	(MSB)							
15				DESIGNATION LENGTH (0008h)				(LSB)
16								
•••				FIBRE CHANNEL WORLD WIDE PORT NAME				
23								

The ALIAS VALUE field is defined in 6.2.2.

The PROTOCOL IDENTIFIER field is defined in 6.2.2 and shall be set as shown in table 481 for the Fibre Channel world wide port name alias entry format.

The FORMAT CODE field is defined in 6.2.2 and shall be set as shown in table 481 for the Fibre Channel world wide port name alias entry format.

The DESIGNATION LENGTH field is defined in 6.2.2 and shall be set as shown in table 481 for the Fibre Channel world wide port name alias entry format.

The FIBRE CHANNEL WORLD WIDE PORT NAME field shall contain the port world wide name defined by the port login (PLOGI) extended link service (see FC-FS-3).

A Fibre Channel world wide port name designation is valid (see 6.2.3) if the device server has access to a SCSI domain formed by a Fibre Channel fabric and the fabric contains a port with the specified port world wide name.

7.6.2.2.3 Fibre Channel world wide port name with N_Port checking alias entry format

The format of a Fibre Channel world wide port name with N_Port checking alias entry IS shown in table 482.

Table 482 — Fibre Channel world wide port name with N_Port checking alias entry format

Bit / Byte	7	6	5	4	3	2	1	0
0	(MSB)							
•••				ALIAS VALUE				
7								(LSB)
8				PROTOCOL IDENTIFIER (00h)				
9				Reserved				
10								
11				FORMAT CODE (01h)				
12				Reserved				
13								
14	(MSB)			DESIGNATION LENGTH (000Ch)				
15								(LSB)
16				FIBRE CHANNEL WORLD WIDE PORT NAME				
•••								
23								
24				Reserved				
25	(MSB)							
26				N_PORT				
27								(LSB)

The ALIAS VALUE field is defined in 6.2.2.

The PROTOCOL IDENTIFIER field is defined in 6.2.2 and shall be set as shown in table 482 for the Fibre Channel world wide port name with N_Port checking alias entry format.

The FORMAT CODE field is defined in 6.2.2 and shall be set as shown in table 482 for the Fibre Channel world wide port name with N_Port checking alias entry format.

The DESIGNATION LENGTH field is defined in 6.2.2 and shall be set as shown in table 482 for the Fibre Channel world wide port name with N_Port checking alias entry format.

The FIBRE CHANNEL WORLD WIDE PORT NAME field shall contain the port world wide name defined by the port login (PLOGI) extended link service (see FC-FS-3).

The N_PORT field shall contain the FC-FS-3 port D_ID to be used to transport frames including PLOGI and FCP-4 related frames.

A Fibre Channel world wide port name with N_Port checking designation is valid (see 6.2.3) if all of the following conditions are true:

a) the device server has access to a SCSI domain formed by a Fibre Channel fabric;
b) the fabric contains a port with the specified port World Wide Name; and

c) the value in the N_PORT field is the N_Port identifier of a Fibre Channel port whose port world wide name matches that in the FIBRE CHANNEL WORLD WIDE PORT NAME field.

7.6.2.3 RDMA specific alias entry formats

7.6.2.3.1 Summary of RDMA specific alias entry formats

The alias entry formats for the SCSI RDMA protocol are summarized in table 483.

Table 483 — RDMA alias entry format codes

Format Code	Description	Designation Length (bytes)	Reference
00h	Target Port Identifier	16	7.6.2.3.2
01h	InfiniBand™ Global Identifier with Target Port Identifier checking	32	7.6.2.3.3
02h to FFh	Reserved		

7.6.2.3.2 RDMA target port identifier alias entry format

The format of a SCSI RDMA target port identifier alias entry IS shown in table 484.

Table 484 — RDMA target port identifier alias entry format

Bit / Byte	7	6	5	4	3	2	1	0
0	(MSB)							
•••				ALIAS VALUE				
7								(LSB)
8	PROTOCOL IDENTIFIER (04h)							
9								
10				Reserved				
11	FORMAT CODE (00h)							
12								
13				Reserved				
14	(MSB)							
15				DESIGNATION LENGTH (0010h)				(LSB)
16								
•••				TARGET PORT IDENTIFIER				
31								

The ALIAS VALUE field is defined in 6.2.2.

The PROTOCOL IDENTIFIER field is defined in 6.2.2 and shall be set as shown in table 484 for the RDMA target port identifier alias entry format.

The FORMAT CODE field is defined in 6.2.2 and shall be set as shown in table 484 for the RDMA target port identifier alias entry format.

The DESIGNATION LENGTH field is defined in 6.2.2 and shall be set as shown in table 484 for the RDMA target port identifier alias entry format.

The TARGET PORT IDENTIFIER field shall contain an SRP target port identifier.

A SCSI RDMA target port identifier designation is valid (see 6.2.3) if the device server has access to an SRP SCSI domain containing the specified SRP target port identifier.

7.6.2.3.3 InfiniBand global identifier with target port identifier checking alias entry format

The format of an InfiniBand global identifier with target port identifier checking alias entry is shown in table 485.

Table 485 — InfiniBand global identifier with target port identifier checking alias entry format

Bit / Byte	7	6	5	4	3	2	1	0
0	(MSB)							
•••				ALIAS VALUE				
7								(LSB)
8				PROTOCOL IDENTIFIER (04h)				
9								
10				Reserved				
11				FORMAT CODE (01h)				
12								
13				Reserved				
14	(MSB)							
15				DESIGNATION LENGTH (0020h)				(LSB)
16								
•••				INFINIBAND GLOBAL IDENTIFIER				
31								
32								
•••				TARGET PORT IDENTIFIER				
47								

The ALIAS VALUE field is defined in 6.2.2.

The PROTOCOL IDENTIFIER field is defined in 6.2.2 and shall be set as shown in table 485 for the InfiniBand global identifier with target port identifier checking alias entry format.

The FORMAT CODE field is defined in 6.2.2 and shall be set as shown in table 485 for the InfiniBand global identifier with target port identifier checking alias entry format.

The DESIGNATION LENGTH field is defined in 6.2.2 and shall be set as shown in table 485 for the InfiniBand global identifier with target port identifier checking alias entry format.

The INFINIBAND GLOBAL IDENTIFIER field specifies an InfiniBand global identifier (GID) of an InfiniBand port connected to an SRP target port.

The TARGET PORT IDENTIFIER field specify an SRP target port identifier.

An InfiniBand global identifier with target port identifier checking designation is valid (see 6.2.3) if all of the following conditions are true:

a) the device server has access to an SRP SCSI domain layered on InfiniBand;
b) the device server has access to an SRP target port based on the InfiniBand global identifier specified in the INFINIBAND GLOBAL IDENTIFIER field; and
c) the value in the TARGET PORT IDENTIFIER field is the SRP target port identifier for the SRP target port that is accessible via the InfiniBand global identifier contained in the INFINIBAND GLOBAL IDENTIFIER field.

7.6.2.4 Internet SCSI specific alias entry formats

7.6.2.4.1 Summary of Internet SCSI specific alias entry formats

The alias entry formats for the iSCSI protocol are summarized in table 486.

Table 486 — iSCSI alias entry format codes

Format Code	Description	Designation Length (bytes, maximum)	Reference
00h	iSCSI Name	224	7.6.2.4.2
01h	iSCSI Name with binary IPv4 address	236	7.6.2.4.3
02h	iSCSI Name with IPName	488	7.6.2.4.4
03h	iSCSI Name with binary IPv6 address	248	7.6.2.4.5
04h to FFh	Reserved		

NOTE 48 - A designation that contains no IP addressing information or contains IP addressing information that does not address the named SCSI target device may require a device server to have access to a name server or to other discovery protocols to resolve the given iSCSI Name to an IP address through which the device server may establish iSCSI Login. Access to such a service is protocol specific and vendor specific.

7.6.2.4.2 iSCSI name alias entry format

The format of an iSCSI name alias entry IS shown in table 487.

Table 487 — iSCSI name alias entry format

Bit / Byte	7	6	5	4	3	2	1	0
0	(MSB)							
•••				ALIAS VALUE				
7								(LSB)
8				PROTOCOL IDENTIFIER (05h)				
9				Reserved				
10								
11				FORMAT CODE (00h)				
12				Reserved				
13								
14	(MSB)							
15				DESIGNATION LENGTH (4m-16)				(LSB)
16	(MSB)							
•••				ISCSI NAME				
4m-1								(LSB)

The ALIAS VALUE field is defined in 6.2.2.

The PROTOCOL IDENTIFIER field is defined in 6.2.2 and shall be set as shown in table 487 for the iSCSI name alias entry format.

The FORMAT CODE field is defined in 6.2.2 and shall be set as shown in table 487 for the iSCSI name alias entry format.

The DESIGNATION LENGTH field is defined in 6.2.2.

The null-terminated, null-padded (see 4.3.2) ISCSI NAME field shall contain the iSCSI name of an iSCSI node (see RFC 7143). The number of bytes in the ISCSI NAME field shall be a multiple of four.

An iSCSI name designation is valid if the device server has access to a SCSI domain containing an Internet protocol network and that network contains an iSCSI node with the specified iSCSI name.

7.6.2.4.3 iSCSI name with binary IPv4 address alias entry format

The format of an iSCSI name with binary IPv4 address alias entry is shown in table 488.

Table 488 — iSCSI name with binary IPv4 address alias entry format

Bit / Byte	7	6	5	4	3	2	1	0
0	(MSB)							
•••				ALIAS VALUE				
7								(LSB)
8				PROTOCOL IDENTIFIER (05h)				
9				Reserved				
10								
11				FORMAT CODE (01h)				
12				Reserved				
13								
14	(MSB)							
15				DESIGNATION LENGTH (4m+12)				(LSB)
16	(MSB)							
•••				ISCSI NAME				
4m-1								(LSB)
4m	(MSB)							
•••				IPV4 ADDRESS				
4m+3								(LSB)
4m+4				Reserved				
4m+5								
4m+6	(MSB)							
4m+7				PORT NUMBER				(LSB)
4m+8				Reserved				
4m+9								
4m+10	(MSB)							
4m+11				INTERNET PROTOCOL NUMBER				(LSB)

The ALIAS VALUE field is defined in 6.2.2.

The PROTOCOL IDENTIFIER field is defined in 6.2.2 and shall be set as shown in table 488 for the iSCSI name with binary IPv4 address alias entry format.

The FORMAT CODE field is defined in 6.2.2 and shall be set as shown in table 488 for the iSCSI name with binary IPv4 address alias entry format.

The DESIGNATION LENGTH field is defined in 6.2.2.

The null-terminated, null-padded (see 4.3.2) ISCSI NAME field shall contain the iSCSI name of an iSCSI node (see RFC 7143). The number of bytes in the ISCSI NAME field shall be a multiple of four.

The IPV4 ADDRESS field shall contain an IPv4 address (see RFC 791).

The PORT NUMBER field shall contain a TCP port number. The TCP port number shall conform to the requirements defined by iSCSI (see RFC 7143).

The INTERNET PROTOCOL NUMBER field shall contain an Internet protocol number. The Internet protocol number shall conform to the requirements defined by iSCSI (see RFC 7143).

An iSCSI name designation is valid if the device server has access to a SCSI domain containing an Internet protocol network and that network contains an iSCSI node with the specified iSCSI name.

The IPv4 address, port number, and Internet protocol number provided in the designation may be used by a device server for addressing to discover and establish communication with the named iSCSI node. Alternatively, the device server may use other protocol specific or vendor specific methods to discover and establish communication with the named iSCSI node.

7.6.2.4.4 iSCSI name with IPname alias entry format

The format of an iSCSI name with IPname alias entry IS shown in table 489.

Table 489 — iSCSI name with IPname alias entry format

Bit / Byte	7	6	5	4	3	2	1	0
0	(MSB)							
•••				ALIAS VALUE				
7								(LSB)
8				PROTOCOL IDENTIFIER (05h)				
9				Reserved				
10								
11				FORMAT CODE (03h)				
12				Reserved				
13								
14	(MSB)			DESIGNATION LENGTH (4m+8)				
15								(LSB)
16	(MSB)							
•••				ISCSI NAME				
k								(LSB)
k+1	(MSB)							
•••				IPNAME				
n								(LSB)
n+1				PAD (if any)				
4m-1								
4m				Reserved				
4m+1								
4m+2	(MSB)			PORT NUMBER				
4m+3								(LSB)
4m+4				Reserved				
4m+5								
4m+6	(MSB)			INTERNET PROTOCOL NUMBER				
4m+7								(LSB)

The ALIAS VALUE field is defined in 6.2.2.

The PROTOCOL IDENTIFIER field is defined in 6.2.2 and shall be set as shown in table 489 for the iSCSI name with IPname alias entry format.

The FORMAT CODE field is defined in 6.2.2 and shall be set as shown in table 489 for the iSCSI name with IPname alias entry format.

The DESIGNATION LENGTH field is defined in 6.2.2.

The null-terminated (see 4.3.2) ISCSI NAME field shall contain the iSCSI name of an iSCSI node (see RFC 7143).

The null-terminated (see 4.3.2) IPNAME field shall contain a Internet protocol domain name.

The PAD field shall contain zero to three bytes set to zero such that the total length of the ISCSI NAME field, IPNAME field, and PAD field is a multiple of four. Device servers shall ignore the PAD field.

The PORT NUMBER field shall contain a TCP port number. The TCP port number shall conform to the requirements defined by iSCSI (see RFC 7143).

The INTERNET PROTOCOL NUMBER field shall contain an Internet protocol number. The Internet protocol number shall conform to the requirements defined by iSCSI (see RFC 7143).

An iSCSI name designation is valid if the device server has access to a SCSI domain containing an Internet protocol network and that network contains an iSCSI node with the specified iSCSI name.

The Internet protocol domain name, port number, and Internet protocol number provided in the designation may be used by a device server for addressing to discover and establish communication with the named iSCSI node. Alternatively, the device server may use other protocol specific or vendor specific methods to discover and establish communication with the named iSCSI node.

7.6.2.4.5 iSCSI name with binary IPv6 address alias entry format

The format of an iSCSI name with binary IPv6 address alias entry IS shown in table 490.

Table 490 — iSCSI name with binary IPv6 address alias entry format

Bit / Byte	7	6	5	4	3	2	1	0
0	(MSB)							
•••				ALIAS VALUE				
7								(LSB)
8				PROTOCOL IDENTIFIER (05h)				
9				Reserved				
10								
11				FORMAT CODE (03h)				
12				Reserved				
13								
14	(MSB)			DESIGNATION LENGTH (4m+24)				
15								(LSB)
16	(MSB)							
•••				ISCSI NAME				
n								(LSB)
4m	(MSB)							
•••				IPV6 ADDRESS				
4m+15								(LSB)
4m+16				Reserved				
4m+17								
4m+18	(MSB)			PORT NUMBER				
4m+19								(LSB)
4m+20				Reserved				
4m+21								
4m+22	(MSB)			INTERNET PROTOCOL NUMBER				
4m+23								(LSB)

The ALIAS VALUE field is defined in 6.2.2.

The PROTOCOL IDENTIFIER field is defined in 6.2.2 and shall be set as shown in table 490 for the iSCSI name with binary IPv6 address alias entry format.

The FORMAT CODE field is defined in 6.2.2 and shall be set as shown in table 490 for the iSCSI name with binary IPv6 address alias entry format.

The DESIGNATION LENGTH field is defined in 6.2.2.

The null-terminated, null-padded (see 4.3.2) ISCSI NAME field shall contain the iSCSI name of an iSCSI node (see RFC 7143).

The IPV6 ADDRESS field shall contain an IPv6 address (see RFC 4291).

The PORT NUMBER field shall contain a TCP port number. The TCP port number shall conform to the requirements defined by iSCSI (see RFC 7143).

The INTERNET PROTOCOL NUMBER field shall contain an Internet protocol number. The Internet protocol number shall conform to the requirements defined by iSCSI (see RFC 7143).

An iSCSI name designation is valid if the device server has access to a SCSI domain containing an Internet protocol network and that network contains an iSCSI node with the specified iSCSI name.

The IPv6 address, port number and Internet protocol number provided in the designation may be used by a device server for addressing to discover and establish communication with the named iSCSI node. Alternatively, the device server may use other protocol specific or vendor specific methods to discover and establish communication with the named iSCSI node.

7.6.3 EXTENDED COPY protocol specific CSCD descriptors

7.6.3.1 Introduction to EXTENDED COPY protocol specific CSCD descriptors

The protocol-specific CSCD descriptors (see 6.4.5.1) in the parameter list (see 5.16.7.1) of the EXTENDED COPY(LID4) command (see 6.4) and the EXTENDED COPY(LID1) command (see 6.5) are described in 7.6.3.

7.6.3.2 Fibre Channel N_Port_Name CSCD descriptor format

The CSCD descriptor format shown in table 491 is used by an EXTENDED COPY command to specify an FCP CSCD using its Fibre Channel N_Port_Name.

Table 491 — Fibre Channel N_Port_Name CSCD descriptor format

Bit / Byte	7	6	5	4	3	2	1	0
0	DESCRIPTOR TYPE CODE (E0h)							
1	LU ID TYPE		Obsolete	PERIPHERAL DEVICE TYPE				
2	(MSB)			RELATIVE INITIATOR PORT IDENTIFIER				
3								(LSB)
4	LU IDENTIFIER							
•••								
11								
12	N_PORT_NAME							
•••								
19								
20	Reserved							
•••								
27								
28	Device type specific parameters							
•••								
31								

The DESCRIPTOR TYPE CODE field, LU ID TYPE field, PERIPHERAL DEVICE TYPE field, RELATIVE INITIATOR PORT IDENTIFIER field, LU IDENTIFIER field, and the device type specific parameters are described in 6.4.5.1. The DESCRIPTOR TYPE CODE field shall be set as shown in table 491 for the Fibre Channel N_Port_Name CSCD descriptor.

The N_PORT_NAME field shall contain the N_Port_Name defined by the port login (PLOGI) extended link service (see FC-LS-2).

NOTE 49 - The Fibre Channel N_Port_Name CSCD descriptor format requests that the copy manager translate the N_Port_Name to an N_Port_ID (see FC-FS-3, FC-LS-2, and 7.6.3.3).

7.6.3.3 Fibre Channel N_Port_ID CSCD descriptor format

The CSCD descriptor format shown in table 492 is used by an EXTENDED COPY command to specify an FCP CSCD using its Fibre Channel N_Port_ID.

Table 492 — Fibre Channel N_Port_ID CSCD descriptor format

Bit / Byte	7	6	5	4	3	2	1	0
0	DESCRIPTOR TYPE CODE (E1h)							
1	LU ID TYPE		Obsolete	PERIPHERAL DEVICE TYPE				
2	(MSB)			RELATIVE INITIATOR PORT IDENTIFIER				
3								(LSB)
4				LU IDENTIFIER				
•••								
11								
12				Reserved				
•••								
20								
21	(MSB)			N_PORT_ID				
•••								
23								(LSB)
24				Reserved				
•••								
27								
28				Device type specific parameters				
•••								
31								

The DESCRIPTOR TYPE CODE field, LU ID TYPE field, PERIPHERAL DEVICE TYPE field, RELATIVE INITIATOR PORT IDENTIFIER field, LU IDENTIFIER field, and the device type specific parameters are described in 6.4.5.1. The DESCRIPTOR TYPE CODE field shall be set as shown in table 492 for the Fibre Channel N_Port_ID CSCD descriptor.

The N_PORT_ID field shall contain the port D_ID (see FC-FS-3) to be used to transport frames including PLOGI (see FC-LS-2) and FCP-4 related frames.

NOTE 50 - Use of N_Port_ID addressing restricts this CSCD descriptor format to a single Fibre Channel fabric.

7.6.3.4 Fibre Channel N_Port_ID With N_Port_Name Checking CSCD descriptor format

The CSCD descriptor format shown in table 493 is used by an EXTENDED COPY command to specify an FCP CSCD using its Fibre Channel N_Port_ID and to require the copy manager to verify that the N_Port_Name of the specified N_Port matches the value in the CSCD descriptor.

Table 493 — Fibre Channel N_Port_ID With N_Port_Name Checking CSCD descriptor format

Bit / Byte	7	6	5	4	3	2	1	0
0	DESCRIPTOR TYPE CODE (E2h)							
1	LU ID TYPE		Obsolete	PERIPHERAL DEVICE TYPE				
2	(MSB)			RELATIVE INITIATOR PORT IDENTIFIER				
3								(LSB)
4				LU IDENTIFIER				
•••								
11								
12				N_PORT_NAME				
•••								
19								
20	Reserved							
21	(MSB)			N_PORT_ID				
•••								
23								(LSB)
24				Reserved				
•••								
27								
28				Device type specific parameters				
•••								
31								

The DESCRIPTOR TYPE CODE field, LU ID TYPE field, PERIPHERAL DEVICE TYPE field, RELATIVE INITIATOR PORT IDENTIFIER field, LU IDENTIFIER field, and the device type specific parameters are described in 6.4.5.1. The DESCRIPTOR TYPE CODE field shall be set as shown in table 493 for the Fibre Channel N_Port_ID With N_Port_Name Checking CSCD descriptor.

The N_PORT_NAME field shall contain the N_Port_Name defined by the port login (PLOGI) extended link service (see FC-FS-3 and FC-LS-2).

The N_PORT_ID field shall contain the port D_ID (see FC-FS-3) to be used to transport frames including PLOGI (see FC-LS-2) and FCP-4 related frames.

NOTE 51 - Use of N_Port addressing restricts this CSCD descriptor format to a single fabric.

If the copy manager first processes a segment descriptor that references this type of CSCD descriptor, the copy manager shall confirm that the D_ID in the N_PORT_ID field is associated with the N_Port_Name in the N_PORT_NAME field. If the confirmation fails, the copy manager shall terminate the copy operation (see 5.16.4.3) because the CSCD is unreachable (see 5.16.7.4). The copy manager processing this CSCD descriptor shall track configuration changes that affect the D_ID value for the duration of the copy operation (see 5.16.4.3). An application client is responsible for tracking configuration changes between commands.

7.6.3.5 SCSI Parallel T_L CSCD descriptor format

The CSCD descriptor format shown in table 494 is used by an EXTENDED COPY command to specify a SPI CSCD using one of its target port identifiers (see SAM-5).

Table 494 — SCSI Parallel T_L CSCD descriptor format

Bit / Byte	7	6	5	4	3	2	1	0
0	DESCRIPTOR TYPE CODE (E3h)							
1	LU ID TYPE		Obsolete	PERIPHERAL DEVICE TYPE				
2	(MSB)			RELATIVE INITIATOR PORT IDENTIFIER				
3								(LSB)
4				LU IDENTIFIER				
•••								
11								
12	Vendor specific							
13	TARGET IDENTIFIER							
14				Reserved				
•••								
27								
28				Device type specific parameters				
•••								
31								

The DESCRIPTOR TYPE CODE field, LU ID TYPE field, PERIPHERAL DEVICE TYPE field, RELATIVE INITIATOR PORT IDENTIFIER field, LU IDENTIFIER field, and the device type specific parameters are described in 6.4.5.1. The DESCRIPTOR TYPE CODE field shall be set as shown in table 494 for the SCSI Parallel T_L CSCD descriptor.

The TARGET IDENTIFIER field specifies the SCSI target identifier (see SPI-5).

7.6.3.6 IEEE 1394 EUI-64 CSCD descriptor format

The CSCD descriptor format shown in table 495 is used by an EXTENDED COPY command to specify an SBP-3 CSCD using its IEEE 1394 Extended Unique Identifier, 64-bits (EUI-64) and configuration ROM directory identifier.

Table 495 — IEEE 1394 EUI-64 CSCD descriptor format

Bit / Byte	7	6	5	4	3	2	1	0
0	DESCRIPTOR TYPE CODE (E8h)							
1	LU ID TYPE		Obsolete	PERIPHERAL DEVICE TYPE				
2	(MSB)			RELATIVE INITIATOR PORT IDENTIFIER				
3								(LSB)
4				LU IDENTIFIER				
•••								
11								
12				EUI-64				
•••								
19								
20				DIRECTORY ID				
•••								
22								
23				Reserved				
•••								
27								
28				Device type specific parameters				
•••								
31								

The DESCRIPTOR TYPE CODE field, LU ID TYPE field, PERIPHERAL DEVICE TYPE field, RELATIVE INITIATOR PORT IDENTIFIER field, LU IDENTIFIER field, and the device type specific parameters are described in 6.4.5.1. The DESCRIPTOR TYPE CODE field shall be set as shown in table 495 for the IEEE 1394 EUI-64 CSCD descriptor.

The EUI-64 field shall contain the SBP-3 node's unique identifier (EUI-64) obtained from the configuration ROM bus information block, as specified by ANSI IEEE 1394a:2000.

NOTE 52 - ANSI IEEE 1394a-2000 separately labels the components of the EUI-64 as NODE_VENDOR_ID, CHIP_ID_HI and CHIP_ID_LO. Collectively these form the node's EUI-64.

The DIRECTORY ID field shall contain the CSCD's directory identifier, as specified by ISO/IEC 13213:1994.

7.6.3.7 RDMA CSCD descriptor format

The CSCD descriptor format shown in table 496 is used by an EXTENDED COPY command to specify an SRP CSCD using its RDMA SRP target port identifier.

Table 496 — RDMA CSCD descriptor format

Bit / Byte	7	6	5	4	3	2	1	0
0	DESCRIPTOR TYPE CODE (E7h)							
1	LU ID TYPE		Obsolete	PERIPHERAL DEVICE TYPE				
2	(MSB)			RELATIVE INITIATOR PORT IDENTIFIER				
3								(LSB)
4				LU IDENTIFIER				
•••								
11								
12				TARGET PORT IDENTIFIER				
•••								
27								
28				Device type specific parameters				
•••								
31								

The DESCRIPTOR TYPE CODE field, LU ID TYPE field, PERIPHERAL DEVICE TYPE field, RELATIVE INITIATOR PORT IDENTIFIER field, LU IDENTIFIER field, and the device type specific parameters are described in 6.4.5.1. The DESCRIPTOR TYPE CODE field shall be set as shown in table 496 for the RDMA CSCD descriptor.

The TARGET PORT IDENTIFIER field specifies the SRP target port identifier (see SRP).

7.6.3.8 iSCSI IPv4 CSCD descriptor format

The CSCD descriptor format shown in table 497 is used by an EXTENDED COPY command to specify an iSCSI CSCD using its binary IPv4 address.

Table 497 — iSCSI IPv4 CSCD descriptor format

Bit / Byte	7	6	5	4	3	2	1	0
0	DESCRIPTOR TYPE CODE (E5h)							
1	LU ID TYPE		Obsolete	PERIPHERAL DEVICE TYPE				
2	(MSB)			RELATIVE INITIATOR PORT IDENTIFIER				
3								(LSB)
4				LU IDENTIFIER				
•••								
11								
12	(MSB)			IPV4 ADDRESS				
•••								
15								(LSB)
16				Reserved				
•••								
21								
22	(MSB)			PORT NUMBER				
23								(LSB)
24				Reserved				
25								
26	(MSB)			INTERNET PROTOCOL NUMBER				
27								(LSB)
28				Device type specific parameters				
•••								
31								

The DESCRIPTOR TYPE CODE field, LU ID TYPE field, PERIPHERAL DEVICE TYPE field, RELATIVE INITIATOR PORT IDENTIFIER field, LU IDENTIFIER field, and the device type specific parameters are described in 6.4.5.1. The DESCRIPTOR TYPE CODE field shall be set as shown in table 497 for the iSCSI IPv4 CSCD descriptor.

The IPV4 ADDRESS field shall contain an IPv4 address (see RFC 791).

The PORT NUMBER field shall contain the TCP port number. The TCP port number shall conform to the requirements defined by iSCSI (see RFC 7143).

The INTERNET PROTOCOL NUMBER field shall contain an Internet protocol number. The Internet protocol number shall conform to the requirements defined by iSCSI (see RFC 7143).

NOTE 53 - The internet protocol number for TCP is 0006h.

7.6.3.9 iSCSI IPv6 CSCD descriptor format

The CSCD descriptor format shown in table 498 is used by an EXTENDED COPY command to specify an iSCSI CSCD using its binary IPv6 address.

Table 498 — iSCSI IPv6 CSCD descriptor format

Bit / Byte	7	6	5	4	3	2	1	0
0	DESCRIPTOR TYPE CODE (EAh)							
1	LU ID TYPE		Obsolete	PERIPHERAL DEVICE TYPE				
2	(MSB)			RELATIVE INITIATOR PORT IDENTIFIER				
3								(LSB)
4				LU IDENTIFIER				
•••								
11								
12	(MSB)			IPV6 ADDRESS				
•••								
27								(LSB)
28				Device type specific parameters				
•••								
31								
32	EXTENSION DESCRIPTOR TYPE CODE (FFh)							
33				Reserved				
34								
35								
36	(MSB)			PORT NUMBER				
37								(LSB)
38	(MSB)			INTERNET PROTOCOL NUMBER				
39								(LSB)
40				Reserved				
•••								
63								

The DESCRIPTOR TYPE CODE field, LU ID TYPE field, PERIPHERAL DEVICE TYPE field, RELATIVE INITIATOR PORT IDENTIFIER field, LU IDENTIFIER field, and the device type specific parameters are described in 6.4.5.1. The DESCRIPTOR TYPE CODE field shall be set as shown in table 498 for the iSCSI IPv6 CSCD descriptor.

The IPV6 ADDRESS field shall contain a unicast IPv6 address (see RFC 4291).

The EXTENSION DESCRIPTOR TYPE CODE field is described in 6.4.5.2. If the EXTENSION DESCRIPTOR TYPE CODE field does not contain the value shown in table 498, then the copy manager shall terminate the copy operation (see 5.16.4.3) with CHECK CONDITION status, with the sense key set to COPY ABORTED, and the additional sense code set to INVALID FIELD IN PARAMETER LIST.

The PORT NUMBER field shall contain the TCP port number. The TCP port number shall conform to the requirements defined by iSCSI (see RFC 7143).

The INTERNET PROTOCOL NUMBER field shall contain an Internet protocol number. The Internet protocol number shall conform to the requirements defined by iSCSI (see RFC 7143).

NOTE 54 - The internet protocol number for TCP is 0006h.

7.6.3.10 SAS Serial SCSI Protocol CSCD descriptor format

The CSCD descriptor format shown in table 499 is used by an EXTENDED COPY command to specify a SAS CSCD using its SAS address.

Table 499 — SAS Serial SCSI Protocol CSCD descriptor format

Bit / Byte	7	6	5	4	3	2	1	0
0	DESCRIPTOR TYPE CODE (E9h)							
1	LU ID TYPE		Obsolete	PERIPHERAL DEVICE TYPE				
2	(MSB)			RELATIVE INITIATOR PORT IDENTIFIER				
3								(LSB)
4	LU IDENTIFIER							
•••								
11								
12	SAS ADDRESS							
•••								
19								
20	Reserved							
•••								
27								
28	Device type specific parameters							
•••								
31								

The DESCRIPTOR TYPE CODE field, LU ID TYPE field, PERIPHERAL DEVICE TYPE field, RELATIVE INITIATOR PORT IDENTIFIER field, LU IDENTIFIER field, and the device type specific parameters are described in 6.4.5.1. The DESCRIPTOR TYPE CODE field shall be set as shown in table 499 for the SAS Serial SCSI Protocol CSCD descriptor.

The SAS ADDRESS field specifies the SAS address (see SPL-3).

7.6.4 TransportID identifiers

7.6.4.1 Overview of TransportID identifiers

An application client may use a TransportID to specify an initiator port other than the initiator port that is transporting the command and parameter data (e.g., as an Access identifiers (see 8.3.1.3.2) in ACL ACEs, as the initiator port in the I_T nexus to which PERSISTENT RESERVE OUT command with REGISTER AND MOVE service action (see 5.12.8) is moving a persistent reservation).

TransportIDs (see table 500) shall be at least 24 bytes long and shall be a multiple of four bytes in length.

Table 500 — TransportID format

Bit / Byte	7	6	5	4	3	2	1	0
0	TPID FORMAT		Reserved		PROTOCOL IDENTIFIER			
1	SCSI transport protocol specific data							
•••								
n								

The TransportID format (TPID FORMAT) field specifies the format of the TransportID. All TransportID format codes not defined in this standard (i.e., in 7.6.4) are reserved.

The PROTOCOL IDENTIFIER field (see table 479 in 7.6.1) specifies the SCSI transport protocol to which the TransportID applies.

The format of the SCSI transport protocol specific data depends on the value in the PROTOCOL IDENTIFIER field. The SCSI transport protocol specific data in a TransportID shall only include initiator port identifiers, initiator port names, or SCSI device names (see SAM-5) that persist across hard resets and I_T nexus losses. TransportID formats specific to SCSI transport protocols are listed in table 501.

Table 501 — TransportID formats for specific SCSI transport protocols

SCSI transport protocol	Protocol standard	Reference
Fibre Channel Protocol (FCP)	FCP-4	7.6.4.2
SCSI Parallel Interface (SPI)	SPI-5	7.6.4.3
Serial Bus Protocol (SBP) (i.e., IEEE 1394)	SBP-3	7.6.4.4
Remote Direct Memory Access (RDMA) (e.g., InfiniBand™)	SRP	7.6.4.5
Internet SCSI (iSCSI)	iSCSI	7.6.4.6
Serial Attached SCSI (SAS) Serial SCSI Protocol (SSP)	SPL-3	7.6.4.7
SCSI over PCI Express (SOP)	SOP	7.6.4.8

7.6.4.2 TransportID for initiator ports using SCSI over Fibre Channel

A Fibre Channel TransportID (see table 502) specifies an FCP-4 initiator port based on the N_Port_Name belonging to that initiator port.

Table 502 — Fibre Channel TransportID format

Bit / Byte	7	6	5	4	3	2	1	0
0	TPID FORMAT (00b)		Reserved		PROTOCOL IDENTIFIER (0h)			
1	Reserved							
•••								
7								
8	N_PORT_NAME							
•••								
15								
16	Reserved							
•••								
23								

The TPID FORMAT field and PROTOCOL IDENTIFIER field are defined in 7.6.4.1, and shall be set as shown in table 502 for the Fibre Channel TransportID format.

The N_PORT_NAME field shall contain the N_Port_Name defined by the N_Port login (PLOGI) extended link service (see FC-FS-3).

7.6.4.3 TransportID for initiator ports using a parallel SCSI bus

A parallel SCSI bus TransportID (see table 503) specifies a SPI-5 initiator port based on the SCSI address of an initiator port and the relative port identifier of the target port through which the application client accesses the SCSI target device.

Table 503 — Parallel SCSI bus TransportID format

Bit / Byte	7	6	5	4	3	2	1	0
0	TPID FORMAT (00b)		Reserved		PROTOCOL IDENTIFIER (1h)			
1	Reserved							
2	(MSB)	SCSI ADDRESS						
3								(LSB)
4	Obsolete							
5								
6	(MSB)	RELATIVE TARGET PORT IDENTIFIER						
7								(LSB)
8	Reserved							
•••								
23								

The TPID FORMAT field and PROTOCOL IDENTIFIER field are defined in 7.6.4.1, and shall be set as shown in table 503 for the Parallel SCSI bus TransportID format.

The SCSI ADDRESS field specifies the SCSI address (see SPI-5) of the initiator port.

The RELATIVE TARGET PORT IDENTIFIER field (see 4.3.4) specifies the relative port identifier of the target port for which the initiator port SCSI address applies. If the RELATIVE TARGET PORT IDENTIFIER does not reference a target port in the SCSI target device, the TransportID is invalid.

7.6.4.4 TransportID for initiator ports using SCSI over IEEE 1394

An IEEE 1394 TransportID (see table 504) specifies an SBP-3 initiator port based on the EUI-64 initiator port name belonging to that initiator port.

Table 504 — IEEE 1394 TransportID format

Bit / Byte	7	6	5	4	3	2	1	0
0	TPID FORMAT (00b)		Reserved		PROTOCOL IDENTIFIER (3h)			
1	Reserved							
•••								
7								
8	EUI-64 NAME							
•••								
15								
16	Reserved							
•••								
23								

The TPID FORMAT field and PROTOCOL IDENTIFIER field are defined in 7.6.4.1, and shall be set as shown in table 504 for the IEEE 1394 TransportID format.

The EUI-64 NAME field shall contain the EUI-64 IEEE 1394 node unique identifier (see SBP-3) for an initiator port.

7.6.4.5 TransportID for initiator ports using SCSI over an RDMA interface

A RDMA TransportID (see table 505) specifies an SRP initiator port based on the world wide unique initiator port name belonging to that initiator port.

Table 505 — RDMA TransportID format

Bit / Byte	7	6	5	4	3	2	1	0
0	TPID FORMAT (00b)		Reserved		PROTOCOL IDENTIFIER (4h)			
1	Reserved							
•••								
7								
8	INITIATOR PORT IDENTIFIER							
•••								
23								

The TPID FORMAT field and PROTOCOL IDENTIFIER field are defined in 7.6.4.1, and shall be set as shown in table 505 for the RDMA TransportID format.

The INITIATOR PORT IDENTIFIER field shall contain an SRP initiator port identifier (see SRP).

7.6.4.6 TransportID for initiator ports using SCSI over iSCSI

An iSCSI TransportID specifies an iSCSI initiator port using one of the TransportID formats listed in table 506.

Table 506 — iSCSI TPID FORMAT field codes

Code	Description	Reference
00b	Initiator port is identified using the world wide unique SCSI device name of the iSCSI initiator device containing the initiator port.	table 507
01b	Initiator port is identified using the world wide unique initiator port identifier.	table 508
10b to 11b	Reserved	

iSCSI TransportIDs with the TPID FORMAT field set to 01b should be processed. iSCSI TransportIDs with the TPID FORMAT field set to 00b may result in the command being terminated.

A iSCSI TransportID with the TPID FORMAT field set to 00b (see table 507) specifies an iSCSI initiator port based on the world wide unique SCSI device name of the iSCSI initiator device containing the initiator port.

Table 507 — iSCSI initiator device TransportID format

Bit / Byte	7	6	5	4	3	2	1	0
0	TPID FORMAT (00b)		Reserved		PROTOCOL IDENTIFIER (5h)			
1	Reserved							
2	(MSB)							
3	ADDITIONAL LENGTH (m-3)							(LSB)
4	(MSB)							
•••	ISCSI NAME							
m								(LSB)

The TPID FORMAT field and PROTOCOL IDENTIFIER field are defined in 7.6.4.1, and shall be set as shown in table 507 for the iSCSI initiator device TransportID format.

The ADDITIONAL LENGTH field specifies the number of bytes that follow in the TransportID. The additional length shall be at least 20 and shall be a multiple of four.

The null-terminated, null-padded (see 4.3.2) ISCSI NAME field shall contain the iSCSI name of an iSCSI initiator node (see RFC 7143). The first ISCSI NAME field byte containing an ASCII null character terminates the ISCSI NAME field without regard for the specified length of the iSCSI TransportID or the contents of the ADDITIONAL LENGTH field.

The iSCSI name length does not exceed 223 bytes. The maximum length of the iSCSI TransportID is 228 bytes.

If a iSCSI TransportID with the TPID FORMAT field set to 00b appears in a PERSISTENT RESERVE OUT parameter list (see 6.16.3), then all initiator ports known to the device server with an iSCSI node name matching the one in the TransportID shall be registered.

If a iSCSI TransportID with the TPID FORMAT field set to 00b appears in an ACE access identifier (see 8.3.1.3.2), the logical units listed in the ACE shall be accessible to any initiator port with an iSCSI node name matching the value in the TransportID. The access controls coordinator shall reject any command that attempts to define more than one ACE with an iSCSI TransportID access identifier containing the same iSCSI name. The access controls coordinator shall terminate the command with CHECK CONDITION status, with

the sense key ILLEGAL REQUEST, and the additional sense code set to INVALID FIELD IN PARAMETER LIST.

A iSCSI TransportID with the TPID FORMAT field set to 01b (see table 508) specifies an iSCSI initiator port based on its world wide unique initiator port identifier.

Table 508 — iSCSI initiator port TransportID format

Bit / Byte	7	6	5	4	3	2	1	0
0	TPID FORMAT (01b)		Reserved		PROTOCOL IDENTIFIER (5h)			
1	Reserved							
2	(MSB)							
3	ADDITIONAL LENGTH (m-3)							(LSB)
4	(MSB)							
•••	ISCSI NAME							
n-1								(LSB)
n	(MSB)							
•••	SEPARATOR (2C 692C 3078h)							
n+4								(LSB)
n+5	(MSB)							
•••	ISCSI INITIATOR SESSION ID							
m								(LSB)

The TPID FORMAT field and PROTOCOL IDENTIFIER field are defined in 7.6.4.1, and shall be set as shown in table 508 for the iSCSI initiator port TransportID format.

The ADDITIONAL LENGTH field specifies the number of bytes that follow in the TransportID encompassing the ISCSI NAME, SEPARATOR, and ISCSI INITIATOR SESSION ID fields. The additional length shall be at least 20 and shall be a multiple of four.

The ISCSI NAME field shall contain the iSCSI name of an iSCSI initiator node (see RFC 7143). The ISCSI NAME field shall not be null-terminated (see 4.3.2) and shall not be padded.

The SEPARATOR field shall be set as shown in table 508 (i.e., the five ASCII characters ‘,i,0x’).

The null-terminated, null-padded ISCSI INITIATOR SESSION ID field shall contain the iSCSI initiator session identifier (see RFC 7143) in the form of ASCII characters that are the hexadecimal digits converted from the binary iSCSI initiator session identifier value. The first ISCSI INITIATOR SESSION ID field byte containing an ASCII null character terminates the ISCSI INITIATOR SESSION ID field without regard for the specified length of the iSCSI TransportID or the contents of the ADDITIONAL LENGTH field.

7.6.4.7 TransportID for initiator ports using SCSI over SAS Serial SCSI Protocol

A SAS Serial SCSI Protocol (SSP) TransportID (see table 509) specifies a SAS initiator port that is communicating via SSP using the SAS address belonging to that initiator port.

Table 509 — SAS Serial SCSI Protocol TransportID format

Bit / Byte	7	6	5	4	3	2	1	0
0	TPID FORMAT (00b)			Reserved		PROTOCOL IDENTIFIER (6h)		
1	Reserved							
•••								
3								
4	SAS ADDRESS							
•••								
11								
12	Reserved							
•••								
23								

The TPID FORMAT field and PROTOCOL IDENTIFIER field are defined in 7.6.4.1, and shall be set as shown in table 509 for the SAS Serial SCSI Protocol TransportID format.

The SAS ADDRESS field specifies the SAS address of the SCSI initiator port (see SPL-3).

7.6.4.8 TransportID for initiator ports using SCSI over PCI Express

A SCSI over PCI Express (SOP) TransportID (see table 510) specifies a SOP initiator port.

Table 510 — SOP TransportID format

Bit / Byte	7	6	5	4	3	2	1	0
0	TPID FORMAT (00b)			Reserved		PROTOCOL IDENTIFIER (Ah)		
1	Reserved							
2	ROUTING ID							
3								
4	Reserved							
•••								
23								

The TPID FORMAT field and PROTOCOL IDENTIFIER field are defined in 7.6.4.1, and shall be set as shown in table 510 for the SOP TransportID format.

The ROUTING ID field shall contain a PCI Express routing ID (see SOP).

7.7 Security protocol parameters

7.7.1 Security protocol information description

7.7.1.1 Overview

The security protocol information security protocol (i.e., the SECURITY PROTOCOL field set to 00h in a SECURITY PROTOCOL IN command) returns security protocol related information. A SECURITY PROTOCOL IN command in which the SECURITY PROTOCOL field is set to 00h is not associated with an previous SECURITY PROTOCOL OUT command and shall be processed without regard for whether a SECURITY PROTOCOL OUT command has been processed.

If the SECURITY PROTOCOL IN command is supported, the SECURITY PROTOCOL field set to 00h shall be supported as defined in this standard.

7.7.1.2 CDB description

If the SECURITY PROTOCOL field is set to 00h in a SECURITY PROTOCOL IN command, the contents of the SECURITY PROTOCOL SPECIFIC field are defined in table 511.

Table 511 — SECURITY PROTOCOL SPECIFIC field for SECURITY PROTOCOL IN protocol 00h

Code	Description	Support	Reference
0000h	Supported security protocol list	Mandatory	7.7.1.3
0001h	Certificate data	Mandatory	7.7.1.4
0002h	Security compliance information	Optional	7.7.1.5
all others	Reserved		

All other CDB fields for SECURITY PROTOCOL IN command shall meet the requirements stated in 6.40.

Each time a SECURITY PROTOCOL IN command with the SECURITY PROTOCOL field set to 00h is received, the device server shall transfer the data defined in 7.7.1 starting with byte 0.

7.7.1.3 Supported security protocols list description

If the SECURITY PROTOCOL field is set to 00h and the SECURITY PROTOCOL SPECIFIC field is set to 0000h in a SECURITY PROTOCOL IN command, then the parameter data shall have the format shown in table 512.

Table 512 — Supported security protocols SECURITY PROTOCOL IN parameter data

Bit / Byte	7	6	5	4	3	2	1	0
0	Reserved							
•••								
5								
6	(MSB)			SUPPORTED SECURITY PROTOCOL LIST LENGTH (m-7)				
7								(LSB)
Supported security protocol list								
8	SUPPORTED SECURITY PROTOCOL (00h) [first]							
	⋮							
m	SUPPORTED SECURITY PROTOCOL [last]							
m+1	Pad bytes (if any)							
•••								
n								

The SUPPORTED SECURITY PROTOCOL LIST LENGTH field indicates the total length, in bytes, of the supported security protocol list that follows.

Each SUPPORTED SECURITY PROTOCOL field in the supported security protocols list shall contain one of the security protocol values (see table 317 in 6.40 and table 319 in 6.41) supported by the logical unit. The values shall be listed in ascending order starting with 00h.

The total data length shall conform to the ALLOCATION LENGTH field requirements (see 6.40). Pad bytes may be appended to meet this length. Pad bytes shall have a value of 00h.

7.7.1.4 Certificate data description

7.7.1.4.1 Certificate overview

A certificate is either an X.509 Public Key Certificate (see 7.7.1.4.2) or an X.509 Attribute Certificate (see 7.7.1.4.3) depending on the capabilities of the logical unit.

If the SECURITY PROTOCOL field is set to 00h and the SECURITY PROTOCOL SPECIFIC field is set to 0001h in a SECURITY PROTOCOL IN command, then the parameter data shall have the format shown in table 513.

Table 513 — Certificate data SECURITY PROTOCOL IN parameter data

Bit / Byte	7	6	5	4	3	2	1	0
0	Reserved							
1								
2	(MSB)			CERTIFICATE LENGTH (m-3)				
3								(LSB)
4	CERTIFICATE							
•••								
m								
m+1	Pad bytes (if any)							
•••								
n								

The CERTIFICATE LENGTH field indicates the total length, in bytes, of the certificate or certificates that follow. The length may include more than one certificate. If the device server doesn't have a certificate to transfer, the CERTIFICATE LENGTH field shall be set to 0000h.

The contents of the CERTIFICATE field are defined in 7.7.1.4.2 and 7.7.1.4.3.

The total data length shall conform to the ALLOCATION LENGTH field requirements (see 6.40). Pad bytes may be appended to meet this length. Pad bytes shall have a value of 00h.

7.7.1.4.2 Public Key certificate description

RFC 5280 and RFC 6818 define the certificate syntax for certificates consistent with X.509v3 Public Key Certificate Specification.

7.7.1.4.3 Attribute certificate description

RFC 5755 defines the certificate syntax for certificates consistent with X.509v2 Attribute Certificate Specification.

7.7.1.5 Security compliance information description

7.7.1.5.1 Security compliance information overview

The security compliance information parameter data contains information about security standards that apply to this SCSI target device.

If the SECURITY PROTOCOL field is set to 00h and the SECURITY PROTOCOL SPECIFIC field is set to 0002h in a SECURITY PROTOCOL IN command, then the parameter data shall have the format shown in table 514.

Table 514 — Security compliance information SECURITY PROTOCOL IN parameter data

Bit / Byte	7	6	5	4	3	2	1	0
0	(MSB)							
•••	SECURITY COMPLIANCE INFORMATION LENGTH (m-3)							
3								(LSB)
Compliance descriptors								
4								
•••	Compliance descriptor [first]							
	⋮							
•••	Compliance descriptor [last]							
n								
m+1								
•••	Pad bytes (if any)							
n								

The SECURITY COMPLIANCE INFORMATION LENGTH field indicates the total length, in bytes, of the compliance descriptors that follow.

Each compliance descriptor (see 7.7.1.5.2) contains information about a security standard that applies to this SCSI target device. Compliance descriptors may be returned in any order.

The total data length shall conform to the ALLOCATION LENGTH field requirements (see 6.40). Pad bytes may be appended to meet this length. Pad bytes shall have a value of 00h.

7.7.1.5.2 Compliance descriptor overview

The format of a compliance descriptor in the security compliance information SECURITY PROTOCOL IN parameter data is shown in table 515.

Table 515 — Compliance descriptor format

Bit / Byte	7	6	5	4	3	2	1	0
0	(MSB)							
1				COMPLIANCE DESCRIPTOR TYPE				(LSB)
2								
3				Reserved				
4	(MSB)							
•••				COMPLIANCE DESCRIPTOR LENGTH (n-3)				
7								(LSB)
8								
•••				Descriptor specific information				
n								

The COMPLIANCE DESCRIPTOR TYPE field (see table 516) indicates the format of the descriptor specific information. The security compliance information SECURITY PROTOCOL IN parameter data may contain more than one compliance descriptor with the same value in the COMPLIANCE DESCRIPTOR TYPE field.

Table 516 — COMPLIANCE DESCRIPTOR TYPE field

Code	Description	Related standards	Reference
0001h	Security requirements for cryptographic modules	FIPS 140-2 FIPS 140-3	7.7.1.5.3
all others	Reserved		

The COMPLIANCE DESCRIPTOR LENGTH field indicates the number of bytes that follow in the compliance descriptor.

The contents of the descriptor specific information depend on the value in the COMPLIANCE DESCRIPTOR TYPE field.

7.7.1.5.3 FIPS 140 compliance descriptor

The FIPS 140 compliance descriptor (see table 517) contains information that may be used to locate information about a FIPS 140 certificate associated with the SCSI target device. The SCSI target device may or may not be operating in the mode specified by that certificate.

Table 517 — FIPS 140 compliance descriptor

Bit / Byte	7	6	5	4	3	2	1	0
0	(MSB)							
1				COMPLIANCE DESCRIPTOR TYPE (0001h)				(LSB)
2								
3				Reserved				
4	(MSB)							
•••				COMPLIANCE DESCRIPTOR LENGTH (0000 0208h)				
7								(LSB)
8				RELATED STANDARD				
9				OVERALL SECURITY LEVEL				
10								
•••				Reserved				
15								
16	(MSB)							
•••				COMPLIANCE DESCRIPTOR HARDWARE VERSION				
143								(LSB)
144	(MSB)							
•••				COMPLIANCE DESCRIPTOR VERSION				
271								(LSB)
272	(MSB)							
•••				COMPLIANCE DESCRIPTOR MODULE NAME				
527								(LSB)

The COMPLIANCE DESCRIPTOR TYPE field and COMPLIANCE DESCRIPTOR LENGTH field are defined in 7.7.1.5.2 and shall be set as shown in table 517 for the FIPS 140 compliance descriptor.

The RELATED STANDARD field (see table 518) is an ASCII data field (see 4.3.1) that indicates the related standard described by this compliance descriptor.

Table 518 — RELATED STANDARD field

Code	Related standard
32h	FIPS 140-2
33h	FIPS 140-3
all others	Reserved

The OVERALL SECURITY LEVEL field is an ASCII data field (see 4.3.1) that indicates the FIPS 140 overall security level that is reported by NIST or CSEC.

The COMPLIANCE DESCRIPTOR HARDWARE VERSION field is null terminated, null padded data (see 4.3.2) that indicates the version number of the hardware in the module, as reported by NIST or CSEC.

The COMPLIANCE DESCRIPTOR VERSION field is null terminated, null padded data (see 4.3.2) that indicates the version number of the firmware or software in the module, as reported by NIST or CSEC. The value in the COMPLIANCE DESCRIPTOR VERSION field is not related to the PRODUCT REVISION LEVEL field of standard INQUIRY data (see 6.6.2).

The COMPLIANCE DESCRIPTOR MODULE NAME field is null terminated, null padded data (see 4.3.2) that indicates the name or identifier of the cryptographic module, as reported by NIST or CSEC.

7.7.2 SA creation capabilities

7.7.2.1 Overview

If the SECURITY PROTOCOL field in a SECURITY PROTOCOL IN command (see 6.40) is set to 40h, then the command returns information related to the SA creation (see 5.13.2.3) capabilities provided by the device server.

The SA creation capabilities protocol is independent of any other SA creation protocols. If any SA creation protocols are supported (e.g., IKEv2-SCSI (see 5.13.4)), then the device server shall not refuse to process an SA creation capabilities SECURITY PROTOCOL IN command, and this processing shall not affect the state maintained for any SA creation CCS (e.g., an IKEv2-SCSI CCS) on any I_T_L nexus. Except for those cases where an SA creation capabilities SECURITY PROTOCOL IN command reports changed SA creation capabilities, processing of the command shall not affect the concurrent processing of any commands that are part of an SA creation CCS.

If any SA creation protocols are supported, the SA creation capabilities protocol shall be supported as described in 7.7.2.

The SA creation capabilities SECURITY PROTOCOL IN CDB requirements are described in 7.7.2.2.

As shown in table 519 (see 7.7.2.2), the format of the parameter data for a SA creation capabilities SECURITY PROTOCOL IN command depends on the value in the SECURITY PROTOCOL SPECIFIC field in the CDB.

7.7.2.2 SA creation capabilities CDB description

The SA creation capabilities SECURITY PROTOCOL IN CDB has the format defined in 6.40 with the additional requirements described in this subclause.

If the SECURITY PROTOCOL field is set to SA creation capabilities (i.e., 40h) in a SECURITY PROTOCOL IN command, the SECURITY PROTOCOL SPECIFIC field (see table 519) identifies the SA creation protocol (see 5.13.2.3) for which the device server shall return capability information.

Table 519 — SECURITY PROTOCOL SPECIFIC field for the SA creation capabilities

Code	Description	Parameter data format
0000h	Supported device server capabilities formats	7.7.2.3.1
0001h to 0100h	Reserved	
0101h	IKEv2-SCSI device server capabilities	7.7.2.3.2
0102h to EFFFh	Reserved	
F000h to FFFFh	Vendor Specific	

If an SA creation capabilities SECURITY PROTOCOL IN command is received with the INC_512 bit set to one, then the device server shall terminate the SECURITY PROTOCOL IN command with CHECK

CONDITION status, with the sense key set to ILLEGAL REQUEST, and the additional sense code set to INVALID FIELD IN CDB.

7.7.2.3 SA creation capabilities parameter data formats

7.7.2.3.1 Supported device server capabilities formats parameter data format

The supported device server capabilities formats parameter data (see table 520) indicates which device server capabilities parameter data formats are supported by the device server.

Table 520 — Supported device server capabilities formats parameter data

Bit / Byte	7	6	5	4	3	2	1	0
0	(MSB)							
•••				PARAMETER DATA LENGTH (n-3)				
3								(LSB)
Supported capabilities parameter data formats								
4	(MSB)							
5				CAPABILITIES PARAMETER DATA FORMAT (0000h) [first]				(LSB)
				⋮				
n-1	(MSB)							
n				CAPABILITIES PARAMETER DATA FORMAT [last]				(LSB)

The PARAMETER DATA LENGTH field indicates the number of bytes that follow in the parameter data.

Each CAPABILITIES PARAMETER DATA FORMAT field in the supported capabilities parameter data formats list shall contain one of the SECURITY PROTOCOL SPECIFIC field values (see table 519) supported by the device server. The values shall be listed in ascending order starting with 0000h.

7.7.2.3.2 IKEv2-SCSI device server capabilities parameter data format

The IKEv2-SCSI device server capabilities parameter data (see table 521) indicates the IKEv2 transforms (i.e., key exchange protocols and authentication protocols) supported by the device server for IKEv2-SCSI.

Table 521 — IKEv2-SCSI device server capabilities parameter data

Bit / Byte	7	6	5	4	3	2	1	0
0	(MSB)							
•••				PARAMETER DATA LENGTH (n-3)				
3								(LSB)
4								
•••				IKEv2-SCSI SA Creation Capabilities payload (see 7.7.3.5.12)				
n								

The PARAMETER DATA LENGTH field indicates the number of bytes that follow in the parameter data.

The IKEv2-SCSI SA Creation Capabilities payload (see 7.7.3.5.12) indicates the algorithms supported by IKEv2-SCSI in the Key Exchange step (see 5.13.4.6) and Authentication step (see 5.13.4.7).

The primary content of the IKEv2-SCSI device server capabilities SA creation capabilities parameter data is an IKEv2-SCSI payload (see 7.7.3.5) that is used by the device server and application client in the construction of other IKEv2-SCSI payloads (see 5.13.4).

7.7.3 IKEv2-SCSI

7.7.3.1 Overview

If the SECURITY PROTOCOL field in a SECURITY PROTOCOL OUT command (see 6.41) or a SECURITY PROTOCOL IN command (see 6.40) is set to 41h, then the command is part of an IKEv2-SCSI CCS (see 5.13.4) and is used to transfer IKEv2-SCSI protocol information to or from the device server.

In an IKEv2-SCSI CCS, a defined sequence of SECURITY PROTOCOL OUT commands and SECURITY PROTOCOL IN commands are sent by the application client and processed by the device server as summarized in 5.13.4.1.

The IKEv2-SCSI SECURITY PROTOCOL OUT CDB format is described in 7.7.3.3.

The IKEv2-SCSI SECURITY PROTOCOL IN CDB format is described in 7.7.3.2.

The IKEv2-SCSI SECURITY PROTOCOL OUT command and the IKEv2-SCSI SECURITY PROTOCOL IN command use the same parameter data format (see 7.7.3.4). The primary content of the IKEv2-SCSI parameter data is one or more IKE payloads (see 7.7.3.5).

If the IKEv2-SCSI SA creation protocol is supported (see 7.7.1), the SA creation capabilities protocol (see 7.7.2) shall also be supported.

7.7.3.2 IKEv2-SCSI SECURITY PROTOCOL IN CDB description

The IKEv2-SCSI SECURITY PROTOCOL IN CDB has the format defined in 6.40 with the additional requirements described in this subclause.

If the SECURITY PROTOCOL field is set to IKEv2-SCSI (i.e., 41h) in a SECURITY PROTOCOL IN command, the SECURITY PROTOCOL SPECIFIC field (see table 522) identifies the IKEv2-SCSI step (see 5.13.4.1) that the device server is to process. If the IKEv2-SCSI SA creation protocol is supported (see 7.7.1), the SECURITY PROTOCOL IN command support requirements are shown in table 522.

Table 522 — SECURITY PROTOCOL SPECIFIC field as defined by the IKEv2-SCSI SECURITY PROTOCOL IN command

Code	Description	Support	References	
			Usage	Data format
0000h to 00FFh	Restricted		RFC 4306	RFC 4306
0100h to 0101h	Reserved			
0102h	Key Exchange step	Mandatory	5.13.4.6.3	7.7.3.4
0103h	Authentication step	Mandatory	5.13.4.7.3	7.7.3.4
0104h to EFFFh	Reserved			
F000h to FFFFh	Vendor Specific			

If an IKEv2-SCSI SECURITY PROTOCOL IN command is received with the INC_512 bit set to one while the device server is maintaining state for an IKEv2-SCSI CCS on the I_T_L nexus on which the command was received (see 5.13.4.1), then:

a) the device server shall terminate the SECURITY PROTOCOL IN command with CHECK CONDITION status, with the sense key set to NOT READY, and the additional sense code set to LOGICAL UNIT NOT READY, SA CREATION IN PROGRESS; and
b) the device server shall continue the IKEv2-SCSI CCS by preparing to receive another SECURITY PROTOCOL IN command or SECURITY PROTOCOL OUT command.

If an IKEv2-SCSI SECURITY PROTOCOL IN command is received with the INC_512 bit set to one while the device server is not maintaining state for an IKEv2-SCSI CCS (see 5.13.4.1), then the device server shall terminate the SECURITY PROTOCOL IN command with CHECK CONDITION status, with the sense key set to ILLEGAL REQUEST, and the additional sense code set to INVALID FIELD IN CDB.

7.7.3.3 IKEv2-SCSI SECURITY PROTOCOL OUT CDB description

The IKEv2-SCSI SECURITY PROTOCOL OUT CDB has the format defined in 6.41 with the additional requirements described in this subclause.

If the SECURITY PROTOCOL field is set to IKEv2-SCSI (i.e., 41h) in a SECURITY PROTOCOL OUT command, the SECURITY PROTOCOL SPECIFIC field (see table 523) identifies the IKEv2-SCSI step (see 5.13.4.1) that the device server is to process. If the IKEv2-SCSI SA creation protocol is supported (see 7.7.1), the SECURITY PROTOCOL OUT command support requirements are shown in table 523.

Table 523 — SECURITY PROTOCOL SPECIFIC field as defined by the IKEv2-SCSI SECURITY PROTOCOL OUT command

Code	Description	Support	References	
			Usage	Data format
0000h to 00FFh	Restricted		RFC 4306	RFC 4306
0100h to 0101h	Reserved			
0102h	Key Exchange step	Mandatory	5.13.4.6.3	7.7.3.4
0103h	Authentication step	Mandatory	5.13.4.7.3	7.7.3.4
0104h	Delete operation	Mandatory	5.13.4.11	7.7.3.4
0105h to EFFFh	Reserved			
F000h to FFFFh	Vendor Specific			

If an IKEv2-SCSI SECURITY PROTOCOL OUT command is received with the INC_512 bit set to one while the device server is maintaining state for an IKEv2-SCSI CCS on the I_T_L nexus on which the command was received (see 5.13.4.1), then:

a) the device server shall terminate the SECURITY PROTOCOL OUT command with CHECK CONDITION status, with the sense key set to NOT READY, and the additional sense code set to LOGICAL UNIT NOT READY, SA CREATION IN PROGRESS; and
b) the device server shall continue the IKEv2-SCSI CCS by preparing to receive another SECURITY PROTOCOL OUT command or SECURITY PROTOCOL IN command.

If an IKEv2-SCSI SECURITY PROTOCOL OUT command is received with the INC_512 bit set to one while the device server is not maintaining state for an IKEv2-SCSI CCS (see 5.13.4.1), then the device server shall terminate the SECURITY PROTOCOL OUT command with CHECK CONDITION status, with the sense key set to ILLEGAL REQUEST, and the additional sense code set to INVALID FIELD IN CDB.

Any IKEv2-SCSI SECURITY PROTOCOL OUT command with a transfer length of up to 16 384 bytes shall not be terminated with an error due to the number of bytes to be transferred and processed.

7.7.3.4 IKEv2-SCSI parameter data format

Table 524 shows the parameter list format used by a SECURITY PROTOCOL OUT command and the parameter data format used by a SECURITY PROTOCOL IN command if the SECURITY PROTOCOL field is set to IKEv2-SCSI (i.e., 41h).

Table 524 — IKEv2-SCSI SECURITY PROTOCOL OUT command and SECURITY PROTOCOL IN command parameter data

Bit / Byte	7	6	5	4	3	2	1	0
IKEv2-SCSI header								
0	Restricted (see RFC 4306)							
•••								
3								
4	(MSB) IKE_SA APPLICATION CLIENT SAI							
•••								
7								(LSB)
8	Restricted (see RFC 4306)							
•••								
11								
12	(MSB) IKE_SA DEVICE SERVER SAI							
•••								
15								(LSB)
16	NEXT PAYLOAD							
17	MAJOR VERSION (2h)				MINOR VERSION			
18	EXCHANGE TYPE							
19	Reserved			INTTR	VERSION	RSPNS	Reserved	
20	(MSB) MESSAGE ID							
•••								
23								(LSB)
24	(MSB) IKE LENGTH (n+1)							
•••								
27								(LSB)
IKEv2-SCSI payloads								
28	IKEv2-SCSI payload (see 7.7.3.5) [first]							
•••								
	⋮							
•••	IKEv2-SCSI payload (see 7.7.3.5) [last]							
n								

The IKE_SA APPLICATION CLIENT SAI field specifies the value that is or is destined to become the AC_SAI SA parameter (see 5.13.2.2) in the generated SA (see 5.13.4.9). The AC_SAI is chosen by the application client to uniquely identify its representation of the SA that is being negotiated or managed (e.g., deleted).

If the device server receives an IKEv2-SCSI header with the IKE_SA APPLICATION CLIENT SAI field set to zero, then the error shall be processed as described in 7.7.3.8.

The application client should compare the IKE_SA APPLICATION CLIENT SAI field contents in any SECURITY PROTOCOL IN parameter data to the value that the application client is maintaining for the IKEv2-SCSI CCS or SA management. If the two values are not identical, the application client should abandon the IKEv2-SCSI CCS, if any, and notify the device server that the IKEv2-SCSI CCS, if any, is being abandoned as described in 5.13.4.10.

Except in the Key Exchange step SECURITY PROTOCOL OUT command (see 5.13.4.6.2), the IKE_SA DEVICE SERVER SAI field specifies the value that is or is destined to become the DS_SAI SA parameter in the generated SA. The DS_SAI is chosen by the device server in accordance with the requirements in 5.13.2.1 to uniquely identify its representation of the SA that is being negotiated. In the Key Exchange step SECURITY PROTOCOL OUT command the IKE_SA DEVICE SERVER SAI field is reserved.

The application client should compare the IKE_SA DEVICE SERVER SAI field contents in the Authentication step SECURITY PROTOCOL IN parameter data to the value that the application client is maintaining for the IKEv2-SCSI CCS. If the two values are not identical, the application client should abandon the IKEv2-SCSI CCS and notify the device server that the IKEv2-SCSI CCS is being abandoned as described in 5.13.4.10.

The device server shall validate the contents of the IKE_SA APPLICATION CLIENT SAI field and the IKE_SA DEVICE SERVER SAI field as shown in table 525.

Table 525 — IKEv2-SCSI header checking of SAIs

Contents of SECURITY PROTOCOL SPECIFIC field in SECURITY PROTOCOL OUT CDB	Expected contents for …		Device server action if expected field contents not received
	IKE_SA APPLICATION CLIENT SAI field	IKE_SA DEVICE SERVER SAI field	
0102h (i.e., Key Exchange step)	any value	reserved	No actions taken based on expected field contents.
0103h (i.e., Authentication step)	A match with the SAI values maintained for an IKEv2-SCSI CCS on the I_T_L nexus on which the command was received		The device server shall: a) terminate the SECURITY PROTOCOL OUT command with CHECK CONDITION status, with the sense key set to ILLEGAL REQUEST, and the additional sense code set to SA CREATION PARAMETER VALUE REJECTED; and b) continue the IKEv2-SCSI CCS by preparing to receive another Authentication step SECURITY PROTOCOL OUT command.
0104h (i.e., Delete operation)	If at least one IKEv2-SCSI CCS is being maintained for the I_T_L nexus on which the command was received, then: a) a match with the SAI values maintained for an IKEv2-SCSI CCS; or b) a match with the SAI values maintained for any active SA		The device server shall: a) terminate the SECURITY PROTOCOL OUT command with CHECK CONDITION status, with the sense key set to ILLEGAL REQUEST, and the additional sense code set to SA CREATION PARAMETER VALUE REJECTED; and b) continue the IKEv2-SCSI CCS by preparing to receive another Authentication step SECURITY PROTOCOL OUT command.
	If no IKEv2-SCSI CCS is being maintained for the I_T_L nexus on which the command was received, then a match with the SAI values maintained for any active SA		The device server shall terminate the SECURITY PROTOCOL OUT command with CHECK CONDITION status, with the sense key set to ILLEGAL REQUEST, and the additional sense code set to INVALID FIELD IN PARAMETER LIST.

The NEXT PAYLOAD field (see table 526) identifies the first IKEv2-SCSI payload that follows the IKEv2-SCSI header.

Table 526 — NEXT PAYLOAD field

Code	IKE Payload Name	Support requirements in SECURITY PROTOCOL …		Reference
		IN	OUT	
00h	No Next Payload	Mandatory		7.7.3.5.2
01h to 20h		Reserved		
21h	Security Association	Prohibited [a]		RFC 4306
22h	Key Exchange	Mandatory		7.7.3.5.3
23h	Identification – Application Client	Prohibited	Mandatory	7.7.3.5.4
24h	Identification – Device Server	Mandatory	Prohibited	7.7.3.5.4
25h	Certificate	Optional		7.7.3.5.5
26h	Certificate Request	Optional		7.7.3.5.6
27h	Authentication	Mandatory		7.7.3.5.7
28h	Nonce	Mandatory		7.7.3.5.8
29h	Notify	Prohibited	Mandatory	7.7.3.5.9
2Ah	Delete	Prohibited	Mandatory	7.7.3.5.10
2Bh	Vendor ID	Prohibited		RFC 4306
2Ch	Traffic Selector – Application Client	Prohibited		RFC 4306
2Dh	Traffic Selector – Device Server	Prohibited		RFC 4306
2Eh	Encrypted	Mandatory		7.7.3.5.11
2Fh	Configuration	Prohibited		RFC 4306
30h	Extensible Authentication	Prohibited		RFC 4306
31h to 7Fh		Restricted		RFC 4306
80h	IKEv2-SCSI SA Creation Capabilities	Mandatory		7.7.3.5.12
81h	IKEv2-SCSI SA Cryptographic Algorithms	Mandatory		7.7.3.5.13
82h	IKEv2-SCSI SAUT Cryptographic Algorithms	Mandatory		7.7.3.5.14
83h	IKEv2-SCSI Timeout Values	Mandatory		7.7.3.5.15
84h to BFh		Reserved		
C0h to FFh	Vendor Specific			

[a] The Security Association payload type value is not used in IKEv2-SCSI. The IKEv2-SCSI SA Cryptographic Algorithms payload (i.e., 81h) and IKEv2-SCSI SAUT Cryptographic Algorithms payload (i.e., 82h) are used instead.

The MAJOR VERSION field shall contain the value 2h. If a device server receives an IKEv2-SCSI header with a MAJOR VERSION field containing a value other than 2h, then the error shall be processed as described in 7.7.3.8.

The MINOR VERSION field is reserved.

The EXCHANGE TYPE field is reserved.

The initiator (INTTR) bit shall be set to:

a) one for SECURITY PROTOCOL OUT commands; and
b) zero for SECURITY PROTOCOL IN commands.

If a device server receives an IKEv2-SCSI header with the INTTR bit set to zero, the error shall be processed as described in 7.7.3.8.

If an application client receives an IKEv2-SCSI header with the INTTR bit set to one, it should abandon the IKEv2-SCSI CCS and notify the device server that the IKEv2-SCSI CCS is being abandoned as described in 5.13.4.10.

The VERSION bit is reserved.

The response (RSPNS) bit shall be set to:

a) zero for SECURITY PROTOCOL OUT commands; and
b) one for SECURITY PROTOCOL IN commands.

If a device server receives an IKEv2-SCSI header with the RSPNS bit set to one, the error shall be processed as described in 7.7.3.8.

If an application client receives an IKEv2-SCSI header with the RSPNS bit set to zero, it should abandon the IKEv2-SCSI CCS and notify the device server that the IKEv2-SCSI CCS is being abandoned as described in 5.13.4.10.

The MESSAGE ID field (see table 527) identifies the function of the parameter data.

Table 527 — MESSAGE ID field

Code	Description
0000 0000h	Key Exchange step SECURITY PROTOCOL OUT command (see 5.13.4.6.2) or SECURITY PROTOCOL IN command (see 5.13.4.6.3)
0000 0001h	Authentication step SECURITY PROTOCOL OUT command (see 5.13.4.7.2) or SECURITY PROTOCOL IN command (see 5.13.4.7.3)
0000 0002h	Delete operation in a SECURITY PROTOCOL OUT command
all others	Reserved

If the device server receives a SECURITY PROTOCOL OUT command with an invalid MESSAGE ID field in its IKEv2-SCSI header, then the error shall be processed as described in 7.7.3.8.

If the application client receives an invalid MESSAGE ID field in the parameter data for a SECURITY PROTOCOL IN command, then the application client should abandon the IKEv2-SCSI CCS and notify the device server that the IKEv2-SCSI CCS is being abandoned as described in 5.13.4.10.

The IKE LENGTH field specifies the total number of bytes in the parameter data, including the IKEv2-SCSI header and all the IKEv2-SCSI payloads.

NOTE 55 - The contents of the IKE LENGTH field differ from those found in most SCSI length fields. However, they are consistent with the IKEv2 usage (see RFC 4306).

Each IKEv2-SCSI payload (see 7.7.3.5) contains specific data related to the operation being performed. A specific combination of IKEv2-SCSI payloads is specified for each operation (e.g., Key Exchange) as summarized in 5.13.4.2. The Encryption payload (see 7.7.3.6.2) nests one set of IKEv2-SCSI payloads inside another.

Based on the contents of the SECURITY PROTOCOL SPECIFIC field in the SECURITY PROTOCOL OUT CDB, the device server shall:

a) validate the contents of the NEXT PAYLOAD field in the IKEv2-SCSI header as shown in table 528, before performing any decryption or integrity checking; and
b) validate that the number of instances of each next payload value shown in table 528 occur in the parameter data after the encrypted data, if any, is decrypted and integrity checked (i.e., if the NEXT PAYLOAD field in the IKEv2-SCSI header contains 2Eh, after the Encrypted payload is decrypted and integrity checked).

If in the parameter data for a SECURITY PROTOCOL OUT command the next payload values in the IKEv2-SCSI header and in the unencrypted payloads in the IKEv2-SCSI payloads (see table 524) do not meet the requirements shown in table 528 for the listed SECURITY PROTOCOL SPECIFIC field contents, then the error shall be processed as described in 7.7.3.8.

Based on the contents of the SECURITY PROTOCOL SPECIFIC field in the SECURITY PROTOCOL IN command, the device server shall:

a) place one of the next payload values allowed by table 528 in the NEXT PAYLOAD field in the IKEv2-SCSI header; and
b) include the number of instances of each next payload value shown in table 528 in the parameter data either before or after encryption, if applicable (i.e., if the NEXT PAYLOAD field in the IKEv2-SCSI header contains 2Eh, before the contents of the Encrypted payload are encrypted and integrity check value is computed).

Table 528 — Next payload values in SECURITY PROTOCOL OUT/IN parameter data (part 1 of 3)

Next payload value	Next payload value allowed in IKEv2-SCSI header	Number of times a next payload value is allowed
SECURITY PROTOCOL OUT command with SECURITY PROTOCOL SPECIFIC field set to 0102h (i.e., Key Exchange step)		**Authentication step not skipped (see 5.13.4.1)**
83h (i.e., IKEv2-SCSI Timeout Values)	Yes	1
81h (i.e., IKEv2-SCSI SA Cryptographic Algorithms)	Yes	1
82h (i.e., IKEv2-SCSI SAUT Cryptographic Algorithms)	No	0
22h (i.e., Key Exchange)	Yes	1
28h (i.e., Nonce)	Yes	1
00h (i.e., No Next Payload)	No	1
All other next payload codes	No	0
SECURITY PROTOCOL OUT command with SECURITY PROTOCOL SPECIFIC field set to 0102h (i.e., Key Exchange step)		**Authentication step skipped (see 5.13.4.1)**
83h (i.e., IKEv2-SCSI Timeout Values)	Yes	1
81h (i.e., IKEv2-SCSI SA Cryptographic Algorithms)	Yes	1
82h (i.e., IKEv2-SCSI SAUT Cryptographic Algorithms)	Yes	1
22h (i.e., Key Exchange)	Yes	1
28h (i.e., Nonce)	Yes	1
00h (i.e., No Next Payload)	No	1

Table 528 — Next payload values in SECURITY PROTOCOL OUT/IN parameter data (part 2 of 3)

Next payload value	Next payload value allowed in IKEv2-SCSI header	Number of times a next payload value is allowed
All other next payload codes	No	0
SECURITY PROTOCOL IN command with SECURITY PROTOCOL SPECIFIC field set to 0102h (i.e., Key Exchange step)		**Authentication step not skipped (see 5.13.4.1)**
81h (i.e., IKEv2-SCSI SA Cryptographic Algorithms)	Yes	1
82h (i.e., IKEv2-SCSI SAUT Cryptographic Algorithms)	No	0
22h (i.e., Key Exchange)	Yes	1
28h (i.e., Nonce)	Yes	1
26h (i.e., Certificate Request)	Yes	0 or more
00h (i.e., No Next Payload)	No	1
All other next payload codes	No	0
SECURITY PROTOCOL IN command with SECURITY PROTOCOL SPECIFIC field set to 0102h (i.e., Key Exchange step)		**Authentication step skipped (see 5.13.4.1)**
81h (i.e., IKEv2-SCSI SA Cryptographic Algorithms)	Yes	1
82h (i.e., IKEv2-SCSI SAUT Cryptographic Algorithms)	Yes	1
22h (i.e., Key Exchange)	Yes	1
28h (i.e., Nonce)	Yes	1
26h (i.e., Certificate Request)	Yes	0 or more
00h (i.e., No Next Payload)	No	1
All other next payload codes	No	0
SECURITY PROTOCOL OUT command with SECURITY PROTOCOL SPECIFIC field set to 0103h (i.e., Authentication step)		
2Eh (i.e., Encrypted)	Yes	1
23h (i.e., Identification – Application Client)	No	1
82h (i.e., IKEv2-SCSI SAUT Cryptographic Algorithms)	No	1
25h (i.e., Certificate)	No	0 or more
26h (i.e., Certificate Request)	No	0 or more
29h (i.e., Notify)	No	0 or 1
27h (i.e., Authentication)	No	1
00h (i.e., No Next Payload)	No	1
All other next payload codes	No	0

Table 528 — Next payload values in SECURITY PROTOCOL OUT/IN parameter data (part 3 of 3)

Next payload value	Next payload value allowed in IKEv2-SCSI header	Number of times a next payload value is allowed
SECURITY PROTOCOL IN command with SECURITY PROTOCOL SPECIFIC field set to 0103h (i.e., Authentication step)		
2Eh (i.e., Encrypted)	Yes	1
24h (i.e., Identification – Device Server)	No	1
82h (i.e., IKEv2-SCSI SAUT Cryptographic Algorithms)	No	1
25h (i.e., Certificate)	No	0 or more
27h (i.e., Authentication)	No	1
00h (i.e., No Next Payload)	No	1
All other next payload codes	No	0
SECURITY PROTOCOL OUT command with SECURITY PROTOCOL SPECIFIC field set to 0104h (i.e., Delete operation)		
2Eh (i.e., Encrypted)	Yes	1
2Ah (i.e., Delete)	No	1
00h (i.e., No Next Payload)	No	1
All other next payload codes	No	0

7.7.3.5 IKEv2-SCSI payloads

7.7.3.5.1 IKEv2-SCSI payload format

Each IKEv2-SCSI payload (see table 529) is composed of a header and data that is specific to the payload type.

Table 529 — IKEv2-SCSI payload format

Bit / Byte	7	6	5	4	3	2	1	0
IKEv2-SCSI payload header								
0	NEXT PAYLOAD							
1	CRIT	Reserved						
2	(MSB)	IKE PAYLOAD LENGTH (n+1)						
3								(LSB)
IKEv2-SCSI payload-specific data								
4	Payload-specific data							
•••								
n								

The NEXT PAYLOAD field identifies the IKEv2-SCSI payload that follows this IKEv2-SCSI payload using one of the code values shown in table 526 (see 7.7.3.4).

The device server shall set the critical (CRIT) bit to one in all IKEv2-SCSI payloads returned to the application client. The application client should set the CRIT bit to one in all IKEv2-SCSI payloads sent to the device server.

If a device server receives an IKEv2-SCSI payload that it does not recognize (e.g., an IKEv2-SCSI payload identified by a next payload value of 01h) with the CRIT bit set to one, then the error shall be processed as described in 7.7.3.8.

If an application client receives an IKEv2-SCSI payload that it does not recognize with the CRIT bit set to one, then the application client should abandon the IKEv2-SCSI CCS and notify the device server that it is abandoning the IKEv2-SCSI CCS as described in 5.13.4.10.

If an application client or device server receives an IKEv2-SCSI payload that it does not recognize with the CRIT bit set to zero, then it should use the NEXT PAYLOAD field and the IKE PAYLOAD LENGTH field to skip processing of the unrecognized IKEv2-SCSI payload and continue processing at the next IKEv2-SCSI payload.

The IKE PAYLOAD LENGTH field specifies the total number of bytes in the payload, including the IKEv2-SCSI payload header. The value in the IKE PAYLOAD LENGTH field is not required to be a multiple of two or four (i.e., no byte alignment is maintained among IKEv2-SCSI payloads).

NOTE 56 - The contents of the IKE PAYLOAD LENGTH field differ from those found in most SCSI length fields. However, they are consistent with the IKEv2 usage (see RFC 4306).

The format and contents of the IKEv2-SCSI payload-specific data depends on the value in the NEXT PAYLOAD field of:

a) the IKEv2-SCSI header (see 7.7.3.4), if this is the first IKEv2-SCSI payload in the parameter data; or
b) the previous IKEv2-SCSI payload, in all other cases.

7.7.3.5.2 No Next payload

A NEXT PAYLOAD field that is set to 00h (i.e., No Next payload) specifies that no more IKEv2-SCSI payloads follow the current payload. The IKEv2-SCSI No Next payload contains no bytes.

7.7.3.5.3 Key Exchange payload

The Key Exchange payload (see table 530) transfers Diffie-Hellman shared key exchange data between an application client and a device server or vice versa.

Table 530 — Key Exchange payload format

Bit / Byte	7	6	5	4	3	2	1	0
0	NEXT PAYLOAD							
1	CRIT	Reserved						
2	(MSB)	IKE PAYLOAD LENGTH (n+1)						
3								(LSB)
4	(MSB)	DIFFIE-HELLMAN GROUP NUMBER						
5								(LSB)
6		Reserved						
7								
8		KEY EXCHANGE DATA						
•••								
n								

The NEXT PAYLOAD field, CRIT bit, and IKE PAYLOAD LENGTH field are described in 7.7.3.5.1.

The DIFFIE-HELLMAN GROUP NUMBER field specifies the least significant 16 bits from ALGORITHM IDENTIFIER field in the D-H IKEv2-SCSI algorithm descriptor (see 7.7.3.6.5) in the IKEv2-SCSI SA Cryptographic Algorithms payload (see 7.7.3.5.13).

The KEY EXCHANGE DATA field specifies the sender's Diffie-Hellman public value for this key exchange. The binary representation of key exchange data is defined in the reference cited for the Diffie-Hellman group that is used (see table 555).

When a prime modulus (i.e., mod p) Diffie-Hellman group is used, the length of the Diffie-Hellman public value shall be equal to the length of the prime modulus over which the exponentiation was performed as shown in table 555 (see 7.7.3.6.5). Bits that are set to zero shall be prepended to the KEY EXCHANGE DATA field if necessary.

Diffie-Hellman exponential reuse and reuse of analogous Diffie-Hellman public values for Diffie-Hellman mechanisms not based on exponentiation:

a) should not be performed; and
b) if performed, shall be constrained (e.g., requirements for discarding Diffie-Hellman information) as defined in RFC 4306.

If Diffie-Hellman exponentials and public values are reused in IKEv2-SCSI, the associated random nonces shall not be reused and the new random nonces should have a large number of bits of entropy (see RFC 4306).

7.7.3.5.4 Identification – Application Client payload and Identification – Device Server payload

The Identification – Application Client payload (see table 531) transfers identification information from the application client to the device server. The Identification – Device Server payload (see table 531) transfers identification information from the device server to the application client.

Table 531 — Identification payload format

Bit / Byte	7	6	5	4	3	2	1	0
0	NEXT PAYLOAD							
1	CRIT	Reserved						
2	(MSB)	IKE PAYLOAD LENGTH (n+1)						
3								(LSB)
4	ID TYPE							
5	Reserved							
7								
8	IDENTIFICATION DATA							
•••								
n								

The NEXT PAYLOAD field, CRIT bit, and IKE PAYLOAD LENGTH field are described in 7.7.3.5.1.

The ID TYPE field describes the contents of the IDENTIFICATION DATA field and shall contain one of the named values shown in table 532.

Table 532 — ID TYPE field

Code	Name [a]	Contents of the IDENTIFICATION DATA field
09h	ID_DER_ASN1_DN	The value of a certificate subject field (see RFC 5280 and RFC 6818)
0Ah	ID_DER_ASN1_GN	The value of a name contained in a Subject Alternative Name (i.e., SubjectAltName) certificate extension (see RFC 5280 and RFC 6818)
0Bh	ID_KEY_ID	Arbitrary identity data (e.g., initiator port names, target port names, and SCSI device names)
0Ch	ID_FC_NAME	FC-SP-2 certificates that certify a Fibre Channel name as an identity to be used (see RFC 4595 and FC-SP-2)
all other values	Prohibited	

[a] See RFC 4306 and RFC 4595 for additional information about the associated identification data for these names.

The contents of the IDENTIFICATION DATA field depend on the value in the ID TYPE field.

If the Certificate payload is included in the parameter data, the identity in the Identification – Application Client payload or Identification – Device Server payload is not required to match anything in the Certificate payload (see RFC 4306). Based on information provided by a configuration method that is outside the scope of this standard, device servers and application clients are required to verify a match between:

a) the identity in an Identification payload; and

b) the subject name or subject alternative name in a Certificate payload that contains an X.509 Certificate (see 7.7.3.5.5).

If a device server receives an Identification – Application Client payload that does not conform to the requirements in RFC 4306 or the requirements in this subclause, then the IKEv2-SCSI CCS state maintained for the I_T_L nexus shall be abandoned as described in 7.7.3.8.3.

If an application client receives an Identification – Device Server payload that does not conform to the requirements in RFC 4306 or the requirements in this subclause, then the application client should abandon the IKEv2-SCSI CCS and notify the device server that it is abandoning the IKEv2-SCSI CCS as described in 5.13.4.10.

7.7.3.5.5 Certificate payload

The Certificate payload (see table 533) delivers a requested identity authentication certificate. The protocol for using Certificate payloads is described in 5.13.4.3.3.4.

Table 533 — Certificate payload format

Bit / Byte	7	6	5	4	3	2	1	0
0	NEXT PAYLOAD							
1	CRIT	Reserved						
2	(MSB)	IKE PAYLOAD LENGTH (n+1)						
3								(LSB)
4	CERTIFICATE ENCODING							
5	CERTIFICATE DATA							
•••								
n								

The NEXT PAYLOAD field, CRIT bit, and IKE PAYLOAD LENGTH field are described in 7.7.3.5.1.

The CERTIFICATE ENCODING field describes the contents of the CERTIFICATE DATA field and shall contain one of the values shown in table 534.

Table 534 — CERTIFICATE ENCODING field

Code	Description	Reference
00h	Reserved	
01h to 03h	Prohibited	Annex E
04h	X.509 Certificate - Signature	RFC 4306
05h to 0Ah	Prohibited	Annex E
0Bh	Raw RSA Key	RFC 4306 and RFC 4718
0Ch to 0Dh	Prohibited	Annex E
0Eh to C8h	Restricted	IANA
C9h to FFh	Reserved	

The contents of the CERTIFICATE DATA field depend on the value in the CERTIFICATE ENCODING field.

The relationship between the Certificate payload and the Identification payload is described in 7.7.3.5.4.

Device servers that support certificates should support a mechanism outside the scope of this standard for replacing certificates and have the ability to store more than one certificate to facilitate such replacements.

7.7.3.5.6 Certificate Request payload

The Certificate Request payload (see table 535) allows an application client or device server to request the use of certificates as part of identity authentication and to name one or more trust anchors (see RFC 4306) for the certificate verification process. The Certificate payload (see table 533) delivers a requested identity authentication certificate. The protocol for using Certificate Request payloads is described in 5.13.4.3.3.4.

Table 535 — Certificate Request payload format

Bit / Byte	7	6	5	4	3	2	1	0
0	NEXT PAYLOAD							
1	CRIT	Reserved						
2	(MSB)	IKE PAYLOAD LENGTH (n+1)						
3								(LSB)
4	CERTIFICATE ENCODING							
Certification authority list								
5	(MSB)	CERTIFICATE AUTHORITY [first]						
•••								
24								(LSB)
	⋮							
n-19	(MSB)	CERTIFICATE AUTHORITY [last]						
•••								
n								(LSB)

The NEXT PAYLOAD field, CRIT bit, and IKE PAYLOAD LENGTH field are described in 7.7.3.5.1.

The value in the CERTIFICATE ENCODING field (see table 534 in 7.7.3.5.5) indicates the type or format of certificate being requested. Multiple Certificate Request payloads may be included in the parameter data transferred by a single command. If the parameter data contains more than one Certificate Request payload, each Certificate Request payload should have a different value in the CERTIFICATE ENCODING field.

Each CERTIFICATION AUTHORITY field contains an indicator of a trusted authority for the certificate type indicated by the CERTIFICATE ENCODING field in this Certificate Request payload. The indicator is a SHA-1 hash of the public key of a trusted certification authority. The indicator is encoded as the SHA-1 hash of the Subject Public Key Info element from the trust anchor certificate (see RFC 5280 and RFC 6818).

Device servers that support certificates should support a mechanism outside the scope of this standard for replacing certification authority values, and shall have the ability to store one or more certification authority values to facilitate such replacements.

7.7.3.5.7 Authentication payload

The Authentication payload (see table 536) allows the application client and a device server to verify that the data transfers in their IKEv2-SCSI CCS have not be compromised by a man-in-the-middle attack (see 5.13.1.4).

Table 536 — Authentication payload format

Bit / Byte	7	6	5	4	3	2	1	0
0	NEXT PAYLOAD							
1	CRIT	Reserved						
2	(MSB)	IKE PAYLOAD LENGTH (n+1)						
3								(LSB)
4	AUTH METHOD							
5	Reserved							
7								
8	AUTHENTICATION DATA							
•••								
n								

The NEXT PAYLOAD field, CRIT bit, and IKE PAYLOAD LENGTH field are described in 7.7.3.5.1.

The AUTH METHOD field indicates the authentication algorithm to be applied to this Authentication payload. The AUTH METHOD field contains the least significant eight bits of the ALGORITHM IDENTIFIER field in an SA_AUTH_OUT algorithm descriptor or an SA_AUTH_IN algorithm descriptor (see 7.7.3.6.6) from the Key Exchange step (see 5.13.4.6).

If the contents of the AUTH METHOD field in the parameter data for the Authentication step SECURITY PROTOCOL OUT command (see 5.13.4.7.2) do not match the least significant eight bits in the ALGORITHM IDENTIFIER field in the SA_AUTH_OUT algorithm descriptor in the IKEv2-SCSI SA Cryptographic Algorithms payload (see 7.7.3.5.13) in the Key Exchange step, then the device server shall:

a) terminate the SECURITY PROTOCOL OUT command CHECK CONDITION status, with the sense key set to ABORTED COMMAND and the additional sense code set to AUTHENTICATION FAILED; and
b) abandon the IKEv2-SCSI CCS (see 5.13.4.10).

If the contents of the AUTH METHOD field in the parameter data for the Authentication step SECURITY PROTOCOL IN command (see 5.13.4.7.3) do not match the least significant eight bits in the ALGORITHM IDENTIFIER field in the SA_AUTH_IN algorithm descriptor in the IKEv2-SCSI SA Cryptographic Algorithms payload in the Key Exchange step, then application client should abandon the IKEv2-SCSI CCS and notify the device server that it is abandoning the IKEv2-SCSI CCS as described in 5.13.4.10.

In the Authentication step SECURITY PROTOCOL OUT command (see 5.13.4.7.2) parameter list, the AUTHENTICATION DATA field contains the result of applying the algorithm specified by the ALGORITHM IDENTIFIER field in the SA_AUTH_OUT IKEv2-SCSI cryptographic algorithm descriptor in the IKEv2-SCSI SA Cryptographic Algorithms payload (see 7.7.3.5.13) in the Key Exchange step (see 5.13.4.6) as described in table 557 (see 7.7.3.6.6) and this subclause to generate the following concatenation of bytes:

1) all the bytes in the Data-In Buffer returned by the Device Server Capabilities step (see 5.13.4.5) SECURITY PROTOCOL IN command;

2) all the bytes in the Data-Out Buffer sent by the Key Exchange step SECURITY PROTOCOL OUT command (see 5.13.4.6.2) in the same IKEv2-SCSI CCS for which GOOD status was returned;
3) all the bytes in the IKEv2-SCSI payload-specific data part (see 7.7.3.5.1) of the Nonce payload (see 7.7.3.5.8) that was received in the Key Exchange step SECURITY PROTOCOL IN command in the same IKEv2-SCSI CCS; and
4) all the bytes produced by applying the PRF selected by the PRF IKEv2-SCSI cryptographic algorithm descriptor (see 7.7.3.6.3) in the IKEv2-SCSI SA Cryptographic Algorithms payload (see 7.7.3.5.13) to the following inputs:
    1) the SK_pi shared key (see 5.13.4.4); and
    2) all the bytes in the IKEv2-SCSI payload-specific data part (see 7.7.3.5.1) of the Identification – Application Client payload (see 7.7.3.5.4).

While processing the Authentication step SECURITY PROTOCOL OUT command, the device server shall verify the contents of the AUTHENTICATION DATA field by applying the algorithm specified by the ALGORITHM IDENTIFIER field in the SA_AUTH_OUT IKEv2-SCSI cryptographic algorithm descriptor in the IKEv2-SCSI SA Cryptographic Algorithms payload (see 7.7.3.5.13) in the Key Exchange step (see 5.13.4.6) as described in table 557 (see 7.7.3.6.6) and this subclause to generate the following concatenation of bytes:

1) all the bytes in the Data-In Buffer that the device server returned to any application client in response to the last received Device Server Capabilities step SECURITY PROTOCOL IN command;
2) all the bytes in the Data-Out Buffer received in the Key Exchange step SECURITY PROTOCOL OUT command (see 5.13.4.6.2) in the same IKEv2-SCSI CCS for which GOOD status was returned;
3) all the bytes in the IKEv2-SCSI payload-specific data part (see 7.7.3.5.1) of the Nonce payload (see 7.7.3.5.8) sent in the Key Exchange step SECURITY PROTOCOL IN command in the same IKEv2-SCSI CCS; and
4) all the bytes produced by applying the PRF selected by the PRF IKEv2-SCSI cryptographic algorithm descriptor (see 7.7.3.6.3) in the IKEv2-SCSI SA Cryptographic Algorithms payload (see 7.7.3.5.13) to the following inputs:
    1) the SK_pi shared key (see 5.13.4.4); and
    2) all the bytes received in the IKEv2-SCSI payload-specific data part (see 7.7.3.5.1) of the Identification – Application Client payload (see 7.7.3.5.4) of the parameter list being processed.

If the verification of the contents of the AUTHENTICATION DATA field is not successful, then the device server shall:

a) terminate the SECURITY PROTOCOL OUT command with CHECK CONDITION status, with the sense key set to ABORTED COMMAND and the additional sense code set to AUTHENTICATION FAILED; and
b) abandon the IKEv2-SCSI CCS (see 5.13.4.10).

For the Authentication step SECURITY PROTOCOL IN command (see 5.13.4.7.3) parameter list, the device server shall compute the AUTHENTICATION DATA field contents by applying the algorithm specified by the ALGORITHM IDENTIFIER field in the SA_AUTH_IN IKEv2-SCSI cryptographic algorithm descriptor in the IKEv2-SCSI SA Cryptographic Algorithms payload (see 7.7.3.5.13) in the Key Exchange step (see 5.13.4.6) as described in table 557 (see 7.7.3.6.6) and this subclause to generate the following concatenation of bytes:

1) all the bytes in the Data-In Buffer that the device server returned to any application client in response to the last received Device Server Capabilities step SECURITY PROTOCOL IN command;
2) all the bytes in the Data-In Buffer sent by the most recent Key Exchange step SECURITY PROTOCOL IN command (see 5.13.4.6.3) in the same IKEv2-SCSI CCS for which GOOD status was returned;
3) all the bytes in the IKEv2-SCSI payload-specific data part (see 7.7.3.5.1) of the Nonce payload (see 7.7.3.5.8) received in the Key Exchange step SECURITY PROTOCOL OUT command in the same IKEv2-SCSI CCS; and
4) all the bytes produced by applying the PRF selected by the PRF IKEv2-SCSI cryptographic algorithm descriptor (see 7.7.3.6.3) in the IKEv2-SCSI SA Cryptographic Algorithms payload (see 7.7.3.5.13) to the following inputs:

1) the SK_pr shared key (see 5.13.4.4); and
2) all the bytes in the IKEv2-SCSI payload-specific data part (see 7.7.3.5.1) of the Identification – Device Server payload (see 7.7.3.5.4).

After the Authentication step SECURITY PROTOCOL IN command completes with GOOD status, the application client should verify the contents of the AUTHENTICATION DATA field by applying the algorithm specified by the ALGORITHM IDENTIFIER field in the SA_AUTH_IN IKEv2-SCSI cryptographic algorithm descriptor in the IKEv2-SCSI SA Cryptographic Algorithms payload (see 7.7.3.5.13) in the Key Exchange step (see 5.13.4.6) as described in table 557 (see 7.7.3.6.6) and this subclause to generate the following concatenation of bytes:

1) all the bytes in the Data-In Buffer returned by the Device Server Capabilities step (see 5.13.4.5) SECURITY PROTOCOL IN command;
2) all the bytes in the Data-In Buffer returned by the most recent Key Exchange step SECURITY PROTOCOL IN command (see 5.13.4.6.3) in the same IKEv2-SCSI CCS for which GOOD status was returned;
3) all the bytes in the IKEv2-SCSI payload-specific data part (see 7.7.3.5.1) of the Nonce payload (see 7.7.3.5.8) sent in the Key Exchange step SECURITY PROTOCOL OUT command in the same IKEv2-SCSI CCS; and
4) all the bytes produced by applying the PRF selected by the PRF IKEv2-SCSI cryptographic algorithm descriptor (see 7.7.3.6.3) in the IKEv2-SCSI SA Cryptographic Algorithms payload (see 7.7.3.5.13) to the following inputs:
   1) the SK_pr shared key (see 5.13.4.4); and
   2) all the bytes in the IKEv2-SCSI payload-specific data part (see 7.7.3.5.1) of the Identification – Device Server payload (see 7.7.3.5.4) received in the SECURITY PROTOCOL IN parameter data.

If the verification of the contents of the AUTHENTICATION DATA field is not successful, then the application client should abandon the IKEv2-SCSI CCS and notify the device server that it is abandoning the IKEv2-SCSI CCS as described in 5.13.4.10.

If the AUTH METHOD field is set to 01h (i.e., RSA Digital Signature), the RSA digital signature shall be encoded with the EMSA-PKCS1-v1_5 signature encoding method as defined in RFC 2437 (see RFC 4718).

7.7.3.5.8 Nonce payload

The Nonce payload (see table 537) transfers one random nonce from the application client to the device server or from the device server to the application client.

Table 537 — Nonce payload format

Bit / Byte	7	6	5	4	3	2	1	0
0	NEXT PAYLOAD							
1	CRIT	Reserved						
2	(MSB)	IKE PAYLOAD LENGTH (n+1)						
3								(LSB)
4	NONCE DATA							
•••								
n								

The NEXT PAYLOAD field, CRIT bit, and IKE PAYLOAD LENGTH field are described in 7.7.3.5.1.

In the Key Exchange step SECURITY PROTOCOL OUT command (see 5.13.4.6.2) the NONCE DATA field specifies the application client's random nonce.

In the Key Exchange step SECURITY PROTOCOL IN command (see 5.13.4.6.3) the NONCE DATA field indicates the device server's random nonce.

The requirements that RFC 4306 places on the nonce data shall apply to this standard.

7.7.3.5.9 Notify payload

This standard uses the Notify payload (see table 538) to provide initial contact notification from the application client to the device server. See Annex E for information about differences between this standard and IKEv2.

Table 538 — Notify payload format

Bit / Byte	7	6	5	4	3	2	1	0
0	NEXT PAYLOAD							
1	CRIT	Reserved						
2	(MSB)	IKE PAYLOAD LENGTH (0010h)						
3								(LSB)
4	PROTOCOL ID (01h)							
5	SAI SIZE (08h)							
6	(MSB)	NOTIFY MESSAGE TYPE (4000h)						
7								(LSB)
8	Restricted (see RFC 4306)							
•••								
11								
12	(MSB)	SAI						
•••								
15								(LSB)

The NEXT PAYLOAD field, CRIT bit, and IKE PAYLOAD LENGTH field are described in 7.7.3.5.1.

The PROTOCOL ID field is set as shown in table 538 for the IKEv2-SCSI Notify payload. If the device server receives a PROTOCOL ID field that is not set as shown in table 538, the IKEv2-SCSI CCS state maintained for the I_T_L nexus shall be abandoned as described in 7.7.3.8.3.

The SAI SIZE field is set as shown in table 538 for the IKEv2-SCSI Notify payload. If the device server receives an SAI SIZE field that is not set as shown in table 538, the IKEv2-SCSI CCS state maintained for the I_T_L nexus shall be abandoned as described in 7.7.3.8.3.

The NOTIFY MESSAGE TYPE field is set as shown in table 538 (i.e., INITIAL_CONTACT) for the IKEv2-SCSI Notify payload. If the device server receives a NOTIFY MESSAGE TYPE field that is not set as shown in table 538, the IKEv2-SCSI CCS state maintained for the I_T_L nexus shall be abandoned as described in 7.7.3.8.3.

The SAI field specifies the device server's SAI. If the contents of the SAI field are not identical to the contents of the IKE_SA DEVICE SERVER SAI field in the IKEv2-SCSI header (see 7.7.3.4), then the IKEv2-SCSI CCS state maintained for the I_T_L nexus shall be abandoned as described in 7.7.3.8.3.

Unless an error is detected, the device server shall process the Notify payload as described in 5.13.4.7.2.

7.7.3.5.10 Delete payload

The Delete payload (see table 539) requests the deletion of an existing SA or the abandonment of an IKEv2-SCSI CCS that is in progress.

Table 539 — Delete payload format

Bit / Byte	7	6	5	4	3	2	1	0
0	NEXT PAYLOAD							
1	CRIT	Reserved						
2	(MSB)	IKE PAYLOAD LENGTH (0018h)						
3								(LSB)
4	PROTOCOL ID (01h)							
5	SAI SIZE (08h)							
6	(MSB)	NUMBER OF SAIS (0002h)						
7								(LSB)
8	Restricted (see RFC 4306)							
•••								
11								
12	(MSB)	AC_SAI						
•••								
15								(LSB)
16	Restricted (see RFC 4306)							
•••								
19								
20	(MSB)	DS_SAI						
•••								
23								(LSB)

The NEXT PAYLOAD field, CRIT bit, and IKE PAYLOAD LENGTH field are described in 7.7.3.5.1.

The PROTOCOL ID field is set as shown in table 539 for the IKEv2-SCSI Delete payload. If the device server receives a PROTOCOL ID field that is not set as shown in table 539, the error shall be processed as described in 7.7.3.8.3.

The SAI SIZE field is set as shown in table 539 for the IKEv2-SCSI Delete payload. If the device server receives an SAI SIZE field that is not set as shown in table 539, the error shall be processed as described in 7.7.3.8.3.

The NUMBER OF SAIS field is set as shown in table 539 for the IKEv2-SCSI Delete payload. If the device server receives a NUMBER OF SAIS field that is not set as shown in table 539, the error shall be processed as described in 7.7.3.8.3.

The AC_SAI field specifies the AC_SAI SA parameter value for the SA to be deleted. If the contents of the AC_SAI field do not match the contents of the IKE_SA APPLICATION CLIENT SAI field in the IKEv2-SCSI header (see 7.7.3.4), then the error shall be processed as described in 7.7.3.8.3.

The DS_SAI field specifies the DS_SAI SA parameter value for the SA to be deleted. If the contents of the DS_SAI field do not match the contents of the IKE_SA DEVICE SERVER SAI field in the IKEv2-SCSI header (see 7.7.3.4), then the error shall be processed as described in 7.7.3.8.3.

If the device server is maintaining SA parameters for which the AC_SAI matches the contents of the AC_SAI field and the DS_SAI matches the contents of the DS_SAI field, then that set of SA parameters shall be deleted.

If the device server is maintaining state for an IKEv2-SCSI CCS on the I_T_L nexus on which the command was received, then the IKEv2-SCSI CCS shall be abandoned (see 5.13.4.10) if:

a) the contents of the AC_SAI field match the application client's SAI in the maintained state; and
b) the contents of the DS_SAI field match the device server's SAI in the maintained state.

7.7.3.5.11 Encrypted payload

7.7.3.5.11.1 Combined mode encryption

The following types of encryption algorithms are used in the Encrypted payload:

a) non-combined modes that use separate algorithms to encrypt/decrypt and integrity check the Encrypted payload; and
b) combined modes in which a single encryption algorithm does both encryption/decryption and integrity checking.

The ALGORITHM IDENTIFIER field in the INTEG IKEv2-SCSI cryptographic algorithm descriptor (see 7.7.3.6.4) in the IKEv2-SCSI SA Cryptographic Algorithms payload (see 7.7.3.5.13) in the Key Exchange step (see 5.13.4.6) indicates the type of encryption algorithm to be applied to the Encrypted payload for IKEv2-SCSI functions as follows:

a) if the ALGORITHM IDENTIFIER field is not set to AUTH_COMBINED, a non-combined mode encryption algorithm is being used; or
b) if the ALGORITHM IDENTIFIER field is set to AUTH_COMBINED, a combined mode encryption algorithm is being used.

If the Encrypted payload is in parameter data that is not associated with an active IKEv2-SCSI CCS, then the integrity checking algorithm identifier that selects between combined and non-combined mode encryption is found in the MGMT_DATA SA parameter (see 5.13.4.9).

7.7.3.5.11.2 Encrypted payload introduction

The Encrypted payload transfers (see table 540) one or more other IKEv2-SCSI payloads that are encrypted and integrity checked from the application client to the device server and vice versa.

If IKEv2-SCSI parameter data contains the Encrypted payload, then the Encrypted payload is the first payload in the parameter data (i.e., the NEXT PAYLOAD field in the IKEv2-SCSI header (see 7.7.3.5.1) contains 2Eh). Since the NEXT PAYLOAD field in an Encrypted payload identifies the first payload in the CIPHERTEXT field, there is no way to identify a payload following the Encrypted payload. As a result, the Encrypted payload is required to be the last payload in the IKEv2-SCSI payloads part of the parameter data (see 7.7.3.4).

Table 540 — Encrypted payload format

Bit / Byte	7	6	5	4	3	2	1	0
0	NEXT PAYLOAD							
1	CRIT	Reserved						
2	(MSB)	IKE PAYLOAD LENGTH (n+1)						
3								(LSB)
4	INITIALIZATION VECTOR							
•••								
i-1								
i	CIPHERTEXT							
•••								
k-1								
k	INTEGRITY CHECK VALUE							
•••								
n								

The NEXT PAYLOAD field identifies the first IKEv2-SCSI payload in the CIPHERTEXT field using the coded values shown in table 526 (see 7.7.3.5.1).

The CRIT bit and IKE PAYLOAD LENGTH field are described in 7.7.3.5.1.

The IKE PAYLOAD LENGTH field specifies the number of bytes that follow in the Encrypted payload. The number of bytes in the CIPHERTEXT field is equal to the number of bytes in the plaintext (see table 541).

The INITIALIZATION VECTOR field specifies the initialization vector encryption algorithm input value. The size of the initialization vector is defined by the encryption algorithm as shown in table 549 (see 7.7.3.6.2).

The CIPHERTEXT field contains an output of the encryption algorithm specified by the ALGORITHM IDENTIFIER field in the ENCR IKEv2-SCSI cryptographic algorithm descriptor (see 7.7.3.6.2) in the IKEv2-SCSI SA Cryptographic Algorithms payload (see 7.7.3.5.13) in the Key Exchange step (see 5.13.4.6) using the inputs:

a) IKEv2-SCSI AAD is an encryption algorithm input as follows:
   A) if a non-combined mode encryption algorithm is being used (see 7.7.3.5.11.1), then there is no AAD input; or
   B) if a combined mode encryption algorithm is being used (see 7.7.3.5.11.1), then the AAD described in 7.7.3.5.11.3 is an input;
b) the contents of the INITIALIZATION VECTOR field (see table 549);
c) the plaintext data shown in table 541; and
d) the shared key value, key length, and salt bytes (see table 549 in 7.7.3.6.2) if any, for one of the following shared keys:

A) if the Encrypted payload appears in the parameter data for an Authentication step SECURITY PROTOCOL OUT command (see 5.13.4.7.2) or the parameter data for a Delete operation SECURITY PROTOCOL OUT command (see 5.13.4.11) that affects an active IKEv2-SCSI CCS, then the SK_ei shared key (see 5.13.4.4) for the IKEv2-SCSI CCS;
B) if the Encrypted payload appears in the parameter data for an Authentication step SECURITY PROTOCOL IN command (see 5.13.4.7.3), then the SK_er shared key (see 5.13.4.4) for the IKEv2-SCSI CCS; or
C) if the Encrypted payload appears in the parameter data for a Delete operation SECURITY PROTOCOL OUT command (see 5.13.4.11) that does not affect an active IKEv2-SCSI CCS, then the SK_ei shared key (see 5.13.4.4) from the MGMT_DATA SA parameter (see 5.13.4.9).

NOTE 57 - Salt bytes (see table 549 in 7.7.3.6.2) are used only by combined mode encryption algorithms (see 7.7.3.5.11.1).

If the Encrypted payload appears in the parameter data for a Delete operation SECURITY PROTOCOL OUT command that does not affect an active IKEv2-SCSI CCS, then the encryption algorithm identifier stored in the MGMT_DATA SA parameter indicates the encryption algorithm to use.

The INTEGRITY CHECK VALUE field contains the integrity check value that is computed as described in 7.7.3.5.11.1 (e.g., if the integrity algorithm is AUTH_COMBINED, then the integrity check value is an output of the encryption algorithm). The size of the integrity check value is defined by the integrity algorithm or encryption algorithm, depending on which algorithm computes the value.

If the integrity algorithm specified by the ALGORITHM IDENTIFIER field in the INTEG IKEv2-SCSI cryptographic algorithm descriptor in the IKEv2-SCSI SA Cryptographic Algorithms payload (see 7.7.3.5.13) in the Key Exchange step is not AUTH_COMBINED, then the contents of the INTEGRITY CHECK VALUE field are computed by processing the integrity check algorithm specified by the ALGORITHM IDENTIFIER field in the INTEG IKEv2-SCSI cryptographic algorithm descriptor using the following inputs:

a) a byte string composed of:
   1) the AAD described in 7.7.3.5.11.3;
   2) the contents of the INITIALIZATION VECTOR field (see table 540); and
   3) the ciphertext that is the result of encrypting the plaintext data;
   and
b) the shared key value from one of the following shared keys:
   A) if the Encrypted payload appears in the parameter data for an Authentication step SECURITY PROTOCOL OUT command (see 5.13.4.7.2) or the parameter data for a Delete operation SECURITY PROTOCOL OUT command (see 5.13.4.11) that affects an active IKEv2-SCSI CCS, then the SK_ai shared key (see 5.13.4.4) for the IKEv2-SCSI CCS;
   B) if the Encrypted payload appears in the parameter data for an Authentication step SECURITY PROTOCOL IN command (see 5.13.4.7.3), then the SK_ar shared key (see 5.13.4.4) for the IKEv2-SCSI CCS; or
   C) if the Encrypted payload appears in the parameter data for a Delete operation SECURITY PROTOCOL OUT command (see 5.13.4.11) that does not affect an active IKEv2-SCSI CCS, then the SK_ai shared key (see 5.13.4.4) from the MGMT_DATA SA parameter (see 5.13.4.9).

If the Encrypted payload appears in the parameter data for a Delete operation SECURITY PROTOCOL OUT command that does not affect an active IKEv2-SCSI CCS, then the integrity checking algorithm identifier stored in the MGMT_DATA SA parameter indicates the integrity checking algorithm to use.

While computing the encrypted CIPHERTEXT field contents for an Encrypted payload, the plaintext format shown in table 541 is used.

Table 541 — Plaintext format for Encrypted payload CIPHERTEXT field

Bit / Byte	7	6	5	4	3	2	1	0
IKEv2-SCSI payloads								
0	IKEv2-SCSI payload (see 7.7.3.5) [first]							
•••								
	⋮							
	IKEv2-SCSI payload (see 7.7.3.5) [last]							
•••								
n								
p	PADDING BYTES (if any)							
•••								
k-1								
k	PAD LENGTH (k-p)							

Each IKEv2-SCSI payload (see 7.7.3.5) contains specific data related to the operation being performed. A specific combination of IKEv2-SCSI payloads is required for each operation (e.g., Authentication) as summarized in 5.13.4.2.

The PADDING BYTES field contains zero to 255 bytes. The number of padding bytes is:

a) defined by the encryption algorithm; or
b) any number of bytes that causes the length of all plaintext bytes (i.e., k+1) to be a whole multiple of the alignment (see table 549 in 7.7.3.6.2) for the encryption algorithm being used.

The contents of the padding bytes are:

a) defined by the encryption algorithm; or
b) if the encryption algorithm does not define the padding bytes contents, a series of one byte binary values starting at one and incrementing by one in each successive byte (i.e., 01h in the first padding byte, 02h in the second padding byte, etc.).

If the encryption algorithm does not place requirements on the contents of the padding bytes (i.e., option b) is in effect), then after decryption the contents of the padding bytes shall be verified to match the series of one byte binary values described in this subclause. If this verification is not successful in a device server, the error shall be processed as described in 7.7.3.8.2. If this verification is not successful in an application client, the decrypted data should be ignored.

The PAD LENGTH field specifies the number of bytes in the PADDING BYTES field.

7.7.3.5.11.3 IKEv2-SCSI AAD

The AAD defined by this standard for IKEv2-SCSI use is as follows:

1) all the bytes in the IKEv2-SCSI Header (see 7.7.3.4); and
2) all the bytes in the IKEv2-SCSI Payload Header (see 7.7.3.5.1) of the Encrypted payload (see 7.7.3.5.11).

7.7.3.5.11.4 Processing a received Encrypted payload

Before performing any checks of data contained in the CIPHERTEXT field, the contents of the INTEGRITY CHECK VALUE field and CIPHERTEXT field shall be integrity checked and decrypted based on the contents of the IKE_SA APPLICATION CLIENT SAI field and the IKE_SA DEVICE SERVER SAI field in the IKEv2-SCSI header (see 7.7.3.4) as described in this subclause.

Computation of the comparison integrity check value and decryption of an Encrypted payload is performed as follows:

1) if a non-combined mode encryption algorithm is being used (see 7.7.3.5.11.1), then the comparison integrity check value is computed by performing the integrity check algorithm specified by the ALGORITHM IDENTIFIER field in the INTEG IKEv2-SCSI cryptographic algorithm descriptor using the following inputs:
   A) a byte string composed of:
      1) the AAD described in 7.7.3.5.11.3;
      2) the contents of the INITIALIZATION VECTOR field (see table 540) in the Encrypted payload; and
      3) the contents of the CIPHERTEXT field (see table 540) in the Encrypted payload;

      and
   B) the shared key value from one of the following shared keys:
      a) if the Encrypted payload appears in the parameter data for an Authentication step SECURITY PROTOCOL OUT command (see 5.13.4.7.2) or the parameter data for a Delete operation SECURITY PROTOCOL OUT command (see 5.13.4.11) that affects an active IKEv2-SCSI CCS, then the SK_ai shared key (see 5.13.4.4) for the IKEv2-SCSI CCS;
      b) if the Encrypted payload appears in the parameter data for an Authentication step SECURITY PROTOCOL IN command (see 5.13.4.7.3), then the SK_ar shared key (see 5.13.4.4) for the IKEv2-SCSI CCS; or
      c) if the Encrypted payload appears in the parameter data for a Delete operation SECURITY PROTOCOL OUT command (see 5.13.4.11) that does not affect an active IKEv2-SCSI CCS, then the SK_ai shared key (see 5.13.4.4) from the MGMT_DATA SA parameter (see 5.13.4.9);

   and
2) process the CIPHERTEXT field using the encryption algorithm specified by the ALGORITHM IDENTIFIER field in the ENCR IKEv2-SCSI cryptographic algorithm descriptor (see 7.7.3.6.2) in the IKEv2-SCSI SA Cryptographic Algorithms payload (see 7.7.3.5.13) in the Key Exchange step (see 5.13.4.6) using the inputs:
   A) IKEv2-SCSI AAD is an input to the encryption algorithm as follows:
      a) if a non-combined mode encryption algorithm is being used (see 7.7.3.5.11.1), then there is no AAD input; or
      b) if a combined mode encryption algorithm is being used (see 7.7.3.5.11.1), then the AAD described in 7.7.3.5.11.3 is an input;
   B) the contents of the INITIALIZATION VECTOR field (see table 540) in the Encrypted payload;
   C) the contents of the CIPHERTEXT field (see table 540) in the Encrypted payload; and
   D) the shared key value, key length, and salt bytes (see table 549 in 7.7.3.6.2) if any, for one of the following shared keys:
      a) if the Encrypted payload appears in the parameter data for an Authentication step SECURITY PROTOCOL OUT command (see 5.13.4.7.2), or the parameter data for a Delete operation SECURITY PROTOCOL OUT command (see 5.13.4.11) that affects an active IKEv2-SCSI CCS, then the SK_ei shared key (see 5.13.4.4) for the IKEv2-SCSI CCS;
      b) if the Encrypted payload appears in the parameter data for an Authentication step SECURITY PROTOCOL IN command (see 5.13.4.7.3), then the SK_er shared key (see 5.13.4.4) for the IKEv2-SCSI CCS; or
      c) if the Encrypted payload appears in the parameter data for a Delete operation SECURITY PROTOCOL OUT command (see 5.13.4.11) that does not affect an active IKEv2-SCSI CCS, then the SK_ei shared key (see 5.13.4.4) from the MGMT_DATA SA parameter (see 5.13.4.9).

The results of step 2) are:

a) the decrypted plaintext, if a non-combined mode encryption algorithm is being used (see 7.7.3.5.11.1); or
b) the decrypted plaintext and the comparison integrity check value, if a combined mode encryption algorithm is being used.

If the Encrypted payload appears in the parameter data for a Delete operation SECURITY PROTOCOL OUT command that does not affect an active IKEv2-SCSI CCS, then the integrity checking algorithm identifier value and encryption algorithm identifier value that are stored in the MGMT_DATA SA parameter indicate the integrity checking algorithm to use.

If the comparison integrity check value differs from the value in the INTEGRITY CHECK VALUE field of the Encrypted payload, then:

a) the application client should abandon the IKEv2-SCSI CCS and notify the device server that it is abandoning the IKEv2-SCSI CCS as described in 5.13.4.10; and
b) the device server shall respond to the mismatch as follows:
   A) if the IKEv2-SCSI header (see 7.7.3.4) specifies an attempt to provide authentication data for or the deletion of an IKE-v2-SCSI CCS on the I_T_L nexus on which the command was received, then the device server shall:
      a) terminate the SECURITY PROTOCOL OUT command with CHECK CONDITION status, with the sense key set to ILLEGAL REQUEST, and the additional sense code set to SA CREATION PARAMETER VALUE REJECTED; and
      b) continue the IKEv2-SCSI CCS by preparing to receive another Authentication step SECURITY PROTOCOL OUT command;

      or
   B) if the IKEv2-SCSI header (see 7.7.3.4) specifies the deletion of an active SA, then the device server shall terminate the SECURITY PROTOCOL OUT command with CHECK CONDITION status, with the sense key set to ILLEGAL REQUEST, and the additional sense code set to INVALID FIELD IN PARAMETER LIST.

7.7.3.5.12 IKEv2-SCSI SA Creation Capabilities payload

The IKEv2-SCSI SA Creation Capabilities payload (see table 542) lists all the security algorithms that the device server allows to be used in an IKEv2-SCSI CCS. Events that are outside the scope of this standard may change the contents of the IKEv2-SCSI SA Creation Capabilities payload at any time.

Table 542 — IKEv2-SCSI SA Creation Capabilities payload format

Bit / Byte	7	6	5	4	3	2	1	0
0	NEXT PAYLOAD (00h)							
1	CRIT	Reserved						
2	(MSB)	IKE PAYLOAD LENGTH (n+1)						
3								(LSB)
4	Reserved							
5								
6								
7	NUMBER OF ALGORITHM DESCRIPTORS							
IKEv2-SCSI cryptographic algorithm descriptors								
8	IKEv2-SCSI cryptographic algorithm descriptor (see 7.7.3.6) [first]							
•••								
	⋮							
	IKEv2-SCSI cryptographic algorithm descriptor (see 7.7.3.6) [last]							
•••								
n								

The NEXT PAYLOAD field, CRIT bit, and IKE PAYLOAD LENGTH field are described in 7.7.3.5.1.

The NEXT PAYLOAD field is set as shown in table 542 (i.e., No Next Payload) in the IKEv2-SCSI SA Creation Capabilities payload.

The NUMBER OF ALGORITHM DESCRIPTORS field specifies the number of IKEv2-SCSI cryptographic algorithm descriptors that follow in the IKEv2-SCSI SA Creation Capabilities payload.

Each IKEv2-SCSI cryptographic algorithm descriptor (see 7.7.3.6) describes one combination of security algorithm and algorithm attributes that the device server allows to be used in an IKEv2-SCSI CCS. If more than one set of algorithm attributes (e.g., key length) is allowed for any allowed security algorithm, then a different SCSI cryptographic algorithms descriptor shall be included for each set of algorithm attributes.

The SCSI cryptographic algorithms descriptors shall be ordered by:

1) increasing algorithm type;
2) increasing algorithm identifier within the same algorithm type; and
3) increasing key length, if any, within the same algorithm identifier.

The algorithms allowed may be a subset of the algorithms supported by the device server.

The method for changing which of the device server supported algorithms are allowed is outside the scope of this standard. Any changes in allowed algorithms do not take effect until the new list is returned to any application client in an IKEv2-SCSI SA Creation Capabilities payload.

7.7.3.5.13 IKEv2-SCSI SA Cryptographic Algorithms payload

The IKEv2-SCSI SA Cryptographic Algorithms payload (see table 543) lists the security algorithms that are being used in the creation and management (e.g., deletion) of an SA using an IKEv2-SCSI CCS.

Table 543 — IKEv2-SCSI SA Cryptographic Algorithms payload format

Bit / Byte	7	6	5	4	3	2	1	0
0	NEXT PAYLOAD							
1	CRIT	Reserved						
2	(MSB)	IKE PAYLOAD LENGTH (n+1)						
3								(LSB)
4	Reserved							
•••								
13								
14	(MSB)	USAGE DATA LENGTH (0000h)						
15								(LSB)
16	Reserved							
•••								
19								
20	NUMBER OF ALGORITHM DESCRIPTORS							
IKEv2-SCSI cryptographic algorithm descriptors								
21	IKEv2-SCSI cryptographic algorithm descriptor (see 7.7.3.6) [first]							
•••								
	⋮							
	IKEv2-SCSI cryptographic algorithm descriptor (see 7.7.3.6) [last]							
•••								
n								

The NEXT PAYLOAD field, CRIT bit, and IKE PAYLOAD LENGTH field are described in 7.7.3.5.1.

The USAGE DATA LENGTH field is set as shown in table 543 in the IKEv2-SCSI SA Cryptographic Algorithms payload.

The NUMBER OF ALGORITHM DESCRIPTORS field specifies the number of IKEv2-SCSI cryptographic algorithm descriptors that follow in the IKEv2-SCSI SA Cryptographic Algorithms payload. If a device server receives a NUMBER OF ALGORITHM DESCRIPTORS field that is not set to six, then the error shall be processed as described in 7.7.3.8.3.

Each IKEv2-SCSI cryptographic algorithm descriptor (see 7.7.3.6) describes one combination of security algorithm and algorithm attributes to be used during the IKEv2-SCSI CCS.

The IKEv2-SCSI cryptographic algorithm descriptors are ordered as follows:

1) one ENCR IKEv2-SCSI cryptographic algorithm descriptor (see 7.7.3.6.2);
2) one PRF IKEv2-SCSI cryptographic algorithm descriptor (see 7.7.3.6.3);
3) one INTEG IKEv2-SCSI cryptographic algorithm descriptor (see 7.7.3.6.4);
4) one D-H IKEv2-SCSI cryptographic algorithm descriptor (see 7.7.3.6.5);

5) one SA_AUTH_OUT IKEv2-SCSI cryptographic algorithm descriptor (see 7.7.3.6.6).
6) one SA_AUTH_IN IKEv2-SCSI cryptographic algorithm descriptor (see 7.7.3.6.6).

If a device server receives an IKEv2-SCSI SA Cryptographic Algorithms payload that does not contain the IKEv2-SCSI cryptographic algorithm descriptors described in this subclause in the order described in this subclause, then the error shall be processed as described in 7.7.3.8.3.

If the device server receives an IKEv2-SCSI SA Cryptographic Algorithms payload that contains an ENCR IKEv2-SCSI cryptographic algorithm descriptor with the ALGORITHM IDENTIFIER field set to ENCR_NULL, then the error shall be processed as described in 7.7.3.8.3.

In the Key Exchange step SECURITY PROTOCOL IN parameter data (see 5.13.4.6.3), the device server returns the IKEv2-SCSI SA Cryptographic Algorithms payload received during the Key Exchange step SECURITY PROTOCOL OUT command (see 5.13.4.6.2) to confirm acceptance of the algorithms.

7.7.3.5.14 IKEv2-SCSI SAUT Cryptographic Algorithms payload

The IKEv2-SCSI SAUT Cryptographic Algorithms payload (see table 544) lists the usage type of and security algorithms to be used by the SA that is created as a result of an IKEv2-SCSI CCS.

Table 544 — IKEv2-SCSI SAUT Cryptographic Algorithms payload format

Bit / Byte	7	6	5	4	3	2	1	0
0	NEXT PAYLOAD							
1	CRIT	Reserved						
2	(MSB)	IKE PAYLOAD LENGTH (n+1)						
3								(LSB)
4		Reserved						
11								
12	(MSB)	SA TYPE						
13								(LSB)
14	(MSB)	USAGE DATA LENGTH (j)						
15								(LSB)
16	USAGE DATA							
•••								
16+j-1								
16+j	Reserved							
16+j+1								
16+j+2								
16+j+3	NUMBER OF ALGORITHM DESCRIPTORS							
IKEv2-SCSI cryptographic algorithm descriptors								
16+j+4	IKEv2-SCSI cryptographic algorithm descriptor (see 7.7.3.6) [first]							
•••								
	⋮							
•••	IKEv2-SCSI cryptographic algorithm descriptor (see 7.7.3.6) [last]							
n								

The NEXT PAYLOAD field, CRIT bit, and IKE PAYLOAD LENGTH field are described in 7.7.3.5.1.

The SA TYPE field specifies the usage type for the SA and is selected from among those listed in table 75 (see 5.13.2.2). An error shall be processed as described in 7.7.3.8.3 if any of the following conditions occur:

a) the device server receives an SA usage type that is not allowed by the device server;
b) an SA usage type between 8000h and BFFFh inclusive is received in a Key Exchange step SECURITY PROTOCOL OUT command (see 5.13.4.6.2); or
c) an SA usage type between A000h and CFFFh inclusive is received for which the device server is unable to verify the applicable usage type constraints.

The method for changing which of the device server supported SA usage types are allowed is outside the scope of this standard.

The USAGE DATA LENGTH field specifies number of bytes of usage data that follow.

The size and format of the usage data depends on the SA type (see table 75 in 5.13.2.2). If the device server receives a USAGE DATA LENGTH field that contains a value that is inconsistent with the SA type, then the error shall be processed as described in 7.7.3.8.3.

The USAGE DATA field contains information to be stored in the USAGE_DATA SA parameter if the SA is generated (see 5.13.4.9).

The NUMBER OF ALGORITHM DESCRIPTORS field specifies the number of IKEv2-SCSI cryptographic algorithm descriptors that follow in the IKEv2-SCSI SAUT Cryptographic Algorithms payload. If a device server receives a NUMBER OF ALGORITHM DESCRIPTORS field that is not set to two, the error shall be processed as described in 7.7.3.8.3.

Each IKEv2-SCSI cryptographic algorithm descriptor (see 7.7.3.6) describes one combination of security algorithm and algorithm attributes to be used by the SA created as a result of the IKEv2-SCSI CCS. The IKEv2-SCSI cryptographic algorithm descriptors are ordered as follows:

1) one ENCR IKEv2-SCSI cryptographic algorithm descriptor (see 7.7.3.6.2); and
2) one INTEG IKEv2-SCSI cryptographic algorithm descriptor (see 7.7.3.6.4).

If a device server receives an IKEv2-SCSI SAUT Cryptographic Algorithms payload that does not contain the IKEv2-SCSI cryptographic algorithm descriptors described in this subclause in the order described in this subclause, then the error shall be processed as described in 7.7.3.8.3.

7.7.3.5.15 IKEv2-SCSI Timeout Values payload

The IKEv2-SCSI Timeout Values payload (see table 545) specifies the timeout intervals associated with an IKEv2-SCSI CCS

Table 545 — IKEv2-SCSI Timeout Values payload format

Bit / Byte	7	6	5	4	3	2	1	0
0	NEXT PAYLOAD							
1	CRIT	Reserved						
2	(MSB)			IKE PAYLOAD LENGTH (0010h)				
3								(LSB)
4	Reserved							
•••								
6								
7	NUMBER OF TIMEOUT VALUES (02h)							
8	(MSB)			IKEV2-SCSI PROTOCOL TIMEOUT				
•••								
11								(LSB)
12	(MSB)			IKEV2-SCSI SA INACTIVITY TIMEOUT				
•••								
15								(LSB)

The NEXT PAYLOAD field, CRIT bit, and IKE PAYLOAD LENGTH field are described in 7.7.3.5.1.

The NUMBER OF TIMEOUT VALUES field specifies the number of four-byte timeout values that follow. If the number of timeout values is less than two, the IKEv2-SCSI CCS state maintained for the I_T_L nexus shall be abandoned as described in 7.7.3.8.3.

The IKEV2-SCSI PROTOCOL TIMEOUT field specifies the number of seconds that the device server shall wait for the next command in the IKEv2-SCSI CCS. If the timeout expires before the device server receives an IKEv2-SCSI CCS command, the device server shall abandon the IKEv2-SCSI CCS as described in 5.13.4.10.

The IKEV2-SCSI SA INACTIVITY TIMEOUT field specifies the number of seconds that the device server shall wait for the next command that uses an SA. This value is copied to the TIMEOUT SA parameter when the SA is generated (see 5.13.4.9).

The device server shall replace any timeout value that is set to zero with a value of ten (i.e., ten seconds).

The maximum value for the protocol timeout should be long enough to allow the application client to continue the IKEv2-SCSI CCS, but short enough that, if an incomplete IKEv2-SCSI CCS is abandoned, the device server discards the state for that IKEv2-SCSI CCS and becomes available to for another IKEv2-SCSI CCS without delays that have adverse effects.

7.7.3.6 IKEv2-SCSI cryptographic algorithm descriptors

7.7.3.6.1 Overview

Each IKEv2-SCSI cryptographic algorithm descriptor (see table 546) specifies a cryptographic algorithm for IKEv2-SCSI (e.g., an encryption algorithm, an integrity checking algorithm, a key generation algorithm, or an authentication algorithm).

Table 546 — IKEv2-SCSI cryptographic algorithm descriptor format

Bit / Byte	7	6	5	4	3	2	1	0
0	ALGORITHM TYPE							
1	Reserved							
2	(MSB)			IKE DESCRIPTOR LENGTH (000Ch)				
3								(LSB)
4	(MSB)			ALGORITHM IDENTIFIER				
•••								
7								(LSB)
8	ALGORITHM ATTRIBUTES							
•••								
11								

The ALGORITHM TYPE field (see table 547) specifies the type of cryptographic algorithm to which the IKEv2-SCSI cryptographic algorithm descriptor applies.

Table 547 — ALGORITHM TYPE field

Code	Name	Description	Reference
01h	ENCR	Encryption algorithm	7.7.3.6.2
02h	PRF	Pseudo-random function	7.7.3.6.3
03h	INTEG	Integrity algorithm	7.7.3.6.4
04h	D-H	Diffie-Hellman group	7.7.3.6.5
05h to F0h	Restricted		RFC 4306
F9h	SA_AUTH_OUT	IKEv2-SCSI authentication algorithm for SECURITY PROTOCOL OUT data	7.7.3.6.6
FAh	SA_AUTH_IN	IKEv2-SCSI authentication algorithm for SECURITY PROTOCOL IN data	7.7.3.6.6
All others	Reserved		

The IKE DESCRIPTOR LENGTH field specifies the total number of bytes in the IKEv2-SCSI SA Cryptographic Algorithms descriptor and shall be set as shown in table 546 in all the IKEv2-SCSI SA Cryptographic Algorithms descriptors summarized in table 547.

NOTE 58 - The contents of the IKE DESCRIPTOR LENGTH field differ from those found in most SCSI length fields, However, they are consistent with the IKEv2 usage (see RFC 4306).

The contents of the ALGORITHM IDENTIFIER field and ALGORITHM ATTRIBUTES field depend on the contents of the ALGORITHM TYPE field (see table 547). The ALGORITHM ATTRIBUTES field is reserved in some IKEv2-SCSI SA Cryptographic Algorithms descriptor formats.

7.7.3.6.2 ENCR IKEv2-SCSI cryptographic algorithm descriptor

If the ALGORITHM TYPE field is set to ENCR (i.e., 01h) in an IKEv2-SCSI cryptographic algorithm descriptor (see table 548), then the descriptor specifies an encryption algorithm to be applied during the IKEv2-SCSI Authentication step (see 5.13.4.7), SA deletion operation (see 5.13.4.11), and when the SA created by the IKEv2-SCSI is applied to user data.

Table 548 — ENCR IKEv2-SCSI cryptographic algorithm descriptor format

Bit / Byte	7	6	5	4	3	2	1	0
0	ALGORITHM TYPE (01h)							
1	Reserved							
2	(MSB)			IKE DESCRIPTOR LENGTH (000Ch)				
3								(LSB)
4	(MSB)			ALGORITHM IDENTIFIER				
•••								
7								(LSB)
8				Reserved				
9								
10	(MSB)			KEY LENGTH				
11								(LSB)

The ALGORITHM TYPE field and IKE DESCRIPTOR LENGTH field are described in 7.7.3.6.1, and shall be set as shown in table 548 for the ENCR IKEv2-SCSI cryptographic algorithm descriptor.

The ALGORITHM IDENTIFIER field (see table 549) specifies the encryption algorithm to which the ENCR IKEv2-SCSI cryptographic algorithm descriptor applies.

Table 549 — ENCR ALGORITHM IDENTIFIER field

Code	Description	Salt bytes length (bytes)	IV [a] length (bytes)	Align-ment [b] (bytes)	Key length (bytes)	Support	Reference
8001 000Bh	ENCR_NULL	n/a	0	4	0	Mandatory	RFC 2410
8001 000Ch	AES-CBC	n/a	16	16	16	Optional	RFC 3602
					24	Prohibited	
					32	Optional	
8001 0010h	AES-CCM with a 16-byte MAC	3	8	4	16	Optional	RFC 4309 and RFC 5282
					24	Prohibited	
					32	Optional	
8001 0014h	AES-GCM with a 16-byte MAC	4	8	4	16	Optional	RFC 4106 and RFC 5282
					24	Prohibited	
					32	Optional	
8001 0400h to 8001 FFFFh	Vendor Specific						
0000 0000h to 0000 FFFFh	Restricted						IANA
All other values	Reserved						

[a] Initialization Vector.
[b] The alignment required in the plaintext prior to encryption..

ENCR_NULL indicates that encryption shall not be applied when the SA created by the IKEv2-SCSI is applied to user data.

The IKEv2-SCSI CCS state maintained for the I_T_L nexus shall be abandoned and an error shall be processed as described in 7.7.3.8.3 if the parameter list contains an IKEv2-SCSI SA Cryptographic Algorithms payload (see 7.7.3.5.13) that contains:

a) an ENCR IKEv2-SCSI cryptographic algorithm descriptor with the ALGORITHM IDENTIFIER field set to ENCR_NULL;
b) the following combination of IKEv2-SCSI cryptographic algorithm descriptors (see 7.7.3.6.4):
   A) an INTEG IKEv2-SCSI cryptographic algorithm descriptor with the ALGORITHM IDENTIFIER field set to a value other than AUTH_COMBINED; and
   B) an ENCR IKEv2-SCSI cryptographic algorithm descriptor with the ALGORITHM IDENTIFIER field set to a value that table 549 describes as requiring AUTH_COMBINED as the integrity check algorithm;

   or
c) the following combination of IKEv2-SCSI cryptographic algorithm descriptors:
   A) an INTEG IKEv2-SCSI cryptographic algorithm descriptor with the ALGORITHM IDENTIFIER field set to AUTH_COMBINED; and
   B) an ENCR IKEv2-SCSI cryptographic algorithm descriptor with the ALGORITHM IDENTIFIER field set to a value that table 549 does not describe as requiring AUTH_COMBINED as the integrity check algorithm.

The KEY LENGTH field specifies the number of bytes in the shared key for the encryption algorithm to which the ENCR IKEv2-SCSI cryptographic algorithm descriptor applies.

The IKEv2-SCSI CCS state maintained for the I_T_L nexus shall be abandoned as described in 7.7.3.8.3 if the parameter list contains:

a) no ENCR IKEv2-SCSI cryptographic algorithm descriptors;
b) more than one ENCR IKEv2-SCSI cryptographic algorithm descriptor;
c) an ENCR IKEv2-SCSI cryptographic algorithm descriptor that does not appear in the SA Creations Capabilities payload (see 7.7.3.5.12) last returned by the device server to any application client (i.e., an ENCR IKEv2-SCSI cryptographic algorithm descriptor for a combination of algorithm identifier and key length that the device server has not reported as one of its SA creation capabilities); or
d) an ENCR IKEv2-SCSI cryptographic algorithm descriptor that contains:
   A) an algorithm identifier that is not shown in table 549; or
   B) a key length that:
      a) does not match one of the values shown in table 549; or
      b) is not supported by the device server.

7.7.3.6.3 PRF IKEv2-SCSI cryptographic algorithm descriptor

If the ALGORITHM TYPE field is set to PRF (i.e., 02h) in an IKEv2-SCSI cryptographic algorithm descriptor (see table 550), then the descriptor specifies the pseudo-random function and KDF to be used during the Key Exchange step completion (see 5.13.4.6.4).

Table 550 — PRF IKEv2-SCSI cryptographic algorithm descriptor format

Bit / Byte	7	6	5	4	3	2	1	0
0	ALGORITHM TYPE (02h)							
1	Reserved							
2	(MSB)			IKE DESCRIPTOR LENGTH (000Ch)				
3								(LSB)
4	(MSB)			ALGORITHM IDENTIFIER				
•••								
7								(LSB)
8	Reserved							
•••								
11								

The ALGORITHM TYPE field and IKE DESCRIPTOR LENGTH field are described in 7.7.3.6.1, and shall be set as shown in table 550 for the PRF IKEv2-SCSI cryptographic algorithm descriptor.

The ALGORITHM IDENTIFIER field (see table 551) specifies PRF and KDF to which the PRF IKEv2-SCSI cryptographic algorithm descriptor applies.

Table 551 — PRF ALGORITHM IDENTIFIER field

Code	Description	Support	Output length (bytes)	Reference	
				PRF [a]	KDF [b]
8002 0002h	IKEv2-use based on SHA-1	Optional	20	RFC 2104	5.13.3.3
8002 0004h	IKEv2-use based on AES-128 in CBC mode	Optional	16	RFC 4434	5.13.3.4
8002 0005h	IKEv2-use based on SHA-256	Optional	32	RFC 4868	5.13.3.3
8002 0007h	IKEv2-use based on SHA-512	Optional	64	RFC 4868	5.13.3.3
8002 0400h to 8002 FFFFh	Vendor Specific				
0000 0000h to 0000 FFFFh	Restricted			IANA	
All others	Reserved				

[a] PRFs are equivalent to the prf() functions defined in RFC 4306.
[b] KDFs are equivalent to the prf+() functions defined in RFC 4306.

The IKEv2-SCSI CCS state maintained for the I_T_L nexus shall be abandoned as described in 7.7.3.8.3 if the parameter list contains:

a) no PRF IKEv2-SCSI cryptographic algorithm descriptors;
b) more than one PRF IKEv2-SCSI cryptographic algorithm descriptor;
c) a PRF IKEv2-SCSI cryptographic algorithm descriptor that contains an algorithm identifier that does not appear in the SA Creations Capabilities payload (see 7.7.3.5.12) last returned by the device server to any application client; or
d) an PRF IKEv2-SCSI cryptographic algorithm descriptor that contains an algorithm identifier that is not shown in table 551.

7.7.3.6.4 INTEG IKEv2-SCSI cryptographic algorithm descriptor

If the ALGORITHM TYPE field is set to INTEG (i.e., 03h) in an IKEv2-SCSI cryptographic algorithm descriptor (see table 552), then the descriptor specifies an integrity checking (i.e., data authentication) algorithm to be applied during the IKEv2-SCSI Authentication step (see 5.13.4.7) and when the SA created by the IKEv2-SCSI is applied to user data.

Table 552 — INTEG IKEv2-SCSI cryptographic algorithm descriptor format

Bit / Byte	7	6	5	4	3	2	1	0
0	ALGORITHM TYPE (03h)							
1	Reserved							
2	(MSB)			IKE DESCRIPTOR LENGTH (000Ch)				
3								(LSB)
4	(MSB)			ALGORITHM IDENTIFIER				
•••								
7								(LSB)
8				Reserved				
•••								
11								

The ALGORITHM TYPE field and IKE DESCRIPTOR LENGTH field are described in 7.7.3.6.1, and shall be set as shown in table 552 for the INTEG IKEv2-SCSI cryptographic algorithm descriptor.

The ALGORITHM IDENTIFIER field (see table 553) specifies integrity checking algorithm and shared key length to which the INTEG IKEv2-SCSI cryptographic algorithm descriptor applies.

Table 553 — INTEG ALGORITHM IDENTIFIER field

Code	IKEv2 Name	ICV [a] length (bytes)	Key length (bytes)	Support	Reference
8003 0002h	AUTH_HMAC_SHA1_96	12	20	Optional	RFC 2404
8003 000Ch	AUTH_HMAC_SHA2_256_128	16	32	Optional	RFC 4868
8003 000Eh	AUTH_HMAC_SHA2_512_256	32	64	Optional	RFC 4868
F003 0001h	AUTH_COMBINED	n/a	0	Optional	this subclause
8003 0400h to 8003 FFFFh	Vendor Specific				
0000 0000h to 0000 FFFFh	Restricted				IANA
All others	Reserved				

[a] Integrity Check Value.

The AUTH_COMBINED integrity checking algorithm is used with encryption algorithms that include integrity checking as described in 7.7.3.6.2. The AUTH_COMBINED algorithm identifier specifies that no additional integrity check is performed, as indicated by the zero-length key.

The key length used with an integrity checking algorithm is determined by the algorithm identifier as shown in table 553.

The IKEv2-SCSI CCS state maintained for the I_T_L nexus shall be abandoned as described in 7.7.3.8.3 if the parameter list contains:

a) no INTEG IKEv2-SCSI cryptographic algorithm descriptors;
b) more than one INTEG IKEv2-SCSI cryptographic algorithm descriptor;
c) an INTEG IKEv2-SCSI cryptographic algorithm descriptor that contains an algorithm identifier that does not appear in the SA Creations Capabilities payload (see 7.7.3.5.12) last returned by the device server to any application client; or
d) an INTEG IKEv2-SCSI cryptographic algorithm descriptor that contains an algorithm identifier that is not shown in table 553.

7.7.3.6.5 D-H IKEv2-SCSI cryptographic algorithm descriptor

If the ALGORITHM TYPE field is set to D-H (i.e., 04h) in an IKEv2-SCSI cryptographic algorithm descriptor (see table 554), then the descriptor specifies Diffie-Hellman group and Diffie-Hellman algorithm used during the IKEv2-SCSI Key Exchange step (see 5.13.4.6) to derive a shared key that is known only to the application client and device server.

Table 554 — D-H IKEv2-SCSI cryptographic algorithm descriptor format

Bit / Byte	7	6	5	4	3	2	1	0
0	ALGORITHM TYPE (04h)							
1	Reserved							
2	(MSB)			IKE DESCRIPTOR LENGTH (000Ch)				
3								(LSB)
4	(MSB)			ALGORITHM IDENTIFIER				
•••								
7								(LSB)
8	Reserved							
•••								
11								

The ALGORITHM TYPE field and IKE DESCRIPTOR LENGTH field are described in 7.7.3.6.1, and shall be set as shown in table 554 for the D-H IKEv2-SCSI cryptographic algorithm descriptor.

The ALGORITHM IDENTIFIER field (see table 555) specifies Diffie-Hellman algorithm, group, and shared key length to which the D-H IKEv2-SCSI cryptographic algorithm descriptor applies.

Table 555 — D-H ALGORITHM IDENTIFIER field

Code	Description	Key length (bytes)	Support	Reference
8004 000Eh	2 048-bit MODP group (finite field D-H)	256	Optional	RFC 3526
8004 000Fh	3 072-bit MODP group (finite field D-H)	384	Optional	RFC 3526
8004 0010h	4 096-bit MODP group (finite field D-H)	512	Optional	RFC 3526
8004 0013h	256-bit random ECP group	64	Optional	RFC 4753 [a]
8004 0015h	521-bit random ECP group	132	Optional	RFC 4753 [a]
8004 0400h to 8004 FFFFh	Vendor Specific			
0000 0000h to 0000 FFFFh	Restricted			IANA
All others	Reserved			

[a] The RFC Errata for RFC 4753 modify the binary representation of ECP key exchange data. The modified binary representation shall be used, see http://rfc-editor.org/errata_search.php?rfc=4753.

The key length of the public value transferred in the KEY EXCHANGE DATA field (see 7.7.3.5.3) is determined by the algorithm identifier as shown in table 555.

The IKEv2-SCSI CCS state maintained for the I_T_L nexus shall be abandoned as described in 7.7.3.8.3 if the parameter list contains:

a) no D-H IKEv2-SCSI cryptographic algorithm descriptors;
b) more than one D-H IKEv2-SCSI cryptographic algorithm descriptor;
c) a D-H IKEv2-SCSI cryptographic algorithm descriptor that contains an algorithm identifier that does not appear in the SA Creations Capabilities payload (see 7.7.3.5.12) last returned by the device server to any application client; or
d) an D-H IKEv2-SCSI cryptographic algorithm descriptor that contains an algorithm identifier that is not shown in table 555.

7.7.3.6.6 IKEv2-SCSI authentication algorithm IKEv2-SCSI cryptographic algorithm descriptor

If the ALGORITHM TYPE field is set to SA_AUTH_OUT (i.e., F9h) in an IKEv2-SCSI cryptographic algorithm descriptor (see table 556), then the descriptor specifies Authentication payload authentication algorithm used by the IKEv2-SCSI Authentication step SECURITY PROTOCOL OUT command (see 5.13.4.7.2).

If the ALGORITHM TYPE field is set to SA_AUTH_IN (i.e., FAh) in an IKEv2-SCSI cryptographic algorithm descriptor (see table 556), then the descriptor specifies Authentication payload authentication algorithm used by the IKEv2-SCSI Authentication step SECURITY PROTOCOL IN command (see 5.13.4.7.3).

Table 556 — SA_AUTH_OUT and SA_AUTH_IN IKEv2-SCSI cryptographic algorithm descriptor format

Bit / Byte	7	6	5	4	3	2	1	0
0	ALGORITHM TYPE (F9h or FAh)							
1	Reserved							
2	(MSB)			IKE DESCRIPTOR LENGTH (000Ch)				
3								(LSB)
4	(MSB)			ALGORITHM IDENTIFIER				
•••								
7								(LSB)
8				Reserved				
•••								
11								

The ALGORITHM TYPE field and IKE DESCRIPTOR LENGTH field are described in 7.7.3.6.1. The IKE DESCRIPTOR LENGTH field shall be set as shown in table 556 for the SA_AUTH_OUT IKEv2-SCSI cryptographic algorithm descriptor and the SA_AUTH_IN IKEv2-SCSI cryptographic algorithm descriptor.

The ALGORITHM IDENTIFIER field (see table 557) specifies Authentication payload authentication algorithm to which the SA_AUTH_OUT IKEv2-SCSI cryptographic algorithm descriptor or SA_AUTH_IN IKEv2-SCSI cryptographic algorithm descriptor applies.

Table 557 — SA_AUTH_OUT and SA_AUTH_IN ALGORITHM IDENTIFIER field

Code	Description	Support	Reference	Authentication Reference [a]
00F9 0000h	SA_AUTH_NONE	Optional	this subclause	n/a
00F9 0001h	RSA Digital Signature with SHA-1 [b]	Optional	RFC 4306	5.13.4.3.3
00F9 0002h	Shared Key Message Integrity Code	Optional	RFC 4306 [c, d]	5.13.4.3.2
00F9 0009h	ECDSA with SHA-256 on the P-256 curve [b]	Optional	RFC 4754	5.13.4.3.3
00F9 000Bh	ECDSA with SHA-512 on the P-521 curve [b]	Optional	RFC 4754	5.13.4.3.3
00F9 00C9h to 00F9 00FFh	Vendor Specific	Optional		
0000 0000h to 0000 FFFFh	Restricted	Prohibited	IANA	
All others	Reserved			

[a] Description of how the algorithm shall be used to generate and verify the contents of the AUTHENTICATION DATA field of the Authentication Payload.

[b] Support for an RSA Digital Signature authentication algorithm does not mandate support for digital certificates.

[c] The 17 ASCII character non-terminated pre-shared key pad string 'Key Pad for IKEv2' specified by RFC 4306 is replaced by the 22 ASCII character non-terminated pre-shared key pad string 'Key Pad for IKEv2-SCSI'.

[d] The pre-shared key requirements used by this standard (see 5.13.4.3.2) apply in addition to those found in RFC 4306.

SA_AUTH_NONE specifies the omission of the IKEv2-SCSI Authentication step (see 5.13.4.7) as follows:

a) the presence of an SA_AUTH_OUT IKEv2-SCSI cryptographic algorithm descriptor and an SA_AUTH_IN IKEv2-SCSI cryptographic algorithm descriptor with the ALGORITHM IDENTIFIER field set to SA_AUTH_NONE in an SA Creation Capabilities payload (see 7.7.3.5.12) indicates that the device server is allowed to negotiate the omission of the IKEv2-SCSI Authentication step; and
b) the presence of an SA_AUTH_OUT IKEv2-SCSI cryptographic algorithm descriptor and an SA_AUTH_IN IKEv2-SCSI cryptographic algorithm descriptor with the ALGORITHM IDENTIFIER field set to SA_AUTH_NONE in an IKEv2-SCSI SA Cryptographic Algorithms payload (see 7.7.3.5.13) indicates the following based upon the command whose parameter data carries the payload:
   A) in the parameter list for a Key Exchange SECURITY PROTOCOL OUT command (see 5.13.4.6.2), SA_AUTH_NONE specifies that the application client is requesting that the IKEv2-SCSI Authentication step be skipped; and
   B) in the parameter data for a Key Exchange SECURITY PROTOCOL IN command (see 5.13.4.6.3), SA_AUTH_NONE indicates that the device server has agreed to skip the IKEv2-SCSI Authentication step.

If the device server indicates that skipping the Authentication step is allowed in its capabilities and the application client specifies that the Authentication step be skipped (see 5.13.4.3.4), then the IKEv2-SCSI Authentication step is skipped and the resulting SAs are not authenticated.

An SA_AUTH_OUT IKEv2-SCSI cryptographic algorithm descriptor with the ALGORITHM IDENTIFIER field set to SA_AUTH_NONE shall not appear in an IKEv2-SCSI SA Cryptographic Algorithms payload except as described in 5.13.4.3.4.

The Shared Key Message Integrity Code is based on a pre-shared key (see 5.13.4.3.2) that is associated with the identity in the Identification payload (see 7.7.3.5.4).

The IKEv2-SCSI CCS state maintained for the I_T_L nexus shall be abandoned as described in 7.7.3.8.3 if the parameter list contains:

a) no SA_AUTH_OUT IKEv2-SCSI cryptographic algorithm descriptors;
b) no SA_AUTH_IN IKEv2-SCSI cryptographic algorithm descriptors;
c) more than one SA_AUTH_OUT IKEv2-SCSI cryptographic algorithm descriptor;
d) more than one SA_AUTH_IN IKEv2-SCSI cryptographic algorithm descriptor;
e) an SA_AUTH_OUT IKEv2-SCSI cryptographic algorithm descriptor that contains an algorithm identifier that does not appear in the SA Creations Capabilities payload (see 7.7.3.5.12) last returned by the device server to any application client;
f) an SA_AUTH_IN IKEv2-SCSI cryptographic algorithm descriptor that contains an algorithm identifier that does not appear in the SA Creations Capabilities payload last returned by the device server to any application client;
g) an SA_AUTH_OUT IKEv2-SCSI cryptographic algorithm descriptor that contains an algorithm identifier that is not shown in table 557;
h) an SA_AUTH_IN IKEv2-SCSI cryptographic algorithm descriptor that contains an algorithm identifier that is not shown in table 557;
i) an SA_AUTH_OUT IKEv2-SCSI cryptographic algorithm descriptor with the ALGORITHM IDENTIFIER field set to SA_AUTH_NONE, and an SA_AUTH_IN IKEv2-SCSI cryptographic algorithm descriptor with the ALGORITHM IDENTIFIER field set to a value other than SA_AUTH_NONE; or
j) an SA_AUTH_IN IKEv2-SCSI cryptographic algorithm descriptor with the ALGORITHM IDENTIFIER field set to SA_AUTH_NONE, and an SA_AUTH_OUT IKEv2-SCSI cryptographic algorithm descriptor with the ALGORITHM IDENTIFIER field set to a value other than SA_AUTH_NONE.

7.7.3.7 Errors in IKEv2-SCSI security protocol commands

For a single I_T_L nexus, the device server shall ensure that two or four IKEv2-SCSI CCS commands are processed in the order described in 5.13.4.1 as shown in table 558. For each SECURITY PROTOCOL OUT command shown in table 558, the parameter data shall not be processed unless table 558 specifies that the command is processed.

Table 558 — IKEv2-SCSI command ordering processing requirements on a single I_T_L nexus (part 1 of 2)

	Time →				
IKEv2-SCSI SECURITY PROTOCOL OUT command or SECURITY PROTOCOL IN command received	**Before Key Exchange step SECURITY PROTOCOL OUT command completes with GOOD status**	**After a Key Exchange step SECURITY PROTOCOL OUT command completes with GOOD status**	**After a Key Exchange step SECURITY PROTOCOL IN command completes with GOOD status**	**After an Authentication step SECURITY PROTOCOL OUT command completes with GOOD status**	**After an Authentication step SECURITY PROTOCOL IN command completes with GOOD status**
Key Exchange step SECURITY PROTOCOL OUT command	Process the command as described in this standard	Do not process the command [a]	Do not process the command [a]	Do not process the command [a]	Same as before Key Exchange step SECURITY PROTOCOL OUT command
Key Exchange step SECURITY PROTOCOL IN command	No IKEv2-SCSI CCS exists [b]	Process the command as described in this standard	Repeat processing of the command [c]	Do not process the command [a]	Do not process the command [a]

[a] The command shall be terminated and the IKEv2-SCSI CCS shall be continued as follows:
a) the device server shall terminate the command with CHECK CONDITION status, with the sense key set to NOT READY, and the additional sense code set to LOGICAL UNIT NOT READY, SA CREATION IN PROGRESS; and
b) the device server shall continue the IKEv2-SCSI CCS by preparing to receive another IKEv2-SCSI CCS SECURITY PROTOCOL OUT command.

[b] The device server shall terminate the command with CHECK CONDITION status, with the sense key set to ILLEGAL REQUEST, and the additional sense code set to INVALID FIELD IN CDB.

[c] Processing of the SECURITY PROTOCOL IN commands in an IKEv2-SCSI CCS may be repeated. The device server should save the information necessary to repeat processing of these commands until the number of seconds specified in the IKEV2-SCSI PROTOCOL TIMEOUT field of the IKEv2-SCSI Timeout Values payload (see 7.7.3.5.15) have elapsed since the processing of the Authentication step SECURITY PROTOCOL OUT command that completed with GOOD status.

Table 558 — IKEv2-SCSI command ordering processing requirements on a single I_T_L nexus (part 2 of 2)

IKEv2-SCSI SECURITY PROTOCOL OUT command or SECURITY PROTOCOL IN command received	Time → Before Key Exchange step SECURITY PROTOCOL OUT command completes with GOOD status	After a Key Exchange step SECURITY PROTOCOL OUT command completes with GOOD status	After a Key Exchange step SECURITY PROTOCOL IN command completes with GOOD status	After an Authentication step SECURITY PROTOCOL OUT command completes with GOOD status	After an Authentication step SECURITY PROTOCOL IN command completes with GOOD status
Authentication step SECURITY PROTOCOL OUT command	No IKEv2-SCSI CCS exists [b]	Do not process the command [a]	Process the command as described in this standard	Do not process the command [a]	Do not process the command [a]
Authentication step SECURITY PROTOCOL IN command		Do not process the command [a]	Do not process the command [a]	Process the command as described in this standard	Repeat processing of the command [c]
Command with an invalid field in the CDB	No IKEv2-SCSI CCS exists [b]	Do not process the command [a]	Do not process the command [a]	Do not process the command [a]	No IKEv2-SCSI CCS exists [b]

[a] The command shall be terminated and the IKEv2-SCSI CCS shall be continued as follows:
a) the device server shall terminate the command with CHECK CONDITION status, with the sense key set to NOT READY, and the additional sense code set to LOGICAL UNIT NOT READY, SA CREATION IN PROGRESS; and
b) the device server shall continue the IKEv2-SCSI CCS by preparing to receive another IKEv2-SCSI CCS SECURITY PROTOCOL OUT command.

[b] The device server shall terminate the command with CHECK CONDITION status, with the sense key set to ILLEGAL REQUEST, and the additional sense code set to INVALID FIELD IN CDB.

[c] Processing of the SECURITY PROTOCOL IN commands in an IKEv2-SCSI CCS may be repeated. The device server should save the information necessary to repeat processing of these commands until the number of seconds specified in the IKEV2-SCSI PROTOCOL TIMEOUT field of the IKEv2-SCSI Timeout Values payload (see 7.7.3.5.15) have elapsed since the processing of the Authentication step SECURITY PROTOCOL OUT command that completed with GOOD status.

The processing shown in table 558 shall be performed before parameter data error handling described in 7.7.3.8.

7.7.3.8 Errors in IKEv2-SCSI security protocol parameter data

7.7.3.8.1 Overview

Errors in the parameter data transferred to the device server by an IKEv2-SCSI SECURITY PROTOCOL OUT command are classified (see table 559) based on the ease with which they may be used to mount denial of service attacks against IKEv2-SCSI SA creation operations by an attacker that has not participated as the application client or device server in the Key Exchange step SECURITY PROTOCOL OUT command (see 5.13.4.6.2) that started the IKEv2-SCSI CCS.

Table 559 — IKEv2-SCSI parameter error categories

Denial of service attack potential	Applicable SECURITY PROTOCOL OUT commands	Error handling description	Reference
High [a]	Authentication step, and Delete operation	If the device sever is able to, the IKEv2-SCSI CCS state maintained for an I_T_L nexus is not changed	7.7.3.8.2
Low [b]	Key Exchange step, Authentication step, and Delete operation	The IKEv2-SCSI CCS state maintained for an I_T_L nexus is abandoned	7.7.3.8.3

[a] Attacks capable of causing significant harm by sending a malformed IKEv2-SCSI SECURITY PROTOCOL OUT command to the device server.

[b] Attacks that produce no significant harm, or collusive attacks (i.e., attacks that require knowledge of the IKEv2-SCSI CCS shared keys and participation in the IKEv2-SCSI CCS). Collusive attacks depend on collusion between the attacker and the application client (i.e., require the application client to act against its own best interests). The device server may abandon the IKEv2-SCSI CCS in response to such attacks.

7.7.3.8.2 Errors with high denial of service attack potential

Errors detected before or during the decryption and integrity checking of an Encrypted payload in an Authentication step SECURITY PROTOCOL OUT command or a Delete operation SECURITY PROTOCOL OUT command have a high potential for being a denial of service attack against one or more application clients (see table 559 in 7.7.3.8.1).

The device server shall respond to these SECURITY PROTOCOL OUT command errors as follows:

a) the device server shall terminate the command with CHECK CONDITION status, with the sense key set to ILLEGAL REQUEST, and the additional sense code set to SA CREATION PARAMETER VALUE REJECTED; and
b) the device server shall continue the IKEv2-SCSI CCS, if any, by preparing to receive an Authentication step SECURITY PROTOCOL OUT command.

If a specific field is the cause for returning the SA CREATION PARAMETER VALUE REJECTED additional sense code, then the SKSV bit may be set to one. If the SKSV bit is set to one, then the SENSE KEY SPECIFIC field shall be set as defined in 4.5.2.4.2.

7.7.3.8.3 Errors with low denial of service attack potential

The errors that have low denial of service attack potential for IKEv2-SCSI SA creation (see table 559 in 7.7.3.8.1) are:

a) all errors detected for a Key Exchange step SECURITY PROTOCOL OUT command or its associated parameter data (e.g., errors detected in the IKEv2-SCSI header) that is attempting to establish a new IKEv2-SCSI CCS on an I_T_L nexus; and
b) all errors that are detected after the Encrypted payload has been successfully decrypted and integrity checked in an Authentication step SECURITY PROTOCOL OUT command (see 5.13.4.7.2) or a Delete operation SECURITY PROTOCOL OUT command (see 5.13.4.11).

The device server shall respond to these SECURITY PROTOCOL OUT command errors by terminating the command with CHECK CONDITION status, with the sense key set to ILLEGAL REQUEST, and the additional sense code set to SA CREATION PARAMETER VALUE INVALID. The state being maintained for an IKEv2-SCSI CCS on the I_T_L nexus on which the command was received, if any, shall be abandoned (see 5.13.4.10).

If a specific field is the cause for returning the SA CREATION PARAMETER VALUE INVALID additional sense code, then the SKSV bit shall be set to one and SENSE KEY SPECIFIC field shall be set as defined in 4.5.2.4.2.

7.7.3.9 Translating IKEv2 errors

IKEv2 (see RFC 4306) defines an error reporting mechanism based on the Notify payload. This standard translates such error reports into the device server and application actions defined in this subclause.

If a device server is required by IKEv2 to report an error using a Notify payload, the device server shall translate the error into CHECK CONDITION status with the sense key and additional sense code as shown in table 560. The device server shall terminate the SECURITY PROTOCOL OUT command (see 6.41) that transferred the parameter list in which the IKE requirements for one or more payloads (see 7.7.3.5) require

use of the Notify payload to report an error. The device server shall terminate the SECURITY PROTOCOL OUT command as described in this subclause. The device server shall not report the IKE errors described in this subclause by terminating a SECURITY PROTOCOL IN command (see 6.40) with CHECK CONDITION status.

Table 560 — IKEv2 Notify payload error translations for IKEv2-SCSI

IKEv2 (see RFC 4306)		IKEv2-SCSI	
Error Type	**Description**	**Additional sense code**	**Sense key**
0000h	Reserved		
0001h	UNSUPPORTED_CRITICAL_PAYLOAD	SA CREATION PARAMETER NOT SUPPORTED	ILLEGAL REQUEST
0004h	INVALID_IKE_SPI	SA CREATION PARAMETER VALUE INVALID or SA CREATION PARAMETER VALUE REJECTED	
0005h	INVALID_MAJOR_VERSION		
0007h	INVALID_SYNTAX [a]		
0009h	INVALID_MESSAGE_ID		
000Bh	INVALID_SPI	SA CREATION PARAMETER VALUE INVALID [b]	
000Eh	NO_PROPOSAL_CHOSEN [c]	SA CREATION PARAMETER VALUE INVALID	
0011h	INVALID_KE_PAYLOAD [c]		
0018h	AUTHENTICATION_FAILED	AUTHENTICATION FAILED	ABORTED COMMAND
0022h to 0027h	See RFC 4306 [d]	n/a	n/a
2000h to 3FFFh	Vendor Specific		
All others	Restricted (see RFC 4306)		

[a] This sense key and one of the additional sense codes shown shall be returned for a syntax error within an Encrypted payload (see 7.7.3.5.11) regardless of conflicting IKEv2 requirements.

[b] SA CREATION PARAMETER VALUE INVALID shall be used for an invalid SAI in an IKEv2-SCSI SECURITY PROTOCOL IN or SECURITY PROTOCOL OUT. The additional sense code for an invalid SAI in all other commands is specified by the applicable usage type definition (see table 75 in 5.13.2.2).

[c] An application client recovers by restarting processing with the Device Capabilities step (see 5.13.4.5) to rediscover the device server's capabilities.

[d] These IKEv2 Error Types are used for features that are not supported by IKEv2-SCSI SA creation.

If an application client detects an IKEv2 error that RFC 4306 requires to be reported with a Notify payload, the application client should abandon the IKEv2-SCSI CCS and notify the device server that it is abandoning the IKEv2-SCSI CCS as described in 5.13.4.10.

7.7.4 CbCS security protocol

7.7.4.1 Overview

If the SECURITY PROTOCOL field in a SECURITY PROTOCOL IN command (see 6.40) is set to 07h, the command specifies one of the CbCS pages (see 7.7.4.2) to be returned by the device server. The information returned by a CbCS SECURITY PROTOCOL IN command indicates the CbCS operating parameters of:

a) the logical unit to which the CbCS SECURITY PROTOCOL IN command is addressed; or

b) the SCSI target device that contains the well-known logical unit to which the CbCS SECURITY PROTOCOL IN command is addressed.

If the SECURITY PROTOCOL field in a SECURITY PROTOCOL OUT command (see 6.41) is set to 07h, the command specifies one of the CbCS pages (see 7.7.4.4) to be sent to the device server. The information sent in a CbCS SECURITY PROTOCOL OUT command specifies the CbCS operating parameters of:

a) the logical unit to which the CbCS SECURITY PROTOCOL OUT command is addressed; or
b) the SCSI target device that contains the well-known logical unit to which the CbCS SECURITY PROTOCOL OUT command is addressed.

7.7.4.2 CbCS SECURITY PROTOCOL IN CDB description

The CbCS SECURITY PROTOCOL IN CDB has the format defined in 6.40 with the additional requirements described in this subclause.

If the SECURITY PROTOCOL field is set to CbCS (i.e., 07h) in a SECURITY PROTOCOL IN command, the SECURITY PROTOCOL SPECIFIC field (see table 561) specifies the CbCS page to be returned in the parameter data (see 7.7.4.3). If the CBCS bit is set to one in the Extended INQUIRY Data VPD page (see 7.8.7), the CbCS SECURITY PROTOCOL IN command support requirements are shown in table 561.

Table 561 — SECURITY PROTOCOL SPECIFIC field for the CbCS SECURITY PROTOCOL IN command

Code [a]	CbCS page returned	Support	Reference
0000h	Supported CbCS SECURITY PROTOCOL IN Pages	Mandatory	7.7.4.3.1
0001h	Supported CbCS SECURITY PROTOCOL OUT Pages	Mandatory	7.7.4.3.2
0002h	Unchangeable CbCS Parameters	Mandatory	7.7.4.3.3
0003h to 003Eh	Reserved		
003Fh	Security Token	See [b]	7.7.4.3.4
0040h	Current CbCS Parameters	Mandatory	7.7.4.3.5
0041h to D00Fh	Reserved		
D010h	Set Master Key – Seed Exchange	See [b]	7.7.4.3.6
D011h to FFFFh	Reserved		

[a] If the SECURITY PROTOCOL SPECIFIC field is set to a value that is less than D000h, the working key specified by the KEY VERSION field in the CbCS capability descriptor (see 6.26.2.3) shall be used to compute the capability key (see 5.13.6.7.12). If the SECURITY PROTOCOL SPECIFIC field is set to a value that is greater than or equal to D000h, the authentication key component of the master key (see 5.13.6.7.11) shall be used to compute the capability key.

[b] Mandatory if the CAPKEY CbCS method (see 5.13.6.7.8.3) is supported.

If a CbCS SECURITY PROTOCOL IN command is received with the INC_512 bit set to one, the device server shall terminate the command with CHECK CONDITION status, with the sense key set to ILLEGAL REQUEST, and the additional sense code set to INVALID FIELD IN CDB.

7.7.4.3 CbCS SECURITY PROTOCOL IN parameter data

7.7.4.3.1 Supported CbCS SECURITY PROTOCOL IN Pages CbCS page

The Supported CbCS SECURITY PROTOCOL IN Pages CbCS page (see table 562) lists the CbCS pages that are supported (i.e., the values that are allowed in the SECURITY PROTOCOL SPECIFIC field) for the SECURITY PROTOCOL IN command (see 6.40).

Table 562 — Supported CbCS SECURITY PROTOCOL IN Pages CbCS page format

Bit / Byte	7	6	5	4	3	2	1	0
0	(MSB)			PAGE CODE (0000h)				
1								(LSB)
2	(MSB)			PAGE LENGTH (n-3)				
3								(LSB)
Supported CbCS SECURITY PROTOCOL IN page list								
4	(MSB)			SUPPORTED CBCS SECURITY PROTOCOL IN PAGE (0000h) [first]				
5								(LSB)
				⋮				
n-1	(MSB)			SUPPORTED CBCS SECURITY PROTOCOL IN PAGE [last]				
n								(LSB)

The PAGE CODE field shall be set as shown in table 562 to indicate that the Supported CbCS SECURITY PROTOCOL IN Pages CbCS page is being returned.

The PAGE LENGTH field indicates the number of bytes that follow in the Supported CbCS SECURITY PROTOCOL IN Pages CbCS page.

Each SUPPORTED CBCS SECURITY PROTOCOL IN PAGE field indicates the page code (see table 561 in 7.7.4.2) of one CbCS page that is supported by the SECURITY PROTOCOL IN command if the SECURITY PROTOCOL field (see 6.40) is set to 07h (i.e., CbCS). The values in the SUPPORTED CBCS SECURITY PROTOCOL IN PAGE fields shall be returned in ascending order beginning with 0000h (i.e., this CbCS page).

7.7.4.3.2 Supported CbCS SECURITY PROTOCOL OUT pages CbCS page

The Supported CbCS SECURITY PROTOCOL OUT Pages CbCS page (see table 563) lists the CbCS pages that are supported (i.e., the values that are allowed in the SECURITY PROTOCOL SPECIFIC field) for the SECURITY PROTOCOL OUT command (see 6.41).

Table 563 — Supported CbCS SECURITY PROTOCOL OUT Pages CbCS page format

Bit / Byte	7	6	5	4	3	2	1	0
0	(MSB)			PAGE CODE (0001h)				
1								(LSB)
2	(MSB)			PAGE LENGTH (n-3)				
3								(LSB)
Supported CbCS SECURITY PROTOCOL OUT page list								
4	(MSB)			SUPPORTED CBCS SECURITY PROTOCOL OUT PAGE [first]				
5								(LSB)
	⋮							
n-1	(MSB)			SUPPORTED CBCS SECURITY PROTOCOL OUT PAGE [last]				
n								(LSB)

The PAGE CODE field shall be set as shown in table 563 to indicate that the Supported CbCS SECURITY PROTOCOL OUT Pages CbCS page is being returned.

The PAGE LENGTH field indicates the number of bytes that follow in the Supported CbCS SECURITY PROTOCOL OUT Pages CbCS page.

Each SUPPORTED CBCS SECURITY PROTOCOL OUT PAGE field indicates the page code (see table 570 in 7.7.4.4) of one CbCS page that is supported by the SECURITY PROTOCOL OUT command if the SECURITY PROTOCOL field (see 6.41) is set to 07h (i.e., CbCS). The values in the SUPPORTED CBCS SECURITY PROTOCOL IN PAGE fields shall be returned in ascending order.

7.7.4.3.3 Unchangeable CbCS Parameters CbCS page

The Unchangeable CbCS Parameters CbCS page (see table 564) indicates the supported CbCS features and algorithms.

Table 564 — Unchangeable CbCS Parameters CbCS page format (part 1 of 2)

Bit / Byte	7	6	5	4	3	2	1	0
0	(MSB)			PAGE CODE (0002h)				
1								(LSB)
2	(MSB)			PAGE LENGTH (n-3)				
3								(LSB)
4	KEYS SUPPORT		MIN CBCS METHOD SUP		Reserved			
5	Reserved							
6	(MSB)			SUPPORTED INTEGRITY CHECK VALUE ALGORITHM LIST LENGTH (c-7)				
7								(LSB)

Table 564 — Unchangeable CbCS Parameters CbCS page format (part 2 of 2)

Bit / Byte	7	6	5	4	3	2	1	0
Supported integrity check value algorithms list								
8	(MSB)			SUPPORTED INTEGRITY CHECK VALUE ALGORITHM [first]				
•••								
11								(LSB)
	⋮							
c-3	(MSB)			SUPPORTED INTEGRITY CHECK VALUE ALGORITHM [last]				
•••								
c								(LSB)
c+1				Reserved				
c+2								
c+3	(MSB)			SUPPORTED D-H ALGORITHM LIST LENGTH (d-c-4)				
c+4								(LSB)
Supported Diffie-Hellman (D-H) algorithms list								
c+5	(MSB)			SUPPORTED D-H ALGORITHM [first]				
•••								
c+8								(LSB)
	⋮							
d-3	(MSB)			SUPPORTED D-H ALGORITHM [last]				
•••								
d								(LSB)
d+1	(MSB)			SUPPORTED CBCS METHODS LIST LENGTH (n-d-2)				
d+2								(LSB)
Supported CbCS methods list								
d+3	SUPPORTED CBCS METHOD [first]							
	⋮							
n	SUPPORTED CBCS METHOD [last]							

The PAGE CODE field shall be set as shown in table 564 to indicate that the Unchangeable CbCS Parameters CbCS page is being returned.

The PAGE LENGTH field indicates the number of bytes that follow in the Unchangeable CbCS Parameters CbCS page.

The KEYS SUPPORT field (see table 565) indicates the type of CbCS master keys and working keys supported.

Table 565 — KEYS SUPPORT field

Code	Description
00b	Reserved
01b	The SCSI target device supports single CbCS master key and a set of CbCS working keys (see 5.13.6.7.11) for the SCSI target device, but the logical units in the SCSI target device do not support CbCS master keys or working keys.
10b	The SCSI target device does not support CbCS master keys or working keys for the SCSI target device, but each logical unit in the SCSI target device supports a single CbCS master key and a set of CbCS working keys for that logical unit.
11b	The SCSI target device supports single CbCS master key and a set of CbCS working keys for the SCSI target device, and each logical unit in the SCSI target device supports a single CbCS master key and a set of CbCS working keys for that logical unit. Keys stored in the logical unit take precedence over keys stored in the SCSI target device (see 5.13.6.7.6).

The MIN CBCS METHOD SUP field (see table 566) indicates how the assignment of the minimum allowable CbCS method is supported.

Table 566 — MIN CBCS METHOD SUP field

Code	Description
00b	Reserved
01b	A single minimum allowed CbCS method (see 5.13.6.7.8) is assigned to all logical units in the SCSI target device, and the SECURITY PROTOCOL well known logical unit is implemented to control its value.
10b	Each logical unit in the SCSI target device is assigned its own minimum allowed CbCS method.
11b	Reserved

The SUPPORTED INTEGRITY CHECK VALUE ALGORITHM LIST LENGTH field indicates the number of bytes that follow in the supported integrity check value algorithms list.

Each SUPPORTED INTEGRITY CHECK VALUE ALGORITHM field indicates one supported algorithm for computing CbCS integrity check values. The values in the SUPPORTED INTEGRITY CHECK VALUE ALGORITHM fields are selected from the codes that table 105 (see 5.13.8) lists as integrity checking (i.e., AUTH) algorithms, except for AUTH_COMBINED.

The SUPPORTED D-H ALGORITHM LIST LENGTH field indicates the number of bytes that follow in the supported Diffie-Hellman (D-H) algorithms list.

Each SUPPORTED D-H ALGORITHM field indicates one supported algorithm for constructing CbCS shared keys. The values in the SUPPORTED D-H ALGORITHM fields are selected from the codes that table 105 (see 5.13.8) lists as Diffie-Hellman algorithms with finite field D-H computations.

The SUPPORTED CBCS METHODS LIST LENGTH field indicates the number of bytes that follow in the supported CbCS methods list.

Each SUPPORTED CBCS METHODS field indicates one supported CbCS method (see 6.26.2.3). The values in the SUPPORTED CBCS METHODS fields are selected from the codes listed in table 267 (see 6.26.2.3).

7.7.4.3.4 Security Token CbCS page

The Security Token CbCS page (see table 567) indicates the value of the security token (see 5.13.6.7.10) for the I_T nexus on which the SECURITY PROTOCOL IN command was received.

Table 567 — Security Token CbCS page format

Bit / Byte	7	6	5	4	3	2	1	0
0	(MSB)			PAGE CODE (003Fh)				
1								(LSB)
2	(MSB)			PAGE LENGTH (n-3)				
3								(LSB)
4	(MSB)			SECURITY TOKEN				
•••								
n								(LSB)

The PAGE CODE field shall be set as shown in table 567 to indicate that the Security Token CbCS page is being returned.

The PAGE LENGTH field indicates the number of bytes that follow in the Security Token CbCS page.

The SECURITY TOKEN field shall contain the security token (see 5.13.6.7.10) for the I_T nexus on which the command was received.

7.7.4.3.5 Current CbCS Parameters CbCS page

The Current CbCS Parameters CbCS page (see table 568) indicates the current values for the CbCS parameters (see 5.13.6.7.15) used by the SCSI target device or logical unit as follows:

a) if the logical unit to which the SECURITY PROTOCOL IN command is addressed is the SECURITY PROTOCOL well known logical unit (see 8.5), then the contents of the Current CbCS Parameters CbCS page apply to the SCSI target device; or

b) if the logical unit to which the SECURITY PROTOCOL IN command is addressed is not the SECURITY PROTOCOL well known logical unit, then the contents of the Current CbCS Parameters CbCS page apply to the addressed logical unit.

Table 568 — Current CbCS Parameters CbCS page format

Bit / Byte	7	6	5	4	3	2	1	0
0	(MSB)			PAGE CODE (0040h)				
1								(LSB)
2	(MSB)			PAGE LENGTH (009Ah)				
3								(LSB)
4				Reserved				
5								
6								
7				MINIMUM ALLOWED CBCS METHOD				
8	(MSB)			POLICY ACCESS TAG				
•••								
11								(LSB)
12				Reserved				
•••								
15								
16	(MSB)			MASTER KEY IDENTIFIER				
•••								
23								(LSB)
24	(MSB)			WORKING KEY 0 IDENTIFIER				
•••								
31								(LSB)
32	(MSB)			WORKING KEY 1 IDENTIFIER				
•••								
39								(LSB)
				⋮				
144	(MSB)			WORKING KEY 15 IDENTIFIER				
•••								
151								(LSB)
152	(MSB)			CLOCK				
•••								
157								(LSB)

The PAGE CODE field shall be set as shown in table 568 to indicate that the Current CbCS Parameters CbCS page is being returned.

The PAGE LENGTH field indicates the number of bytes that follow in the Current CbCS Parameters CbCS page.

The contents of the MINIMUM ALLOWED CBCS METHOD field depend on the logical unit that returned the Current CbCS Parameters CbCS page as follows:

a) if the logical unit is not the SECURITY PROTOCOL well known logical unit, then the MINIMUM ALLOWED CBCS METHOD field indicates the smallest value allowed in the CBCS METHOD field (see 6.26.2.3) of a capability descriptor processed by the enforcement manager (see 5.13.6.7.7) as described in 5.13.6.7.13.2 (i.e., the value of the minimum CbCS method CbCS parameter for the logical unit (see 5.13.6.7.15); or
b) if the SECURITY PROTOCOL well known logical unit is returning the Current CbCS Parameters CbCS page, then the MINIMUM ALLOWED CBCS METHOD field indicates the value that shall be copied to the minimum CbCS method CbCS parameter of any dynamically created logical units (i.e., the initial minimum CbCS method CbCS parameter summarized in 5.13.6.7.15).

The value in the MINIMUM ALLOWED CBCS METHOD field is selected from those listed in table 267 (see 6.26.2.3).

The contents of the POLICY ACCESS TAG field depend on the logical unit that returned the Current CbCS Parameters CbCS page as follows:

a) if the logical unit is not the SECURITY PROTOCOL well known logical unit, then the POLICY ACCESS TAG field indicates the value required in the POLICY ACCESS TAG field (see 6.26.2.3) of a capability descriptor processed by the enforcement manager (see 5.13.6.7.7) as described in 5.13.6.7.13.2 (i.e., the value of the policy access tag CbCS parameter for the logical unit (see 5.13.6.7.15)); or
b) if the SECURITY PROTOCOL well known logical unit is returning the Current CbCS Parameters CbCS page, then the POLICY ACCESS TAG field indicates the value that shall be copied to the policy access tag CbCS parameter of any dynamically created logical units (i.e., the initial policy access tag CbCS parameter summarized in 5.13.6.7.15).

The MASTER KEY IDENTIFIER field specifies the current CbCS shared key identifier (see 5.13.6.7.11.2) for the master key (see 5.13.6.7.11).

The WORKING KEY 0 IDENTIFIER field, the WORKING KEY 1 IDENTIFIER field, the WORKING KEY 2 IDENTIFIER field, through the WORKING KEY 15 IDENTIFIER field contain the current CbCS shared key identifier (see 5.13.6.7.11.2) for the working keys (see 5.13.6.7.11).

The CLOCK field shall be set to the current value of the device clock using the device clock value format defined in 5.2.

7.7.4.3.6 Set Master Key – Seed Exchange CbCS page

The Set Master Key – Seed Exchange CbCS page (see table 569) in a SECURITY PROTOCOL IN command continues a CbCS master key update CCS (see 5.13.6.7.11.4) that was initiated by processing of a Set Master Key – Seed Exchange CbCS page in a SECURITY PROTOCOL OUT command (see 7.7.4.5.5), and completes a Diffie-Hellman key exchange protocol as part of that CCS.

The Diffie-Hellman (D-H) algorithm being used in the CbCS master key update CCS is determined by the contents of the D-H ALGORITHM field in the Set Master Key – Seed Exchange CbCS page in the SECURITY PROTOCOL OUT command (see 7.7.4.5.5) that initiated the CCS. This Diffie-Hellman algorithm specifies the DH_generator value and DH_prime value to be used in the computation of the D-H DATA field as described in this subclause.

Table 569 — Set Master Key – Seed Exchange CbCS page format

Bit / Byte	7	6	5	4	3	2	1	0
0	(MSB)			PAGE CODE (D010h)				
1								(LSB)
2	(MSB)			PAGE LENGTH (n-3)				
3								(LSB)
4	(MSB)			D-H DATA				
•••								
n								(LSB)

The PAGE CODE field shall be set as shown in table 569 to indicate that the Set Master Key – Seed Exchange CbCS page is being returned.

The PAGE LENGTH field indicates the number of bytes that follow in the Set Master Key – Seed Exchange CbCS page.

The contents of the D-H DATA field are computed as follows:

1) a random number, y, is generated having a value between 0 and DH_prime minus one as defined in RFC 4086; and
2) the D-H DATA field is set to DH_generator^{y} modulo DH_prime, where the DH_generator and DH_prime values are identified by the value in the D-H ALGORITHM field.

As part of the successful completion of processing for the SECURITY PROTOCOL IN command transfer of the Set Master Key – Seed Exchange CbCS page the device server and application client store as part of the state maintained for CbCS master key update CCS (see 5.13.6.7.11.4):

a) the authentication key component of the next master key; and
b) the generation key component of the next master key.

The new master key is computed as follows:

1) an initial_seed value that is the value DH_generator^{xy} modulo DH_prime is computed as follows:
   A) the device server computes (DH_generator^{x} modulo DH_prime)^{y}, where DH_generator^{x} is the contents of the D-H DATA field in the SECURITY PROTOCOL OUT command and y is the random number generated by the device server; and
   B) the application client computes (DH_generator^{y} modulo DH_prime)^{x}, where DH_generator^{y} is the contents of the D-H DATA field in the SECURITY PROTOCOL IN command and x is the random number generated by the application client (see 7.7.4.5.5);

2) the generation key component of the new master key is computed using the integrity check value algorithm specified by the integrity check value algorithm field in the capability (see 6.26.2.3) in the CbCS extension descriptor (see 5.13.6.7.16) associated with the SECURITY PROTOCOL IN command that transferred the Set Master Key – Seed Exchange CbCS page. The following inputs are used with the specified integrity check value algorithm:
   A) the concatenation of the initial_seed value computed in step 1) and all of the bytes in the Device Identification VPD page (see 7.8.6) for the logical unit that is processing the CbCS master key update CCS (see 5.13.6.7.11.4) as the string for which the integrity check value is to be computed; and
   B) the generation key component of the current master key (see 5.13.6.7.11) for the logical unit that is processing the CbCS master key update CCS as the cryptographic key;
3) a modified_seed value is computed as follows:
   A) if the least significant bit of the initial_seed value is zero, then the modified_seed value is equal to the initial_seed value with the least significant bit set to one; and
   B) if the least significant bit of the initial_seed value is one, then the modified_seed value is equal to the initial_seed value with the least significant bit set to zero;

   and
4) the authentication key component of the new master key is computed using the integrity check value algorithm specified by the integrity check value algorithm field in the capability (see 6.26.2.3) in the CbCS extension descriptor (see 5.13.6.7.16) associated with the SECURITY PROTOCOL IN command that transferred the Set Master Key – Seed Exchange CbCS page. The following inputs are used with the specified integrity check value algorithm:
   A) the concatenation of the modified_seed value computed in step 3) and all of the bytes in the Device Identification VPD page for the logical unit that is processing the CbCS master key update CCS as the string for which the integrity check value is to be computed; and
   B) the generation key component of the current master key for the logical unit that is processing the CbCS master key update CCS as the cryptographic key.

7.7.4.4 CbCS SECURITY PROTOCOL OUT CDB description

The CbCS SECURITY PROTOCOL OUT CDB has the format defined in 6.41 with the additional requirements described in this subclause.

If the SECURITY PROTOCOL field is set to CbCS (i.e., 07h) in a SECURITY PROTOCOL OUT command, the SECURITY PROTOCOL SPECIFIC field (see table 570) specifies the CbCS page to being sent in the parameter list (see 7.7.4.5). If the CBCS bit is set to one in the Extended INQUIRY Data VPD page (see 7.8.7), the CbCS SECURITY PROTOCOL OUT command support requirements are shown in table 570.

Table 570 — SECURITY PROTOCOL SPECIFIC field for the CbCS SECURITY PROTOCOL OUT command

Code [a]	CbCS page sent	Support	Reference
0000h to 0040h	Reserved		
0041h	Set Policy Access Tag	Optional	7.7.4.5.1
0042h	Set Minimum CbCS Method	Optional	7.7.4.5.2
0043h to CFFFh	Reserved		
D000h	Invalidate Key	See [b]	7.7.4.5.3
D001h	Set Key	See [b]	7.7.4.5.4
D003h to D00Fh	Reserved		
D010h	Set Master Key – Seed Exchange	See [b]	7.7.4.5.5
D011h	Set Master Key – Change Master Key	See [b]	7.7.4.5.6
D012h to FFFFh	Reserved		

[a] If the SECURITY PROTOCOL SPECIFIC field is set to a value that is less than D000h, then the working key specified by the KEY VERSION field in the CbCS capability descriptor (see 6.26.2.3) shall be used to compute the capability key (see 5.13.6.7.12). If the SECURITY PROTOCOL SPECIFIC field is set to a value that is greater than or equal to D000h, then the authentication key component of the master key (see 5.13.6.7.11) shall be used to compute the capability key.

[b] Mandatory if the CAPKEY CbCS method (see 5.13.6.7.8.3) is supported.

If a CbCS SECURITY PROTOCOL OUT command is received with the INC_512 bit set to one, the device server shall terminate the command with CHECK CONDITION status, with the sense key set to ILLEGAL REQUEST, and the additional sense code set to INVALID FIELD IN CDB.

7.7.4.5 CbCS SECURITY PROTOCOL OUT parameter list

7.7.4.5.1 Set Policy Access Tag CbCS page

The Set Policy Access Tag CbCS page (see table 571) specifies a new policy access tag CbCS parameter value or a new initial policy access tag CbCS parameter value.

Table 571 — Set Policy Access Tag CbCS page format

Bit / Byte	7	6	5	4	3	2	1	0
0	(MSB)			PAGE CODE (0041h)				
1								(LSB)
2	(MSB)			PAGE LENGTH (0004h)				
3								(LSB)
4	(MSB)			POLICY ACCESS TAG				
•••								
7								(LSB)

The PAGE CODE field set as shown in table 571 specifies that the Set Policy Access Tag CbCS page is being sent.

The PAGE LENGTH field specifies the number of bytes that follow and is set as shown in table 571 for the Set Policy Access Tag CbCS page. If the PAGE LENGTH field is not set as shown in table 571 or to a larger value, the device server shall terminate the command with CHECK CONDITION status, with the sense key set to ILLEGAL REQUEST, and the additional sense code set to INVALID FIELD IN PARAMETER LIST.

The CbCS parameter (see 5.13.6.7.15) that is set to the contents of the POLICY ACCESS TAG field depends on the logical unit that is processing the Set Policy Access Tag CbCS page as follows:

a) if the logical unit is not the SECURITY PROTOCOL well known logical unit, then the contents of the POLICY ACCESS TAG field shall be placed in the policy access tag CbCS parameter for the logical unit; or
b) if the SECURITY PROTOCOL well known logical unit is processing the Set Policy Access Tag CbCS page, then the contents of the POLICY ACCESS TAG field shall be placed in the initial policy access tag CbCS parameter.

7.7.4.5.2 Set Minimum CbCS Method CbCS page

The Set Minimum CbCS Method CbCS page (see table 572) specifies a new minimum CbCS method CbCS parameter value or a new initial minimum CbCS method CbCS parameter value.

Table 572 — Set Minimum CbCS Method CbCS page format

Bit / Byte	7	6	5	4	3	2	1	0
0	(MSB)			PAGE CODE (0042h)				
1								(LSB)
2	(MSB)			PAGE LENGTH (0002h)				
3								(LSB)
4	Reserved							
5	MINIMUM ALLOWED CBCS METHOD							

The PAGE CODE field set as shown in table 572 specifies that the Set Minimum CbCS Method CbCS page is being sent.

The PAGE LENGTH field specifies the number of bytes that follow and is set as shown in table 572 for the Set Minimum CbCS Method CbCS page. If the PAGE LENGTH field is not set as shown in table 572 or to a larger value, the device server shall terminate the command with CHECK CONDITION status, with the sense key set to ILLEGAL REQUEST, and the additional sense code set to INVALID FIELD IN PARAMETER LIST.

The CbCS parameter or CbCS parameters (see 5.13.6.7.15) that are set to the contents of the MINIMUM ALLOWED CBCS METHOD field depends on the logical unit that is processing the Set Minimum CbCS Method CbCS page and the contents of the MIN CBCS METHOD SUP field in the Unchangeable CbCS Parameters CbCS page (see 7.7.4.3.3) as shown in table 573.

Table 573 — Minimum CbCS Method CbCS Parameter set

MIN CBCS METHOD SUP field	CbCS Parameter set based on the logical unit that processes the Set Minimum CbCS Method CbCS page	
	Not the SECURITY PROTOCOL well known logical unit	SECURITY PROTOCOL well known logical unit
00b	Reserved	
01b	The device server terminates the command with CHECK CONDITION status with the sense key set to ILLEGAL REQUEST, and the additional sense code set to INVALID FIELD IN PARAMETER LIST	All minimum CbCS method CbCS parameters for all logical units in the SCSI target device, and the initial minimum CbCS method, if any
10b	The minimum CbCS method CbCS parameter for the logical unit that processes the Set Minimum CbCS Method CbCS page	The initial minimum CbCS method CbCS parameter
11b	Reserved	

If the MINIMUM ALLOWED CBCS METHOD field specifies a value that does not appear in the supported CbCS methods list in the Unchangeable CbCS Parameters CbCS page (see 7.7.4.3.3), then the device server shall terminate the command with CHECK CONDITION status, with the sense key set to ILLEGAL REQUEST, and the additional sense code set to INVALID FIELD IN PARAMETER LIST.

7.7.4.5.3 Invalidate Key CbCS page

The Invalidate Key CbCS page (see table 574) causes a working key (see 5.13.6.7.11) to be invalidated. After the successful processing of an Invalidate Key CbCS page, the working key with the specified key version shall not be valid for the purposes of enforcement manager (see 5.13.6.7.7) CbCS extension descriptor

validation (see 5.13.6.7.13.2).

Table 574 — Invalidate Key CbCS page format

Bit / Byte	7	6	5	4	3	2	1	0
0	(MSB)							
1				PAGE CODE (D000h)				(LSB)
2	(MSB)							
3				PAGE LENGTH (0004h)				(LSB)
4								
5				Reserved				
6								
7	Reserved				KEY VERSION			

The PAGE CODE field set as shown in table 574 specifies that the Invalidate Key CbCS page is being sent.

The PAGE LENGTH field specifies the number of bytes that follow and is set as shown in table 574 for the Invalidate Key CbCS page. If the PAGE LENGTH field is is not set as shown in table 574 or to a larger value, the device server shall terminate the command with CHECK CONDITION status, with the sense key set to ILLEGAL REQUEST, and the additional sense code set to INVALID FIELD IN PARAMETER LIST.

The KEY VERSION field specifies which working key (e.g., a KEY VERSION field set to four specifies that the working key whose Current CbCS Parameters CbCS page (see 7.7.4.3.5) key identifier is returned in the WORKING KEY 4 IDENTIFIER field) is to be invalidated as follows:

a) if the Invalidate Key CbCS page is processed by a logical unit that is not the SECURITY PROTOCOL well known logical unit, then the specified working key for that logical unit shall be invalidated; or
b) if the Invalidate Key CbCS page is processed by the SECURITY PROTOCOL well known logical unit, then the specified target-wide working key (see 5.13.6.7.11) shall be invalidated.

The CbCS shared key identifier (see 5.13.6.7.11.2) for the invalidated working key shall be set to FFFF FFFF FFFF FFFEh.

It shall not be an error to invalidate a working key that is already invalid.

7.7.4.5.4 Set Key CbCS page

The Set Key CbCS page (see table 575) causes a working key (see 5.13.6.7.11) to be set to a new value. After the successful processing of a Set Key CbCS page, the working key with the specified key version shall be valid for the purposes of enforcement manager (see 5.13.6.7.7) CbCS extension descriptor validation (see 5.13.6.7.13.2).

Table 575 — Set Key CbCS page format

Bit / Byte	7	6	5	4	3	2	1	0
0	(MSB)			PAGE CODE (D001h)				
1								(LSB)
2	(MSB)			PAGE LENGTH (0020h)				
3								(LSB)
4				Reserved				
5								
6								
7	Reserved				KEY VERSION			
8	(MSB)			KEY IDENTIFIER				
•••								
15								(LSB)
16	(MSB)			SEED				
•••								
35								(LSB)

The PAGE CODE field set as shown in table 575 specifies that the Set Key CbCS page is being sent.

The PAGE LENGTH field specifies the number of bytes that follow and is set as shown in table 575 for the Set Key CbCS page. If the PAGE LENGTH field is is not set as shown in table 575 or to a larger value, the device server shall terminate the command with CHECK CONDITION status, with the sense key set to ILLEGAL REQUEST, and the additional sense code set to INVALID FIELD IN PARAMETER LIST.

The KEY VERSION field specifies which working key (e.g., a KEY VERSION field set to six specifies that the working key whose Current CbCS Parameters CbCS page (see 7.7.4.3.5) key identifier is returned in the WORKING KEY 6 IDENTIFIER field) is to be set as follows:

a) if the Set Key CbCS page is processed by a logical unit that is not the SECURITY PROTOCOL well known logical unit, then the specified working key for that logical unit shall be set; or
b) if the Set Key CbCS page is processed by the SECURITY PROTOCOL well known logical unit, then the specified target-wide working key (see 5.13.6.7.11) shall be set.

The KEY IDENTIFIER field specifies the value to which the CbCS shared key identifier (see 5.13.6.7.11.2) shall be set for the working key affected by the Set Key CbCS page. If the KEY IDENTIFIER field is set to a value that table 89 (see 5.13.6.7.11.2) describes as reserved in the CbCS pages that change CbCS shared key values, then the device server shall terminate the command with CHECK CONDITION status, with the sense key set to ILLEGAL REQUEST, and the additional sense code set to INVALID FIELD IN PARAMETER LIST.

The SEED field specifies a random number that is an input to the computation of the new working key value.

The value to which the specified working key is set shall be computed using the integrity check value algorithm specified by the INTEGRITY CHECK VALUE ALGORITHM field in the capability (see 6.26.2.3) in the CbCS

extension descriptor (see 5.13.6.7.16) associated with the SECURITY PROTOCOL OUT command that sent the Set Key CbCS page. The following inputs shall be used with the specified integrity check value algorithm:

a) the contents of the SEED field in the Set Key CbCS page as the string for which the integrity check value is to be computed; and
b) the generation key component of the master key (see 5.13.6.7.11) for the logical unit that is processing the Set Key CbCS page as the cryptographic key.

7.7.4.5.5 Set Master Key – Seed Exchange CbCS page

The Set Master Key – Seed Exchange CbCS page (see table 576) in a SECURITY PROTOCOL OUT command initiates a CbCS master key update CCS (see 5.13.6.7.11.4) and begins a Diffie-Hellman key exchange protocol as part of that CCS.

Table 576 — Set Master Key – Seed Exchange CbCS page format

Bit / Byte	7	6	5	4	3	2	1	0
0	(MSB)			PAGE CODE (D010h)				
1								(LSB)
2	(MSB)			PAGE LENGTH (n-3)				
3								(LSB)
4	(MSB)			D-H ALGORITHM				
•••								
7								(LSB)
8	(MSB)			D-H DATA LENGTH (n-11)				
•••								
11								(LSB)
12	(MSB)			D-H DATA				
•••								
n								(LSB)

The PAGE CODE field set as shown in table 576 specifies that the Set Master Key – Seed Exchange CbCS page is being sent.

The PAGE LENGTH field specifies the number of bytes that follow in the Set Master Key – Seed Exchange CbCS page. If the PAGE LENGTH field is set to a value that is smaller than ten, then the device server shall terminate the command with CHECK CONDITION status, with the sense key set to ILLEGAL REQUEST, and the additional sense code set to INVALID FIELD IN PARAMETER LIST.

The D-H ALGORITHM field specifies the Diffie-Hellman (D-H) algorithm to be used in the seed exchange protocol. The value in the D-H ALGORITHM field is selected from the codes that table 105 (see 5.13.8) lists as Diffie-Hellman algorithms with finite field D-H computations. The Diffie-Hellman algorithm selected specifies the DH_generator value and DH_prime value to be used in the computation of the D-H DATA field as described in this subclause.

If the value in the D-H ALGORITHM field does not appear in the Supported Diffie-Hellman algorithms list in the Unchangeable CbCS Parameters CbCS page (see 7.7.4.3.3), then the device server shall terminate the command with CHECK CONDITION status, with the sense key set to ILLEGAL REQUEST, and the additional sense code set to INVALID FIELD IN PARAMETER LIST.

The D-H DATA LENGTH field specifies the number of bytes that follow in D-H DATA field. If the D-H DATA LENGTH field is set to a value that is inconsistent with the Diffie-Hellman algorithm specified by the D-H ALGORITHM field,

then the device server shall terminate the command with CHECK CONDITION status, with the sense key set to ILLEGAL REQUEST, and the additional sense code set to INVALID FIELD IN PARAMETER LIST.

The contents of the D-H DATA field are computed as follows:

1) a random number, x, is generated having a value between 0 and DH_prime minus one observing the requirements in RFC 4086; and
2) the D-H DATA field is set to DH_generator^{x} modulo DH_prime, where the DH_generator and DH_prime values are identified by the value in the D-H ALGORITHM field.

7.7.4.5.6 Set Master Key – Change Master Key CbCS page

The Set Master Key – Change Master Key CbCS page (see table 577) concludes a CbCS master key update CCS (see 5.13.6.7.11.4) and changes the master key components to the values computed as part of processing completion (see 7.7.4.3.6) for the SECURITY PROTOCOL IN command that transferred the Set Master Key – Seed Exchange CbCS page.

Table 577 — Set Master Key – Change Master Key CbCS page format

Bit / Byte	7	6	5	4	3	2	1	0
0	(MSB)							
1				PAGE CODE (D011h)				(LSB)
2	(MSB)							
3				PAGE LENGTH (n-3)				(LSB)
4								
•••				Reserved				
7								
8	(MSB)							
•••				KEY IDENTIFIER				
15								(LSB)
16	(MSB)							
•••				APPLICATION CLIENT D-H DATA LENGTH (k-19)				
19								(LSB)
20	(MSB)							
•••				APPLICATION CLIENT D-H DATA				
k								(LSB)
k+1	(MSB)							
•••				DEVICE SERVER D-H DATA LENGTH (n-(k-4))				
k+4								(LSB)
k+5	(MSB)							
•••				DEVICE SERVER D-H DATA				
n								(LSB)

The PAGE CODE field set as shown in table 577 specifies that the Set Master Key – Change Master Key CbCS page is being sent.

The PAGE LENGTH field specifies the number of bytes that follow in the Set Master Key – Change Master Key CbCS page. If the PAGE LENGTH field is set to a value that is smaller than 24, then the device server shall terminate the command with CHECK CONDITION status, with the sense key set to ILLEGAL REQUEST, and the additional sense code set to INVALID FIELD IN PARAMETER LIST.

The KEY IDENTIFIER field specifies the value to which the master key CbCS shared key identifier (see 5.13.6.7.11.2) shall be set. If the KEY IDENTIFIER field is set to a value that table 89 (see 5.13.6.7.11.2) describes as reserved in the CbCS pages that change CbCS shared key values, then the device server shall terminate the command with CHECK CONDITION status, with the sense key set to ILLEGAL REQUEST, and the additional sense code set to INVALID FIELD IN PARAMETER LIST.

The APPLICATION CLIENT D-H DATA LENGTH field specifies the number of bytes that follow in the APPLICATION CLIENT D-H DATA field. If the contents of the APPLICATION CLIENT D-H DATA LENGTH field do not match the number of Diffie-Hellman (D-H) data bytes that the device server received in the SECURITY PROTOCOL OUT command that sent the Set Master Key – Seed Exchange CbCS page (see 7.7.4.5.5) and initiated the current CbCS master key update CCS (see 5.13.6.7.11.4), then the master key shall not be modified and the command shall be terminated with CHECK CONDITION status, with the sense key set to ILLEGAL REQUEST, and the additional sense code set to INVALID FIELD IN PARAMETER LIST.

The APPLICATION CLIENT D-H DATA field contains Diffie-Hellman data field that was sent in the Set Master Key – Seed Exchange CbCS page that initiated the current CbCS master key update CCS. If the contents of the APPLICATION CLIENT D-H DATA field do not match Diffie-Hellman data that the device server received in the SECURITY PROTOCOL OUT command that sent the Set Master Key – Seed Exchange CbCS page that initiated the current CbCS master key update CCS (see 5.13.6.7.11.4), then the master key shall not be modified and the command shall be terminated with CHECK CONDITION status, with the sense key set to ILLEGAL REQUEST, and the additional sense code set to INVALID FIELD IN PARAMETER LIST.

The DEVICE SERVER D-H DATA LENGTH field specifies the number of bytes that follow in the DEVICE SERVER D-H DATA field. If the contents of the DEVICE SERVER D-H DATA LENGTH field do not match the number of Diffie-Hellman data bytes that the device server returned in the SECURITY PROTOCOL IN command that returned the Set Master Key – Seed Exchange CbCS page (see 7.7.4.3.6) in the current CbCS master key update CCS, then the master key shall not be modified and the command shall be terminated with CHECK CONDITION status, with the sense key set to ILLEGAL REQUEST, and the additional sense code set to INVALID FIELD IN PARAMETER LIST.

The DEVICE SERVER D-H DATA field contains Diffie-Hellman data field that was returned in the Set Master Key – Seed Exchange CbCS page that the device server returned in response to a SECURITY PROTOCOL IN command as part of the current CbCS master key update CCS. If the contents of the DEVICE SERVER D-H DATA field do not match Diffie-Hellman data that the device server returned in the SECURITY PROTOCOL IN command that send the Set Master Key – Seed Exchange CbCS page, then the master key shall not be modified and the command shall be terminated with CHECK CONDITION status, with the sense key set to ILLEGAL REQUEST, and the additional sense code set to INVALID FIELD IN PARAMETER LIST.

7.8 Vital product data parameters

7.8.1 Vital product data parameters overview and page codes

This subclause describes the vital product data (VPD) page structure and the VPD pages (see table 578) that are applicable to all SCSI devices. These VPD pages are returned by an INQUIRY command with the EVPD bit set to one (see 6.6) and contain product information about a logical unit and SCSI target device.

Table 578 — Vital product data page codes

VPD Page Name	Page code	Reference	Support
ASCII Information	01h to 7Fh	7.8.3	Optional
ATA Information	89h	SAT-3	See SAT-3
CFA Profile Information	8Ch	7.8.4	Optional
Device Constituents	8Bh	7.8.5	Optional
Device Identification	83h	7.8.6	Mandatory
Extended INQUIRY Data	86h	7.8.7	Optional
Management Network Addresses	85h	7.8.8	Optional
Mode Page Policy	87h	7.8.9	Optional
Power Condition	8Ah	7.8.10	Optional
Power Consumption	8Dh	7.8.11	Optional
Protocol Specific Logical Unit Information	90h	7.8.12	Protocol specific [a]
Protocol Specific Port Information	91h	7.8.13	Protocol specific [a]
SCSI Ports	88h	7.8.14	Optional
Software Interface Identification	84h	7.8.15	Optional
Supported VPD Pages	00h	7.8.16	Mandatory
Third-party Copy	8Fh	7.8.17	Optional
Unit Serial Number	80h	7.8.18	Optional
Restricted (see applicable command standard)	B0h to BFh		
Obsolete [b]			
Vendor specific [c]			
Reserved	All other codes		
A numeric ordered listing of VPD page codes is provided in F.7.			

[a] See applicable SCSI transport protocol standard for support requirements.
[b] The following page codes are obsolete: 81h and 82h.
[c] The following page codes are vendor specific: C0h to FFh.

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

7.8.2 VPD page format for all device types

This subclause describes the VPD page structure that is applicable to all SCSI devices. VPD pages specific to each device type are described in the command standard that applies to that device type.

An INQUIRY command with the EVPD bit set to one (see 6.6) specifies that the device server return a VPD page using the format defined in table 579.

Table 579 — VPD page format

Bit / Byte	7	6	5	4	3	2	1	0
0	PERIPHERAL QUALIFIER			PERIPHERAL DEVICE TYPE				
1	PAGE CODE							
2	(MSB)			PAGE LENGTH (n-3)				
3								(LSB)
4	VPD parameters							
•••								
n								

The PERIPHERAL QUALIFIER field and the PERIPHERAL DEVICE TYPE field are the same as defined for standard INQUIRY data (see 6.6.2).

The PAGE CODE field (see 7.8.1) identifies the VPD page and contains the same value as in the PAGE CODE field in the INQUIRY CDB (see 6.6).

The PAGE LENGTH field indicates the length in bytes of the VPD parameters that follow this field. The contents of the PAGE LENGTH field are not altered based on the allocation length (see 4.2.5.6).

The VPD parameters are defined for each VPD page code. If the PERIPHERAL QUALIFIER field is not set to 000b, the contents of the PAGE LENGTH field and the VPD parameters are outside the scope of this standard.

7.8.3 ASCII Information VPD page

The ASCII Information VPD page (see table 580) contains information for the field replaceable unit code returned in the sense data (see 4.5).

Table 580 — ASCII Information VPD page

Bit / Byte	7	6	5	4	3	2	1	0
0	PERIPHERAL QUALIFIER			PERIPHERAL DEVICE TYPE				
1	PAGE CODE (01h to 7Fh)							
2	(MSB)			PAGE LENGTH (n-3)				
3								(LSB)
4	ASCII LENGTH (m-4)							
5	(MSB)			ASCII INFORMATION				
•••								
m								(LSB)
m+1	Vendor specific information							
•••								
n								

The PERIPHERAL QUALIFIER field, PERIPHERAL DEVICE TYPE field, PAGE CODE field, and PAGE LENGTH field are defined in 7.8.2.

The value in the PAGE CODE field is associated with the FIELD REPLACEABLE UNIT CODE field returned in the sense data.

NOTE 59 - 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 VPD pages for the upper code values.

The ASCII LENGTH field indicates the length in bytes of the ASCII INFORMATION field. A value of zero in this field indicates that no ASCII information is available for the indicated page code. The contents of the ASCII LENGTH field are not altered based on the allocation length (see 4.2.5.6).

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 ASCII printable characters (i.e., code values 20h through 7Eh) and shall be terminated with a NULL (00h) character.

The contents of the vendor specific information are not defined in this standard.

7.8.4 CFA Profile Information VPD page

The CFA Profile Information VPD page (see table 581) provides information on the CFA profiles, if any, that are supported by the device server.

Table 581 — CFA Profile Information VPD page

Bit / Byte	7	6	5	4	3	2	1	0
0	PERIPHERAL QUALIFIER			PERIPHERAL DEVICE TYPE				
1	PAGE CODE (8Ch)							
2	(MSB)			PAGE LENGTH (n-3)				
3								(LSB)
	CFA profile descriptor list							
4				CFA profile descriptor (see table 582) [first]				
•••								
7								
	⋮							
n-3				CFA profile descriptor (see table 582) [last]				
•••								
n								

The PERIPHERAL QUALIFIER field, PERIPHERAL DEVICE TYPE field, and PAGE LENGTH field are defined in 7.8.2.

The PAGE CODE field is defined in 7.8.2 and shall be set as shown in table 581 for the CFA Profile Information VPD page.

Each CFA profile descriptor (see table 582) contains identifying information for one CFA profile supported by the device server described in the standard INQUIRY data (see 6.6.2).

Table 582 — CFA profile descriptor

Bit / Byte	7	6	5	4	3	2	1	0
0	CFA PROFILE SUPPORTED							
1	Reserved							
2	(MSB)			SEQUENTIAL WRITE DATA SIZE				
3								(LSB)

The CFA PROFILE SUPPORTED field indicates a CFA profile number (see VPG1) that the device server supports.

The SEQUENTIAL WRITE DATA SIZE field indicates the preferred number of logical blocks in a stream field write operation (see VPG1) for the CFA profile indicated by the CFA PROFILE SUPPORTED field. If the SEQUENTIAL WRITE DATA SIZE field is set to zero, then no preferred number of logical blocks is indicated for stream field write operations by this CFA profile descriptor.

7.8.5 Device Constituents VPD page

The Device Constituents VPD page (see table 583) provides identifying information for the constituents (e.g., logical units in other SCSI devices, microcode, storage medium, or vendor specific information technology components) of the logical unit described in the standard INQUIRY data (see 6.6.2).

Table 583 — Device Constituents VPD page

Bit / Byte	7	6	5	4	3	2	1	0
0	PERIPHERAL QUALIFIER			PERIPHERAL DEVICE TYPE				
1	PAGE CODE (8Bh)							
2	(MSB)			PAGE LENGTH (n-3)				
3								(LSB)
	Constituent descriptor list							
4	Constituent descriptor (see table 584) [first]							
•••								
	⋮							
•••	Constituent descriptor (see table 584) [last]							
n								

The PERIPHERAL QUALIFIER field, PERIPHERAL DEVICE TYPE field, and PAGE LENGTH field are defined in 7.8.2.

The PAGE CODE field is defined in 7.8.2 and shall be set as shown in table 583 for the Device Constituents VPD page.

Each constituent descriptor (see table 584) contains identifying information for one constituent of the logical unit described in the standard INQUIRY data (see 6.6.2).

Table 584 — Constituent descriptor

Bit / Byte	7	6	5	4	3	2	1	0
0	(MSB)			CONSTITUENT TYPE				
1								(LSB)
2	CONSTITUENT DEVICE TYPE							
3	Reserved							
4	(MSB)			T10 VENDOR IDENTIFICATION				
•••								
11								(LSB)
12	(MSB)			PRODUCT IDENTIFICATION				
•••								
27								(LSB)
28	(MSB)			PRODUCT REVISION LEVEL				
•••								
31								(LSB)
32	Reserved							
33								
34	(MSB)			CONSTITUENT DESCRIPTOR LENGTH (n-35)				
35								(LSB)
	Constituent specific descriptor list							
36	Constituent specific descriptor [first]							
•••								
	⋮							
•••	Constituent specific descriptor [last]							
n								

The CONSTITUENT TYPE field (see table 585) indicates the constituent descriptor type.

Table 585 — CONSTITUENT TYPE field

Code	Description
0000h	Reserved
0001h	Virtual tape library
0002h	Virtual tape drive
0003h	Direct access block device
0004h to FFFFh	Reserved

The CONSTITUENT DEVICE TYPE field (see table 586) indicates the constituent descriptor type.

Table 586 — CONSTITUENT DEVICE TYPE field

Code	Description	Reference
00h to 1Fh	One of the values defined for the PERIPHERAL DEVICE TYPE field in the standard INQUIRY data	6.6.2
20h to FEh	Reserved	
FFh	Unknown	

The T10 VENDOR IDENTIFICATION field, PRODUCT IDENTIFICATION field, and the PRODUCT REVISION LEVEL field are identifying information for one constituent of the logical unit described in the standard INQUIRY data, and are as defined in 6.6.2.

The CONSTITUENT DESCRIPTOR LENGTH field indicates the length of the constituent specific descriptor list.

Each constituent specific descriptor (see table 587) contains information that is specific to the constituent.

Table 587 — Constituent specific descriptor format

Bit / Byte	7	6	5	4	3	2	1	0
0	CONSTITUENT SPECIFIC TYPE							
1	Reserved							
2	(MSB)			ADDITIONAL LENGTH (n-3)				
3								(LSB)
4	Constituent specific data							
•••								
n								

The CONSTITUENT SPECIFIC TYPE field (see table 588) indicates the type of constituent specific data in this descriptor.

Table 588 — CONSTITUENT SPECIFIC TYPE field

Code	Description
01h	VPD page
FFh	Vendor specific
all others	Reserved

The ADDITIONAL LENGTH field indicates the number of bytes that follow in this descriptor.

The contents of the constituent specific data depend on the value in the CONSTITUENT SPECIFIC TYPE field.

If the CONSTITUENT SPECIFIC TYPE field is set to 01h, then:

a) the constituent specific data contains a VPD page (see 7.8.1); and
b) the device server shall not set the PAGE CODE field in that VPD page to 8Bh (i.e., the VPD page in the constituent specific data shall not be a Device Constituents VPD page).

If SCSI port specific information or protocol specific information is reported in the Device Constituents VPD page, then that information shall be associated with the device constituent, not the addressed logical unit.

7.8.6 Device Identification VPD page

7.8.6.1 Device Identification VPD page overview

The Device Identification VPD page (see table 589) provides the means to retrieve designation descriptors applying to the logical unit. Logical units may have more than one designation descriptor (e.g., if several types or associations of designator are supported). Designators consist of one or more of the following:

a) logical unit names;
b) SCSI target port identifiers;
c) SCSI target port names;
d) SCSI device names;
e) relative target port identifiers;
f) primary target port group number; or
g) logical unit group number.

Designation descriptors 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. Application clients should use the designation descriptors during system configuration activities to determine whether alternate paths exist for the same peripheral device.

Table 589 — Device Identification VPD page

Bit / Byte	7	6	5	4	3	2	1	0
0	PERIPHERAL QUALIFIER			PERIPHERAL DEVICE TYPE				
1	PAGE CODE (83h)							
2	(MSB)			PAGE LENGTH (n-3)				
3								(LSB)
	Designation descriptor list							
4	Designation descriptor [first]							
•••								
	⋮							
•••	Designation descriptor [last]							
n								

The PERIPHERAL QUALIFIER field, PERIPHERAL DEVICE TYPE field, and PAGE LENGTH field are defined in 7.8.2.

The PAGE CODE field is defined in 7.8.2 and shall be set as shown in table 589 for the Device Identification VPD page.

Each designation descriptor (see table 590) contains information identifying the logical unit, SCSI target device containing the logical unit, or access path (i.e., target port) used by the command and returned parameter data. The Device Identification VPD page shall contain the designation descriptors enumerated in 7.8.6.2.

Table 590 — Designation descriptor

Bit / Byte	7	6	5	4	3	2	1	0
0	PROTOCOL IDENTIFIER				CODE SET			
1	PIV	Reserved	ASSOCIATION		DESIGNATOR TYPE			
2	Reserved							
3	DESIGNATOR LENGTH (n-3)							
4	DESIGNATOR							
•••								
n								

The PROTOCOL IDENTIFIER field may indicate the SCSI transport protocol to which the designation descriptor applies. If the ASSOCIATION field is set to a value other than 01b (i.e., target port) or 10b (i.e., SCSI target device) or the PIV bit is set to zero, then the PROTOCOL IDENTIFIER field contents are reserved. If the ASSOCIATION field is set to a value of 01b or 10b and the PIV bit is set to one, then the PROTOCOL IDENTIFIER field shall contain one of the values shown in table 479 (see 7.6.1) to indicate the SCSI transport protocol to which the designation descriptor applies.

The CODE SET field contains a code set enumeration (see 4.3.3) that indicates the format of the DESIGNATOR field.

A protocol identifier valid (PIV) bit set to zero indicates the PROTOCOL IDENTIFIER field contents are reserved. If the ASSOCIATION field is set to a value of 01b or 10b, then a PIV bit set to one indicates the PROTOCOL IDENTIFIER field contains a valid protocol identifier selected from the values shown in table 479 (see 7.6.1). If the ASSOCIATION field is set to a value other than 01b or 10b, then the PIV bit contents are reserved.

The ASSOCIATION field indicates the entity with which the DESIGNATOR field is associated, as described in table 591. If a logical unit returns a designation descriptor with the ASSOCIATION field set to 00b or 10b, the logical unit shall return the same descriptor when it is accessed through any other I_T nexus.

Table 591 — ASSOCIATION field

Code	Description
00b	The DESIGNATOR field is associated with the addressed logical unit.
01b	The DESIGNATOR field is associated with the target port that received the command.
10b	The DESIGNATOR field is associated with the SCSI target device that contains the addressed logical unit.
11b	Reserved

The DESIGNATOR TYPE field (see table 592) indicates the format and assignment authority for the designator.

Table 592 — DESIGNATOR TYPE field

Code	Description	Reference
0h	Vendor specific	7.8.6.3
1h	T10 vendor ID based	7.8.6.4
2h	EUI-64 based	7.8.6.5
3h	NAA	7.8.6.6
4h	Relative target port identifier	7.8.6.7
5h	Target port group	7.8.6.8
6h	Logical unit group	7.8.6.9
7h	MD5 logical unit identifier	7.8.6.10
8h	SCSI name string	7.8.6.11
9h	Protocol specific port identifier	7.8.6.12
Ah to Fh	Reserved	

The DESIGNATOR LENGTH field indicates the length in bytes of the DESIGNATOR field. The contents of the DESIGNATOR LENGTH field are not altered based on the allocation length (see 4.2.5.6).

The DESIGNATOR field contains the designator as described by the ASSOCIATION field, DESIGNATOR TYPE field, CODE SET field, and DESIGNATOR LENGTH field.

7.8.6.2 Device designation descriptor requirements

7.8.6.2.1 Designation descriptors for logical units other than well known logical units

For each logical unit that is not a well known logical unit, the Device Identification VPD page shall include at least one designation descriptor in which a logical unit name (see SAM-5) is indicated. The designation descriptor shall have the ASSOCIATION field set to 00b (i.e., logical unit) and the DESIGNATOR TYPE field set to:

a) 1h (i.e., T10 vendor ID based);
b) 2h (i.e., EUI-64-based);
c) 3h (i.e., NAA); or
d) 8h (i.e., SCSI name string).

At least one designation descriptor should have the DESIGNATOR TYPE field set to:

a) 2h (i.e., EUI-64-based);
b) 3h (i.e., NAA); or
c) 8h (i.e., SCSI name string).

In the case of virtual logical units (e.g., volume sets as defined by SCC-2), designation descriptors should contain a DESIGNATOR TYPE field set to:

a) 2h (i.e., EUI-64-based);
b) 3h (i.e., NAA); or
c) 8h (i.e., SCSI name string).

In the case of virtual logical units that have an EUI-64 based designation descriptor (see 7.8.6.5) the DESIGNATOR LENGTH field should be set to:

a) 0Ch (i.e., EUI-64-based 12-byte); or
b) 10h (i.e., EUI-64-based 16-byte).

In the case of virtual logical units that have an NAA designation descriptor (see 7.8.6.6) the NAA field should be set to 6h (i.e., IEEE Registered Extended).

The Device Identification VPD page shall contain the same set of designation descriptors with the ASSOCIATION field set to 00b (i.e., logical unit) regardless of the I_T nexus being used to retrieve the designation descriptors.

For logical units that are not well known logical units, the requirements for SCSI target device designation descriptors are defined in 7.8.6.2.4 and the requirements for SCSI target port designation descriptors are defined in 7.8.6.2.3.

7.8.6.2.2 Designation descriptors for well known logical units

Well known logical units shall not return any designation descriptors with the ASSOCIATION field set to 00b (i.e., logical unit).

The Device Identification VPD page shall contain the same set of designation descriptors with the ASSOCIATION field set to 10b (i.e., SCSI target device) regardless of the I_T nexus being used to retrieve the designation descriptors.

7.8.6.2.3 Designation descriptors for SCSI target ports

7.8.6.2.3.1 Relative target port identifiers

For the target port through which the Device Identification VPD page is accessed, the Device Identification VPD page should include one designation descriptor with the ASSOCIATION field set to 01b (i.e., target port) and the DESIGNATOR TYPE field set to 4h (i.e., relative target port identifier) identifying the target port being used to retrieve the designation descriptors.

7.8.6.2.3.2 Target port names or identifiers

For the SCSI target port through which the Device Identification VPD page is accessed, the Device Identification VPD page should include one designation descriptor in which the target port name or identifier (see SAM-5) is indicated. The designation descriptor, if any, shall have the ASSOCIATION field set to 01b (i.e., target port) and the DESIGNATOR TYPE field set to:

a) 2h (i.e., EUI-64-based);
b) 3h (i.e., NAA); or
c) 8h (i.e., SCSI name string).

If the SCSI transport protocol standard for the target port defines target port names, the designation descriptor, if any, shall contain the target port name. If the SCSI transport protocol for the target port does not define target port names, the designation descriptor, if any, shall contain the target port identifier.

7.8.6.2.4 Designation descriptors for SCSI target devices

If the SCSI target device contains a well known logical unit, the Device Identification VPD page shall have one or more designation descriptors for the SCSI target device. If the SCSI target device does not contain a well known logical unit, the Device Identification VPD page should have one or more designation descriptors for the SCSI target device.

Each SCSI target device designation descriptor, if any, shall have the ASSOCIATION field set to 10b (i.e., SCSI target device) and the DESIGNATOR TYPE field set to:

a) 2h (i.e., EUI-64-based);
b) 3h (i.e., NAA); or
c) 8h (i.e., SCSI name string).

The Device Identification VPD page shall contain designation descriptors, if any, for all the SCSI device names for all the SCSI transport protocols supported by the SCSI target device.

7.8.6.3 Vendor specific designator format

If the designator type is 0h (i.e., vendor specific), no assignment authority was used and there is no guarantee that the designator is globally unique (i.e., the identifier is vendor specific). Table 593 defines the DESIGNATOR field format.

Table 593 — Vendor specific DESIGNATOR field format

Bit / Byte	7	6	5	4	3	2	1	0
0	VENDOR SPECIFIC IDENTIFIER							
•••								
n								

7.8.6.4 T10 vendor ID based designator format

If the designator type is 1h (i.e., T10 vendor ID based), the DESIGNATOR field has the format shown in table 594.

Table 594 — T10 vendor ID based DESIGNATOR field format

Bit / Byte	7	6	5	4	3	2	1	0
0	(MSB) T10 VENDOR IDENTIFICATION							
•••								
7								(LSB)
8	VENDOR SPECIFIC IDENTIFIER							
•••								
n								

The T10 VENDOR IDENTIFICATION field contains eight bytes of left-aligned ASCII data (see 4.3.1) identifying the manufacturer of the product. The data shall be left aligned within this field.The T10 vendor identification shall be one assigned by INCITS. A list of assigned T10 vendor identifications is in Annex G and on the T10 web site (http://www.T10.org).

NOTE 60 - The T10 web site (http://www.t10.org) provides a convenient means to request an identification code.

The organization associated with the T10 vendor identification is responsible for ensuring that the VENDOR SPECIFIC DESIGNATOR field is unique in a way that makes the entire DESIGNATOR field unique. A recommended method of constructing a unique DESIGNATOR field is to concatenate the PRODUCT IDENTIFICATION field from the standard INQUIRY data (see 6.6.2) and the PRODUCT SERIAL NUMBER field from the Unit Serial Number VPD page (see 7.8.18).

7.8.6.5 EUI-64 based designator format

7.8.6.5.1 EUI-64 based designator format overview

If the designator type is 2h (i.e., EUI-64 based identifier), the DESIGNATOR LENGTH field (see table 595) indicates the format of the designation descriptor.

Table 595 — EUI-64 based designator lengths

Designator Length	Description	Reference
08h	EUI-64 identifier	7.8.6.5.2
0Ch	EUI-64 based 12-byte identifier	7.8.6.5.3
10h	EUI-64 based 16-byte identifier	7.8.6.5.4
All other values	Reserved	

7.8.6.5.2 EUI-64 designator format

If the designator type is 2h (i.e., EUI-64 based identifier) and the DESIGNATOR LENGTH field is set to 08h, the DESIGNATOR field has the format shown in table 596. The CODE SET field shall be set to 1h (i.e., binary).

Table 596 — EUI-64 DESIGNATOR field format

Bit / Byte	7	6	5	4	3	2	1	0
0	(MSB)							
1	IEEE COMPANY_ID							
2								(LSB)
3	(MSB)							
•••	VENDOR SPECIFIC EXTENSION IDENTIFIER							
7								(LSB)

The IEEE COMPANY_ID field contains a 24-bit canonical form OUI assigned by the IEEE.

The VENDOR SPECIFIC EXTENSION IDENTIFIER field contains a 40-bit numeric value that is assigned by the organization associated with the IEEE company_id as required by the IEEE definition of EUI-64 in a way that makes the entire DESIGNATOR field (see table 596) unique.

7.8.6.5.3 EUI-64 based 12-byte designator format

If the designator type is 2h (i.e., EUI-64 based identifier) and the DESIGNATOR LENGTH field is set to 0Ch, the DESIGNATOR field has the format shown in table 597. The CODE SET field shall be set to 1h (i.e., binary).

Table 597 — EUI-64 based 12-byte DESIGNATOR field format

Bit / Byte	7	6	5	4	3	2	1	0
0	(MSB)							
1	IEEE COMPANY_ID							
2								(LSB)
3	(MSB)							
•••	VENDOR SPECIFIC EXTENSION IDENTIFIER							
7								(LSB)
8	(MSB)							
•••	DIRECTORY ID							
11								(LSB)

The IEEE COMPANY_ID field and VENDOR SPECIFIC EXTENSION IDENTIFIER field are defined in 7.8.6.5.2.

The DIRECTORY ID field contains a directory identifier, as specified by ISO/IEC 13213:1994.

NOTE 61 - The EUI-64 based 12-byte format is used to return IEEE 1394 target port identifiers (see SBP-3).

7.8.6.5.4 EUI-64 based 16-byte designator format

If the designator type is 2h (i.e., EUI-64 based identifier) and the DESIGNATOR LENGTH field is set to 10h, the DESIGNATOR field has the format shown in table 598. The CODE SET field shall be set to 1h (i.e., binary).

Table 598 — EUI-64 based 16-byte DESIGNATOR field format

Bit / Byte	7	6	5	4	3	2	1	0
0	(MSB)							
•••	IDENTIFIER EXTENSION							
7								(LSB)
8	(MSB)							
•••	IEEE COMPANY_ID							
10								(LSB)
11	(MSB)							
•••	VENDOR SPECIFIC EXTENSION IDENTIFIER							
15								(LSB)

The IDENTIFIER EXTENSION field contains a 64-bit numeric value.

The IEEE COMPANY_ID field and VENDOR SPECIFIC EXTENSION IDENTIFIER field are defined in 7.8.6.5.2.

NOTE 62 - The EUI-64 based 16-byte format is used to return SCSI over RDMA target port identifiers (see SRP).

7.8.6.6 NAA designator format

7.8.6.6.1 NAA identifier basic format

If the designator type is 3h (i.e., NAA identifier), the DESIGNATOR field has the format shown in table 599. This format is compatible with the Name_Identifier format defined in FC-FS-3.

Table 599 — NAA DESIGNATOR field format

Bit / Byte	7	6	5	4	3	2	1	0
0	NAA							
1	NAA specific data							
•••								
n								

The Network Address Authority (NAA) field (see table 600) defines the format of the NAA specific data in the designator.

Table 600 — Network Address Authority (NAA) field

Code	Description	Reference
2h	IEEE Extended	7.8.6.6.2
3h	Locally Assigned	7.8.6.6.3
5h	IEEE Registered	7.8.6.6.4
6h	IEEE Registered Extended	7.8.6.6.5
All others	Reserved	

7.8.6.6.2 NAA IEEE Extended designator format

If NAA is 2h (i.e., IEEE Extended), the eight-byte fixed length DESIGNATOR field shall have the format shown in table 601. The CODE SET field shall be set to 1h (i.e., binary) and the DESIGNATOR LENGTH field shall be set to 08h.

Table 601 — NAA IEEE Extended DESIGNATOR field format

Bit / Byte	7	6	5	4	3	2	1	0
0	NAA (2h)				(MSB)			
1	VENDOR SPECIFIC IDENTIFIER A							(LSB)
2	(MSB)							
3	IEEE COMPANY_ID							
4								(LSB)
5	(MSB)							
6	VENDOR SPECIFIC IDENTIFIER B							
7								(LSB)

The IEEE COMPANY_ID field contains a 24-bit canonical form OUI assigned by the IEEE.

The VENDOR SPECIFIC IDENTIFIER A field contains a 12-bit numeric value that is assigned by the organization associated with the IEEE company_id in a way that combines with the VENDOR SPECIFIC IDENTIFIER B field to make the entire DESIGNATOR field (see table 601) unique.

The VENDOR SPECIFIC IDENTIFIER B field contains a 24-bit numeric value that is assigned by the organization associated with the IEEE company_id in a way that combines with the VENDOR SPECIFIC IDENTIFIER A field to make the entire DESIGNATOR field (see table 601) unique.

NOTE 63 - The EUI-64 format includes a 40-bit vendor specific identifier. The IEEE Extended format includes 36 bits of vendor specific identifier in two fields.

7.8.6.6.3 NAA Locally Assigned designator format

If NAA is 3h (i.e., Locally Assigned), the eight-byte fixed length DESIGNATOR field shall have the format shown in table 602. The CODE SET field shall be set to 1h (i.e., binary) and the DESIGNATOR LENGTH field shall be set to 08h.

Table 602 — NAA Locally Assigned DESIGNATOR field format

Bit / Byte	7	6	5	4	3	2	1	0
0	NAA (3h)							
1	LOCALLY ADMINISTERED VALUE							
•••								
7								

The LOCALLY ADMINISTERED VALUE field contains a 60-bit value that is assigned by an administrator to be unique within the set of SCSI domains that are accessible by a common instance of an administrative tool or tools.

7.8.6.6.4 NAA IEEE Registered designator format

If NAA is 5h (i.e., IEEE Registered), the eight-byte fixed length DESIGNATOR field shall have the format shown in table 603. The CODE SET field shall be set to 1h (i.e., binary) and the DESIGNATOR LENGTH field shall be set to 08h.

Table 603 — NAA IEEE Registered DESIGNATOR field format

Bit / Byte	7	6	5	4	3	2	1	0
0	NAA (5h)				(MSB)			
1	IEEE COMPANY_ID							
2								
3				(LSB)	(MSB)			
4	VENDOR SPECIFIC IDENTIFIER							
•••								
7								(LSB)

The IEEE COMPANY_ID field contains a 24-bit canonical form OUI assigned by the IEEE.

The VENDOR SPECIFIC IDENTIFIER field contains a 36-bit numeric value that is assigned by the organization associated with the IEEE company_id in a way that makes the entire DESIGNATOR field (see table 603) unique.

NOTE 64 - The EUI-64 identifier includes a 40-bit vendor specific identifier. The IEEE Registered format includes a 36-bit vendor specific identifier.

7.8.6.6.5 NAA IEEE Registered Extended designator format

If NAA is 6h (i.e., IEEE Registered Extended), the 16-byte fixed length DESIGNATOR field shall have the format shown in table 604. The CODE SET field shall be set to 1h (i.e., binary) and the DESIGNATOR LENGTH field shall be set to 10h.

Table 604 — NAA IEEE Registered Extended DESIGNATOR field format

Bit / Byte	7	6	5	4	3	2	1	0
0	NAA (6h)				(MSB)			
1	IEEE COMPANY_ID							
2								
3				(LSB)	(MSB)			
4	VENDOR SPECIFIC IDENTIFIER							
•••								
7								(LSB)
8	(MSB)							
•••	VENDOR SPECIFIC IDENTIFIER EXTENSION							
15								(LSB)

The IEEE COMPANY_ID field contains a 24-bit canonical form OUI assigned by the IEEE.

The VENDOR SPECIFIC IDENTIFIER field contains a 36-bit numeric value that is assigned by the organization associated with the IEEE company_id in a way that combines with the VENDOR SPECIFIC IDENTIFIER EXTENSION field to make the entire DESIGNATOR field (see table 604) unique.

NOTE 65 - The EUI-64 format includes a 40-bit vendor specific identifier. The IEEE Registered Extended format includes a 36-bit vendor specific identifier.

The VENDOR SPECIFIC IDENTIFIER EXTENSION field contains a 64-bit numeric value that is assigned by the organization associated with the IEEE company_id in a way that combines with the VENDOR SPECIFIC IDENTIFIER field to make the entire DESIGNATOR field (see table 604) unique.

7.8.6.7 Relative target port designator format

If the designator type is 4h (i.e., relative target port identifier) and the ASSOCIATION field is set to 01b (i.e., target port), then the DESIGNATOR field shall have the format shown in table 605. The CODE SET field shall be set to 1h (i.e., binary) and the DESIGNATOR LENGTH field shall be set to 04h. If the ASSOCIATION field does not contain 01b, use of this designator type is reserved.

Table 605 — Relative target port DESIGNATOR field format

Bit / Byte	7	6	5	4	3	2	1	0
0	Reserved							
1								
2	(MSB)							
3	RELATIVE TARGET PORT IDENTIFIER							(LSB)

The RELATIVE TARGET PORT IDENTIFIER field (see 4.3.4) contains the relative port identifier of the target port on which the INQUIRY command was received.

7.8.6.8 Target port group designator format

If the designator type is 5h (i.e., target port group) and the ASSOCIATION value is 01b (i.e., target port), the four-byte fixed length DESIGNATOR field shall have the format shown in table 606. The CODE SET field shall be set to 1h (i.e., binary) and the DESIGNATOR LENGTH field shall be set to 04h. If the ASSOCIATION field does not contain 01b, use of this designator type is reserved.

Table 606 — Target port group DESIGNATOR field format

Bit / Byte	7	6	5	4	3	2	1	0
0	Reserved							
1								
2	(MSB)	TARGET PORT GROUP						
3								(LSB)

The TARGET PORT GROUP field indicates the primary target port group to which the target port is a member (see 5.15).

7.8.6.9 Logical unit group designator format

A logical unit group is a group of logical units that share the same primary target port group (see 5.15) definitions. The primary target port groups maintain the same primary target port group asymmetric access states for all logical units in the same logical unit group. A logical unit shall be in no more than one logical unit group.

If the designator type is 6h (i.e., logical unit group) and the ASSOCIATION value is 00b (i.e., logical unit), the four-byte fixed length DESIGNATOR field shall have the format shown in table 607. The CODE SET field shall be set to 1h (i.e., binary) and the DESIGNATOR LENGTH field shall be set to 04h. If the ASSOCIATION field does not contain 00b, use of this designator type is reserved.

Table 607 — Logical unit group DESIGNATOR field format

Bit / Byte	7	6	5	4	3	2	1	0
0	Reserved							
1								
2	(MSB)	LOGICAL UNIT GROUP						
3								(LSB)

The LOGICAL UNIT GROUP field indicates the logical unit group to which the logical unit is a member.

7.8.6.10 MD5 logical unit designator format

If the designator type is 7h (i.e., MD5 logical unit identifier) and the ASSOCIATION value is 00b (i.e., logical unit), the DESIGNATOR field has the format shown in table 608. The CODE SET field shall be set to 1h (i.e., binary). The MD5 logical unit designator shall not be used if a logical unit provides unique identification using designator types 2h (i.e., EUI-64 based identifier), 3h (i.e., NAA identifier), or 8h (i.e., SCSI name string). A bridge device may return a MD5 logical unit designator type for that logical unit that does not support the Device Identification VPD page (see 7.8.6).