

INTERNATIONAL STANDARD

ISO/IEC
14776-452

First edition
2005-08

Information technology – Small computer system interface (SCSI) – Part 452: Primary Commands-2 (SPC-2)



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

INTERNATIONAL STANDARD

ISO/IEC 14776-452

First edition
2005-08

Information technology – Small computer system interface (SCSI) – Part 452: Primary Commands-2 (SPC-2)

© ISO/IEC 2005

All rights reserved. Unless otherwise specified, no part of this publication may be reproduced or utilized in any form or by any means, electronic or mechanical, including photocopying and microfilm, without permission in writing from the publisher.

ISO/IEC Copyright Office • Case postale 56 • CH-1211 Genève 20 • Switzerland



PRICE CODE **XF**

For price, see current catalogue

Contents

	Page
Foreword	13
Introduction	14
1 Scope	17
2 Normative references.....	17
2.1 General.....	17
2.2 Approved references	17
2.3 References under development	17
3 Definitions, symbols, abbreviations, and conventions	18
3.1 Definitions.....	18
3.2 Acronyms.....	23
3.3 Keywords.....	24
3.4 Conventions.....	25
3.5 Notation for procedures and functions.....	26
4 General concepts	27
4.1 Introduction.....	27
4.2 The request-response model.....	27
4.3 The Command Descriptor Block (CDB).....	27
4.3.1 CDB usage and structure	27
4.3.2 The fixed length CDB formats	28
4.3.3 The variable length CDB formats	30
4.3.4 Common CDB fields	31
4.3.4.1 Operation code	31
4.3.4.2 Service action	31
4.3.4.3 Logical block address	32
4.3.4.4 Transfer length	32
4.3.4.5 Parameter list length.....	32
4.3.4.6 Allocation length	32
4.3.4.7 Control	32
5 Model common to all device types	33
5.1 Introduction to the model common to all device types.....	33
5.2 Commands implemented by all SCSI device servers.....	33
5.2.1 Summary of commands implemented by all SCSI device servers	33
5.2.2 Using the INQUIRY command.....	33
5.2.3 Using the REQUEST SENSE command	33
5.2.4 Using the TEST UNIT READY command.....	33
5.3 Parameter rounding.....	33
5.4 Self-test Operations	34
5.4.1 Default self-test.....	34
5.4.2 The short and extended self-tests	34
5.4.3 Self-test modes.....	34
5.4.3.1 Foreground mode	34
5.4.3.2 Background mode	35
5.4.3.3 Elements common to foreground and background self-test modes	36
5.5 Reservations.....	36
5.5.1 Reservations overview	36
5.5.2 The Reserve/Release management method	39
5.5.3 The Persistent Reservations management method	40
5.5.3.1 Overview of the Persistent Reservations management method.....	40
5.5.3.2 Preserving persistent reservations	40
5.5.3.3 Finding persistent reservations and reservation keys	41
5.5.3.3.1 Summary of commands for finding persistent reservations and reservation keys	41

5.5.3.3.2 Reporting reservation keys	41
5.5.3.3.3 Reporting persistent reservations	42
5.5.3.4 Registering	42
5.5.3.5 Creating a persistent reservation when there is no persistent reservation	43
5.5.3.6 Removing registrations and persistent reservations	44
5.5.3.6.1 Overview of removing registrations and persistent reservations	44
5.5.3.6.2 Releasing a persistent reservation	44
5.5.3.6.3 Preempting an existing persistent reservation with the PREEMPT service action	45
5.5.3.6.3.1 Overview of preempting an existing persistent reservation with the PREEMPT service action	45
5.5.3.6.3.2 Failed persistent reservation preempt	46
5.5.3.6.3.3 Preempting reservations	46
5.5.3.6.3.4 Removing registrations	47
5.5.3.6.4 Preempting an existing persistent reservation with the PREEMPT AND ABORT service action	47
5.5.3.6.5 Clearing a persistent reservation	48
5.6 Multiple port and multiple initiator behavior	49
5.7 Removable medium devices with an attached medium changer	49
6 Model for processor devices	50
7 Commands for all device types	52
7.1 Summary of commands for all device types	52
7.2 EXTENDED COPY command	53
7.2.1 EXTENDED COPY command introduction	53
7.2.2 Errors detected before starting processing of the segment descriptors	55
7.2.3 Errors detected during processing of segment descriptors	56
7.2.4 Abort task management functions	57
7.2.5 Descriptor type codes	58
7.2.6 Target descriptors	59
7.2.6.1 Target descriptors introduction	59
7.2.6.2 Fibre Channel World Wide Name target descriptor format	61
7.2.6.3 Fibre Channel N_Port target descriptor format	62
7.2.6.4 Fibre Channel N_Port with World Wide Name checking target descriptor format	63
7.2.6.5 Parallel Interface T_L target descriptor format	64
7.2.6.6 Identification descriptor target descriptor format	65
7.2.6.7 Device type specific target descriptor parameters for block device types	66
7.2.6.8 Device type specific target descriptor parameters for sequential-access device types	66
7.2.6.9 Device type specific target descriptor parameters for processor device types	67
7.2.7 Segment Descriptors	68
7.2.7.1 Segment descriptors introduction	68
7.2.7.2 Segment descriptor processing	68
7.2.7.3 Block device to stream device operations	72
7.2.7.4 Stream device to block device operations	73
7.2.7.5 Block device to block device operations	74
7.2.7.6 Stream device to stream device operations	75
7.2.7.7 Inline data to stream device operation	77
7.2.7.8 Embedded data to stream device operation	78
7.2.7.9 Stream device to discard operation	79
7.2.7.10 Verify device operation	80
7.2.7.11 Block device with offset to stream device operation	81
7.2.7.12 Stream device to block device with offset operation	82
7.2.7.13 Block device with offset to block device with offset operation	83
7.2.7.14 Write filemarks operation	84
7.2.7.15 Space operation	85
7.2.7.16 Locate operation	86
7.2.7.17 Tape device image copy operation	87
7.2.7.18 Register key operation	88
7.3 INQUIRY command	89
7.3.1 INQUIRY command introduction	89
7.3.2 Standard INQUIRY data	91
7.3.3 SCSI Parallel Interface specific INQUIRY data	98

7.3.4 Vital product data.....	99
7.3.5 Command support data	100
7.4 LOG SELECT command	102
7.5 LOG SENSE command	104
7.6 MODE SELECT(6) command.....	106
7.7 MODE SELECT(10) command.....	108
7.8 MODE SENSE(6) command	108
7.8.1 MODE SENSE(6) command introduction	108
7.8.2 Current values	110
7.8.3 Changeable values.....	110
7.8.4 Default values.....	110
7.8.5 Saved values	110
7.8.6 Initial responses.....	110
7.9 MODE SENSE(10) command	111
7.10 PERSISTENT RESERVE IN command	112
7.10.1 PERSISTENT RESERVE IN command introduction	112
7.10.2 PERSISTENT RESERVE IN service actions	112
7.10.2.1 Summary of PERSISTENT RESERVE IN service actions	112
7.10.2.2 Read Keys.....	112
7.10.2.3 Read Reservations	113
7.10.3 PERSISTENT RESERVE IN parameter data for READ KEYS	113
7.10.4 PERSISTENT RESERVE IN parameter data for READ RESERVATION.....	114
7.10.4.1 Format of PERSISTENT RESERVE IN parameter data for READ RESERVATION	114
7.10.4.2 Persistent reservations Scope	115
7.10.4.2.1 Summary of persistent reservations Scope	115
7.10.4.2.2 Logical unit scope.....	115
7.10.4.2.3 Element scope	115
7.10.4.3 Persistent Reservations Type.....	116
7.11 PERSISTENT RESERVE OUT command	117
7.11.1 PERSISTENT RESERVE OUT command introduction.....	117
7.11.2 PERSISTENT RESERVE OUT Service Actions	118
7.11.3 PERSISTENT RESERVE OUT parameter list	119
7.12 PREVENT ALLOW MEDIUM REMOVAL command	121
7.13 READ BUFFER command	122
7.13.1 READ BUFFER command introduction.....	122
7.13.2 Combined header and data mode (0000b).....	123
7.13.3 Vendor specific mode (0001b).....	123
7.13.4 Data mode (0010b).....	123
7.13.5 Descriptor mode (0011b).....	124
7.13.6 Read Data from echo buffer (1010b).....	124
7.13.7 Echo buffer descriptor mode (1011b)	125
7.14 RECEIVE COPY RESULTS command	126
7.14.1 RECEIVE COPY RESULTS command introduction.....	126
7.14.2 COPY STATUS service action	127
7.14.3 RECEIVE DATA service action	129
7.14.4 OPERATING PARAMETERS service action.....	130
7.14.5 FAILED SEGMENT DETAILS service action	132
7.15 RECEIVE DIAGNOSTIC RESULTS command	133
7.16 RELEASE(10) command.....	134
7.16.1 RELEASE(10) command introduction	134
7.16.2 Logical unit release.....	134
7.16.3 Third-party release	134
7.17 RELEASE(6) command.....	135
7.18 REPORT DEVICE IDENTIFIER command	136
7.19 REPORT LUNS command	138
7.20 REQUEST SENSE command	140
7.20.1 REQUEST SENSE command introduction.....	140
7.20.2 Sense data format	141
7.20.3 Sense-key specific.....	143
7.20.4 Current errors	144

7.20.5 Deferred errors	144
7.20.6 Sense key and sense code definitions	146
7.21 RESERVE(10) command	158
7.21.1 RESERVE(10) command introduction.....	158
7.21.2 Logical unit reservation.....	158
7.21.3 Third-party reservation	159
7.21.4 Superseding reservations.....	160
7.22 RESERVE(6) command	160
7.23 SEND DIAGNOSTIC command	161
7.24 SET DEVICE IDENTIFIER command	163
7.25 TEST UNIT READY command.....	165
7.26 WRITE BUFFER command.....	166
7.26.1 WRITE BUFFER command introduction	166
7.26.2 Combined header and data mode (0000b).....	167
7.26.3 Vendor specific mode (0001b).....	167
7.26.4 Data mode (0010b).....	167
7.26.5 Download microcode mode (0100b).....	167
7.26.6 Download microcode and save mode (0101b)	167
7.26.7 Download microcode with offsets (0110b).....	168
7.26.8 Download microcode with offsets and save mode (0111b)	168
7.26.9 Write data to echo buffer (1010b).....	169
8 Parameters for all device types.....	170
8.1 Diagnostic parameters.....	170
8.1.1 Diagnostic page format and page codes for all device types	170
8.1.2 Supported diagnostic pages	172
8.2 Log parameters	173
8.2.1 Log page structure and page codes for all device types	173
8.2.2 Application client page.....	176
8.2.3 Buffer over-run/under-run page	177
8.2.4 Error counter pages.....	179
8.2.5 Last n deferred errors or asynchronous events page.....	179
8.2.6 Last n error events page.....	179
8.2.7 Non-medium error page	180
8.2.8 Self-test results page.....	180
8.2.9 Start-stop cycle counter page.....	183
8.2.10 Supported log pages	185
8.2.11 Temperature page	185
8.3 Mode parameters	187
8.3.1 Mode parameters overview	187
8.3.2 Mode parameter list format.....	187
8.3.3 Mode parameter header formats	187
8.3.4 Mode parameter block descriptor formats	188
8.3.4.1 General block descriptor format	188
8.3.4.2 Direct-access device block descriptor format for LONGLBA=0	189
8.3.4.3 Long LBA block descriptor format	190
8.3.5 Mode page format and page codes.....	191
8.3.6 Control mode page	192
8.3.7 Disconnect-reconnect page.....	196
8.3.8 Informational exceptions control page.....	198
8.3.9 Power condition page	200
8.3.10 Protocol specific LUN page	202
8.3.11 Protocol specific port page	203
8.4 Vital product data parameters	204
8.4.1 Vital product data parameters overview and page codes.....	204
8.4.2 ASCII implemented operating definition page	204
8.4.3 ASCII information page.....	205
8.4.4 Device identification page.....	205
8.4.5 Supported vital product data pages.....	208
8.4.6 Unit serial number page	209

9 Commands for processor type devices.....	210
9.1 Summary of commands for processor type devices.....	210
9.2 RECEIVE command	211
9.3 SEND command	211
10 Parameters for processor type devices	212
10.1 Diagnostic parameters.....	212
10.2 Log parameters	213
10.3 Vital product data parameters	213
Annex A	
Procedures for logging operations in SCSI	214
A.1 Procedures for logging operations in SCSI introduction	214
A.2 Logging operations terminology	214
A.3 LOG SENSE command.....	215
A.4 LOG SELECT command.....	218
A.5 Exception conditions during logging.....	221
Annex B	
Commands allowed in the presence of various reservations.....	224
B.1 SBC commands	224
B.2 SMC commands.....	228
Annex C	
Numeric order codes.....	230
C.1 Numeric order codes introduction	230
C.2 Additional Sense Codes.....	230
C.3 Operation Codes.....	242
C.4 Log Page Codes	248
C.5 Mode Page Codes	249
C.6 Version Descriptor Values	251
C.7 Variable Length CDB Service Action Codes.....	258
Annex D	
Vendor identification	259
Annex E	
References and general structure of SCSI	266

Tables

	Page
1 Typical CDB for 6-byte commands	28
2 Typical CDB for 10-byte commands	28
3 Typical CDB for 12-byte commands	29
4 Typical CDB for 16-byte commands	29
5 Typical CDB for long LBA 16-byte commands	30
6 Typical variable length CDB	30
7 Typical variable length CDB for long LBA 32-byte commands	31
8 Exception commands for background self-tests	35
9 Self-test mode summary	36
10 SPC commands that are allowed in the presence of various reservations	38
11 PERSISTENT RESERVE OUT service actions that are allowed in the presence of various reservations	39
12 Processor commands that are allowed in the presence of various reservations	51
13 Commands for all device types	52
14 EXTENDED COPY command	53
15 EXTENDED COPY parameter list	54
16 EXTENDED COPY descriptor type codes	58
17 Target descriptor format	59
18 Device type specific parameters in target descriptors	60
19 Fibre Channel World Wide Name target descriptor format	61
20 Fibre Channel N_Port target descriptor format	62
21 Fibre Channel N_Port with World Wide Name checking target descriptor format	63
22 Parallel Interface T_L target descriptor format	64
23 Identification descriptor target descriptor format	65
24 Device type specific target descriptor parameters for block device types	66
25 Device type specific target descriptor parameters for sequential-access device types	66
26 Stream device transfer lengths	67
27 Device type specific target descriptor parameters for processor device types	67
28 Segment descriptor header	68
29 Descriptor Type Code Dependent Copy Manager Processing	69
30 PAD and CAT bit definitions	71
31 Block device to or from stream device segment descriptor	72
32 Block device to block device segment descriptor	74
33 Stream device to stream device segment descriptor	75
34 Inline data to stream device segment descriptor	77
35 Embedded data to stream device segment descriptor	78
36 Stream device to discard segment descriptor	79
37 Verify device operation segment descriptor	80
38 Block device with offset to or from stream device segment descriptor	81
39 Block device with offset to block device with offset segment descriptor	83
40 Write filemarks operation segment descriptor	84
41 Space operation segment descriptor	85
42 Locate operation segment descriptor	86
43 Tape device image copy segment descriptor	87
44 Register key segment descriptor	88
45 INQUIRY command	89
46 Standard INQUIRY data format	91
47 Peripheral qualifier	92
48 Peripheral device type	92
49 Version	93
50 Relationship of BQUE and CMDQUE bits	94
51 Version descriptor values	95
52 SPI-specific standard INQUIRY bits	98
53 Maximum logical device configuration table	98
54 CLOCKING field	99
55 Command support data format	100
56 SUPPORT values and meanings	100
57 LOG SELECT command	102
58 Page control field	102

59 LOG SENSE command	104
60 MODE SELECT(6) command	106
61 MODE SELECT(10) command	108
62 MODE SENSE(6) command	108
63 Page control field	109
64 Mode page code usage for all devices	109
65 MODE SENSE(10) command	111
66 PERSISTENT RESERVE IN command	112
67 PERSISTENT RESERVE IN service action codes	112
68 PERSISTENT RESERVE IN parameter data for READ KEYS	113
69 PERSISTENT RESERVE IN parameter data for READ RESERVATION	114
70 PERSISTENT RESERVE IN reservation descriptor	114
71 Persistent reservation scope codes	115
72 Persistent reservation type codes	116
73 PERSISTENT RESERVE OUT command	117
74 PERSISTENT RESERVE OUT service action codes	118
75 PERSISTENT RESERVE OUT parameter list	119
76 PERSISTENT RESERVE OUT service actions and valid parameters	120
77 PREVENT ALLOW MEDIUM REMOVAL command	121
78 PREVENT ALLOW MEDIUM REMOVAL PREVENT field	121
79 READ BUFFER command	122
80 READ BUFFER MODE field	122
81 READ BUFFER header	123
82 READ BUFFER descriptor	124
83 Buffer offset boundary	124
84 Echo buffer descriptor	125
85 RECEIVE COPY RESULTS command	126
86 RECEIVE COPY RESULTS service action codes	126
87 Parameter data for the COPY STATUS service action	127
88 COPY STATUS STATUS values	128
89 COPY STATUS TRANSFER COUNT UNITS values	128
90 Parameter data for the RECEIVE DATA service action	129
91 Parameter data for the OPERATING PARAMETERS service action	130
92 Parameter data for the FAILED SEGMENT DETAILS service action	132
93 RECEIVE DIAGNOSTIC RESULTS command	133
94 RELEASE(10) command	134
95 RELEASE(10) parameter list	135
96 RELEASE(6) command	135
97 REPORT DEVICE IDENTIFIER command	136
98 REPORT DEVICE IDENTIFIER parameter list	137
99 REPORT LUNS command	138
100 REPORT LUNS parameter list format	139
101 REQUEST SENSE command	140
102 Response codes 70h and 71h sense data format	141
103 Field pointer bytes	143
104 Actual retry count bytes	143
105 Progress indication bytes	144
106 Segment pointer bytes	144
107 Sense key descriptions	146
108 ASC and ASCQ assignments	147
109 RESERVE(10) command	158
110 RESERVE(10) ID only parameter list	159
111 RESERVE(6) command	160
112 SEND DIAGNOSTIC command	161
113 SELF-TEST CODE field values	161
114 SET DEVICE IDENTIFIER command	163
115 SET DEVICE IDENTIFIER parameter list	164
116 TEST UNIT READY command	165
117 Preferred TEST UNIT READY responses	165
118 WRITE BUFFER command	166

119 WRITE BUFFER MODE field	166
120 Diagnostic page format	170
121 Diagnostic page codes	171
122 Supported diagnostic pages	172
123 Log page format	173
124 Log parameter	173
125 Threshold met criteria	174
126 Log page codes	176
127 Application client page	176
128 General usage application client parameter data	177
129 Parameter control bits for general usage parameters (0000h through 0FFFh)	177
130 Parameter code field for buffer over-run/under-run counters	178
131 Count basis definition	178
132 Cause field definition	178
133 Parameter codes for error counter pages	179
134 Non-medium error event parameter codes	180
135 Self-test results page	180
136 Self-test results log parameter format	181
137 Parameter control bits for self-test results log parameters	181
138 Self-test results values	182
139 Start-stop cycle counter page	183
140 Parameter control bits for date of manufacture parameter (0001h)	184
141 Parameter control bits for accounting date parameter (0002h)	184
142 Parameter control bits for start-stop cycle counter parameters (0003h and 0004h)	184
143 Supported log pages	185
144 Temperature page	185
145 Parameter control bits for temperature parameters (0000h and 0001h)	186
146 Mode parameter list	187
147 Mode parameter header(6)	187
148 Mode parameter header(10)	187
149 General mode parameter block descriptor	188
150 Direct-access device mode parameter block descriptor	189
151 Long LBA mode parameter block descriptor	190
152 Mode page format	191
153 Mode page codes	192
154 Control mode page	192
155 Task set type	193
156 Queue algorithm modifier	193
157 Queue error management (QERR) field	194
158 AUTOLOAD MODE field	195
159 Disconnect-reconnect page	196
160 Data transfer disconnect control	197
161 Informational exceptions control page	198
162 Method of reporting informational exceptions (MRIE) field	199
163 Power condition page	201
164 Protocol specific LUN page	202
165 PROTOCOL IDENTIFIER values	203
166 Protocol specific port page	203
167 Vital product data page codes	204
168 ASCII implemented operating definition	204
169 ASCII information page	205
170 Device identification page	206
171 Identification descriptor	206
172 Code set	206
173 Association	207
174 Identifier type	207
175 Relative port identifier values	207
176 Device identification page example	208
177 Supported vital product data pages	208
178 Unit serial number page	209

179 Commands for processor devices	210
180 RECEIVE command	211
181 SEND command	211
182 SEND command – AER data format.....	212
183 Processor diagnostic page codes	212
184 Processor log page codes	213
185 Processor vital product data page codes	213
A.1 LOG SENSE Command CDB fields	215
A.2 LOG SENSE returned parameter values.....	216
A.3 LOG SENSE save options.....	217
A.4 LOG SELECT CDB fields	218
A.5 LOG SELECT save options.....	219
A.6 LOG SELECT controller parameter values	220
A.7 Log Parameter Control Byte saving definitions	221
A.9 Logging exception conditions	222
A.8 Log Parameter Control Byte updating definitions	222
B.1 SBC direct access commands that are allowed in the presence of various reservations.....	225
B.2 SBC optical memory commands that are allowed in the presence of various reservations	226
B.3 SBC write-once commands that are allowed in the presence of various reservations	227
B.4 SMC commands that are allowed in the presence of various reservations	228
C.1 ASC and ASCQ assignments.....	230
C.2 Operation Codes	242
C.3 Log Page Codes.....	248
C.4 Mode Page Codes.....	249
C.5 Version descriptor assignments	251
C.6 Standard code value guidelines	255
C.7 Revision code value guidelines	257
C.8 Variable Length CDB Service Action Code Ranges.....	258
C.9 Variable Length CDB Service Action Codes Used by All Device Types	258
D.1 Vendor identification list	259

Figures

	Page
1 SCSI document relationships.....	14
2 Device server interpretation of PREEMPT service action.....	46
3 Power conditions flowchart	202

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

FOREWORD

- 1) ISO (International Organization for Standardization) and IEC (International Electrotechnical Commission) form the specialized system for worldwide standardization. National bodies that are members of ISO or IEC participate in the development of International Standards through technical committees established by the respective organization to deal with particular fields of technical activity. ISO and IEC technical committees collaborate in fields of mutual interest. Other international organizations, governmental and non-governmental, in liaison with ISO and IEC, also take part in the work.
- 2) In the field of information technology, ISO and IEC have established a joint technical committee, ISO/IEC JTC 1. Draft International Standards adopted by the joint technical committee are circulated to national bodies for voting. Publication as an International Standard requires approval by at least 75 % of the national bodies casting a vote.
- 3) All users should ensure that they have the latest edition of this publication.
- 4) No liability shall attach to IEC or ISO or its directors, employees, servants or agents including individual experts and members of their technical committees and IEC or ISO National Committees for any personal injury, property damage or other damage of any nature whatsoever, whether direct or indirect, or for costs (including legal fees) and expenses arising out of the publication of, use of, or reliance upon this ISO/IEC publication or any other IEC or IEC/ISO publications.
- 5) Attention is drawn to the normative references cited in this publication. Use of the referenced publications is indispensable for the correct application of this publication.
- 6) Attention is drawn to the possibility that some of the elements of this ISO/IEC Publication may be the subject of patent rights. ISO/IEC shall not be held responsible for identifying any or all such patent rights.

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

Introduction

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

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

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

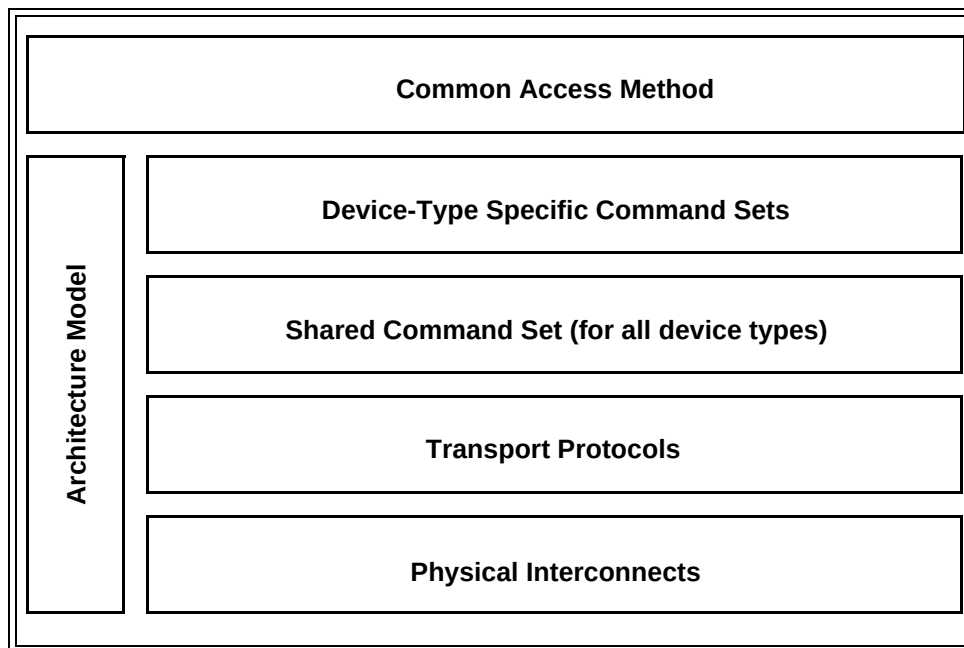


Figure 1 — SCSI document relationships

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

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

This International Standard is divided into ten clauses.

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

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

1 Scope

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

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

2 Normative references

2.1 General

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

2.2 Approved references

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

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

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

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

2.3 References under development

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

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

1 [ANSI INCITS 270:1996]

2 [ANSI INCITS 314:1998]

3 Definitions, symbols, abbreviations and conventions

3.1 Definitions

3.1.1

active condition:

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

3.1.2

additional sense code:

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

3.1.3

application client:

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

3.1.4

attached medium changer:

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

3.1.5

asynchronous event reporting (AER):

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

3.1.6

auto contingent allegiance (ACA):

One of the conditions of a task set following the return of a CHECK CONDITION status. A detailed definition of ACA may be found in SAM-2.

3.1.7

autosense data:

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

3.1.8

blocked task:

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

3.1.9

byte:

Indicates an 8-bit construct.

3.1.10

command:

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

3.1.11

command descriptor block (CDB):

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

3.1.12**command standard:**

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

3.1.13**contingent allegiance (CA):**

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

3.1.14**copy manager:**

The device server that receives an EXTENDED COPY command and performs the operation requested.

3.1.15**data-in buffer:**

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

3.1.16**data-out buffer:**

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

3.1.17**data packet:**

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

3.1.18**device server:**

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

3.1.19**device service request:**

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

3.1.20**device service response:**

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

3.1.21**device type:**

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

3.1.22**element:**

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

3.1.23**enabled task state:**

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

3.1.24

field:

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

3.1.25

hard reset:

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

3.1.26

host:

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

3.1.27

idle condition:

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

3.1.28

initiator:

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

3.1.29

linked command:

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

3.1.30

logical unit:

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

3.1.31

logical unit identifier:

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

3.1.32

logical unit inventory:

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

3.1.33

logical unit number (LUN):

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

3.1.34

medium:

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

3.1.35**medium auxiliary memory (MAM):**

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

3.1.36**medium changer:**

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

3.1.37**one:**

The logical true condition of a variable.

3.1.38**page:**

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

3.1.39**persist through power loss:**

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

3.1.40**protocol specific:**

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

3.1.41**protocol standard:**

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

3.1.42**resource:**

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

3.1.43**registered:**

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

3.1.44**registrant:**

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

3.1.45**SCSI device:**

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

3.1.46**SCSI domain:**

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

3.1.47

sense data:

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

3.1.48

sense key:

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

3.1.49

service action:

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

3.1.50

service delivery subsystem:

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

3.1.51

standby condition:

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

3.1.52

status:

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

3.1.53

system:

One or more SCSI domains operating as a single configuration.

3.1.54

target:

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

3.1.55

task:

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

3.1.56

task set:

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

3.1.57

third-party:

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

3.1.58**unit attention condition:**

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

3.1.59**vendor specific (VS):**

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

3.1.60**zero:**

The logical false condition of a variable.

3.2 Acronyms

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

3.3 Keywords

3.3.1

expected:

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

3.3.2

ignored:

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

3.3.3

invalid:

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

3.3.4

mandatory:

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

3.3.5

may:

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

3.3.6

may not:

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

3.3.7

obsolete:

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

3.3.8

optional:

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

3.3.9

reserved:

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

3.3.10

restricted:

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

3.3.11

shall:

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

3.3.12**should:**

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

3.3.13**x or xx:**

The value of the bit or field is not relevant.

3.4 Conventions

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

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

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

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

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

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

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

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

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

3.5 Notation for procedures and functions

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

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

Where:

Result: A single value representing the outcome of the procedure or function.

Procedure Name: A descriptive name for the function to be performed.

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

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

"[...]": Brackets enclosing optional or conditional parameters and arguments.

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

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

Where:

Found = Flag

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

Input Arguments:

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

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

Output Arguments:

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

4 General concepts

4.1 Introduction

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

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

4.2 The request-response model

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

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

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

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

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

4.3 The Command Descriptor Block (CDB)

4.3.1 CDB usage and structure

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

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

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

4.3.2 The fixed length CDB formats

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

Table 1 — Typical CDB for 6-byte commands

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

Table 2 — Typical CDB for 10-byte commands

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

Table 3 — Typical CDB for 12-byte commands

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

Table 4 — Typical CDB for 16-byte commands

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

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

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

4.3.3 The variable length CDB formats

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

Table 6 — Typical variable length CDB

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

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

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

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

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

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

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

4.3.4 Common CDB fields

4.3.4.1 Operation code

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

4.3.4.2 Service action

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

4.3.4.3 Logical block address

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

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

4.3.4.4 Transfer length

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

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

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

Refer to the specific command description for further information.

4.3.4.5 Parameter list length

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

4.3.4.6 Allocation length

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

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

4.3.4.7 Control

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

5 Model common to all device types

5.1 Introduction to the model common to all device types

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

5.2 Commands implemented by all SCSI device servers

5.2.1 Summary of commands implemented by all SCSI device servers

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

5.2.2 Using the INQUIRY command

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

5.2.3 Using the REQUEST SENSE command

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

5.2.4 Using the TEST UNIT READY command

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

5.3 Parameter rounding

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

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

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

5.4 Self-test Operations

5.4.1 Default self-test

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

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

5.4.2 The short and extended self-tests

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

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

The following are examples of segments:

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

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

5.4.3 Self-test modes

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

5.4.3.1 Foreground mode

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

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

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

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

5.4.3.2 Background mode

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

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

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

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

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

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

Table 8 — Exception commands for background self-tests

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

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

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

5.4.3.3 Elements common to foreground and background self-test modes

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

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

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

Table 9 — Self-test mode summary

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

5.5 Reservations

5.5.1 Reservations overview

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

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

The general description of reservations involves two groups of considerations;

- the type of reservation established, and
- the method used to establish, rescind, and manage the reservation.

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

Command	Addressed LU is reserved by another initiator [A]	Addressed LU has this type of persistent reservation held by another initiator [B]				
		From any initiator		From registered initiator (RO all types)	From initiator not registered	
		Write Excl	Excl Access		Write Excl RO	Excl Access – RO
EXTENDED COPY	Conflict	Conflict	Conflict	Allowed	Conflict	Conflict
INQUIRY	Allowed	Allowed	Allowed	Allowed	Allowed	Allowed
LOG SELECT	Conflict	Conflict	Conflict	Allowed	Conflict	Conflict
LOG SENSE	Allowed	Allowed	Allowed	Allowed	Allowed	Allowed
MODE SELECT(6)/ MODE SELECT(10)	Conflict	Conflict	Conflict	Allowed	Conflict	Conflict
MODE SENSE(6)/ MODE SENSE(10)	Conflict	Conflict	Conflict	Allowed	Conflict	Conflict
PERSISTENT RESERVE IN	Conflict	Allowed	Allowed	Allowed	Allowed	Allowed
PERSISTENT RESERVE OUT	see table 11					
PREVENT/ALLOW (Prevent=0)	Allowed	Allowed	Allowed	Allowed	Allowed	Allowed
PREVENT/ALLOW (Prevent<>0)	Conflict	Conflict	Conflict	Allowed	Conflict	Conflict
READ BUFFER	Conflict	Conflict	Conflict	Allowed	Conflict	Conflict
RECEIVE COPY RESULTS	Conflict	Conflict	Conflict	Allowed	Conflict	Conflict
RECEIVE DIAGNOSTICS	Conflict	Conflict	Conflict	Allowed	Conflict	Conflict
RELEASE(6)/RELEASE(10)	Allowed ^a	Conflict	Conflict	Conflict	Conflict	Conflict
REPORT DEVICE IDENTIFIER	Allowed	Allowed	Allowed	Allowed	Allowed	Allowed
REPORT LUNS	Allowed	Allowed	Allowed	Allowed	Allowed	Allowed
REQUEST SENSE	Allowed	Allowed	Allowed	Allowed	Allowed	Allowed
RESERVE(6)/RESERVE(10)	Conflict	Conflict	Conflict	Conflict	Conflict	Conflict
SEND DIAGNOSTICS	Conflict	Conflict	Conflict	Allowed	Conflict	Conflict
SET DEVICE IDENTIFIER	Conflict	Conflict	Conflict	Allowed	Conflict	Conflict
TEST UNIT READY	Conflict	Conflict	Conflict	Allowed	Conflict	Conflict
WRITE BUFFER	Conflict	Conflict	Conflict	Allowed	Conflict	Conflict
Key: LU=Logical Unit, Excl=Exclusive, RO=Registrants Only, <> Not Equal						
^a The reservation is not released (see 7.16 and 7.17).						

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

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

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

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

5.5.2 The Reserve/Release management method

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

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

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

5.5.3 The Persistent Reservations management method

5.5.3.1 Overview of the Persistent Reservations management method

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

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

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

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

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

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

5.5.3.2 Preserving persistent reservations

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

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

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

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

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

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

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

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

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

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

5.5.3.3 Finding persistent reservations and reservation keys

5.5.3.3.1 Summary of commands for finding persistent reservations and reservation keys

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

5.5.3.3.2 Reporting reservation keys

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

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

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

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

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

5.5.3.3 Reporting persistent reservations

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

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

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

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

5.5.3.4 Registering

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

5.5.3.5 Creating a persistent reservation when there is no persistent reservation

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

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

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

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

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

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

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

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

5.5.3.6 Removing registrations and persistent reservations

5.5.3.6.1 Overview of removing registrations and persistent reservations

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

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

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

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

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

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

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

5.5.3.6.2 Releasing a persistent reservation

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

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

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

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

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

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

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

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

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

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

5.5.3.6.3 Preempting an existing persistent reservation with the PREEMPT service action

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

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

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

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

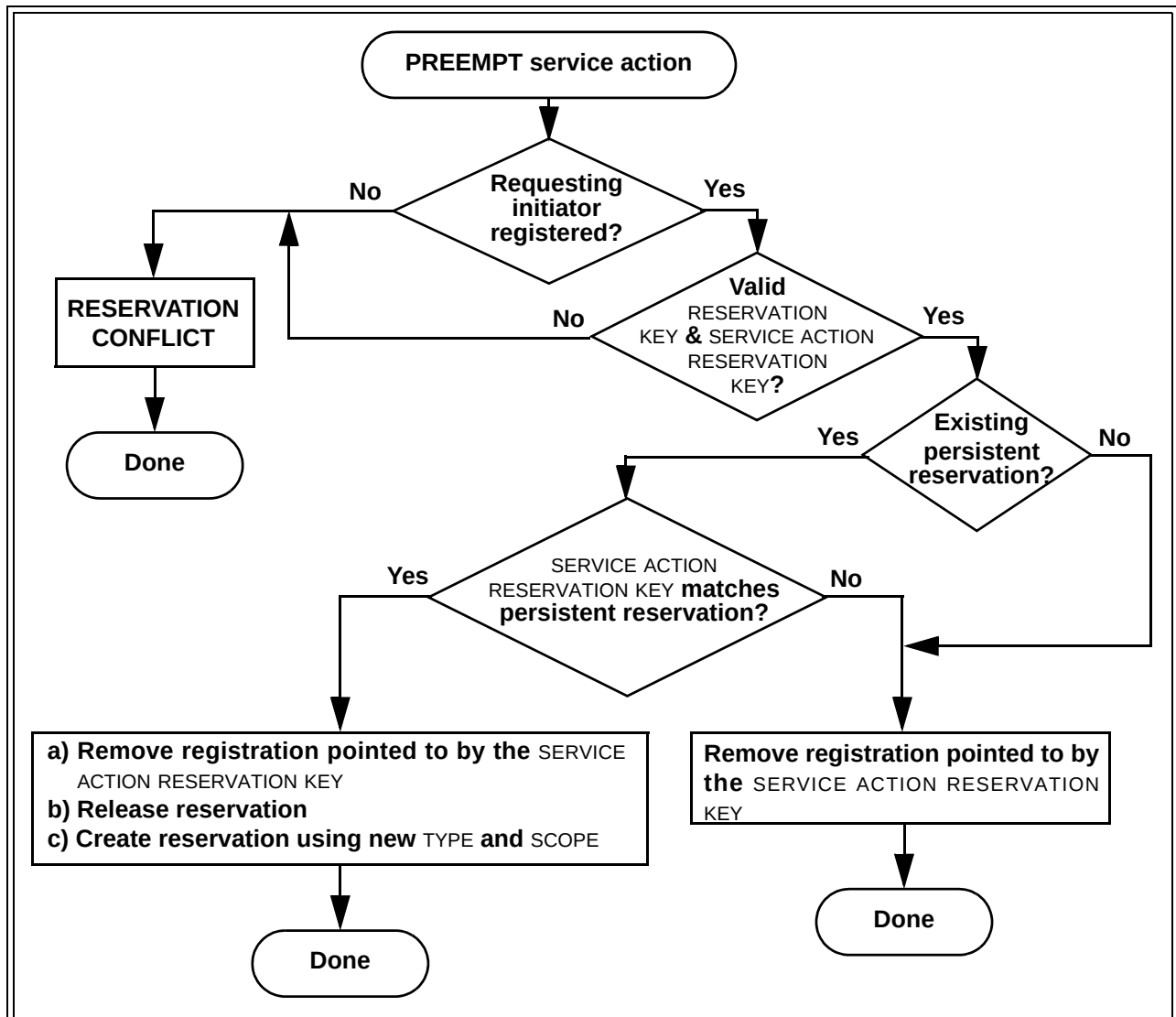


Figure 2 — Device server interpretation of PREEMPT service action

5.5.3.6.3.2 Failed persistent reservation preempt

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

5.5.3.6.3.3 Preempting reservations

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

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

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

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

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

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

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

Completion status shall be returned for each task.

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

5.5.3.6.3.4 Removing registrations

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

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

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

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

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

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

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

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

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

If a terminated task is a command that causes the device server to generate additional commands and data transfers (e.g., EXTENDED COPY), all commands and data transfers generated by the command shall be terminated before the ABORT TASK SET task management function is considered completed. After the ABORT TASK SET function has completed, all new tasks are subject to the persistent reservation restrictions established by the preempting initiator;

- c) The device server shall clear any ACA or CA condition associated with an initiator being preempted and shall clear any tasks with an ACA attribute from that initiator. If the TST field is 000b (see 8.3.6) and ACA or CA conditions exist for initiators other than the initiator being preempted, the PERSISTENT RESERVE OUT command shall be terminated prior to processing with a status of ACA ACTIVE if the NACA bit equals one (see SAM-2) or BUSY if the NACA equals zero. If the TST field contains 001b, then ACA or CA conditions for initiators other than the initiator being preempted shall not prevent the execution of the PERSISTENT RESERVE OUT command; and
- d) For SCSI devices that implement the PREVENT ALLOW MEDIUM REMOVAL command, the device server shall perform an action equivalent to the execution of a PREVENT ALLOW MEDIUM REMOVAL command with the PREVENT field equal to zero for the initiator or initiators being preempted (see 7.12).

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

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

5.5.3.6.5 Clearing a persistent reservation

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

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

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

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

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