

Storage Maintenance (storm) WG  
Internet Draft  
draft-ietf-storm-iscsi-cons-02.txt  
Intended status: Proposed Standard  
Expires: September 2011  
Updates: 3720, 3721, 3980, 4850, 5048

Mallikarjun Chadalapaka  
Microsoft  
  
Julian Satran  
  
Kalman Meth  
IBM

## iSCSI Protocol (Consolidated)

### Status of this Memo

This Internet-Draft is submitted to IETF in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF), its areas, and its working groups. Note that other groups may also distribute working documents as Internet-Drafts.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

The list of current Internet-Drafts can be accessed at <http://www.ietf.org/ietf/lid-abstracts.txt>

The list of Internet-Draft Shadow Directories can be accessed at <http://www.ietf.org/shadow.html>

#### Copyright Notice

Copyright (c) 2011 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<http://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

#### Abstract

This document describes a transport protocol for SCSI that works on top of TCP. The iSCSI protocol aims to be fully compliant with the standardized SCSI Architecture Model (SAM). RFC 3720 defined the original iSCSI protocol. RFC 3721 discusses iSCSI Naming examples and discovery techniques. Subsequently, RFC 3980 added an additional naming format to iSCSI protocol. RFC 4850 followed up by adding a new public extension key to iSCSI. RFC 5048 offered a number of clarifications and a few improvements and corrections to the original iSCSI protocol.

This document consolidates RFCs 3720, 3980, 4850 and 5048 into a single document and makes additional updates to the consolidated specification. This document also updates RFC 3721. The text in this document thus supersedes the text in RFCs 3720, 3721, 3980, 4850 and 5048 whenever there is such a question.

1. Introduction.....	14
2. Definitions and Acronyms.....	15
2.1. Definitions .....	15
2.2. Acronyms .....	21
2.3. Conventions .....	23
2.3.1. Word Rule .....	24
2.3.2. Half-Word Rule .....	25
2.3.3. Byte Rule .....	25
3. UML Conventions.....	26
3.1. UML Conventions Overview .....	26
3.2. Multiplicity Notion .....	26
3.3. Class Diagram Conventions .....	27
3.4. Class Diagram Notation for Associations .....	28
3.5. Class Diagram Notation for Aggregations .....	29
3.6. Class Diagram Notation for Generalizations .....	29
4. Overview.....	31
4.1. SCSI Concepts .....	31
4.2. iSCSI Concepts and Functional Overview .....	32
4.2.1. Layers and Sessions .....	33
4.2.2. Ordering and iSCSI Numbering .....	33
4.2.2.1. Command Numbering and Acknowledging .....	34
4.2.2.2. Response/Status Numbering and Acknowledging .....	38
4.2.2.3. Response Ordering .....	39
4.2.2.3.1. Need for Response Ordering .....	39
4.2.2.3.2. Response Ordering Model Description .....	39
4.2.2.3.3. iSCSI Semantics with the Interface Model .....	40
4.2.2.3.4. Current List of Fenced Response Use Cases .....	40
4.2.2.4. Data Sequencing .....	42
4.2.3. iSCSI Task Management .....	42
4.2.3.1. Task Management Overview .....	42
4.2.3.2. Notion of Affected Tasks .....	43
4.2.3.3. Standard Multi-task Abort Semantics .....	43

	iSCSI (Consolidated)	3/11/11
4.2.3.4.	FastAbort Multi-task Abort Semantics .....	45
4.2.3.5.	Affected Tasks Shared across Standard and FastAbort Sessions .....	47
4.2.3.6.	Rationale behind the FastAbort Semantics .....	48
4.2.4.	iSCSI Login .....	49
4.2.5.	iSCSI Full Feature Phase .....	51
4.2.5.1.	Command Connection Allegiance .....	51
4.2.5.2.	Data Transfer Overview .....	52
4.2.5.3.	Tags and Integrity Checks .....	54
4.2.5.4.	Task Management .....	54
4.2.6.	iSCSI Connection Termination .....	55
4.2.7.	iSCSI Names .....	55
4.2.7.1.	iSCSI Name Properties .....	56
4.2.7.2.	iSCSI Name Encoding .....	58
4.2.7.3.	iSCSI Name Structure .....	59
4.2.7.4.	Type "iqn." (iSCSI Qualified Name) .....	60
4.2.7.5.	Type "eui." (IEEE EUI-64 format) .....	62
4.2.7.6.	Type "naa." - Network Address Authority .....	63
4.2.8.	Persistent State .....	64
4.2.9.	Message Synchronization and Steering .....	64
4.2.9.1.	Sync/Steering and iSCSI PDU Length .....	65
4.3.	iSCSI Session Types .....	66
4.4.	SCSI to iSCSI Concepts Mapping Model .....	66
4.4.1.	iSCSI Architecture Model .....	67
4.4.2.	SCSI Architecture Model .....	70
4.4.3.	Consequences of the Model .....	72
4.4.3.1.	I_T Nexus State .....	74
4.5.	iSCSI UML Model .....	74
4.6.	Request/Response Summary .....	77
4.6.1.	Request/Response Types Carrying SCSI Payload .....	77
4.6.1.1.	SCSI-Command .....	77
4.6.1.2.	SCSI-Response .....	78
4.6.1.3.	Task Management Function Request .....	78
4.6.1.4.	Task Management Function Response .....	79
4.6.1.5.	SCSI Data-out and SCSI Data-in .....	79
4.6.1.6.	Ready To Transfer (R2T) .....	80

4.6.2. Requests/Responses carrying SCSI and iSCSI Payload .....	81
4.6.2.1. Asynchronous Message .....	81
4.6.3. Requests/Responses Carrying iSCSI Only Payload .....	81
4.6.3.1. Text Request and Text Response .....	81
4.6.3.2. Login Request and Login Response .....	82
4.6.3.3. Logout Request and Response .....	83
4.6.3.4. SNACK Request .....	83
4.6.3.5. Reject .....	83
4.6.3.6. NOP-Out Request and NOP-In Response .....	84
5. SCSI Mode Parameters for iSCSI.....	85
6. Login and Full Feature Phase Negotiation.....	86
6.1. Text Format .....	87
6.2. Text Mode Negotiation .....	92
6.2.1. List negotiations .....	96
6.2.2. Simple-value Negotiations .....	97
6.3. Login Phase .....	97
6.3.1. Login Phase Start .....	101
6.3.2. iSCSI Security Negotiation .....	104
6.3.3. Operational Parameter Negotiation During the Login Phase	105
6.3.4. Connection Reinstatement .....	106
6.3.5. Session Reinstatement, Closure, and Timeout .....	106
6.3.5.1. Loss of Nexus Notification .....	107
6.3.6. Session Continuation and Failure .....	108
6.4. Operational Parameter Negotiation Outside the Login Phase .	108
7. iSCSI Error Handling and Recovery.....	110
7.1. Overview .....	110
7.1.1. Background .....	110
7.1.2. Goals .....	110
7.1.3. Protocol Features and State Expectations .....	111
7.1.4. Recovery Classes .....	112
7.1.4.1. Recovery Within-command .....	113
7.1.4.2. Recovery Within-connection .....	114
7.1.4.3. Connection Recovery .....	115
7.1.4.4. Session Recovery .....	116

7.1.5. Error Recovery Hierarchy .....	116
7.2. Retry and Reassign in Recovery .....	118
7.2.1. Usage of Retry .....	119
7.2.2. Allegiance Reassignment .....	119
7.3. Usage Of Reject PDU in Recovery .....	121
7.4. Error Recovery Considerations for Discovery Sessions .....	121
7.4.1. ErrorRecoveryLevel for Discovery Sessions .....	121
7.4.2. Reinstatement Semantics for Discovery Sessions .....	122
7.4.2.1. Unnamed Discovery Sessions .....	123
7.4.2.2. Named Discovery Session .....	123
7.4.3. Target PDUs During Discovery .....	123
7.5. Connection Timeout Management .....	124
7.5.1. Timeouts on Transport Exception Events .....	124
7.5.2. Timeouts on Planned Decommissioning .....	124
7.6. Implicit Termination of Tasks .....	125
7.7. Format Errors .....	126
7.8. Digest Errors .....	126
7.9. Sequence Errors .....	128
7.10. Message Error Checking .....	129
7.11. SCSI Timeouts .....	129
7.12. Negotiation Failures .....	130
7.13. Protocol Errors .....	131
7.14. Connection Failures .....	131
7.15. Session Errors .....	132
8. State Transitions.....	134
8.1. Standard Connection State Diagrams .....	134
8.1.1. State Descriptions for Initiators and Targets .....	134
8.1.2. State Transition Descriptions for Initiators and Targets	135
8.1.3. Standard Connection State Diagram for an Initiator ....	139
8.1.4. Standard Connection State Diagram for a Target .....	141
8.2. Connection Cleanup State Diagram for Initiators and Targets	143
8.2.1. State Descriptions for Initiators and Targets .....	145
8.2.2. State Transition Descriptions for Initiators and Targets	146
8.3. Session State Diagrams .....	148
8.3.1. Session State Diagram for an Initiator .....	148

	iSCSI (Consolidated)	3/11/11
8.3.2.	Session State Diagram for a Target .....	149
8.3.3.	State Descriptions for Initiators and Targets .....	150
8.3.4.	State Transition Descriptions for Initiators and Targets .....	151
9.	Security Considerations.....	153
9.1.	iSCSI Security Mechanisms .....	153
9.2.	In-band Initiator-Target Authentication .....	154
9.2.1.	CHAP Considerations .....	155
9.2.2.	SRP Considerations .....	157
9.3.	IPsec .....	158
9.3.1.	Data Integrity and Authentication .....	158
9.3.2.	Confidentiality .....	159
9.3.3.	Policy, Security Associations, and Cryptographic Key Management .....	159
9.4.	Security Considerations for the X#NodeArchitecture Key ...	161
10.	Notes to Implementers.....	164
10.1.	Multiple Network Adapters .....	164
10.1.1.	Conservative Reuse of ISIDs .....	164
10.1.2.	iSCSI Name, ISID, and TPGT Use .....	165
10.2.	Autosense and Auto Contingent Allegiance (ACA) .....	167
10.3.	iSCSI Timeouts .....	167
10.4.	Command Retry and Cleaning Old Command Instances .....	168
10.5.	Synch and Steering Layer and Performance .....	169
10.6.	Considerations for State-dependent Devices and Long-lasting SCSI Operations .....	169
10.6.1.	Determining the Proper ErrorRecoveryLevel .....	170
10.7.	Multi-task Abort Implementation Considerations .....	171
11.	iSCSI PDU Formats.....	172
11.1.	iSCSI PDU Length and Padding .....	172
11.2.	PDU Template, Header, and Opcodes .....	172
11.2.1.	Basic Header Segment (BHS) .....	173
11.2.1.1.	I .....	174
11.2.1.2.	Opcode .....	174
11.2.1.3.	Final (F) bit .....	176
11.2.1.4.	Opcode-specific Fields .....	176

iSCSI (Consolidated)	3/11/11
11.2.1.5. TotalAHSLength .....	176
11.2.1.6. DataSegmentLength .....	176
11.2.1.7. LUN .....	176
11.2.1.8. Initiator Task Tag .....	177
11.2.2. Additional Header Segment (AHS) .....	177
11.2.2.1. AHSType .....	177
11.2.2.2. AHSLength .....	178
11.2.2.3. Extended CDB AHS .....	178
11.2.2.4. Bidirectional Expected Read-Data Length AHS .....	178
11.2.3. Header Digest and Data Digest .....	179
11.2.4. Data Segment .....	179
11.3. SCSI Command .....	179
11.3.1. Flags and Task Attributes (byte 1) .....	180
11.3.2. CmdSN - Command Sequence Number .....	181
11.3.3. ExpStatSN .....	182
11.3.4. Expected Data Transfer Length .....	182
11.3.5. CDB - SCSI Command Descriptor Block .....	183
11.3.6. Data Segment - Command Data .....	183
11.4. SCSI Response .....	183
11.4.1. Flags (byte 1) .....	184
11.4.2. Status .....	185
11.4.3. Response .....	186
11.4.4. SNACK Tag .....	187
11.4.5. Residual Count .....	187
11.4.5.1. Field Semantics .....	187
11.4.5.2. Residuals Concepts Overview .....	188
11.4.5.3. SCSI REPORT LUNS and Residual Overflow .....	188
11.4.6. Bidirectional Read Residual Count .....	190
11.4.7. Data Segment - Sense and Response Data Segment .....	190
11.4.7.1. SenseLength .....	191
11.4.7.2. Sense Data .....	191
11.4.8. ExpDataSN .....	192
11.4.9. StatSN - Status Sequence Number .....	192
11.4.10. ExpCmdSN - Next Expected CmdSN from this Initiator .....	193
11.4.11. MaxCmdSN - Maximum CmdSN from this Initiator .....	193
11.5. Task Management Function Request .....	194



iSCSI (Consolidated)	3/11/11
11.5.1. Function .....	194
11.5.2. TotalAHSLength and DataSegmentLength .....	198
11.5.3. LUN .....	198
11.5.4. Referenced Task Tag .....	198
11.5.5. RefCmdSN .....	198
11.5.6. ExpDataSN .....	199
11.6. Task Management Function Response .....	199
11.6.1. Response .....	200
11.6.2. TotalAHSLength and DataSegmentLength .....	202
11.7. SCSI Data-out & SCSI Data-in .....	202
11.7.1. F (Final) Bit .....	205
11.7.2. A (Acknowledge) bit .....	205
11.7.3. Flags (byte 1) .....	206
11.7.4. Target Transfer Tag and LUN .....	207
11.7.5. DataSN .....	207
11.7.6. Buffer Offset .....	207
11.7.7. DataSegmentLength .....	208
11.8. Ready To Transfer (R2T) .....	209
11.8.1. TotalAHSLength and DataSegmentLength .....	211
11.8.2. R2TSN .....	211
11.8.3. StatSN .....	211
11.8.4. Desired Data Transfer Length and Buffer Offset .....	211
11.8.5. Target Transfer Tag .....	211
11.9. Asynchronous Message .....	212
11.9.1. AsyncEvent .....	214
11.9.2. AsyncVCode .....	217
11.9.3. LUN .....	217
11.9.4. Sense Data and iSCSI Event Data .....	217
11.9.4.1. SenseLength .....	217
11.10. Text Request .....	218
11.10.1. F (Final) Bit .....	219
11.10.2. C (Continue) Bit .....	219
11.10.3. Initiator Task Tag .....	219
11.10.4. Target Transfer Tag .....	219
11.10.5. Text .....	220
11.11. Text Response .....	221

	iSCSI (Consolidated)	3/11/11
11.11.1.	F (Final) Bit .....	222
11.11.2.	C (Continue) Bit .....	223
11.11.3.	Initiator Task Tag .....	223
11.11.4.	Target Transfer Tag .....	223
11.11.5.	StatSN .....	224
11.11.6.	Text Response Data .....	224
11.12.	Login Request .....	224
11.12.1.	T (Transit) Bit .....	225
11.12.2.	C (Continue) Bit .....	226
11.12.3.	CSG and NSG .....	226
11.12.4.	Version .....	226
11.12.4.1.	Version-max .....	226
11.12.4.2.	Version-min .....	227
11.12.5.	ISID .....	227
11.12.6.	TSIH .....	229
11.12.7.	Connection ID - CID .....	229
11.12.8.	CmdSN .....	229
11.12.9.	ExpStatSN .....	230
11.12.10.	Login Parameters .....	230
11.13.	Login Response .....	231
11.13.1.	Version-max .....	231
11.13.2.	Version-active .....	232
11.13.3.	TSIH .....	232
11.13.4.	StatSN .....	232
11.13.5.	Status-Class and Status-Detail .....	233
11.13.6.	T (Transit) bit .....	236
11.13.7.	C (Continue) Bit .....	237
11.13.8.	Login Parameters .....	237
11.14.	Logout Request .....	237
11.14.1.	Reason Code .....	240
11.14.2.	TotalAHSLength and DataSegmentLength .....	241
11.14.3.	CID .....	241
11.14.4.	ExpStatSN .....	241
11.14.5.	Implicit termination of tasks .....	241
11.15.	Logout Response .....	242
11.15.1.	Response .....	243

	iSCSI (Consolidated)	3/11/11
11.15.2.	TotalAHSLength and DataSegmentLength .....	244
11.15.3.	Time2Wait .....	244
11.15.4.	Time2Retain .....	244
11.16.	SNACK Request .....	246
11.16.1.	Type .....	247
11.16.2.	Data Acknowledgement .....	248
11.16.3.	Resegmentation .....	248
11.16.4.	Initiator Task Tag .....	249
11.16.5.	Target Transfer Tag or SNACK Tag .....	249
11.16.6.	BegRun .....	250
11.16.7.	RunLength .....	250
11.17.	Reject .....	251
11.17.1.	Reason .....	252
11.17.2.	DataSN/R2TSN .....	253
11.17.3.	StatSN, ExpCmdSN and MaxCmdSN .....	253
11.17.4.	Complete Header of Bad PDU .....	254
11.18.	NOP-Out .....	254
11.18.1.	Initiator Task Tag .....	255
11.18.2.	Target Transfer Tag .....	255
11.18.3.	Ping Data .....	256
11.19.	NOP-In .....	257
11.19.1.	Target Transfer Tag .....	258
11.19.2.	StatSN .....	258
11.19.3.	LUN .....	258
12.	iSCSI Security Text Keys and Authentication Methods.....	259
12.1.	AuthMethod .....	259
12.1.1.	Kerberos .....	261
12.1.2.	Secure Remote Password (SRP) .....	262
12.1.3.	Challenge Handshake Authentication Protocol (CHAP) ..	263
13.	Login/Text Operational Text Keys.....	266
13.1.	HeaderDigest and DataDigest .....	266
13.2.	MaxConnections .....	269
13.3.	SendTargets .....	269
13.4.	TargetName .....	270
13.5.	InitiatorName .....	270

	iSCSI (Consolidated)	3/11/11
13.6.	TargetAlias .....	271
13.7.	InitiatorAlias .....	271
13.8.	TargetAddress .....	272
13.9.	TargetPortalGroupTag .....	273
13.10.	InitialR2T .....	274
13.11.	ImmediateData .....	274
13.12.	MaxRecvDataSegmentLength .....	275
13.13.	MaxBurstLength .....	276
13.14.	FirstBurstLength .....	276
13.15.	DefaultTime2Wait .....	277
13.16.	DefaultTime2Retain .....	277
13.17.	MaxOutstandingR2T .....	278
13.18.	DataPDUInOrder .....	278
13.19.	DataSequenceInOrder .....	279
13.20.	ErrorRecoveryLevel .....	280
13.21.	SessionType .....	280
13.22.	The Private or Public Extension Key Format .....	281
13.23.	Task Reporting .....	281
13.24.	iSCSIProtocolLevel Negotiation .....	282
13.25.	Obsoleted Keys .....	283
13.26.	X#NodeArchitecture .....	283
13.26.1.	Definition .....	283
13.26.2.	Implementation Requirements .....	284
14.	IANA Considerations.....	285
Appendix A.	Examples .....	291
Read Operation Example	.....	291
Write Operation Example	.....	292
R2TSN/DataSN Use Examples	.....	292
CRC Examples	.....	296
Appendix B.	Login Phase Examples .....	298
Appendix C.	SendTargets Operation .....	308
Appendix D.	Algorithmic Presentation of Error Recovery Classes .	313

iSCSI (Consolidated)	3/11/11
D.2.1. Procedure Descriptions .....	316
Appendix E. Clearing Effects of Various Events on Targets .....	332

## 1. Introduction

The Small Computer Systems Interface (SCSI) is a popular family of protocols for communicating with I/O devices, especially storage devices. SCSI is a client-server architecture. Clients of a SCSI interface are called "initiators". Initiators issue SCSI "commands" to request services from components, logical units of a server known as a "target". A "SCSI transport" maps the client-server SCSI protocol to a specific interconnect. An Initiator is one endpoint of a SCSI transport and a target is the other endpoint.

The SCSI protocol has been mapped over various transports, including Parallel SCSI, IPI, IEEE-1394 (firewire) and Fibre Channel. These transports are I/O specific and have limited distance capabilities.

The iSCSI protocol defined in this document describes a means of transporting of the SCSI packets over TCP/IP, providing for an interoperable solution which can take advantage of existing Internet infrastructure, Internet management facilities and address distance limitations.

## 2. Definitions and Acronyms

### 2.1. Definitions

- Alias: An alias string can also be associated with an iSCSI Node. The alias allows an organization to associate a user-friendly string with the iSCSI Name. However, the alias string is not a substitute for the iSCSI Name.
- CID (Connection ID): Connections within a session are identified by a connection ID. It is a unique ID for this connection within the session for the initiator. It is generated by the initiator and presented to the target during login requests and during logouts that close connections.
- Connection: A connection is a TCP connection. Communication between the initiator and target occurs over one or more TCP connections. The TCP connections carry control messages, SCSI commands, parameters, and data within iSCSI Protocol Data Units (iSCSI PDUs).
- I/O Buffer: A buffer that is used in a SCSI Read or Write operation so SCSI data may be sent from or received into that buffer. For a read or write data transfer to take place for a task, an I/O Buffer is required on the initiator and at least one is required on the target.
- INCITS: INCITS stands for InterNational Committee of Information Technology Standards. The INCITS has a broad standardization scope within the field of Information and Communications Technologies (ICT), encompassing storage, processing, transfer, display, management, organization, and retrieval of information. INCITS serves as ANSI's Technical Advisory Group for the ISO/IEC Joint Technical Committee 1 (JTC 1). See <http://www.incits.org>.
- InfiniBand: An I/O architecture originally intended to replace PCI and to address high performance server interconnectivity [IB].
- iSCSI Device: A SCSI Device using an iSCSI service delivery subsystem. Service Delivery Subsystem is defined by [SAM2] as a transport mechanism for SCSI commands and responses.

- iSCSI Initiator Name: The iSCSI Initiator Name specifies the worldwide unique name of the initiator.
- iSCSI Initiator Node: The "initiator" device. The word "initiator" has been appropriately qualified as either a port or a device in the rest of the document when the context is ambiguous. All unqualified usages of "initiator" refer to an initiator port (or device) depending on the context.
- iSCSI Layer: This layer builds/receives iSCSI PDUs and relays/receives them to/from one or more TCP connections that form an initiator-target "session".
- iSCSI Name: The name of an iSCSI initiator or iSCSI target.
- iSCSI Node: The iSCSI Node represents a single iSCSI initiator or iSCSI target or a single instance of each. There are one or more iSCSI Nodes within a Network Entity. The iSCSI Node is accessible via one or more Network Portals. An iSCSI Node is identified by its iSCSI Name. The separation of the iSCSI Name from the addresses used by and for the iSCSI Node allows multiple iSCSI nodes to use the same address, and the same iSCSI node to use multiple addresses.
- iSCSI Target Name: The iSCSI Target Name specifies the worldwide unique name of the target.
- iSCSI Target Node: The "target" device. The word "target" has been appropriately qualified as either a port or a device in the rest of the document when the context is ambiguous. All unqualified usages of "target" refer to a target port (or device) depending on the context.
- iSCSI Task: An iSCSI task is an iSCSI request for which a response is expected.
- iSCSI Transfer Direction: The iSCSI transfer direction is defined with regard to the initiator. Outbound or outgoing transfers are transfers from the initiator to the target, while



inbound or incoming transfers are from the target to the initiator.

- ISID: The initiator part of the Session Identifier. It is explicitly specified by the initiator during Login.

- I\_T nexus: According to [SAM2], the I\_T nexus is a relationship between a SCSI Initiator Port and a SCSI Target Port. For iSCSI, this relationship is a session, defined as a relationship between an iSCSI Initiator's end of the session (SCSI Initiator Port) and the iSCSI Target's Portal Group. The I\_T nexus can be identified by the conjunction of the SCSI port names; that is, the I\_T nexus identifier is the tuple (iSCSI Initiator Name + ',i,'+ ISID, iSCSI Target Name + ',t,'+ Portal Group Tag).

- NAA: Network Address Authority, a naming format defined by the INCITS T11 Fibre Channel protocols [FC-FS].

- Network Entity: The Network Entity represents a device or gateway that is accessible from the IP network. A Network Entity must have one or more Network Portals, each of which can be used to gain access to the IP network by some iSCSI Nodes contained in that Network Entity.

- Network Portal: The Network Portal is a component of a Network Entity that has a TCP/IP network address and that may be used by an iSCSI Node within that Network Entity for the connection(s) within one of its iSCSI sessions. A Network Portal in an initiator is identified by its IP address. A Network Portal in a target is identified by its IP address and its listening TCP port.

- Originator: In a negotiation or exchange, the party that initiates the negotiation or exchange.

- PDU (Protocol Data Unit): The initiator and target divide their communications into messages. The term "iSCSI protocol data unit" (iSCSI PDU) is used for these messages.

- Portal Groups: iSCSI supports multiple connections within the same session; some implementations will have the ability to combine connections in a session across multiple Network Portals.

A Portal Group defines a set of Network Portals within an iSCSI Network Entity that collectively supports the capability of coordinating a session with connections spanning these portals. Not all Network Portals within a Portal Group need participate in every session connected through that Portal Group. One or more Portal Groups may provide access to an iSCSI Node. Each Network Portal, as utilized by a given iSCSI Node, belongs to exactly one portal group within that node.

- Portal Group Tag: This 16-bit quantity identifies a Portal Group within an iSCSI Node. All Network Portals with the same portal group tag in the context of a given iSCSI Node are in the same Portal Group.

- Recovery R2T: An R2T generated by a target upon detecting the loss of one or more Data-Out PDUs through one of the following means: a digest error, a sequence error, or a sequence reception timeout. A recovery R2T carries the next unused R2TSN, but requests all or part of the data burst that an earlier R2T (with a lower R2TSN) had already requested.

- Responder: In a negotiation or exchange, the party that responds to the originator of the negotiation or exchange.

- SAS: Serial Attached SCSI. The Serial Attached SCSI (SAS) standard contains both a physical layer compatible with Serial ATA, and protocols for transporting SCSI commands to SAS devices and ATA commands to SATA devices [SAS].

- SCSI Device: This is the SAM2 term for an entity that contains one or more SCSI ports that are connected to a service delivery subsystem and supports a SCSI application protocol. For example, a SCSI Initiator Device contains one or more SCSI Initiator Ports and zero or more application clients. A Target Device contains one or more SCSI Target Ports and one or more device servers and associated logical units. For iSCSI, the SCSI Device is the component within an iSCSI Node that provides the SCSI functionality. As such, there can be, at most, one SCSI Device within a given iSCSI Node. Access to the SCSI Device can only be

achieved in an iSCSI normal operational session. The SCSI Device Name is defined to be the iSCSI Name of the node.

- SCSI Layer: This builds/receives SCSI CDBs (Command Descriptor Blocks) and relays/receives them with the remaining command execute [SAM2] parameters to/from the iSCSI Layer.

- Session: The group of TCP connections that link an initiator with a target form a session (loosely equivalent to a SCSI I-T nexus). TCP connections can be added and removed from a session. Across all connections within a session, an initiator sees one and the same target.

- SCSI Initiator Port: This maps to the endpoint of an iSCSI normal operational session. An iSCSI normal operational session is negotiated through the login process between an iSCSI initiator node and an iSCSI target node. At successful completion of this process, a SCSI Initiator Port is created within the SCSI Initiator Device. The SCSI Initiator Port Name and SCSI Initiator Port Identifier are both defined to be the iSCSI Initiator Name together with (a) a label that identifies it as an initiator port name/identifier and (b) the ISID portion of the session identifier.

- SCSI Port: This is the SAM2 term for an entity in a SCSI Device that provides the SCSI functionality to interface with a service delivery subsystem. For iSCSI, the definition of the SCSI Initiator Port and the SCSI Target Port are different.

- SCSI Port Name: A name made up as UTF-8 characters and includes the iSCSI Name + 'i' or 't' + ISID or Portal Group Tag.

- SCSI-Presented Data Transfer Length (SPDTL): SPDTL is the aggregate data length of the data that the SCSI layer logically "presents" to the iSCSI layer for a Data-In or Data-Out transfer in the context of a SCSI task. For a bidirectional task, there are two SPDTL values -- one for Data-In and one for Data-Out. Note that the notion of "presenting" includes immediate data per the data transfer model in [SAM2], and excludes overlapping data transfers, if any, requested by the SCSI layer.

- SCSI Target Port: This maps to an iSCSI Target Portal Group.
- SCSI Target Port Name and SCSI Target Port Identifier: These are both defined to be the iSCSI Target Name together with (a) a label that identifies it as a target port name/identifier and (b) the portal group tag.
- SRP: SCSI RDMA Protocol. SRP defines a SCSI protocol mapping onto the InfiniBand (tm) Architecture and/or functionally similar Remote DMA-capable protocols [SRP].
- SSID (Session ID): A session between an iSCSI initiator and an iSCSI target is defined by a session ID that is a tuple composed of an initiator part (ISID) and a target part (Target Portal Group Tag). The ISID is explicitly specified by the initiator at session establishment. The Target Portal Group Tag is implied by the initiator through the selection of the TCP endpoint at connection establishment. The TargetPortalGroupTag key must also be returned by the target as a confirmation during connection establishment.
- T10: A technical committee within INCITS that develops standards and technical reports on I/O interfaces, particularly the series of SCSI (Small Computer Systems Interface) standards. See <http://www.t10.org>.
- T11: A technical committee within INCITS responsible for standards development in the areas of Intelligent Peripheral Interface (IPI), High-Performance Parallel Interface (HIPPI) and Fibre Channel (FC). See <http://www.t11.org>.
- Target Portal Group Tag: A numerical identifier (16-bit) for an iSCSI Target Portal Group.
- Third-party: A term used in this document to denote nexus objects (I\_T or I\_T\_L) and iSCSI sessions that reap the side effects of actions that take place in the context of a separate iSCSI session, while being third parties to the action that caused the side effects. One example of a third-party session is an iSCSI session hosting an I\_T\_L nexus to an LU that is reset with an LU Reset TMF via a separate I\_T nexus.

- TSIH (Target Session Identifying Handle): A target assigned tag for a session with a specific named initiator. The target generates it during session establishment. Other than defining it as a 16 bit binary string, its internal format and content are not defined by this protocol but for the all 0 value that is reserved and used by the initiator to indicate a new session. It is given to the target during additional connection establishment for the same session.

## 2.2. Acronyms

Acronym	Definition
3DES	Triple Data Encryption Standard
ACA	Auto Contingent Allegiance
AEN	Asynchronous Event Notification
AES	Advanced Encryption Standard
AH	Additional Header (not the IPsec AH!)
AHS	Additional Header Segment
API	Application Programming Interface
ASC	Additional Sense Code
ASCII	American Standard Code for Information Interchange
ASCQ	Additional Sense Code Qualifier
BHS	Basic Header Segment
CBC	Cipher Block Chaining
CD	Compact Disk
CDB	Command Descriptor Block
CHAP	Challenge Handshake Authentication Protocol
CID	Connection ID
CO	Connection Only
CRC	Cyclic Redundancy Check
CRL	Certificate Revocation List
CSG	Current Stage
CSM	Connection State Machine
DES	Data Encryption Standard
DNS	Domain Name Server
DOI	Domain of Interpretation
DVD	Digital Versatile Disk
EDTL	Expected Data Transfer Length
ESP	Encapsulating Security Payload
EUI	Extended Unique Identifier
FFP	Full Feature Phase

FFPO	Full Feature Phase Only
Gbps	Gigabits per Second
HBA	Host Bus Adapter
HMAC	Hashed Message Authentication Code
I_T	Initiator_Target
I_T_L	Initiator_Target_LUN
IANA	Internet Assigned Numbers Authority
IB	InfiniBand
ID	Identifier
IDN	Internationalized Domain Name
IEEE	Institute of Electrical & Electronics Engineers
IETF	Internet Engineering Task Force
IKE	Internet Key Exchange
I/O	Input-Output
IO	Initialize Only
IP	Internet Protocol
IPsec	Internet Protocol Security
IPv4	Internet Protocol Version 4
IPv6	Internet Protocol Version 6
IQN	iSCSI Qualified Name
iSCSI	Internet SCSI
iSER	iSCSI Extensions for RDMA
ISID	Initiator Session ID
iSNS	Internet Storage Name Service (see [RFC4171])
ITN	iSCSI Target Name
ITT	Initiator Task Tag
KRB5	Kerberos V5
LFL	Lower Functional Layer
LTDS	Logical-Text-Data-Segment
LO	Leading Only
LU	Logical Unit
LUN	Logical Unit Number
MAC	Message Authentication Codes
NA	Not Applicable
NAA	Network Address Authority
NIC	Network Interface Card
NOP	No Operation
NSG	Next Stage
OS	Operating System
PDU	Protocol Data Unit
PKI	Public Key Infrastructure

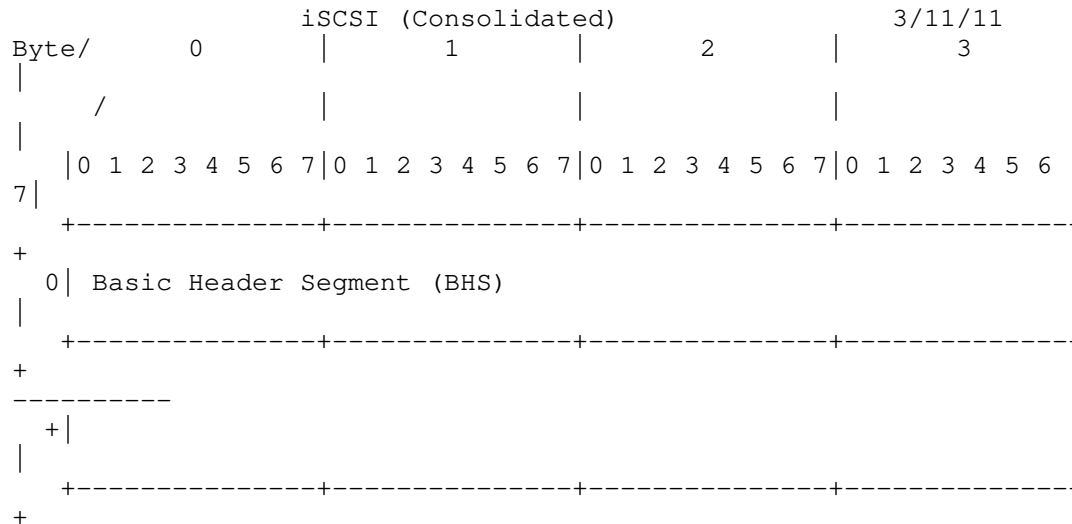
R2T	Ready To Transfer
R2TSN	Ready To Transfer Sequence Number
RDMA	Remote Direct Memory Access
RFC	Request For Comments
SAM	SCSI Architecture Model
SAM2	SCSI Architecture Model - 2
SAN	Storage Area Network
SAS	Serial Attached SCSI
SCSI	Small Computer Systems Interface
SN	Sequence Number
SNACK	Selective Negative Acknowledgment - also Sequence Number Acknowledgement for data
SPDTL	SCSI-Presented Data Transfer Length
SPKM	Simple Public-Key Mechanism
SRP	Secure Remote Password, also SCSI RDMA Protocol
SSID	Session ID
SW	Session Wide
TCB	Task Control Block
TCP	Transmission Control Protocol
TMF	Task Management Function
TPGT	Target Portal Group Tag
TSIH	Target Session Identifying Handle
TTT	Target Transfer Tag
UFL	Upper Functional Layer
ULP	Upper Level Protocol
URN	Uniform Resource Names
UTF	Universal Transformation Format
WG	Working Group

### 2.3. Conventions

In examples, "I->" and "T->" show iSCSI PDUs sent by the initiator and target respectively.

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in RFC 2119 [RFC2119].

iSCSI messages - PDUs - are represented by diagrams as in the following example:



The diagrams include byte and bit numbering.

The following representation and ordering rules are observed in this document:

- Word Rule
- Half-word Rule
- Byte Rule

#### 2.3.1. Word Rule

A word holds four consecutive bytes. Whenever a word has numeric content, it is considered an unsigned number in base 2 positional representation with the lowest numbered byte (e.g., byte 0) bit 0 representing  $2^{31}$  and bit 1 representing  $2^{30}$  through lowest numbered byte + 3 (e.g., byte 3) bit 7 representing  $2^0$ .

Decimal and hexadecimal representation of word values map this representation to decimal or hexadecimal positional notation.



### 2.3.2. Half-Word Rule

A half-word holds two consecutive bytes. Whenever a half-word has numeric content it is considered an unsigned number in base 2 positional representation with the lowest numbered byte (e.g., byte 0) bit 0 representing  $2^{15}$  and bit 1 representing  $2^{14}$  through lowest numbered byte + 1 (e.g., byte 1) bit 7 representing  $2^0$ .

Decimal and hexadecimal representation of half-word values map this representation to decimal or hexadecimal positional notation.

### 2.3.3. Byte Rule

For every PDU, bytes are sent and received in increasing numbered order (network order).

Whenever a byte has numerical content it is considered an unsigned number in base 2 positional representation with bit 0 representing  $2^7$  and bit 1 representing  $2^6$  through bit 7 representing  $2^0$ .

### 3. UML Conventions

#### 3.1. UML Conventions Overview

The SCSI Architecture Model (SAM) uses class diagrams and object diagrams with notation that is based on the Unified Modeling Language [UML]. Therefore, this document also uses UML to model the relationships for SCSI and iSCSI objects.

A treatise on the graphical notation used in UML is beyond the scope of this document. However, given the use of ASCII drawing for UML static class diagrams, a description of the notational conventions used in this document is included in the remainder of this section.

#### 3.2. Multiplicity Notion

Not specified      The number of instances of an attribute is not specified.

1      One instance of the class or attribute exists.

0..\*      Zero or more instances of the class or attribute exist.

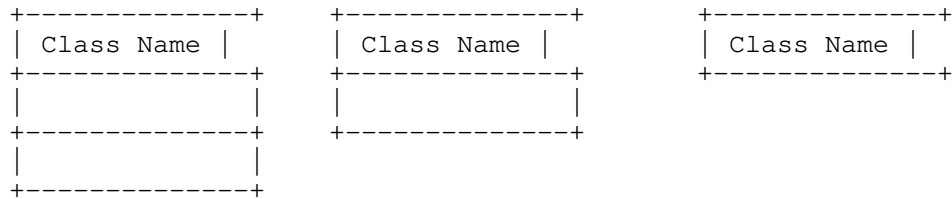
1..\*      One or more instances of the class or attribute exist.

0..1      Zero or one instance of the class or attribute exists.

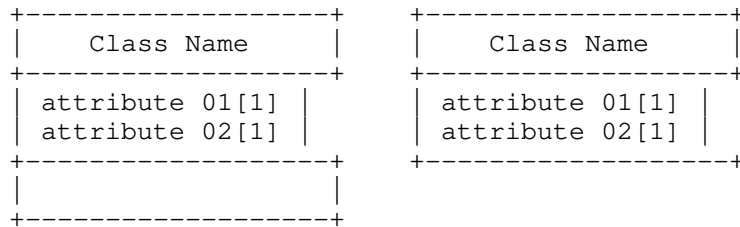
n..m      n to m instances of the class or attribute exist (e.g., 2..8).

x, n..m      Multiple disjoint instances of the class or attribute exist (e.g., 2, 8..15).

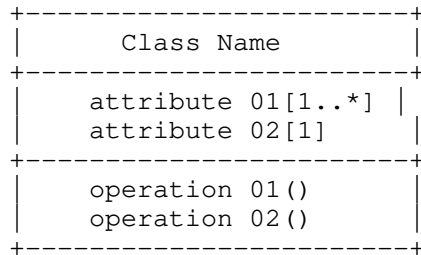
## 3.3. Class Diagram Conventions



The previous three diagrams are examples of a class with no attributes and with no operations.

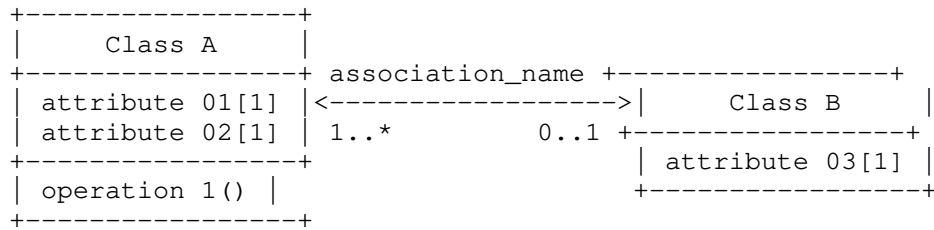


The preceding two diagrams are examples of a class with attributes and with no operations.

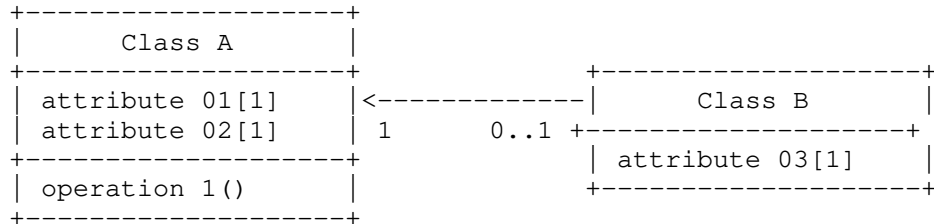


The preceding diagram is an example of a class with attributes that have a specified multiplicity and operations.

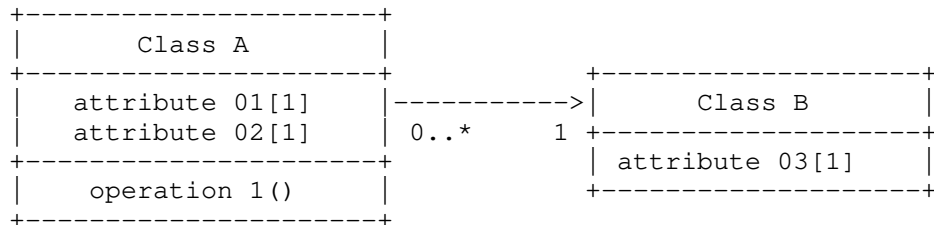
## 3.4. Class Diagram Notation for Associations



The preceding diagram is an example where Class A knows about Class B (i.e., read as "Class A association\_name ClassB") and Class B knows about Class A (i.e., read as "Class B association\_name Class A"). The use of association\_name is optional. The multiplicity notation (1..\* and 0..1) indicates the number of instances of the object.

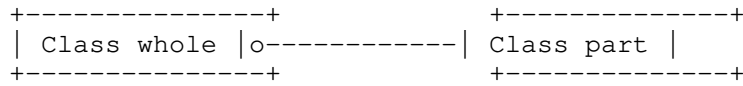


The preceding diagram is an example where Class B knows about Class A (i.e., read as "Class B knows about Class A") but Class A does not know about Class B.

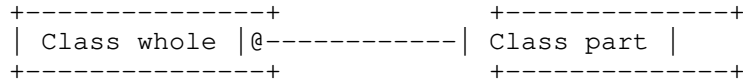


The preceding diagram is an example where Class A knows about Class B (i.e., read as "Class A knows about Class B") but Class B does not know about Class A.

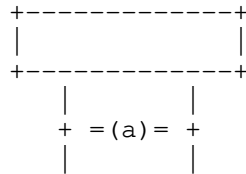
## 3.5. Class Diagram Notation for Aggregations



The preceding diagram is an example where Class whole is an aggregate that contains Class part and where Class part may continue to exist even if Class whole is removed (i.e., read as "the whole contains the part").

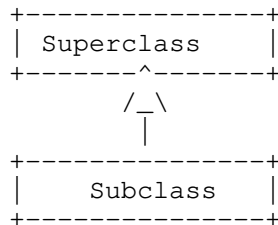


The preceding diagram is an example where Class whole is an aggregate that contains Class part where Class part shall only belong to one Class whole, and the Class part shall not continue to exist if the Class whole is removed (i.e., read as "the whole contains the part").



The preceding diagram is an example where there is a constraint between the associations where the (a) footnote describes the constraint.

## 3.6. Class Diagram Notation for Generalizations



The preceding diagram is an example where the subclass is a kind of superclass. A subclass shares all the attributes and

iSCSI (Consolidated) 3/11/11  
operations of the superclass (i.e., the subclass inherits from the  
superclass).

## 4. Overview

### 4.1. SCSI Concepts

The SCSI Architecture Model-2 [SAM2] describes in detail the architecture of the SCSI family of I/O protocols. This section provides a brief background of the SCSI architecture and is intended to familiarize readers with its terminology.

At the highest level, SCSI is a family of interfaces for requesting services from I/O devices, including hard drives, tape drives, CD and DVD drives, printers, and scanners. In SCSI terminology, an individual I/O device is called a "logical unit" (LU).

SCSI is a client-server architecture. Clients of a SCSI interface are called "initiators". Initiators issue SCSI "commands" to request services from components, logical units, of a server known as a "target". The "device server" on the logical unit accepts SCSI commands and processes them.

A "SCSI transport" maps the client-server SCSI protocol to a specific interconnect. Initiator is one endpoint of a SCSI transport. The "target" is the other endpoint. A target can contain multiple Logical Units (LUs). Each Logical Unit has an address within a target called a Logical Unit Number (LUN).

A SCSI task is a SCSI command or possibly a linked set of SCSI commands. Some LUs support multiple pending (queued) tasks, but the queue of tasks is managed by the logical unit. The target uses an initiator provided "task tag" to distinguish between tasks. Only one command in a task can be outstanding at any given time.

Each SCSI command results in an optional data phase and a required response phase. In the data phase, information can travel from the initiator to target (e.g., WRITE), target to initiator (e.g., READ), or in both directions. In the response phase, the target returns the final status of the operation, including any errors.

Command Descriptor Blocks (CDB) are the data structures used to contain the command parameters that an initiator sends to a

target. The CDB content and structure is defined by [SAM2] and device-type specific SCSI standards.

#### 4.2. iSCSI Concepts and Functional Overview

The iSCSI protocol is a mapping of the SCSI command, event and task management model (see [SAM2]) over the TCP protocol. SCSI commands are carried by iSCSI requests and SCSI responses and status are carried by iSCSI responses. iSCSI also uses the request response mechanism for iSCSI protocol mechanisms.

For the remainder of this document, the terms "initiator" and "target" refer to "iSCSI initiator node" and "iSCSI target node", respectively (see iSCSI) unless otherwise qualified.

In keeping with similar protocols, the initiator and target divide their communications into messages. This document uses the term "iSCSI protocol data unit" (iSCSI PDU) for these messages.

For performance reasons, iSCSI allows a "phase-collapse". A command and its associated data may be shipped together from initiator to target, and data and responses may be shipped together from targets.

The iSCSI transfer direction is defined with respect to the initiator. Outbound or outgoing transfers are transfers from an initiator to a target, while inbound or incoming transfers are from a target to an initiator.

An iSCSI task is an iSCSI request for which a response is expected.

In this document "iSCSI request", "iSCSI command", request, or (unqualified) command have the same meaning. Also, unless otherwise specified, status, response, or numbered response have the same meaning.



#### 4.2.1. Layers and Sessions

The following conceptual layering model is used to specify initiator and target actions and the way in which they relate to transmitted and received Protocol Data Units:

- the SCSI layer builds/receives SCSI CDBs (Command Descriptor Blocks) and passes/receives them with the remaining command execute parameters ([SAM2]) to/from
- the iSCSI layer that builds/receives iSCSI PDUs and relays/receives them to/from one or more TCP connections;
- the group of connections form an initiator-target "session".

Communication between the initiator and target occurs over one or more TCP connections. The TCP connections carry control messages, SCSI commands, parameters, and data within iSCSI Protocol Data Units (iSCSI PDUs). The group of TCP connections that link an initiator with a target form a session (equivalent to a SCSI I\_T nexus, see SCSI ). A session is defined by a session ID that is composed of an initiator part and a target part. TCP connections can be added and removed from a session. Each connection within a session is identified by a connection ID (CID).

Across all connections within a session, an initiator sees one "target image". All target identifying elements, such as LUN, are the same. A target also sees one "initiator image" across all connections within a session. Initiator-identifying elements, such as the Initiator Task Tag, are global across the session regardless of the connection on which they are sent or received.

iSCSI targets and initiators MUST support at least one TCP connection and MAY support several connections in a session. For error recovery purposes, targets and initiators that support a single active connection in a session SHOULD support two connections during recovery.

#### 4.2.2. Ordering and iSCSI Numbering

iSCSI uses Command and Status numbering schemes and a Data sequencing scheme.

Command numbering is session-wide and is used for ordered command delivery over multiple connections. It can also be used as a mechanism for command flow control over a session.

Status numbering is per connection and is used to enable missing status detection and recovery in the presence of transient or permanent communication errors.

Data sequencing is per command or part of a command (R2T-triggered sequence) and is used to detect missing data and/or R2T PDUs due to header digest errors.

Typically, fields in the iSCSI PDUs communicate the Sequence Numbers between the initiator and target. During periods when traffic on a connection is unidirectional, iSCSI NOP-Out/In PDUs may be utilized to synchronize the command and status ordering counters of the target and initiator.

The iSCSI session abstraction is equivalent to the SCSI I\_T nexus, and the iSCSI session provides an ordered command delivery from the SCSI initiator to the SCSI target. For detailed design considerations that led to the iSCSI session model as it is defined here and how it relates the SCSI command ordering features defined in SCSI specifications to the iSCSI concepts see [RFC3783].

#### 4.2.2.1. Command Numbering and Acknowledging

iSCSI performs ordered command delivery within a session. All commands (initiator-to-target PDUs) in transit from the initiator to the target are numbered.

iSCSI considers a task to be instantiated on the target in response to every request issued by the initiator. A set of task management operations including abort and reassign (see Section 10.5 "Task Management Function Request") may be performed on any iSCSI task.

Some iSCSI tasks are SCSI tasks, and many SCSI activities are related to a SCSI task ([SAM2]). In all cases, the task is identified by the Initiator Task Tag for the life of the task.

The command number is carried by the iSCSI PDU as CmdSN (Command-Sequence-Number). The numbering is session-wide. Outgoing iSCSI PDUs carry this number. The iSCSI initiator allocates CmdSNs with a 32-bit unsigned counter (modulo  $2^{32}$ ). Comparisons and arithmetic on CmdSN use Serial Number Arithmetic as defined in [RFC1982] where SERIAL\_BITS = 32.

Commands meant for immediate delivery are marked with an immediate delivery flag; they MUST also carry the current CmdSN. CmdSN does not advance after a command marked for immediate delivery is sent.

Command numbering starts with the first login request on the first connection of a session (the leading login on the leading connection) and command numbers are incremented by 1 for every non-immediate command issued afterwards.

If immediate delivery is used with task management commands, these commands may reach the target before the tasks on which they are supposed to act. However their CmdSN serves as a marker of their position in the stream of commands. The initiator and target must ensure that the task management commands act as specified by [SAM2]. For example, both commands and responses appear as if delivered in order. Whenever CmdSN for an outgoing PDU is not specified by an explicit rule, CmdSN will carry the current value of the local CmdSN variable (see later in this section).

The means by which an implementation decides to mark a PDU for immediate delivery or by which iSCSI decides by itself to mark a PDU for immediate delivery are beyond the scope of this document.

The number of commands used for immediate delivery is not limited and their delivery to execution is not acknowledged through the numbering scheme. Immediate commands MAY be rejected by the iSCSI target layer due to lack of resources. An iSCSI target MUST be able to handle at least one immediate task management command and one immediate non-task-management iSCSI command per connection at any time.

In this document, delivery for execution means delivery to the SCSI execution engine or an iSCSI protocol specific execution

engine (e.g., for text requests with public or private extension keys involving an execution component). With the exception of the commands marked for immediate delivery, the iSCSI target layer MUST deliver the commands for execution in the order specified by CmdSN. Commands marked for immediate delivery may be delivered by the iSCSI target layer for execution as soon as detected. iSCSI may avoid delivering some commands to the SCSI target layer if required by a prior SCSI or iSCSI action (e.g., CLEAR TASK SET Task Management request received before all the commands on which it was supposed to act).

On any connection, the iSCSI initiator MUST send the commands in increasing order of CmdSN, except for commands that are retransmitted due to digest error recovery and connection recovery.

For the numbering mechanism, the initiator and target maintain the following three variables for each session:

- CmdSN - the current command Sequence Number, advanced by 1 on each command shipped except for commands marked for immediate delivery. CmdSN always contains the number to be assigned to the next Command PDU.
- ExpCmdSN - the next expected command by the target. The target acknowledges all commands up to, but not including, this number. The initiator treats all commands with CmdSN less than ExpCmdSN as acknowledged. The target iSCSI layer sets the ExpCmdSN to the largest non-immediate CmdSN that it can deliver for execution "plus 1" per [RFC1982]. There MUST NOT be any holes in the acknowledged CmdSN sequence.
- MaxCmdSN - the maximum number to be shipped. The queuing capacity of the receiving iSCSI layer is  $\text{MaxCmdSN} - \text{ExpCmdSN} + 1$ .

The initiator's ExpCmdSN and MaxCmdSN are derived from target-to-initiator PDU fields. Comparisons and arithmetic on ExpCmdSN and MaxCmdSN MUST use Serial Number Arithmetic as defined in [RFC1982] where SERIAL\_BITS = 32.

The target MUST NOT transmit a MaxCmdSN that is less than ExpCmdSN-1. For non-immediate commands, the CmdSN field can take any value from ExpCmdSN to MaxCmdSN inclusive. The target MUST silently ignore any non-immediate command outside of this range or non-immediate duplicates within the range. The CmdSN carried by immediate commands may lie outside the ExpCmdSN to MaxCmdSN range. For example, if the initiator has previously sent a non-immediate command carrying the CmdSN equal to MaxCmdSN, the target window is closed. For group task management commands issued as immediate commands, CmdSN indicates the scope of the group action (e.g., on ABORT TASK SET indicates which commands are to be aborted).

MaxCmdSN and ExpCmdSN fields are processed by the initiator as follows:

- If the PDU MaxCmdSN is less than the PDU ExpCmdSN-1 (in Serial Arithmetic Sense), they are both ignored.
- If the PDU MaxCmdSN is greater than the local MaxCmdSN (in Serial Arithmetic Sense), it updates the local MaxCmdSN; otherwise, it is ignored.
- If the PDU ExpCmdSN is greater than the local ExpCmdSN (in Serial Arithmetic Sense), it updates the local ExpCmdSN; otherwise, it is ignored.

This sequence is required because updates may arrive out of order (e.g., the updates are sent on different TCP connections).

iSCSI initiators and targets MUST support the command numbering scheme.

A numbered iSCSI request will not change its allocated CmdSN, regardless of the number of times and circumstances in which it is reissued (see Section 6.2.1 "Usage of Retry"). At the target, CmdSN is only relevant while the command has not created any state related to its execution (execution state); afterwards, CmdSN becomes irrelevant. Testing for the execution state (represented by identifying the Initiator Task Tag) MUST precede any other action at the target. If no execution state is found, it is

followed by ordering and delivery. If an execution state is found, it is followed by delivery if it has not already been delivered.

If an initiator issues a command retry for a command with CmdSN R on a connection when the session CmdSN value is Q, it MUST NOT advance the CmdSN past  $R + 2^{31} - 1$  unless the connection is no longer operational (i.e., it has returned to the FREE state, see Section 7.1.3 "Standard Connection State Diagram for an Initiator"), the connection has been reinstated (see Section 5.3.4 "Connection Reinstatement"), or a non-immediate command with CmdSN equal or greater than Q was issued subsequent to the command retry on the same connection and the reception of that command is acknowledged by the target (see Section 9.4 "Command Retry and Cleaning Old Command Instances").

A target command response or Data-in PDU with status MUST NOT precede the command acknowledgement. However, the acknowledgement MAY be included in the response or the Data-in PDU.

#### 4.2.2.2. Response/Status Numbering and Acknowledging

Responses in transit from the target to the initiator are numbered. The StatSN (Status Sequence Number) is used for this purpose. StatSN is a counter maintained per connection. ExpStatSN is used by the initiator to acknowledge status. The status sequence number space is 32-bit unsigned-integers and the arithmetic operations are the regular  $\text{mod}(2^{32})$  arithmetic.

Status numbering starts with the Login response to the first Login request of the connection. The Login response includes an initial value for status numbering (any initial value is valid).

To enable command recovery, the target MAY maintain enough state information for data and status recovery after a connection failure. A target doing so can safely discard all of the state information maintained for recovery of a command after the delivery of the status for the command (numbered StatSN) is acknowledged through ExpStatSN.

A large absolute difference between StatSN and ExpStatSN may indicate a failed connection. Initiators MUST undertake recovery

actions if the difference is greater than an implementation defined constant that MUST NOT exceed  $2^{31}-1$ .

Initiators and Targets MUST support the response-numbering scheme.

#### 4.2.2.3. Response Ordering

##### 4.2.2.3.1. Need for Response Ordering

Whenever an iSCSI session is composed of multiple connections, the Response PDUs (task responses or TMF responses) originating in the target SCSI layer are distributed onto the multiple connections by the target iSCSI layer according to iSCSI connection allegiance rules. This process generally may not preserve the ordering of the responses by the time they are delivered to the initiator SCSI layer.

Since ordering is not expected across SCSI responses anyway, this approach works fine in the general case. However, to address the special cases where some ordering is desired by the SCSI layer, the following "Response Fence" semantics are defined with respect to handling SCSI response messages as they are handed off from the SCSI protocol layer to the iSCSI layer.

##### 4.2.2.3.2. Response Ordering Model Description

The target SCSI protocol layer hands off the SCSI response messages to the target iSCSI layer by invoking the "Send Command Complete" protocol data service ([SAM2], clause 5.4.2) and "Task Management Function Executed" ([SAM2], clause 6.9) service. On receiving the SCSI response message, the iSCSI layer exhibits the Response Fence behavior for certain SCSI response messages (Section 4.2.2.3.4 describes the specific instances where the semantics must be realized).

Whenever the Response Fence behavior is required for a SCSI response message, the target iSCSI layer MUST ensure that the following conditions are met in delivering the response message to the initiator iSCSI layer:

Response with Response Fence MUST be delivered chronologically after all the "preceding" responses on the I\_T\_L nexus, if

the preceding responses are delivered at all, to the initiator iSCSI layer.

Response with Response Fence MUST be delivered chronologically prior to all the "following" responses on the I\_T\_L nexus.

The "preceding" and "following" notions refer to the order of handoff of a response message from the target SCSI protocol layer to the target iSCSI layer.

#### 4.2.2.3.3. iSCSI Semantics with the Interface Model

Whenever the TaskReporting key (Section 12.23 "Task Reporting") is negotiated to ResponseFence or FastAbort for an iSCSI session and the Response Fence behavior is required for a SCSI response message, the target iSCSI layer MUST perform the actions described in this section for that session.

If it is a single-connection session, no special processing is required. The standard SCSI Response PDU build and dispatch process happens.

If it is a multi-connection session, the target iSCSI layer takes note of the last-sent and unacknowledged StatSN on each of the connections in the iSCSI session, and waits for an acknowledgement (NOP-In PDUs MAY be used to solicit acknowledgements as needed in order to accelerate this process) of each such StatSN to clear the fence. The SCSI response requiring Response Fence behavior MUST NOT be sent to the initiator before acknowledgements are received for each of the unacknowledged StatSNs.

The target iSCSI layer must wait for an acknowledgement of the SCSI Response PDU that carried the SCSI response requiring the Response Fence behavior. The fence MUST be considered cleared only after receiving the acknowledgement.

All further status processing for the LU is resumed only after clearing the fence. If any new responses for the I\_T\_L nexus are received from the SCSI layer before the fence is cleared, those Response PDUs MUST be held and queued at the iSCSI layer until the fence is cleared.

#### 4.2.2.3.4. Current List of Fenced Response Use Cases



This section lists the fenced response use cases that iSCSI implementations MUST comply with. However, this is not an exhaustive enumeration. It is expected that as SCSI protocol specifications evolve, the specifications will specify when response fencing is required on a case-by-case basis.

Whenever the TaskReporting key (Section 13.23) is negotiated to ResponseFence or FastAbort for an iSCSI session, the target iSCSI layer MUST assume that the Response Fence is required for the following SCSI completion messages:

1. The first completion message carrying the UA after the multi-task abort on issuing and third-party sessions. See Section 4.2.3.2 for related TMF discussion.
2. The TMF Response carrying the multi-task TMF Response on the issuing session.
3. The completion message indicating ACA establishment on the issuing session.
4. The first completion message carrying the ACA ACTIVE status after ACA establishment on issuing and third-party sessions.
5. The TMF Response carrying the Clear ACA response on the issuing session.
6. The response to a PERSISTENT RESERVE OUT/PREEMPT AND ABORT command.

Note:

- Due to the absence of ACA-related fencing requirements in [RFC3720], initiator implementations SHOULD NOT use ACA on multi-connection iSCSI sessions with targets complying only with [RFC3720].
- Initiators that want to employ ACA on multi-connection iSCSI sessions SHOULD first assess response-fencing behavior via negotiating for ResponseFence or FastAbort values for the TaskReporting (Section 13.23) key.

#### 4.2.2.4. Data Sequencing

Data and R2T PDUs transferred as part of some command execution MUST be sequenced. The DataSN field is used for data sequencing. For input (read) data PDUs, DataSN starts with 0 for the first data PDU of an input command and advances by 1 for each subsequent data PDU. For output data PDUs, DataSN starts with 0 for the first data PDU of a sequence (the initial unsolicited sequence or any data PDU sequence issued to satisfy an R2T) and advances by 1 for each subsequent data PDU. R2Ts are also sequenced per command. For example, the first R2T has an R2TSN of 0 and advances by 1 for each subsequent R2T. For bidirectional commands, the target uses the DataSN/R2TSN to sequence Data-In and R2T PDUs in one continuous sequence (undifferentiated). Unlike command and status, data PDUs and R2Ts are not acknowledged by a field in regular outgoing PDUs. Data-In PDUs can be acknowledged on demand by a special form of the SNACK PDU. Data and R2T PDUs are implicitly acknowledged by status for the command. The DataSN/R2TSN field enables the initiator to detect missing data or R2T PDUs.

For any read or bidirectional command, a target MUST issue less than  $2^{32}$  combined R2T and Data-In PDUs. Any output data sequence MUST contain less than  $2^{32}$  Data-Out PDUs.

#### 4.2.3. iSCSI Task Management

##### 4.2.3.1. Task Management Overview

iSCSI task management features allow an initiator to control the active iSCSI tasks on an operational iSCSI session that it has with an iSCSI target. Section 11.5 defines the task management function types that this specification defines - ABORT TASK, ABORT TASK SET, CLEAR ACA, CLEAR TASK SET, LOGICAL UNIT RESET, TARGET WARM RESET, TARGET COLD RESET, and TASK REASSIGN.

Out of these function types, ABORT TASK and TASK REASSIGN functions manage a single active task, whereas ABORT TASK SET, CLEAR TASK SET, LOGICAL UNIT RESET, TARGET WARM RESET and TARGET COLD RESET functions can each potentially affect multiple active tasks.

## 4.2.3.2. Notion of Affected Tasks

This section defines the notion of "affected tasks" in multi-task abort scenarios. Scope definitions in this section apply to both the Standard Multi-task Abort semantics (Section 4.2.3.3) and the FastAbort Multi-task Abort semantics behavior (Section 4.2.3.4).

**ABORT TASK SET:** All outstanding tasks for the I\_T\_L nexus identified by the LUN field in the ABORT TASK SET TMF Request PDU (Section 11.5).

**CLEAR TASK SET:** All outstanding tasks in the task set for the LU identified by the LUN field in the CLEAR TASK SET TMF Request PDU. See [SPC3] for the definition of a "task set".

**LOGICAL UNIT RESET:** All outstanding tasks from all initiators for the LU identified by the LUN field in the LOGICAL UNIT RESET Request PDU.

**TARGET WARM RESET/TARGET COLD RESET:** All outstanding tasks from all initiators across all LUs to which the TMF-issuing session has access on the SCSI target device hosting the iSCSI session.

**Usage:** An "ABORT TASK SET TMF Request PDU" in the preceding text is an iSCSI TMF Request PDU with the "Function" field set to "ABORT TASK SET" as defined in Section 11.5. Similar usage is employed for other scope descriptions.

## 4.2.3.3. Standard Multi-task Abort Semantics

All iSCSI implementations MUST support the protocol behavior defined in this section as the default behavior. The execution of ABORT TASK SET, CLEAR TASK SET, LOGICAL UNIT RESET, TARGET WARM RESET, and TARGET COLD RESET TMF Requests consists of the following sequence of actions in the specified order on the specified party.

The initiator iSCSI layer:

- a. MUST continue to respond to each TTT received for the affected tasks.
- b. SHOULD process any responses received for affected tasks in the normal fashion. This is acceptable because the

responses are guaranteed to have been sent prior to the TMF response.

- c. SHOULD receive the TMF Response concluding all the tasks in the set of affected tasks unless the initiator has done something (e.g., LU reset, connection drop) that may prevent the TMF Response from being sent or received. The initiator MUST thus conclude all affected tasks as part of this step in either case, and MUST discard any TMF Response received after the affected tasks are concluded.

The target iSCSI layer:

- a. MUST wait for responses on currently valid target-transfer tags of the affected tasks from the issuing initiator. MAY wait for responses on currently valid target-transfer tags of the affected tasks from third-party initiators.
- b. MUST wait (concurrent with the wait in Step a) for all commands of the affected tasks to be received based on the CmdSN ordering. SHOULD NOT wait for new commands on third-party affected sessions -- only the instantiated tasks have to be considered for the purpose of determining the affected tasks. In the case of target-scoped requests (i.e., TARGET WARM RESET and TARGET COLD RESET), all of the commands that are not yet received on the issuing session in the command stream however can be considered to have been received with no command waiting period -- i.e., the entire CmdSN space up to the CmdSN of the task management function can be "plugged".
- c. MUST propagate the TMF request to and receive the response from the target SCSI layer.
- d. MUST provide the Response Fence behavior for the TMF Response on the issuing session as specified in Section 4.2.2.3.2.
- e. MUST provide the Response Fence behavior on the first post-TMF Response on third-party sessions as specified in Section 4.2.2.3.3. If some tasks originate from non-iSCSI I\_T\_L nexuses, then the means by which the target ensures that all affected tasks have returned their status to the initiator are defined by the specific non-iSCSI transport protocol(s).

Technically, the TMF servicing is complete in Step d. Data transfers corresponding to terminated tasks may however still be in progress on third-party iSCSI sessions even at the end of Step e. The TMF Response MUST NOT be sent by the target iSCSI layer before the end of Step d, and MAY be sent at the end of Step d despite these outstanding data transfers until after Step e.

#### 4.2.3.4. FastAbort Multi-task Abort Semantics

Protocol behavior defined in this section MUST be implemented by all iSCSI implementations complying with this document. Protocol behavior defined in this section MUST be exhibited by iSCSI implementations on an iSCSI session when they negotiate the TaskReporting (Section 13.23) key to "FastAbort" on that session. The execution of ABORT TASK SET, CLEAR TASK SET, LOGICAL UNIT RESET, TARGET WARM RESET, and TARGET COLD RESET TMF Requests consists of the following sequence of actions in the specified order on the specified party.

The initiator iSCSI layer:

- a. MUST NOT send any more Data-Out PDUs for affected tasks on the issuing connection of the issuing iSCSI session once the TMF is sent to the target.
- b. SHOULD process any responses received for affected tasks in the normal fashion. This is acceptable because the responses are guaranteed to have been sent prior to the TMF response.
- c. MUST respond to each Async Message PDU with FAST\_ABORT AsyncEvent as defined in Section 11.9.
- d. MUST treat the TMF response as terminating all affected tasks for which responses have not been received, and MUST discard any responses for affected tasks received after the TMF response is passed to the SCSI layer (although the semantics defined in this section ensure that such an out-of-order scenario will never happen with a compliant target implementation).

The target iSCSI layer:

- a. MUST wait for all commands of the affected tasks to be received based on the CmdSN ordering on the issuing session. SHOULD NOT wait for new commands on third-party

affected sessions - only the instantiated tasks have to be considered for the purpose of determining the affected tasks. In the case of target-scoped requests (i.e., TARGET WARM RESET and TARGET COLD RESET), all the commands that are not yet received on the issuing session in the command stream can be considered to have been received with no command waiting period -- i.e., the entire CmdSN space up to the CmdSN of the task management function can be "plugged".

- b. MUST propagate the TMF request to and receive the response from the target SCSI layer.
- c. MUST leave all active "affected TTTs" (i.e., active TTTs associated with affected tasks) valid.
- d. MUST send an Asynchronous Message PDU with AsyncEvent=5 (Section 11.9) on:
  - i) each connection of each third-party session to which at least one affected task is allegiant if TaskReporting=FastAbort is operational on that third-party session, and,
  - ii) each connection except the issuing connection of the issuing session that has at least one allegiant affected task.

If there are multiple affected LUs (say, due to a target reset), then one Async Message PDU MUST be sent for each such LU on each connection that has at least one allegiant affected task. The LUN field in the Asynchronous Message PDU MUST be set to match the LUN for each such LU.
- e. MUST address the Response Fence flag on the TMF Response on the issuing session as defined in Section 4.2.2.3.3.
- f. MUST address the Response Fence flag on the first post-TMF Response on third-party sessions as defined in Section 4.2.2.3.3. If some tasks originate from non-iSCSI I\_T\_L nexuses, then the means by which the target ensures that all affected tasks have returned their status to the initiator are defined by the specific non-iSCSI transport protocol(s).
- g. MUST free up the affected TTTs (and STags, if applicable) and the corresponding buffers, if any, once it receives each associated NOP-Out acknowledgement that the initiator generated in response to each Async Message.

Technically, the TMF servicing is complete in Step e. Data transfers corresponding to terminated tasks may however still be in progress even at the end of Step f. A TMF Response MUST NOT be sent by the target iSCSI layer before the end of Step e, and MAY be sent at the end of Step e despite these outstanding Data transfers until Step g. Step g specifies an event to free up any such resources that may have been reserved to support outstanding data transfers.

#### 4.2.3.5. Affected Tasks Shared across Standard and FastAbort Sessions

If an iSCSI target implementation is capable of supporting TaskReporting=FastAbort functionality (Section 13.23), it may end up in a situation where some sessions have TaskReporting=RFC3720 operational (RFC 3720 sessions) while some other sessions have TaskReporting=FastAbort operational (FastAbort sessions) even while accessing a shared set of affected tasks (Section 4.2.3.2). If the issuing session is an RFC 3720 session, the iSCSI target implementation is FastAbort-capable, and the third-party affected session is a FastAbort session, the following behavior SHOULD be exhibited by the iSCSI target layer:

- a. Between Steps c and d of the target behavior in Section 4.2.3.3, send an Asynchronous Message PDU with AsyncEvent=5 (Section 11.9) on each connection of each third-party session to which at least one affected task is allegiant. If there are multiple affected LUs, then send one Async Message PDU for each such LU on each connection that has at least one allegiant affected task. When sent, the LUN field in the Asynchronous Message PDU MUST be set to match the LUN for each such LU.
- b. After Step e of the target behavior in Section 4.2.3.3, free up the affected TTTs (and STags, if applicable) and the corresponding buffers, if any, once each associated NOP-Out acknowledgement is received that the third-party initiator generated in response to each Async Message sent in Step a.

If the issuing session is a FastAbort session, the iSCSI target implementation is FastAbort-capable, and the third-party affected session is an RFC 3720 session, the following behavior MUST be

exhibited by the iSCSI target layer: Asynchronous Message PDUs MUST NOT be sent on the third-party session to prompt the FastAbort behavior.

If the third-party affected session is a FastAbort session and the issuing session is a FastAbort session, the initiator in the third-party role MUST respond to each Async Message PDU with AsyncEvent=5 as defined in Section 11.9. Note that an initiator MAY thus receive these Async Messages on a third-party affected session even if the session is a single-connection session.

#### 4.2.3.6. Rationale behind the FastAbort Semantics

There are fundamentally three basic objectives behind the semantics specified in Sections 4.2.3.3 and 4.2.3.4.

1. Maintaining an ordered command flow I\_T nexus abstraction to the target SCSI layer even with multi-connection sessions.
  - Target iSCSI processing of a TMF request must maintain the single flow illusion. Target behavior in Step b of Section 4.2.3.3 and Step a of Section 4.2.3.4 correspond to this objective.
2. Maintaining a single ordered response flow I\_T nexus abstraction to the initiator SCSI layer even with multi-connection sessions when one response (i.e., TMF response) could imply the status of other unfinished tasks from the initiator's perspective.
  - The target must ensure that the initiator does not see "old" task responses (that were placed on the wire chronologically earlier than the TMF Response) after seeing the TMF response. The target behavior in Step d of Section 4.2.3.3 and Step e of Section 4.2.3.4 correspond to this objective.
  - Whenever the result of a TMF action is visible across multiple I\_T\_L nexuses, [SAM2] requires the SCSI device server to trigger a UA on each of the other I\_T\_L nexuses. Once an initiator is notified of such an UA, the application client on the receiving initiator is required to clear its task state (clause 5.5 in [SAM2]) for the affected tasks. It would thus be inappropriate



to deliver a SCSI Response for a task after the task state is cleared on the initiator, i.e., after the UA is notified. The UA notification contained in the first SCSI Response PDU on each affected Third-party I\_T\_L nexus after the TMF action thus MUST NOT pass the affected task responses on any of the iSCSI sessions accessing the LU. The target behavior in Step e of Section 4.2.3.3 and Step f of Section 4.2.3.4 correspond to this objective.

3. Draining all active TTTs corresponding to affected tasks in a deterministic fashion.
  - Data-Out PDUs with stale TTTs arriving after the tasks are terminated can create a buffer management problem even for traditional iSCSI implementations, and is fatal for the connection for iSCSI/iSER implementations. Either the termination of affected tasks should be postponed until the TTTs are retired (as in Step a of Section 4.2.3.3), or the TTTs and the buffers should stay allocated beyond task termination to be deterministically freed up later (as in Steps c and g of Section 4.2.3.4).

The only other notable optimization is the plugging. If all tasks on an I\_T nexus will be aborted anyway (as with a target reset), there is no need to wait to receive all commands to plug the CmdSN holes. The target iSCSI layer can simply plug all missing CmdSN slots and move on with TMF processing. The first objective (maintaining a single ordered command flow) is still met with this optimization because the target SCSI layer only sees ordered commands.

#### 4.2.4. iSCSI Login

The purpose of the iSCSI login is to enable a TCP connection for iSCSI use, authentication of the parties, negotiation of the session's parameters and marking of the connection as belonging to an iSCSI session.

A session is used to identify to a target all the connections with a given initiator that belong to the same I\_T nexus. (For more

details on how a session relates to an I\_T nexus, see section 4.4.2).

The targets listen on a well-known TCP port or other TCP port for incoming connections. The initiator begins the login process by connecting to one of these TCP ports.

As part of the login process, the initiator and target SHOULD authenticate each other and MAY set a security association protocol for the session. This can occur in many different ways and is subject to negotiation.

To protect the TCP connection, an IPsec security association MAY be established before the Login request. For information on using IPsec security for iSCSI see Chapter 8 and [RFC3723].

The iSCSI Login Phase is carried through Login requests and responses. Once suitable authentication has occurred and operational parameters have been set, the session transitions to Full Feature Phase and the initiator may start to send SCSI commands. The security policy for whether, and by what means, a target chooses to authorize an initiator is beyond the scope of this document. For a more detailed description of the Login Phase, see Chapter 5.

The login PDU includes the ISID part of the session ID (SSID). The target portal group that services the login is implied by the selection of the connection endpoint. For a new session, the TSIH is zero. As part of the response, the target generates a TSIH.

During session establishment, the target identifies the SCSI initiator port (the "I" in the "I\_T nexus") through the value pair (InitiatorName, ISID). We describe InitiatorName later in this section. Any persistent state (e.g., persistent reservations) on the target that is associated with a SCSI initiator port is identified based on this value pair. Any state associated with the SCSI target port (the "T" in the "I\_T nexus") is identified externally by the TargetName and portal group tag (see Section 4.4.1). ISID is subject to reuse restrictions because it is used to identify a persistent state (see Section 4.4.3).

Before the Full Feature Phase is established, only Login Request and Login Response PDUs are allowed. Login requests and responses MUST be used exclusively during Login. On any connection, the login phase MUST immediately follow TCP connection establishment and a subsequent Login Phase MUST NOT occur before tearing down a connection.

A target receiving any PDU except a Login request before the Login phase is started MUST immediately terminate the connection on which the PDU was received. Once the Login phase has started, if the target receives any PDU except a Login request, it MUST send a Login reject (with Status "invalid during login") and then disconnect. If the initiator receives any PDU except a Login response, it MUST immediately terminate the connection.

#### 4.2.5. iSCSI Full Feature Phase

Once the initiator is authorized to do so, the iSCSI session is in the iSCSI Full Feature Phase. A session is in Full Feature Phase after successfully finishing the Login Phase on the first (leading) connection of a session. A connection is in Full Feature Phase if the session is in Full Feature Phase and the connection login has completed successfully. An iSCSI connection is not in Full Feature Phase

when it does not have an established transport connection,  
OR  
when it has a valid transport connection, but a successful login was not performed or the connection is currently logged out.

In a normal Full Feature Phase, the initiator may send SCSI commands and data to the various LUs on the target by encapsulating them in iSCSI PDUs that go over the established iSCSI session.

##### 4.2.5.1. Command Connection Allegiance

For any iSCSI request issued over a TCP connection, the corresponding response and/or other related PDU(s) MUST be sent over the same connection. We call this "connection allegiance". If the original connection fails before the command is completed, the

connection allegiance of the command may be explicitly reassigned to a different transport connection as described in detail in Section 6.2 "Retry and Reassign in Recovery".

Thus, if an initiator issues a READ command, the target MUST send the requested data, if any, followed by the status to the initiator over the same TCP connection that was used to deliver the SCSI command. If an initiator issues a WRITE command, the initiator MUST send the data, if any, for that command over the same TCP connection that was used to deliver the SCSI command. The target MUST return Ready To Transfer (R2T), if any, and the status over the same TCP connection that was used to deliver the SCSI command. Retransmission requests (SNACK PDUs) and the data and status that they generate MUST also use the same connection.

However, consecutive commands that are part of a SCSI linked command-chain task (see [SAM2]) MAY use different connections. Connection allegiance is strictly per-command and not per-task. During the iSCSI Full Feature Phase, the initiator and target MAY interleave unrelated SCSI commands, their SCSI Data, and responses over the session.

#### 4.2.5.2. Data Transfer Overview

Outgoing SCSI data (initiator to target user data or command parameters) is sent as either solicited data or unsolicited data. Solicited data are sent in response to R2T PDUs. Unsolicited data can be sent as part of an iSCSI command PDU ("immediate data") or in separate iSCSI data PDUs.

Immediate data are assumed to originate at offset 0 in the initiator SCSI write-buffer (outgoing data buffer). All other Data PDUs have the buffer offset set explicitly in the PDU header.

An initiator may send unsolicited data up to FirstBurstLength as immediate (up to the negotiated maximum PDU length), in a separate PDU sequence or both. All subsequent data MUST be solicited. The maximum length of an individual data PDU or the immediate-part of the first unsolicited burst MAY be negotiated at login.

The maximum amount of unsolicited data that can be sent with a command is negotiated at login through the FirstBurstLength key. A

target MAY separately enable immediate data (through the ImmediateData key) without enabling the more general (separate data PDUs) form of unsolicited data (through the InitialR2T key).

Unsolicited data on write are meant to reduce the effect of latency on throughput (no R2T is needed to start sending data). In addition, immediate data is meant to reduce the protocol overhead (both bandwidth and execution time).

An iSCSI initiator MAY choose not to send unsolicited data, only immediate data or FirstBurstLength bytes of unsolicited data with a command. If any non-immediate unsolicited data is sent, the total unsolicited data MUST be either FirstBurstLength, or all of the data if the total amount is less than the FirstBurstLength.

It is considered an error for an initiator to send unsolicited data PDUs to a target that operates in R2T mode (only solicited data are allowed). It is also an error for an initiator to send more unsolicited data, whether immediate or as separate PDUs, than FirstBurstLength.

An initiator MUST honor an R2T data request for a valid outstanding command (i.e., carrying a valid Initiator Task Tag) and deliver all the requested data provided the command is supposed to deliver outgoing data and the R2T specifies data within the command bounds. The initiator action is unspecified for receiving an R2T request that specifies data, all or part, outside of the bounds of the command.

A target SHOULD NOT silently discard data and then request retransmission through R2T. Initiators SHOULD NOT keep track of the data transferred to or from the target (scoreboarding). SCSI targets perform residual count calculation to check how much data was actually transferred to or from the device by a command. This may differ from the amount the initiator sent and/or received for reasons such as retransmissions and errors. Read or bidirectional commands implicitly solicit the transmission of the entire amount of data covered by the command. SCSI data packets are matched to their corresponding SCSI commands by using tags specified in the protocol.

In addition, iSCSI initiators and targets MUST enforce some ordering rules. When unsolicited data is used, the order of the unsolicited data on each connection MUST match the order in which the commands on that connection are sent. Command and unsolicited data PDUs may be interleaved on a single connection as long as the ordering requirements of each are maintained (e.g., command N+1 MAY be sent before the unsolicited Data-Out PDUs for command N, but the unsolicited Data-Out PDUs for command N MUST precede the unsolicited Data-Out PDUs of command N+1). A target that receives data out of order MAY terminate the session.

#### 4.2.5.3. Tags and Integrity Checks

Initiator tags for pending commands are unique initiator-wide for a session. Target tags are not strictly specified by the protocol. It is assumed that target tags are used by the target to tag (alone or in combination with the LUN) the solicited data. Target tags are generated by the target and "echoed" by the initiator. These mechanisms are designed to accomplish efficient data delivery along with a large degree of control over the data flow.

As the Initiator Task Tag is used to identify a task during its execution the iSCSI initiator and target MUST verify that all other fields used in task related PDUs have values that are consistent with the values used at the task instantiation based on Initiator Task Tag (e.g., the LUN used in an R2T PDU MUST be the same as the one used in the SCSI command PDU used to instantiate the task). Using inconsistent field values is considered a protocol error.

#### 4.2.5.4. Task Management

SCSI task management assumes that individual tasks and task groups can be aborted solely based on the task tags (for individual tasks) or the timing of the task management command (for task groups) and that the task management action is executed synchronously - i.e., no message involving an aborted task will be seen by the SCSI initiator after receiving the task management response. In iSCSI initiators and targets interact asynchronously over several connections. iSCSI specifies the protocol mechanism and implementation requirements needed to present a synchronous view while using an asynchronous infrastructure.

#### 4.2.6. iSCSI Connection Termination

An iSCSI connection may be terminated by use of a transport connection shutdown or a transport reset. Transport reset is assumed to be an exceptional event.

Graceful TCP connection shutdowns are done by sending TCP FINs. A graceful transport connection shutdown SHOULD only be initiated by either party when the connection is not in iSCSI Full Feature Phase. A target MAY terminate a Full Feature Phase connection on internal exception events, but it SHOULD announce the fact through an Asynchronous Message PDU. Connection termination with outstanding commands may require recovery actions.

If a connection is terminated while in Full Feature Phase, connection cleanup (see section 7) is required prior to recovery. By doing connection cleanup before starting recovery, the initiator and target will avoid receiving stale PDUs after recovery.

#### 4.2.7. iSCSI Names

Both targets and initiators require names for the purpose of identification. In addition, names enable iSCSI storage resources to be managed regardless of location (address). An iSCSI node name is also the SCSI device name contained in the iSCSI Node. The iSCSI name of a SCSI device is the principal object used in authentication of targets to initiators and initiators to targets. This name is also used to identify and manage iSCSI storage resources.

iSCSI names must be unique within the operation domain of the end user. However, because the operation domain of an IP network is potentially worldwide, the iSCSI name formats are architected to be worldwide unique. To assist naming authorities in the construction of worldwide unique names, iSCSI provides three name formats for different types of naming authorities.

iSCSI names are associated with iSCSI nodes, and not iSCSI network adapter cards, to ensure that the replacement of network adapter

cards does not require reconfiguration of all SCSI and iSCSI resource allocation information.

Some SCSI commands require that protocol-specific identifiers be communicated within SCSI CDBs. See SCSI for the definition of the SCSI port name/identifier for iSCSI ports.

An initiator may discover the iSCSI Target Names to which it has access, along with their addresses, using the SendTargets text request, or other techniques discussed in [RFC3721].

iSCSI equipment that needs discovery functions beyond SendTargets SHOULD implement iSNS (see [RFC4171]) for extended discovery management capabilities and interoperability. Although [RFC3721] implies an SLP implementation requirement, SLP has not been widely implemented or deployed for use with iSCSI in practice. iSCSI implementations therefore SHOULD NOT rely on SLP-based discovery interoperability.

#### 4.2.7.1. iSCSI Name Properties

Each iSCSI node, whether it is an initiator or target or both, MUST have an iSCSI name.

Initiators and targets MUST support the receipt of iSCSI names of up to the maximum length of 223 bytes.

The initiator MUST present both its iSCSI Initiator Name and the iSCSI Target Name to which it wishes to connect in the first login request of a new session or connection. The only exception is if a discovery session (see Section 2.3 iSCSI Session Types) is to be established. In this case, the iSCSI Initiator Name is still required, but the iSCSI Target Name MAY be omitted.

iSCSI names have the following properties:

- iSCSI names are globally unique. No two initiators or targets can have the same name.

- iSCSI names are permanent. An iSCSI initiator node or target node has the same name for its lifetime.



iSCSI names do not imply a location or address. An iSCSI initiator or target can move, or have multiple addresses. A change of address does not imply a change of name.

iSCSI names do not rely on a central name broker; the naming authority is distributed.

iSCSI names support integration with existing unique naming schemes.

iSCSI names rely on existing naming authorities. iSCSI does not create any new naming authority.

The encoding of an iSCSI name has the following properties:

iSCSI names have the same encoding method regardless of the underlying protocols.

iSCSI names are relatively simple to compare. The algorithm for comparing two iSCSI names for equivalence does not rely on an external server.

iSCSI names are composed only of displayable characters. iSCSI names allow the use of international character sets but are not case sensitive. No whitespace characters are used in iSCSI names.

iSCSI names may be transported using both binary and ASCII-based protocols.

An iSCSI name really names a logical software entity, and is not tied to a port or other hardware that can be changed. For instance, an initiator name should name the iSCSI initiator node, not a particular NIC or HBA. When multiple NICs are used, they should generally all present the same iSCSI initiator name to the targets, because they are simply paths to the same SCSI layer. In most operating systems, the named entity is the operating system image.

Similarly, a target name should not be tied to hardware interfaces that can be changed. A target name should identify the logical target and must be the same for the target regardless of the physical portion being addressed. This assists iSCSI initiators in determining that the two targets it has discovered are really two paths to the same target.

The iSCSI name is designed to fulfill the functional requirements for Uniform Resource Names (URN) [RFC1737]. For example, it is required that the name have a global scope, be independent of address or location, and be persistent and globally unique. Names must be extensible and scalable with the use of naming authorities. The name encoding should be both human and machine readable. See [RFC1737] for further requirements.

#### 4.2.7.2. iSCSI Name Encoding

An iSCSI name MUST be a UTF-8 encoding of a string of Unicode characters with the following properties:

- It is in Normalization Form C (see "Unicode Normalization Forms" [UNICODE]).
- It only contains characters allowed by the output of the iSCSI stringprep template (described in [RFC3722]).
- The following characters are used for formatting iSCSI names:
  - dash ('-'=U+002d)
  - dot ('.'=U+002e)
  - colon (':'=U+003a)
- The UTF-8 encoding of the name is not larger than 223 bytes.

The stringprep process is described in [RFC3454]; iSCSI's use of the stringprep process is described in [RFC3722]. Stringprep is a method designed by the Internationalized Domain Name (IDN) working group to translate human-typed strings into a format that can be compared as opaque strings. Strings MUST NOT include punctuation, spacing, diacritical marks, or other characters that could get in the way of readability. The stringprep process also converts strings into equivalent strings of lower-case characters.

The stringprep process does not need to be implemented if the names are only generated using numeric and lower-case (any character set) alphabetic characters.

Once iSCSI names encoded in UTF-8 are "normalized" they may be safely compared byte-for-byte.

#### 4.2.7.3. iSCSI Name Structure

An iSCSI name consists of two parts--a type designator followed by a unique name string.

iSCSI uses three existing naming authorities in constructing globally unique iSCSI names. Type designator in an iSCSI name indicates the naming authority on which the name is based. The three iSCSI name formats are the following:

- iSCSI-Qualified Name: it is based on domain names to identify a naming authority,
- NAA format Name: it is based on a naming format defined by [FC-FS] for constructing globally unique identifiers, referred to as the Network Address Authority (NAA), and,
- EUI format Name: it is based on EUI names where the IEEE Registration Authority assists in the formation of worldwide unique names (EUI-64 format).

The corresponding type designator strings currently defined are:

iqn. - iSCSI Qualified name

naa. - Remainder of the string is an INCITS T11-defined Network Address Authority identifier, in ASCII-encoded hexadecimal.

eui. - Remainder of the string is an IEEE EUI-64 identifier, in ASCII-encoded hexadecimal.

These three naming authority designators were considered sufficient at the time of writing this document. The creation of

additional naming type designators for iSCSI may be considered by the IETF and detailed in separate RFCs.

The following table summarizes the current SCSI transport protocols and their naming formats.

SCSI Transport Protocol	Naming Format		
	EUI-64	NAA	IQN
iSCSI (Internet SCSI)	X	X	X
FCP (Fibre Channel)		X	
SAS (Serial Attached SCSI)		X	
SRP (for InfiniBand)	X		

#### 4.2.7.4. Type "iqn." (iSCSI Qualified Name)

This iSCSI name type can be used by any organization that owns a domain name. This naming format is useful when an end user or service provider wishes to assign iSCSI names for targets and/or initiators.

To generate names of this type, the person or organization generating the name must own a registered domain name. This domain name does not have to be active, and does not have to resolve to an address; it just needs to be reserved to prevent others from generating iSCSI names using the same domain name.

Since a domain name can expire, be acquired by another entity, or may be used to generate iSCSI names by both owners, the domain name must be additionally qualified by a date during which the naming authority owned the domain name. A date code is provided as part of the "iqn." format for this reason.

The iSCSI qualified name string consists of:

- The string "iqn.", used to distinguish these names from "eui." formatted names.
- A date code, in yyyy-mm format. This date MUST be a date during which the naming authority owned the domain name used in this format, and SHOULD be the first month in which the domain name was owned by this naming authority at 00:01 GMT of the first day of the month. This date code uses the Gregorian calendar. All four digits in the year must be present. Both digits of the month must be present, with January == "01" and December == "12". The dash must be included.
- A dot "."
- The reversed domain name of the naming authority (person or organization) creating this iSCSI name.
- An optional, colon (:) prefixed, string within the character set and length boundaries that the owner of the domain name deems appropriate. This may contain product types, serial numbers, host identifiers, or software keys (e.g, it may include colons to separate organization boundaries). With the exception of the colon prefix, the owner of the domain name can assign everything after the reversed domain name as desired. It is the responsibility of the entity that is the naming authority to ensure that the iSCSI names it assigns are worldwide unique. For example, "Example Storage Arrays, Inc.", might own the domain name "example.com".

The following are examples of iSCSI qualified names that might be generated by "EXAMPLE Storage Arrays, Inc."

Type	Date	Naming Auth	String defined by "example.com" naming authority
iqn.2001-04.com.example:storage:diskarrays-sn-a8675309			
iqn.2001-04.com.example			
iqn.2001-04.com.example:storage.tapel.sys1.xyz			
iqn.2001-04.com.example:storage.disk2.sys1.xyz			

#### 4.2.7.5. Type "eui." (IEEE EUI-64 format)

The IEEE Registration Authority provides a service for assigning globally unique identifiers [EUI]. The EUI-64 format is used to build a global identifier in other network protocols. For example, Fibre Channel defines a method of encoding it into a WorldWideName. For more information on registering for EUI identifiers, see [OUI].

The format is "eui." followed by an EUI-64 identifier (16 ASCII-encoded hexadecimal digits).

Example iSCSI name:

Type	EUI-64 identifier (ASCII-encoded hexadecimal)
eui.02004567A425678D	

The IEEE EUI-64 iSCSI name format might be used when a manufacturer is already registered with the IEEE Registration Authority and uses EUI-64 formatted worldwide unique names for its products.

More examples of name construction are discussed in [RFC3721].

## 4.2.7.6. Type "naa." - Network Address Authority

The INCITS T11 Framing and Signaling Specification [FC-FS] defines a format called the Network Address Authority (NAA) format for constructing worldwide unique identifiers that use various identifier registration authorities. This identifier format is used by the Fibre Channel and SAS SCSI transport protocols. As FC and SAS constitute a large fraction of networked SCSI ports, the NAA format is a widely used format for SCSI transports. The objective behind iSCSI supporting a direct representation of an NAA-format name is to facilitate construction of a target device name that translates easily across multiple namespaces for a SCSI storage device containing ports served by different transports. More specifically, this format allows implementations wherein one NAA identifier can be assigned as the basis for the SCSI device name for a SCSI target with both SAS ports and iSCSI ports.

The iSCSI NAA naming format is "naa.", followed by an NAA identifier represented in ASCII-encoded hexadecimal digits.

An example of an iSCSI name with a 64-bit NAA value follows:

Type NAA identifier (ASCII-encoded hexadecimal)

```
+---++-----+
|  ||          |
naa.52004567BA64678D
```

An example of an iSCSI name with a 128-bit NAA value follows:

Type NAA identifier (ASCII-encoded hexadecimal)

```
+---++-----+
|  ||          |
naa.62004567BA64678D0123456789ABCDEF
```

The iSCSI NAA naming format might be used in an implementation when the infrastructure for generating NAA worldwide unique names is already in place because the device contains both SAS and iSCSI SCSI ports.

The NAA identifier formatted in an ASCII-hexadecimal representation has a maximum size of 32 characters (128 bit NAA

format). As a result, there is no issue with this naming format exceeding the maximum size for iSCSI node names.

#### 4.2.8. Persistent State

iSCSI does not require any persistent state maintenance across sessions. However, in some cases, SCSI requires persistent identification of the SCSI initiator port name (See Section 4.4.2 and Section 4.4.3).

iSCSI sessions do not persist through power cycles and boot operations.

All iSCSI session and connection parameters are re-initialized on session and connection creation.

Commands persist beyond connection termination if the session persists and command recovery within the session is supported. However, when a connection is dropped, command execution, as perceived by iSCSI (i.e., involving iSCSI protocol exchanges for the affected task), is suspended until a new allegiance is established by the 'task reassign' task management function. (See Section 10.5 "Task Management Function Request".)

#### 4.2.9. Message Synchronization and Steering

iSCSI presents a mapping of the SCSI protocol onto TCP. This encapsulation is accomplished by sending iSCSI PDUs of varying lengths. Unfortunately, TCP does not have a built-in mechanism for signaling message boundaries at the TCP layer. iSCSI overcomes this obstacle by placing the message length in the iSCSI message header. This serves to delineate the end of the current message as well as the beginning of the next message.

In situations where IP packets are delivered in order from the network, iSCSI message framing is not an issue and messages are processed one after the other. In the presence of IP packet reordering (i.e., frames being dropped), legacy TCP implementations store the "out of order" TCP segments in temporary buffers until the missing TCP segments arrive, upon which the data must be copied to the application buffers. In iSCSI, it is desirable to steer the SCSI data within these out of order TCP



segments into the pre-allocated SCSI buffers rather than store them in temporary buffers. This decreases the need for dedicated reassembly buffers as well as the latency and bandwidth related to extra copies.

Relying solely on the "message length" information from the iSCSI message header may make it impossible to find iSCSI message boundaries in subsequent TCP segments due to the loss of a TCP segment that contains the iSCSI message length. The missing TCP segment(s) must be received before any of the following segments can be steered to the correct SCSI buffers (due to the inability to determine the iSCSI message boundaries). Since these segments cannot be steered to the correct location, they must be saved in temporary buffers that must then be copied to the SCSI buffers.

Different schemes can be used to recover synchronization. The details of any such schemes are beyond this protocol specification, but it suffices to note that [RFC4297] provides an overview of the direct data placement problem on IP networks, and [RFC5046] specifies a protocol extension for iSCSI that facilitates this direct data placement objective.

Under normal circumstances (no PDU loss or data reception out of order), iSCSI data steering can be accomplished by using the identifying tag and the data offset fields in the iSCSI header in addition to the TCP sequence number from the TCP header. The identifying tag helps associate the PDU with a SCSI buffer address while the data offset and TCP sequence number are used to determine the offset within the buffer.

#### 4.2.9.1. Sync/Steering and iSCSI PDU Length

When a large iSCSI message is sent, the TCP segment(s) that contain the iSCSI header may be lost. The remaining TCP segment(s) up to the next iSCSI message must be buffered (in temporary buffers) because the iSCSI header that indicates to which SCSI buffers the data are to be steered was lost. To minimize the amount of buffering, it is recommended that the iSCSI PDU length be restricted to a small value (perhaps a few TCP segments in length). During login, each end of the iSCSI session specifies the maximum iSCSI PDU length it will accept.

#### 4.3. iSCSI Session Types

iSCSI defines two types of sessions:

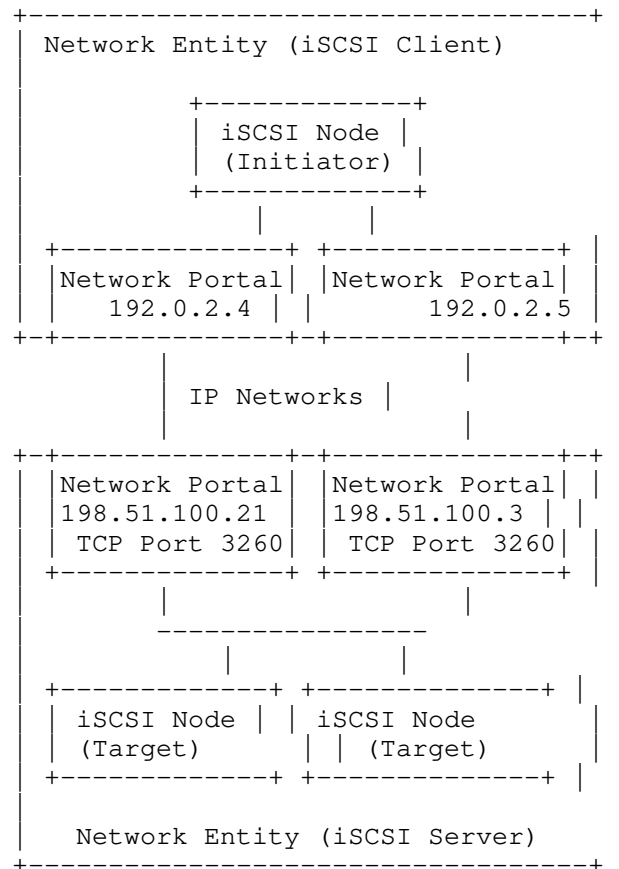
Normal operational session - an unrestricted session.

Discovery-session - a session only opened for target discovery. The target MUST ONLY accept text requests with the SendTargets key and a logout request with reason "close the session". All other requests MUST be rejected.

The session type is defined during login with key=value parameter in the login command.

#### 4.4. SCSI to iSCSI Concepts Mapping Model

The following diagram shows an example of how multiple iSCSI Nodes (targets in this case) can coexist within the same Network Entity and can share Network Portals (IP addresses and TCP ports). Other more complex configurations are also possible. For detailed descriptions of the components of these diagrams, see section 4.4.1.



#### 4.4.1. iSCSI Architecture Model

This section describes the part of the iSCSI architecture model that has the most bearing on the relationship between iSCSI and the SCSI Architecture Model.

**Network Entity** - represents a device or gateway that is accessible from the IP network. A Network Entity must have one or more Network Portals (see a following item), each of which can be used by some iSCSI Nodes (see the following

item) contained in that Network Entity to gain access to the IP network.

iSCSI Node - represents a single iSCSI initiator or iSCSI target or an instance of each. There are one or more iSCSI Nodes within a Network Entity. The iSCSI Node is accessible via one or more Network Portals (see item d). An iSCSI Node is identified by its iSCSI Name (see Section 4.2.7 and Section 13). The separation of the iSCSI Name from the addresses used by and for the iSCSI node allows multiple iSCSI nodes to use the same addresses, and the same iSCSI node to use multiple addresses.

An alias string may also be associated with an iSCSI Node. The alias allows an organization to associate a user friendly string with the iSCSI Name. However, the alias string is not a substitute for the iSCSI Name.

Network Portal - a component of a Network Entity that has a TCP/IP network address and that may be used by an iSCSI Node within that Network Entity for the connection(s) within one of its iSCSI sessions. In an initiator, it is identified by its IP address. In a target, it is identified by its IP address and its listening TCP port.

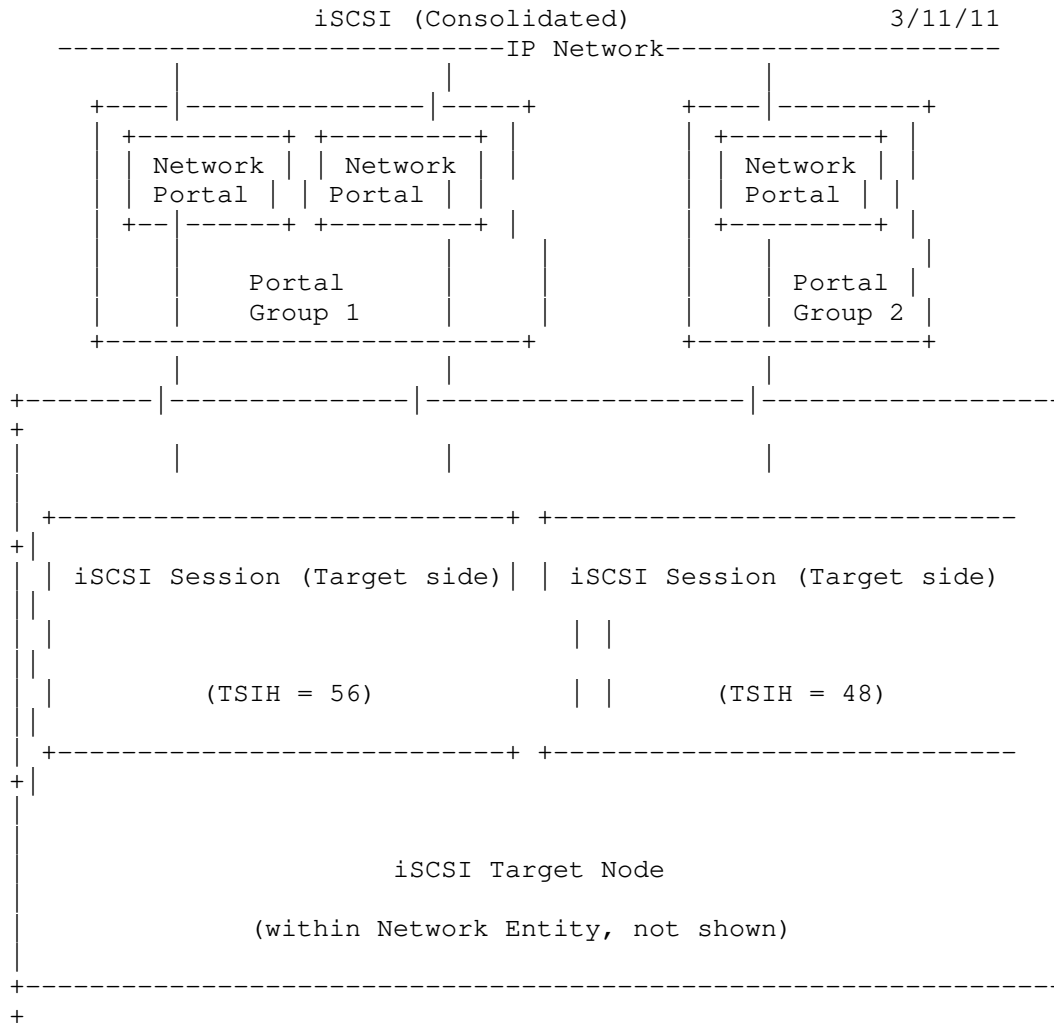
Portal Groups - iSCSI supports multiple connections within the same session; some implementations will have the ability to combine connections in a session across multiple Network Portals. A Portal Group defines a set of Network Portals within an iSCSI Node that collectively supports the capability of coordinating a session with connections that span these portals. Not all Network Portals within a Portal Group need to participate in every session connected through that Portal Group. One or more Portal Groups may provide access to an iSCSI Node. Each Network Portal, as utilized by a given iSCSI Node, belongs to exactly one portal group within that node. Portal Groups are identified within an iSCSI Node by a portal group tag, a simple unsigned-integer between 0 and 65535 (see Section 12.3 "SendTargets"). All Network Portals with the same portal

group tag in the context of a given iSCSI Node are in the same Portal Group.

Both iSCSI Initiators and iSCSI Targets have portal groups, though only the iSCSI Target Portal Groups are used directly in the iSCSI protocol (e.g., in SendTargets). For references to the Initiator Portal Groups, see Section 10.1.1.

Portals within a Portal Group should support similar session parameters, because they may participate in a common session.

The following diagram shows an example of one such configuration on a target and how a session that shares Network Portals within a Portal Group may be established.



#### 4.4.2. SCSI Architecture Model

This section describes the relationship between the SCSI Architecture Model [SAM2] and constructs of the SCSI device, SCSI port and I\_T nexus, and the iSCSI constructs described in Section 4.4.1.

This relationship implies implementation requirements in order to conform to the SAM2 model and other SCSI operational functions. These requirements are detailed in Section 4.4.3.

The following list outlines mappings of SCSI architectural elements to iSCSI.

SCSI Device - the SAM2 term for an entity that contains one or more SCSI ports that are connected to a service delivery subsystem and supports a SCSI application protocol. For example, a SCSI Initiator Device contains one or more SCSI Initiator Ports and zero or more application clients. A SCSI Target Device contains one or more SCSI Target Ports and one or more logical units. For iSCSI, the SCSI Device is the component within an iSCSI Node that provides the SCSI functionality. As such, there can be one SCSI Device, at most, within an iSCSI Node. Access to the SCSI Device can only be achieved in an iSCSI normal operational session (see Section 4.3). The SCSI Device Name is defined to be the iSCSI Name of the node and MUST be used in the iSCSI protocol.

SCSI Port - the SAM2 term for an entity in a SCSI Device that provides the SCSI functionality to interface with a service delivery subsystem or transport. For iSCSI, the definition of SCSI Initiator Port and SCSI Target Port are different.

SCSI Initiator Port: This maps to one endpoint of an iSCSI normal operational session (see Section 4.3). An iSCSI normal operational session is negotiated through the login process between an iSCSI initiator node and an iSCSI target node. At successful completion of this process, a SCSI Initiator Port is created within the SCSI Initiator Device. The SCSI Initiator Port Name and SCSI Initiator Port Identifier are both defined to be the iSCSI Initiator Name together with (a) a label that identifies it as an initiator port name/identifier and (b) the ISID portion of the session identifier.

SCSI Target Port: This maps to an iSCSI Target Portal

Group. The SCSI Target Port Name and the SCSI Target Port Identifier are both defined to be the iSCSI Target Name together with (a) a label that identifies it as a target port name/identifier and (b) the portal group tag.

The SCSI Port Name MUST be used in iSCSI. When used in SCSI parameter data, the SCSI port name MUST be encoded as:

- The iSCSI Name in UTF-8 format, followed by
- a comma separator (1 byte), followed by
- the ASCII character 'i' (for SCSI Initiator Port) or the ASCII character 't' (for SCSI Target Port) (1 byte), followed by
- a comma separator (1 byte), followed by
- a text encoding as a hex-constant (see Section 5.1 "Text Format") of the ISID (for SCSI initiator port) or the portal group tag (for SCSI target port) including the initial 0X or 0x and the terminating null (14 bytes).

The ASCII character 'i' or 't' is the label that identifies this port as either a SCSI Initiator Port or a SCSI Target Port.

I\_T nexus - a relationship between a SCSI Initiator Port and a SCSI Target Port, according to [SAM2]. For iSCSI, this relationship is a session, defined as a relationship between an iSCSI Initiator's end of the session (SCSI Initiator Port) and the iSCSI Target's Portal Group. The I\_T nexus can be identified by the conjunction of the SCSI port names or by the iSCSI session identifier SSID. iSCSI defines the I\_T nexus identifier to be the tuple (iSCSI Initiator Name + 'i' + ISID, iSCSI Target Name + 't' + Portal Group Tag).

NOTE: The I\_T nexus identifier is not equal to the session identifier (SSID).

#### 4.4.3. Consequences of the Model

This section describes implementation and behavioral requirements that result from the mapping of SCSI constructs to the iSCSI



constructs defined above. Between a given SCSI initiator port and a given SCSI target port, only one I\_T nexus (session) can exist. No more than one nexus relationship (parallel nexus) is allowed by [SAM2]. Therefore, at any given time, only one session with the same session identifier (SSID) can exist between a given iSCSI initiator node and an iSCSI target node.

These assumptions lead to the following conclusions and requirements:

**ISID RULE:** Between a given iSCSI Initiator and iSCSI Target Portal Group (SCSI target port), there can only be one session with a given value for ISID that identifies the SCSI initiator port. See Section 10.12.5 "ISID".

The structure of the ISID that contains a naming authority component (see Section 10.12.5 "ISID" and [RFC3721]) provides a mechanism to facilitate compliance with the ISID rule. (See Section 9.1.1 "Conservative Reuse of ISIDs".)

The iSCSI Initiator Node should manage the assignment of ISIDs prior to session initiation. The "ISID RULE" does not preclude the use of the same ISID from the same iSCSI Initiator with different Target Portal Groups on the same iSCSI target or on other iSCSI targets (see Section 9.1.1 "Conservative Reuse of ISIDs"). Allowing this would be analogous to a single SCSI Initiator Port having relationships (nexus) with multiple SCSI target ports on the same SCSI target device or SCSI target ports on other SCSI target devices. It is also possible to have multiple sessions with different ISIDs to the same Target Portal Group. Each such session would be considered to be with a different initiator even when the sessions originate from the same initiator device. The same ISID may be used by a different iSCSI initiator because it is the iSCSI Name together with the ISID that identifies the SCSI Initiator Port.

**NOTE:** A consequence of the ISID RULE and the specification for the I\_T nexus identifier is that two nexus with the same identifier should never exist at the same time.

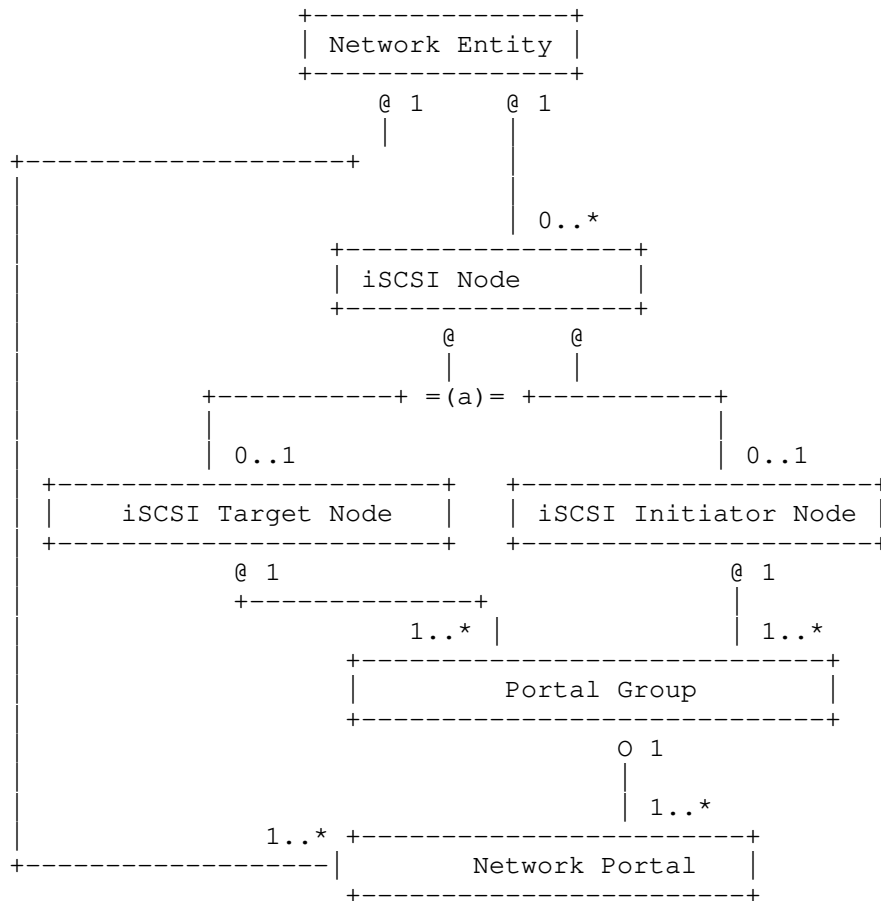
TSIH RULE: The iSCSI Target selects a non-zero value for the TSIH at session creation (when an initiator presents a 0 value at Login). After being selected, the same TSIH value MUST be used whenever initiator or target refers to the session and a TSIH is required.

#### 4.4.3.1. I\_T Nexus State

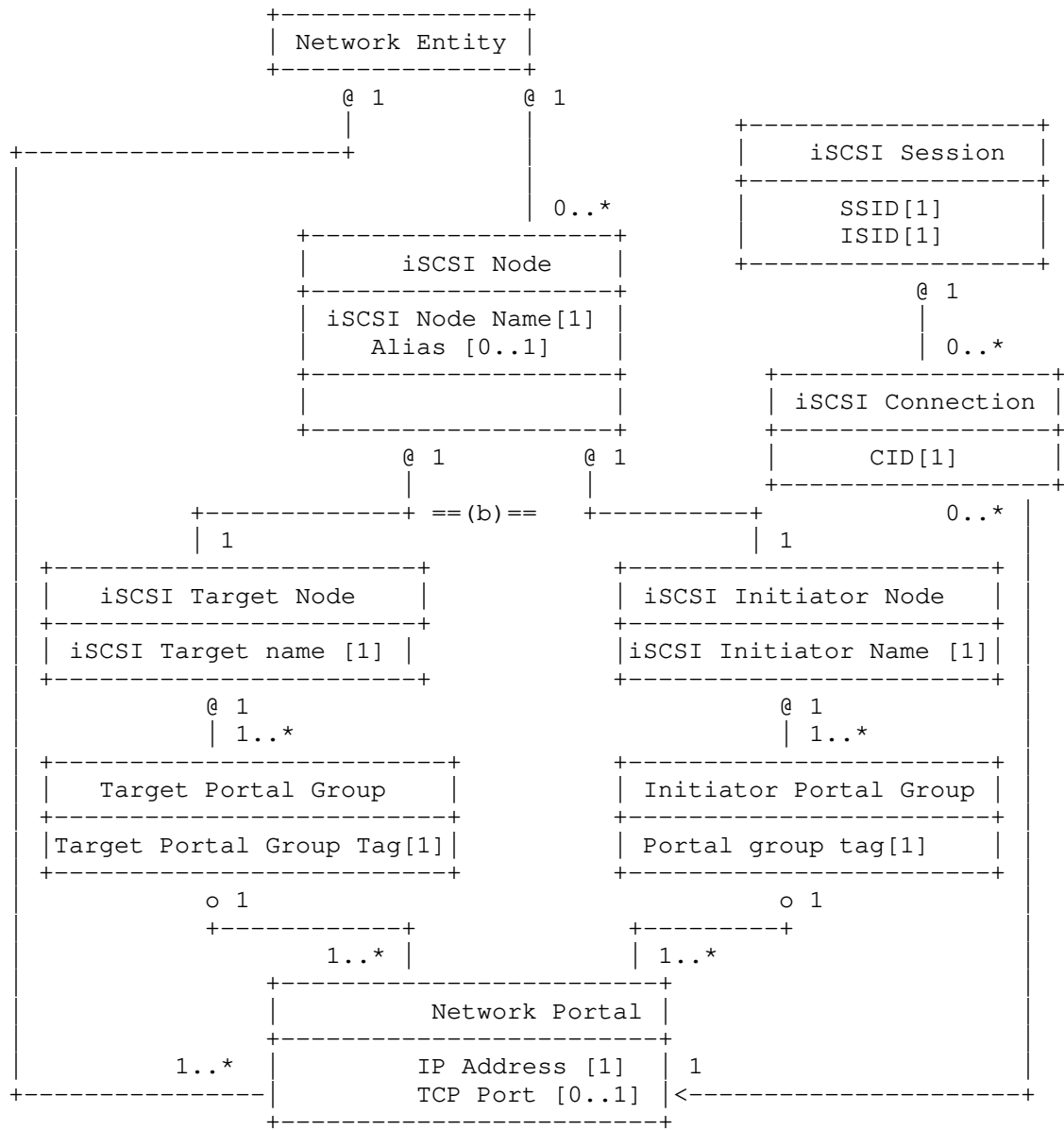
Certain nexus relationships contain an explicit state (e.g., initiator-specific mode pages) that may need to be preserved by the device server [SAM2] in a logical unit through changes or failures in the iSCSI layer (e.g., session failures). In order for that state to be restored, the iSCSI initiator should reestablish its session (re-login) to the same Target Portal Group using the previous ISID. That is, it should perform session recovery as described in Chapter 6. This is because the SCSI initiator port identifier and the SCSI target port identifier (or relative target port) form the datum that the SCSI logical unit device server uses to identify the I\_T nexus.

#### 4.5. iSCSI UML Model

This section presents the application of the UML modeling concepts discussed in Section 3 to the iSCSI and SCSI architecture model discussed in Section 4.4.



(a) Each instance of an iSCSI Node class MUST contain one iSCSI Target Node instance or one iSCSI Initiator Node instance, or both.



- (b) Each instance of an iSCSI Node class MUST contain one iSCSI Target Node instance or one iSCSI Initiator Node instance, or both.

#### 4.6. Request/Response Summary

This section lists and briefly describes all the iSCSI PDU types (request and responses).

All iSCSI PDUs are built as a set of one or more header segments (basic and auxiliary) and zero or one data segments. The header group and the data segment may each be followed by a CRC (digest).

The basic header segment has a fixed length of 48 bytes.

##### 4.6.1. Request/Response Types Carrying SCSI Payload

###### 4.6.1.1. SCSI-Command

This request carries the SCSI CDB and all the other SCSI execute command procedure call (see [SAM2]) IN arguments such as task attributes, Expected Data Transfer Length for one or both transfer directions (the latter for bidirectional commands), and Task Tag (as part of the I\_T\_L\_x nexus). The I\_T\_L nexus is derived by the initiator and target from the LUN field in the request and the I\_T nexus is implicit in the session identification.

In addition, the SCSI-command PDU carries information required for the proper operation of the iSCSI protocol - the command sequence number (CmdSN) and the expected status number (ExpStatSN) on the connection it is issued.

All or part of the SCSI output (write) data associated with the SCSI command may be sent as part of the SCSI-Command PDU as a data segment.

## 4.6.1.2. SCSI-Response

The SCSI-Response carries all the SCSI execute-command procedure call (see [SAM2]) OUT arguments and the SCSI execute-command procedure call return value.

The SCSI-Response contains the residual counts from the operation, if any, an indication of whether the counts represent an overflow or an underflow, and the SCSI status if the status is valid or a response code (a non-zero return value for the execute-command procedure call) if the status is not valid.

For a valid status that indicates that the command has been processed, but resulted in an exception (e.g., a SCSI CHECK CONDITION), the PDU data segment contains the associated sense data. The use of Autosense ([SAM2]) is REQUIRED by iSCSI.

Some data segment content may also be associated (in the data segment) with a non-zero response code.

In addition, the SCSI-Response PDU carries information required for the proper operation of the iSCSI protocol:

- The number of Data-In PDUs that a target has sent (to enable the initiator to check that all have arrived).
- StatSN - the Status Sequence Number on this connection.
- ExpCmdSN - the next Expected Command Sequence Number at the target.
- MaxCmdSN - the maximum CmdSN acceptable at the target from this initiator.

## 4.6.1.3. Task Management Function Request

The Task Management function request provides an initiator with a way to explicitly control the execution of one or more SCSI Tasks or iSCSI functions. The PDU carries a function identifier (which task management function to perform) and enough information to unequivocally identify the task or task-set on which to perform

the action, even if the task(s) to act upon has not yet arrived or has been discarded due to an error.

The referenced tag identifies an individual task if the function refers to an individual task.

The I\_T\_L nexus identifies task sets. In iSCSI the I\_T\_L nexus is identified by the LUN and the session identification (the session identifies an I\_T nexus).

For task sets, the CmdSN of the Task Management function request helps identify the tasks upon which to act, namely all tasks associated with a LUN and having a CmdSN preceding the Task Management function request CmdSN.

For a Task Management function, the coordination between responses to the tasks affected and the Task Management function response is done by the target.

#### 4.6.1.4. Task Management Function Response

The Task Management function response carries an indication of function completion for a Task Management function request including how it completed (response and qualifier) and additional information for failure responses.

After the Task Management response indicates Task Management function completion, the initiator will not receive any additional responses from the affected tasks.

#### 4.6.1.5. SCSI Data-out and SCSI Data-in

SCSI Data-out and SCSI Data-in are the main vehicles by which SCSI data payload is carried between initiator and target. Data payload is associated with a specific SCSI command through the Initiator Task Tag. For target convenience, outgoing solicited data also carries a Target Transfer Tag (copied from R2T) and the LUN. Each PDU contains the payload length and the data offset relative to the buffer address contained in the SCSI execute command procedure call.

In each direction, the data transfer is split into "sequences". An end-of-sequence is indicated by the F bit.

An outgoing sequence is either unsolicited (only the first sequence can be unsolicited) or consists of all the Data-Out PDUs sent in response to an R2T.

Input sequences enable the switching of direction for bidirectional commands as required.

For input, the target may request positive acknowledgement of input data. This is limited to sessions that support error recovery and is implemented through the A bit in the SCSI Data-in PDU header.

Data-in and Data-out PDUs also carry the DataSN to enable the initiator and target to detect missing PDUs (discarded due to an error).

In addition, StatSN is carried by the Data-In PDUs.

To enable a SCSI command to be processed while involving a minimum number of messages, the last SCSI Data-in PDU passed for a command may also contain the status if the status indicates termination with no exceptions (no sense or response involved).

#### 4.6.1.6. Ready To Transfer (R2T)

R2T is the mechanism by which the SCSI target "requests" the initiator for output data. R2T specifies to the initiator the offset of the requested data relative to the buffer address from the execute command procedure call and the length of the solicited data.

To help the SCSI target associate the resulting Data-out with an R2T, the R2T carries a Target Transfer Tag that will be copied by the initiator in the solicited SCSI Data-out PDUs. There are no protocol specific requirements with regard to the value of these tags, but it is assumed that together with the LUN, they will enable the target to associate data with an R2T.



R2T also carries information required for proper operation of the iSCSI protocol, such as:

- R2TSN (to enable an initiator to detect a missing R2T)
- StatSN
- ExpCmdSN
- MaxCmdSN

#### 4.6.2. Requests/Responses carrying SCSI and iSCSI Payload

##### 4.6.2.1. Asynchronous Message

Asynchronous Messages are used to carry SCSI asynchronous events (AEN) and iSCSI asynchronous messages.

When carrying an AEN, the event details are reported as sense data in the data segment.

#### 4.6.3. Requests/Responses Carrying iSCSI Only Payload

##### 4.6.3.1. Text Request and Text Response

Text requests and responses are designed as a parameter negotiation vehicle and as a vehicle for future extension.

In the data segment Text Requests/Responses carry text information using a simple "key=value" syntax.

Text Request/Responses may form extended sequences using the same Initiator Task Tag. The initiator uses the F (Final) flag bit in the text request header to indicate its readiness to terminate a sequence. The target uses the F (Final) flag bit in the text response header to indicate its consent to sequence termination.

Text Request and Responses also use the Target Transfer Tag to indicate continuation of an operation or a new beginning. A target that wishes to continue an operation will set the Target Transfer Tag in a Text Response to a value different from the default

0xffffffff. An initiator willing to continue will copy this value into the Target Transfer Tag of the next Text Request. If the initiator wants to restart the current target negotiation (start fresh) will set the Target Transfer Tag to 0xffffffff.

Although a complete exchange is always started by the initiator, specific parameter negotiations may be initiated by the initiator or target.

#### 4.6.3.2. Login Request and Login Response

Login Requests and Responses are used exclusively during the Login Phase of each connection to set up the session and connection parameters. (The Login Phase consists of a sequence of login requests and responses carrying the same Initiator Task Tag.)

A connection is identified by an arbitrarily selected connection-ID (CID) that is unique within a session.

Similar to the Text Requests and Responses, Login Requests/Responses carry key=value text information with a simple syntax in the data segment.

The Login Phase proceeds through several stages (security negotiation, operational parameter negotiation) that are selected with two binary coded fields in the header -- the "current stage" (CSG) and the "next stage" (NSG) with the appearance of the latter being signaled by the "transit" flag (T).

The first Login Phase of a session plays a special role, called the leading login, which determines some header fields (e.g., the version number, the maximum number of connections, and the session identification).

The CmdSN initial value is also set by the leading login.

StatSN for each connection is initiated by the connection login.

A login request may indicate an implied logout (cleanup) of the connection to be logged in (a connection restart) by using the same Connection ID (CID) as an existing connection as well as the

same session identifying elements of the session to which the old connection was associated.

#### 4.6.3.3. Logout Request and Response

Logout Requests and Responses are used for the orderly closing of connections for recovery or maintenance. The logout request may be issued following a target prompt (through an asynchronous message) or at an initiators initiative. When issued on the connection to be logged out no other request may follow it.

The Logout response indicates that the connection or session cleanup is completed and no other responses will arrive on the connection (if received on the logging out connection). In addition, the Logout Response indicates how long the target will continue to hold resources for recovery (e.g., command execution that continues on a new connection) in the text key Time2Retain and how long the initiator must wait before proceeding with recovery in the text key Time2Wait.

#### 4.6.3.4. SNACK Request

With the SNACK Request, the initiator requests retransmission of numbered-responses or data from the target. A single SNACK request covers a contiguous set of missing items, called a run, of a given type of items. The type is indicated in a type field in the PDU header. The run is composed of an initial item (StatSN, DataSN, R2TSN) and the number of missed Status, Data, or R2T PDUs. For long data-in sequences, the target may request (at predefined minimum intervals) a positive acknowledgement for the data sent. A SNACK request with a type field that indicates ACK and the number of Data-In PDUs acknowledged conveys this positive acknowledgement.

#### 4.6.3.5. Reject

Reject enables the target to report an iSCSI error condition (e.g., protocol, unsupported option) that uses a Reason field in the PDU header and includes the complete header of the bad PDU in the Reject PDU data segment.

## 4.6.3.6. NOP-Out Request and NOP-In Response

This request/response pair may be used by an initiator and target as a "ping" mechanism to verify that a connection/session is still active and all of its components are operational. Such a ping may be triggered by the initiator or target. The triggering party indicates that it wants a reply by setting a value different from the default 0xffffffff in the corresponding Initiator/Target Transfer Tag.

NOP-In/NOP-Out may also be used "unidirectional" to convey to the initiator/target command, status or data counter values when there is no other "carrier" and there is a need to update the initiator/target.

iSCSI (Consolidated)  
5. SCSI Mode Parameters for iSCSI

3/11/11

There are no iSCSI specific mode pages.

Chadalapaka et al.

Expires September 11, 2011

[Page 85]

## 6. Login and Full Feature Phase Negotiation

iSCSI parameters are negotiated at session or connection establishment by using Login Requests and Responses (see Section 3.2.3 - "iSCSI Login") and during Full Feature Phase (Section 3.2.4 - "iSCSI Full Feature Phase") by using Text Requests and Responses. In both cases the mechanism used is an exchange of iSCSI-text-key=value pairs. For brevity, iSCSI-text-keys are called just keys in the rest of this document.

Keys are either declarative or require negotiation and the key description indicates if the key is declarative or requires negotiation.

For the declarative keys the declaring party sets a value for the key. The key specification indicates if the key can be declared by the initiator, target or both.

For the keys that require negotiation, one of the parties (the proposing party) proposes a value or set of values by including the key=value in the data part of a Login or Text Request or Response. The other party (the accepting party) makes a selection based on the value or list of values proposed and includes the selected value in a key=value in the data part of the following Login or Text Response or Request. For most of the keys, both the initiator and target can be proposing parties.

The login process proceeds in two stages - the security negotiation stage and the operational parameter negotiation stage. Both stages are optional but at least one of them has to be present to enable setting some mandatory parameters.

If present, the security negotiation stage precedes the operational parameter negotiation stage.

Progression from stage to stage is controlled by the T (Transition) bit in the Login Request/Response PDU header. Through the T bit set to 1, the initiator indicates that it would like to transition. The target agrees to the transition (and selects the next stage) when ready. A field in the Login PDU header indicates the current stage (CSG) and during transition, another field

indicates the next stage (NSG) proposed (initiator) and selected (target).

The Text negotiation process is used to negotiate or declare operational parameters. The negotiation process is controlled by the F (final) bit in the PDU header. During text negotiations, the F bit is used by the initiator to indicate that it is ready to finish the negotiation and by the Target to acquiesce the end of negotiation.

Since some key=value pairs may not fit entirely in a single PDU, the C (continuation) bit is used (both in Login and Text) to indicate that "more follows".

The text negotiation uses an additional mechanism by which a target may deliver larger amounts of data to an enquiring initiator. The target sets a Target Task Tag to be used as a bookmark which when returned by the initiator, means "go on". If reset to a "neutral value", it means "forget about the rest".

This chapter details types of keys and values used, the syntax rules for parameter formation, and the negotiation schemes to be used with different types of parameters.

#### 6.1. Text Format

The initiator and target send a set of key=value pairs encoded in UTF-8 Unicode. All the text keys and text values specified in this document are to be presented and interpreted in the case in which they appear in this document. They are case sensitive.

The following character symbols are used in this document for text items (the hexadecimal values represent Unicode code points):

- (a-z, A-Z) - letters
- (0-9) - digits
- " " (0x20) - space
- "." (0x2e) - dot
- "-" (0x2d) - minus
- "+" (0x2b) - plus
- "@" (0x40) - commercial at
- "\_" (0x5f) - underscore

"="	(0x3d)	-	equal
":"	(0x3a)	-	colon
"/"	(0x2f)	-	solidus or slash
"["	(0x5b)	-	left bracket
"]"	(0x5d)	-	right bracket
null	(0x00)	-	null separator
","	(0x2c)	-	comma
"~"	(0x7e)	-	tilde

Key=value pairs may span PDU boundaries. An initiator or target that sends partial key=value text within a PDU indicates that more text follows by setting the C bit in the Text or Login Request or Text or Login Response to 1. Data segments in a series of PDUs that have the C bit set to 1 and end with a PDU that have the C bit set to 0, or include a single PDU that has the C bit set to 0 have to be considered as forming a single logical-text-data-segment (LTDS).

Every key=value pair, including the last or only pair in a LTDS, MUST be followed by one null (0x00) delimiter.

A key-name is whatever precedes the first = in the key=value pair. The term key is used frequently in this document in place of key-name.

A value is whatever follows the first = in the key=value pair up to the end of the key=value pair, but not including the null delimiter.

The following definitions will be used in the rest of this document:

standard-label: A string of one or more characters that consist of letters, digits, dot, minus, plus, commercial at, or underscore. A standard-label MUST begin with a capital letter and must not exceed 63 characters.

key-name: A standard-label.



text-value: A string of zero or more characters that consist of letters, digits, dot, minus, plus, commercial at, underscore, slash, left bracket, right bracket, or colon.

iSCSI-name-value: A string of one or more characters that consist of minus, dot, colon, or any character allowed by the output of the iSCSI string-prep template as specified in [RFC3722] (see also Section 3.2.6.2 - "iSCSI Name Encoding").

iSCSI-local-name-value: A UTF-8 string; no null characters are allowed in the string. This encoding is to be used for localized (internationalized) aliases.

boolean-value: The string "Yes" or "No".

hex-constant: A hexadecimal constant encoded as a string that starts with "0x" or "0X" followed by one or more digits or the letters a, b, c, d, e, f, A, B, C, D, E, or F. Hex-constants are used to encode numerical values or binary strings. When used to encode numerical values, the excessive use of leading 0 digits is discouraged. The string following 0X (or 0x) represents a base16 number that starts with the most significant base16 digit, followed by all other digits in decreasing order of significance and ending with the least-significant base16 digit. When used to encode binary strings, hexadecimal constants have an implicit byte-length that includes four bits for every hexadecimal digit of the constant, including leading zeroes. For example, a hex-constant of n hexadecimal digits has a byte-length of (the integer part of)  $(n+1)/2$ .

decimal-constant: An unsigned decimal number with the digit 0 or a string of one or more digits that start with a non-zero digit. Decimal-constants are used to encode numerical values or binary strings. Decimal constants can only be used to encode binary strings if the string length is explicitly specified. There is no implicit length for decimal strings. Decimal-constant MUST NOT be used for parameter values if the values can be equal or greater than  $2^{64}$  (numerical) or for binary strings that can be longer than 64 bits.

base64-constant: base64 constant encoded as a string that starts with "0b" or "0B" followed by 1 or more digits or letters or plus or slash or equal. The encoding is done according to [RFC2045] and each character, except equal, represents a base64 digit or a 6-bit binary string. Base64-constants are used to encode numerical-values or binary strings. When used to encode numerical values, the excessive use of leading 0 digits (encoded as A) is discouraged. The string following 0B (or 0b) represents a base64 number that starts with the most significant base64 digit, followed by all other digits in decreasing order of significance and ending with the least-significant base64 digit; the least significant base64 digit may be optionally followed by pad digits (encoded as equal) that are not considered as part of the number. When used to encode binary strings, base64-constants have an implicit byte-length that includes six bits for every character of the constant, excluding trailing equals (i.e., a base64-constant of n base64 characters excluding the trailing equals has a byte-length of ((the integer part of)  $(n \cdot 3/4)$ ). Correctly encoded base64 strings cannot have n values of 1, 5 ...  $k \cdot 4 + 1$ .

numerical-value: An unsigned integer always less than  $2^{64}$  encoded as a decimal-constant or a hex-constant. Unsigned integer arithmetic applies to numerical-values.

large-numerical-value: An unsigned integer that can be larger than or equal to  $2^{64}$  encoded as a hex constant, or base64-constant. Unsigned integer arithmetic applies to large-numeric-values.

numeric-range: Two numerical-values separated by a tilde where the value to the right of tilde must not be lower than the value to the left.

regular-binary-value: A binary string not longer than 64 bits encoded as a decimal constant, hex constant, or base64-constant. The length of the string is either specified by the key definition or is the implicit byte-length of the encoded string.

large-binary-value: A binary string longer than 64 bits encoded as a hex-constant or base64-constant. The length of the string is either specified by the key definition or is the implicit byte-length of the encoded string.

binary-value: A regular-binary-value or a large-binary-value. Operations on binary values are key specific.

simple-value: Text-value, iSCSI-name-value, boolean-value, numeric-value, a numeric-range, or a binary-value.

list-of-values: A sequence of text-values separated by a comma.

If not otherwise specified, the maximum length of a simple-value (not its encoded representation) is 255 bytes not including the delimiter (comma or zero byte).

Any iSCSI target or initiator MUST support receiving at least 8192 bytes of key=value data in a negotiation sequence. When proposing or accepting authentication methods that explicitly require support for very long authentication items, the initiator and target MUST support receiving of at least 64 kilobytes of key=value data.

## 6.2. Text Mode Negotiation

During login, and thereafter, some session or connection parameters are either declared or negotiated through an exchange of textual information.

The initiator starts the negotiation and/or declaration through a Text or Login request and indicates when it is ready for completion (by setting the F bit to 1 and keeping it to 1 in a Text Request or the T bit in the Login Request). As negotiation text may span PDU boundaries, a Text or Login Request or Text or Login Response PDU that have the C bit set to 1 MUST NOT have the F/T bit set to 1.

A target receiving a Text or Login Request with the C bit set to 1 MUST answer with a Text or Login Response with no data segment (DataSegmentLength 0). An initiator receiving a Text or Login Response with the C bit set to 1 MUST answer with a Text or Login Request with no data segment (DataSegmentLength 0).

A target or initiator SHOULD NOT use a Text or Login Response or Text or Login Request with no data segment (DataSegmentLength 0) unless explicitly required by a general or a key-specific negotiation rule.

There MUST NOT be more than one outstanding Text Request, or Text Response PDU on an iSCSI connection. An outstanding PDU in this

context is one that has not been acknowledged by the remote iSCSI side.

The format of a declaration is:

Declarer-> <key>=<valuex>

The general format of text negotiation is:

Proposer-> <key>=<valuex>

Acceptor-> <key>={<valuey>|NotUnderstood|Irrelevant|Reject}

Thus a declaration is a one-way textual exchange (unless the key is not understood by the receiver) while a negotiation is a two-way exchange.

The proposer or declarer can either be the initiator or the target, and the acceptor can either be the target or initiator, respectively. Targets are not limited to respond to key=value pairs as proposed by the initiator. The target may propose key=value pairs of its own.

All negotiations are explicit (i.e., the result MUST only be based on newly exchanged or declared values). There are no implicit proposals. If a proposal is not made, then a reply cannot be expected. Conservative design also requires that default values should not be relied upon when use of some other value has serious consequences.

The value proposed or declared can be a numerical-value, a numerical-range defined by lower and upper value with both integers separated by tilde, a binary value, a text-value, an iSCSI-name-value, an iSCSI-local-name-value, a boolean-value (Yes or No), or a list of comma separated text-values. A range, a large-numerical-value, an iSCSI-name-value and an iSCSI-local-name-value MAY ONLY be used if it is explicitly allowed. An accepted value can be a numerical-value, a large-numerical-value, a text-value, or a boolean-value.

If a specific key is not relevant for the current negotiation, the acceptor may answer with the constant "Irrelevant" for all types of negotiation. However the negotiation is not considered as failed if the answer is "Irrelevant". The "Irrelevant" answer is meant for those cases in which several keys are presented by a proposing party but the selection made by the acceptor for one of the keys makes other keys irrelevant. The following example illustrates the use of "Irrelevant":

I->T InitialR2T=No, ImmediateData=Yes, FirstBurstLength=4192

T->I InitialR2T=Yes, ImmediateData=No, FirstBurstLength=Irrelevant

I->T X#vkey1=(bla, alb, None), X#vkey2=(bla, alb)

T->I X#vkey1=None, X#vkey2=Irrelevant

Any key not understood by the acceptor may be ignored by the acceptor without affecting the basic function. However, the answer for a key not understood MUST be key=NotUnderstood. Note that NotUnderstood is a valid answer for both declarative and negotiated keys. The general iSCSI philosophy is that comprehension precedes processing for any iSCSI key. A proposer of an iSCSI key, negotiated or declarative, in a text key exchange MUST thus be able to properly handle a NotUnderstood response.

The proper way to handle a NotUnderstood response depends on where the key is specified and whether the key is declarative vs. negotiated. All keys defined in [RFC3720] MUST be supported by all compliant implementations; a NotUnderstood answer on any of the [RFC3720] keys therefore MUST be considered a protocol error and handled accordingly. For all other later keys, a NotUnderstood answer concludes the negotiation for a negotiated key whereas for a declarative key, a NotUnderstood answer simply informs the declarer of a lack of comprehension by the receiver.

In either case, a NotUnderstood answer always requires that the protocol behavior associated with that key not be used within the scope of the key (connection/session) by either side.

The constants "None", "Reject", "Irrelevant", and "NotUnderstood" are reserved and MUST ONLY be used as described here. Violation of

this rule is a protocol error (in particular the use of "Reject", "Irrelevant", and "NotUnderstood" as proposed values).

Reject or Irrelevant are legitimate negotiation options where allowed but their excessive use is discouraged. A negotiation is considered complete when the acceptor has sent the key value pair even if the value is "Reject", "Irrelevant", or "NotUnderstood". Sending the key again would be a re-negotiation and is forbidden for many keys.

If the acceptor sends "Reject" as an answer the negotiated key is left at its current value (or default if no value was set). If the current value is not acceptable to the proposer on the connection or to the session it is sent, the proposer MAY choose to terminate the connection or session.

All keys in this document, except for the X extension formats, MUST be supported by iSCSI initiators and targets when used as specified here. If used as specified, these keys MUST NOT be answered with NotUnderstood.

Implementers may introduce new keys by prefixing them with X- followed by their (reversed) domain name, or with new keys registered with IANA prefixing them with X#. For example, the entity owning the domain example.com can issue:

```
X-com.example.bar.foo.do_something=3
```

or a new registered key may be used as in:

```
X#SuperCalyPhraGilistic=Yes
```

Implementers MAY also introduce new values, but ONLY for new keys or authentication methods (see Section 11 - "iSCSI Security Text Keys and Authentication Methods"), or digests (see Section 12.1 - "HeaderDigest and DataDigest").

Whenever parameter action or acceptance are dependent on other parameters, the dependency rules and parameter sequence must be specified with the parameters.

In the Login Phase (see Login Phase), every stage is a separate negotiation. In the FullFeaturePhase, a Text Request Response sequence is a negotiation. Negotiations MUST be handled as atomic operations. For example, all negotiated values go into effect after the negotiation concludes in agreement or are ignored if the negotiation fails.

Some parameters may be subject to integrity rules (e.g., parameter-x must not exceed parameter-y or parameter-u not 1 implies parameter-v be Yes). Whenever required, integrity rules are specified with the keys. Checking for compliance with the integrity rule must only be performed after all the parameters are available (the existent and the newly negotiated). An iSCSI target MUST perform integrity checking before the new parameters take effect. An initiator MAY perform integrity checking.

An iSCSI initiator or target MAY terminate a negotiation that does not end within a reasonable time or number of exchanges.

#### 6.2.1. List negotiations

In list negotiation, the originator sends a list of values (which may include "None") in its order of preference.

The responding party MUST respond with the same key and the first value that it supports (and is allowed to use for the specific originator) selected from the originator list.

The constant "None" MUST always be used to indicate a missing function. However, "None" is only a valid selection if it is explicitly proposed.

If an acceptor does not understand any particular value in a list, it MUST ignore it. If an acceptor does not support, does not understand, or is not allowed to use any of the proposed options with a specific originator, it may use the constant "Reject" or terminate the negotiation. The selection of a value not proposed MUST be handled as a protocol error.



### 6.2.2. Simple-value Negotiations

For simple-value negotiations, the accepting party **MUST** answer with the same key. The value it selects becomes the negotiation result.

Proposing a value not admissible (e.g., not within the specified bounds) **MAY** be answered with the constant "Reject" or the acceptor **MAY** select an admissible value.

The selection, by the acceptor, of a value not admissible under the selection rules is considered a protocol error. The selection rules are key-specific.

For a numerical range the value selected must be an integer within the proposed range or "Reject" (if the range is unacceptable).

For Boolean negotiations (i.e., keys taking the values Yes or No), the accepting party **MUST** answer with the same key and the result of the negotiation when the received value does not determine that result by itself. The last value transmitted becomes the negotiation result. The rules for selecting the value to answer with are expressed as Boolean functions of the value received, and the value that the accepting party would have selected if given a choice.

Specifically, the two cases in which answers are **OPTIONAL** are:

- The Boolean function is "AND" and the value "No" is received. The outcome of the negotiation is "No".
- The Boolean function is "OR" and the value "Yes" is received. The outcome of the negotiation is "Yes".

Responses are **REQUIRED** in all other cases, and the value chosen and sent by the acceptor becomes the outcome of the negotiation.

### 6.3. Login Phase

The Login Phase establishes an iSCSI connection between an initiator and a target; it creates also a new session or

associates the connection to an existing session. The Login Phase sets the iSCSI protocol parameters, security parameters, and authenticates the initiator and target to each other.

The Login Phase is only implemented via Login request and responses. The whole Login Phase is considered as a single task and has a single Initiator Task Tag (similar to the linked SCSI commands).

There MUST NOT be more than one outstanding Login Request, or Login Response on an iSCSI connection. An outstanding PDU in this context is one that has not been acknowledged by the remote iSCSI side.

The default MaxRecvDataSegmentLength is used during Login.

The Login Phase sequence of requests and responses proceeds as follows:

- Login initial request
- Login partial response (optional)
- More Login requests and responses (optional)
- Login Final-Response (mandatory)

The initial login request of any connection MUST include the InitiatorName key=value pair. The initial login request of the first connection of a session MAY also include the SessionType key=value pair. For any connection within a session whose type is not "Discovery", the first login request MUST also include the TargetName key=value pair.

The Login Final-response accepts or rejects the Login request.

The Login Phase MAY include a SecurityNegotiation stage and a LoginOperationalNegotiation stage and MUST include at least one of them, but the included stage MAY be empty except for the mandatory names.

The login requests and responses contain a field (CSG) that indicates the current negotiation stage (SecurityNegotiation or LoginOperationalNegotiation). If both stages are used, the SecurityNegotiation MUST precede the LoginOperationalNegotiation.

Some operational parameters can be negotiated outside the login through Text requests and responses.

Security MUST be completely negotiated within the Login Phase. The use of underlying IPsec security is specified in Chapter 8 and in [RFC3723]. iSCSI support for security within the protocol only consists of authentication in the Login Phase.

In some environments, a target or an initiator is not interested in authenticating its counterpart. It is possible to bypass authentication through the Login request and response.

The initiator and target MAY want to negotiate iSCSI authentication parameters. Once this negotiation is completed, the channel is considered secure.

Most of the negotiation keys are only allowed in a specific stage. The SecurityNegotiation keys appear in Chapter 11 and the LoginOperationalNegotiation keys appear in Chapter 12. Only a limited set of keys (marked as Any-Stage in Chapter 12) may be used in any of the two stages.

Any given Login request or response belongs to a specific stage; this determines the negotiation keys allowed with the request or response. It is considered to be a protocol error to send a key not allowed in the current stage.

Stage transition is performed through a command exchange (request/response) that carries the T bit and the same CSG code. During this exchange, the next stage is selected by the target through the "next stage" code (NSG). The selected NSG MUST NOT exceed the value stated by the initiator. The initiator can request a transition whenever it is ready, but a target can only respond with a transition after one is proposed by the initiator.

In a negotiation sequence, the T bit settings in one pair of login request-responses have no bearing on the T bit settings of the next pair. An initiator that has a T bit set to 1 in one pair and is answered with a T bit setting of 0 may issue the next request with T bit set to 0.

When a transition is requested by the initiator and acknowledged by the target, both the initiator and target switch to the selected stage.

Targets MUST NOT submit parameters that require an additional initiator login request in a login response with the T bit set to 1.

Stage transitions during login (including entering and exit) are only possible as outlined in the following table:

From   V	To ->	Security	Operational	FullFeature
(start)		yes	yes	no
Security		no	yes	yes
Operational		no	no	yes

The Login Final-Response that accepts a Login Request can only come as a response to a Login request with the T bit set to 1, and both the request and response MUST indicate FullFeaturePhase as the next phase via the NSG field.

Neither the initiator nor the target should attempt to declare or negotiate a parameter more than once during login except for responses to specific keys that explicitly allow repeated key declarations (e.g., TargetAddress). An attempt to renegotiate/redeclare parameters not specifically allowed MUST be detected by the initiator and target. If such an attempt is detected by the target, the target MUST respond with Login reject

iSCSI (Consolidated) 3/11/11  
(initiator error); if detected by the initiator, the initiator  
MUST drop the connection.

#### 6.3.1. Login Phase Start

The Login Phase starts with a login request from the initiator to the target. The initial login request includes:

- Protocol version supported by the initiator.
- iSCSI Initiator Name and iSCSI Target Name
- ISID, TSIH, and connection Ids
- Negotiation stage that the initiator is ready to enter.

A login may create a new session or it may add a connection to an existing session. Between a given iSCSI Initiator Node (selected only by an InitiatorName) and a given iSCSI target defined by an iSCSI TargetName and a Target Portal Group Tag, the login results are defined by the following table:

ISID	TSIH	CID	Target action
new	non-zero	any	fail the login ("session does not exist")
new	zero	any	instantiate a new session
existing	zero	any	do session reinstatement (see Section 6.3.5)

iSCSI (Consolidated)			3/11/11
existing	non-zero existing	new	add a new connection to the session
existing	non-zero existing	existing	do connection reinstatement (see Section 7.1.4.3)
existing	non-zero new	any	fail the login ("session does not exist")

Determination of "existing" or "new" are made by the target.

Optionally, the login request may include:

- Security parameters  
OR
- iSCSI operational parameters  
AND/OR
- The next negotiation stage that the initiator is ready to enter.

The target can answer the login in the following ways:

- Login Response with Login reject. This is an immediate rejection from the target that causes the connection to terminate and the session to terminate if this is the first (or only) connection of a new session. The T bit and the CSG and NSG fields are reserved.
- Login Response with Login accept as a final response (T bit set to 1 and the NSG in both request and response are set to FullFeaturePhase). The response includes the protocol version supported by the target and the session ID, and may include iSCSI operational or security parameters (that depend on the current stage).
- Login Response with Login Accept as a partial response (NSG not set to FullFeaturePhase in both request and response)

that indicates the start of a negotiation sequence. The response includes the protocol version supported by the target and either security or iSCSI parameters (when no security mechanism is chosen) supported by the target.

If the initiator decides to forego the SecurityNegotiation stage, it issues the Login with the CSG set to LoginOperationalNegotiation and the target may reply with a Login Response that indicates that it is unwilling to accept the connection (see Section 10.13 - "Login Response") without SecurityNegotiation and will terminate the connection with a response of Authentication failure (see Section 10.13.5 - "Status-Class and Status-Detail").

If the initiator is willing to negotiate iSCSI security, but is unwilling to make the initial parameter proposal and may accept a connection without iSCSI security, it issues the Login with the T bit set to 1, the CSG set to SecurityNegotiation, and NSG set to LoginOperationalNegotiation. If the target is also ready to skip security, the login response only contains the TargetPortalGroupTag key (see Section 12.9 - "TargetPortalGroupTag"), the T bit set to 1, the CSG set to SecurityNegotiation, and NSG set to LoginOperationalNegotiation.

An initiator that chooses to operate without iSCSI security and with all the operational parameters taking the default values issues the Login with the T bit set to 1, the CSG set to LoginOperationalNegotiation, and NSG set to FullFeaturePhase. If the target is also ready to forego security and can finish its LoginOperationalNegotiation, the Login response has T bit set to 1, the CSG set to LoginOperationalNegotiation, and NSG set to FullFeaturePhase in the next stage.

During the Login Phase the iSCSI target MUST return the TargetPortalGroupTag key with the first Login Response PDU with which it is allowed to do so (i.e., the first Login Response issued after the first Login Request with the C bit set to 0) for all session types. The TargetPortalGroupTag key value indicates the iSCSI portal group servicing the Login Request PDU. If the reconfiguration of iSCSI portal groups is a concern in a given environment, the iSCSI initiator should use this key to ascertain

that it had indeed initiated the Login Phase with the intended target portal group.

### 6.3.2. iSCSI Security Negotiation

The security exchange sets the security mechanism and authenticates the initiator user and the target to each other. The exchange proceeds according to the authentication method chosen in the negotiation phase and is conducted using the login requests' and responses' key=value parameters.

An initiator directed negotiation proceeds as follows:

- The initiator sends a login request with an ordered list of the options it supports (authentication algorithm). The options are listed in the initiator's order of preference. The initiator MAY also send private or public extension options.
- The target MUST reply with the first option in the list it supports and is allowed to use for the specific initiator unless it does not support any in which case it MUST answer with "Reject" (see Text Mode Negotiation). The parameters are encoded in UTF8 as key=value. For security parameters, see Chapter 11.
- When the initiator considers that it is ready to conclude the SecurityNegotiation stage, it sets the T bit to 1 and the NSG to what it would like the next stage to be. The target will then set the T bit to 1 and set NSG to the next stage in the Login response when it finishes sending its security keys. The next stage selected will be the one the target selected. If the next stage is FullFeaturePhase, the target MUST respond with a Login Response with the TSIH value.

If the security negotiation fails at the target, then the target MUST send the appropriate Login Response PDU. If the security



negotiation fails at the initiator, the initiator SHOULD close the connection.

It should be noted that the negotiation might also be directed by the target if the initiator does support security, but is not ready to direct the negotiation (propose options).

### 6.3.3. Operational Parameter Negotiation During the Login Phase

Operational parameter negotiation during the login MAY be done:

- Starting with the first Login request if the initiator does not propose any security/ integrity option.
  
- Starting immediately after the security negotiation if the initiator and target perform such a negotiation.

Operational parameter negotiation MAY involve several Login request-response exchanges started and terminated by the initiator. The initiator MUST indicate its intent to terminate the negotiation by setting the T bit to 1; the target sets the T bit to 1 on the last response.

If the target responds to a Login request that has the T bit set to 1 with a Login response that has the T bit set to 0, the initiator should keep sending the Login request (even empty) with the T bit set to 1, while it still wants to switch stage, until it receives the Login Response that has the T bit set to 1 or it receives a key that requires it to set the T bit to 0.

Some session specific parameters can only be specified during the Login Phase of the first connection of a session (i.e., begun by a login request that contains a zero-valued TSIH) - the leading Login Phase (e.g., the maximum number of connections that can be used for this session).

A session is operational once it has at least one connection in FullFeaturePhase. New or replacement connections can only be added to a session after the session is operational.

For operational parameters, see Chapter 12.

#### 6.3.4. Connection Reinstatement

Connection reinstatement is the process of an initiator logging in with a ISID-TSIH-CID combination that is possibly active from the target's perspective, which causes the implicit logging out of the connection corresponding to the CID and reinstating a new Full Feature Phase iSCSI connection in its place (with the same CID). Thus, the TSIH in the Login Request PDU MUST be non-zero and CID does not change during a connection reinstatement. The Login request performs the logout function of the old connection if an explicit logout was not performed earlier. In sessions with a single connection, this may imply the opening of a second connection with the sole purpose of cleaning up the first. Targets MUST support opening a second connection even when they do not support multiple connections in Full Feature Phase if ErrorRecoveryLevel is 2 and SHOULD support opening a second connection if ErrorRecoveryLevel is less than 2.

If the operational ErrorRecoveryLevel is 2, connection reinstatement enables future task reassignment. If the operational ErrorRecoveryLevel is less than 2, connection reinstatement is the replacement of the old CID without enabling task reassignment. In this case, all the tasks that were active on the old CID must be immediately terminated without further notice to the initiator.

The initiator connection state MUST be CLEANUP\_WAIT (section 7.1.3) when the initiator attempts a connection reinstatement.

In practical terms, in addition to the implicit logout of the old connection, reinstatement is equivalent to a new connection login.

#### 6.3.5. Session Reinstatement, Closure, and Timeout

Session reinstatement is the process of the initiator logging in with an ISID that is possibly active from the target's

perspective. Thus implicitly logging out the session that corresponds to the ISID and reinstating a new iSCSI session in its place (with the same ISID). Therefore, the TSIH in the Login PDU MUST be zero to signal session reinstatement. Session reinstatement causes all the tasks that were active on the old session to be immediately terminated by the target without further notice to the initiator.

The initiator session state MUST be FAILED (Section 7.3 - "Session State Diagrams") when the initiator attempts a session reinstatement.

Session closure is an event defined to be one of the following:

- A successful "session close" logout.
- A successful "connection close" logout for the last Full Feature Phase connection when no other connection in the session is waiting for cleanup (Section 7.2 - "Connection Cleanup State Diagram for Initiators and Targets") and no tasks in the session are waiting for reassignment.

Session timeout is an event defined to occur when the last connection state timeout expires and no tasks are waiting for reassignment. This takes the session to the FREE state (N6 transition in the session state diagram).

#### 6.3.5.1. Loss of Nexus Notification

The iSCSI layer provides the SCSI layer with the "I\_T nexus loss" notification when any one of the following events happens:

- Successful completion of session reinstatement.
- Session closure event.
- Session timeout event.

Certain SCSI object clearing actions may result due to the notification in the SCSI end nodes, as documented in Appendix F. - Clearing Effects of Various Events on Targets.

#### 6.3.6. Session Continuation and Failure

Session continuation is the process by which the state of a preexisting session continues to be used by connection reinstatement (Section 6.3.4), or by adding a connection with a new CID. Either of these actions associates the new transport connection with the session state.

Session failure is an event where the last Full Feature Phase connection reaches the CLEANUP\_WAIT state (Section 8.2), or completes a successful recovery logout thus causing all active tasks (that are formerly allegiant to the connection) to start waiting for task reassignment.

#### 6.4. Operational Parameter Negotiation Outside the Login Phase

Some operational parameters MAY be negotiated outside (after) the Login Phase.

Parameter negotiation in Full Feature Phase is done through Text requests and responses. Operational parameter negotiation MAY involve several Text request-response exchanges, which the initiator always starts, terminates, and uses the same Initiator Task Tag. The initiator MUST indicate its intent to terminate the negotiation by setting the F bit to 1; the target sets the F bit to 1 on the last response.

If the target responds to a Text request with the F bit set to 1 with a Text response with the F bit set to 0, the initiator should keep sending the Text request (even empty) with the F bit set to 1, while it still wants to finish the negotiation, until it receives the Text response with the F bit set to 1. Responding to a Text request with the F bit set to 1 with an empty (no key=value pairs) response with the F bit set to 0 is discouraged.

Targets MUST NOT submit parameters that require an additional initiator Text request in a Text response with the F bit set to 1.

In a negotiation sequence, the F bit settings in one pair of Text request-responses have no bearing on the F bit settings of the next pair. An initiator that has the F bit set to 1 in a request

and is being answered with an F bit setting of 0 may issue the next request with the F bit set to 0.

Whenever the target responds with the F bit set to 0, it MUST set the Target Transfer Tag to a value other than the default 0xffffffff.

An initiator MAY reset an operational parameter negotiation by issuing a Text request with the Target Transfer Tag set to the value 0xffffffff after receiving a response with the Target Transfer Tag set to a value other than 0xffffffff. A target may reset an operational parameter negotiation by answering a Text request with a Reject PDU.

Neither the initiator nor the target should attempt to declare or negotiate a parameter more than once during any negotiation sequence, except for responses to specific keys that explicitly allow repeated key declarations (e.g., TargetAddress). If detected by the target, this MUST result in a Reject PDU with a reason of "protocol error". The initiator MUST reset the negotiation as outlined above.

Parameters negotiated by a text exchange negotiation sequence only become effective after the negotiation sequence is completed.

## 7. iSCSI Error Handling and Recovery

### 7.1. Overview

#### 7.1.1. Background

The following two considerations prompted the design of much of the error recovery functionality in iSCSI:

An iSCSI PDU may fail the digest check and be dropped, despite being received by the TCP layer. The iSCSI layer must optionally be allowed to recover such dropped PDUs.

A TCP connection may fail at any time during the data transfer. All the active tasks must optionally be allowed to be continued on a different TCP connection within the same session.

Implementations have considerable flexibility in deciding what degree of error recovery to support, when to use it and by which mechanisms to achieve the required behavior. Only the externally visible actions of the error recovery mechanisms must be standardized to ensure interoperability.

This chapter describes a general model for recovery in support of interoperability. See Appendix E. - "Algorithmic Presentation of Error Recovery Classes" for further detail on how the described model may be implemented. Compliant implementations do not have to match the implementation details of this model as presented, but the external behavior of such implementations must correspond to the externally observable characteristics of the presented model.

#### 7.1.2. Goals

The major design goals of the iSCSI error recovery scheme are as follows:

Allow iSCSI implementations to meet different requirements by defining a collection of error recovery mechanisms that implementations may choose from.

Ensure interoperability between any two implementations supporting different sets of error recovery capabilities.

Define the error recovery mechanisms to ensure command ordering even in the face of errors, for initiators that demand ordering.

Do not make additions in the fast path, but allow moderate complexity in the error recovery path.

Prevent both the initiator and target from attempting to recover the same set of PDUs at the same time. For example, there must be a clear "error recovery functionality distribution" between the initiator and target.

### 7.1.3. Protocol Features and State Expectations

The initiator mechanisms defined in connection with error recovery are:

- NOP-OUT to probe sequence numbers of the target (Section 10.18)

- Command retry (Section 7.2.1)

- Recovery R2T support (Section 7.8)

- Requesting retransmission of status/data/R2T using the SNACK facility (section 10.16)

- Acknowledging the receipt of the data (section 10.16)

  - Reassigning the connection allegiance of a task to a different TCP connection (Section 7.2.2)

- Terminating the entire iSCSI session to start afresh (Session Recovery)

The target mechanisms defined in connection with error recovery are:

- NOP-IN to probe sequence numbers of the initiator (section 10.19)

- Requesting retransmission of data using the recovery R2T feature (iSCSI Error)

- SNACK support (section 10.16)

  - Requesting that parts of read data be acknowledged (section 10.7.2)

  - Allegiance reassignment support (Section 7.2.2)

  - Terminating the entire iSCSI session to force the initiator to start over (Session Recovery)

For any outstanding SCSI command, it is assumed that iSCSI, in conjunction with SCSI at the initiator, is able to keep enough information to be able to rebuild the command PDU, and that outgoing data is available (in host memory) for retransmission while the command is outstanding. It is also assumed that at the target, incoming data (read data) MAY be kept for recovery or it can be reread from a device server.

It is further assumed that a target will keep the "status & sense" for a command it has executed if it supports status retransmission.

A target that agrees to support data retransmission is expected to be prepared to retransmit the outgoing data (i.e., Data-In) on request until either the status for the completed command is acknowledged, or the data in question has been separately acknowledged.

#### 7.1.4. Recovery Classes

iSCSI enables the following classes of recovery (in the order of increasing scope of affected iSCSI tasks):

- Within a command (i.e., without requiring command restart).
- Within a connection (i.e., without requiring the connection to be rebuilt, but perhaps requiring command restart).
- Connection recovery (i.e., perhaps requiring connections to be rebuilt and commands to be reissued).
- Session recovery.

The recovery scenarios detailed in the rest of this section are representative rather than exclusive. In every case, they detail the lowest class recovery that MAY be attempted. The implementer is left to decide under which circumstances to escalate to the next recovery class and/or what recovery classes to implement. Both the iSCSI target and initiator MAY escalate the error handling to an error recovery class, which impacts a larger number



of iSCSI tasks in any of the cases identified in the following discussion.

In all classes, the implementer has the choice of deferring errors to the SCSI initiator (with an appropriate response code), in which case the task, if any, has to be removed from the target and all the side-effects, such as ACA, must be considered.

Use of within-connection and within-command recovery classes MUST NOT be attempted before the connection is in Full Feature Phase.

In the detailed description of the recover classes the mandating terms (MUST, SHOULD, MAY, etc.) indicate normative actions to be executed if the recovery class is supported and used.

#### 7.1.4.1. Recovery Within-command

At the target, the following cases lend themselves to within-command recovery:

- Lost data PDU - realized through one of the following:

Data digest error - dealt with as specified in Section 7.8, using the option of a recovery R2T.

Sequence reception timeout (no data or partial-data-and-no-F-bit) - considered an implicit sequence error and dealt with as specified in Section 7.9, using the option of a recovery R2T.

Header digest error, which manifests as a sequence reception timeout or a sequence error - dealt with as specified in Section 7.9, using the option of a recovery R2T.

At the initiator, the following cases lend themselves to within-command recovery:

- Lost data PDU or lost R2T - realized through one of the following:

Data digest error - dealt with as specified in Section 7.8, using the option of a SNACK.

Sequence reception timeout (no status) or response reception timeout - dealt with as specified in Section 7.9, using the option of a SNACK.

Header digest error, which manifests as a sequence reception timeout or a sequence error - dealt with as specified in Section 7.9, using the option of a SNACK.

To avoid a race with the target, which may already have a recovery R2T or a termination response on its way, an initiator SHOULD NOT originate a SNACK for an R2T based on its internal timeouts (if any). Recovery in this case is better left to the target.

The timeout values used by the initiator and target are outside the scope of this document. Sequence reception timeout is generally a large enough value to allow the data sequence transfer to be complete.

#### 7.1.4.2. Recovery Within-connection

At the initiator, the following cases lend themselves to within-connection recovery:

- Requests not acknowledged for a long time. Requests are acknowledged explicitly through ExpCmdSN or implicitly by receiving data and/or status. The initiator MAY retry non-acknowledged commands as specified in Retry an.
- Lost iSCSI numbered Response. It is recognized by either identifying a data digest error on a Response PDU or a Data-In PDU carrying the status, or by receiving a Response PDU with a higher StatSN than expected. In the first case, digest error handling is done as specified in Section 7.8 using the option of a SNACK. In the second case, sequence error handling is done as specified in Section 7.9, using the option of a SNACK.

At the target, the following cases lend themselves to within-connection recovery:

- Status/Response not acknowledged for a long time. The target MAY issue a NOP-IN (with a valid Target Transfer Tag or otherwise) that carries the next status sequence number it is going to use in the StatSN field. This helps the initiator detect any missing StatSN(s) and issue a SNACK for the status.

The timeout values used by the initiator and the target are outside the scope of this document.

#### 7.1.4.3. Connection Recovery

At an iSCSI initiator, the following cases lend themselves to connection recovery:

- TCP connection failure: The initiator MUST close the connection. It then MUST either implicitly or explicitly logout the failed connection with the reason code "remove the connection for recovery" and reassign connection allegiance for all commands still in progress associated with the failed connection on one or more connections (some or all of which MAY be newly established connections) using the "Task reassign" task management function (see Section 10.5.1 - "Function"). For an initiator, a command is in progress as long as it has not received a response or a Data-In PDU including status.

Note: The logout function is mandatory. However, a new connection establishment is only mandatory if the failed connection was the last or only connection in the session.

- Receiving an Asynchronous Message that indicates one or all connections in a session has been dropped. The initiator MUST handle it as a TCP connection failure for the connection(s) referred to in the Message.

At an iSCSI target, the following cases lend themselves to connection recovery:

- TCP connection failure. The target MUST close the connection and, if more than one connection is available, the target SHOULD send an Asynchronous Message that indicates it has dropped the connection. Then, the target will wait for the initiator to continue recovery.

#### 7.1.4.4. Session Recovery

Session recovery should be performed when all other recovery attempts have failed. Very simple initiators and targets MAY perform session recovery on all iSCSI errors and rely on recovery on the SCSI layer and above.

Session recovery implies the closing of all TCP connections, internally aborting all executing and queued tasks for the given initiator at the target, terminating all outstanding SCSI commands with an appropriate SCSI service response at the initiator, and restarting a session on a new set of connection(s) (TCP connection establishment and login on all new connections).

For possible clearing effects of session recovery on SCSI and iSCSI objects, refer to Appendix F. - "Clearing Effects of Various Events on Targets".

#### 7.1.5. Error Recovery Hierarchy

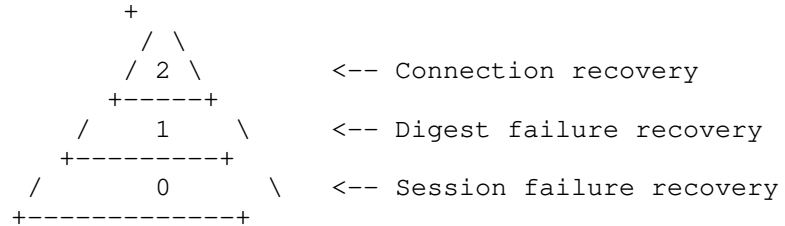
The error recovery classes described so far are organized into a hierarchy for ease in understanding and to limit the implementation complexity. With few and well defined recovery levels interoperability is easier to achieve. The attributes of this hierarchy are as follows:

- Each level is a superset of the capabilities of the previous level. For example, Level 1 support implies supporting all capabilities of Level 0 and more.

As a corollary, supporting a higher error recovery level means increased sophistication and possibly an increase in resource requirements.

Supporting error recovery level "n" is advertised and negotiated by each iSCSI entity by exchanging the text key "ErrorRecoveryLevel=n". The lower of the two exchanged values is the operational ErrorRecoveryLevel for the session.

The following diagram represents the error recovery hierarchy.



The following table lists the error recovery capabilities expected from the implementations that support each error recovery level.

ErrorRecoveryLevel	Associated Error recovery capabilities
0	Session recovery class (Session Recovery)
1	Digest failure recovery (See Note below.) plus the capabilities of ER Level 0
2	Connection recovery class (Connection Recovery) plus the capabilities of ER Level 1

Note: Digest failure recovery is comprised of two recovery classes: Within-Connection recovery class (Recovery Within-connection) and Within-Command recovery class (Recovery Within-command ).

When a defined value of `ErrorRecoveryLevel` is proposed by an originator in a text negotiation, the originator **MUST** support the functionality defined for the proposed value and additionally, functionality corresponding to any defined value numerically less than the proposed. When a defined value of `ErrorRecoveryLevel` is returned by a responder in a text negotiation, the responder **MUST** support the functionality corresponding to the `ErrorRecoveryLevel` it is accepting.

When either party attempts to use error recovery functionality beyond what is negotiated, the recovery attempts **MAY** fail unless an apriori agreement outside the scope of this document exists between the two parties to provide such support.

Implementations **MUST** support error recovery level "0", while the rest are **OPTIONAL** to implement. In implementation terms, the above striation means that the following incremental sophistication with each level is required.

Level transition	Incremental requirement
0->1	PDU retransmissions on the same connection
1->2	Retransmission across connections and allegiance reassignment

## 7.2. Retry and Reassign in Recovery

This section summarizes two important and somewhat related iSCSI protocol features used in error recovery.

### 7.2.1. Usage of Retry

By resending the same iSCSI command PDU ("retry") in the absence of a command acknowledgement (by way of an ExpCmdSN update) or a response, an initiator attempts to "plug" (what it thinks are) the discontinuities in CmdSN ordering on the target end. Discarded command PDUs, due to digest errors, may have created these discontinuities.

Retry MUST NOT be used for reasons other than plugging command sequence gaps, and in particular, cannot be used for requesting PDU retransmissions from a target. Any such PDU retransmission requests for a currently allegiant command in progress may be made using the SNACK mechanism described in section 10.16, although the usage of SNACK is OPTIONAL.

If initiators, as part of plugging command sequence gaps as described above, inadvertently issue retries for allegiant commands already in progress (i.e., targets did not see the discontinuities in CmdSN ordering), the duplicate commands are silently ignored by targets as specified in section 3.2.2.1.

When an iSCSI command is retried, the command PDU MUST carry the original Initiator Task Tag and the original operational attributes (e.g., flags, function names, LUN, CDB etc.) as well as the original CmdSN. The command being retried MUST be sent on the same connection as the original command unless the original connection was already successfully logged out.

### 7.2.2. Allegiance Reassignment

By issuing a "task reassign" task management request (Section 10.5.1 - "Function"), the initiator signals its intent to continue an already active command (but with no current connection allegiance) as part of connection recovery. This means that a new connection allegiance is requested for the command, which seeks to associate it to the connection on which the task management request is being issued. Before the allegiance reassignment is attempted for a task, an implicit or explicit Logout with the reason code "remove the connection for recovery" ( see section 10.14) MUST be successfully completed for the previous connection to which the task was allegiant.

In reassigning connection allegiance for a command, the targets SHOULD continue the command from its current state. For example, when reassigning read commands, the target SHOULD take advantage of the ExpDataSN field provided by the Task Management function request (which must be set to zero if there was no data transfer) and bring the read command to completion by sending the remaining data and sending (or resending) the status. ExpDataSN acknowledges all data sent up to, but not including, the Data-In PDU and or R2T with DataSN (or R2TSN) equal to ExpDataSN. However, targets may choose to send/receive all unacknowledged data or all of the data on a reassignment of connection allegiance if unable to recover or maintain accurate an state. Initiators MUST NOT subsequently request data retransmission through Data SNACK for PDUs numbered less than ExpDataSN (i.e., prior to the acknowledged sequence number). For all types of commands, a reassignment request implies that the task is still considered in progress by the initiator and the target must conclude the task appropriately if the target returns the "Function Complete" response to the reassignment request. This might possibly involve retransmission of data/R2T/status PDUs as necessary, but MUST involve the (re)transmission of the status PDU.

It is OPTIONAL for targets to support the allegiance reassignment. This capability is negotiated via the ErrorRecoveryLevel text key during the login time. When a target does not support allegiance reassignment, it MUST respond with a Task Management response code of "Allegiance reassignment not supported". If allegiance reassignment is supported by the target, but the task is still allegiant to a different connection, or a successful recovery Logout of the previously allegiant connection was not performed, the target MUST respond with a Task Management response code of "Task still allegiant".

If allegiance reassignment is supported by the target, the Task Management response to the reassignment request MUST be issued before the reassignment becomes effective.

If a SCSI Command that involves data input is reassigned, any SNACK Tag it holds for a final response from the original



connection is deleted and the default value of 0 MUST be used instead.

### 7.3. Usage Of Reject PDU in Recovery

Targets MUST NOT implicitly terminate an active task by sending a Reject PDU for any PDU exchanged during the life of the task. If the target decides to terminate the task, a Response PDU (SCSI, Text, Task, etc.) must be returned by the target to conclude the task. If the task had never been active before the Reject (i.e., the Reject is on the command PDU), targets should not send any further responses because the command itself is being discarded.

The above rule means that the initiator can eventually expect a response on receiving Rejects, if the received Reject is for a PDU other than the command PDU itself. The non-command Rejects only have diagnostic value in logging the errors, and they can be used for retransmission decisions by the initiators.

The CmdSN of the rejected command PDU (if it is a non-immediate command) MUST NOT be considered received by the target (i.e., a command sequence gap must be assumed for the CmdSN), even though the CmdSN of the rejected command PDU may be reliably ascertained. Upon receiving the Reject, the initiator MUST plug the CmdSN gap in order to continue to use the session. The gap may be plugged either by transmitting a command PDU with the same CmdSN, or by aborting the task (see SCSI on how an abort may plug a CmdSN gap).

When a data PDU is rejected and its DataSN can be ascertained, a target MUST advance ExpDataSN for the current data burst if a recovery R2T is being generated. The target MAY advance its ExpDataSN if it does not attempt to recover the lost data PDU.

### 7.4. Error Recovery Considerations for Discovery Sessions

#### 7.4.1. ErrorRecoveryLevel for Discovery Sessions

The negotiation of the key ErrorRecoveryLevel is not required for Discovery sessions -- i.e., for sessions that negotiated "SessionType=Discovery" -- because the default value of 0 is necessary and sufficient for Discovery sessions. It is however possible that some legacy iSCSI implementations might attempt to

negotiate the `ErrorRecoveryLevel` key on Discovery sessions. When such a negotiation attempt is made by the remote side, a compliant iSCSI implementation **MUST** propose a value of 0 (zero) in response. The operational `ErrorRecoveryLevel` for Discovery sessions thus **MUST** be 0. This naturally follows from the functionality constraints that Section 4.3 imposes on Discovery sessions.

#### 7.4.2. Reinstatement Semantics for Discovery Sessions

Discovery sessions are intended to be relatively short-lived. Initiators are not expected to establish multiple Discovery sessions to the same iSCSI Network Portal. An initiator may use the same iSCSI Initiator Name and ISID when establishing different unique sessions with different targets and/or different portal groups. This behavior is discussed in Section 10.1.1 and is, in fact, encouraged as conservative reuse of ISIDs.

The ISID RULE in Section 4.4.3 states that there must not be more than one session with a matching 4-tuple: `<InitiatorName, ISID, TargetName, TargetPortalGroupTag>`. While the spirit of the ISID RULE applies to Discovery sessions the same as it does for Normal sessions, note that some Discovery sessions differ from the Normal sessions in two important aspects:

Because Appendix D allows a Discovery session to be established without specifying a `TargetName` key in the Login Request PDU (let us call such a session an "Unnamed" Discovery session), there is no Target Node context to enforce the ISID RULE.

Portal Groups are defined only in the context of a Target Node. When the `TargetName` key is NULL-valued (i.e., not specified), the `TargetPortalGroupTag` thus cannot be ascertained to enforce the ISID RULE.

The following two sections describe each of the two scenarios -- Named Discovery sessions and Unnamed Discovery sessions.

#### 7.4.2.1. Unnamed Discovery Sessions

For Unnamed Discovery sessions, neither the TargetName nor the TargetPortalGroupTag is available to the targets in order to enforce the ISID RULE. So the following rule applies.

UNNAMED ISID RULE: Targets MUST enforce the uniqueness of the following 4-tuple for Unnamed Discovery sessions: <InitiatorName, ISID, NULL, TargetAddress>. The following semantics are implied by this uniqueness requirement.

Targets SHOULD allow concurrent establishment of one Discovery session with each of its Network Portals by the same initiator port with a given iSCSI Node Name and an ISID. Each of the concurrent Discovery sessions, if established by the same initiator port to other Network Portals, MUST be treated as independent sessions -- i.e., one session MUST NOT reinstate the other.

A new Unnamed Discovery session that has a matching <InitiatorName, ISID, NULL, TargetAddress> to an existing Discovery session MUST reinstate the existing Unnamed Discovery session. Note thus that only an Unnamed Discovery session may reinstate an Unnamed Discovery session.

#### 7.4.2.2. Named Discovery Session

For a Named Discovery session, the TargetName key is specified by the initiator and thus the target can unambiguously ascertain the TargetPortalGroupTag as well. Since all the four elements of the 4-tuple are known, the ISID RULE MUST be enforced by targets with no changes from Section 4.4.3 semantics. A new session with a matching <InitiatorName, ISID, TargetName, TargetPortalGroupTag> thus will reinstate an existing session. Note in this case that any new iSCSI session (Discovery or Normal) with the matching 4-tuple may reinstate an existing Named Discovery iSCSI session.

#### 7.4.3. Target PDUs During Discovery

Targets SHOULD NOT send any responses other than a Text Response and Logout Response on a Discovery session, once in Full Feature Phase.

Implementation Note: A target may simply drop the connection in a Discovery session when it would have requested a Logout via an Async Message on Normal sessions.

## 7.5. Connection Timeout Management

iSCSI defines two session-global timeout values (in seconds) - Time2Wait and Time2Retain - that are applicable when an iSCSI Full Feature Phase connection is taken out of service either intentionally or by an exception. Time2Wait is the initial "respite time" before attempting an explicit/implicit Logout for the CID in question or task reassignment for the affected tasks (if any). Time2Retain is the maximum time after the initial respite interval that the task and/or connection state(s) is/are guaranteed to be maintained on the target to cater to a possible recovery attempt. Recovery attempts for the connection and/or task(s) SHOULD NOT be made before Time2Wait seconds, but MUST be completed within Time2Retain seconds after that initial Time2Wait waiting period.

### 7.5.1. Timeouts on Transport Exception Events

A transport connection shutdown or a transport reset without any preceding iSCSI protocol interactions informing the end-points of the fact causes a Full Feature Phase iSCSI connection to be abruptly terminated. The timeout values to be used in this case are the negotiated values of defaultTime2Wait (Section 12.15 - "DefaultTime2Wait") and DefaultTime2Retain (Section 12.16 - "DefaultTime2Retain") text keys for the session.

### 7.5.2. Timeouts on Planned Decommissioning

Any planned decommissioning of a Full Feature Phase iSCSI connection is preceded by either a Logout Response PDU, or an Async Message PDU. The Time2Wait and Time2Retain field values (section 10.15) in a Logout Response PDU, and the Parameter2 and Parameter3 fields of an Async Message (AsyncEvent types "drop the connection" or "drop all the connections"; section 10.9.1) specify the timeout values to be used in each of these cases.

These timeout values are only applicable for the affected connection, and the tasks active on that connection. These timeout values have no bearing on initiator timers (if any) that are already running on connections or tasks associated with that session.

#### 7.6. Implicit Termination of Tasks

A target implicitly terminates the active tasks due to iSCSI protocol dynamics in the following cases:

When a connection is implicitly or explicitly logged out with the reason code of "Close the connection" and there are active tasks allegiant to that connection.

When a connection fails and eventually the connection state times out (state transition M1 in Section 7.2.2 - "State Transition Descriptions for Initiators and Targets") and there are active tasks allegiant to that connection.

When a successful Logout with the reason code of "remove the connection for recovery" is performed while there are active tasks allegiant to that connection, and those tasks eventually time out after the Time2Wait and Time2Retain periods without allegiance reassignment.

When a connection is implicitly or explicitly logged out with the reason code of "Close the session" and there are active tasks in that session.

If the tasks terminated in the above cases a), b, c) and d) are SCSI tasks, they must be internally terminated as if with CHECK CONDITION status. This status is only meaningful for appropriately handling the internal SCSI state and SCSI side effects with respect to ordering because this status is never communicated back as a terminating status to the initiator. However additional actions may have to be taken at SCSI level depending on the SCSI context as defined by the SCSI standards (e.g., queued commands and ACA, UA for the next command on the I\_T nexus in cases a), b), and c) etc. - see [SAM2] and [SPC3]).

## 7.7. Format Errors

The following two explicit violations of PDU layout rules are format errors:

- Illegal contents of any PDU header field except the Opcode (legal values are specified in Section 10 - "iSCSI PDU Formats").
- Inconsistent field contents (consistent field contents are specified in Section 10 - "iSCSI PDU Formats").

Format errors indicate a major implementation flaw in one of the parties.

When a target or an initiator receives an iSCSI PDU with a format error, it MUST immediately terminate all transport connections in the session either with a connection close or with a connection reset and escalate the format error to session recovery (see Section Session Recovery).

All initiator-detected PDU construction errors MUST be considered as format errors. Some examples of such errors are:

- NOP-In with a valid TTT but an invalid LUN
- NOP-In with a valid ITT (i.e., a NOP-In response) and also a valid TTT
- SCSI Response PDU with Status=CHECK CONDITION, but DataSegmentLength = 0

## 7.8. Digest Errors

The discussion of the legal choices in handling digest errors below excludes session recovery as an explicit option, but either party detecting a digest error may choose to escalate the error to session recovery.

When a target or an initiator receives any iSCSI PDU, with a header digest error, it MUST either discard the header and all data up to the beginning of a later PDU or close the connection. Because the digest error indicates that the length field of the

header may have been corrupted, the location of the beginning of a later PDU needs to be reliably ascertained by other means such as the operation of a sync and steering layer.

When a target receives any iSCSI PDU with a payload digest error, it MUST answer with a Reject PDU with a reason code of Data-Digest-Error and discard the PDU.

- If the discarded PDU is a solicited or unsolicited iSCSI data PDU (for immediate data in a command PDU, non-data PDU rule below applies), the target MUST do one of the following:
  - i) Request retransmission with a recovery R2T.
  - ii) Terminate the task with a response PDU with a CHECK CONDITION Status and an iSCSI Condition of "protocol service CRC error" (Section 10.4.7.2 - "Sense Data"). If the target chooses to implement this option, it MUST wait to receive all the data (signaled by a Data PDU with the final bit set for all outstanding R2Ts) before sending the response PDU. A task management command (such as an abort task) from the initiator during this wait may also conclude the task.
- No further action is necessary for targets if the discarded PDU is a non-data PDU. In case of immediate data being present on a discarded command, the immediate data is implicitly recovered when the task is retried (see Section 7.2.1) followed by the entire data transfer for the task.

When an initiator receives any iSCSI PDU with a payload digest error, it MUST discard the PDU.

- If the discarded PDU is an iSCSI data PDU, the initiator MUST do one of the following:

Request the desired data PDU through SNACK. In response to the SNACK, the target MUST either resend the data PDU or reject the SNACK with a Reject PDU with a reason code of "SNACK reject" in which case:

If the status has not already been sent for the command, the target MUST terminate the command with a CHECK CONDITION Status and an iSCSI Condition of "SNACK rejected" (Section 10.4.7.2 - "Sense Data").

If the status was already sent, no further action is necessary for the target. The initiator in this case MUST wait for the status to be received and then discard it, so as to internally signal the completion with CHECK CONDITION Status and an iSCSI Condition of "protocol service CRC error" (Section 10.4.7.2 - "Sense Data").

Abort the task and terminate the command with an error.

- If the discarded PDU is a response PDU or an unsolicited PDU (e.g. Async, Reject), the initiator MUST do one of the following:

- Request PDU retransmission with a status SNACK.

- Logout the connection for recovery and continue the tasks on a different connection instance as described in Retry an.

- Logout to close the connection (abort all the commands associated with the connection).

Note that an unsolicited PDU carries the next StatSN value on an iSCSI connection, thereby advancing the StatSN. When an initiator discards one of these PDUs due to a payload digest error, the entire PDU including the header MUST be discarded. Consequently, the initiator MUST treat the exception like a loss of any other solicited response PDU.

## 7.9. Sequence Errors

When an initiator receives an iSCSI R2T/data PDU with an out of order R2TSN/DataSN or a SCSI response PDU with an ExpDataSN that implies missing data PDU(s), it means that the initiator must have detected a header or payload digest error on one or more earlier R2T/data PDUs. The initiator MUST address these implied digest errors as described in Section 7.8. When a target receives a data PDU with an out of order DataSN, it means that the target must



have hit a header or payload digest error on at least one of the earlier data PDUs. The target MUST address these implied digest errors as described in Section 7.8.

When an initiator receives an iSCSI status PDU with an out of order StatSN that implies missing responses, it MUST address the one or more missing status PDUs as described in Section 7.8. As a side effect of receiving the missing responses, the initiator may discover missing data PDUs. If the initiator wants to recover the missing data for a command, it MUST NOT acknowledge the received responses that start from the StatSN of the relevant command, until it has completed receiving all the data PDUs of the command.

When an initiator receives duplicate R2TSNs (due to proactive retransmission of R2Ts by the target) or duplicate DataSNs (due to proactive SNACKs by the initiator), it MUST discard the duplicates.

#### 7.10. Message Error Checking

In the iSCSI implementations till date, there has been some uncertainty on the extent to which incoming messages have to be checked for protocol errors, beyond what is strictly required for processing the inbound message. This section addresses this question.

Unless this document requires it, an iSCSI implementation is not required to do an exhaustive protocol conformance check on an incoming iSCSI PDU. The iSCSI implementation especially is not required to double-check the remote iSCSI implementation's conformance to protocol requirements.

#### 7.11. SCSI Timeouts

An iSCSI initiator MAY attempt to plug a command sequence gap on the target end (in the absence of an acknowledgement of the command by way of ExpCmdSN) before the ULP timeout by retrying the unacknowledged command, as described in Section 7.2.

On a ULP timeout for a command (that carried a CmdSN of n), if the iSCSI initiator intends to continue the session it MUST abort the command by either using an appropriate Task Management function

request for the specific command, or a "close the connection" Logout. When using an ABORT TASK, if the ExpCmdSN is still less than (n+1), the target may see the abort request while missing the original command itself due to one of the following reasons:

- Original command was dropped due to digest error.
- Connection on which the original command was sent was successfully logged out. On logout, the unacknowledged commands issued on the connection being logged out are discarded.

If the abort request is received and the original command is missing, targets MUST consider the original command with that RefCmdSN to be received and issue a Task Management response with the response code: "Function Complete". This response concludes the task on both ends. If the abort request is received and the target can determine (based on the Referenced Task Tag) that the command was received and executed and also that the response was sent prior to the abort, then the target MUST respond with the response code of "Task Does Not Exist".

#### 7.12. Negotiation Failures

Text request and response sequences, when used to set/negotiate operational parameters, constitute the negotiation/parameter setting. A negotiation failure is considered to be one or more of the following:

- None of the choices, or the stated value, is acceptable to one of the sides in the negotiation.
- The text request timed out and possibly terminated.
- The text request was answered with a Reject PDU.

The following two rules should be used to address negotiation failures:

- During Login, any failure in negotiation MUST be considered a login process failure and the Login Phase must be terminated, and with it, the connection. If the target detects the failure, it must terminate the login with the appropriate login response code.
  
- A failure in negotiation, while in the Full Feature Phase, will terminate the entire negotiation sequence that may consist of a series of text requests that use the same Initiator Task Tag. The operational parameters of the session or the connection MUST continue to be the values agreed upon during an earlier successful negotiation (i.e., any partial results of this unsuccessful negotiation MUST NOT take effect and MUST be discarded).

#### 7.13. Protocol Errors

Mapping framed messages over a "stream" connection, such as TCP, makes the proposed mechanisms vulnerable to simple software framing errors. On the other hand, the introduction of framing mechanisms to limit the effects of these errors may be onerous on performance for simple implementations. Command Sequence Numbers and the above mechanisms for connection drop and reestablishment help handle this type of mapping errors.

All violations of iSCSI PDU exchange sequences specified in this draft are also protocol errors. This category of errors can only be addressed by fixing the implementations; iSCSI defines Reject and response codes to enable this.

#### 7.14. Connection Failures

iSCSI can keep a session in operation if it is able to keep/establish at least one TCP connection between the initiator and the target in a timely fashion. Targets and/or initiators may recognize a failing connection by either transport level means (TCP), a gap in the command sequence number, a response stream that is not filled for a long time, or by a failing iSCSI NOP

(acting as a ping). The latter MAY be used periodically to increase the speed and likelihood of detecting connection failures. Initiators and targets MAY also use the keep-alive option on the TCP connection to enable early link failure detection on otherwise idle links.

On connection failure, the initiator and target MUST do one of the following:

- Attempt connection recovery within the session (Connection Recovery).
- Logout the connection with the reason code "closes the connection" (Section 10.14.5 - "Implicit termination of tasks"), re-issue missing commands, and implicitly terminate all active commands. This option requires support for the within-connection recovery class (Recovery Within-connection).
- Perform session recovery (Session Recovery).

Either side may choose to escalate to session recovery (via the initiator dropping all the connections, or via an Async Message that announces the similar intent from a target), and the other side MUST give it precedence. On a connection failure, a target MUST terminate and/or discard all of the active immediate commands regardless of which of the above options is used (i.e., immediate commands are not recoverable across connection failures).

#### 7.15. Session Errors

If all of the connections of a session fail and cannot be reestablished in a short time, or if initiators detect protocol errors repeatedly, an initiator may choose to terminate a session and establish a new session.

In this case, the initiator takes the following actions:

- Resets or closes all the transport connections.
- Terminates all outstanding requests with an appropriate response before initiating a new session. If the same I\_T nexus is intended to be reestablished, the initiator MUST employ session reinstatement (see section 5.3.5).

When the session timeout (the connection state timeout for the last failed connection) happens on the target, it takes the following actions:

- Resets or closes the TCP connections (closes the session).
- Terminates all active tasks that were allegiant to the connection(s) that constituted the session.

A target MUST also be prepared to handle a session reinstatement request from the initiator, that may be addressing session errors.

## 8. State Transitions

iSCSI connections and iSCSI sessions go through several well-defined states from the time they are created to the time they are cleared.

The connection state transitions are described in two separate but dependent state diagrams for ease in understanding. The first diagram, "standard connection state diagram", describes the connection state transitions when the iSCSI connection is not waiting for, or undergoing, a cleanup by way of an explicit or implicit Logout. The second diagram, "connection cleanup state diagram", describes the connection state transitions while performing the iSCSI connection cleanup.

The "session state diagram" describes the state transitions an iSCSI session would go through during its lifetime, and it depends on the states of possibly multiple iSCSI connections that participate in the session.

States and transitions are described in text, tables and diagrams. The diagrams are used for illustration. The text and the tables are the governing specification.

### 8.1. Standard Connection State Diagrams

#### 8.1.1. State Descriptions for Initiators and Targets

State descriptions for the standard connection state diagram are as follows:

-S1: FREE

-initiator: State on instantiation, or after successful connection closure.

-target: State on instantiation, or after successful connection closure.

-S2: XPT\_WAIT

-initiator: Waiting for a response to its transport connection establishment request.

-target: Illegal

-S3: XPT\_UP

-initiator: Illegal

-target: Waiting for the Login process to commence.

- S4: IN\_LOGIN
  - initiator: Waiting for the Login process to conclude, possibly involving several PDU exchanges.
  - target: Waiting for the Login process to conclude, possibly involving several PDU exchanges.
- S5: LOGGED\_IN
  - initiator: In Full Feature Phase, waiting for all internal, iSCSI, and transport events.
  - target: In Full Feature Phase, waiting for all internal, iSCSI, and transport events.
- S6: IN\_LOGOUT
  - initiator: Waiting for a Logout response.
  - target: Waiting for an internal event signaling completion of logout processing.
- S7: LOGOUT\_REQUESTED
  - initiator: Waiting for an internal event signaling readiness to proceed with Logout.
  - target: Waiting for the Logout process to start after having requested a Logout via an Async Message.
- S8: CLEANUP\_WAIT
  - initiator: Waiting for the context and/or resources to initiate the cleanup processing for this CSM.
  - target: Waiting for the cleanup process to start for this CSM.

#### 8.1.2. State Transition Descriptions for Initiators and Targets

- T1:
  - initiator: Transport connect request was made (e.g., TCP SYN sent).
  - target: Illegal
- T2:
  - initiator: Transport connection request timed out, a transport reset was received, or an internal event of receiving a Logout response (success) on another connection for a "close the session" Logout request was received.
  - target: Illegal
- T3:
  - initiator: Illegal
  - target: Received a valid transport connection request that establishes the transport connection.
- T4:

-initiator: Transport connection established, thus prompting the initiator to start the iSCSI Login.  
-target: Initial iSCSI Login request was received.

-T5:

-initiator: The final iSCSI Login response with a Status-Class of zero was received.  
-target: The final iSCSI Login request to conclude the Login Phase was received, thus prompting the target to send the final iSCSI Login response with a Status-Class of zero.

-T6:

-initiator: Illegal  
-target: Timed out waiting for an iSCSI Login, transport disconnect indication was received, transport reset was received, or an internal event indicating a transport timeout was received. In all these cases, the connection is to be closed.

-T7:

-initiator - one of the following events caused the transition:  
- The final iSCSI Login response was received with a non-zero Status-Class.  
- Login timed out.  
- A transport disconnect indication was received.  
- A transport reset was received.  
- An internal event indicating a transport timeout was received.  
- An internal event of receiving a Logout response (success) on another connection for a "close the session" Logout request was received.

In all these cases, the transport connection is closed.

-target - one of the following events caused the transition:  
- The final iSCSI Login request to conclude the Login Phase was received, prompting the target to send the final iSCSI Login response with a non-zero Status-Class.  
- Login timed out.  
- Transport disconnect indication was received.



- Transport reset was received.
- An internal event indicating a transport timeout was received .
- On another connection a "close the session" Logout request was received.

In all these cases, the connection is to be closed.

-T8:

- initiator: An internal event of receiving a Logout response (success) on another connection for a "close the session" Logout request was received, thus closing this connection requiring no further cleanup.
- target: An internal event of sending a Logout response (success) on another connection for a "close the session" Logout request was received, or an internal event of a successful connection/session reinstatement is received, thus prompting the target to close this connection cleanly.

-T9, T10:

- initiator: An internal event that indicates the readiness to start the Logout process was received, thus prompting an iSCSI Logout to be sent by the initiator.
- target: An iSCSI Logout request was received.

-T11, T12:

- initiator: Async PDU with AsyncEvent "Request Logout" was received.
- target: An internal event that requires the decommissioning of the connection is received, thus causing an Async PDU with an AsyncEvent "Request Logout" to be sent.

-T13:

- initiator: An iSCSI Logout response (success) was received, or an internal event of receiving a Logout response (success) on another connection for a "close the session" Logout request was received.
- target: An internal event was received that indicates successful processing of the Logout, which prompts an iSCSI Logout response (success) to be sent; an internal event of sending a Logout response (success) on another connection for a "close the session" Logout request was received; or an internal event of a successful connection/session

reinstatement is received. In all these cases, the transport connection is closed.

-T14:

- initiator: Async PDU with AsyncEvent "Request Logout" was received again.
- target: Illegal

-T15, T16:

- initiator: One or more of the following events caused this transition:

- Internal event that indicates a transport connection timeout was received thus prompting transport RESET or transport connection closure.
- A transport RESET.
- A transport disconnect indication.
- Async PDU with AsyncEvent "Drop connection" (for this CID).
- Async PDU with AsyncEvent "Drop all connections".

- target: One or more of the following events caused this transition:

- Internal event that indicates a transport connection timeout was received, thus prompting transport RESET or transport connection closure.
- An internal event of a failed connection/session reinstatement is received.
- A transport RESET.
- A transport disconnect indication.
- Internal emergency cleanup event was received which prompts an Async PDU with AsyncEvent "Drop connection" (for this CID), or event "Drop all connections".

-T17:

- initiator: One or more of the following events caused this transition:

- Logout response, (failure i.e., a non-zero status) was received, or Logout timed out.
- Any of the events specified for T15 and T16.

- target: One or more of the following events caused this transition:

- Internal event that indicates a failure of the Logout processing was received, which prompts a Logout response (failure, i.e., a non-zero status) to be sent.
- Any of the events specified for T15 and T16.

-T18:

-initiator: An internal event of receiving a Logout response (success) on another connection for a "close the session" Logout request was received.

-target: An internal event of sending a Logout response (success) on another connection for a "close the session" Logout request was received, or an internal event of a successful connection/session reinstatement is received. In both these cases, the connection is closed.

The CLEANUP\_WAIT state (S8) implies that there are possible iSCSI tasks that have not reached conclusion and are still considered busy.

#### 8.1.3. Standard Connection State Diagram for an Initiator

Symbolic names for States:

S1: FREE

S2: XPT\_WAIT

S4: IN\_LOGIN

S5: LOGGED\_IN

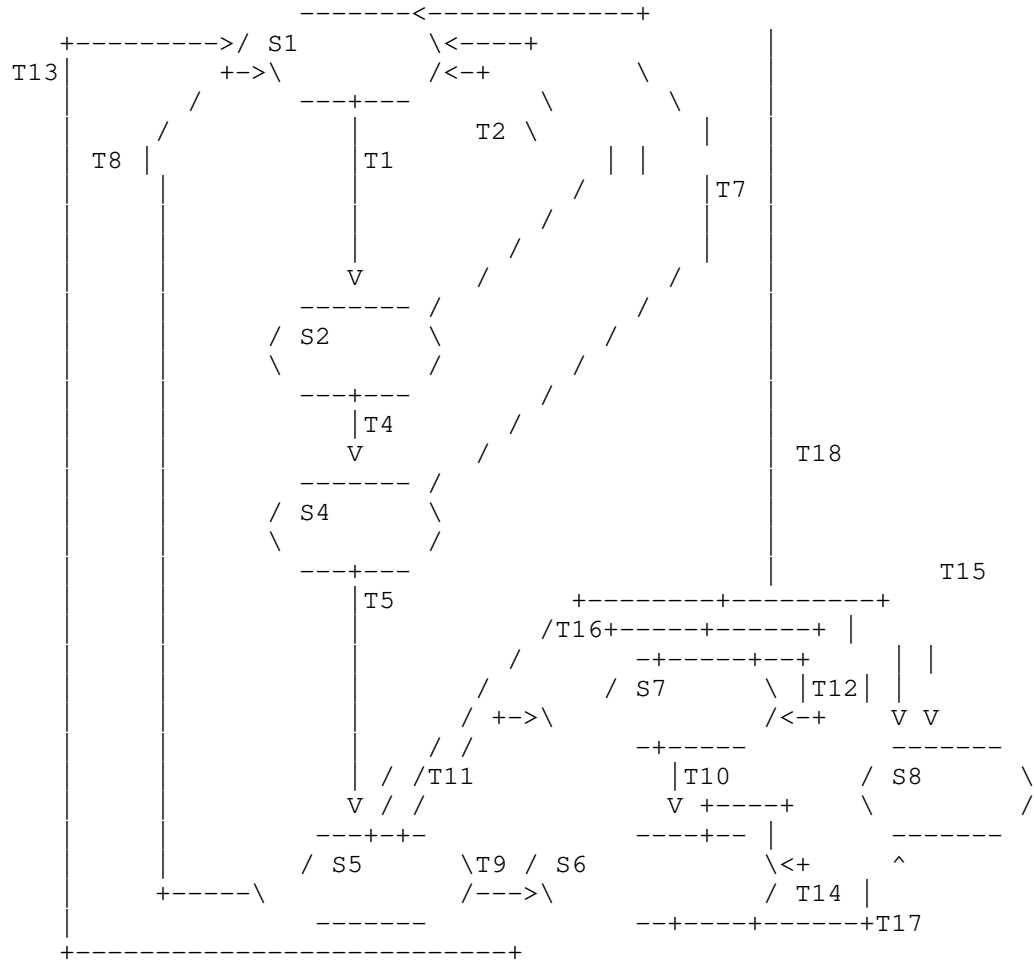
S6: IN\_LOGOUT

S7: LOGOUT\_REQUESTED

S8: CLEANUP\_WAIT

States S5, S6, and S7 constitute the Full Feature Phase operation of the connection.

The state diagram is as follows:



The following state transition table represents the above diagram. Each row represents the starting state for a given transition,

which after taking a transition marked in a table cell would end in the state represented by the column of the cell. For example, from state S1, the connection takes the T1 transition to arrive at state S2. The fields marked "-" correspond to undefined transitions.

	S1	S2	S4	S5	S6	S7	S8
S1	-	T1	-	-	-	-	-
S2	T2	-	T4	-	-	-	-
S4	T7	-	-	T5	-	-	-
S5	T8	-	-	-	T9	T11	T15
S6	T13	-	-	-	T14	-	T17
S7	T18	-	-	-	T10	T12	T16
S8	-	-	-	-	-	-	-

#### 8.1.4. Standard Connection State Diagram for a Target

Symbolic names for States:

S1: FREE

S3: XPT\_UP

S4: IN\_LOGIN

S5: LOGGED\_IN

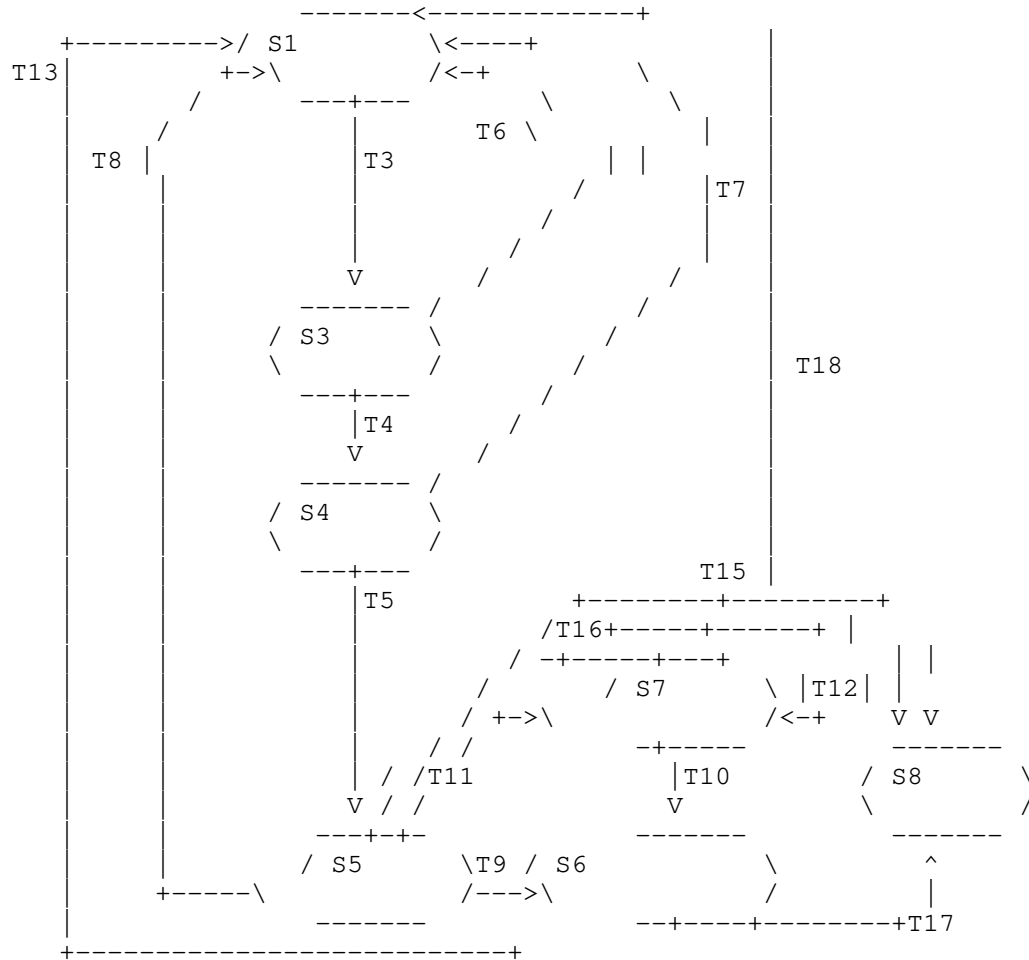
S6: IN\_LOGOUT

S7: LOGOUT\_REQUESTED

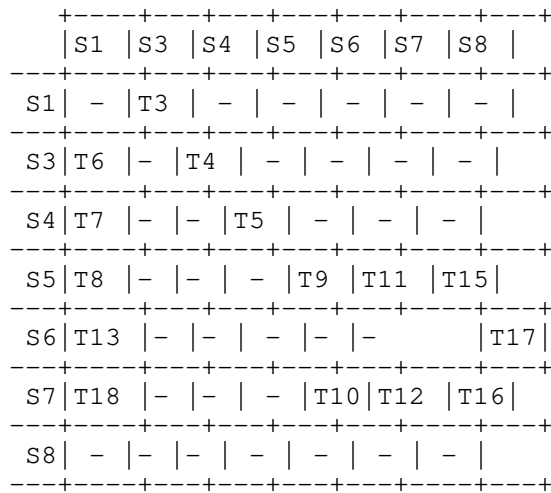
S8: CLEANUP\_WAIT

States S5, S6, and S7 constitute the Full Feature Phase operation of the connection.

The state diagram is as follows:



The following state transition table represents the above diagram, and follows the conventions described for the initiator diagram.



## 8.2. Connection Cleanup State Diagram for Initiators and Targets

Symbolic names for states:

R1: CLEANUP\_WAIT (same as S8)

R2: IN\_CLEANUP

R3: FREE (same as S1)

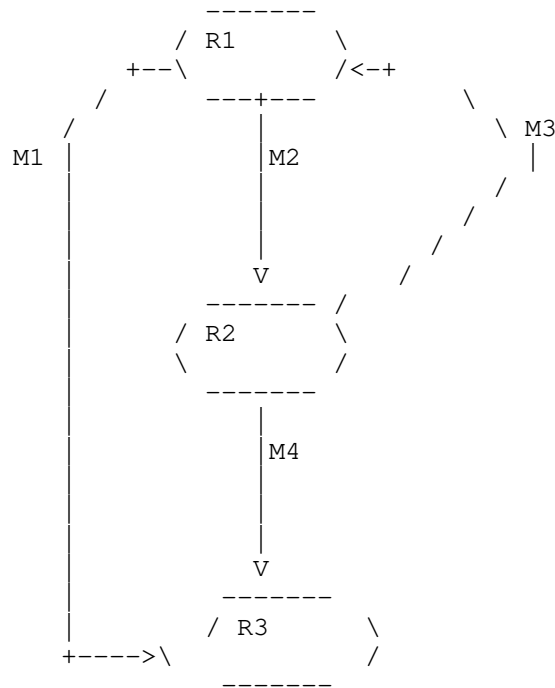
Whenever a connection state machine in cleanup (let's call it CSM-C) enters the CLEANUP\_WAIT state (S8), it must go through the state transitions described in the connection cleanup state diagram either a) using a separate full-feature phase connection (let's call it CSM-E, for explicit) in the LOGGED\_IN state in the same session, or b) using a new transport connection (let's call it CSM-I, for implicit) in the FREE state that is to be added to the same session. In the CSM-E case, an explicit logout for the CID that corresponds to CSM-C (either as a connection or session logout) needs to be performed to complete the cleanup. In the CSM-I case, an implicit logout for the CID that corresponds to CSM-C

needs to be performed by way of connection reinstatement (section 5.3.4) for that CID. In either case, the protocol exchanges on CSM-E or CSM-I determine the state transitions for CSM-C. Therefore, this cleanup state diagram is only applicable to the instance of the connection in cleanup (i.e., CSM-C). In the case of an implicit logout for example, CSM-C reaches FREE (R3) at the time CSM-I reaches LOGGED\_IN. In the case of an explicit logout, CSM-C reaches FREE (R3) when CSM-E receives a successful logout response while continuing to be in the LOGGED\_IN state.

An initiator must initiate an explicit or implicit connection logout for a connection in the CLEANUP\_WAIT state, if the initiator intends to continue using the associated iSCSI session.

The following state diagram applies to both initiators and targets.





The following state transition table represents the above diagram, and follows the same conventions as in earlier sections.

	R1	R2	R3
R1	-	M2	M1
R2	M3	-	M4
R3	-	-	-

#### 8.2.1. State Descriptions for Initiators and Targets

-R1: CLEANUP\_WAIT (Same as S8)

- initiator: Waiting for the internal event to initiate the cleanup processing for CSM-C.
- target: Waiting for the cleanup process to start for CSM-C.
- R2: IN\_CLEANUP
  - initiator: Waiting for the connection cleanup process to conclude for CSM-C.
  - target: Waiting for the connection cleanup process to conclude for CSM-C.
- R3: FREE (Same as S1)
  - initiator: End state for CSM-C.
  - target: End state for CSM-C.

#### 8.2.2. State Transition Descriptions for Initiators and Targets

- M1: One or more of the following events was received:
  - initiator:
    - An internal event that indicates connection state timeout.
    - An internal event of receiving a successful Logout response on a different connection for a "close the session" Logout.
  - target:
    - An internal event that indicates connection state timeout.
    - An internal event of sending a Logout response (success) on a different connection for a "close the session" Logout request.
- M2: An implicit/explicit logout process was initiated by the initiator.
  - In CSM-I usage:
    - initiator: An internal event requesting the connection (or session) reinstatement was received, thus prompting a connection (or session) reinstatement Login to be sent transitioning CSM-I to state IN\_LOGIN.
    - target: A connection/session reinstatement Login was received while in state XPT\_UP.
  - In CSM-E usage:

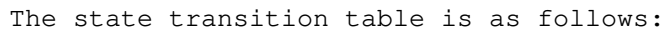
- initiator: An internal event that indicates that an explicit logout was sent for this CID in state LOGGED\_IN.
  - target: An explicit logout was received for this CID in state LOGGED\_IN.
- M3: Logout failure detected
- In CSM-I usage:
    - initiator: CSM-I failed to reach LOGGED\_IN and arrived into FREE instead.
    - target: CSM-I failed to reach LOGGED\_IN and arrived into FREE instead.
  - In CSM-E usage:
    - initiator: CSM-E either moved out of LOGGED\_IN, or Logout timed out and/or aborted, or Logout response (failure) was received.
    - target: CSM-E either moved out of LOGGED\_IN, Logout timed out and/or aborted, or an internal event that indicates a failed Logout processing was received. A Logout response (failure) was sent in the last case.
- M4: Successful implicit/explicit logout was performed.
- In CSM-I usage:
    - initiator: CSM-I reached state LOGGED\_IN, or an internal event of receiving a Logout response (success) on another connection for a "close the session" Logout request was received.
    - target: CSM-I reached state LOGGED\_IN, or an internal event of sending a Logout response (success) on a different connection for a "close the session" Logout request was received.
  - In CSM-E usage:
    - initiator: CSM-E stayed in LOGGED\_IN and received a Logout response (success), or an internal event of receiving a Logout response (success) on another connection for a "close the session" Logout request was received.
    - target: CSM-E stayed in LOGGED\_IN and an internal event indicating a successful Logout processing was received, or an internal event of sending a Logout

### 8.3.1. Session State Diagram for an Initiator

Q1: FREE

Q4 : FAILED

The state diagram is as follows:



	Q1	Q3	Q4	
Q1	-	N1	-	
Q3	N3	-	N5	
Q4	N6	N4	-	

### 8.3.2. Session State Diagram for a Target

Symbolic Names for States:

Q1: FREE

Q2: ACTIVE

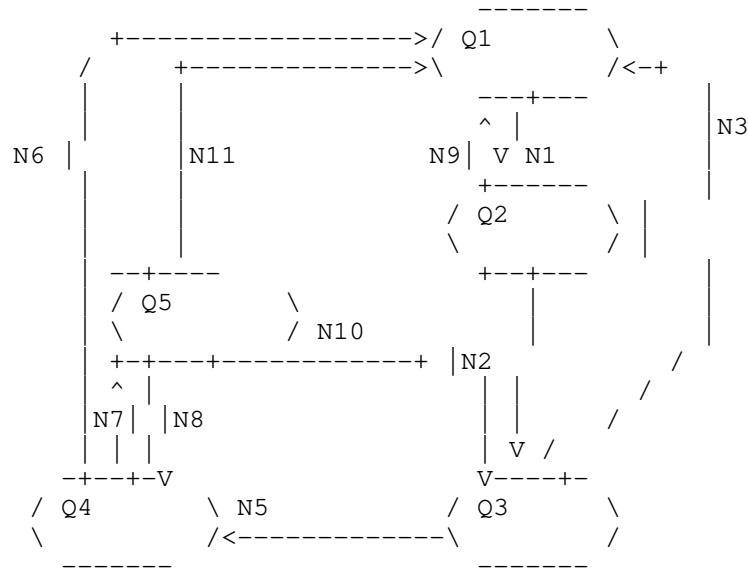
Q3: LOGGED\_IN

Q4: FAILED

Q5: IN\_CONTINUE

State Q3 represents the Full Feature Phase operation of the session.

The state diagram is as follows:



The state transition table is as follows:

	Q1	Q2	Q3	Q4	Q5
Q1	-	N1	-	-	-
Q2	N9	-	N2	-	-
Q3	N3	-	-	N5	-
Q4	N6	-	-	-	N7
Q5	N11	-	N10	N8	-

### 8.3.3. State Descriptions for Initiators and Targets

-Q1: FREE

-initiator: State on instantiation or after cleanup.

- target: State on instantiation or after cleanup.
- Q2: ACTIVE
  - initiator: Illegal.
  - target: The first iSCSI connection in the session transitioned to IN\_LOGIN, waiting for it to complete the login process.
- Q3: LOGGED\_IN
  - initiator: Waiting for all session events.
  - target: Waiting for all session events.
- Q4: FAILED
  - initiator: Waiting for session recovery or session continuation.
  - target: Waiting for session recovery or session continuation.
- Q5: IN\_CONTINUE
  - initiator: Illegal.
  - target: Waiting for session continuation attempt to reach a conclusion.

#### 8.3.4. State Transition Descriptions for Initiators and Targets

- N1:
  - initiator: At least one transport connection reached the LOGGED\_IN state.
  - target: The first iSCSI connection in the session had reached the IN\_LOGIN state.
- N2:
  - initiator: Illegal.
  - target: At least one iSCSI connection reached the LOGGED\_IN state.
- N3:
  - initiator: Graceful closing of the session via session closure (Section 5.3.6 - "Session Continuation and Failure").
  - target: Graceful closing of the session via session closure (Section 5.3.6 - "Session Continuation and Failure") or a successful session reinstatement cleanly closed the session.
- N4:
  - initiator: A session continuation attempt succeeded.

- target: Illegal.
- N5:
  - initiator: Session failure (Section 5.3.6 - "Session Continuation and Failure") occurred.
  - target: Session failure (Section 5.3.6 - "Session Continuation and Failure") occurred.
- N6:
  - initiator: Session state timeout occurred, or a session reinstatement cleared this session instance. This results in the freeing of all associated resources and the session state is discarded.
  - target: Session state timeout occurred, or a session reinstatement cleared this session instance. This results in the freeing of all associated resources and the session state is discarded.
- N7:
  - initiator: Illegal.
  - target: A session continuation attempt is initiated.
- N8:
  - initiator: Illegal.
  - target: The last session continuation attempt failed.
- N9:
  - initiator: Illegal.
  - target: Login attempt on the leading connection failed.
- N10:
  - initiator: Illegal.
  - target: A session continuation attempt succeeded.
- N11:
  - initiator: Illegal.
  - target: A successful session reinstatement cleanly closed the session.



## 9. Security Considerations

Historically, native storage systems have not had to consider security because their environments offered minimal security risks. That is, these environments consisted of storage devices either directly attached to hosts or connected via a Storage Area Network (SAN) distinctly separate from the communications network. The use of storage protocols, such as SCSI, over IP-networks requires that security concerns be addressed. iSCSI implementations MUST provide means of protection against active attacks (e.g., pretending to be another identity, message insertion, deletion, modification, and replaying) and passive attacks (e.g., eavesdropping, gaining advantage by analyzing the data sent over the line).

Although technically possible, iSCSI SHOULD NOT be configured without security. iSCSI configured without security should be confined, in extreme cases, to closed environments without any security risk. [RFC3723] specifies the mechanisms that must be used in order to mitigate risks fully described in that document.

The following section describes the security mechanisms provided by an iSCSI implementation.

### 9.1. iSCSI Security Mechanisms

The entities involved in iSCSI security are the initiator, target, and the IP communication end points. iSCSI scenarios in which multiple initiators or targets share a single communication end point are expected. To accommodate such scenarios, iSCSI uses two separate security mechanisms: In-band authentication between the initiator and the target at the iSCSI connection level (carried out by exchange of iSCSI Login PDUs), and packet protection (integrity, authentication, and confidentiality) by IPsec at the IP level. The two security mechanisms complement each other. The in-band authentication provides end-to-end trust (at login time) between the iSCSI initiator and the target while IPsec provides a secure channel between the IP communication end points.

Further details on typical iSCSI scenarios and the relation between the initiators, targets, and the communication end points can be found in [RFC3723].

## 9.2. In-band Initiator-Target Authentication

During login, the target MUST authenticate the initiator and the initiator MAY authenticate the target. The authentication is performed on every new iSCSI connection by an exchange of iSCSI Login PDUs using a negotiated authentication method.

The authentication method cannot assume an underlying IPsec protection, because IPsec is optional to use. An attacker should gain as little advantage as possible by inspecting the authentication phase PDUs. Therefore, a method using clear text (or equivalent) passwords is not acceptable; on the other hand, identity protection is not strictly required.

The authentication mechanism protects against an unauthorized login to storage resources by using a false identity (spoofing). Once the authentication phase is completed, if the underlying IPsec is not used, all PDUs are sent and received in clear. The authentication mechanism alone (without underlying IPsec) should only be used when there is no risk of eavesdropping, message insertion, deletion, modification, and replaying.

Section 11 - "iSCSI Security Text Keys and Authentication Methods" defines several authentication methods and the exact steps that must be followed in each of them, including the iSCSI-text-keys and their allowed values in each step. Whenever an iSCSI initiator gets a response whose keys, or their values, are not according to the step definition, it MUST abort the connection. Whenever an iSCSI target gets a response whose keys, or their values, are not according to the step definition, it MUST answer with a Login reject with the "Initiator Error" or "Missing Parameter" status. These statuses are not intended for cryptographically incorrect values such as the CHAP response, for which "Authentication Failure" status MUST be specified. The importance of this rule can be illustrated in CHAP with target authentication (see Section 11.1.4 - "Challenge Handshake Authentication Protocol (CHAP)") where the initiator would have been able to conduct a reflection attack by omitting his response key (CHAP\_R) using the same CHAP challenge as the target and reflecting the target's response back to the target. In CHAP, this is prevented because the target must

answer the missing CHAP\_R key with a Login reject with the "Missing Parameter" status.

For some of the authentication methods, a key specifies the identity of the iSCSI initiator or target for authentication purposes. The value associated with that key MAY be different from the iSCSI name and SHOULD be configurable. (CHAP\_N, see Section 11.1.4 - "Challenge Handshake Authentication Protocol (CHAP)" and SRP\_U, see Section 11.1.3 - "Secure Remote Password (SRP)").

#### 9.2.1. CHAP Considerations

Compliant iSCSI initiators and targets MUST implement the CHAP authentication method [RFC1994] (according to Section 11.1.4 - "Challenge Handshake Authentication Protocol (CHAP)" including the target authentication option).

When CHAP is performed over a non-encrypted channel, it is vulnerable to an off-line dictionary attack. Implementations MUST support use of up to 128 bit random CHAP secrets, including the means to generate such secrets and to accept them from an external generation source. Implementations MUST NOT provide secret generation (or expansion) means other than random generation.

An administrative entity of an environment in which CHAP is used with a secret that has less than 96 random bits MUST enforce IPsec encryption (according to the implementation requirements in Confidentiality) to protect the connection. Moreover, in this case IKE authentication with group pre-shared cryptographic keys SHOULD NOT be used unless it is not essential to protect group members against off-line dictionary attacks by other members.

CHAP secrets MUST be an integral number of bytes (octets). A compliant implementation SHOULD NOT continue with the login step in which it should send a CHAP response (CHAP\_R, Section 11.1.4 - "Challenge Handshake Authentication Protocol (CHAP)") unless it can verify that the CHAP secret is at least 96 bits, or that IPsec encryption is being used to protect the connection.

Any CHAP secret used for initiator authentication MUST NOT be configured for authentication of any target, and any CHAP secret used for target authentication MUST NOT be configured for

authentication of any initiator. If the CHAP response received by one end of an iSCSI connection is the same as the CHAP response that the receiving endpoint would have generated for the same CHAP challenge, the response MUST be treated as an authentication failure and cause the connection to close (this ensures that the same CHAP secret is not used for authentication in both directions). Also, if an iSCSI implementation can function as both initiator and target, different CHAP secrets and identities MUST be configured for these two roles. The following is an example of the attacks prevented by the above requirements:

Rogue wants to impersonate Storage to Alice, and knows that a single secret is used for both directions of Storage-Alice authentication.

Rogue convinces Alice to open two connections to Rogue, and Rogue identifies itself as Storage on both connections.

Rogue issues a CHAP challenge on connection 1, waits for Alice to respond, and then reflects Alice's challenge as the initial challenge to Alice on connection 2.

If Alice doesn't check for the reflection across connections, Alice's response on connection 2 enables Rogue to impersonate Storage on connection 1, even though Rogue does not know the Alice-Storage CHAP secret.

Originators MUST NOT reuse the CHAP challenge sent by the Responder for the other direction of a bidirectional authentication. Responders MUST check for this condition and close the iSCSI TCP connection if it occurs.

The same CHAP secret SHOULD NOT be configured for authentication of multiple initiators or multiple targets, as this enables any of them to impersonate any other one of them, and compromising one of them enables the attacker to impersonate any of them. It is recommended that iSCSI implementations check for use of identical

CHAP secrets by different peers when this check is feasible, and take appropriate measures to warn users and/or administrators when this is detected.

When an iSCSI initiator or target authenticates itself to counterparts in multiple administrative domains, it SHOULD use a different CHAP secret for each administrative domain to avoid propagating security compromises across domains.

Within a single administrative domain:

- A single CHAP secret MAY be used for authentication of an initiator to multiple targets.
- A single CHAP secret MAY be used for an authentication of a target to multiple initiators when the initiators use an external server (e.g., RADIUS) to verify the target's CHAP responses and do not know the target's CHAP secret.

If an external response verification server (e.g., RADIUS) is not used, employing a single CHAP secret for authentication of a target to multiple initiators requires that all such initiators know that target secret. Any of these initiators can impersonate the target to any other such initiator, and compromise of such an initiator enables an attacker to impersonate the target to all such initiators. Targets SHOULD use separate CHAP secrets for authentication to each initiator when such risks are of concern; in this situation it may be useful to configure a separate logical iSCSI target with its own iSCSI Node Name for each initiator or group of initiators among which such separation is desired.

#### 9.2.2. SRP Considerations

The strength of the SRP authentication method (specified in [RFC2945]) is dependent on the characteristics of the group being used (i.e., the prime modulus  $N$  and generator  $g$ ). As described in [RFC2945],  $N$  is required to be a Sophie-German prime (of the form  $N = 2q + 1$ , where  $q$  is also prime) and the generator  $g$  is a primitive root of  $GF(n)$ . In iSCSI authentication, the prime modulus  $N$  MUST be at least 768 bits.

The list of allowed SRP groups is provided in [RFC3723].

### 9.3. IPsec

iSCSI uses the IPsec mechanism for packet protection (cryptographic integrity, authentication, and confidentiality) at the IP level between the iSCSI communicating end points. The following sections describe the IPsec protocols that must be implemented for data integrity and authentication, confidentiality, and cryptographic key management.

An iSCSI initiator or target may provide the required IPsec support fully integrated or in conjunction with an IPsec front-end device. In the latter case, the compliance requirements with regard to IPsec support apply to the "combined device". Only the "combined device" is to be considered an iSCSI device.

Detailed considerations and recommendations for using IPsec for iSCSI are provided in [RFC3723].

#### 9.3.1. Data Integrity and Authentication

Data authentication and integrity is provided by a cryptographic keyed Message Authentication Code in every sent packet. This code protects against message insertion, deletion, and modification. Protection against message replay is realized by using a sequence counter.

An iSCSI compliant initiator or target **MUST** provide data integrity and authentication by implementing IPsec [RFC4301] with ESP [RFC4303] in tunnel mode and **MAY** provide data integrity and authentication by implementing IPsec with ESP in transport mode. The IPsec implementation **MUST** fulfill the following iSCSI specific requirements:

- HMAC-SHA1 **MUST** be implemented [RFC2404].
- AES CBC MAC with XCBC extensions **SHOULD** be implemented [RFC3566].

The ESP anti-replay service **MUST** also be implemented.

At the high speeds iSCSI is expected to operate, a single IPsec SA could rapidly cycle through the 32-bit IPsec sequence number space.

In view of this, an iSCSI implementation that operates at speeds of 1 Gbps or greater MUST implement the IPsec sequence number extension [RFC4303] and SHOULD use it on iSCSI connections.

#### 9.3.2. Confidentiality

Confidentiality is provided by encrypting the data in every packet. When confidentiality is used it MUST be accompanied by data integrity and authentication to provide comprehensive protection against eavesdropping, message insertion, deletion, modification, and replaying.

An iSCSI compliant initiator or target MUST provide confidentiality by implementing IPsec [RFC4301] with ESP [RFC4303] in tunnel mode and MAY provide confidentiality by implementing IPsec with ESP in transport mode, with the following iSCSI specific requirements:

- 3DES in CBC mode MUST be implemented [RFC2451].
- AES in Counter mode SHOULD be implemented [RFC3686].

DES in CBC mode SHOULD NOT be used due to its inherent weakness. The NULL encryption algorithm MUST also be implemented.

#### 9.3.3. Policy, Security Associations, and Cryptographic Key Management

A compliant iSCSI implementation MUST meet the cryptographic key management requirements of the IPsec protocol suite. Authentication, security association negotiation, and cryptographic key management MUST be provided by implementing IKE [RFC5996] using the IPsec DOI [RFC5996] with the following iSCSI specific requirements:

- Peer authentication using a pre-shared cryptographic key MUST be supported. Certificate-based peer authentication using digital signatures MAY be supported. Peer

authentication using the public key encryption methods outlined in IKE sections 5.2 and 5.3[7] SHOULD NOT be used.

- When digital signatures are used to achieve authentication, an IKE negotiator SHOULD use IKE Certificate Request Payload(s) to specify the certificate authority. IKE negotiators SHOULD check the pertinent Certificate Revocation List (CRL) before accepting a PKI certificate for use in IKE authentication procedures.
- Conformant iSCSI implementations MUST support IKE Main Mode and SHOULD support Aggressive Mode. IKE main mode with pre-shared key authentication method SHOULD NOT be used when either the initiator or the target uses dynamically assigned IP addresses. While in many cases pre-shared keys offer good security, situations in which dynamically assigned addresses are used force the use of a group pre-shared key, which creates vulnerability to a man-in-the-middle attack.
- In the IKE Phase 2 Quick Mode, exchanges for creating the Phase 2 SA, the Identity Payload, fields MUST be present. ID\_IPV4\_ADDR, ID\_IPV6\_ADDR (if the protocol stack supports IPv6) and ID\_FQDN Identity payloads MUST be supported; ID\_USER\_FQDN SHOULD be supported. The IP Subnet, IP Address Range, ID\_DER\_ASN1\_DN, and ID\_DER\_ASN1\_GN formats SHOULD NOT be used. The ID\_KEY\_ID Identity Payload MUST NOT be used.

Manual cryptographic keying MUST NOT be used because it does not provide the necessary re-keying support.

When IPsec is used, the receipt of an SHOULD NOT be interpreted as a reason TCP connection. If additional traffic Phase 2 SA will be created to protect

IKE Phase 2 delete message for tearing down the iSCSI is sent on it, a new IKE it.



The method used by the initiator to determine whether the target should be connected using IPsec is regarded as an issue of IPsec policy administration, and thus not defined in the iSCSI standard.

If an iSCSI target is discovered via a SendTargets request in a discovery session not using IPsec, the initiator should assume that it does not need IPsec to establish a session to that target. If an iSCSI target is discovered using a discovery session that does use IPsec, the initiator SHOULD use IPsec when establishing a session to that target.

#### 9.4. Security Considerations for the X#NodeArchitecture Key

The security considerations in this section are specific to the X#NodeArchitecture discussed in Section 12.24 - "X#NodeArchitecture".

This extension key transmits specific implementation details about the node that sends it; such details may be considered sensitive in some environments. For example, if a certain software or firmware version is known to contain security weaknesses, announcing the presence of that version via this key may not be desirable. The countermeasures for this security concern are:

- sending less detailed information in the key values,
- not sending the extension key, or
- using IPsec ([RFC4303]) to provide confidentiality for the iSCSI connection on which the key is sent

To support the first and second countermeasures, all implementations of this extension key MUST provide an administrative mechanism to disable sending the key. In addition, all implementations SHOULD provide an administrative mechanism to configure a verbosity level of the key value, thereby controlling the amount of information sent.

For example, a lower verbosity might enable transmission of node architecture component names only, but no version numbers. The choice of which countermeasure is most appropriate depends on the environment. However, sending less detailed information in the key

values may be an acceptable countermeasure in many environments, since it provides a compromise between sending too much information and the other more complete countermeasures of not sending the key at all or using IPsec.

In addition to security considerations involving transmission of the key contents, any logging method(s) used for the key values MUST keep the information secure from intruders. For all implementations, the requirements to address this security concern are:

- Display of the log MUST only be possible with administrative rights to the node.
- Options to disable logging to disk and to keep logs for a fixed duration SHOULD be provided.

Finally, it is important to note that different nodes may have different levels of risk, and these differences may affect the implementation. The components of risk include assets, threats, and vulnerabilities. Consider the following example iSCSI nodes, which demonstrate differences in assets and vulnerabilities of the nodes, and as a result, differences in implementation:

One iSCSI target based on a special-purpose operating system:  
Since the iSCSI target controls access to the data storage containing company assets, the asset level is seen as very high. Also, because of the special-purpose operating system, in which vulnerabilities are less well-known, the vulnerability level is viewed as low.

Multiple iSCSI initiators in a blade farm, each running a general purpose operating system: The asset level of each node is viewed as low, since blades are replaceable and low cost. However, the vulnerability level is viewed as high, since there may be many wellknown vulnerabilities to that general-purpose operating system. For this target, an appropriate implementation might be logging of received key values, but no transmission of the key. For this initiator,

iSCSI (Consolidated) 3/11/11  
an appropriate implementation might be transmission of the  
key, but no logging of received key values.

## 10. Notes to Implementers

This section notes some of the performance and reliability considerations of the iSCSI protocol. This protocol was designed to allow efficient silicon and software implementations. The iSCSI task tag mechanism was designed to enable Direct Data Placement (DDP - a DMA form) at the iSCSI level or lower.

The guiding assumption made throughout the design of this protocol is that targets are resource constrained relative to initiators.

Implementers are also advised to consider the implementation consequences of the iSCSI to SCSI mapping model as outlined in Section 3.4.3 - "Consequences of the Model".

## 10.1. Multiple Network Adapters

The iSCSI protocol allows multiple connections, not all of which need to go over the same network adapter. If multiple network connections are to be utilized with hardware support, the iSCSI protocol command-data-status allegiance to one TCP connection ensures that there is no need to replicate information across network adapters or otherwise require them to cooperate.

However, some task management commands may require some loose form of cooperation or replication at least on the target.

## 10.1.1. Conservative Reuse of ISIDs

Historically, the SCSI model (and implementations and applications based on that model) has assumed that SCSI ports are static, physical entities. Recent extensions to the SCSI model have taken advantage of persistent worldwide unique names for these ports. In iSCSI however, the SCSI initiator ports are the endpoints of dynamically created sessions, so the presumptions of "static and physical" do not apply. In any case, the model clauses (particularly, Section 3.4.2 - "SCSI Architecture Model") provide for persistent, reusable names for the iSCSI-type SCSI initiator ports even though there does not need to be any physical entity bound to these names.

To both minimize the disruption of legacy applications and to better facilitate the SCSI features that rely on persistent names for SCSI ports, iSCSI implementations SHOULD attempt to provide a stable presentation of SCSI Initiator Ports (both to the upper OS-layers and to the targets to which they connect). This can be achieved in an initiator implementation by conservatively reusing ISIDs. In other words, the same ISID should be used in the Login process to multiple target portal groups (of the same iSCSI Target or different iSCSI Targets). The ISID RULE (Section 3.4.3 - "Consequences of the Model") only prohibits reuse to the same target portal group. It does not "preclude" reuse to other target portal groups.

The principle of conservative reuse "encourages" reuse to other target portal groups. When a SCSI target device sees the same (InitiatorName, ISID) pair in different sessions to different target portal groups, it can identify the underlying SCSI Initiator Port on each session as the same SCSI port. In effect, it can recognize multiple paths from the same source.

#### 10.1.2. iSCSI Name, ISID, and TPGT Use

The designers of the iSCSI protocol are aware that legacy SCSI transports rely on initiator identity to assign access to storage resources. Although newer techniques are available and simplify access control, support for configuration and authentication schemes that are based on initiator identity is deemed important in order to support legacy systems and administration software. iSCSI thus supports the notion that it should be possible to assign access to storage resources based on "initiator device" identity.

When there are multiple hardware or software components coordinated as a single iSCSI Node, there must be some (logical) entity that represents the iSCSI Node that makes the iSCSI Node Name available to all components involved in session creation and login. Similarly, this entity that represents the iSCSI Node must be able to coordinate session identifier resources (ISID for initiators) to enforce both the ISID and TSIH RULES (see Section 3.4.3 - "Consequences of the Model").

For targets, because of the closed environment, implementation of this entity should be straightforward. However, vendors of iSCSI

hardware (e.g., NICs or HBAs) intended for targets, SHOULD provide mechanisms for configuration of the iSCSI Node Name across the portal groups instantiated by multiple instances of these components within a target.

However, complex targets making use of multiple Target Portal Group Tags may reconfigure them to achieve various quality goals. The initiators have two mechanisms at their disposal to discover and/or check reconfiguring targets - the discovery session type and a key returned by the target during login to confirm the TPGT. An initiator should attempt to "rediscover" the target configuration anytime a session is terminated unexpectedly.

For initiators, in the long term, it is expected that operating system vendors will take on the role of this entity and provide standard APIs that can inform components of their iSCSI Node Name and can configure and/or coordinate ISID allocation, use, and reuse.

Recognizing that such initiator APIs are not available today, other implementations of the role of this entity are possible. For example, a human may instantiate the (common) Node name as part of the installation process of each iSCSI component involved in session creation and login. This may be done either by pointing the component to a vendor-specific location for this datum or to a system-wide location. The structure of the ISID namespace (see Section 10.12.5 - "ISID" and [RFC3721]) facilitates implementation of the ISID coordination by allowing each component vendor to independently (of other vendor's components) coordinate allocation, use, and reuse of its own partition of the ISID namespace in a vendor-specific manner. Partitioning of the ISID namespace within initiator portal groups managed by that vendor allows each such initiator portal group to act independently of all other portal groups when selecting an ISID for a login; this facilitates enforcement of the ISID RULE (see Section 3.4.3 - "Consequences of the Model") at the initiator.

A vendor of iSCSI hardware (e.g., NICs or HBAs) intended for use in initiators MUST implement a mechanism for configuring the iSCSI Node Name. Vendors, and administrators must ensure that iSCSI Node Names are unique worldwide. It is therefore important that when

one chooses to reuse the iSCSI Node Name of a disabled unit, not to re-assign that name to the original unit unless its worldwide uniqueness can be ascertained again.

In addition, a vendor of iSCSI hardware must implement a mechanism to configure and/or coordinate ISIDs for all sessions managed by multiple instances of that hardware within a given iSCSI Node. Such configuration might be either permanently pre-assigned at the factory (in a necessarily globally unique way), statically assigned (e.g., partitioned across all the NICs at initialization in a locally unique way), or dynamically assigned (e.g., on-line allocator, also in a locally unique way). In the latter two cases, the configuration may be via public APIs (perhaps driven by an independent vendor's software, such as the OS vendor) or via private APIs driven by the vendor's own software.

The process of name assignment and coordination has to be as encompassing and automated as possible as years of legacy usage have shown it to be highly error-prone. It is to be mentioned that SCSI has today alternative schemes of access control that can be used by all transports and their security is not dependent on strict naming coordination.

#### 10.2. Autosense and Auto Contingent Allegiance (ACA)

Autosense refers to the automatic return of sense data to the initiator in case a command did not complete successfully. iSCSI initiators and targets MUST support and use autosense.

ACA helps preserve ordered command execution in the presence of errors. As iSCSI can have many commands in-flight between initiator and target, iSCSI initiators and targets SHOULD support ACA.

#### 10.3. iSCSI Timeouts

iSCSI recovery actions are often dependent on iSCSI time-outs being recognized and acted upon before SCSI time-outs. Determining the right time-outs to use for various iSCSI actions (command acknowledgements expected, status acknowledgements, etc.) is very much dependent on infrastructure (hardware, links, TCP/IP stack, iSCSI driver). As a guide, the implementer may use an average Nop-

Out/Nop-In turnaround delay multiplied by a "safety factor" (e.g., 4) as a good estimate for the basic delay of the iSCSI stack for a given connection. The safety factor should account for the network load variability. For connection teardown the implementer may want to consider also TCP common practice for the given infrastructure.

Text negotiations MAY also be subject to either time-limits or limits in the number of exchanges. Those SHOULD be generous enough to avoid affecting interoperability (e.g., allowing each key to be negotiated on a separate exchange).

The relation between iSCSI timeouts and SCSI timeouts should also be considered. SCSI timeouts should be longer than iSCSI timeouts plus the time required for iSCSI recovery whenever iSCSI recovery is planned. Alternatively, an implementer may choose to interlock iSCSI timeouts and recovery with SCSI timeouts so that SCSI recovery will become active only where iSCSI is not planned to, or failed to, recover.

The implementer may also want to consider the interaction between various iSCSI exception events - such as a digest failure - and subsequent timeouts. When iSCSI error recovery is active, a digest failure is likely to result in discovering a missing command or data PDU. In these cases, an implementer may want to lower the timeout values to enable faster initiation for recovery procedures.

#### 10.4. Command Retry and Cleaning Old Command Instances

To avoid having old, retried command instances appear in a valid command window after a command sequence number wrap around, the protocol requires (see Section 3.2.2.1 - "Command Numbering and Acknowledging") that on every connection on which a retry has been issued, a non-immediate command be issued and acknowledged within a  $2^{31}-1$  commands interval from the CmdSN of the retried command. This requirement can be fulfilled by an implementation in several ways.

The simplest technique to use is to send a (non-retry) non-immediate SCSI command (or a NOP if no SCSI command is available for a while) after every command retry on the connection on which



the retry was attempted. As errors are deemed rare events, this technique is probably the most effective, as it does not involve additional checks at the initiator when issuing commands.

#### 10.5. Synch and Steering Layer and Performance

While a synch and steering layer is optional, an initiator/target that does not have it working against a target/initiator that demands synch and steering may experience performance degradation caused by packet reordering and loss. Providing a synch and steering mechanism is recommended for all high-speed implementations.

#### 10.6. Considerations for State-dependent Devices and Long-lasting SCSI Operations

Sequential access devices operate on the principle that the position of the device is based on the last command processed. As such, command processing order and knowledge of whether or not the previous command was processed is of the utmost importance to maintain data integrity. For example, inadvertent retries of SCSI commands when it is not known if the previous SCSI command was processed is a potential data integrity risk.

For a sequential access device, consider the scenario in which a SCSI SPACE command to backspace one filemark is issued and then re-issued due to no status received for the command. If the first SPACE command was actually processed, the re-issued SPACE command, if processed, will cause the position to change. Thus, a subsequent write operation will write data to the wrong position and any previous data at that position will be overwritten.

For a medium changer device, consider the scenario in which an EXCHANGE MEDIUM command (the SOURCE ADDRESS and DESTINATION ADDRESS are the same thus performing a swap) is issued and then re-issued due to no status received for the command. If the first EXCHANGE MEDIUM command was actually processed, the re-issued EXCHANGE MEDIUM command, if processed, will perform the swap again. The net effect is no swap was performed thus leaving a data integrity exposure.

All commands that change the state of the device (as in SPACE commands for sequential access devices, and EXCHANGE MEDIUM for medium changer device), MUST be issued as non-immediate commands for deterministic and in order delivery to iSCSI targets.

For many of those state changing commands, the execution model also assumes that the command is executed exactly once. Devices implementing READ POSITION and LOCATE provide a means for SCSI level command recovery and new tape-class devices should support those commands. In their absence a retry at SCSI level is difficult and error recovery at iSCSI level is advisable.

Devices operating on long latency delivery subsystems and performing long lasting SCSI operations may need mechanisms that enable connection replacement while commands are running (e.g., during an extended copy operation).

#### 10.6.1. Determining the Proper ErrorRecoveryLevel

The implementation and use of a specific ErrorRecoveryLevel should be determined based on the deployment scenarios of a given iSCSI implementation. Generally, the following factors must be considered before deciding on the proper level of recovery:

- Application resilience to I/O failures.

- Required level of availability in the face of transport connection failures.

- Probability of transport layer "checksum escape". This in turn decides the iSCSI digest failure frequency, and thus the criticality of iSCSI-level error recovery. The details of estimating this probability are outside the scope of this document.

A consideration of the above factors for SCSI tape devices as an example suggests that implementations SHOULD use ErrorRecoveryLevel=1 when transport connection failure is not a concern and SCSI level recovery is unavailable, and ErrorRecoveryLevel=2 when the connection failure is also of high likelihood during a backup/retrieval.

For extended copy operations, implementations SHOULD use ErrorRecoveryLevel=2 whenever there is a relatively high likelihood of connection failure.

#### 10.7. Multi-task Abort Implementation Considerations

Multi-task abort operations are typically issued in emergencies - such as clearing a device lock-up, HA failover/failback etc. In these circumstances, it is desirable to rapidly go through the error handling process as opposed to the target waiting on multiple third-party initiators who may not even be functional anymore - especially if this emergency is triggered because of one such initiator failure. Therefore, both iSCSI target and initiator implementations SHOULD support FastAbort multi-task abort semantics (Section 4.2.3.4).

Note that both in standard semantics (Section 4.2.3.3) and FastAbort semantics (Section 4.2.3.4), there may be outstanding data transfers even after the TMF completion is reported on the issuing session. In the case of iSCSI/iSER [iSER], these would be tagged data transfers for STags not owned by any active tasks. Whether or not real buffers support these data transfers is implementation-dependent. However, the data transfers logically MUST be silently discarded by the target iSCSI layer in all cases. A target MAY, on an implementation-defined internal timeout, also choose to drop the connections on which it did not receive the expected Data-Out sequences (Section 4.2.3.3) or NOP-Out acknowledgements (Section 4.2.3.4) so as to reclaim the associated buffer, STag, and TTT resources as appropriate.

## 11. iSCSI PDU Formats

All multi-byte integers that are specified in formats defined in this document are to be represented in network byte order (i.e., big endian). Any field that appears in this document assumes that the most significant byte is the lowest numbered byte and the most significant bit (within byte or field) is the lowest numbered bit unless specified otherwise.

Any compliant sender MUST set all bits not defined and all reserved fields to zero unless specified otherwise. Any compliant receiver MUST ignore any bit not defined and all reserved fields unless specified otherwise. Receipt of reserved code values in defined fields MUST be reported as a protocol error.

Reserved fields are marked by the word "reserved", some abbreviation of "reserved", or by "." for individual bits when no other form of marking is technically feasible.

### 11.1. iSCSI PDU Length and Padding

iSCSI PDUs are padded to the closest integer number of four byte words. The padding bytes SHOULD be sent as 0.

### 11.2. PDU Template, Header, and Opcodes

All iSCSI PDUs have one or more header segments and, optionally, a data segment. After the entire header segment group a header-digest MAY follow. The data segment MAY also be followed by a data-digest.

The Basic Header Segment (BHS) is the first segment in all of the iSCSI PDUs. The BHS is a fixed-length 48-byte header segment. It MAY be followed by Additional Header Segments (AHS), a Header-Digest, a Data Segment, and/or a Data-Digest.

The overall structure of an iSCSI PDU is as follows:

iSCSI (Consolidated)																															
0								1								2								3							
Byte/																															
/																															
0 1 2 3 4 5 6 7								0 1 2 3 4 5 6 7								0 1 2 3 4 5 6 7								0 1 2 3 4 5 6 7							
+								+								+								+							
0/ Basic Header Segment (BHS)																															
+ /																															
+								+								+								+							
48/ Additional Header Segment 1 (AHS) (optional)																															
+ /																															
+								+								+								+							
/ Additional Header Segment 2 (AHS) (optional)																															
+ /																															
+								+								+								+							
----																															
+								+								+								+							
/ Additional Header Segment n (AHS) (optional)																															
+ /																															
+								+								+								+							
----																															
+								+								+								+							
k/ Header-Digest (optional)																															
+ /																															
+								+								+								+							
l/ Data Segment (optional)																															
+ /																															
+								+								+								+							
m/ Data-Digest (optional)																															
+ /																															
+								+								+								+							

All PDU segments and digests are padded to the closest integer number of four byte words. For example, all PDU segments and digests start at a four byte word boundary and the padding ranges from 0 to 3 bytes. The padding bytes SHOULD be sent as 0.

iSCSI response PDUs do not have AH Segments.

### 11.2.1. Basic Header Segment (BHS)

The BHS is 48 bytes long. The Opcode and DataSegmentLength fields appear in all iSCSI PDUs. In addition, when used, the Initiator

Task Tag and Logical Unit Number always appear in the same location in the header.

The format of the BHS is:

Byte/	0								1								2								3							
/	0	1	2	3	4	5	6	7	0	1	2	3	4	5	6	7	0	1	2	3	4	5	6	7	0	1	2	3	4	5	6	7
0	I   Opcode								F   Opcode-specific fields																							
4	TotalAHSLength								DataSegmentLength																							
8	LUN or Opcode-specific fields																															
12																																
16	Initiator Task Tag																															
20	Opcode-specific fields																															
48																																

#### 11.2.1.1. I

For request PDUs, the I bit set to 1 is an immediate delivery marker.

#### 11.2.1.2. Opcode

The Opcode indicates the type of iSCSI PDU the header encapsulates.

The Opcodes are divided into two categories: initiator opcodes and target opcodes. Initiator opcodes are in PDUs sent by the initiator (request PDUs). Target opcodes are in PDUs sent by the target (response PDUs).

Initiators MUST NOT use target opcodes and targets MUST NOT use initiator opcodes.

Initiator opcodes defined in this specification are:

- 0x00 NOP-Out
- 0x01 SCSI Command (encapsulates a SCSI Command Descriptor Block)
- 0x02 SCSI Task Management function request
- 0x03 Login Request
- 0x04 Text Request
- 0x05 SCSI Data-out (for WRITE operations)
- 0x06 Logout Request
- 0x10 SNACK Request
- 0x1c-0x1e Vendor specific codes

Target opcodes are:

- 0x20 NOP-In
- 0x21 SCSI Response - contains SCSI status and possibly sense information or other response information.
- 0x22 SCSI Task Management function response
- 0x23 Login Response
- 0x24 Text Response
- 0x25 SCSI Data-in - for READ operations.
- 0x26 Logout Response

0x31 Ready To Transfer (R2T) - sent by target when it is ready to receive data.

0x32 Asynchronous Message - sent by target to indicate certain special conditions.

0x3c-0x3e Vendor specific codes

0x3f Reject

All other opcodes are reserved.

#### 11.2.1.3. Final (F) bit

When set to 1 it indicates the final (or only) PDU of a sequence.

#### 11.2.1.4. Opcode-specific Fields

These fields have different meanings for different opcode types.

#### 11.2.1.5. TotalAHSLength

Total length of all AHS header segments in units of four byte words including padding, if any.

The TotalAHSLength is only used in PDUs that have an AHS and MUST be 0 in all other PDUs.

#### 11.2.1.6. DataSegmentLength

This is the data segment payload length in bytes (excluding padding). The DataSegmentLength MUST be 0 whenever the PDU has no data segment.

#### 11.2.1.7. LUN

Some opcodes operate on a specific Logical Unit. The Logical Unit Number (LUN) field identifies which Logical Unit. If the opcode does not relate to a Logical Unit, this field is either ignored or may be used in an opcode specific way. The LUN field is 64-bits and should be formatted in accordance with [SAM2]. For example,



LUN[0] from [SAM2] is BHS byte 8 and so on up to LUN[7] from [SAM2], which is BHS byte 15.

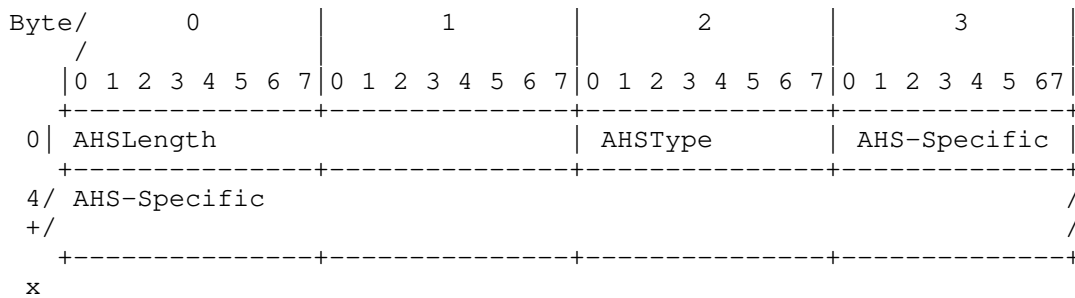
#### 11.2.1.8. Initiator Task Tag

The initiator assigns a Task Tag to each iSCSI task it issues. While a task exists, this tag MUST uniquely identify the task session-wide. SCSI may also use the initiator task tag as part of the SCSI task identifier when the timespan during which an iSCSI initiator task tag must be unique extends over the timespan during which a SCSI task tag must be unique. However, the iSCSI Initiator Task Tag must exist and be unique even for untagged SCSI commands.

An ITT value of 0xffffffff is reserved and MUST NOT be assigned for a task by the initiator. The only instance in which it may be seen on the wire is in a target-initiated NOP-In PDU (Section 11.19) and in the initiator response to that PDU, if necessary.

#### 11.2.2. Additional Header Segment (AHS)

The general format of an AHS is:



##### 11.2.2.1. AHSType

The AHSType field is coded as follows:

bit 0-1 - Reserved

bit 2-7 - AHS code

0 - Reserved

- 1 - Extended CDB
- 2 - Expected Bidirectional Read Data Length
- 3 - 63 Reserved

#### 11.2.2.2. AHSLength

This field contains the effective length in bytes of the AHS excluding AHSType and AHSLength and padding, if any. The AHS is padded to the smallest integer number of 4 byte words (i.e., from 0 up to 3 padding bytes).

#### 11.2.2.3. Extended CDB AHS

The format of the Extended CDB AHS is:

Byte/ /	0								1								2								3								
	0	1	2	3	4	5	6	7	0	1	2	3	4	5	6	7	0	1	2	3	4	5	6	7	0	1	2	3	4	5	6	7	
0	AHSLength (CDBLength-15)								0x01								Reserved																
4/	ExtendedCDB...+padding																																/
+/																																	/
x																																	

This type of AHS MUST NOT be used if the CDBLength is less than 17.  
The length includes the reserved byte 3.

#### 11.2.2.4. Bidirectional Expected Read-Data Length AHS

The format of the Bidirectional Read Expected Data Transfer Length AHS is:

iSCSI (Consolidated)																												3/11/11																													
0								1								2								3																																	
Byte/ /	0	1	2	3	4	5	6	7	0	1	2	3	4	5	6	7	0	1	2	3	4	5	6	7	0	1	2	3	4	5	6	7																									
0	AHSLength (0x0005)								0x02								Reserved																																								
4	Expected Read-Data Length																																																								
8																																																									

### 11.2.3. Header Digest and Data Digest

Optional header and data digests protect the integrity of the header and data, respectively. The digests, if present, are located, respectively, after the header and PDU-specific data, and cover respectively the header and the PDU data, each including the padding bytes, if any.

The existence and type of digests are negotiated during the Login Phase.

The separation of the header and data digests is useful in iSCSI routing applications, in which only the header changes when a message is forwarded. In this case, only the header digest should be recalculated.

Digests are not included in data or header length fields.

A zero-length Data Segment also implies a zero-length data-digest.

### 11.2.4. Data Segment

The (optional) Data Segment contains PDU associated data. Its payload effective length is provided in the BHS field - DataSegmentLength. The Data Segment is also padded to an integer number of 4 byte words.

### 11.3. SCSI Command

The format of the SCSI Command PDU is:

iSCSI (Consolidated)																3/11/11															
0								1								2								3							
Byte/																															
/																															
0 1 2 3 4 5 6 7								0 1 2 3 4 5 6 7								0 1 2 3 4 5 6 7								0 1 2 3 4 5 6 7							
+								+								+								+							
0 .  I  0x01								F R W . . ATTR								Reserved															
+								+								+								+							
4  TotalAHSLength								DataSegmentLength																							
+								+								+								+							
8  Logical Unit Number (LUN)																															
+																															
12																															
+								+								+								+							
16  Initiator Task Tag																															
+								+								+								+							
20  Expected Data Transfer Length																															
+								+								+								+							
24  CmdSN																															
+								+								+								+							
28  ExpStatSN																															
+								+								+								+							
32/ SCSI Command Descriptor Block (CDB)																															
+/																															
+								+								+								+							
48/ AHS (Optional)																															
+								+								+								+							
x/ Header Digest (Optional)																															
+								+								+								+							
y/ (DataSegment, Command Data) (Optional)																															
+/																															
+								+								+								+							
z/ Data Digest (Optional)																															
+								+								+								+							

### 11.3.1. Flags and Task Attributes (byte 1)

The flags for a SCSI Command are:

bit 0 (F) is set to 1 when no unsolicited SCSI Data-Out PDUs follow this PDU. When F=1 for a write and if Expected Data Transfer Length is larger than the DataSegmentLength, the target may solicit additional data through R2T.

bit 1 (R) is set to 1 when the command is expected to input data.

bit 2 (W) is set to 1 when the command is expected to output data.

bit 3-4 Reserved.

bit 5-7 contains Task Attributes.

Task Attributes (ATTR) have one of the following integer values (see [SAM2] for details):

0 - Untagged

1 - Simple

2 - Ordered

3 - Head of Queue

4 - ACA

5-7 - Reserved

Setting both the W and the F bit to 0 is an error.

Either or both of R and W MAY be 1 when either the Expected Data Transfer Length and/or Bidirectional Read Expected Data Transfer Length are 0, but they MUST NOT both be 0 when the Expected Data Transfer Length and/or Bidirectional Read Expected Data Transfer Length are not 0 (i.e., when some data transfer is expected the transfer direction is indicated by the R and/or W bit).

#### 11.3.2. CmdSN - Command Sequence Number

Enables ordered delivery across multiple connections in a single session.

## 11.3.3. ExpStatSN

Command responses up to ExpStatSN-1 (mod  $2^{32}$ ) have been received (acknowledges status) on the connection.

## 11.3.4. Expected Data Transfer Length

For unidirectional operations, the Expected Data Transfer Length field contains the number of bytes of data involved in this SCSI operation. For a unidirectional write operation (W flag set to 1 and R flag set to 0), the initiator uses this field to specify the number of bytes of data it expects to transfer for this operation. For a unidirectional read operation (W flag set to 0 and R flag set to 1), the initiator uses this field to specify the number of bytes of data it expects the target to transfer to the initiator. It corresponds to the SAM2 byte count.

For bidirectional operations (both R and W flags are set to 1), this field contains the number of data bytes involved in the write transfer. For bidirectional operations, an additional header segment MUST be present in the header sequence that indicates the Bidirectional Read Expected Data Transfer Length. The Expected Data Transfer Length field and the Bidirectional Read Expected Data Transfer Length field correspond to the SAM2 byte count.

If the Expected Data Transfer Length for a write and the length of the immediate data part that follows the command (if any) are the same, then no more data PDUs are expected to follow. In this case, the F bit MUST be set to 1.

If the Expected Data Transfer Length is higher than the FirstBurstLength (the negotiated maximum amount of unsolicited data the target will accept), the initiator MUST send the maximum amount of unsolicited data OR ONLY the immediate data, if any.

Upon completion of a data transfer, the target informs the initiator (through residual counts) of how many bytes were actually processed (sent and/or received) by the target.

#### 11.3.5. CDB - SCSI Command Descriptor Block

There are 16 bytes in the CDB field to accommodate the commonly used CDBs. Whenever the CDB is larger than 16 bytes, an Extended CDB AHS MUST be used to contain the CDB spillover.

#### 11.3.6. Data Segment - Command Data

Some SCSI commands require additional parameter data to accompany the SCSI command. This data may be placed beyond the boundary of the iSCSI header in a data segment. Alternatively, user data (e.g., from a WRITE operation) can be placed in the data segment (both cases are referred to as immediate data). These data are governed by the rules for solicited vs. unsolicited data outlined in Section 3.2.4.2 - "Data Transfer Overview".

#### 11.4. SCSI Response

The format of the SCSI Response PDU is:

iSCSI (Consolidated)																																3/11/11							
0								1								2								3															
Byte/ /																																							
0 1 2 3 4 5 6 7								0 1 2 3 4 5 6 7								0 1 2 3 4 5 6 7								0 1 2 3 4 5 6 7															
0 . .0x21								1 . .o u O U .Response								Status																							
4 TotalAHSLength								DataSegmentLength																															
8 Reserved																																							
12																																							
16 Initiator Task Tag																																							
20 SNACK Tag or Reserved																																							
24 StatSN																																							
28 ExpCmdSN																																							
32 MaxCmdSN																																							
36 ExpDataSN or Reserved																																							
40 Bidirectional Read Residual Count or Reserved																																							
44 Residual Count or Reserved																																							
48 Header-Digest (Optional)																																							
/ Data Segment (Optional)																																							
+/																																							
Data-Digest (Optional)																																							

#### 11.4.1. Flags (byte 1)

bit 1-2 Reserved.

bit 3 - (o) set for Bidirectional Read Residual Overflow. In this case, the Bidirectional Read Residual Count indicates



the number of bytes that were not transferred to the initiator because the initiator's Expected Bidirectional Read Data Transfer Length was not sufficient.

bit 4 - (u) set for Bidirectional Read Residual Underflow. In this case, the Bidirectional Read Residual Count indicates the number of bytes that were not transferred to the initiator out of the number of bytes expected to be transferred.

bit 5 - (O) set for Residual Overflow. In this case, the Residual Count indicates the number of bytes that were not transferred because the initiator's Expected Data Transfer Length was not sufficient. For a bidirectional operation, the Residual Count contains the residual for the write operation.

bit 6 - (U) set for Residual Underflow. In this case, the Residual Count indicates the number of bytes that were not transferred out of the number of bytes that were expected to be transferred. For a bidirectional operation, the Residual Count contains the residual for the write operation.

bit 7 - (0) Reserved.

Bits O and U and bits o and u are mutually exclusive (i.e., having both o and u or O and U set to 1 is a protocol error). For a response other than "Command Completed at Target", bits 3-6 MUST be 0.

#### 11.4.2. Status

The Status field is used to report the SCSI status of the command (as specified in [SAM2]) and is only valid if the Response Code is Command Completed at target.

Some of the status codes defined in [SAM2] are:

0x00 GOOD

0x02 CHECK CONDITION

0x08 BUSY

0x18 RESERVATION CONFLICT

0x28 TASK SET FULL

0x30 ACA ACTIVE

0x40 TASK ABORTED

See [SAM2] for the complete list and definitions.

If a SCSI device error is detected while data from the initiator is still expected (the command PDU did not contain all the data and the target has not received a Data PDU with the final bit Set), the target MUST wait until it receives a Data PDU with the F bit set in the last expected sequence before sending the Response PDU.

#### 11.4.3. Response

This field contains the iSCSI service response.

iSCSI service response codes defined in this specification are:

0x00 - Command Completed at Target

0x01 - Target Failure

0x80-0xff - Vendor specific

All other response codes are reserved.

The Response is used to report a Service Response. The mapping of the response code into a SCSI service response code value, if needed, is outside the scope of this document. However, in symbolic terms response value 0x00 maps to the SCSI service response (see [SAM2] and [SPC3]) of TASK COMPLETE or LINKED

COMMAND COMPLETE. All other Response values map to the SCSI service response of SERVICE DELIVERY OR TARGET FAILURE.

If a SCSI Response PDU does not arrive before the session is terminated, the SCSI service response is SERVICE DELIVERY OR TARGET FAILURE.

A non-zero response field indicates a failure to execute the command in which case the Status and Flag fields are undefined.

#### 11.4.4. SNACK Tag

This field contains a copy of the SNACK Tag of the last SNACK Tag accepted by the target on the same connection and for the command for which the response is issued. Otherwise it is reserved and should be set to 0.

After issuing a R-Data SNACK the initiator must discard any SCSI status unless contained in an SCSI Response PDU carrying the same SNACK Tag as the last issued R-Data SNACK for the SCSI command on the current connection.

For a detailed discussion on R-Data SNACK see SNACK.

#### 11.4.5. Residual Count

##### 11.4.5.1. Field Semantics

The Residual Count field MUST be valid in the case where either the U bit or the O bit is set. If neither bit is set, the Residual Count field is reserved. Targets may set the residual count and initiators may use it when the response code is "completed at target" (even if the status returned is not GOOD). If the O bit is set, the Residual Count indicates the number of bytes that were not transferred because the initiator's Expected Data Transfer Length was not sufficient. If the U bit is set, the Residual Count indicates the number of bytes that were not transferred out of the number of bytes expected to be transferred.

## 11.4.5.2. Residuals Concepts Overview

SCSI-Presented Data Transfer Length (SPDTL) is the term this document uses (see Section 1.1 for definition) to represent the aggregate data length that the target SCSI layer attempts to transfer using the local iSCSI layer for a task. Expected Data Transfer Length (EDTL) is the iSCSI term that represents the length of data that the iSCSI layer expects to transfer for a task. EDTL is specified in the SCSI Command PDU.

When  $SPDTL = EDTL$  for a task, the target iSCSI layer completes the task with no residuals. Whenever SPDTL differs from EDTL for a task, that task is said to have a residual.

If  $SPDTL > EDTL$  for a task, iSCSI Overflow MUST be signaled in the SCSI Response PDU as specified in Section 11.4.5.1. The Residual Count MUST be set to the numerical value of  $(SPDTL - EDTL)$ .

If  $SPDTL < EDTL$  for a task, iSCSI Underflow MUST be signaled in the SCSI Response PDU as specified in Section 11.4.5.1. The Residual Count MUST be set to the numerical value of  $(EDTL - SPDTL)$ .

Note that the Overflow and Underflow scenarios are independent of Data-In and Data-Out. Either scenario is logically possible in either direction of data transfer.

## 11.4.5.3. SCSI REPORT LUNS and Residual Overflow

This section discusses the residual overflow issues citing the example of the SCSI REPORT LUNS command. Note however that there are several SCSI commands (e.g., INQUIRY) with ALLOCATION LENGTH fields following the same underlying rules. The semantics in the rest of the section apply to all such SCSI commands.

The specification of the SCSI REPORT LUNS command requires that the SCSI target limit the amount of data transferred to a maximum size (ALLOCATION LENGTH) provided by the initiator in the REPORT LUNS CDB.

If the Expected Data Transfer Length (EDTL) in the iSCSI header of the SCSI Command PDU for a REPORT LUNS command is set to at least

as large as that ALLOCATION LENGTH, the SCSI layer truncation prevents an iSCSI Residual Overflow from occurring. A SCSI initiator can detect that such truncation has occurred via other information at the SCSI layer. The rest of the section elaborates this required behavior.

The SCSI REPORT LUNS command requests a target SCSI layer to return a logical unit inventory (LUN list) to the initiator SCSI layer (see Section 6.21 of [SPC3]). The size of this LUN list may not be known to the initiator SCSI layer when it issues the REPORT LUNS command; to avoid transferring more LUN list data than the initiator is prepared for, the REPORT LUNS CDB contains an ALLOCATION LENGTH field to specify the maximum amount of data to be transferred to the initiator for this command. If the initiator SCSI layer has underestimated the number of logical units at the target, it is possible that the complete logical unit inventory does not fit in the specified ALLOCATION LENGTH. In this situation, Section 4.3.3.6 in [SPC3] requires that the target SCSI layer "shall terminate transfers to the Data-In Buffer" when the number of bytes specified by the ALLOCATION LENGTH field have been transferred.

Therefore, in response to a REPORT LUNS command, the SCSI layer at the target presents at most ALLOCATION LENGTH bytes of data (logical unit inventory) to iSCSI for transfer to the initiator. For a REPORT LUNS command, if the iSCSI EDTL is at least as large as the ALLOCATION LENGTH, the SCSI truncation ensures that the EDTL will accommodate all of the data to be transferred. If all of the logical unit inventory data presented to the iSCSI layer -- i.e., the data remaining after any SCSI truncation -- is transferred to the initiator by the iSCSI layer, an iSCSI Residual Overflow has not occurred and the iSCSI (O) bit MUST NOT be set in the SCSI Response or final SCSI Data-Out PDU. Note that this behavior is implied by the combination of Section 11.4.5.1 along with the specification of the REPORT LUNS command in [SPC3]. However, if the iSCSI EDTL is larger than the ALLOCATION LENGTH in this scenario, note that the iSCSI Underflow MUST be signaled in the SCSI Response PDU. An iSCSI Underflow MUST also be signaled when the iSCSI EDTL is equal to the ALLOCATION LENGTH but the logical unit inventory data presented to the iSCSI layer is smaller than the ALLOCATION LENGTH.

The LUN LIST LENGTH field in the logical unit inventory (the first field in the inventory) is not affected by truncation of the inventory to fit in ALLOCATION LENGTH; this enables a SCSI initiator to determine that the received inventory is incomplete by noticing that the LUN LIST LENGTH in the inventory is larger than the ALLOCATION LENGTH that was sent in the REPORT LUNS CDB. A common initiator behavior in this situation is to re-issue the REPORT LUNS command with a larger ALLOCATION LENGTH.

#### 11.4.6. Bidirectional Read Residual Count

The Bidirectional Read Residual Count field MUST be valid in the case where either the u bit or the o bit is set. If neither bit is set, the Bidirectional Read Residual Count field is reserved. Targets may set the Bidirectional Read Residual Count and initiators may use it when the response code is "completed at target". If the o bit is set, the Bidirectional Read Residual Count indicates the number of bytes that were not transferred to the initiator because the initiator's Expected Bidirectional Read Transfer Length was not sufficient. If the u bit is set, the Bidirectional Read Residual Count indicates the number of bytes that were not transferred to the initiator out of the number of bytes expected to be transferred.

#### 11.4.7. Data Segment - Sense and Response Data Segment

iSCSI targets MUST support and enable autosense. If Status is CHECK CONDITION (0x02), then the Data Segment MUST contain sense data for the failed command.

For some iSCSI responses, the response data segment MAY contain some response related information, (e.g., for a target failure, it may contain a vendor specific detailed description of the failure).

If the DataSegmentLength is not 0, the format of the Data Segment is as follows:

		iSCSI (Consolidated)																												3/11/11	
Byte/ /	0	1							2							3															
		0	1	2	3	4	5	6	7	0	1	2	3	4	5	6	7	0	1	2	3	4	5	6	7	0	1	2	3	4	5
0		SenseLength														Sense Data															
x/		Sense Data																													
y/		Response Data																													
/																															
z/																															

#### 11.4.7.1. SenseLength

Length of Sense Data.

#### 11.4.7.2. Sense Data

The Sense Data contains detailed information about a check condition and [SPC3] specifies the format and content of the Sense Data.

Certain iSCSI conditions result in the command being terminated at the target (response Command Completed at Target) with a SCSI Check Condition Status as outlined in the next table:

iSCSI Condition	Sense Key	Additional Sense Code & Qualifier
Unexpected unsolicited data	Aborted Command-0B	ASC = 0x0c ASCQ = 0x0c Write Error
Incorrect amount of data	Aborted Command-0B	ASC = 0x0c ASCQ = 0x0d Write Error
Protocol Service CRC error	Aborted Command-0B	ASC = 0x47 ASCQ = 0x05 CRC Error Detected
SNACK rejected	Aborted Command-0B	ASC = 0x11 ASCQ = 0x13 Read Error

The target reports the "Incorrect amount of data" condition if during data output the total data length to output is greater than FirstBurstLength and the initiator sent unsolicited non-immediate data but the total amount of unsolicited data is different than FirstBurstLength. The target reports the same error when the amount of data sent as a reply to an R2T does not match the amount requested.

#### 11.4.8. ExpDataSN

The number of Data-In (read) PDUs the target has sent for the command.

This field MUST be 0 if the response code is not Command Completed at Target or the target sent no Data-In PDUs for the command.

#### 11.4.9. StatSN - Status Sequence Number

StatSN is a Sequence Number that the target iSCSI layer generates per connection and that in turn, enables the initiator to acknowledge status reception. StatSN is incremented by 1 for every response/status sent on a connection except for responses sent as a result of a retry or SNACK. In the case of responses sent due to a retransmission request, the StatSN MUST be the same as the first time the PDU was sent unless the connection has since been restarted.



## 11.4.10. ExpCmdSN - Next Expected CmdSN from this Initiator

ExpCmdSN is a Sequence Number that the target iSCSI returns to the initiator to acknowledge command reception. It is used to update a local variable with the same name. An ExpCmdSN equal to MaxCmdSN+1 indicates that the target cannot accept new commands.

## 11.4.11. MaxCmdSN - Maximum CmdSN from this Initiator

MaxCmdSN is a Sequence Number that the target iSCSI returns to the initiator to indicate the maximum CmdSN the initiator can send. It is used to update a local variable with the same name. If MaxCmdSN is equal to ExpCmdSN-1, this indicates to the initiator that the target cannot receive any additional commands. When MaxCmdSN changes at the target while the target has no pending PDUs to convey this information to the initiator, it MUST generate a NOP-IN to carry the new MaxCmdSN.

## 11.5. Task Management Function Request

Byte/ /	0	1	2	3
	0 1 2 3 4 5 6 7	0 1 2 3 4 5 6 7	0 1 2 3 4 5 6 7	0 1 2 3 4 5 6 7
0	. I  0x02	1  Function	Reserved	
4	TotalAHSLength	DataSegmentLength		
8	Logical Unit Number (LUN) or Reserved			
12				
16	Initiator Task Tag			
20	Referenced Task Tag or 0xffffffff			
24	CmdSN			
28	ExpStatSN			
32	RefCmdSN or Reserved			
36	ExpDataSN or Reserved			
40	Reserved			/
+				/
48	Header-Digest (Optional)			

## 11.5.1. Function

The Task Management functions provide an initiator with a way to explicitly control the execution of one or more Tasks (SCSI and iSCSI tasks). The Task Management function codes are listed below. For a more detailed description of SCSI task management, see [SAM2].

- 1 - ABORT TASK - aborts the task identified by the Referenced Task Tag field.

- 2     - ABORT TASK SET - aborts all Tasks issued via this session on the logical unit.
- 3     - CLEAR ACA - clears the Auto Contingent Allegiance condition.
- 4     - CLEAR TASK SET - aborts all Tasks in the appropriate task set as defined by the TST field in the Control mode page (see [SPC3]).
- 5     -       LOGICAL UNIT RESET
- 6     -       TARGET WARM RESET
- 7 -       TARGET COLD RESET
- 8     - TASK REASSIGN - reassigns connection allegiance for the task identified by the Initiator Task Tag field to this connection, thus resuming the iSCSI exchanges for the task.

For all these functions, the Task Management function response MUST be returned as detailed in Section 11.6. All these functions apply to the referenced tasks regardless of whether they are proper SCSI tasks or tagged iSCSI operations. Task management requests must act on all the commands from the same session having a CmdSN lower than the task management CmdSN. LOGICAL UNIT RESET, TARGET WARM RESET and TARGET COLD RESET may affect commands from other sessions or commands from the same session regardless of their CmdSN value.

If the task management request is marked for immediate delivery, it must be considered immediately for execution, but the operations involved (all or part of them) may be postponed to allow the target to receive all relevant tasks. According to [SAM2], for all the tasks covered by the Task Management response (i.e., with CmdSN lower than the task management command CmdSN) but except the Task Management response to a TASK REASSIGN, additional responses MUST NOT be delivered to the SCSI layer after the Task Management response. The iSCSI initiator MAY deliver to the SCSI layer all responses received before the Task Management response (i.e., it is a matter of implementation if the SCSI

responses, received before the Task Management response but after the task management request was issued, are delivered to the SCSI layer by the iSCSI layer in the initiator). The iSCSI target MUST ensure that no responses for the tasks covered by a task management function are delivered to the iSCSI initiator after the Task Management response except for a task covered by a TASK REASSIGN.

For ABORT TASK SET and CLEAR TASK SET, the issuing initiator MUST continue to respond to all valid target transfer tags (received via R2T, Text Response, NOP-In, or SCSI Data-in PDUs) related to the affected task set, even after issuing the task management request. The issuing initiator SHOULD however terminate (i.e., by setting the F-bit to 1) these response sequences as quickly as possible. The target on its part MUST wait for responses on all affected target transfer tags before acting on either of these two task management requests. In case all or part of the response sequence is not received (due to digest errors) for a valid TTT, the target MAY treat it as a case of within-command error recovery class (see Section 6.1.4.1 - "Recovery Within-command") if it is supporting ErrorRecoveryLevel >= 1, or alternatively may drop the connection to complete the requested task set function.

If an ABORT TASK is issued for a task created by an immediate command then RefCmdSN MUST be that of the Task Management request itself (i.e. CmdSN and RefCmdSN are equal); otherwise RefCmdSN MUST be set to the CmdSN of the task to be aborted (lower than CmdSN).

If the connection is still active (it is not undergoing an implicit or explicit logout), ABORT TASK MUST be issued on the same connection to which the task to be aborted is allegiant at the time the Task Management Request is issued. If the connection is implicitly or explicitly logged out (i.e., no other request will be issued on the failing connection and no other response will be received on the failing connection), then an ABORT TASK function request may be issued on another connection. This Task Management request will then establish a new allegiance for the command to be aborted as well as abort it (i.e., the task to be aborted will not have to be retried or reassigned, and its status,

if sent but not acknowledged, will be resent followed by the Task Management response).

At the target an ABORT TASK function MUST NOT be executed on a Task Management request; such a request MUST result in Task Management response of "Function rejected".

For the LOGICAL UNIT RESET function, the target MUST behave as dictated by the Logical Unit Reset function in [SAM2].

The implementation of the TARGET WARM RESET function and the TARGET COLD RESET function is OPTIONAL and when implemented, should act as described below. The TARGET WARM RESET is also subject to SCSI access controls on the requesting initiator as defined in [SPC3]. When authorization fails at the target, the appropriate response as described in Target MUST be returned by the target. The TARGET COLD RESET function is not subject to SCSI access controls, but its execution privileges may be managed by iSCSI mechanisms such as login authentication.

When executing the TARGET WARM RESET and TARGET COLD RESET functions, the target cancels all pending operations on all Logical Units known by the issuing initiator. Both functions are equivalent to the Target Reset function specified by [SAM2]. They can affect many other initiators logged in with the servicing SCSI target port.

The target MUST treat the TARGET COLD RESET function additionally as a power on event, thus terminating all of its TCP connections to all initiators (all sessions are terminated). For this reason, the Service Response (defined by [SAM2]) for this SCSI task management function may not be reliably delivered to the issuing initiator port.

For the TASK REASSIGN function, the target should reassign the connection allegiance to this new connection (and thus resume iSCSI exchanges for the task). TASK REASSIGN MUST ONLY be received by the target after the connection on which the command was previously executing has been successfully logged-out. The Task Management response MUST be issued before the reassignment becomes effective.

For additional usage semantics see Section 6.2 - "Retry and Reassign in Recovery".

At the target a TASK REASSIGN function request MUST NOT be executed to reassign the connection allegiance of a Task Management function request, an active text negotiation task, or a Logout task; such a request MUST result in Task Management response of "Function rejected".

TASK REASSIGN MUST be issued as an immediate command.

#### 11.5.2. TotalAHSLength and DataSegmentLength

For this PDU, TotalAHSLength and DataSegmentLength MUST be 0.

#### 11.5.3. LUN

This field is required for functions that address a specific LU (ABORT TASK, CLEAR TASK SET, ABORT TASK SET, CLEAR ACA, LOGICAL UNIT RESET) and is reserved in all others.

#### 11.5.4. Referenced Task Tag

The Initiator Task Tag of the task to be aborted for the ABORT TASK function or reassigned for the TASK REASSIGN function. For all the other functions this field MUST be set to the reserved value 0xffffffff.

#### 11.5.5. RefCmdSN

If an ABORT TASK is issued for a task created by an immediate command then RefCmdSN MUST be that of the Task Management request itself (i.e. CmdSN and RefCmdSN are equal).

For an ABORT TASK of a task created by non-immediate command RefCmdSN MUST be set to the CmdSN of the task identified by the Referenced Task Tag field. Targets must use this field as described in Res when the task identified by the Referenced Task Tag field is not with the target.

Otherwise, this field is reserved.

## 11.5.6. ExpDataSN

For recovery purposes, the iSCSI target and initiator maintain a data acknowledgement reference number - the first input DataSN number unacknowledged by the initiator. When issuing a new command, this number is set to 0. If the function is TASK REASSIGN, which establishes a new connection allegiance for a previously issued Read or Bidirectional command, ExpDataSN will contain an updated data acknowledgement reference number or the value 0; the latter indicating that the data acknowledgement reference number is unchanged. The initiator MUST discard any data PDUs from the previous execution that it did not acknowledge and the target MUST transmit all Data-in PDUs (if any) starting with the data acknowledgement reference number. The number of retransmitted PDUs may or may not be the same as the original transmission depending on if there was a change in MaxRecvDataSegmentLength in the reassignment. The target MAY also send no more Data-In PDUs if all data has been acknowledged.

The value of ExpDataSN MUST be 0 or higher than the DataSN of the last acknowledged Data-In PDU, but not larger than DataSN+1 of the last Data-IN PDU sent by the target. Any other value MUST be ignored by the target.

For other functions this field is reserved.

## 11.6. Task Management Function Response

iSCSI (Consolidated)																																	3/11/11
0								1								2								3									
Byte/	0	1	2	3	4	5	6	7	0	1	2	3	4	5	6	7	0	1	2	3	4	5	6	7	0	1	2	3	4	5	6	7	
0	.	.	.	0x22				1	Reserved							Response							Reserved										
4	TotalAHSLength								DataSegmentLength																								
8	Reserved																																/
16	Initiator Task Tag																																/
20	Reserved																																/
24	StatSN																																/
28	ExpCmdSN																																/
32	MaxCmdSN																																/
36	Reserved																																/
48	Header-Digest (Optional)																																/

For the functions ABORT TASK, ABORT TASK SET, CLEAR ACA, CLEAR TASK SET, LOGICAL UNIT RESET, TARGET COLD RESET, TARGET WARM RESET and TASK REASSIGN, the target performs the requested Task Management function and sends a Task Management response back to the initiator. For TASK REASSIGN, the new connection allegiance MUST ONLY become effective at the target after the target issues the Task Management Response.

#### 11.6.1. Response

The target provides a Response, which may take on the following values:

- 0 - Function complete.
- 1 - Task does not exist.



	iSCSI (Consolidated)	3/11/11
2	- LUN does not exist.	
3	- Task still allegiant.	
4	- Task allegiance reassignment not supported.	
5	- Task management function not supported.	
6	- Function authorization failed.	
255	- Function rejected.	

All other values are reserved.

For a discussion on usage of response codes 3 and 4, see Section 6.2.2 - "Allegiance Reassignment".

For the TARGET COLD RESET and TARGET WARM RESET functions, the target cancels all pending operations across all Logical Units known to the issuing initiator. For the TARGET COLD RESET function, the target MUST then close all of its TCP connections to all initiators (terminates all sessions).

The mapping of the response code into a SCSI service response code value, if needed, is outside the scope of this document. However, in symbolic terms Response values 0 and 1 map to the SCSI service response of FUNCTION COMPLETE. All other Response values map to the SCSI service response of FUNCTION REJECTED. If a Task Management function response PDU does not arrive before the session is terminated, the SCSI service response is SERVICE DELIVERY OR TARGET FAILURE.

The response to ABORT TASK SET and CLEAR TASK SET MUST only be issued by the target after all of the commands affected have been received by the target, the corresponding task management functions have been executed by the SCSI target, and the delivery of all responses delivered until the task management function completion have been confirmed (acknowledged through ExpStatSN) by the initiator on all connections of this session. For the exact timeline of events, refer to Section 4.2.3.3 and Section 4.2.3.4.

For the ABORT TASK function,

If the Referenced Task Tag identifies a valid task leading to a successful termination, then targets must return the "Function complete" response.

If the Referenced Task Tag does not identify an existing task, but if the CmdSN indicated by the RefCmdSN field in the Task Management function request is within the valid CmdSN window and less than the CmdSN of the Task Management function request itself, then targets must consider the CmdSN received and return the "Function complete" response.

If the Referenced Task Tag does not identify an existing task and if the CmdSN indicated by the RefCmdSN field in the Task Management function request is outside the valid CmdSN window, then targets must return the "Task does not exist" response.

For response semantics on function types that can potentially impact multiple active tasks on the target, see Section 4.2.3.

#### 11.6.2. TotalAHSLength and DataSegmentLength

For this PDU TotalAHSLength and DataSegmentLength MUST be 0.

#### 11.7. SCSI Data-out & SCSI Data-in

The SCSI Data-out PDU for WRITE operations has the following format:

iSCSI (Consolidated)																												3/11/11											
0								1								2								3															
Byte/	0	1	2	3	4	5	6	7	0	1	2	3	4	5	6	7	0	1	2	3	4	5	6	7	0	1	2	3	4	5	6	7							
0		.		.		0	x	0	5		F		Reserved																										
4	TotalAHSLength								DataSegmentLength																														
8	LUN or Reserved																																						
12																																							
16	Initiator Task Tag																																						
20	Target Transfer Tag or 0xffffffff																																						
24	Reserved																																						
28	ExpStatSN																																						
32	Reserved																																						
36	DataSN																																						
40	Buffer Offset																																						
44	Reserved																																						
48	Header-Digest (Optional)																																						
/ DataSegment																																/							
+ /																																/							
Data-Digest (Optional)																																							

The SCSI Data-in PDU for READ operations has the following format:

iSCSI (Consolidated)																												3/11/11							
0								1								2								3											
Byte/	0	1	2	3	4	5	6	7	0	1	2	3	4	5	6	7	0	1	2	3	4	5	6	7	0	1	2	3	4	5	6	7			
0	.	.	.	0x25				F	A	0	0	0	0	O	U	S	Reserved						Status or Rsvd												
4	TotalAHSLength								DataSegmentLength																										
8	LUN or Reserved																																		
12																																			
16	Initiator Task Tag																																		
20	Target Transfer Tag or 0xffffffff																																		
24	StatSN or Reserved																																		
28	ExpCmdSN																																		
32	MaxCmdSN																																		
36	DataSN																																		
40	Buffer Offset																																		
44	Residual Count																																		
48	Header-Digest (Optional)																																		
/ DataSegment																																			
+ /																																			
Data-Digest (Optional)																																			

Status can accompany the last Data-in PDU if the command did not end with an exception (i.e., the status is "good status" - GOOD, CONDITION MET or INTERMEDIATE CONDITION MET). The presence of status (and of a residual count) is signaled though the S flag bit. Although targets MAY choose to send even non-exception

status in separate responses, initiators MUST support non-exception status in Data-In PDUs.

#### 11.7.1. F (Final) Bit

For outgoing data, this bit is 1 for the last PDU of unsolicited data or the last PDU of a sequence that answers an R2T.

For incoming data, this bit is 1 for the last input (read) data PDU of a sequence. Input can be split into several sequences, each having its own F bit. Splitting the data stream into sequences does not affect DataSN counting on Data-In PDUs. It MAY be used as a "change direction" indication for Bidirectional operations that need such a change.

DataSegmentLength MUST NOT exceed MaxRecvDataSegmentLength for the direction it is sent and the total of all the DataSegmentLength of all PDUs in a sequence MUST NOT exceed MaxBurstLength (or FirstBurstLength for unsolicited data). However the number of individual PDUs in a sequence (or in total) may be higher than the MaxBurstLength (or FirstBurstLength) to MaxRecvDataSegmentLength ratio (as PDUs may be limited in length by the sender capabilities). Using DataSegmentLength of 0 may increase beyond what is reasonable for the number of PDUs and should therefore be avoided.

For Bidirectional operations, the F bit is 1 for both the end of the input sequences and the end of the output sequences.

#### 11.7.2. A (Acknowledge) bit

For sessions with ErrorRecoveryLevel 1 or higher, the target sets this bit to 1 to indicate that it requests a positive acknowledgement from the initiator for the data received. The target should use the A bit moderately; it MAY only set the A bit to 1 once every MaxBurstLength bytes, or on the last Data-In PDU that concludes the entire requested read data transfer for the task from the target's perspective, and it MUST NOT do so more frequently. The target MUST NOT set to 1 the A bit for sessions with ErrorRecoveryLevel=0. The initiator MUST ignore the A bit set to 1 for sessions with ErrorRecoveryLevel=0.

On receiving a Data-In PDU with the A bit set to 1 on a session with ErrorRecoveryLevel greater than 0, if there are no holes in the read data until that Data-In PDU, the initiator MUST issue a SNACK of type DataACK except when it is able to acknowledge the status for the task immediately via ExpStatSN on other outbound PDUs if the status for the task is also received. In the latter case (acknowledgement through ExpStatSN), sending a SNACK of type DataACK in response to the A bit is OPTIONAL, but if it is done, it must not be sent after the status acknowledgement through ExpStatSN. If the initiator has detected holes in the read data prior to that Data-In PDU, it MUST postpone issuing the SNACK of type DataACK until the holes are filled. An initiator also MUST NOT acknowledge the status for the task before those holes are filled. A status acknowledgement for a task that generated the Data-In PDUs is considered by the target as an implicit acknowledgement of the Data-In PDUs if such an acknowledgement was requested by the target.

#### 11.7.3. Flags (byte 1)

The last SCSI Data packet sent from a target to an initiator for a SCSI command that completed successfully (with a status of GOOD, CONDITION MET, INTERMEDIATE or INTERMEDIATE CONDITION MET) may also optionally contain the Status for the data transfer. In this case, Sense Data cannot be sent together with the Command Status. If the command is completed with an error, then the response and sense data MUST be sent in a SCSI Response PDU (i.e., MUST NOT be sent in a SCSI Data packet). For Bidirectional commands, the status MUST be sent in a SCSI Response PDU.

bit 2-4 - Reserved.

bit 5-6 - used the same as in a SCSI Response. These bits are only valid when S is set to 1. For details see SNACK .

bit 7 S (status)- set to indicate that the Command Status field contains status. If this bit is set to 1, the F bit MUST also be set to 1.

The fields StatSN, Status, and Residual Count only have meaningful content if the S bit is set to 1 and their values are defined in SNACK .

#### 11.7.4. Target Transfer Tag and LUN

On outgoing data, the Target Transfer Tag is provided to the target if the transfer is honoring an R2T. In this case, the Target Transfer Tag field is a replica of the Target Transfer Tag provided with the R2T.

On incoming data, the Target Transfer Tag and LUN MUST be provided by the target if the A bit is set to 1; otherwise they are reserved. The Target Transfer Tag and LUN are copied by the initiator into the SNACK of type DataACK that it issues as a result of receiving a SCSI Data-in PDU with the A bit set to 1.

The Target Transfer Tag values are not specified by this protocol except that the value 0xffffffff is reserved and means that the Target Transfer Tag is not supplied. If the Target Transfer Tag is provided, then the LUN field MUST hold a valid value and be consistent with whatever was specified with the command; otherwise, the LUN field is reserved.

#### 11.7.5. DataSN

For input (read) or bidirectional Data-In PDUs, the DataSN is the input PDU number within the data transfer for the command identified by the Initiator Task Tag.

R2T and Data-In PDUs, in the context of bidirectional commands, share the numbering sequence (see Section 3.2.2.4 - "Data Sequencing").

For output (write) data PDUs, the DataSN is the Data-Out PDU number within the current output sequence. The current output sequence is either identified by the Initiator Task Tag (for unsolicited data) or is a data sequence generated for one R2T (for data solicited through R2T).

#### 11.7.6. Buffer Offset

The Buffer Offset field contains the offset of this PDU payload data within the complete data transfer. The sum of the buffer

offset and length should not exceed the expected transfer length for the command.

The order of data PDUs within a sequence is determined by DataPDUInOrder. When set to Yes, it means that PDUs have to be in increasing Buffer Offset order and overlays are forbidden.

The ordering between sequences is determined by DataSequenceInOrder. When set to Yes, it means that sequences have to be in increasing Buffer Offset order and overlays are forbidden.

#### 11.7.7. DataSegmentLength

This is the data payload length of a SCSI Data-In or SCSI Data-Out PDU. The sending of 0 length data segments should be avoided, but initiators and targets MUST be able to properly receive 0 length data segments.

The Data Segments of Data-in and Data-out PDUs SHOULD be filled to the integer number of 4 byte words (real payload) unless the F bit is set to 1.



## 11.8. Ready To Transfer (R2T)

Byte/ /	0	1	2	3
	0 1 2 3 4 5 6 7	0 1 2 3 4 5 6 7	0 1 2 3 4 5 6 7	0 1 2 3 4 5 6 7
0	0x31	1	Reserved	
4	TotalAHSLength	DataSegmentLength		
8	LUN			
12				
16	Initiator Task Tag			
20	Target Transfer Tag			
24	StatSN			
28	ExpCmdSN			
32	MaxCmdSN			
36	R2TSN			
40	Buffer Offset			
44	Desired Data Transfer Length			
48	Header-Digest (Optional)			

When an initiator has submitted a SCSI Command with data that passes from the initiator to the target (WRITE), the target may specify which blocks of data it is ready to receive. The target may request that the data blocks be delivered in whichever order is convenient for the target at that particular instant. This information is passed from the target to the initiator in the Ready To Transfer (R2T) PDU.

In order to allow write operations without an explicit initial R2T, the initiator and target MUST have negotiated the key InitialR2T to No during Login.

An R2T MAY be answered with one or more SCSI Data-out PDUs with a matching Target Transfer Tag. If an R2T is answered with a single Data-out PDU, the Buffer Offset in the Data PDU MUST be the same as the one specified by the R2T, and the data length of the Data PDU MUST be the same as the Desired Data Transfer Length specified in the R2T. If the R2T is answered with a sequence of Data PDUs, the Buffer Offset and Length MUST be within the range of those specified by R2T, and the last PDU MUST have the F bit set to 1. If the last PDU (marked with the F bit) is received before the Desired Data Transfer Length is transferred, a target MAY choose to Reject that PDU with "Protocol error" reason code. DataPDUInOrder governs the Data-Out PDU ordering. If DataPDUInOrder is set to Yes, the Buffer Offsets and Lengths for consecutive PDUs MUST form a continuous non-overlapping range and the PDUs MUST be sent in increasing offset order.

The target may send several R2T PDUs. It, therefore, can have a number of pending data transfers. The number of outstanding R2T PDUs are limited by the value of the negotiated key MaxOutstandingR2T. Within a task, outstanding R2Ts MUST be fulfilled by the initiator in the order in which they were received.

R2T PDUs MAY also be used to recover Data Out PDUs. Such an R2T (Recovery-R2T) is generated by a target upon detecting the loss of one or more Data-Out PDUs due to:

- Digest error
- Sequence error
- Sequence reception timeout

A Recovery-R2T carries the next unused R2TSN, but requests part of or the entire data burst that an earlier R2T (with a lower R2TSN) had already requested.

DataSequenceInOrder governs the buffer offset ordering in consecutive R2Ts. If DataSequenceInOrder is Yes, then consecutive R2Ts MUST refer to continuous non-overlapping ranges except for Recovery-R2Ts.

#### 11.8.1. TotalAHSLength and DataSegmentLength

For this PDU TotalAHSLength and DataSegmentLength MUST be 0.

#### 11.8.2. R2TSN

R2TSN is the R2T PDU input PDU number within the command identified by the Initiator Task Tag.

For bidirectional commands R2T and Data-In PDUs share the input PDU numbering sequence (see Section 3.2.2.4 - "Data Sequencing").

#### 11.8.3. StatSN

The StatSN field will contain the next StatSN. The StatSN for this connection is not advanced after this PDU is sent.

#### 11.8.4. Desired Data Transfer Length and Buffer Offset

The target specifies how many bytes it wants the initiator to send because of this R2T PDU. The target may request the data from the initiator in several chunks, not necessarily in the original order of the data. The target, therefore, also specifies a Buffer Offset that indicates the point at which the data transfer should begin, relative to the beginning of the total data transfer. The Desired Data Transfer Length MUST NOT be 0 and MUST NOT exceed MaxBurstLength.

#### 11.8.5. Target Transfer Tag

The target assigns its own tag to each R2T request that it sends to the initiator. This tag can be used by the target to easily identify the data it receives. The Target Transfer Tag and LUN are copied in the outgoing data PDUs and are only used by the target. There is no protocol rule about the Target Transfer Tag except that the value 0xffffffff is reserved and MUST NOT be sent by a target in an R2T.

## 11.9. Asynchronous Message

An Asynchronous Message may be sent from the target to the initiator without correspondence to a particular command. The target specifies the reason for the event and sense data.

		iSCSI (Consolidated)																																3/11/11	
Byte/ /		0								1								2								3									
		0	1	2	3	4	5	6	7	0	1	2	3	4	5	6	7	0	1	2	3	4	5	6	7	0	1	2	3	4	5	6	7		
0	.   .	0x32							1   Reserved																										
4		TotalAHSLength									DataSegmentLength																								
8		LUN or Reserved																																	
12																																			
16		0xffffffff																																	
20		Reserved																																	
24		StatSN																																	
28		ExpCmdSN																																	
32		MaxCmdSN																																	
36		AsyncEvent									AsyncVCode									Parameter1 or Reserved															
40		Parameter2 or Reserved																	Parameter3 or Reserved																
44		Reserved																																	
48		Header-Digest (Optional)																																	
/		DataSegment - Sense Data and iSCSI Event Data																																	
+ /																																			
		Data-Digest (Optional)																																	

Some Asynchronous Messages are strictly related to iSCSI while others are related to SCSI [SAM2].

StatSN counts this PDU as an acknowledgeable event (StatSN is advanced), which allows for initiator and target state synchronization.

#### 11.9.1. AsyncEvent

The codes used for iSCSI Asynchronous Messages (events) are:

- 0 (SCSI\_ASYNC) - a SCSI Asynchronous Event is reported in the sense data. Sense Data that accompanies the report, in the data segment, identifies the condition. The sending of a SCSI Event (Asynchronous Event Reporting in SCSI terminology) is dependent on the target support for SCSI asynchronous event reporting (see [SAM2]) as indicated in the standard INQUIRY data (see [SPC3]). Its use may be enabled by parameters in the SCSI Control mode page (see [SPC3]).
- 1 (REQUEST\_LOGOUT) - target requests Logout. This Async Message MUST be sent on the same connection as the one requesting to be logged out. The initiator MUST honor this request by issuing a Logout as early as possible, but no later than Parameter3 seconds. Initiator MUST send a Logout with a reason code of "Close the connection" OR "Close the session" to close all the connections. Once this message is received, the initiator SHOULD NOT issue new iSCSI commands on the connection to be logged out. The target MAY reject any new I/O requests that it receives after this Message with the reason code "Waiting for Logout". If the initiator does not Logout in Parameter3 seconds, the target should send an Async PDU with iSCSI event code "Dropped the connection" if possible, or simply terminate the transport connection. Parameter1 and Parameter2 are reserved.
- 2 (CONNECTION\_DROP) - target indicates it will drop the connection.  
The Parameter1 field indicates the CID of the connection going to be dropped.

The Parameter2 field (Time2Wait) indicates, in seconds, the minimum time to wait before attempting to reconnect or reassign.

The Parameter3 field (Time2Retain) indicates the maximum time allowed to reassign commands after the initial wait (in Parameter2).

If the initiator does not attempt to reconnect and/or reassign the outstanding commands within the time specified by Parameter3, or if Parameter3 is 0, the target will terminate all outstanding commands on this connection. In this case, no other responses should be expected from the target for the outstanding commands on this connection.

A value of 0 for Parameter2 indicates that reconnect can be attempted immediately.

3 (SESSION\_DROP) - target indicates it will drop all the connections of this session.

Parameter1 field is reserved.

The Parameter2 field (Time2Wait) indicates, in seconds, the minimum time to wait before attempting to reconnect. The Parameter3 field (Time2Retain) indicates the maximum time allowed to reassign commands after the initial wait (in Parameter2).

If the initiator does not attempt to reconnect and/or reassign the outstanding commands within the time specified by Parameter3, or if Parameter3 is 0, the session is terminated. In this case, the target will terminate all

outstanding commands in this session; no other responses should be expected from the target for the outstanding commands in this session. A value of 0 for Parameter2 indicates that reconnect can be attempted immediately.

- 4 (RENEGOTIATE) - target requests parameter negotiation on this connection. The initiator MUST honor this request by issuing a Text Request (that can be empty) on the same connection as early as possible, but no later than Parameter3 seconds, unless a Text Request is already pending on the connection, or by issuing a Logout Request. If the initiator does not issue a Text Request the target may reissue the Asynchronous Message requesting parameter negotiation.
- 5 (FAST\_ABORT) - all active tasks for LU with a matching LUN field in the Async Message PDU are being terminated. The receiving initiator iSCSI layer MUST respond to this Message by taking the following steps in order.
- Stop Data-Out transfers on that connection for all active TTTs for the affected LUN quoted in the Async Message PDU.
  - Acknowledge the StatSN of the Async Message PDU via a NOP-Out PDU with ITT=0xffffffff (i.e., non-ping flavor), while copying the LUN field from the Async Message to NOP-Out.

This value of AsyncEvent however MUST NOT be used on an iSCSI session unless the new TaskReporting text key defined in Section 13.23 was negotiated to FastAbort on the session.

255 - vendor-specific iSCSI Event. The AsyncVCode details the vendor code, and data MAY accompany the report.

All other event codes are reserved.



## 11.9.2. AsyncVCode

AsyncVCode is a vendor specific detail code that is only valid if the AsyncEvent field indicates a vendor specific event. Otherwise, it is reserved.

## 11.9.3. LUN

The LUN field MUST be valid if AsyncEvent is 0. Otherwise, this field is reserved.

## 11.9.4. Sense Data and iSCSI Event Data

For a SCSI event, this data accompanies the report in the data segment and identifies the condition.

For an iSCSI event, additional vendor-unique data MAY accompany the Async event. Initiators MAY ignore the data when not understood while processing the rest of the PDU.

If the DataSegmentLength is not 0, the format of the DataSegment is as follows:

Byte/ /	0	1	2	3
	0 1 2 3 4 5 6 7	0 1 2 3 4 5 6 7	0 1 2 3 4 5 6 7	0 1 2 3 4 5 6 7
0	SenseLength	Sense Data		
x	Sense Data			/
y	iSCSI Event Data			/
/				/
z				

## 11.9.4.1. SenseLength

This is the length of Sense Data. When the Sense Data field is empty (e.g., the event is not a SCSI event) SenseLength is 0.

## 11.10. Text Request

The Text Request is provided to allow for the exchange of information and for future extensions. It permits the initiator to inform a target of its capabilities or to request some special operations.

Byte/ /	0	1	2	3
	0 1 2 3 4 5 6 7	0 1 2 3 4 5 6 7	0 1 2 3 4 5 6 7	0 1 2 3 4 5 6 7
0	. I  0x04	F C  Reserved		
4	TotalAHSLength	DataSegmentLength		
8	LUN or Reserved			
12				
16	Initiator Task Tag			
20	Target Transfer Tag or 0xffffffff			
24	CmdSN			
28	ExpStatSN			
32	Reserved			/
+	/			/
48	Header-Digest (Optional)			
+	/ DataSegment (Text)			/
+	/			/
	Data-Digest (Optional)			
+				

An initiator MUST NOT have more than one outstanding Text Request on a connection at any given time.

On a connection failure, an initiator must either explicitly abort any active allegiant text negotiation task or must cause such a task to be implicitly terminated by the target.

#### 11.10.1. F (Final) Bit

When set to 1, indicates that this is the last or only text request in a sequence of Text Requests; otherwise, it indicates that more Text Requests will follow.

#### 11.10.2. C (Continue) Bit

When set to 1, indicates that the text (set of key=value pairs) in this Text Request is not complete (it will be continued on subsequent Text Requests); otherwise, it indicates that this Text Request ends a set of key=value pairs. A Text Request with the C bit set to 1 MUST have the F bit set to 0.

#### 11.10.3. Initiator Task Tag

The initiator assigned identifier for this Text Request. If the command is sent as part of a sequence of text requests and responses, the Initiator Task Tag MUST be the same for all the requests within the sequence (similar to linked SCSI commands). The I bit for all requests in a sequence also MUST be the same.

#### 11.10.4. Target Transfer Tag

When the Target Transfer Tag is set to the reserved value 0xffffffff, it tells the target that this is a new request and the target resets any internal state associated with the Initiator Task Tag (resets the current negotiation state).

The target sets the Target Transfer Tag in a text response to a value other than the reserved value 0xffffffff whenever it indicates that it has more data to send or more operations to perform that are associated with the specified Initiator Task Tag. It MUST do so whenever it sets the F bit to 0 in the response. By copying the Target Transfer Tag from the response to the next Text Request, the initiator tells the target to continue the operation for the specific Initiator Task Tag. The initiator MUST ignore the

Target Transfer Tag in the Text Response when the F bit is set to 1.

This mechanism allows the initiator and target to transfer a large amount of textual data over a sequence of text-command/text-response exchanges, or to perform extended negotiation sequences.

If the Target Transfer Tag is not 0xffffffff, the LUN field MUST be sent by the target in the Text Response.

A target MAY reset its internal negotiation state if an exchange is stalled by the initiator for a long time or if it is running out of resources.

Long text responses are handled as in the following example:

```
I->T Text SendTargets=All (F=1,TTT=0xffffffff)
```

```
T->I Text <part 1> (F=0,TTT=0x12345678)
```

```
I->T Text <empty> (F=1, TTT=0x12345678)
```

```
T->I Text <part 2> (F=0, TTT=0x12345678)
```

```
I->T Text <empty> (F=1, TTT=0x12345678)
```

```
...
```

```
T->I Text <part n> (F=1, TTT=0xffffffff)
```

#### 11.10.5. Text

The data lengths of a text request MUST NOT exceed the iSCSI target MaxRecvDataSegmentLength (a per connection and per direction negotiated parameter). The text format is specified in Section 5.2 - "Text Mode Negotiation".

Chapter 11 and Chapter 12 list some basic Text key=value pairs, some of which can be used in Login Request/Response and some in Text Request/Response.

A key=value pair can span Text request or response boundaries. A key=value pair can start in one PDU and continue on the next. In other words the end of a PDU does not necessarily signal the end of a key=value pair.

The target responds by sending its response back to the initiator. The response text format is similar to the request text format. The text response MAY refer to key=value pairs presented in an earlier text request and the text in the request may refer to earlier responses.

Chapter 5 details the rules for the Text Requests and Responses.

Text operations are usually meant for parameter setting/negotiations, but can also be used to perform some long lasting operations.

Text operations that take a long time should be placed in their own Text request.

#### 11.11. Text Response

The Text Response PDU contains the target's responses to the initiator's Text request. The format of the Text field matches that of the Text request.

		iSCSI (Consolidated)																												3/11/11				
Byte/ /		0								1								2								3								
		0	1	2	3	4	5	6	7	0	1	2	3	4	5	6	7	0	1	2	3	4	5	6	7	0	1	2	3	4	5	6	7	
0	.   .   0x24									F	C																							
4	TotalAHSLength																																	
8	LUN or Reserved																																	
12																																		
16	Initiator Task Tag																																	
20	Target Transfer Tag or 0xffffffff																																	
24	StatSN																																	
28	ExpCmdSN																																	
32	MaxCmdSN																																	
36	Reserved																																	
+	/																																	
48	Header-Digest (Optional)																																	
	/ DataSegment (Text)																																	
+	/																																	
	Data-Digest (Optional)																																	

#### 11.11.1. F (Final) Bit

When set to 1, in response to a Text Request with the Final bit set to 1, the F bit indicates that the target has finished the whole operation. Otherwise, if set to 0 in response to a Text Request with the Final Bit set to 1, it indicates that the target has more work to do (invites a follow-on text request). A Text Response with the F bit set to 1 in response to a Text Request with the F bit set to 0 is a protocol error.

A Text Response with the F bit set to 1 MUST NOT contain key=value pairs that may require additional answers from the initiator.

A Text Response with the F bit set to 1 MUST have a Target Transfer Tag field set to the reserved value of 0xffffffff.

A Text Response with the F bit set to 0 MUST have a Target Transfer Tag field set to a value other than the reserved 0xffffffff.

#### 11.11.2. C (Continue) Bit

When set to 1, indicates that the text (set of key=value pairs) in this Text Response is not complete (it will be continued on subsequent Text Responses); otherwise, it indicates that this Text Response ends a set of key=value pairs. A Text Response with the C bit set to 1 MUST have the F bit set to 0.

#### 11.11.3. Initiator Task Tag

The Initiator Task Tag matches the tag used in the initial Text Request.

#### 11.11.4. Target Transfer Tag

When a target has more work to do (e.g., cannot transfer all the remaining text data in a single Text Response or has to continue the negotiation) and has enough resources to proceed, it MUST set the Target Transfer Tag to a value other than the reserved value of 0xffffffff. Otherwise, the Target Transfer Tag MUST be set to 0xffffffff.

When the Target Transfer Tag is not 0xffffffff, the LUN field may be significant.

The initiator MUST copy the Target Transfer Tag and LUN in its next request to indicate that it wants the rest of the data.

When the target receives a Text Request with the Target Transfer Tag set to the reserved value of 0xffffffff, it resets its

internal information (resets state) associated with the given Initiator Task Tag (restarts the negotiation).

When a target cannot finish the operation in a single Text Response, and does not have enough resources to continue, it rejects the Text Request with the appropriate Reject code.

A target may reset its internal state associated with an Initiator Task Tag (the current negotiation state), state expressed through the Target Transfer Tag if the initiator fails to continue the exchange for some time. The target may reject subsequent Text Requests with the Target Transfer Tag set to the "stale" value.

#### 11.11.5. StatSN

The target StatSN variable is advanced by each Text Response sent.

#### 11.11.6. Text Response Data

The data lengths of a text response MUST NOT exceed the iSCSI initiator MaxRecvDataSegmentLength (a per connection and per direction negotiated parameter).

The text in the Text Response Data is governed by the same rules as the text in the Text Request Data (see C (Con)).

Although the initiator is the requesting party and controls the request-response initiation and termination, the target can offer key=value pairs of its own as part of a sequence and not only in response to the initiator.

#### 11.12. Login Request

After establishing a TCP connection between an initiator and a target, the initiator MUST start a Login Phase to gain further access to the target's resources.

The Login Phase (see Chapter 5) consists of a sequence of Login requests and responses that carry the same Initiator Task Tag.

Login requests are always considered as immediate.



iSCSI (Consolidated)																												3/11/11																														
0								1								2								3																																		
Byte/	0	1	2	3	4	5	6	7	0	1	2	3	4	5	6	7	0	1	2	3	4	5	6	7	0	1	2	3	4	5	6	7																										
0	.	1	0x03					T	C	.	.	CSG	NSG	Version-max				Version-min																																								
4	TotalAHSLength								DataSegmentLength																																																	
8	ISID																																																									
12																	TSIH																																									
16	Initiator Task Tag																																																									
20	CID																Reserved																																									
24	CmdSN																																																									
28	ExpStatSN								or								Reserved																																									
32	Reserved																																																									
36	Reserved																																																									
40/	Reserved																																/																									
+																																	/																									
48/	DataSegment - Login Parameters in Text request Format																																	/																								
+																																		/																								

#### 11.12.1. T (Transit) Bit

If set to 1, indicates that the initiator is ready to transit to the next stage.

If the T bit is set to 1 and NSG is FullFeaturePhase, then this also indicates that the initiator is ready for the Final Login Response (see Chapter 5).

## 11.12.2. C (Continue) Bit

When set to 1, indicates that the text (set of key=value pairs) in this Login Request is not complete (it will be continued on subsequent Login Requests); otherwise, it indicates that this Login Request ends a set of key=value pairs. A Login Request with the C bit set to 1 MUST have the T bit set to 0.

## 11.12.3. CSG and NSG

Through these fields, Current Stage (CSG) and Next Stage (NSG), the Login negotiation requests and responses are associated with a specific stage in the session (SecurityNegotiation, LoginOperationalNegotiation, FullFeaturePhase) and may indicate the next stage to which they want to move (see Chapter 5). The next stage value is only valid when the T bit is 1; otherwise, it is reserved.

The stage codes are:

- 0 - SecurityNegotiation
- 1 - LoginOperationalNegotiation
- 3 - FullFeaturePhase

All other codes are reserved.

## 11.12.4. Version

The version number of the current draft is 0x00. As such, all devices MUST carry version 0x00 for both Version-min and Version-max.

## 11.12.4.1. Version-max

Maximum Version number supported.

All Login requests within the Login Phase MUST carry the same Version-max.

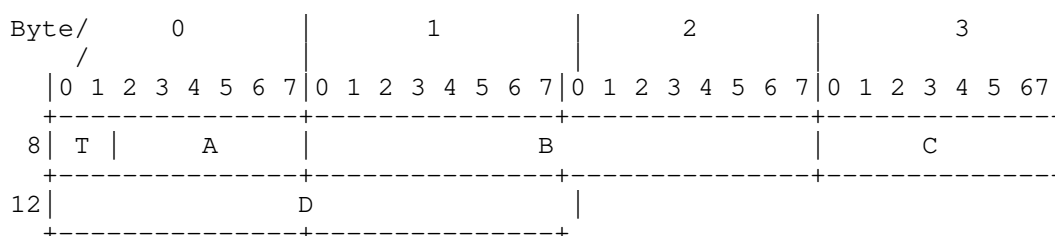
The target MUST use the value presented with the first login request.

#### 11.12.4.2. Version-min

All Login requests within the Login Phase MUST carry the same Version-min. The target MUST use the value presented with the first login request.

#### 11.12.5. ISID

This is an initiator-defined component of the session identifier and is structured as follows (see Section 10.1.1 for details):



The T field identifies the format and usage of A, B, C, and D as indicated below:

T

00b OUI-Format

A&B are a 22 bit OUI

(the I/G & U/L bits are omitted)

C&D 24 bit qualifier

01b EN - Format (IANA Enterprise Number)

A - Reserved

D - Qualifier

10b     "Random"

A - Reserved

B&C Random

D - Qualifier

11b     A,B,C&D Reserved

For the T field values 00b and 01b, a combination of A and B (for 00b) or B and C (for 01b) identifies the vendor or organization whose component (software or hardware) generates this ISID. A vendor or organization with one or more OUIs, or one or more Enterprise Numbers, MUST use at least one of these numbers and select the appropriate value for the T field when its components generate ISIDs. An OUI or EN MUST be set in the corresponding fields in network byte order (byte big-endian).

If the T field is 10b, B and C are set to a random 24-bit unsigned integer value in network byte order (byte big-endian). See [RFC3721] for how this affects the principle of "conservative reuse".

The Qualifier field is a 16 or 24-bit unsigned integer value that provides a range of possible values for the ISID within the selected namespace. It may be set to any value within the constraints specified in the iSCSI protocol (see Section 3.4.3 - "Consequences of the Model" and Section 9.1.1 - "Conservative Reuse of ISIDs").

The T field value of 11b is reserved.

If the ISID is derived from something assigned to a hardware adapter or interface by a vendor, as a preset default value, it MUST be configurable to a value assigned according to the SCSI

port behavior desired by the system in which it is installed (see Section 9.1.1 - "Conservative Reuse of ISIDs" and Section 9.1.2 - "iSCSI Name, ISID, and TPGT Use"). The resultant ISID MUST also be persistent over power cycles, reboot, card swap, etc.

#### 11.12.6. TSIH

TSIH must be set in the first Login Request. The reserved value 0 MUST be used on the first connection for a new session. Otherwise, the TSIH sent by the target at the conclusion of the successful login of the first connection for this session MUST be used. The TSIH identifies to the target the associated existing session for this new connection.

All Login requests within a Login Phase MUST carry the same TSIH.

The target MUST check the value presented with the first login request and act as specified in Section 5.3.1 - "Login Phase Start".

#### 11.12.7. Connection ID - CID

A unique ID for this connection within the session.

All Login requests within the Login Phase MUST carry the same CID.

The target MUST use the value presented with the first login request.

A Login request with a non-zero TSIH and a CID equal to that of an existing connection implies a logout of the connection followed by a Login (see Section 6.3.4). For the details of the implicit Logout Request, see Section 11.14.

#### 11.12.8. CmdSN

CmdSN is either the initial command sequence number of a session (for the first Login request of a session - the "leading" login), or the command sequence number in the command stream if the login is for a new connection in an existing session.

Examples:

- Login on a leading connection - if the leading login carries the CmdSN 123, all other login requests in the same login phase carry the CmdSN 123 and the first non-immediate command in FullFeaturePhase also carries the CmdSN 123.
- Login on other than a leading connection - if the current CmdSN at the time the first login on the connection is issued is 500, then that PDU carries CmdSN=500. Subsequent login requests that are needed to complete this login phase may carry a CmdSN higher than 500 if non-immediate requests that were issued on other connections in the same session advance CmdSN.

If the login request is a leading login request, the target MUST use the value presented in CmdSN as the target value for ExpCmdSN.

#### 11.12.9. ExpStatSN

For the first Login Request on a connection this is ExpStatSN for the old connection and this field is only valid if the Login request restarts a connection (see Section 5.3.4 - "Connection Reinstatement").

For subsequent Login Requests it is used to acknowledge the Login Responses with their increasing StatSN values.

#### 11.12.10. Login Parameters

The initiator MUST provide some basic parameters in order to enable the target to determine if the initiator may use the target's resources and the initial text parameters for the security exchange.

All the rules specified in Section 11.10.5 for text requests also hold for login requests. Keys and their explanations are listed in Chapter 11 (security negotiation keys) and Section 13 (operational parameter negotiation keys). All keys in Section 13, except for the X extension formats, MUST be supported by iSCSI initiators and targets. Keys in Section 12 only need to be

supported when the function to which they refer is mandatory to implement.

### 11.13. Login Response

The Login Response indicates the progress and/or end of the Login Phase.

Byte/ /	0	1	2	3
	0 1 2 3 4 5 6 7	0 1 2 3 4 5 6 7	0 1 2 3 4 5 6 7	0 1 2 3 4 5 6 7
0	. . 0x23	T C . .CSG NSG	Version-max	Version-active
4	TotalAHSLength	DataSegmentLength		
8	ISID			
12		TSIH		
16	Initiator Task Tag			
20	Reserved			
24	StatSN			
28	ExpCmdSN			
32	MaxCmdSN			
36	Status-Class	Status-Detail	Reserved	
40/	Reserved			/
+/				/
48/	DataSegment - Login Parameters in Text request Format			/
+/				/

#### 11.13.1. Version-max

This is the highest version number supported by the target.

All Login responses within the Login Phase MUST carry the same Version-max.

The initiator MUST use the value presented as a response to the first login request.

#### 11.13.2. Version-active

Indicates the highest version supported by the target and initiator. If the target does not support a version within the range specified by the initiator, the target rejects the login and this field indicates the lowest version supported by the target.

All Login responses within the Login Phase MUST carry the same Version-active.

The initiator MUST use the value presented as a response to the first login request.

#### 11.13.3. TSIH

The TSIH is the target assigned session identifying handle. Its internal format and content are not defined by this protocol except for the value 0 that is reserved. With the exception of the Login Final-Response in a new session, this field should be set to the TSIH provided by the initiator in the Login Request. For a new session, the target MUST generate a non-zero TSIH and ONLY return it in the Login Final-Response (see Section 5.3 - "Login Phase").

#### 11.13.4. StatSN

For the first Login Response (the response to the first Login Request), this is the starting status Sequence Number for the connection. The next response of any kind, including the next login response, if any, in the same Login Phase, will carry this number + 1. This field is only valid if the Status-Class is 0.



## 11.13.5. Status-Class and Status-Detail

The Status returned in a Login Response indicates the execution status of the Login Phase. The status includes:

Status-Class

Status-Detail

0 Status-Class indicates success.

A non-zero Status-Class indicates an exception. In this case, Status-Class is sufficient for a simple initiator to use when handling exceptions, without having to look at the Status-Detail. The Status-Detail allows finer-grained exception handling for more sophisticated initiators and for better information for logging.

The status classes are as follows:

0 - Success - indicates that the iSCSI target successfully received, understood, and accepted the request. The numbering fields (StatSN, ExpCmdSN, MaxCmdSN) are only valid if Status-Class is 0.

1 - Redirection - indicates that the initiator must take further action to complete the request. This is usually due to the target moving to a different address. All of the redirection status class responses MUST return one or more text key parameters of the type "TargetAddress", which indicates the target's new address. A redirection response MAY be issued by a target prior or after completing a security negotiation if a security negotiation is required. A redirection SHOULD be accepted by an initiator even without having the target complete a security negotiation if any security negotiation is required, and MUST be accepted by the initiator after the completion of the security negotiation if any security negotiation is required.

2 - Initiator Error (not a format error) - indicates that the initiator most likely caused the error. This MAY be due to a

request for a resource for which the initiator does not have permission. The request should not be tried again.

- 3 - Target Error - indicates that the target sees no errors in the initiator's login request, but is currently incapable of fulfilling the request. The initiator may re-try the same login request later.

The table below shows all of the currently allocated status codes. The codes are in hexadecimal; the first byte is the status class and the second byte is the status detail.

Status	Code (hex)	Description
Success	0000	Login is proceeding OK (*1).
Target moved temporarily	0101	The requested iSCSI Target Name (ITN) has temporarily moved to the address provided.
Target moved permanently	0102	The requested ITN has permanently moved to the address provided.
Initiator error	0200	Miscellaneous iSCSI initiator errors.
Authentication failure	0201	The initiator could not be successfully authenticated or target authentication is not supported.
Authorization failure	0202	The initiator is not allowed access to the given target.
Not found	0203	The requested ITN does not exist at this address.
Target removed	0204	The requested ITN has been removed and no forwarding address is provided.
Unsupported version	0205	The requested iSCSI version range is not supported by the target.
Too many connections	0206	Too many connections on this SSID.
Missing parameter	0207	Missing parameters (e.g., iSCSI Initiator and/or Target Name).
Can't include in session	0208	Target does not support session spanning to this connection (address).

Session type not supported	0209	Target does not support this type of session or not from this Initiator.
Session does not exist	020a	Attempt to add a connection to a non-existent session.
Invalid during login	020b	Invalid Request type during Login.
Target error	0300	Target hardware or software error.
Service unavailable	0301	The iSCSI service or target is not currently operational.
Out of resources	0302	The target has insufficient session, connection, or other resources.

(\*1) If the response T bit is 1 in both the request and the matching response, and the NSG is FullFeaturePhase in both the request and the matching response, the Login Phase is finished and the initiator may proceed to issue SCSI commands.

If the Status Class is not 0, the initiator and target MUST close the TCP connection.

If the target wishes to reject the login request for more than one reason, it should return the primary reason for the rejection.

#### 11.13.6. T (Transit) bit

The T bit is set to 1 as an indicator of the end of the stage. If the T bit is set to 1 and NSG is FullFeaturePhase, then this is also the Final Login Response (see Chapter 5). A T bit of 0 indicates a "partial" response, which means "more negotiation needed".

A login response with a T bit set to 1 MUST NOT contain key=value pairs that may require additional answers from the initiator within the same stage.

If the status class is 0, the T bit MUST NOT be set to 1 if the T bit in the request was set to 0.

#### 11.13.7. C (Continue) Bit

When set to 1, indicates that the text (set of key=value pairs) in this Login Response is not complete (it will be continued on subsequent Login Responses); otherwise, it indicates that this Login Response ends a set of key=value pairs. A Login Response with the C bit set to 1 MUST have the T bit set to 0.

#### 11.13.8. Login Parameters

The target MUST provide some basic parameters in order to enable the initiator to determine if it is connected to the correct port and the initial text parameters for the security exchange.

All the rules specified in Section 11.11.6 for text responses also hold for login responses. Keys and their explanations are listed in Chapter 11 (security negotiation keys) and Chapter 12 (operational parameter negotiation keys). All keys in Section 13, except for the X extension formats, MUST be supported by iSCSI initiators and targets. Keys in Section 12, only need to be supported when the function to which they refer is mandatory to implement.

#### 11.14. Logout Request

The Logout request is used to perform a controlled closing of a connection.

An initiator MAY use a logout request to remove a connection from a session or to close an entire session.

After sending the Logout request PDU, an initiator MUST NOT send any new iSCSI requests on the closing connection. If the Logout request is intended to close the session, new iSCSI requests MUST NOT be sent on any of the connections participating in the session.

When receiving a Logout request with the reason code of "close the connection" or "close the session", the target MUST terminate all pending commands, whether acknowledged via ExpCmdSN or not, on that connection or session respectively.

When receiving a Logout request with the reason code "remove connection for recovery", the target MUST discard all requests not yet acknowledged via ExpCmdSN that were issued on the specified connection, and suspend all data/status/R2T transfers on behalf of pending commands on the specified connection.

The target then issues the Logout response and half-closes the TCP connection (sends FIN). After receiving the Logout response and attempting to receive the FIN (if still possible), the initiator MUST completely close the logging-out connection. For the terminated commands, no additional responses should be expected.

A Logout for a CID may be performed on a different transport connection when the TCP connection for the CID has already been terminated. In such a case, only a logical "closing" of the iSCSI connection for the CID is implied with a Logout.

All commands that were not terminated or not completed (with status) and acknowledged when the connection is closed completely can be reassigned to a new connection if the target supports connection recovery.

If an initiator intends to start recovery for a failing connection, it MUST use the Logout request to "clean-up" the target end of a failing connection and enable recovery to start, or the Login request with a non-zero TSIH and the same CID on a new connection for the same effect. In sessions with a single connection, the connection can be closed and then a new connection reopened. A connection reinstatement login can be used for recovery (see Section 5.3.4 - "Connection Reinstatement").

A successful completion of a logout request with the reason code of "close the connection" or "remove the connection for recovery" results at the target in the discarding of unacknowledged commands received on the connection being logged out. These are commands that have arrived on the connection being logged out, but have not

been delivered to SCSI because one or more commands with a smaller CmdSN has not been received by iSCSI. See Section 3.2.2.1 - "Command Numbering and Acknowledging". The resulting holes in the command sequence numbers will have to be handled by appropriate recovery (see Chapter 6) unless the session is also closed.

The entire logout discussion in this section is also applicable for an implicit Logout realized by way of a connection reinstatement or session reinstatement. When a Login Request performs an implicit Logout, the implicit Logout is performed as if having the reason codes specified below:

Reason code	Type of implicit Logout
-----	
0	session reinstatement
1	connection reinstatement when the operational ErrorRecoveryLevel < 2
2	connection reinstatement when the operational ErrorRecoveryLevel = 2

iSCSI (Consolidated)																																3/11/11								
0								1								2								3																
Byte/																																								
/	0	1	2	3	4	5	6	7	0	1	2	3	4	5	6	7	0	1	2	3	4	5	6	7	0	1	2	3	4	5	6	7								
+																																								
0	.I 0x06								1	Reason Code							Reserved																							
+																																								
4	TotalAHSLength								DataSegmentLength																															
+																																								
8	Reserved																																							
+																																								
16	Initiator Task Tag																																							
+																																								
20	CID or Reserved																Reserved																							
+																																								
24	CmdSN																																							
+																																								
28	ExpStatSN																																							
+																																								
32	Reserved																																							
+																																								
48	Header-Digest (Optional)																																							
+																																								

#### 11.14.1. Reason Code

Reason Code indicates the reason for Logout as follows:

- 0 - close the session. All commands associated with the session (if any) are terminated.
  
- 1 - close the connection. All commands associated with connection (if any) are terminated.
  
- 2 - remove the connection for recovery. Connection is closed and all commands associated with it, if any, are to be prepared for a new allegiance.



All other values are reserved.

#### 11.14.2. TotalAHSLength and DataSegmentLength

For this PDU TotalAHSLength and DataSegmentLength MUST be 0.

#### 11.14.3. CID

This is the connection ID of the connection to be closed (including closing the TCP stream). This field is only valid if the reason code is not "close the session".

#### 11.14.4. ExpStatSN

This is the last ExpStatSN value for the connection to be closed.

#### 11.14.5. Implicit termination of tasks

A target implicitly terminates the active tasks due to the iSCSI protocol in the following cases:

When a connection is implicitly or explicitly logged out with the reason code of "Close the connection" and there are active tasks allegiant to that connection.

When a connection fails and eventually the connection state times out (state transition M1 in Section 7.2.2 - "State Transition Descriptions for Initiators and Targets") and there are active tasks allegiant to that connection.

When a successful recovery Logout is performed while there are active tasks allegiant to that connection, and those tasks eventually time out after the Time2Wait and Time2Retain periods without allegiance reassignment.

When a connection is implicitly or explicitly logged out with the reason code of "Close the session" and there are active tasks in that session.

If the tasks terminated in any of the above cases are SCSI tasks, they must be internally terminated as if with CHECK CONDITION status. This status is only meaningful for appropriately handling the internal SCSI state and SCSI side effects with respect to ordering because this status is never communicated back as a terminating status to the initiator. However additional actions may have to be taken at SCSI level depending on the SCSI context as defined by the SCSI standards (e.g., queued commands and ACA, UA for the next command on the I\_T nexus in cases a), b), and c), after the tasks are terminated, the target MUST report a Unit Attention condition on the next command processed on any connection for each affected I\_T\_L nexus with the status of CHECK CONDITION, and the ASC/ASCQ value of 47h/7Fh - "SOME COMMANDS CLEARED BY ISCSI PROTOCOL EVENT" - etc. - see [SAM4] and [SPC3]).

#### 11.15. Logout Response

The logout response is used by the target to indicate if the cleanup operation for the connection(s) has completed.

After Logout, the TCP connection referred by the CID MUST be closed at both ends (or all connections must be closed if the logout reason was session close).

iSCSI (Consolidated)																												3/11/11												
0								1								2								3																
Byte/	0	1	2	3	4	5	6	7	0	1	2	3	4	5	6	7	0	1	2	3	4	5	6	7	0	1	2	3	4	5	6	7								
0	.	.	.	0x26					1	Reserved							Response							Reserved																
4	TotalAHSLength								DataSegmentLength																															
8	Reserved																												/											
+																													/											
16	Initiator Task Tag																																							
20	Reserved																																							
24	StatSN																																							
28	ExpCmdSN																																							
32	MaxCmdSN																																							
36	Reserved																																							
40	Time2Wait																Time2Retain																							
44	Reserved																																							
48	Header-Digest (Optional)																																							

#### 11.15.1. Response

Logout response:

0 - connection or session closed successfully.

1 - CID not found.

2 - connection recovery is not supported. If Logout reason code was recovery and target does not support it as indicated by the ErrorRecoveryLevel.

3 - cleanup failed for various reasons.

#### 11.15.2. TotalAHSLength and DataSegmentLength

For this PDU TotalAHSLength and DataSegmentLength MUST be 0.

#### 11.15.3. Time2Wait

If the Logout response code is 0 and if the operational ErrorRecoveryLevel is 2, this is the minimum amount of time, in seconds, to wait before attempting task reassignment. If the Logout response code is 0 and if the operational ErrorRecoveryLevel is less than 2, this field is to be ignored.

This field is invalid if the Logout response code is 1.

If the Logout response code is 2 or 3, this field specifies the minimum time to wait before attempting a new implicit or explicit logout.

If Time2Wait is 0, the reassignment or a new Logout may be attempted immediately.

#### 11.15.4. Time2Retain

If the Logout response code is 0 and if the operational ErrorRecoveryLevel is 2, this is the maximum amount of time, in seconds, after the initial wait (Time2Wait), the target waits for the allegiance reassignment for any active task after which the task state is discarded. If the Logout response code is 0 and if the operational ErrorRecoveryLevel is less than 2, this field is to be ignored.

This field is invalid if the Logout response code is 1.

If the Logout response code is 2 or 3, this field specifies the maximum amount of time, in seconds, after the initial wait (Time2Wait), the target waits for a new implicit or explicit logout.

If it is the last connection of a session, the whole session state is discarded after Time2Retain.

If Time2Retain is 0, the target has already discarded the connection (and possibly the session) state along with the task states. No reassignment or Logout is required in this case.

## 11.16. SNACK Request

Byte/ /	0	1	2	3
	0 1 2 3 4 5 6 7	0 1 2 3 4 5 6 7	0 1 2 3 4 5 6 7	0 1 2 3 4 5 6 7
0	. . . 0x10	1 . . . Type	Reserved	
4	TotalAHSLength	DataSegmentLength		
8	LUN or Reserved			
12				
16	Initiator Task Tag or 0xffffffff			
20	Target Transfer Tag or SNACK Tag or 0xffffffff			
24	Reserved			
28	ExpStatSN			
32/	Reserved			/
+	/			/
40	BegRun			
44	RunLength			
48	Header-Digest (Optional)			

If the implementation supports `ErrorRecoveryLevel` greater than zero, it MUST support all SNACK types.

The SNACK is used by the initiator to request the retransmission of numbered-responses, data, or R2T PDUs from the target. The SNACK request indicates the numbered-responses or data "runs" whose retransmission is requested by the target, where the run starts with the first StatSN, DataSN, or R2TSN whose retransmission is requested and indicates the number of Status,

Data, or R2T PDUs requested including the first. 0 has special meaning when used as a starting number and length:

- When used in RunLength, it means all PDUs starting with the initial.
- When used in both BegRun and RunLength, it means all unacknowledged PDUs.

The numbered-response(s) or R2T(s), requested by a SNACK, MUST be delivered as exact replicas of the ones that the target transmitted originally except for the fields ExpCmdSN, MaxCmdSN, and ExpDataSN, which MUST carry the current values. R2T(s) requested by SNACK MUST also carry the current value of StatSN.

The numbered Data-In PDUs, requested by a Data SNACK MUST be delivered as exact replicas of the ones that the target transmitted originally except for the fields ExpCmdSN and MaxCmdSN, which MUST carry the current values and except for resegmentation (see Resegmentation).

Any SNACK that requests a numbered-response, Data, or R2T that was not sent by the target or was already acknowledged by the initiator, MUST be rejected with a reason code of "Protocol error".

#### 11.16.1. Type

This field encodes the SNACK function as follows:

- 0-Data/R2T SNACK - requesting retransmission of one or more Data-In or R2T PDUs.
- 1-Status SNACK - requesting retransmission of one or more numbered responses.

2-DataACK - positively acknowledges Data-In PDUs.

3-R-Data SNACK - requesting retransmission of Data-In PDUs with possible resegmentation and status tagging.

All other values are reserved.

Data/R2T SNACK, Status SNACK, or R-Data SNACK for a command MUST precede status acknowledgement for the given command.

#### 11.16.2. Data Acknowledgement

If an initiator operates at ErrorRecoveryLevel 1 or higher, it MUST issue a SNACK of type DataACK after receiving a Data-In PDU with the A bit set to 1. However, if the initiator has detected holes in the input sequence, it MUST postpone issuing the SNACK of type DataACK until the holes are filled. An initiator MAY ignore the A bit if it deems that the bit is being set aggressively by the target (i.e., before the MaxBurstLength limit is reached).

The DataACK is used to free resources at the target and not to request or imply data retransmission.

An initiator MUST NOT request retransmission for any data it had already acknowledged.

#### 11.16.3. Resegmentation

If the initiator MaxRecvDataSegmentLength changed between the original transmission and the time the initiator requests retransmission, the initiator MUST issue a R-Data SNACK (see Type). With R-Data SNACK, the initiator indicates that it discards all the unacknowledged data and expects the target to resend it. It also expects resegmentation. In this case, the retransmitted Data-In PDUs MAY be different from the ones originally sent in order to reflect changes in MaxRecvDataSegmentLength. Their DataSN starts with the BegRun of the last DataACK received by the target



if any was received; otherwise it starts with 0 and is increased by 1 for each resent Data-In PDU.

A target that has received a R-Data SNACK MUST return a SCSI Response that contains a copy of the SNACK Tag field from the R-Data SNACK in the SCSI Response SNACK Tag field as its last or only Response. For example, if it has already sent a response containing another value in the SNACK Tag field or had the status included in the last Data-In PDU, it must send a new SCSI Response PDU. If a target sends more than one SCSI Response PDU due to this rule, all SCSI responses must carry the same StatSN (see SNACK ). If an initiator attempts to recover a lost SCSI Response (with a Status-SNACK, see Type) when more than one response has been sent, the target will send the SCSI Response with the latest content known to the target, including the last SNACK Tag for the command.

For considerations in allegiance reassignment of a task to a connection with a different MaxRecvDataSegmentLength, refer to Section 6.2.2 - "Allegiance Reassignment".

#### 11.16.4. Initiator Task Tag

For Status SNACK and DataACK, the Initiator Task Tag MUST be set to the reserved value 0xffffffff. In all other cases, the Initiator Task Tag field MUST be set to the Initiator Task Tag of the referenced command.

#### 11.16.5. Target Transfer Tag or SNACK Tag

For an R-Data SNACK, this field MUST contain a value that is different from 0 or 0xffffffff and is unique for the task (identified by the Initiator Task Tag). This value MUST be copied by the iSCSI target in the last or only SCSI Response PDU it issues for the command.

For DataACK, the Target Transfer Tag MUST contain a copy of the Target Transfer Tag and LUN provided with the SCSI Data-In PDU with the A bit set to 1.

In all other cases, the Target Transfer Tag field MUST be set to the reserved value of 0xffffffff.

## 11.16.6. BegRun

The DataSN, R2TSN, or StatSN of the first PDU whose retransmission is requested (Data/R2T and Status SNACK), or the next expected DataSN (DataACK SNACK).

BegRun 0 when used in conjunction with RunLength 0 means resend all unacknowledged Data-In, R2T or Response PDUs.

BegRun MUST be 0 for a R-Data SNACK.

## 11.16.7. RunLength

The number of PDUs whose retransmission is requested.

RunLength 0 signals that all Data-In, R2T, or Response PDUs carrying the numbers equal to or greater than BegRun have to be resent.

The RunLength MUST also be 0 for a DataACK SNACK in addition to R-Data SNACK.

## 11.17. Reject

Byte/	0	1	2	3
/				
0 1 2 3 4 5 6 7	0 1 2 3 4 5 6 7	0 1 2 3 4 5 6 7	0 1 2 3 4 5 6 7	
0	. . . 0x3f	1  Reserved	Reason	Reserved
4	TotalAHSLength	DataSegmentLength		
8/	Reserved			/
+/				/
16	0xffffffff			
20	Reserved			
24	StatSN			
28	ExpCmdSN			
32	MaxCmdSN			
36	DataSN/R2TSN or Reserved			
40	Reserved			
44	Reserved			
48	Header-Digest (Optional)			
xx/	Complete Header of Bad PDU			/
+/				/
yy/	Vendor specific data (if any)			/
/				/
zz	Data-Digest (Optional)			

Reject is used to indicate an iSCSI error condition (protocol, unsupported option, etc.).

#### 11.17.1. Reason

The reject Reason is coded as follows:

Code (hex)	Explanation	Can the original PDU be re-sent?
0x01	Reserved	no
0x02	Data (payload) Digest Error	yes (Note 1)
0x03	SNACK Reject	yes
0x04	Protocol Error (e.g., SNACK request for a status that was already acknowledged)	no
0x05	Command not supported	no
0x06	Immediate Command Reject - too many immediate commands	yes
0x07	Task in progress	no
0x08	Invalid Data ACK	no
0x09	Invalid PDU field	no (Note 2)
0x0a	Long Operation Reject - Can't generate Target Transfer Tag - out of resources	yes
0x0c	Waiting for Logout	no

Note 1: For iSCSI, Data-Out PDU retransmission is only done if the target requests retransmission with a recovery R2T. However, if this is the data digest error on immediate data, the initiator may choose to retransmit the whole PDU including the immediate data.

Note 2: A target should use this reason code for all invalid values of PDU fields that are meant to describe a task, a response, or a data transfer. Some examples are invalid TTT/ITT, buffer offset, LUN qualifying a TTT, and an invalid sequence number in a SNACK.

Note 3: Reason code 0x0b ("Negotiation reset") defined in [RFC3720] is deprecated and MUST NOT be used by implementations. An implementation receiving reason code 0x0b MUST treat it as a negotiation failure that terminates the Login Phase and the TCP connection, as specified in Section 7.12.

All other values for Reason are reserved.

In all the cases in which a pre-instantiated SCSI task is terminated because of the reject, the target MUST issue a proper SCSI command response with CHECK CONDITION as described in Section 11.4.3. In these cases in which a status for the SCSI task was already sent before the reject, no additional status is required. If the error is detected while data from the initiator is still expected (i.e., the command PDU did not contain all the data and the target has not received a Data-out PDU with the Final bit set to 1 for the unsolicited data, if any, and all outstanding R2Ts, if any), the target MUST wait until it receives the last expected Data-out PDUs with the F bit set to 1 before sending the Response PDU.

For additional usage semantics of Reject PDU, see Section 6.3 - "Usage Of Reject PDU in Recovery".

#### 11.17.2. DataSN/R2TSN

This field is only valid if the rejected PDU is a Data/R2T SNACK and the Reject reason code is "Protocol error" (see SNACK). The DataSN/R2TSN is the next Data/R2T sequence number that the target would send for the task, if any.

#### 11.17.3. StatSN, ExpCmdSN and MaxCmdSN

These fields carry their usual values and are not related to the rejected command. StatSN is advanced after a Reject.

## 11.17.4. Complete Header of Bad PDU

The target returns the header (not including digest) of the PDU in error as the data of the response.

## 11.18. NOP-Out

Byte/ /	0	1	2	3
	0 1 2 3 4 5 6 7	0 1 2 3 4 5 6 7	0 1 2 3 4 5 6 7	0 1 2 3 4 5 6 7
0	.   I   0x00	1   Reserved		
4	TotalAHSLength	DataSegmentLength		
8	LUN or Reserved			
12				
16	Initiator Task Tag or 0xffffffff			
20	Target Transfer Tag or 0xffffffff			
24	CmdSN			
28	ExpStatSN			
32	Reserved			/
+	/			/
48	Header-Digest (Optional)			
+	/ DataSegment - Ping Data (optional)			/
+	/			/
	Data-Digest (Optional)			
+				

A NOP-Out may be used by an initiator as a "ping request" to verify that a connection/session is still active and all its

components are operational. The NOP-In response is the "ping echo".

A NOP-Out is also sent by an initiator in response to a NOP-In.

A NOP-Out may also be used to confirm a changed ExpStatSN if another PDU will not be available for a long time.

Upon receipt of a NOP-In with the Target Transfer Tag set to a valid value (not the reserved 0xffffffff), the initiator MUST respond with a NOP-Out. In this case, the NOP-Out Target Transfer Tag MUST contain a copy of the NOP-In Target Transfer Tag.

#### 11.18.1. Initiator Task Tag

The NOP-Out MUST have the Initiator Task Tag set to a valid value only if a response in the form of NOP-In is requested (i.e., the NOP-Out is used as a ping request). Otherwise, the Initiator Task Tag MUST be set to 0xffffffff.

When a target receives the NOP-Out with a valid Initiator Task Tag, it MUST respond with a Nop-In Response (see Login and Full Feature Phase Negotiation).

If the Initiator Task Tag contains 0xffffffff, the I bit MUST be set to 1 and the CmdSN is not advanced after this PDU is sent.

#### 11.18.2. Target Transfer Tag

A target assigned identifier for the operation.

The NOP-Out MUST only have the Target Transfer Tag set if it is issued in response to a NOP-In with a valid Target Transfer Tag. In this case, it copies the Target Transfer Tag from the NOP-In PDU. Otherwise, the Target Transfer Tag MUST be set to 0xffffffff.

When the Target Transfer Tag is set to a value other than 0xffffffff, the LUN field MUST also be copied from the NOP-In.

## 11.18.3. Ping Data

Ping data are reflected in the NOP-In Response. The length of the reflected data are limited to MaxRecvDataSegmentLength. The length of ping data are indicated by the DataSegmentLength. 0 is a valid value for the DataSegmentLength and indicates the absence of ping data.



## 11.19. NOP-In

Byte/ /	0	1	2	3
	0 1 2 3 4 5 6 7	0 1 2 3 4 5 6 7	0 1 2 3 4 5 6 7	0 1 2 3 4 5 6 7
0	. .   0x20	1   Reserved		
4	TotalAHSLength	DataSegmentLength		
8	LUN or Reserved			
12				
16	Initiator Task Tag or 0xffffffff			
20	Target Transfer Tag or 0xffffffff			
24	StatSN			
28	ExpCmdSN			
32	MaxCmdSN			
36	Reserved			/
+	/			/
48	Header-Digest (Optional)			
+	/ DataSegment - Return Ping Data			/
+	/			/
	Data-Digest (Optional)			

NOP-In is either sent by a target as a response to a NOP-Out, as a "ping" to an initiator, or as a means to carry a changed ExpCmdSN and/or MaxCmdSN if another PDU will not be available for a long time (as determined by the target).

When a target receives the NOP-Out with a valid Initiator Task Tag (not the reserved value 0xffffffff), it MUST respond with a NOP-In with the same Initiator Task Tag that was provided in the NOP-Out request. It MUST also duplicate up to the first MaxRecvDataSegmentLength bytes of the initiator provided Ping Data. For such a response, the Target Transfer Tag MUST be 0xffffffff.

Otherwise, when a target sends a NOP-In that is not a response to a Nop-Out received from the initiator, the Initiator Task Tag MUST be set to 0xffffffff and the Data Segment MUST NOT contain any data (DataSegmentLength MUST be 0).

#### 11.19.1. Target Transfer Tag

If the target is responding to a NOP-Out, this is set to the reserved value 0xffffffff.

If the target is sending a NOP-In as a Ping (intending to receive a corresponding NOP-Out), this field is set to a valid value (not the reserved 0xffffffff).

If the target is initiating a NOP-In without wanting to receive a corresponding NOP-Out, this field MUST hold the reserved value of 0xffffffff.

#### 11.19.2. StatSN

The StatSN field will always contain the next StatSN. However, when the Initiator Task Tag is set to 0xffffffff StatSN for the connection is not advanced after this PDU is sent.

#### 11.19.3. LUN

A LUN MUST be set to a correct value when the Target Transfer Tag is valid (not the reserved value 0xffffffff).

## 12. iSCSI Security Text Keys and Authentication Methods

Only the following keys are used during the SecurityNegotiation stage of the Login Phase:

SessionType

InitiatorName

TargetName

TargetAddress

InitiatorAlias

TargetAlias

TargetPortalGroupTag

AuthMethod and the keys used by the authentication methods specified under Section 12.1 along with all of their associated keys as well as Vendor Specific Authentication Methods.

Other keys MUST NOT be used.

SessionType, InitiatorName, TargetName, InitiatorAlias, TargetAlias, and TargetPortalGroupTag are described in Section 13 as they can be used also in the OperationalNegotiation stage.

All security keys have connection-wide applicability.

## 12.1. AuthMethod

Use: During Login - Security Negotiation

Senders: Initiator and Target

Scope: connection

AuthMethod = <list-of-values>

The main item of security negotiation is the authentication method (AuthMethod).

The authentication methods that can be used (appear in the list-of-values) are either those listed in the following table or are vendor-unique methods:

Name	Description
KRB5	Kerberos V5 - defined in [RFC4120]
SRP	Secure Remote Password defined in [RFC2945]
CHAP	Challenge Handshake Authentication Protocol defined in [RFC1994]
None	No authentication

The AuthMethod selection is followed by an "authentication exchange" specific to the authentication method selected.

The authentication method proposal may be made by either the initiator or the target. However the initiator **MUST** make the first step specific to the selected authentication method as soon as it is selected. It follows that if the target makes the authentication method proposal the initiator sends the first key(s) of the exchange together with its authentication method selection.

The authentication exchange authenticates the initiator to the target, and optionally, the target to the initiator. Authentication is **OPTIONAL** to use but **MUST** be supported by the target and initiator.

The initiator and target **MUST** implement CHAP. All other authentication methods are **OPTIONAL**.

Private or public extension algorithms **MAY** also be negotiated for authentication methods. Whenever a private or public extension

algorithm is part of the default offer (the offer made in absence of explicit administrative action) the implementer MUST ensure that CHAP is listed as an alternative in the default offer and "None" is not part of the default offer.

Extension authentication methods MUST be named using one of the following two formats:

- i) Z-reversed.vendor.dns\_name.do\_something=
- ii) Z<#><IANA-registered-string>=

Authentication methods named using the Z- format are used as private extensions. Authentication methods named using the Z# format are used as public extensions that must be registered with IANA and MUST be described by a standards track RFC, an experimental RFC, or an informational RFC.

For all of the public or private extension authentication methods, the method specific keys MUST conform to the format specified in Section 5.1 - "Text Format" for standard-label.

To identify the vendor for private extension authentication methods, we suggest you use the reversed DNS-name as a prefix to the proper digest names.

The part of digest-name following Z- and Z# MUST conform to the format for standard-label specified in Section 5.1 - "Text Format".

Support for public or private extension authentication methods is OPTIONAL.

The following subsections define the specific exchanges for each of the standardized authentication methods. As mentioned earlier the first step is always done by the initiator.

#### 12.1.1.1. Kerberos

For KRB5 (Kerberos V5) [RFC4120] and [RFC1964], the initiator MUST use:

where KRB\_AP\_REQ is the client message as defined in [RFC4120].

The default principal name assumed by an iSCSI initiator or target (prior to any administrative configuration action) MUST be the iSCSI Initiator Name or iSCSI Target Name respectively, prefixed by the string "iscsi/".

If the initiator authentication fails, the target MUST respond with a Login reject with "Authentication Failure" status. Otherwise, if the initiator has selected the mutual authentication option (by setting MUTUAL-REQUIRED in the ap-options field of the KRB\_AP\_REQ), the target MUST reply with:

KRB\_AP\_REP=<KRB\_AP\_REP>

where KRB\_AP\_REP is the server's response message as defined in [RFC4120].

If mutual authentication was selected and target authentication fails, the initiator MUST close the connection.

KRB\_AP\_REQ and KRB\_AP\_REP are binary-values and their binary length (not the length of the character string that represents them in encoded form) MUST NOT exceed 65536 bytes.

#### 12.1.2. Secure Remote Password (SRP)

For SRP [RFC2945], the initiator MUST use:

SRP\_U=<U> TargetAuth=Yes /\* or TargetAuth=No \*/

The target MUST answer with a Login reject with the "Authorization Failure" status or reply with:

SRP\_GROUP=<G1,G2...> SRP\_s=<s>

Where G1,G2... are proposed groups, in order of preference.

The initiator MUST either close the connection or continue with:

SRP\_A=<A> SRP\_GROUP=<G>

Where G is one of G1,G2... that were proposed by the target.

The target MUST answer with a Login reject with the "Authentication Failure" status or reply with:

SRP\_B=<B>

The initiator MUST close the connection or continue with:

SRP\_M=<M>

If the initiator authentication fails, the target MUST answer with a Login reject with "Authentication Failure" status. Otherwise, if the initiator sent TargetAuth=Yes in the first message (requiring target authentication), the target MUST reply with:

SRP\_HM=<H(A | M | K)>

If the target authentication fails, the initiator MUST close the connection.

Where U, s, A, B, M, and H(A | M | K) are defined in [RFC2945] (using the SHA1 hash function, such as SRP-SHA1) and G,Gn (Gn stands for G1,G2...) are identifiers of SRP groups specified in [RFC3723]. G, Gn, and U are text strings, s,A,B,M, and H(A | M | K) are binary-values. The length of s,A,B,M and H(A | M | K) in binary form (not the length of the character string that represents them in encoded form) MUST NOT exceed 1024 bytes.

For the SRP\_GROUP, all the groups specified in [RFC3723] up to 1536 bits (i.e., SRP-768, SRP-1024, SRP-1280, SRP-1536) must be supported by initiators and targets. To guarantee interoperability, targets MUST always offer "SRP-1536" as one of the proposed groups.

#### 12.1.3. Challenge Handshake Authentication Protocol (CHAP)

For CHAP [RFC1994], the initiator MUST use:

CHAP\_A=<A1,A2...>

Where A1,A2... are proposed algorithms, in order of preference.

The target MUST answer with a Login reject with the "Authentication Failure" status or reply with:

CHAP\_A=<A> CHAP\_I=<I> CHAP\_C=<C>

Where A is one of A1,A2... that were proposed by the initiator.

The initiator MUST continue with:

CHAP\_N=<N> CHAP\_R=<R>

or, if it requires target authentication, with:

CHAP\_N=<N> CHAP\_R=<R> CHAP\_I=<I> CHAP\_C=<C>

If the initiator authentication fails, the target MUST answer with a Login reject with "Authentication Failure" status. Otherwise, if the initiator required target authentication, the target MUST either answer with a Login reject with "Authentication Failure" or reply with:

CHAP\_N=<N> CHAP\_R=<R>

If target authentication fails, the initiator MUST close the connection.

Where N, (A,A1,A2), I, C, and R are (correspondingly) the Name, Algorithm, Identifier, Challenge, and Response as defined in [RFC1994], N is a text string, A,A1,A2, and I are numbers, and C and R are binary-values and their binary length (not the length of the character string that represents them in encoded form) MUST NOT exceed 1024 bytes.

For the Algorithm, as stated in [RFC1994], one value is required to be implemented:

5 (CHAP with MD5)



To guarantee interoperability, initiators MUST always offer it as one of the proposed algorithms.

## 13. Login/Text Operational Text Keys

Some session specific parameters MUST only be carried on the leading connection and cannot be changed after the leading connection login (e.g., MaxConnections, the maximum number of connections). This holds for a single connection session with regard to connection restart. The keys that fall into this category have the use: LO (Leading Only).

Keys that can only be used during login have the use: IO (initialize only), while those that can be used in both the Login Phase and Full Feature Phase have the use: ALL.

Keys that can only be used during Full Feature Phase use FFPO (Full Feature Phase only).

Keys marked as Any-Stage may also appear in the SecurityNegotiation stage while all other keys described in this chapter are operational keys.

Keys that do not require an answer are marked as Declarative.

Key scope is indicated as session-wide (SW) or connection-only (CO).

Result function, wherever mentioned, states the function that can be applied to check the validity of the responder selection. Minimum means that the selected value cannot exceed the offered value. Maximum means that the selected value cannot be lower than the offered value. AND means that the selected value must be a possible result of a Boolean "and" function with an arbitrary Boolean value (e.g., if the offered value is No the selected value must be No). OR means that the selected value must be a possible result of a Boolean "or" function with an arbitrary Boolean value (e.g., if the offered value is Yes the selected value must be Yes).

## 13.1. HeaderDigest and DataDigest

Use: IO  
Senders: Initiator and Target  
Scope: CO

HeaderDigest = <list-of-values>

DataDigest = <list-of-values>

Default is None for both HeaderDigest and DataDigest.

Digests enable the checking of end-to-end, non-cryptographic data integrity beyond the integrity checks provided by the link layers and the covering of the whole communication path including all elements that may change the network level PDUs such as routers, switches, and proxies.

The following table lists cyclic integrity checksums that can be negotiated for the digests and that MUST be implemented by every iSCSI initiator and target. These digest options only have error detection significance.

Name	Description	Generator
CRC32C	32 bit CRC	0x11edc6f41
None	no digest	

The generator polynomial for this digest is given in hex-notation (e.g., 0x3b stands for 0011 1011 and the polynomial is  $x^{**5}+X^{**4}+x^{**3}+x+1$ ).

When the Initiator and Target agree on a digest, this digest MUST be used for every PDU in Full Feature Phase.

Padding bytes, when present in a segment covered by a CRC, SHOULD be set to 0 and are included in the CRC.

The CRC MUST be calculated by a method that produces the same results as the following process:

- The PDU bits are considered as the coefficients of a polynomial  $M(x)$  of degree  $n-1$ ; bit 7 of the lowest numbered byte is considered the most significant bit ( $x^{n-1}$ ),

followed by bit 6 of the lowest numbered byte through bit 0 of the highest numbered byte ( $x^0$ ).

- The most significant 32 bits are complemented.
- The polynomial is multiplied by  $x^{32}$  then divided by  $G(x)$ . The generator polynomial produces a remainder  $R(x)$  of degree  $\leq 31$ .
- The coefficients of  $R(x)$  are considered a 32 bit sequence.
- The bit sequence is complemented and the result is the CRC.
- The CRC bits are mapped into the digest word. The  $x^{31}$  coefficient in bit 7 of the lowest numbered byte of the digest continuing through to the byte up to the  $x^{24}$  coefficient in bit 0 of the lowest numbered byte, continuing with the  $x^{23}$  coefficient in bit 7 of next byte through  $x^0$  in bit 0 of the highest numbered byte.
- Computing the CRC over any segment (data or header) extended to include the CRC built using the generator  $0x11edc6f41$  will always get the value  $0x1c2d19ed$  as its final remainder ( $R(x)$ ). This value is given here in its polynomial form (i.e., not mapped as the digest word).

For a discussion about selection criteria for the CRC, see [RFC3385]. For a detailed analysis of the iSCSI polynomial, see [Castagnoli93].

Private or public extension algorithms MAY also be negotiated for digests. Whenever a private or public digest extension algorithm is part of the default offer (the offer made in absence of explicit administrative action) the implementer MUST ensure that CRC32C is listed as an alternative in the default offer and "None" is not part of the default offer.

Extension digest algorithms MUST be named using one of the following two formats:

- iii) Y-reversed.vendor.dns\_name.do\_something=
- iv) Y<#><IANA-registered-string>=

Digests named using the Y- format are used for private purposes (unregistered). Digests named using the Y# format (public extension) must be registered with IANA and MUST be described by a standards track RFC, an experimental RFC, or an informational RFC.

For private extension digests, to identify the vendor, we suggest you use the reversed DNS-name as a prefix to the proper digest names.

The part of digest-name following Y- and Y# MUST conform to the format for standard-label specified in Section 6.1.

Support for public or private extension digests is OPTIONAL.

### 13.2. MaxConnections

Use: LO

Senders: Initiator and Target

Scope: SW

Irrelevant when: SessionType=Discovery

MaxConnections=<numerical-value-from-1-to-65535>

Default is 1.

Result function is Minimum.

Initiator and target negotiate the maximum number of connections requested/acceptable.

### 13.3. SendTargets

Use: FFPO

Senders: Initiator

Scope: SW

For a complete description, see Appendix D. - "SendTargets Operation".

## 13.4. TargetName

Use: IO by initiator, FFPO by target - only as response to a  
SendTargets, Declarative, Any-Stage  
Senders: Initiator and Target  
Scope: SW

TargetName=<iSCSI-name-value>

Examples:

TargetName=iqn.1993-11.com.disk-vendor:diskarrays.sn.45678

TargetName=eui.020000023B040506

TargetName=naa.62004567BA64678D0123456789ABCDEF

The initiator of the TCP connection MUST provide this key to the remote endpoint in the first login request if the initiator is not establishing a discovery session. The iSCSI Target Name specifies the worldwide unique name of the target.

The TargetName key may also be returned by the "SendTargets" text request (which is its only use when issued by a target).

TargetName MUST NOT be redeclared within the login phase.

## 13.5. InitiatorName

Use: IO, Declarative, Any-Stage  
Senders: Initiator  
Scope: SW

InitiatorName=<iSCSI-name-value>

Examples:

InitiatorName=iqn.1992-04.com.os-vendor.plan9:cdrom.12345

InitiatorName=iqn.2001-02.com.ssp.users:customer235.host90

iSCSI (Consolidated)  
InitiatorName=naa.52004567BA64678D

3/11/11

The initiator of the TCP connection MUST provide this key to the remote endpoint at the first Login of the Login Phase for every connection. The InitiatorName key enables the initiator to identify itself to the remote endpoint.

InitiatorName MUST NOT be redeclared within the login phase.

#### 13.6. TargetAlias

Use: ALL, Declarative, Any-Stage  
Senders: Target  
Scope: SW

TargetAlias=<iSCSI-local-name-value>

Examples:

TargetAlias=Bob-s Disk

TargetAlias=Database Server 1 Log Disk

TargetAlias=Web Server 3 Disk 20

If a target has been configured with a human-readable name or description, this name SHOULD be communicated to the initiator during a Login Response PDU if SessionType=Normal (see 13.21). This string is not used as an identifier, nor is it meant to be used for authentication or authorization decisions. It can be displayed by the initiator's user interface in a list of targets to which it is connected.

#### 13.7. InitiatorAlias

Use: ALL, Declarative, Any-Stage  
Senders: Initiator  
Scope: SW

InitiatorAlias=<iSCSI-local-name-value>

## Examples:

```
InitiatorAlias=Web Server 4
```

```
InitiatorAlias=spyalley.nsa.gov
```

```
InitiatorAlias=Exchange Server
```

If an initiator has been configured with a human-readable name or description, it SHOULD be communicated to the target during a Login Request PDU. If not, the host name can be used instead. This string is not used as an identifier, nor is meant to be used for authentication or authorization decisions. It can be displayed by the target's user interface in a list of initiators to which it is connected.

## 13.8. TargetAddress

Use: ALL, Declarative, Any-Stage

Senders: Target

Scope: SW

```
TargetAddress=domainname[:port][,portal-group-tag]
```

The domainname can be specified as either a DNS host name, a dotted-decimal IPv4 address, or a bracketed IPv6 address as specified in [RFC3986].

If the TCP port is not specified, it is assumed to be the IANA-assigned default port for iSCSI (see Section 13 -"IANA Considerations").

If the TargetAddress is returned as the result of a redirect status in a login response, the comma and portal group tag MUST be omitted.

If the TargetAddress is returned within a SendTargets response, the portal group tag MUST be included.

## Examples:



TargetAddress=10.0.0.1:5003,1

TargetAddress=[1080:0:0:0:8:800:200C:417A],65

TargetAddress=[1080::8:800:200C:417A]:5003,1

TargetAddress=computingcenter.example.com,23

Use of the portal-group-tag is described in Appendix D. -  
"SendTargets Operation". The formats for the port and portal-  
group-tag are the same as the one specified in  
TargetPortalGroupTag.

### 13.9. TargetPortalGroupTag

Use: IO by target, Declarative, Any-Stage

Senders: Target

Scope: SW

TargetPortalGroupTag=<16-bit-binary-value>

Examples:

TargetPortalGroupTag=1

The target portal group tag is a 16-bit binary-value that uniquely identifies a portal group within an iSCSI target node. This key carries the value of the tag of the portal group that is servicing the Login request. The iSCSI target returns this key to the initiator in the Login Response PDU to the first Login Request PDU that has the C bit set to 0 when TargetName is given by the initiator.

[SAM2] and [SAM3] specifications note in their informative text that TPGT value should be non-zero, note that it is incorrect. A zero value is allowed as a legal value for TPGT. This discrepancy currently stands corrected in [SAM4].

For the complete usage expectations of this key see Section 6.3.

## 13.10. InitialR2T

Use: LO  
Senders: Initiator and Target  
Scope: SW  
Irrelevant when: SessionType=Discovery

InitialR2T=<boolean-value>

Examples:

I->InitialR2T=No

T->InitialR2T=No

Default is Yes.  
Result function is OR.

The InitialR2T key is used to turn off the default use of R2T for unidirectional and the output part of bidirectional commands, thus allowing an initiator to start sending data to a target as if it has received an initial R2T with Buffer Offset=Immediate Data Length and Desired Data Transfer Length=(min(FirstBurstLength, Expected Data Transfer Length) - Received Immediate Data Length).

The default action is that R2T is required, unless both the initiator and the target send this key-pair attribute specifying InitialR2T=No. Only the first outgoing data burst (immediate data and/or separate PDUs) can be sent unsolicited (i.e., not requiring an explicit R2T).

## 13.11. ImmediateData

Use: LO  
Senders: Initiator and Target  
Scope: SW  
Irrelevant when: SessionType=Discovery

ImmediateData=<boolean-value>

Default is Yes.

Result function is AND.

The initiator and target negotiate support for immediate data. To turn immediate data off, the initiator or target must state its desire to do so. ImmediateData can be turned on if both the initiator and target have ImmediateData=Yes.

If ImmediateData is set to Yes and InitialR2T is set to Yes (default), then only immediate data are accepted in the first burst.

If ImmediateData is set to No and InitialR2T is set to Yes, then the initiator MUST NOT send unsolicited data and the target MUST reject unsolicited data with the corresponding response code.

If ImmediateData is set to No and InitialR2T is set to No, then the initiator MUST NOT send unsolicited immediate data, but MAY send one unsolicited burst of Data-OUT PDUs.

If ImmediateData is set to Yes and InitialR2T is set to No, then the initiator MAY send unsolicited immediate data and/or one unsolicited burst of Data-OUT PDUs.

The following table is a summary of unsolicited data options:

InitialR2T	ImmediateData	Unsolicited Data Out PDUs	Immediate Data
No	No	Yes	No
No	Yes	Yes	Yes
Yes	No	No	No
Yes	Yes	No	Yes

### 13.12. MaxRecvDataSegmentLength

Use: ALL, Declarative

Senders: Initiator and Target  
Scope: CO

MaxRecvDataSegmentLength=<numerical-value-512-to- $(2^{24}-1)$ >

Default is 8192 bytes.

The initiator or target declares the maximum data segment length in bytes it can receive in an iSCSI PDU.

The transmitter (initiator or target) is required to send PDUs with a data segment that does not exceed MaxRecvDataSegmentLength of the receiver.

A target receiver is additionally limited by MaxBurstLength for solicited data and FirstBurstLength for unsolicited data. An initiator MUST NOT send solicited PDUs exceeding MaxBurstLength nor unsolicited PDUs exceeding FirstBurstLength (or FirstBurstLength-Immediate Data Length if immediate data were sent).

#### 13.13. MaxBurstLength

Use: LO  
Senders: Initiator and Target  
Scope: SW  
Irrelevant when: SessionType=Discovery

MaxBurstLength=<numerical-value-512-to- $(2^{24}-1)$ >

Default is 262144 (256 Kbytes).  
Result function is Minimum.

The initiator and target negotiate maximum SCSI data payload in bytes in a Data-In or a solicited Data-Out iSCSI sequence. A sequence consists of one or more consecutive Data-In or Data-Out PDUs that end with a Data-In or Data-Out PDU with the F bit set to one.

#### 13.14. FirstBurstLength

Use: LO

Senders: Initiator and Target

Scope: SW

Irrelevant when: SessionType=Discovery

Irrelevant when: ( InitialR2T=Yes and ImmediateData=No )

FirstBurstLength=<numerical-value-512-to-(2\*\*24-1)>

Default is 65536 (64 Kbytes).

Result function is Minimum.

The initiator and target negotiate the maximum amount in bytes of unsolicited data an iSCSI initiator may send to the target during the execution of a single SCSI command. This covers the immediate data (if any) and the sequence of unsolicited Data-Out PDUs (if any) that follow the command.

FirstBurstLength MUST NOT exceed MaxBurstLength.

#### 13.15. DefaultTime2Wait

Use: LO

Senders: Initiator and Target

Scope: SW

DefaultTime2Wait=<numerical-value-0-to-3600>

Default is 2.

Result function is Maximum.

The initiator and target negotiate the minimum time, in seconds, to wait before attempting an explicit/implicit logout or an active task reassignment after an unexpected connection termination or a connection reset.

A value of 0 indicates that logout or active task reassignment can be attempted immediately.

#### 13.16. DefaultTime2Retain

Use: LO

Senders: Initiator and Target

Scope: SW

DefaultTime2Retain=<numerical-value-0-to-3600>

Default is 20.

Result function is Minimum.

The initiator and target negotiate the maximum time, in seconds after an initial wait (Time2Wait), before which an active task reassignment is still possible after an unexpected connection termination or a connection reset.

This value is also the session state timeout if the connection in question is the last LOGGED\_IN connection in the session.

A value of 0 indicates that connection/task state is immediately discarded by the target.

#### 13.17. MaxOutstandingR2T

Use: LO

Senders: Initiator and Target

Scope: SW

MaxOutstandingR2T=<numerical-value-from-1-to-65535>

Irrelevant when: SessionType=Discovery

Default is 1.

Result function is Minimum.

Initiator and target negotiate the maximum number of outstanding R2Ts per task, excluding any implied initial R2T that might be part of that task. An R2T is considered outstanding until the last data PDU (with the F bit set to 1) is transferred, or a sequence reception timeout (Section 7.1.4.1) is encountered for that data sequence.

#### 13.18. DataPDUInOrder

Use: LO

Senders: Initiator and Target

Scope: SW

Irrelevant when: SessionType=Discovery

DataPDUInOrder=<boolean-value>

Default is Yes.

Result function is OR.

No is used by iSCSI to indicate that the data PDUs within sequences can be in any order. Yes is used to indicate that data PDUs within sequences have to be at continuously increasing addresses and overlays are forbidden.

### 13.19. DataSequenceInOrder

Use: LO

Senders: Initiator and Target

Scope: SW

Irrelevant when: SessionType=Discovery

DataSequenceInOrder=<boolean-value>

Default is Yes.

Result function is OR.

A Data Sequence is a sequence of Data-In or Data-Out PDUs that end with a Data-In or Data-Out PDU with the F bit set to one. A Data-out sequence is sent either unsolicited or in response to an R2T. Sequences cover an offset-range.

If DataSequenceInOrder is set to No, Data PDU sequences may be transferred in any order.

If DataSequenceInOrder is set to Yes, Data Sequences MUST be transferred using continuously non-decreasing sequence offsets (R2T buffer offset for writes, or the smallest SCSI Data-In buffer offset within a read data sequence).

If DataSequenceInOrder is set to Yes, a target may retry at most the last R2T, and an initiator may at most request retransmission for the last read data sequence. For this reason, if ErrorRecoveryLevel is not 0 and DataSequenceInOrder is set to Yes then MaxOustandingR2T MUST be set to 1.

## 13.20. ErrorRecoveryLevel

Use: LO

Senders: Initiator and Target

Scope: SW

ErrorRecoveryLevel=<numerical-value-0-to-2>

Default is 0.

Result function is Minimum.

The initiator and target negotiate the recovery level supported.

Recovery levels represent a combination of recovery capabilities. Each recovery level includes all the capabilities of the lower recovery levels and adds some new ones to them.

In the description of recovery mechanisms, certain recovery classes are specified. Section 7.1.5 describes the mapping between the classes and the levels.

## 13.21. SessionType

Use: LO, Declarative, Any-Stage

Senders: Initiator

Scope: SW

SessionType= <Discovery|Normal>

Default is Normal.

The Initiator indicates the type of session it wants to create. The target can either accept it or reject it.

A Discovery session indicates to the Target that the only purpose of this Session is discovery. The only requests a target accepts in this type of session are a text request with a SendTargets key and a logout request with reason "close the session".

The Discovery session implies MaxConnections = 1 and overrides both the default and an explicit setting. As section 7.4.1



states, ErrorRecoveryLevel MUST be 0 (zero) for Discovery sessions.

Depending on the type of the session, a target may decide on resources to allocate and the security to enforce, etc. for the session. If the SessionType key is thus going to be offered as "Discovery", it SHOULD be offered in the initial Login request by the initiator.

#### 13.22. The Private or Public Extension Key Format

Use: ALL

Senders: Initiator and Target

Scope: specific key dependent

X-reversed.vendor.dns\_name.do\_something=

or

X<#><IANA-registered-string>=

Keys with this format are used for public or private extension purposes. These keys always start with X- if unregistered with IANA (private) or X# if registered with IANA (public).

For unregistered keys, to identify the vendor, we suggest you use the reversed DNS-name as a prefix to the key-proper.

The part of key-name following X- and X# MUST conform to the format for key-name specified in Section 5.1 -"Text Format".

For IANA registered keys the string following X# must be registered with IANA and the use of the key MUST be described by a standards track RFC, an experimental RFC, or an informational RFC.

Vendor specific keys MUST ONLY be used in normal sessions.

Support for public or private extension keys is OPTIONAL.

#### 13.23. Task Reporting

Use: LO

Senders: Initiator and Target

Scope: SW

Irrelevant when: SessionType=Discovery

TaskReporting=<list-of-values>

Default is RFC3720.

Result function is AND.

This key is used to negotiate the task completion reporting semantics from the SCSI target. The following table describes the semantics that an iSCSI target MUST support for respective negotiated key values. Whenever this key is negotiated, at least the RFC3720 and ResponseFence values MUST be offered as options by the negotiation originator.

Name	Description
RFC3720	RFC 3720-compliant semantics. Response fencing is not guaranteed and fast completion of multi-task aborting is not supported
ResponseFence	Response Fence (section 4.2.2.3.3) semantics MUST be supported in reporting task completions
FastAbort	Updated fast multi-task abort semantics defined in Section 4.2.3.4 MUST be supported. Support for Response Fence is implied -- i.e., (Section 4.2.2.3.3) semantics MUST be supported as well

When TaskReporting is not negotiated to FastAbort, the standard multi-task abort semantics in Section 4.2.3.3 MUST be used.

#### 13.24. iSCSIProtocolLevel Negotiation

The iSCSIProtocolLevel associated with this document is "1". As a responder or an originator in a negotiation of this key, an iSCSI implementation compliant to this document alone, without any future protocol extensions, MUST use this value as defined by the [iSCSI-SAM] document.

## 13.25. Obsoleted Keys

This document obsoletes the following keys defined in [RFC3720]: IFMarker, OFMarker, OFMarkInt, IFMarkInt. However, iSCSI implementations compliant to this document may still receive these obsoleted keys - i.e. in a responder role - in a text negotiation.

When IFMarker or OFMarker key is received, a compliant iSCSI implementation SHOULD respond with the constant "Reject" value. The implementation MAY alternatively respond with a "No" value. However, the implementation MUST NOT respond with a "NotUnderstood" value for either of these keys.

When IFMarkInt or OFMarkInt key is received, a compliant iSCSI implementation MUST respond with the constant "Reject" value. The implementation MUST NOT respond with a "NotUnderstood" value for either of these keys.

## 13.26. X#NodeArchitecture

## 13.26.1. Definition

Use: LO, Declarative  
Senders: Initiator and Target  
Scope: SW

X#NodeArchitecture=<list-of-values>

Default is none.

Examples:

X#NodeArchitecture=ExampleOS/v1234,ExampleInc\_SW\_Initiator/1.05a  
X#NodeArchitecture=ExampleInc\_HW\_Initiator/4010,Firmware/2.0.0.5  
X#NodeArchitecture=ExampleInc\_SW\_Initiator/2.1,CPU\_Arch/i686

This document does not define the structure or content of the list of values.

The initiator or target declares the details of its iSCSI node architecture to the remote endpoint. These details may include, but are not limited to, iSCSI vendor software, firmware, or

hardware versions, the OS version, or hardware architecture. This key may be declared on a Discovery session or a Normal session.

The length of the key value (total length of the list-of-values) MUST NOT be greater than 255 bytes.

X#NodeArchitecture MUST NOT be redeclared during the login phase.

#### 13.26.2. Implementation Requirements

Functional behavior of the iSCSI node (this includes the iSCSI protocol logic -- the SCSI, iSCSI, and TCP/IP protocols) MUST NOT depend on the presence, absence, or content of the X#NodeArchitecture key. The key MUST NOT be used by iSCSI nodes for interoperability, or exclusion of other nodes. To ensure proper use, key values SHOULD be set by the node itself, and there SHOULD NOT be provisions for the key values to contain user-defined text.

Nodes implementing this key MUST choose one of the following implementation options:

- only transmit the key,
- only log the key values received from other nodes, or
- both transmit and log the key values.

Each node choosing to implement transmission of the key values MUST be prepared to handle the response of iSCSI Nodes that do not understand the key.

Nodes that implement transmission and/or logging of the key values may also implement administrative mechanisms that disable and/or change the logging and key transmission detail (see Section 8.4 - "Security Considerations for the X#NodeArchitecture Key"). Thus, a valid behavior for this key may be that a node is completely silent (the node does not transmit any key value, and simply discards any key values it receives without issuing a NotUnderstood response).

## 14. IANA Considerations

The well-known TCP port number for iSCSI connections assigned by IANA is 3260 and this is the default iSCSI port. Implementations needing a system TCP port number may use port 860, the port assigned by IANA as the iSCSI system port; however in order to use port 860, it MUST be explicitly specified - implementations MUST NOT default to use of port 860, as 3260 is the only allowed default.

[RFC3720] instructs that three text key registries be set up, one for each of Extension keys, authentication methods, or digest keys - with the stipulation that the key prefix (X#, Y# or Z#) be not recorded. However, [RFC4850] indicates that the key prefix was recorded by IANA as part of the key name. Consequently, storm working group (which published this document) instructs IANA that:

- (i) It maintain a single text key registry for iSCSI keys, and,
- (ii) MUST always record the key prefix as part of the recorded string

This is being done with the intent to not have to change what IANA already did while publishing [RFC4850].

All the other IANA considerations stated in [RFC3720] and [RFC5048] remain unchanged.

This document obsoletes the SPKM1 and SPKM2 key values for the AuthMethod text key. Consequently, the SPKM\_ text key prefix MUST be treated as obsolete and be not reused.

## References

## Normative References

[EUI] "Guidelines for 64-bit Global Identifier (EUI-64)",  
<http://standards.ieee.org/regauth/oui/tutorials/EUI64.html>

[FC-FS] INCITS 373-2003, Fibre Channel Framing and Signaling Interface (FC-FS).

- iSCSI (Consolidated) 3/11/11
- [iSCSI-SAM] Knight, F., Chadalapaka, M., "Internet Small Computer Systems Interface (iSCSI) Update", draft-ietf-storm-iscsi-sam-01.txt (work in progress), April 2010
- [OUI] "IEEE OUI and Company\_Id Assignments",  
<http://standards.ieee.org/regauth/oui>
- [RFC791] INTERNET PROTOCOL, DARPA INTERNET PROGRAM PROTOCOL SPECIFICATION, September 1981.
- [RFC793] TRANSMISSION CONTROL PROTOCOL, DARPA INTERNET PROGRAM PROTOCOL SPECIFICATION, September 1981.
- [RFC1035] P. Mockapetris, DOMAIN NAMES - IMPLEMENTATION AND SPECIFICATION, November 1987.
- [RFC1122] Requirements for Internet Hosts-Communication Layer RFC1122, R. Braden (editor).
- [RFC1964] J. Linn, "The Kerberos Version 5 GSS-API Mechanism", June 1996.
- [RFC1982] Elz, R., Bush, R., "Serial Number Arithmetic", August 1996.
- [RFC1994] W. Simpson, "PPP Challenge Handshake Authentication Protocol (CHAP)", August 1996.
- [RFC2025] C. Adams, "The Simple Public-Key GSS-API Mechanism (SPKM)", October 1996.
- [RFC2045] N. Borenstein, N. Freed, "MIME (Multipurpose Internet Mail Extensions) Part One: Mechanisms for Specifying and Describing the Format of Internet Message Bodies", November 1996.
- [RFC2119] Bradner, S. "Key Words for use in RFCs to Indicate Requirement Levels", BCP 14, March 1997.
- [RFC4291] Hinden, R. and S. Deering, "IP Version 6 Addressing Architecture", February 2006.

- [RFC2404] C. Madson, R. Glenn, "The Use of HMAC-SHA-1-96 within ESP and AH", November 1998.
- [RFC2451] R. Pereira, R. Adams " The ESP CBC-Mode Cipher Algorithms".
- [RFC2945] Wu, T., "The SRP Authentication and Key Exchange System", September 2000.
- [RFC3454] Hoffman, P. and M. Blanchet, "Preparation of Internationalized Strings ("stringprep")", RFC 3454, December 2002.
- [RFC3566] Frankel, S. and H. Herbert, "The AES-XCBC-MAC-96 Algorithm and Its Use With IPsec", RFC 3566, September 2003.
- [RFC3629] Yergeau, F., "UTF-8, a Transformation Format of ISO 10646", RFC 3629, November 2003
- [RFC3686] Housley, R., "Using Advanced Encryption Standard (AES) Counter Mode with IPsec Encapsulating Security Payload (ESP)", RFC 3686, January 2004.
- [RFC3720] Satran, J., Meth, K., Sapuntzakis, C., Chadalapaka, M., and E. Zeidner, "Internet Small Computer Systems Interface (iSCSI)", RFC 3720, April 2004.
- [RFC3722] Bakke, M., "String Profile for Internet Small Computer Systems Interface (iSCSI) Names", RFC 3722, March 2004.
- [RFC3723] Aboba, B., Tseng, J., Walker, J., Rangan, V. and F. Travostino, "Securing Block Storage Protocols over IP", RFC 3723, March 2004.
- [RFC3980] Krueger, M., Chadalapaka, M., Elliott, R., "T11 Network Address Authority (NAA) Naming Format for iSCSI Node Names", RFC 3980, February 2005.

- [RFC3986] T. Berners-Lee, R. Fielding, L. Masinter "Uniform Resource Identifier (URI): Generic Syntax", January 2005.
- [RFC4120] Neuman, C., Yu, T., Hartman, S., Raeburn, K, "The Kerberos Network Authentication Service (V5)", RFC 4120, July 2005.
- [RFC4171] J. Tseng, K. Gibbons, F. Travostino, C. Du Laney, J. Souza, "Internet Storage Name Service (iSNS)", September 2005.
- [RFC4301] S. Kent, K.Seo, "Security Architecture for the Internet Protocol", December 2005.
- [RFC4303] Kent, S., "IP Encapsulating Security Payload (ESP)", RFC 4303, December 2005
- [RFC5996] C. Kaufman, P. Hoffman, Y. Nir, P. Eronen, "Internet Key Exchange Protocol Version 2 (IKEv2) ", RFC 5996, September 2010..
- [RFC5646] A. Phillips, M. Davis, "Tags for Identifying Languages", RFC 5646, September 2009.
- [RFC4850] Wysochanski, D., "Declarative Public Extension Key for Internet Small Computer Systems Interface (iSCSI) Node Architecture", RFC 4850, April 2007.
- [RFC5048] Chadalapaka, M., "Internet Small Computer Systems Interface (iSCSI) Corrections and Clarifications", RFC 5048, October 2007.
- [RFC5226] T. Narten, and H. Avestrand, "Guidelines for Writing an IANA Considerations Section in RFCs.", May 2008.
- [SAM2] T10/1157-D, SCSI Architecture Model - 2 (SAM-2).
- [SAM3] T10/1561-D, SCSI Architecture Model - 3 (SAM-3).
- [SAM4] T10/1683-D, SCSI Architecture Model - 4 (SAM-4).
- [SBC] NCITS.306-1998, SCSI-3 Block Commands (SBC).
- [SPC3] T10/1416-D, SCSI Primary Commands-3.



iSCSI (Consolidated) 3/11/11  
[UML] ISO/IEC 19501, Unified Modeling Language  
Specification Version 1.4.2.  
  
[UNICODE] Unicode Standard Annex #15, "Unicode Normalization  
Forms", <http://www.unicode.org/unicode/reports/tr15>

#### Informative References

- [RFC1737] K. Sollins, L. Masinter "Functional Requirements for  
Uniform Resource Names".
- [IB] InfiniBand<sup>tm</sup> Architecture Specification, Vol. 1,  
Rel.1.0.a, InfiniBand  
Trade Association(<http://www.infinibandta.org>).
- [RFC4173] P. Sarkar, D. Missimer, C. Sapuntzakis,  
"Bootstrapping Clients using the Internet Small Computer  
System Interface (iSCSI) Protocol", RFC 4173, September  
2005.
- [Castagnoli93] G. Castagnoli, S. Braeuer and M. Herrman  
"Optimization of Cyclic Redundancy-Check Codes with 24 and  
32 Parity Bits", IEEE Transact. on Communications, Vol. 41,  
No. 6, June 1993.
- [CRC] ISO 3309, High-Level Data Link Control (CRC 32).
- [RFC3347] Krueger, M., Haagens, R., Sapuntzakis, C. and M.  
Bakke, "Small Computer Systems Interface protocol over the  
Internet (iSCSI) Requirements and Design Considerations",  
RFC 3347, July 2002.
- [RFC3385] Sheinwald, D., Staran, J., Thaler, P. and V.  
Cavanna, "Internet Protocol Small Computer System Interface  
(iSCSI) Cyclic Redundancy Check (CRC)/Checksum  
Considerations", RFC 3385, September 2002.
- [RFC3721] Bakke, M., Hafner, J., Hufferd, J., Voruganti, K.  
and M. Krueger, "Internet Small Computer Systems Interface  
(iSCSI) Naming and Discovery", RFC 3721, March 2004

[RFC3783] M. Chadalapaka, R. Elliott, "Small Computer Systems Interface (SCSI) Command Ordering Considerations with iSCSI", RFC 3783, May 2004.

[RFC4297] Romanow, A., Mogul, J., Talpey, T., and S. Bailey, "Remote Direct Memory Access (RDMA) over IP Problem Statement", RFC 4297, October 2004

[RFC5046] Ko, M., Chadalapaka, M., Hufferd, J., Elzur, U., Shah, H., and P. Thaler, "Internet Small Computer System Interface (iSCSI) Extensions for Remote Direct Memory Access (RDMA)", RFC 5046, October 2007

[Schneier] B. Schneier, "Applied Cryptography: Protocols, Algorithms, and Source Code in C", 2nd edition, John Wiley & Sons, New York, NY, 1996.

[SAS] INCITS 376-2003, Serial Attached SCSI (SAS).

[SRP] INCITS 365-2002, SCSI RDMA Protocol (SRP).

## Appendix A. Examples

## Read Operation Example

Initiator Function	PDU Type	Target Function
Command request (read)	SCSI Command (READ)>>>	
		Prepare Data Transfer
Receive Data	<<< SCSI Data-in	Send Data
Receive Data	<<< SCSI Data-in	Send Data
Receive Data	<<< SCSI Data-in	Send Data
	<<< SCSI Response	Send Status and Sense
Command Complete		

## Write Operation Example

Initiator Function	PDU Type	Target Function
Command request (write)	SCSI Command (WRITE)>>>	Receive command and queue it
		Process old commands
	<<< R2T	Ready to process WRITE command
Send Data	SCSI Data-out >>>	Receive Data
	<<< R2T	Ready for data
	<<< R2T	Ready for data
Send Data	SCSI Data-out >>>	Receive Data
Send Data	SCSI Data-out >>>	Receive Data
	<<< SCSI Response	Send Status and Sense
Command Complete		

## R2TSN/DataSN Use Examples

Output (write) data DataSN/R2TSN Example

Initiator Function	PDU Type & Content	Target Function
Command request (write)	SCSI Command (WRITE)>>>	Receive command and queue it
		Process old commands
	<<< R2T R2TSN = 0	Ready for data
	<<< R2T R2TSN = 1	Ready for more data
Send Data for R2TSN 0	SCSI Data-out >>> DataSN = 0, F=0	Receive Data
Send Data for R2TSN 0	SCSI Data-out >>> DataSN = 1, F=1	Receive Data
Send Data for R2TSN 1	SCSI Data >>> DataSN = 0, F=1	Receive Data
	<<< SCSI Response ExpDataSN = 0	Send Status and Sense
Command Complete		

Input (read) data DataSN Example

Initiator Function	PDU Type	Target Function
Command request (read)	SCSI Command (READ)>>>	
		Prepare Data Transfer
Receive Data	<<< SCSI Data-in DataSN = 0, F=0	Send Data
Receive Data	<<< SCSI Data-in DataSN = 1, F=0	Send Data
Receive Data	<<< SCSI Data-in DataSN = 2, F=1	Send Data
	<<< SCSI Response ExpDataSN = 3	Send Status and Sense
Command Complete		

## Bidirectional DataSN Example

Initiator Function	PDU Type	Target Function
Command request (Read-Write)	SCSI Command >>> Read-Write	
		Process old commands
	<<< R2T R2TSN = 0	Ready to process WRITE command
* Receive Data	<<< SCSI Data-in DataSN = 0, F=0	Send Data
* Receive Data	<<< SCSI Data-in DataSN = 1, F=1	Send Data
* Send Data for R2TSN 0	SCSI Data-out >>> DataSN = 0, F=1	Receive Data
	<<< SCSI Response ExpDataSN = 2	Send Status and Sense
Command Complete		

\*) Send data and Receive Data may be transferred simultaneously as in an atomic Read-Old-Write-New or sequentially as in an atomic Read-Update-Write (in the latter case the R2T may follow the received data).

Unsolicited and immediate output (write) data with DataSN Example

iSCSI (Consolidated)		3/11/11
Initiator Function	PDU Type & Content	Target Function
Command request (write) + immediate data	SCSI Command (WRITE)>>> F=0	Receive command and data and queue it
Send Unsolicited Data	SCSI Write Data >>> DataSN = 0, F=1	Receive more Data
		Process old commands
	<<< R2T R2TSN = 0	Ready for more data
Send Data for R2TSN 0	SCSI Write Data >>> DataSN = 0, F=1	Receive Data
	<<< SCSI Response	Send Status and Sense
Command Complete		

#### CRC Examples

N.B. all Values are Hexadecimal

32 bytes of zeroes:

```

Byte:      0    1    2    3
0:         00 00 00 00
...
28:        00 00 00 00

CRC:       aa 36 91 8a

```

32 bytes of ones:

```

Byte:      0    1    2    3

```



```

                                iSCSI (Consolidated)
0:      ff ff ff ff
...
28:     ff ff ff ff

CRC:     43 ab a8 62

```

32 bytes of incrementing 00..1f:

```

Byte:      0      1      2      3

0:      00 01 02 03
...
28:     1c 1d 1e 1f

CRC:     4e 79 dd 46

```

32 bytes of decrementing 1f..00:

```

Byte:      0      1      2      3

0:      1f 1e 1d 1c
...
28:     03 02 01 00

CRC:     5c db 3f 11

```

An iSCSI - SCSI Read (10) Command PDU

```

Byte:      0      1      2      3

0:      01      c0      00      00
4:      00      00      00      00
8:      00      00      00      00
12:     00      00      00      00
16:     14      00      00      00
20:     00      00      04      00
24:     00      00      00      14
28:     00      00      00      18
32:     28      00      00      00
36:     00      00      00      00
40:     02      00      00      00

```

```
                                iSCSI (Consolidated)                3/11/11
44:          00 00 00 00

CRC:         56 3a 96 d9
```

#### Appendix B. Login Phase Examples

In the first example, the initiator and target authenticate each other via Kerberos:

```
I-> Login (CSG,NSG=0,1 T=1)
    InitiatorName=iqn.1999-07.com.os:hostid.77
    TargetName=iqn.1999-07.com.example:diskarray.sn.88
    AuthMethod=KRB5,SRP,None
```

```
T-> Login (CSG,NSG=0,0 T=0)
    AuthMethod=KRB5
```

```
I-> Login (CSG,NSG=0,1 T=1)
    KRB_AP_REQ=<krb_ap_req>
```

(krb\_ap\_req contains the Kerberos V5 ticket and authenticator with MUTUAL-REQUIRED set in the ap-options field)

If the authentication is successful, the target proceeds with:

```
T-> Login (CSG,NSG=0,1 T=1)
    KRB_AP_REP=<krb_ap_rep>
```

(krb\_ap\_rep is the Kerberos V5 mutual authentication reply)

If the authentication is successful, the initiator may proceed with:

```
I-> Login (CSG,NSG=1,0 T=0) FirstBurstLength=8192
T-> Login (CSG,NSG=1,0 T=0) FirstBurstLength=4096
MaxBurstLength=8192
I-> Login (CSG,NSG=1,0 T=0) MaxBurstLength=8192
    ... more iSCSI Operational Parameters

T-> Login (CSG,NSG=1,0 T=0)
    ... more iSCSI Operational Parameters
```

And at the end:

I-> Login (CSG,NSG=1,3 T=1)  
optional iSCSI parameters

T-> Login (CSG,NSG=1,3 T=1) "login accept"

If the initiator's authentication by the target is not successful, the target responds with:

T-> Login "login reject"

instead of the Login KRB\_AP\_REP message, and terminates the connection.

If the target's authentication by the initiator is not successful, the initiator terminates the connection (without responding to the Login KRB\_AP\_REP message).

In the next example only the initiator is authenticated by the target via Kerberos:

I-> Login (CSG,NSG=0,1 T=1)  
InitiatorName=iqn.1999-07.com.os:hostid.77  
TargetName=iqn.1999-07.com.example:diskarray.sn.88  
AuthMethod=SRP,KRB5,None

T-> Login-PR (CSG,NSG=0,0 T=0)  
AuthMethod=KRB5

I-> Login (CSG,NSG=0,1 T=1)  
KRB\_AP\_REQ=krb\_ap\_req

(MUTUAL-REQUIRED not set in the ap-options field of krb\_ap\_req)

If the authentication is successful, the target proceeds with:

T-> Login (CSG,NSG=0,1 T=1)

I-> Login (CSG,NSG=1,0 T=0)

```

                                iSCSI (Consolidated)
                                3/11/11
... iSCSI parameters

T-> Login (CSG,NSG=1,0 T=0)
... iSCSI parameters

. . .

T-> Login (CSG,NSG=1,3 T=1) "login accept"

```

In the next example, the initiator and target authenticate each other via SRP:

```

I-> Login (CSG,NSG=0,1 T=1)
    InitiatorName=iqn.1999-07.com.os:hostid.77
    TargetName=iqn.1999-07.com.example:diskarray.sn.88
    AuthMethod=KRB5,SRP,None

T-> Login-PR (CSG,NSG=0,0 T=0)
    AuthMethod=SRP

I-> Login (CSG,NSG=0,0 T=0)
    SRP_U=<user>
    TargetAuth=Yes

T-> Login (CSG,NSG=0,0 T=0)
    SRP_N=<N>
    SRP_g=<g>
    SRP_s=<s>

I-> Login (CSG,NSG=0,0 T=0)
    SRP_A=<A>

T-> Login (CSG,NSG=0,0 T=0)
    SRP_B=<B>

I-> Login (CSG,NSG=0,1 T=1)
    SRP_M=<M>

```

If the initiator authentication is successful, the target proceeds:

T-> Login (CSG,NSG=0,1 T=1)  
 SRP\_HM=<H(A | M | K)>

Where N, g, s, A, B, M, and H(A | M | K) are defined in [RFC2945].

If the target authentication is not successful, the initiator terminates the connection; otherwise, it proceeds.

I-> Login (CSG,NSG=1,0 T=0)  
 ... iSCSI parameters

T-> Login (CSG,NSG=1,0 T=0)  
 ... iSCSI parameters

And at the end:

I-> Login (CSG,NSG=1,3 T=1)  
 optional iSCSI parameters

T-> Login (CSG,NSG=1,3 T=1) "login accept"

If the initiator authentication is not successful, the target responds with:

T-> Login "login reject"

Instead of the T-> Login SRP\_HM=<H(A | M | K)> message and terminates the connection.

In the next example, only the initiator is authenticated by the target via SRP:

I-> Login (CSG,NSG=0,1 T=1)  
 InitiatorName=iqn.1999-07.com.os:hostid.77  
 TargetName=iqn.1999-07.com.example:diskarray.sn.88  
 AuthMethod=KRB5,SRP,None

T-> Login-PR (CSG,NSG=0,0 T=0)  
AuthMethod=SRP

I-> Login (CSG,NSG=0,0 T=0)  
SRP\_U=<user>  
TargetAuth=No

T-> Login (CSG,NSG=0,0 T=0)  
SRP\_N=<N>  
SRP\_g=<g>  
SRP\_s=<s>

I-> Login (CSG,NSG=0,0 T=0)  
SRP\_A=<A>

T-> Login (CSG,NSG=0,0 T=0)  
SRP\_B=<B>

I-> Login (CSG,NSG=0,1 T=1)  
SRP\_M=<M>

If the initiator authentication is successful, the target  
proceeds:

T-> Login (CSG,NSG=0,1 T=1)

I-> Login (CSG,NSG=1,0 T=0)  
... iSCSI parameters

T-> Login (CSG,NSG=1,0 T=0)  
... iSCSI parameters

And at the end:

I-> Login (CSG,NSG=1,3 T=1)

optional iSCSI parameters

T-> Login (CSG,NSG=1,3 T=1) "login accept"

In the next example the initiator and target authenticate each other via CHAP:

I-> Login (CSG,NSG=0,0 T=0)

InitiatorName=iqn.1999-07.com.os:hostid.77

TargetName=iqn.1999-07.com.example:diskarray.sn.88

AuthMethod=KRB5,CHAP,None

T-> Login-PR (CSG,NSG=0,0 T=0)

AuthMethod=CHAP

I-> Login (CSG,NSG=0,0 T=0)

CHAP\_A=<A1,A2>

T-> Login (CSG,NSG=0,0 T=0)

CHAP\_A=<A1>  
CHAP\_I=<I>  
CHAP\_C=<C>

I-> Login (CSG,NSG=0,1 T=1)

CHAP\_N=<N>

CHAP\_R=<R>

CHAP\_I=<I>

CHAP\_C=<C>

If the initiator authentication is successful, the target proceeds:

T-> Login (CSG,NSG=0,1 T=1)

CHAP\_N=<N>

CHAP\_R=<R>

If the target authentication is not successful, the initiator aborts the connection; otherwise, it proceeds.

I-> Login (CSG,NSG=1,0 T=0)

... iSCSI parameters

T-> Login (CSG,NSG=1,0 T=0)

... iSCSI parameters

And at the end:



I-> Login (CSG,NSG=1,3 T=1)

optional iSCSI parameters

T-> Login (CSG,NSG=1,3 T=1) "login accept"

If the initiator authentication is not successful, the target responds with:

T-> Login "login reject"

Instead of the Login CHAP\_R=<response> "proceed and change stage" message and terminates the connection.

In the next example, only the initiator is authenticated by the target via CHAP:

I-> Login (CSG,NSG=0,1 T=0)

InitiatorName=iqn.1999-07.com.os:hostid.77

TargetName=iqn.1999-07.com.example:diskarray.sn.88

AuthMethod=KRB5,CHAP,None

T-> Login-PR (CSG,NSG=0,0 T=0)

AuthMethod=CHAP

I-> Login (CSG,NSG=0,0 T=0)

CHAP\_A=<A1,A2>

T-> Login (CSG,NSG=0,0 T=0)

CHAP\_A=<A1>

CHAP\_I=<I>

CHAP\_C=<C>

I-> Login (CSG,NSG=0,1 T=1)

CHAP\_N=<N>

CHAP\_R=<R>

If the initiator authentication is successful, the target proceeds:

T-> Login (CSG,NSG=0,1 T=1)

I-> Login (CSG,NSG=1,0 T=0)

... iSCSI parameters

T-> Login (CSG,NSG=1,0 T=0)

... iSCSI parameters

And at the end:

I-> Login (CSG,NSG=1,3 T=1)

optional iSCSI parameters

T-> Login (CSG,NSG=1,3 T=1) "login accept"

In the next example, the initiator does not offer any security parameters. It therefore may offer iSCSI parameters on the Login PDU with the T bit set to 1, and the target may respond with a final Login Response PDU immediately:

I-> Login (CSG,NSG=1,3 T=1)

InitiatorName=iqn.1999-07.com.os:hostid.77

TargetName=iqn.1999-07.com.example:diskarray.sn.88

... iSCSI parameters

T-> Login (CSG,NSG=1,3 T=1) "login accept"

... iSCSI parameters

In the next example, the initiator does offer security parameters on the Login PDU, but the target does not choose any (i.e., chooses the "None" values):

I-> Login (CSG,NSG=0,1 T=1)

InitiatorName=iqn.1999-07.com.os:hostid.77

TargetName=iqn.1999-07.com.example:diskarray.sn.88

AuthMethod=KRB5,SRP,None

iSCSI (Consolidated) 3/11/11  
T-> Login-PR (CSG,NSG=0,1 T=1)  
AuthMethod=None

I-> Login (CSG,NSG=1,0 T=0)  
... iSCSI parameters

T-> Login (CSG,NSG=1,0 T=0)  
... iSCSI parameters

And at the end:

I-> Login (CSG,NSG=1,3 T=1)  
optional iSCSI parameters

T-> Login (CSG,NSG=1,3 T=1) "login accept"

#### Appendix C. SendTargets Operation

To reduce the amount of configuration required on an initiator, iSCSI provides the SendTargets text request. The initiator uses the SendTargets request to get a list of targets to which it may have access, as well as the list of addresses (IP address and TCP port) on which these targets may be accessed.

To make use of SendTargets, an initiator must first establish one of two types of sessions. If the initiator establishes the session using the key "SessionType=Discovery", the session is a discovery session, and a target name does not need to be specified. Otherwise, the session is a normal, operational session. The SendTargets command MUST only be sent during the Full Feature Phase of a normal or discovery session.

A system that contains targets MUST support discovery sessions on each of its iSCSI IP address-port pairs, and MUST support the SendTargets command on the discovery session. In a discovery session, a target MUST return all path information (IP address-port pairs and portal group tags) for the targets on the target network entity which the requesting initiator is authorized to access.

A target MUST support the SendTargets command on operational sessions; these will only return path information about the target to which the session is connected, and do not need to return information about other target names that may be defined in the responding system.

An initiator MAY make use of the SendTargets as it sees fit.

A SendTargets command consists of a single Text request PDU. This PDU contains exactly one text key and value. The text key MUST be SendTargets. The expected response depends upon the value, as well as whether the session is a discovery or operational session.

The value must be one of:

All

The initiator is requesting that information on all relevant targets known to the implementation be returned. This value MUST be supported on a discovery session, and MUST NOT be supported on an operational session.

<iSCSI-target-name>

If an iSCSI target name is specified, the session should respond with addresses for only the named target, if possible. This value MUST be supported on discovery sessions. A discovery session MUST be capable of returning addresses for those targets that would have been returned had value=All been designated.

<nothing>

The session should only respond with addresses for the target to which the session is logged in. This MUST be supported on operational sessions, and MUST NOT return targets other than the one to which the session is logged in.

The response to this command is a text response that contains a list of zero or more targets and, optionally, their addresses. Each target is returned as a target record. A target record begins with the TargetName text key, followed by a list of TargetAddress text keys, and bounded by the end of the text response or the next TargetName key, which begins a new record. No text keys other than TargetName and TargetAddress are permitted within a SendTargets response.

For the format of the TargetName, see Section 12.4 - "TargetName".

A discovery session MAY respond to a SendTargets request with its complete list of targets, or with a list of targets that is based on the name of the initiator logged in to the session.

A SendTargets response MUST NOT contain target names if there are no targets for the requesting initiator to access.

Each target record returned includes zero or more TargetAddress fields.

Each target record starts with one text key of the form:

TargetName=<target-name-goes-here>

Followed by zero or more address keys of the form:

TargetAddress=<hostname-or-ipaddress>[:<tcp-port>],<portal-group-tag>

The hostname-or-ipaddress contains a domain name, IPv4 address, or IPv6 address, as specified for the TargetAddress key.

A hostname-or-ipaddress duplicated in TargetAddress responses for a given node (the port is absent or equal) would probably indicate that multiple address families are in use at once (IPv6 and IPv4).

Each TargetAddress belongs to a portal group, identified by its numeric portal group tag (as in Section 12.9 - "TargetPortalGroupTag"). The iSCSI target name, together with this tag, constitutes the SCSI port identifier; the tag only needs to be unique within a given target's name list of addresses.

Multiple-connection sessions can span iSCSI addresses that belong to the same portal group.

Multiple-connection sessions cannot span iSCSI addresses that belong to different portal groups.

If a SendTargets response reports an iSCSI address for a target, it SHOULD also report all other addresses in its portal group in the same response.

A SendTargets text response can be longer than a single Text Response PDU, and makes use of the long text responses as specified.

After obtaining a list of targets from the discovery target session, an iSCSI initiator may initiate new sessions to log in to the discovered targets for full operation. The initiator MAY keep the discovery session open, and MAY send subsequent SendTargets commands to discover new targets.

Examples:

This example is the SendTargets response from a single target that has no other interface ports.

Initiator sends text request that contains:

```
SendTargets=All
```

Target sends a text response that contains:

```
TargetName=iqn.1993-11.com.example:diskarray.sn.8675309
```

All the target had to return in the simple case was the target name. It is assumed by the initiator that the IP address and TCP port for this target are the same as used on the current connection to the default iSCSI target.

The next example has two internal iSCSI targets, each accessible via two different ports with different IP addresses. The following is the text response:

```
TargetName=iqn.1993-11.com.example:diskarray.sn.8675309
```

```
TargetAddress=10.1.0.45:3000,1
```

```
TargetAddress=10.1.1.45:3000,2
```

```
TargetName=iqn.1993-11.com.example:diskarray.sn.1234567
```

```
TargetAddress=10.1.0.45:3000,1
```

```
TargetAddress=10.1.1.45:3000,2
```

Both targets share both addresses; the multiple addresses are likely used to provide multi-path support. The initiator may connect to either target name on either address. Each of the addresses has its own portal group tag; they do not support spanning multiple-connection sessions with each other. Keep in mind that the portal group tags for the two named targets are



independent of one another; portal group "1" on the first target is not necessarily the same as portal group "1" on the second target.

In the above example, a DNS host name or an IPv6 address could have been returned instead of an IPv4 address.

The next text response shows a target that supports spanning sessions across multiple addresses, and further illustrates the use of the portal group tags:

```
TargetName=iqn.1993-11.com.example:diskarray.sn.8675309
```

```
TargetAddress=10.1.0.45:3000,1
```

```
TargetAddress=10.1.1.46:3000,1
```

```
TargetAddress=10.1.0.47:3000,2
```

```
TargetAddress=10.1.1.48:3000,2
```

```
TargetAddress=10.1.1.49:3000,3
```

In this example, any of the target addresses can be used to reach the same target. A single-connection session can be established to any of these TCP addresses. A multiple-connection session could span addresses .45 and .46 or .47 and .48, but cannot span any other combination. A TargetAddress with its own tag (.49) cannot be combined with any other address within the same session.

This SendTargets response does not indicate whether .49 supports multiple connections per session; it is communicated via the MaxConnections text key upon login to the target.

#### Appendix D. Algorithmic Presentation of Error Recovery Classes

This appendix illustrates the error recovery classes using a pseudo-programming-language. The procedure names are chosen to be obvious to most implementers. Each of the recovery classes described has initiator procedures as well as target procedures. These algorithms focus on outlining the mechanics of error

recovery classes, and do not exhaustively describe all other aspects/cases. Examples of this approach are:

- Handling for only certain Opcode types is shown.
- Only certain reason codes (e.g., Recovery in Logout command) are outlined.
- Resultant cases, such as recovery of Synchronization on a header digest error are considered out-of-scope in these algorithms. In this particular example, a header digest error may lead to connection recovery if some type of sync and steering layer is not implemented.

These algorithms strive to convey the iSCSI error recovery concepts in the simplest terms, and are not designed to be optimal.

#### D.1. General Data Structure and Procedure Description

This section defines the procedures and data structures that are commonly used by all the error recovery algorithms. The structures may not be the exhaustive representations of what is required for a typical implementation.

Data structure definitions -

```
struct TransferContext {
    int TargetTransferTag;
    int ExpectedDataSN;
};

struct TCB {
    /* task control block */
    Boolean SoFarInOrder;
    int ExpectedDataSN; /* used for both R2Ts, and Data */
    int MissingDataSNList[MaxMissingDPDU];
    Boolean FbitReceived;
    Boolean StatusXferd;
    Boolean CurrentlyAllegiant;
    int ActiveR2Ts;
```

```

    int Response;
    char *Reason;
    struct TransferContext
        TransferContextList [MaxOutStandingR2T];
    int InitiatorTaskTag;
    int CmdSN;
    int SNACK_Tag;
};

struct Connection {
    struct Session SessionReference;
    Boolean SoFarInOrder;
    int CID;
    int State;
    int CurrentTimeout;
    int ExpectedStatSN;
    int MissingStatSNList [MaxMissingSPDU];
    Boolean PerformConnectionCleanup;
};

struct Session {
    int NumConnections;
    int CmdSN;
    int Maxconnections;
    int ErrorRecoveryLevel;
    struct iSCSIEndpoint OtherEndInfo;
    struct Connection ConnectionList [MaxSupportedConns];
};

```

Procedure descriptions -

```

Receive-a-In-PDU (transport connection, inbound PDU);
check-basic-validity (inbound PDU);
Start-Timer (timeout handler, argument, timeout value);
Build-And-Send-Reject (transport connection, bad PDU, reason
code);

```

## D.2. Algorithms

### Within-command Error Recovery

#### D.2.1. Procedure Descriptions

```

Recover-Data-if-Possible(last required DataSN, task control
block);
Build-And-Send-DSnack(task control block);
Build-And-Send-RDSnack(task control block);
Build-And-Send-Abort(task control block);
SCSI-Task-Completion(task control block);
Build-And-Send-A-Data-Burst(transport connection, data-
descriptor,
                                task control
block);
Build-And-Send-R2T(transport connection, data-descriptor,
                                task control
block);
Build-And-Send-Status(transport connection, task control block);
Transfer-Context-Timeout-Handler(transfer context);

```

#### Notes:

- One procedure used in this section: Handle-Status-SNACK-request is defined in Within-connection recovery algorithms.
- The Response processing pseudo-code, shown in the target algorithms, applies to all solicited PDUs that carry StatSN
  - SCSI Response, Text Response etc.

#### D.2.2. Initiator Algorithms

```

Recover-Data-if-Possible(LastRequiredDataSN, TCB)
{
    if (operational ErrorRecoveryLevel > 0) {
        if (# of missing PDUs is trackable) {
            Note the missing DataSNs in TCB.
            if (the task spanned a change in
                MaxRecvDataSegmentLength) {

```

```

        iSCSI (Consolidated)
        if (TCB.StatusXferd is TRUE)
            drop the status PDU;
            Build-And-Send-RDSnack(TCB);
        } else {
            Build-And-Send-DSnack(TCB);
        }
    } else {
        TCB.Reason = "Protocol service CRC error";
    }
} else {
    TCB.Reason = "Protocol service CRC error";
}
if (TCB.Reason == "Protocol service CRC error") {
    Clear the missing PDU list in the TCB.
    if (TCB.StatusXferd is not TRUE)
        Build-And-Send-Abort(TCB);
}
}

Receive-a-In-PDU(Connection, CurrentPDU)
{
    check-basic-validity(CurrentPDU);
    if (Header-Digest-Bad) discard, return;
    Retrieve TCB for CurrentPDU.InitiatorTaskTag.
    if ((CurrentPDU.type == Data)
        or (CurrentPDU.type = R2T)) {
        if (Data-Digest-Bad for Data) {
            send-data-SNACK = TRUE;
            LastRequiredDataSN = CurrentPDU.DataSN;
        } else {
            if (TCB.SoFarInOrder = TRUE) {
                if (current DataSN is expected) {
                    Increment TCB.ExpectedDataSN.
                } else {
                    TCB.SoFarInOrder = FALSE;
                    send-data-SNACK = TRUE;
                }
            } else {
                if (current DataSN was considered
missing) {

```

```

iSCSI (Consolidated) 3/11/11
    remove current DataSN from missing
PDU list.
    } else if (current DataSN is higher than
expected) {
        send-data-SNACK = TRUE;
    } else {
        discard, return;
    }
    Adjust TCB.ExpectedDataSN if
appropriate.
    }
    LastRequiredDataSN = CurrentPDU.DataSN - 1;
    }
    if (send-data-SNACK is TRUE and
        task is not already considered failed) {
        Recover-Data-if-Possible(LastRequiredDataSN, TCB);
    }
    if (missing data PDU list is empty) {
        TCB.SoFarInOrder = TRUE;
    }
    if (CurrentPDU.type == R2T) {
        Increment ActiveR2Ts for this task.
        Create a data-descriptor for the data burst.
        Build-And-Send-A-Data-Burst(Connection, data-
descriptor,
TCB);
    }
    } else if (CurrentPDU.type == Response) {
        if (Data-Digest-Bad) {
            send-status-SNACK = TRUE;
        } else {
            TCB.StatusXferd = TRUE;
            Store the status information in TCB.
            if (ExpDataSN does not match) {
                TCB.SoFarInOrder = FALSE;
                Recover-Data-if-Possible(current DataSN, TCB);
            }
            if (missing data PDU list is empty) {
                TCB.SoFarInOrder = TRUE;
            }
        }
    }

```

```

    }
  } else { /* REST UNRELATED TO WITHIN-COMMAND-RECOVERY, NOT
    SHOWN */
    }
    if ((TCB.SoFarInOrder == TRUE) and
        (TCB.StatusXferd == TRUE)) {
      SCSI-Task-Completion(TCB);
    }
  }
}

```

### D.2.3. Target Algorithms

```

Receive-a-In-PDU(Connection, CurrentPDU)
{
  check-basic-validity(CurrentPDU);
  if (Header-Digest-Bad) discard, return;
  Retrieve TCB for CurrentPDU.InitiatorTaskTag.
  if (CurrentPDU.type == Data) {
    Retrieve TContext from CurrentPDU.TargetTransferTag;
    if (Data-Digest-Bad) {
      Build-And-Send-Reject(Connection, CurrentPDU,
        Payload-Digest-Error);
      Note the missing data PDUs in MissingDataRange[].
      send-recovery-R2T = TRUE;
    } else {
      if (current DataSN is not expected) {
        Note the missing data PDUs in MissingDataRange[].
        send-recovery-R2T = TRUE;
      }
      if (CurrentPDU.Fbit == TRUE) {
        if (current PDU is solicited) {
          Decrement TCB.ActiveR2Ts.
        }
        if ((current PDU is unsolicited and
            data received is less than I/O length and
            data received is less than
FirstBurstLength)
            or (current PDU is solicited and the length of
                this burst is less than expected)) {
          send-recovery-R2T = TRUE;
        }
      }
    }
  }
}

```

```

        Note the missing data in MissingDataRange[].
```

}  
 }  
 }  
 Increment TContext.ExpectedDataSN.  
 if (send-recovery-R2T is TRUE and  
 task is not already considered failed) {  
 if (operational ErrorRecoveryLevel > 0) {  
 Increment TCB.ActiveR2Ts.  
 Create a data-descriptor for the data burst  
 from MissingDataRange.  
 Build-And-Send-R2T(Connection, data-descriptor,  
TCB);  
 } else {  
 if (current PDU is the last unsolicited)  
 TCB.Reason = "Not enough unsolicited data";  
 else  
 TCB.Reason = "Protocol service CRC error";  
 }  
 }  
 if (TCB.ActiveR2Ts == 0) {  
 Build-And-Send-Status(Connection, TCB);  
 }  
} else if (CurrentPDU.type == SNACK) {  
 snack-failure = FALSE;  
 if (operational ErrorRecoveryLevel > 0) {  
 if (CurrentPDU.type == Data/R2T) {  
 if (the request is satisfiable) {  
 if (request for Data) {  
 Create a data-descriptor for the data burst  
 from BegRun and RunLength.  
 Build-And-Send-A-Data-Burst(Connection,  
 data-descriptor, TCB);  
 } else { /\* R2T \*/  
 Create a data-descriptor for the data burst  
 from BegRun and RunLength.  
 Build-And-Send-R2T(Connection, data-  
descriptor,  
TCB);  
 }  
 } else {



iSCSI (Consolidated)  
snack-failure = TRUE;

3/11/11

```
    }
  } else if (CurrentPDU.type == status) {
    Handle-Status-SNACK-request(Connection,
CurrentPDU);
    } else if (CurrentPDU.type == DataACK) {
      Consider all data upto CurrentPDU.BegRun as
      acknowledged.
      Free up the retransmission resources for that data.
    } else if (CurrentPDU.type == R-Data SNACK) {
      Create a data descriptor for a data burst
covering
      all unacknowledged data.
      Build-And-Send-A-Data-Burst(Connection,
                                data-descriptor, TCB);
      TCB.SNACK_Tag = CurrentPDU.SNACK_Tag;
      if (there's no more data to send) {
        Build-And-Send-Status(Connection, TCB);
      }
    }
  } else { /* operational ErrorRecoveryLevel = 0 */
    snack-failure = TRUE;
  }
  if (snack-failure == TRUE) {
    Build-And-Send-Reject(Connection, CurrentPDU,
                                SNACK-Reject);

    if (TCB.StatusXferd != TRUE) {
      TCB.Reason = "SNACK Rejected";
      Build-And-Send-Status(Connection, TCB);
    }
  }

  } else { /* REST UNRELATED TO WITHIN-COMMAND-RECOVERY, NOT
SHOWN */
  }
}

Transfer-Context-Timeout-Handler(TContext)
{
  Retrieve TCB and Connection from TContext.
  Decrement TCB.ActiveR2Ts.
```

```

                                iSCSI (Consolidated)                                3/11/11
if (operational ErrorRecoveryLevel > 0 and
    task is not already considered failed) {
    Note the missing data PDUs in MissingDataRange[].
    Create a data-descriptor for the data burst
        from MissingDataRange[].
    Build-And-Send-R2T(Connection, data-descriptor, TCB);
} else {
    TCB.Reason = "Protocol service CRC error";
    if (TCB.ActiveR2Ts = 0) {
        Build-And-Send-Status(Connection, TCB);
    }
}
}

```

### D.3. Within-connection Recovery Algorithms

#### D.3.1. Procedure Descriptions

Procedure descriptions:

```

Recover-Status-if-Possible(transport connection,
                           currently received PDU);
Evaluate-a-StatSN(transport connection, currently received PDU);
Retransmit-Command-if-Possible(transport connection, CmdSN);
Build-And-Send-SSnack(transport connection);
Build-And-Send-Command(transport connection, task control
block);
Command-Acknowledge-Timeout-Handler(task control block);
Status-Expect-Timeout-Handler(transport connection);
Build-And-Send-Nop-Out(transport connection);
Handle-Status-SNACK-request(transport connection, status SNACK
PDU);
Retransmit-Status-Burst(status SNACK, task control block);
Is-Acknowledged(beginning StatSN, run length);

```

Implementation-specific tunables:

InitiatorProactiveSNACKEnabled

Notes:

- The initiator algorithms only deal with unsolicited Nop-In PDUs for generating status SNACKs. A solicited Nop-In PDU

has an assigned StatSN, which, when out of order, could trigger the out of order StatSN handling in Within-command algorithms, again leading to Recover-Status-if-Possible.

- The pseudo-code shown may result in the retransmission of unacknowledged commands in more cases than necessary. This will not, however, affect the correctness of the operation because the target is required to discard the duplicate CmdSNs.
- The procedure Build-And-Send-Async is defined in the Connection recovery algorithms.
- The procedure Status-Expect-Timeout-Handler describes how initiators may proactively attempt to retrieve the Status if they so choose. This procedure is assumed to be triggered much before the standard ULP timeout.

#### D.3.2. Initiator Algorithms

```
Recover-Status-if-Possible(Connection, CurrentPDU)
{
    if ((Connection.state == LOGGED_IN) and
        connection is not already considered
failed) {
        if (operational ErrorRecoveryLevel > 0) {
            if (# of missing PDUs is trackable) {
                Note the missing StatSNs in Connection
                that were not already requested with SNACK;
                Build-And-Send-SSnack(Connection);
            } else {
                Connection.PerformConnectionCleanup = TRUE;
            }
        } else {
            Connection.PerformConnectionCleanup = TRUE;
        }
    }
}
```

```

    }
    if (Connection.PerformConnectionCleanup == TRUE) {
        Start-Timer(Connection-Cleanup-Handler, Connection,
0);
    }
}

Retransmit-Command-if-Possible(Connection, CmdSN)
{
    if (operational ErrorRecoveryLevel > 0) {
        Retrieve the InitiatorTaskTag, and thus TCB for the
CmdSN.
        Build-And-Send-Command(Connection, TCB);
    }
}

Evaluate-a-StatSN(Connection, CurrentPDU)
{
    send-status-SNACK = FALSE;
    if (Connection.SoFarInOrder == TRUE) {
        if (current StatSN is the expected) {
            Increment Connection.ExpectedStatSN.
        } else {
            Connection.SoFarInOrder = FALSE;
            send-status-SNACK = TRUE;
        }
    } else {
        if (current StatSN was considered missing) {
            remove current StatSN from the missing list.
        } else {
            if (current StatSN is higher than expected){
                send-status-SNACK = TRUE;
            } else {
                send-status-SNACK = FALSE;
                discard the PDU;
            }
        }
        Adjust Connection.ExpectedStatSN if appropriate.
        if (missing StatSN list is empty) {

```

```

        iSCSI (Consolidated)
        Connection.SoFarInOrder = TRUE;
    }
}
return send-status-SNACK;
}

Receive-a-In-PDU(Connection, CurrentPDU)
{
    check-basic-validity(CurrentPDU);
    if (Header-Digest-Bad) discard, return;
    Retrieve TCB for CurrentPDU.InitiatorTaskTag.
    if (CurrentPDU.type == Nop-In) {
        if (the PDU is unsolicited) {
            if (current StatSN is not expected) {
                Recover-Status-if-Possible(Connection,
CurrentPDU);
            }
            if (current ExpCmdSN is not Session.CmdSN) {
                Retransmit-Command-if-Possible(Connection,
                    CurrentPDU.ExpCmdSN);
            }
        }
    } else if (CurrentPDU.type == Reject) {
        if (it is a data digest error on immediate data) {
            Retransmit-Command-if-Possible(Connection,
CurrentPDU.BadPDUHeader.CmdSN);
        }
    } else if (CurrentPDU.type == Response) {
        send-status-SNACK = Evaluate-a-StatSN(Connection,
            CurrentPDU);
        if (send-status-SNACK == TRUE)
            Recover-Status-if-Possible(Connection, CurrentPDU);
    } else { /* REST UNRELATED TO WITHIN-CONNECTION-RECOVERY,
        * NOT SHOWN */
    }
}

Command-Acknowledge-Timeout-Handler(TCB)
{

```

```

    Retrieve the Connection for TCB.
    Retransmit-Command-if-Possible(Connection, TCB.CmdSN);
}

```

```

Status-Expect-Timeout-Handler(Connection)
{
    if (operational ErrorRecoveryLevel > 0) {
        Build-And-Send-Nop-Out(Connection);
    } else if (InitiatorProactiveSNACKEnabled){
        if ((Connection.state == LOGGED_IN) and
            connection is not already considered
failed) {
            Build-And-Send-SSnack(Connection);
        }
    }
}

```

### E.3.3 Target Algorithms

```

Handle-Status-SNACK-request(Connection, CurrentPDU)
{
    if (operational ErrorRecoveryLevel > 0) {
        if (request for an acknowledged run) {
            Build-And-Send-Reject(Connection, CurrentPDU,
                                   Protocol-Error);
        } else if (request for an untransmitted run) {
            discard, return;
        } else {
            Retransmit-Status-Burst(CurrentPDU, TCB);
        }
    } else {
        Build-And-Send-Async(Connection, DroppedConnection,
                               DefaultTime2Wait,
DefaultTime2Retain);
    }
}

```

## D.4. Connection Recovery Algorithms

## D.4.1. Procedure Descriptions

```

Build-And-Send-Async(transport connection, reason code,
                    minimum time, maximum time);
Pick-A-Logged-In-Connection(session);
Build-And-Send-Logout(transport connection, logout connection
                    identifier, reason code);
PerformImplicitLogout(transport connection, logout connection
                    identifier, target information);
PerformLogin(transport connection, target information);
CreateNewTransportConnection(target information);
Build-And-Send-Command(transport connection, task control
block);
Connection-Cleanup-Handler(transport connection);
Connection-Resource-Timeout-Handler(transport connection);
Quiesce-And-Prepare-for-New-Allegiance(session, task control
block);
Build-And-Send-Logout-Response(transport connection,
                            CID of connection in recovery, reason
code);
Build-And-Send-TaskMgmt-Response(transport connection,
                            task mgmt command PDU, response code);
Establish-New-Allegiance(task control block, transport
connection);
Schedule-Command-To-Continue(task control block);

```

## Notes:

- Transport exception conditions, such as unexpected connection termination, connection reset, and hung connection while the connection is in the full-feature phase, are all assumed to be asynchronously signaled to the iSCSI layer using the `Transport_Exception_Handler` procedure.

## D.4.2. Initiator Algorithms

```

Receive-a-In-PDU(Connection, CurrentPDU)
{

```

```

check-basic-validity(CurrentPDU);
if (Header-Digest-Bad) discard, return;
Retrieve TCB from CurrentPDU.InitiatorTaskTag.
if (CurrentPDU.type == Async) {
    if (CurrentPDU.AsyncEvent == ConnectionDropped) {
        Retrieve the AffectedConnection for
CurrentPDU.Parameter1.
        AffectedConnection.CurrentTimeout =
CurrentPDU.Parameter3;
        AffectedConnection.State = CLEANUP_WAIT;
        Start-Timer(Connection-Cleanup-Handler,
            AffectedConnection,
CurrentPDU.Parameter2);
    } else if (CurrentPDU.AsyncEvent == LogoutRequest)) {
        AffectedConnection = Connection;
        AffectedConnection.State = LOGOUT_REQUESTED;
        AffectedConnection.PerformConnectionCleanup = TRUE;
        AffectedConnection.CurrentTimeout =
CurrentPDU.Parameter3;
        Start-Timer(Connection-Cleanup-Handler,
            AffectedConnection, 0);
    } else if (CurrentPDU.AsyncEvent == SessionDropped)) {
        for (each Connection) {
            Connection.State = CLEANUP_WAIT;
            Connection.CurrentTimeout = CurrentPDU.Parameter3;
            Start-Timer(Connection-Cleanup-Handler,
                Connection, CurrentPDU.Parameter2);
        }
        Session.state = FAILED;
    }

} else if (CurrentPDU.type == LogoutResponse) {
    Retrieve the CleanupConnection for CurrentPDU.CID.
    if (CurrentPDU.Response = failure) {
        CleanupConnection.State = CLEANUP_WAIT;
    } else {
        CleanupConnection.State = FREE;
    }
} else if (CurrentPDU.type == LoginResponse) {
    if (this is a response to an implicit Logout) {

```



```

iSCSI (Consolidated) 3/11/11
Retrieve the CleanupConnection.
if (successful) {
    CleanupConnection.State = FREE;
    Connection.State = LOGGED_IN;
} else {
    CleanupConnection.State = CLEANUP_WAIT;
    DestroyTransportConnection(Connection);
}
}
} else { /* REST UNRELATED TO CONNECTION-RECOVERY,
        * NOT SHOWN */
}
if (CleanupConnection.State == FREE) {
    for (each command that was active on CleanupConnection) {
        /* Establish new connection allegiance */
        NewConnection = Pick-A-Logged-In-
Connection(Session);
        Build-And-Send-Command(NewConnection, TCB);
    }
}
}

Connection-Cleanup-Handler(Connection)
{
    Retrieve Session from Connection.
    if (Connection can still exchange iSCSI PDUs) {
        NewConnection = Connection;
    } else {
        Start-Timer(Connection-Resource-Timeout-Handler,
            Connection, Connection.CurrentTimeout);
        if (there are other logged-in connections) {
            NewConnection = Pick-A-Logged-In-
Connection(Session);
        } else {
            NewConnection =

CreateTransportConnection(Session.OtherEndInfo);
        Initiate an implicit Logout on NewConnection for
            Connection.CID.

        return;
    }
}

```

```

    }
}
Build-And-Send-Logout(NewConnection, Connection.CID,
                      RecoveryRemove);
}

Transport_Exception_Handler(Connection)
{
    Connection.PerformConnectionCleanup = TRUE;
    if (the event is an unexpected transport disconnect) {
        Connection.State = CLEANUP_WAIT;
        Connection.CurrentTimeout = DefaultTime2Retain;
        Start-Timer(Connection-Cleanup-Handler, Connection,
                    DefaultTime2Wait);

    } else {
        Connection.State = FREE;
    }
}

```

#### D.4.3. Target Algorithms

```

Receive-a-In-PDU(Connection, CurrentPDU)
{
    check-basic-validity(CurrentPDU);
    if (Header-Digest-Bad) discard, return;
    else if (Data-Digest-Bad) {
        Build-And-Send-Reject(Connection, CurrentPDU,
                              Payload-Digest-Error);
        discard, return;
    }
    Retrieve TCB and Session.
    if (CurrentPDU.type == Logout) {
        if (CurrentPDU.ReasonCode = RecoveryRemove) {
            Retrieve the CleanupConnection from CurrentPDU.CID).
            for (each command active on CleanupConnection) {
                Quiesce-And-Prepare-for-New-Allegiance(Session,
TCB);
                TCB.CurrentlyAllegiant = FALSE;
            }
        }
    }
}

```

```

        iSCSI (Consolidated) 3/11/11
        Cleanup-Connection-State(CleanupConnection);
        if ((quiescing successful) and (cleanup successful))
{
            Build-And-Send-Logout-Response(Connection,
            CleanupConnection.CID,
Success);
        } else {
            Build-And-Send-Logout-Response(Connection,
            CleanupConnection.CID,
Failure);
        }
    }
    } else if ((CurrentPDU.type == Login) and
        operational ErrorRecoveryLevel == 2) {
        Retrieve the CleanupConnection from CurrentPDU.CID).
        for (each command active on CleanupConnection) {
            Quiesce-And-Prepare-for-New-Allegiance(Session,
TCB);
            TCB.CurrentlyAllegiant = FALSE;
        }
        Cleanup-Connection-State(CleanupConnection);
        if ((quiescing successful) and (cleanup successful))
{
            Continue with the rest of the Login processing;
        } else {
            Build-And-Send-Login-Response(Connection,
            CleanupConnection.CID, Target
Error);
        }
    }
}
    } else if (CurrentPDU.type == TaskManagement) {
        if (CurrentPDU.function == "TaskReassign") {
            if (Session.ErrorRecoveryLevel < 2) {
                Build-And-Send-TaskMgmt-Response(Connection,
                CurrentPDU, "Allegiance reassignment
                not supported");
            } else if (task is not found) {
                Build-And-Send-TaskMgmt-Response(Connection,
                CurrentPDU, "Task not in task set");
            } else if (task is currently allegiant) {

```

```

        iSCSI (Consolidated)                                3/11/11
        Build-And-Send-TaskMgmt-Response(Connection,
            CurrentPDU, "Task still allegiant");
    } else {
        Establish-New-Allegiance(TCB, Connection);
        TCB.CurrentlyAllegiant = TRUE;
        Schedule-Command-To-Continue(TCB);
    }
}
} else { /* REST UNRELATED TO CONNECTION-RECOVERY,
    * NOT SHOWN */
}
}

Transport_Exception_Handler(Connection)
{
    Connection.PerformConnectionCleanup = TRUE;
    if (the event is an unexpected transport disconnect) {
        Connection.State = CLEANUP_WAIT;
        Start-Timer(Connection-Resource-Timeout-Handler,
Connection,

(DefaultTime2Wait+DefaultTime2Retain));
        if (this Session has full-feature phase connections
left) {
            DifferentConnection =
                Pick-A-Logged-In-Connection(Session);
            Build-And-Send-Async(DifferentConnection,
                DroppedConnection, DefaultTime2Wait,
                DefaultTime2Retain);
        }
    } else {
        Connection.State = FREE;
    }
}

```

## Appendix E. Clearing Effects of Various Events on Targets

### E.1. Clearing Effects on iSCSI Objects

The following tables describe the target behavior on receiving the events specified in the rows of the table. The second table is

an extension of the first table and defines clearing actions for more objects on the same events. The legend is:

Y = Yes (cleared/discarded/reset on the event specified in the row). Unless otherwise noted, the clearing action is only applicable for the issuing initiator port.

N = No (not affected on the event specified in the row, i.e., stays at previous value).

NA = Not Applicable or Not Defined.

	IT (1)	IC (2)	CT (5)	ST (6)	PP (7)
connection failure(8)	Y	Y	N	N	Y
connection state timeout (9)	NA	NA	Y	N	NA
session timeout/ closure/reinstatement (10)	Y	Y	Y	Y	Y (14)

## iSCSI (Consolidated)

3/11/11

session continuation (12)	NA	NA	N (11)	N	NA
successful connection close logout	Y	Y	Y	N	Y (13)
session failure (18)	Y	Y	N	N	Y
successful recovery Logout	Y	Y	N	N	Y (13)
failed Logout	Y	Y	N	N	Y
connection Login (leading)	NA	NA	NA	Y (15)	NA
connection Login (non-leading)	NA	NA	N (11)	N	Y
target cold reset (16)	Y (20)	Y	Y	Y	Y
target warm reset (16)	Y (20)	Y	Y	Y	Y
LU reset (19)	Y (20)	Y	Y	Y	Y
powercycle (16)	Y	Y	Y	Y	Y

1. Incomplete TTTs - Target Transfer Tags on which the target is still expecting PDUs to be received. Examples include TTTs received via R2T, NOP-IN, etc.
2. Immediate Commands - immediate commands, but waiting for execution on a target. For example, Abort Task Set.
5. Connection Tasks - tasks that are active on the iSCSI connection in question.

6. Session Tasks - tasks that are active on the entire iSCSI session. A union of "connection tasks" on all participating connections.
7. Partial PDUs (if any) - PDUs that are partially sent and waiting for transport window credit to complete the transmission.
8. Connection failure is a connection exception condition - one of the transport connections shutdown, transport connections reset, or transport connections timed out, which abruptly terminated the iSCSI full-feature phase connection. A connection failure always takes the connection state machine to the CLEANUP\_WAIT state.
9. Connection state timeout happens if a connection spends more time than agreed upon during Login negotiation in the CLEANUP\_WAIT state, and this takes the connection to the FREE state (M1 transition in connection cleanup state diagram).
10. These are defined in Section 6.3.5.
11. This clearing effect is "Y" only if it is a connection reinstatement and the operational ErrorRecoveryLevel is less than 2.
12. Session continuation is defined in Section 6.3.5.
13. This clearing effect is only valid if the connection is being logged out on a different connection and when the connection being logged out on the target may have some partial PDUs pending to be sent. In all other cases, the effect is "NA".
14. This clearing effect is only valid for a "close the session" logout in a multi-connection session. In all other cases, the effect is "NA".
15. Only applicable if this leading connection login is a session reinstatement. If this is not the case, it is "NA".
16. This operation affects all logged-in initiators.
18. Session failure is defined in Section 6.3.5.
19. This operation affects all logged-in initiators and the clearing effects are only applicable to the LU being reset.

20. With Standard multi-task abort semantics (Section 4.2.3.3), a target warm reset or a target cold reset or an LU reset would clear the active TTTs upon completion. However, the FastAbort multi-task abort semantics defined by Section 4.2.3.4 do not guarantee that the active TTTs are cleared by the end of the reset operations. In fact, the FastAbort semantics are designed to allow clearing the TTTs in a "lazy" fashion after the TMF Response is delivered. Thus, when TaskReporting=FastAbort (Section 13.23) is operational on a session, the clearing effects of reset operations on "Incomplete TTTs" is "N".



## iSCSI (Consolidated)

3/11/11

	DC (1)	DD (2)	SS (3)	CS (4)	DS (5)
connection failure	N	Y	N	N	N
connection state timeout	Y	NA	Y	N	NA
session timeout/ closure/reinstatement	Y	Y	Y (7)	Y	NA
session continuation	N (11)	NA*12	NA	N	NA*13
successful connection close Logout	Y	Y	Y	N	NA
session failure	N	Y	N	N	N
successful recovery Logout	Y	Y	Y	N	N
failed Logout	N	Y (9)	N	N	N
connection Login (leading)	NA	NA	N (8)	N (8)	NA
connection Login (non-leading)	N (11)	NA*12	N (8)	N	NA*13
target cold reset	Y	Y	Y	Y (10)	NA
target warm reset	Y	Y	N	N	NA
LU reset	N	Y	N	N	N
powercycle	Y	Y	Y	Y (10)	NA

1. Discontiguous Commands - commands allegiant to the connection in question and waiting to be reordered in the iSCSI layer. All "Y"s in this column assume that the task causing the event (if

indeed the event is the result of a task) is issued as an immediate command, because the discontinuities can be ahead of the task.

2. Discontiguous Data - data PDUs received for the task in question and waiting to be reordered due to prior discontinuities in DataSN.

3. StatSN

4. CmdSN

5. DataSN

7. It clears the StatSN on all the connections.

8. This sequence number is instantiated on this event.

9. A logout failure drives the connection state machine to the CLEANUP\_WAIT state, similar to the connection failure event. Hence, it has a similar effect on this and several other protocol aspects.

10. This is cleared by virtue of the fact that all sessions with all initiators are terminated.

11. This clearing effect is "Y" if it is a connection reinstatement.

12. This clearing effect is "Y" only if it is a connection reinstatement and the operational ErrorRecoveryLevel is 2.

13. This clearing effect is "N" only if it is a connection reinstatement and the operational ErrorRecoveryLevel is 2.

#### E.2.

#### Clearing Effects on SCSI Objects

The only iSCSI protocol action that can effect clearing actions on SCSI objects is the "I\_T nexus loss" notification (Section 4.3.5.1 Loss of Nexus notification). [SPC3] describes the clearing effects of this notification on a variety of SCSI attributes. In addition, SCSI standards documents (such as [SAM2] and [SBC]) define

additional clearing actions that may take place for several SCSI objects on SCSI events such as LU resets and power-on resets.

Since iSCSI defines a target cold reset as a protocol-equivalent to a target power-cycle, the iSCSI target cold reset must also be considered as the power-on reset event in interpreting the actions defined in the SCSI standards.

When the iSCSI session is reconstructed (between the same SCSI ports with the same nexus identifier) reestablishing the same I\_T nexus, all SCSI objects that are defined to not clear on the "I\_T nexus loss" notification event, such as persistent reservations, are automatically associated to this new session.

#### Acknowledgments

Several individuals on the original IPS Working Group made significant contributions to the original RFCs 3720, 3980, 4850 and 5048.

Specifically, the authors of the original RFCs - which this draft consolidates into a single document - were the following:

RFC 3720: Julian Satran, Kalman Meth, Costa Sapuntzakis, Mallikarjun Chadalapaka, Efri Zeidner

RFC 3980: Marjorie Krueger, Mallikarjun Chadalapaka, Rob Elliott

RFC 4850: David Wysochanski

RFC 5048: Mallikarjun Chadalapaka

Many thanks to Fred Knight for contributing to the UML notations and drawings in this draft.

We would in addition like to acknowledge the following individuals who contributed to this revised draft: David Black, David Harrington, Paul Koning, Mark Edwards.

Authors' Addresses

Mallikarjun Chadalapaka  
Microsoft  
One Microsoft Way  
Redmond WA 98052 USA  
E-mail: cbm@chadalapaka.com

Julian Satran  
E-mail: Julian.Satran@gmail.com

Kalman Meth  
IBM Haifa Research Lab  
Haifa University Campus - Mount Carmel  
Haifa 31905, Israel  
Phone +972.4.829.6341  
E-mail: meth@il.ibm.com

Comments may be sent to Mallikarjun Chadalapaka

Acknowledgement

Funding for the RFC Editor function is currently provided by the  
Internet Society



Storage Maintenance (StorM) Working Group  
Internet Draft  
Intended status: Standards Track  
Expires: October 2011  
Updates: 3720, 5048

F. Knight  
NetApp  
M. Chadalapaka  
Hewlett-Packard Co.  
March 2011

Internet Small Computer Systems Interface (iSCSI) SAM  
draft-ietf-storm-iscsi-sam-02.txt

Abstract

Internet Small Computer Systems Interface (iSCSI) is a SCSI transport protocol that maps the SCSI family of protocols onto TCP/IP. RFC 3720 defines the iSCSI protocol. The current iSCSI protocol (RFC 3720 and RFC 5048) is based on the SAM-2 version of the SCSI family of protocols). This document defines additions and changes to the iSCSI protocol to enable additional features that were added to the SCSI family of protocols through SAM-3, SAM-4, and SAM-5.

This document updates RFC 3720 and RFC 5048 and the text in this document supersedes the text in RFC 3720 and RFC 5048 when the two differ.

Status of this Memo

This Internet-Draft is submitted to IETF in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF), its areas, and its working groups. Note that other groups may also distribute working documents as Internet-Drafts.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

The list of current Internet-Drafts can be accessed at <http://www.ietf.org/lid-abstracts.html>.

The list of Internet-Draft Shadow Directories can be accessed at <http://www.ietf.org/shadow.html>.

This Internet-Draft will expire October, 2011.

Copyright Notice

Copyright (c) 2011 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<http://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

#### Table of Contents

1	Introduction.....	3
2	Definitions, Acronyms, and Document Summary.....	3
2.1	Definitions.....	3
2.2	Acronyms.....	3
2.3	New Semantics.....	3
3	Terminology Mapping.....	4
4	Negotiation of New Feature Use.....	7
5	SCSI Commands.....	8
5.1	SCSI Command Additions.....	8
5.1.1	Command Priority (byte 2).....	8
5.2	SCSI Response Additions.....	9
5.2.1	Status Qualifier.....	10
5.2.2	Data Segment - Sense and Response Data Segment	10
6	Task Management Functions.....	10
6.1	Existing Task Management Functions.....	10
6.2	Task Management Function Additions.....	10
6.2.1	LUN field.....	12
6.2.2	Referenced Task Tag.....	12
6.2.3	RefCmdSN.....	12
6.3	Task Management Function Responses.....	13
6.3.1	Task Management Function Response Additions...	14
6.4	Task Management Requests Affecting Multiple Tasks	14
7	Login/Text Operational Text Keys.....	15
7.1	New Operational Text Keys.....	15
7.1.1	iSCSIProtocolLevel.....	15
8	Security Considerations.....	16
9	IANA Considerations.....	16
10	References .....	17
11	Acknowledgements .....	18

## 1. Introduction

The original [RFC3720] was built based on the SAM-2 model for SCSI. Several new features and capabilities have been added to the SCSI Architecture Model in the intervening years (SAM5 is now the current version of the SCSI Architecture Model). This document is not a complete revision of [RFC3720]. Instead, this document is intended as a companion document to [RFC3720] and [RFC5048].

The text in this document, however, updates and supersedes the text in [RFC3720] and [RFC5048] whenever there is any conflict.

## 2. Definitions, Acronyms, and Document Summary

### 2.1 Definitions

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in [RFC2119].

### 2.2 Acronyms

SAM4	SCSI Architecture Model - 4
SAM5	SCSI Architecture Model - 5
SAM	SAM4 or SAM5

### 2.3 New Semantics

This document specifies new iSCSI semantics. This section summarizes the contents of the document.

Section 3: The mapping of iSCSI objects to SAM5 objects  
The iSCSI node may contain both initiator and target capabilities.

Section 4: The protocol used to negotiate the use of the new capabilities described in this document.

Section 5: New Command operations  
The PRI field for SCSI command priority has been added to the SCSI command PDU (see 5.1.1).  
The Status Qualifier field has been added to the SCSI response PDU (see 5.2.1).  
Sense data may be returned (via autosense) for any SCSI status, not just CHECK CONDITION (see 5.2.2).

Section 6: New Task Management Functions



Four new task management functions (QUERY TASK, QUERY TASK SET, I\_T NEXUS RESET, and QUERY ASYNCHRONOUS EVENT have been added (see 6.2). A new "function succeeded" response has been added (see 6.3.1).

Section 7: New Negotiation key

A new negotiation key has been added to enable the use of the new features in section 5 and section 6.

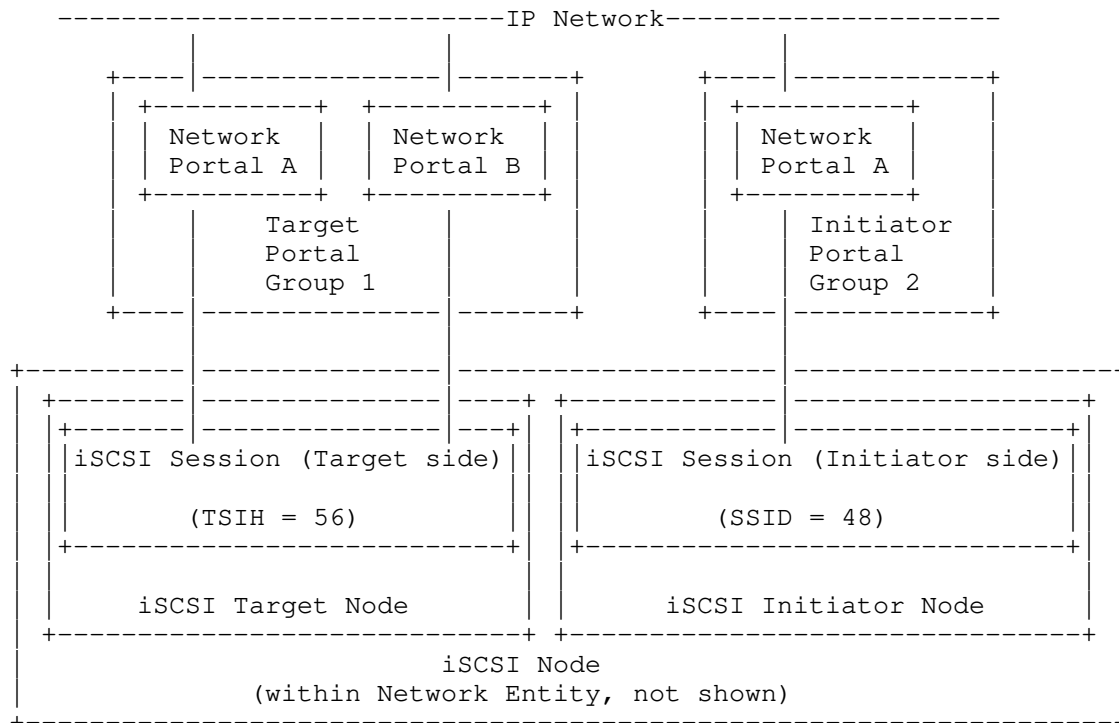
### 3. Terminology Mapping

The iSCSI model (defined in [RFC3720]) uses different terminology than the SCSI Architecture Model. In some cases, iSCSI uses multiple terms to describe what in the SCSI Architecture Model is described with a single term. The iSCSI terms and SAM terms are not necessarily equivalent, but rather, the iSCSI terms represent examples of the objects or classes described in SAM as follows:

RFCxxx Terminology	SAM Terminology
Network Entity	none
iSCSI Node	SCSI Device
iSCSI Name	SCSI Device Name
iSCSI Node Name	SCSI Device Name
iSCSI Initiator Node	SCSI Initiator Device
iSCSI Initiator Name	SCSI Device Name
iSCSI Initiator Port Name iSCSI Node Name + ',i,' + ISID	SCSI Initiator Port Name
iSCSI Target Node	SCSI Target Device
iSCSI Target Name	SCSI Device Name
iSCSI Target Port Name iSCSI Node Name + ',t,' + Target Portal Group Tag	SCSI Target Port Name
iSCSI Target Portal Group	SCSI Target Port
iSCSI Initiator Node + active ISID	SCSI Initiator Port
iSCSI Initiator Name + ,i, + ISID, iSCSI Target Name + ',t,' + Portal Group Tag	I_T Nexus
Target Portal Group Tag	Relative Port ID

RFC EDITORS NOTE: The above reference (in row 1) to [RFCxxx] should reference this RFC, and this note should be removed.

The following diagram shows an example of a combination target device and initiator device. Such a configuration may exist in a target device that implements a SCSI Copy Manager. This example shows how a session that shares Network Portals within a Portal Group may be established (see Target Portal Group 1). In addition, this example shows the Initiator using a different Portal Group than the Target Portal Group, but the Initiator Portal group sharing Network Portal A with the Target Portal Group.



#### 4. Negotiation of New Feature Use

The iSCSIProtocolLevel operational text key (see 7.1.1) containing a value of "2" or higher MUST be negotiated to enable the use of features described in this RFC.

Note that an operational value of "2" or higher for this key on an iSCSI session does not influence the SCSI level features in any way on that I\_T nexus. An operational value of "2" or higher for this key permits the iSCSI-related features defined in this document to be used on all connections on this iSCSI session. SCSI level hand-shakes (e.g. commands, mode pages) eventually determine the existence or lack of various SAM features available for the I\_T nexus between the two SCSI end points). To summarize, negotiation of this key to "2" or higher is a necessary but not a sufficient condition of SAM-4 compliant feature usage at the SCSI protocol level.

For example, an iSCSI implementation may negotiate this new key to "2" but respond to the new task management functions (see 6.2) with a "Task management function not supported" (which indicates a SCSI error that prevents the function from being performed). In contrast, if the key is negotiated to "2", an iSCSI implementation MUST NOT reject a task management function request PDU that requests one of the new task management functions (such a reject would report an iSCSI protocol error).

## 5. SCSI Commands

### 5.1 SCSI Command Additions

The format of the SCSI Command PDU is:

Byte/ /	0								1								2								3							
	0	1	2	3	4	5	6	7	0	1	2	3	4	5	6	7	0	1	2	3	4	5	6	7	0	1	2	3	4	5	6	7
0		.		I		0x01				F		R		W		.	.		ATTR		PRI					Reserved						
4	TotalAHSLength								DataSegmentLength																							
8	Logical Unit Number (LUN)																															
12																																
16	Initiator Task Tag																															
20	Expected Data Transfer Length																															
24	CmdSN																															
28	ExpStatSN																															
32/	SCSI Command Descriptor Block (CDB)																															
+/																																
48/	AHS (Optional)																															
x/	Header Digest (Optional)																															
y/	(DataSegment, Command Data) (Optional)																															
+/																																
z/	Data Digest (Optional)																															

#### 5.1.1 Command Priority (byte 2)

The Command Priority (PRI) specifies the relative scheduling importance of this task in relation to other SIMPLE tasks already in the task set (see [SAM4]).

Section 10, iSCSI PDU Formats of [RFC3720], requires that senders set this field to zero. A sender MUST NOT set this field to a value other than zero unless the iSCSIProtocolLevel text key

defined in section Error! Reference source not found. has been negotiated on the session with a value of "2" or higher.

This field MUST be ignored by iSCSI targets unless the iSCSIProtocolLevel text key with a value of "2" or higher as defined in section Error! Reference source not found. was negotiated on the session.

## 5.2 SCSI Response Additions

The format of the SCSI Response PDU is:

Byte/ /	0								1								2								3																	
	0	1	2	3	4	5	6	7	0	1	2	3	4	5	6	7	0	1	2	3	4	5	6	7	0	1	2	3	4	5	6	7										
0		.		.		0	x	2	1		1		.		.		o		u		0		U		.		Response								Status							
4	TotalAHSLength																DataSegmentLength																									
8	Status Qualifier																Reserved																									
12	Reserved																																									
16	Initiator Task Tag																																									
20	SNACK Tag or Reserved																																									
24	StatSN																																									
28	ExpCmdSN																																									
32	MaxCmdSN																																									
36	ExpDataSN or Reserved																																									
40	Bidirectional Read Residual Count or Reserved																																									
44	Residual Count or Reserved																																									
48	Header-Digest (Optional)																																									
/ Data Segment (Optional)																																										
+ /																																										
Data-Digest (Optional)																																										

### 5.2.1 Status Qualifier

The Status Qualifier provides additional status information (see [SAM4]).

As defined in Section 10, iSCSI PDU Formats of [RFC3720], compliant senders already set this field to zero. Compliant senders MUST NOT set this field to a value other than zero unless the iSCSIProtocolLevel text key with a value of "2" or higher as defined in section 7.1.1. was negotiated on the session.

This field MUST be ignored by receivers unless the iSCSIProtocolLevel text key with a value of "2" or higher as defined in section 7.1.1. was negotiated on the session.

### 5.2.2 Data Segment - Sense and Response Data Segment

Section 10.4.7 of [RFC3720] specifies that iSCSI targets MUST support and enable autosense. If Status is CHECK CONDITION (0x02), then the Data Segment MUST contain sense data for the failed command. While [RFC3720] does not make any statements about the state of the Data Segment when the Status is not CHECK CONDITION (0x02) (i.e., the Data Segment is not prohibited from containing sense data when the Status is not CHECK CONDITION), negotiation of the iSCSIProtocolLevel text key with a value of "2" or higher as defined in section Error! Reference source not found. explicitly indicates that the Data Segment MAY contain sense data at any time, no matter what value is set in the Status field.

## 6. Task Management Functions

### 6.1 Existing Task Management Functions

Section 10.5 of [RFC3720] defines the semantics used to request SCSI Task Management Functions be performed. The following task management functions are defined:

- 1 - ABORT TASK
- 2 - ABORT TASK SET
- 3 - CLEAR ACA
- 4 - CLEAR TASK SET
- 5 - LOGICAL UNIT RESET
- 6 - TARGET WARM RESET
- 7 - TARGET COLD RESET
- 8 - TASK REASSIGN

## 6.2 Task Management Function Additions

Additional task Management function codes are listed below. For a more detailed description of SCSI task management, see [SAM5].

9 - QUERY TASK - determines if the task identified by the Referenced Task Tag field is present in the task set.

10 - QUERY TASK SET - determine if any task is present in the task set.

11 - I\_T NEXUS RESET - perform an I\_T nexus loss function for the I\_T nexus of each logical unit accessible through the I\_T Nexus on which the task management function was received.

12 - QUERY ASYNCHRONOUS EVENT - determine if there is a unit attention condition or a deferred error pending for the I\_T\_L nexus on which the task management function was received.

These task management function requests MUST NOT be sent unless the iSCSIProtocolLevel text key with a value of "2" or higher as defined in section 7.1.1. was negotiated on the session.

Any compliant initiator that sends any of the new task management functions defined in this section MUST also support all new task management function responses (see 6.3.1).

For all of the task management functions detailed in this section, the Task Management function response MUST be returned as detailed in section 6.3 Task Management Function Response.

The iSCSI target MUST ensure that no responses for the tasks covered by a task management function are sent to the iSCSI initiator after the Task Management response except for a task covered by a TASK REASSIGN, QUERY TASK, or QUERY TASK SET.

If a QUERY TASK is issued for a task created by an immediate command then RefCmdSN MUST be that of the Task Management request itself (i.e., CmdSN and RefCmdSN are equal); otherwise RefCmdSN MUST be set to the CmdSN of the task to be queried (lower than CmdSN).

At the target a QUERY TASK function MUST NOT be executed on a Task Management request; such a request MUST result in Task Management response of "Function rejected".

For the I\_T NEXUS RESET function, the target device MUST respond to the function as defined in [SAM4]. Each logical unit accessible via the receiving I\_T NEXUS MUST behave as dictated by



the I\_T nexus loss function in [SAM4] for the I\_T nexus on which the task management function was received. The target device MUST drop all connections in the session over which this function is received. Independent of the DefaultTime2Wait and DefaultTime2Retain value applicable to the session over which this function is received, the target device MUST consider each participating connection in the session to have immediately timed out, leading to FREE state. The resulting timeouts cause the session timeout event defined in [RFC3720], which in turn triggers the I\_T nexus loss notification to the SCSI layer as described in [RFC3720].

#### 6.2.1 LUN field

This field is required for functions that address a specific LU (i.e., ABORT TASK, CLEAR TASK SET, ABORT TASK SET, CLEAR ACA, LOGICAL UNIT RESET, QUERY TASK, QUERY TASK SET, and QUERY ASYNCHRONOUS EVENT) and is reserved in all others.

#### 6.2.2 Referenced Task Tag

The Initiator Task Tag of the task to be aborted for the ABORT TASK function, reassigned for the TASK REASSIGN function, or queried for the QUERY TASK function. For all other functions this field MUST be set to the reserved value 0xffffffff.

#### 6.2.3 RefCmdSN

If a QUERY TASK is issued for a task created by an immediate command then RefCmdSN MUST be that of the Task Management request itself (i.e., CmdSN and RefCmdSN are equal).

For a QUERY TASK of a task created by non-immediate command RefCmdSN MUST be set to the CmdSN of the task identified by the Referenced Task Tag field. Targets must use this field as described in section 10.6.1 of [RFC3720] when the task identified by the Referenced Task Tag field is not in the task set.

### 6.3 Task Management Function Responses

Byte/ /	0	1	2	3
	0 1 2 3 4 5 6 7	0 1 2 3 4 5 6 7	0 1 2 3 4 5 6 7	0 1 2 3 4 5 6 7
0	. . 0x22	1  Reserved	Response	Reserved
4	TotalAHSLength	DataSegmentLength		
8	Additional Response Information			Reserved
12	Reserved			
16	Initiator Task Tag			
20	Reserved			
24	StatSN			
28	ExpCmdSN			
32	MaxCmdSN			
36/	Reserved			
+/				
48	Header-Digest (Optional)			

Section 10.6 of [RFC3720] defines the semantics used for responses to SCSI Task Management Functions. The following responses are defined in [RFC3720]:

- 0 - Function Complete.
- 1 - Task does not exist.
- 2 - LUN does not exist.
- 3 - Task still allegiant.
- 4 - Task allegiance reassignment not supported.
- 5 - Task management function not supported.
- 6 - Function authorization failed.
- 255 - Function rejected.

Responses to new task management functions (see 6.3.1) are listed below. In addition, a new task Management response is listed below. For a more detailed description of SCSI task management responses, see [SAM5].

For the functions QUERY TASK, QUERY TASK SET, I\_T NEXUS RESET, and QUERY ASYNCHRONOUS EVENT, the target performs the requested Task Management function and sends a Task Management response back to the initiator.

#### 6.3.1 Task Management Function Response Additions

The new response is listed below:

7 - Function succeeded.

In symbolic terms Response value 7 maps to the SCSI service response of FUNCTION SUCCEEDED.

The task management function response of function succeeded MUST be supported by an initiator that sends any of the new task management functions (see 6.2).

For the QUERY TASK function, if the specified task is in the task set, then the target returns a Response value of Function succeeded and additional response information is returned as specified in [SAM5]. If the specified task is not in the task set, then the target returns a Response value of Function complete.

For the QUERY TASK SET function, if there is any command present in the task set from the specified I\_T\_L nexus, then the target returns a Response value of Function succeeded. If there are no commands present in the task set from the specified I\_T\_L nexus, then the target returns a Response value of Function complete.

For the I\_T NEXUS RESET function, after completion of the events described in section 6.2 for this function, the target returns a Response value of Function complete. However, because the target drops all connections, the Service Response (defined by [SAM4]) for this SCSI task management function may not be reliably delivered to the issuing initiator port.

For the QUERY ASYNCHRONOUS EVENT, if there is a unit attention condition or deferred error pending for the specified I\_T\_L nexus, then the target returns a Response value of Function succeeded and additional response information is returned as specified in [SAM5]. If there is no unit attention or deferred error pending for the specified I\_T\_L nexus then the target returns a Response value of Function complete.

#### 6.4 Task Management Requests Affecting Multiple Tasks

Section 4.1 of [RFC5048] defines the notion of "affected tasks" in multi-task abort scenarios. This section adds to the list

include in that section by defining the tasks affected by the I\_T NEXUS RESET function.

I\_T NEXUS RESET: All outstanding tasks received on the I\_T nexus on which the function request was received for all logical units accessible to the I\_T nexus.

Section 4.1.2 of [RFC5048] and section 4.1.3 of [RFC5048] identify semantics for task management functions that involve multi-task abort operations. If an iSCSI implementation supports the I\_T NEXUS RESET function, it MUST also support the protocol behavior as defined in those sections and follow the sequence of actions as described in those sections when processing the I\_T NEXUS RESET function.

## 7. Login/Text Operational Text Keys

### 7.1 New Operational Text Keys

#### 7.1.1 iSCSIProtocolLevel

Use: LO

Irrelevant when: SessionType = Discovery

Senders: Initiator and Target

Scope: SW

iSCSIProtocolLevel=<numerical-value-from-1-to-65535>

Default is 1.

Result function is Minimum.

This key is used to negotiate the use of iSCSI features that require different levels of protocol support for proper operation. This key is negotiated on the iSCSI session once the session is in full feature phase.

Negotiation of the iSCSIProtocolLevel key to a value claimed by an RFC indicates that both negotiating parties are compliant to the RFC in question, and agree to support the corresponding semantics on that iSCSI session. An operational value of iSCSI ProtocolLevel = "x" on an iSCSI session requires that the iSCSI protocol semantics on that iSCSI session be a logical superset of the capabilities in all RFCs that have claimed values of an iSCSIProtocolLevel less than "x".

An iSCSIProtocolLevel key negotiated to "2" or higher is required to enable use of features defined in this RFC.

If the negotiation answer is ignored by the acceptor, or the answer from the remote iSCSI end point is key=NotUnderstood, then

the features defined in this RFC, and the features defined in any RFC requiring a key value greater than "2" MUST NOT be used.

## 8. Security Considerations

At the time of writing this document does not introduce any new security considerations other than those described in [RFC3720]. Consequently, all the iSCSI-related security text in [RFC3723] is also directly applicable to this document.

## 9. IANA Considerations

This document modifies or creates a number of iSCSI-related registries. The following iSCSI-related registries are modified:

### 1. iSCSI Task Management Functions Codes

Name of the existing registry: "iSCSI TMF Codes"

Additional entries:

9, QUERY TASK, [RFCxxx]

10, QUERY TASK SET, [RFCxxx]

11, I\_T NEXUS RESET, [RFCxxx]

12, QUERY ASYNCHRONOUS EVENT, [RFCxxx]

-----  
RFC EDITORS NOTE: The above reference to [RFCxxx] should  
reference this RFC, and this note should be removed.  
-----

### 2. iSCSI Login/Text Keys

Name of the existing registry: "iSCSI Text Keys"

Additional entry:

iSCSIProtocolLevel = "2", [RFCxxx]

-----  
RFC EDITORS NOTE: The above references to [RFCxxx] should  
reference this RFC, and this note should be removed.  
-----

This document creates the following iSCSI-related registries for IANA to manage.

### 3. iSCSI Task Management Response Codes

Name of new registry: "iSCSI TMF Response Codes"

Namespace details: Numerical values that can fit in 8 bits.

Information that must be provided to assign a new value: An IESG-approved specification defining the semantics and interoperability requirements of the proposed new value and the fields to be recorded in the registry.

Assignment policy:

If the requested value is not already assigned, it may be assigned to the requester.

8-254: Range reserved by IANA for assignment in this registry.

Fields to record in the registry: Assigned value, Operation Name, and its associated RFC reference.

0x0, Function complete, [RFC3720]

0x1, Task does not exist, [RFC3720]

0x2, LUN does not exist, [RFC3720]

0x3, Task still allegiant, [RFC3720]

0x4, Task allegiance reassignment not supported, [RFC3720]

0x5, Task management function not supported, [RFC3720]

0x6, Function authorization failed, [RFC3720]

0x7, Function succeeded, [RFCxxx]

255, Function rejected, [RFC3720]

---

RFC EDITORS NOTE: The above reference to [RFCxxx] should reference this RFC, and this note should be removed.

---

Allocation Policy:

Standards Action ([IANA])

## 10. References

### 10.1 Normative References

- [RFC2119] Bradner, S. "Key Words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.
- [RFC3720] Satran, J., Meth, K., Sapuntzakis, C., Chadalapaka, M., and E. Zeidner, "Internet Small Computer Systems Interface (iSCSI)", RFC 3720, April 2004.
- [RFC3723] Aboba, B., Tseng, J., Walker, J., Rangan, V., and Travostino, F., "Securing Block Storage Protocols over IP", RFC 3723, April 2004.
- [RFC5048] Chadalapaka, M., "Internet Small Computer System Interface (iSCSI) Corrections and Clarifications", RFC 5048, October 2007.
- [IANA] Narten, T. and H. Alvestrand, "Guidelines for Writing an IANA Considerations Section in RFCs", BCP 26, RFC 5226, May 2008.
- [SAM2] T10/1157D, SCSI Architecture Model - 2 (SAM-2).
- [SAM4] ISO/IEC 14776-414, SCSI Architecture Model - 4 (SAM-4).
- [SAM5] T10/2104D rev r05, SCSI Architecture Model - 5 (SAM-5), Committee Draft.

## 10.2 Additional Reference Sources

For more information on the SCSI Architecture Model, contact the T10 group at <http://www.t10.org>.

## 11. Acknowledgements

The Storage Maintenance (STORM) Working Group in the Transport Area of the IETF has been responsible for defining these additions to the iSCSI protocol (apart from other relevant IP Storage protocols). The editor acknowledges the contributions of the entire working group.

The following individuals directly contributed to identifying [RFCxxx] issues and/or suggesting resolutions to the issues clarified in this document: David Black, Rob Elliott. This document benefited from all of these contributions.

---

RFC EDITORS NOTE: The above reference to [RFCxxx] should reference this RFC, and this note should be removed.

---

Editor's Addresses

Frederick Knight

7301 Kit Creek Road

P.O. Box 13917

Research Triangle Park, NC 27709, USA

Phone: +1-919-476-5362

Email: knight@netapp.com

Mallikarjun Chadalapaka

Hewlett-Packard Company

8000 Foothills Blvd.

Roseville, CA 95747-5668, USA

Phone: +1-916-785-5621

EMail: cbm@rose.hp.com



Storage Maintenance (StorM) Working Group  
Internet Draft  
Intended status : Proposed Standard  
Expires : January 2012  
Obsoletes: RFC 5046

Michael Ko  
Huawei Symantec  
Alexander Nezhinsky  
Mellanox  
July 1, 2011

iSCSI Extensions for RDMA Specification  
draft-ietf-storm-iser-02.txt

Status of this Memo

This Internet-Draft is submitted to IETF in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF), its areas, and its working groups. Note that other groups may also distribute working documents as Internet-Drafts.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

The list of current Internet-Drafts can be accessed at <http://www.ietf.org/lid-abstracts.html>.

The list of Internet-Draft Shadow Directories can be accessed at <http://www.ietf.org/shadow.html>.

This Internet-Draft will expire on January, 2012.

Abstract

iSCSI Extensions for RDMA provides the RDMA data transfer capability to iSCSI by layering iSCSI on top of an RDMA-Capable Protocol. An RDMA-Capable Protocol provides RDMA Read and Write services, which enable data to be transferred directly into SCSI I/O Buffers without intermediate data copies. This document describes the extensions to the iSCSI protocol to support RDMA services as provided by an RDMA-Capable Protocol.

This document obsoletes RFC5046

## Table of Contents

1	Definitions and Acronyms.....	6
1.1	Definitions.....	6
1.2	Acronyms.....	12
1.3	Conventions.....	14
2	Introduction.....	15
2.1	Motivation.....	15
2.2	Architectural Goals.....	16
2.3	Protocol Overview.....	17
2.4	RDMA services and iSER.....	18
2.4.1	STag.....	18
2.4.2	Send.....	19
2.4.3	RDMA Write.....	19
2.4.4	RDMA Read.....	19
2.5	SCSI Read Overview.....	20
2.6	SCSI Write Overview.....	20
2.7	iSCSI/iSER Layering.....	21
3	Upper Layer Interface Requirements.....	22
3.1	Operational Primitives offered by iSER.....	22
3.1.1	Send_Control.....	23
3.1.2	Put_Data.....	23
3.1.3	Get_Data.....	23
3.1.4	Allocate_Connection_Resources.....	24
3.1.5	Deallocate_Connection_Resources.....	24
3.1.6	Enable_Datamover.....	24
3.1.7	Connection_Terminate.....	25
3.1.8	Notice_Key_Values.....	25
3.1.9	Deallocate_Task_Resources.....	25
3.2	Operational Primitives used by iSER.....	26
3.2.1	Control_Notify.....	26
3.2.2	Data_Completion_Notify.....	26
3.2.3	Data_ACK_Notify.....	27
3.2.4	Connection_Terminate_Notify.....	27
3.3	iSCSI Protocol Usage Requirements.....	27
4	Lower Layer Interface Requirements.....	29
4.1	Interactions with the RCaP Layer.....	29
4.2	Interactions with the Transport Layer.....	30
5	Connection Setup and Termination.....	31
5.1	iSCSI/iSER Connection Setup.....	31
5.1.1	Initiator Behavior.....	32
5.1.2	Target Behavior.....	34
5.1.3	iSER Hello Exchange.....	35
5.2	iSCSI/iSER Connection Termination.....	36
5.2.1	Normal Connection Termination at the Initiator.....	36
5.2.2	Normal Connection Termination at the Target.....	37
5.2.3	Termination without Logout Request/Response PDUs.....	37

Internet-Draft	iSER Specification	July 2011
6	Login/Text Operational Keys.....	40
6.1	HeaderDigest and DataDigest.....	40
6.2	MaxRecvDataSegmentLength.....	40
6.3	RDMAExtensions.....	41
6.4	TargetRecvDataSegmentLength.....	42
6.5	InitiatorRecvDataSegmentLength.....	42
6.6	OFMarker and IFMarker.....	43
6.7	MaxOutstandingUnexpectedPDUs.....	43
6.8	MaxAHSLength.....	43
6.9	WriteAddressForSolicitedDataOnly.....	44
6.10	iSERHelloRequired.....	44
7	iSCSI PDU Considerations.....	46
7.1	iSCSI Data-Type PDU.....	46
7.2	iSCSI Control-Type PDU.....	47
7.3	iSCSI PDUs.....	47
7.3.1	SCSI Command.....	47
7.3.2	SCSI Response.....	49
7.3.3	Task Management Function Request/Response.....	50
7.3.4	SCSI Data-out.....	52
7.3.5	SCSI Data-in.....	52
7.3.6	Ready To Transfer (R2T).....	55
7.3.7	Asynchronous Message.....	57
7.3.8	Text Request & Text Response.....	57
7.3.9	Login Request & Login Response.....	57
7.3.10	Logout Request & Logout Response.....	57
7.3.11	SNACK Request.....	58
7.3.12	Reject.....	58
7.3.13	NOP-Out & NOP-In.....	58
8	Flow Control and STag Management.....	59
8.1	Flow Control for RDMA Send Message Types.....	59
8.1.1	Flow Control for Control-Type PDUs from the Initiator.....	59
8.1.2	Flow Control for Control-Type PDUs from the Target.....	62
8.2	Flow Control for RDMA Read Resources.....	63
8.3	STag Management.....	63
8.3.1	Allocation of STags.....	64
8.3.2	Invalidation of STags.....	64
9	iSER Control and Data Transfer.....	66
9.1	iSER Header Format.....	66
9.2	iSER Header Format for iSCSI Control-Type PDU.....	66
9.3	iSER Header Format for iSER Hello Message.....	69
9.4	iSER Header Format for iSER HelloReply Message.....	70
9.5	SCSI Data Transfer Operations.....	71
9.5.1	SCSI Write Operation.....	71
9.5.2	SCSI Read Operation.....	72
9.5.3	Bidirectional Operation.....	72
10	iSER Error Handling and Recovery.....	73
10.1	Error Handling.....	73

Internet-Draft	iSER Specification	July 2011
10.1.1	Errors in the Transport Layer.....	73
10.1.2	Errors in the RCaP Layer.....	74
10.1.3	Errors in the iSER Layer.....	74
10.1.4	Errors in the iSCSI Layer.....	76
10.2	Error Recovery.....	78
10.2.1	PDU Recovery.....	78
10.2.2	Connection Recovery.....	79
11	Security Considerations.....	80
12	IANA Considerations.....	81
13	References.....	82
13.1	Normative References.....	82
13.2	Informative References.....	82
14	Appendix A: Summary of Changes from RFC 5046.....	84
15	Appendix B: Message Format for iSER.....	86
15.1	iWARP Message Format for iSER Hello Message.....	86
15.2	iWARP Message Format for iSER HelloReply Message.....	87
15.3	iWARP Message Format for SCSI Read Command PDU.....	88
15.4	iWARP Message Format for SCSI Read Data.....	89
15.5	iWARP Message Format for SCSI Write Command PDU.....	90
15.6	iWARP Message Format for RDMA Read Request.....	91
15.7	iWARP Message Format for Solicited SCSI Write Data.....	92
15.8	iWARP Message Format for SCSI Response PDU.....	93
16	Appendix C: Architectural discussion of iSER over InfiniBand.....	94
16.1	Host side of iSCSI & iSER connections in Infiniband.....	94
16.2	Storage side of iSCSI & iSER mixed network environment....	95
16.3	Discovery processes for an InfiniBand Host.....	95
16.4	IBTA Connection specifications.....	96
17	Acknowledgments.....	97

## Table of Figures

Figure 1 Example of iSCSI/iSER Layering in Full Feature Phase....	21
Figure 2 iSER Header Format.....	66
Figure 3 iSER Header Format for iSCSI Control-Type PDU.....	67
Figure 4 iSER Header Format for iSER Hello Message.....	69
Figure 5 iSER Header Format for iSER HelloReply Message.....	70
Figure 6 SendSE Message containing an iSER Hello Message.....	86
Figure 7 SendSE Message containing an iSER HelloReply Message....	87
Figure 8 SendSE Message containing a SCSI Read Command PDU.....	88
Figure 9 RDMA Write Message containing SCSI Read Data.....	89
Figure 10 SendSE Message containing a SCSI Write Command PDU.....	90
Figure 11 RDMA Read Request Message.....	91
Figure 12 RDMA Read Response Message containing SCSI Write Data..	92
Figure 13 SendInvSE Message containing SCSI Response PDU.....	93
Figure 14 iSCSI and iSER on IB.....	94
Figure 15 Storage Controller with TCP, iWARP, and IB Connections.	95

## 1 Definitions and Acronyms

## 1.1 Definitions

Advertisement (Advertised, Advertise, Advertisements, Advertises) - The act of informing a remote iSER Layer that a local node's buffer is available to it. A Node makes a buffer available for incoming RDMA Read Request Message or incoming RDMA Write Message access by informing the remote iSER Layer of the Tagged Buffer identifiers (STag, TO, and buffer length). Note that this Advertisement of Tagged Buffer information is the responsibility of the iSER Layer on either end and is not defined by the RDMA-Capable Protocol. A typical method would be for the iSER Layer to embed the Tagged Buffer's STag, TO, and buffer length in a Send Message destined for the remote iSER Layer.

Completion (Completed, Complete, Completes) - Completion is defined as the process by the RDMA-Capable Protocol layer to inform the iSER Layer, that a particular RDMA Operation has performed all functions specified for the RDMA Operation.

Connection - A connection is a logical circuit between the initiator and the target, e.g., a TCP connection. Communication between the initiator and the target occurs over one or more connections. The connections carry control messages, SCSI commands, parameters, and data within iSCSI Protocol Data Units (iSCSI PDUs).

Connection Handle - An information element that identifies the particular iSCSI connection and is unique for a given iSCSI-iSER pair. Every invocation of an Operational Primitive is qualified with the Connection Handle.

Data Sink - The peer receiving a data payload. Note that the Data Sink can be required to both send and receive RCaP Messages to transfer a data payload.

Data Source - The peer sending a data payload. Note that the Data Source can be required to both send and receive RCaP Messages to transfer a data payload.

Datamover Interface (DI) - The interface between the iSCSI Layer and the Datamover Layer as described in [DA].

Datamover Layer - A layer that is directly below the iSCSI Layer and above the underlying transport layers. This layer exposes and

uses a set of transport independent Operational Primitives for the communication between the iSCSI Layer and itself. The Datamover layer, operating in conjunction with the transport layers, moves the control and data information on the iSCSI connection. In this specification, the iSER Layer is the Datamover layer.

Datamover Protocol - A Datamover protocol is the wire-protocol that is defined to realize the Datamover layer functionality. In this specification, the iSER protocol is the Datamover protocol.

Event - An indication provided by the RDMA-Capable Protocol layer to the iSER Layer to indicate a Completion or other condition requiring immediate attention.

Inbound RDMA Read Queue Depth (IRD) - The maximum number of incoming outstanding RDMA Read Requests that the RDMA-Capable Controller can handle on a particular RCaP Stream at the Data Source. For some RDMA-Capable Protocol layers, the term "IRD" may be known by a different name. For example, for InfiniBand, the equivalent for IRD is the Responder Resources.

Invalidate STag - A mechanism used to prevent the Remote Peer from reusing a previous explicitly Advertised STag, until the iSER Layer at the local node makes it available through a subsequent explicit Advertisement.

I/O Buffer - A buffer that is used in a SCSI Read or Write operation so SCSI data may be sent from or received into that buffer.

iSCSI - The iSCSI protocol as defined in [RFC3720] is a mapping of the SCSI Architecture Model of SAM-2 over TCP.

iSCSI control-type PDU - Any iSCSI PDU that is not an iSCSI data-type PDU and also not a SCSI Data-out PDU carrying solicited data is defined as an iSCSI control-type PDU. Specifically, it is to be noted that SCSI Data-out PDUs for unsolicited data are defined as iSCSI control-type PDUs.

iSCSI data-type PDU - An iSCSI data-type PDU is defined as an iSCSI PDU that causes data transfer, transparent to the remote iSCSI Layer, to take place between the peer iSCSI nodes on a full feature phase iSCSI connection. An iSCSI data-type PDU, when requested for transmission by the sender iSCSI Layer, results in the associated data transfer without the participation of the remote iSCSI Layer, i.e. the PDU itself is not delivered as-is

to the remote iSCSI Layer. The following iSCSI PDUs constitute the set of iSCSI data-type PDUs - SCSI Data-In PDU and R2T PDU.

iSCSI Layer - A layer in the protocol stack implementation within an end node that implements the iSCSI protocol and interfaces with the iSER Layer via the Datamover Interface.

iSCSI PDU (iSCSI Protocol Data Unit) - The iSCSI Layer at the initiator and the iSCSI Layer at the target divide their communications into messages. The term "iSCSI protocol data unit" (iSCSI PDU) is used for these messages.

iSCSI/iSER Connection - An iSER-assisted iSCSI connection.

iSCSI/iSER Session - An iSER-assisted iSCSI session.

iSCSI-iSER Pair - The iSCSI Layer and the underlying iSER Layer.

iSER - iSCSI Extensions for RDMA, the protocol defined in this document.

iSER-assisted - A term generally used to describe the operation of iSCSI when the iSER functionality is also enabled below the iSCSI Layer for the specific iSCSI/iSER connection in question.

iSER-IRD - This variable represents the maximum number of incoming outstanding RDMA Read Requests that the iSER Layer at the initiator declares on a particular RCaP Stream.

iSER-ORD - This variable represents the maximum number of outstanding RDMA Read Requests that the iSER Layer can initiate on a particular RCaP Stream. This variable is maintained only by the iSER Layer at the target.

iSER Layer - The layer that implements the iSCSI Extensions for RDMA (iSER) protocol.

iWARP - A suite of wire protocols comprising of [RDMAP], [DDP], and [MPA] when layered above [TCP]. [RDMAP] and [DDP] may be layered above SCTP or other transport protocols.

Local Mapping - A task state record maintained by the iSER Layer that associates the Initiator Task Tag to the Local STag(s). The specifics of the record structure are implementation dependent.



Local Peer - The implementation of the RDMA-Capable Protocol on the local end of the connection. Used to refer to the local entity when describing protocol exchanges or other interactions between two Nodes.

Node - A computing device attached to one or more links of a network. A Node in this context does not refer to a specific application or protocol instantiation running on the computer. A Node may consist of one or more RDMA-Capable Controllers installed in a host computer.

Operational Primitive - An Operational Primitive is an abstract functional interface procedure that requests another layer to perform a specific action on the requestor's behalf or notifies the other layer of some event. The Datamover Interface between an iSCSI Layer and a Datamover layer within an iSCSI end node uses a set of Operational Primitives to define the functional interface between the two layers. Note that not every invocation of an Operational Primitive may elicit a response from the requested layer. A full discussion of the Operational Primitive types and request-response semantics available to iSCSI and iSER can be found in [DA].

Outbound RDMA Read Queue Depth (ORD) - The maximum number of outstanding RDMA Read Requests that the RDMA-Capable Controller can initiate on a particular RCaP Stream at the Data Sink. For some RDMA-Capable Protocol layer, the term "ORD" may be known by a different name. For example, for InfiniBand, the equivalent for ORD is the Initiator Depth.

Phase-Collapse - Refers to the optimization in iSCSI where the SCSI status is transferred along with the final SCSI Data-in PDU from a target. See section 3.2 in [RFC3720].

RCaP Message - One or more packets of the network layer comprising a single RDMA operation or a part of an RDMA Read Operation of the RDMA-Capable Protocol. For iWARP, an RCaP Message is known as an RDMAP Message.

RCaP Stream - A single bidirectional association between the peer RDMA-Capable Protocol layers on two Nodes over a single transport-level stream. For iWARP, an RCaP Stream is known as an RDMAP Stream, and the association is created following a successful Login Phase during which iSER support is negotiated.

RDMA-Capable Protocol (RCaP) - The protocol or protocol suite that provides a reliable RDMA transport functionality, e.g., iWARP, InfiniBand, etc.

RDMA-Capable Controller - A network I/O adapter or embedded controller with RDMA functionality. For example, for iWARP, this could be an RNIC, and for InfiniBand, this could be a HCA (Host Channel Adapter) or TCA (Target Channel Adapter).

RDMA-enabled Network Interface Controller (RNIC) - A network I/O adapter or embedded controller with iWARP functionality.

RDMA Operation - A sequence of RCaP Messages, including control Messages, to transfer data from a Data Source to a Data Sink. The following RDMA Operations are defined - RDMA Write Operation, RDMA Read Operation, Send Operation, Send with Invalidate Operation, Send with Solicited Event Operation, Send with Solicited Event and Invalidate Operation, and Terminate Operation.

RDMA Protocol (RDMAP) - A wire protocol that supports RDMA Operations to transfer ULP data between a Local Peer and the Remote Peer as described in [RDMAP].

RDMA Read Operation - An RDMA Operation used by the Data Sink to transfer the contents of a Data Source buffer from the Remote Peer to a Data Sink buffer at the Local Peer. An RDMA Read operation consists of a single RDMA Read Request Message and a single RDMA Read Response Message.

RDMA Read Request - An RCaP Message used by the Data Sink to request the Data Source to transfer the contents of a buffer. The RDMA Read Request Message describes both the Data Source and the Data Sink buffers.

RDMA Read Response - An RCaP Message used by the Data Source to transfer the contents of a buffer to the Data Sink, in response to an RDMA Read Request. The RDMA Read Response Message only describes the Data Sink buffer.

RDMA Write Operation - An RDMA Operation used by the Data Source to transfer the contents of a Data Source buffer from the Local Peer to a Data Sink buffer at the Remote Peer. The RDMA Write Message only describes the Data Sink buffer.

Remote Direct Memory Access (RDMA) - A method of accessing memory on a remote system in which the local system specifies the remote

location of the data to be transferred. Employing an RDMA-Capable Controller in the remote system allows the access to take place without interrupting the processing of the CPU(s) on the system.

Remote Mapping - A task state record maintained by the iSER Layer that associates the Initiator Task Tag to the Advertised STag(s). The specifics of the record structure are implementation dependent.

Remote Peer - The implementation of the RDMA-Capable Protocol on the opposite end of the connection. Used to refer to the remote entity when describing protocol exchanges or other interactions between two Nodes.

SCSI Layer - This layer builds/receives SCSI CDBs (Command Descriptor Blocks) and sends/receives them with the remaining command execute [SAM2] parameters to/from the iSCSI Layer.

Send - An RDMA Operation that transfers the contents of a Buffer from the Local Peer to a Buffer at the Remote Peer.

Send Message Type - A Send Message, Send with Invalidate Message, Send with Solicited Event Message, or Send with Solicited Event and Invalidate Message.

SendInvSE Message - A Send with Solicited Event and Invalidate Message.

SendSE Message - A Send with Solicited Event Message

Sequence Number (SN) - DataSN for a SCSI Data-in PDU and R2TSN for an R2T PDU. The semantics for both types of sequence numbers are as defined in [RFC3720].

Session, iSCSI Session - The group of Connections that link an initiator SCSI port with a target SCSI port form an iSCSI session (equivalent to a SCSI I-T nexus). Connections can be added to and removed from a session even while the I-T nexus is intact. Across all connections within a session, an initiator sees one and the same target.

Solicited Event (SE) - A facility by which an RDMA Operation sender may cause an Event to be generated at the recipient, if the recipient is configured to generate such an Event, when a Send with Solicited Event or Send with Solicited Event and Invalidate Message is received.

Steering Tag (STag) - An identifier of a Tagged Buffer on a Node (Local or Remote) as defined in [RDMAP] and [DDP]. For other RDMA-Capable Protocols, the Steering Tag may be known by different names but will be herein referred to as STags. For example, for Infiniband, a Remote STag is known as an R-Key, and a Local STag is known as an L-Key, and both will be considered STags.

Tagged Buffer - A buffer that is explicitly Advertised to the iSER Layer at the remote node through the exchange of an STag, Tagged Offset, and length.

Tagged Offset (TO) - The offset within a Tagged Buffer.

Traditional iSCSI - Refers to the iSCSI protocol as defined in [RFC3720] (i.e. without the iSER enhancements).

Untagged Buffer - A buffer that is not explicitly Advertised to the iSER Layer at the remote node.

## 1.2 Acronyms

Acronym	Definition
-----	
AHS	Additional Header Segment
BHS	Basic Header Segment
CO	Connection Only
CRC	Cyclic Redundancy Check
DDP	Direct Data Placement Protocol
DI	Datamover Interface
HCA	Host Channel Adapter
IANA	Internet Assigned Numbers Authority
IB	Infiniband
IETF	Internet Engineering Task Force
I/O	Input - Output

IO	Initialize Only
IP	Internet Protocol
IPoIB	IP over Infiniband
IPsec	Internet Protocol Security
iSER	iSCSI Extensions for RDMA
ITT	Initiator Task Tag
LO	Leading Only
MPA	Marker PDU Aligned Framing for TCP
NOP	No Operation
NSG	Next Stage (during the iSCSI Login Phase)
OS	Operating System
PDU	Protocol Data Unit
R2T	Ready To Transfer
R2TSN	Ready To Transfer Sequence Number
RDMA	Remote Direct Memory Access
RDMAP	Remote Direct Memory Access Protocol
RFC	Request For Comments
RNIC	RDMA-enabled Network Interface Controller
SAM2	SCSI Architecture Model - 2
SCSI	Small Computer Systems Interface
SNACK	Selective Negative Acknowledgment - also Sequence Number Acknowledgement for data
STag	Steering Tag
SW	Session Wide

TCA	Target Channel Adapter
TCP	Transmission Control Protocol
TMF	Task Management Function
TTT	Target Transfer Tag
TO	Tagged Offset
ULP	Upper Level Protocol

### 1.3 Conventions

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in RFC 2119.

## 2.1 Motivation

The iSCSI protocol ([RFC3720]) is a mapping of the SCSI Architecture Model (see [SAM2]) over the TCP protocol. SCSI commands are carried by iSCSI requests and SCSI responses and status are carried by iSCSI responses. Other iSCSI protocol exchanges and SCSI Data are also transported in iSCSI PDUs.

Out-of-order TCP segments in the Traditional iSCSI model have to be stored and reassembled before the iSCSI protocol layer within an end node can place the data in the iSCSI buffers. This reassembly is required because not every TCP segment is likely to contain an iSCSI header to enable its placement and TCP itself does not have a built-in mechanism for signaling ULP message boundaries to aid placement of out-of-order segments. This TCP reassembly at high network speeds is quite counter-productive for the following reasons: wasted memory bandwidth in data copying, need for reassembly memory, wasted CPU cycles in data copying, and the general store-and-forward latency from an application perspective. TCP reassembly was recognized as a serious issue in [RFC3720], and the notion of a "sync and steering layer" was introduced that is optional to implement and use. One specific sync and steering mechanism, called "markers", was defined in [RFC3720] which provides an application-level way of framing iSCSI PDUs within the TCP data stream even when the TCP segments are not yet reassembled to be in-order.

With these defined techniques in [RFC3720], a Network Interface Controller customized for iSCSI (SNIC) could offload the TCP/IP processing and support direct data placement, but most iSCSI implementations do not support iSCSI "markers", making SNIC marker-based direct data placement unusable in practice.

The generic term RDMA-Capable Protocol (RCaP) is used to refer to protocol stacks that provide the RDMA functionality, such as iWARP and InfiniBand.

With the availability of RDMA-Capable Controllers within a host system, which does not have SNICs, it is appropriate for iSCSI to be able to exploit the direct data placement function of the RDMA-Capable Controller like other applications.

iSCSI Extensions for RDMA (iSER) is designed precisely to take advantage of generic RDMA technologies - iSER's goal is to permit iSCSI to employ direct data placement and RDMA capabilities using a generic RDMA-Capable Controller. In summary, iSCSI/iSER protocol

stack is designed to enable scaling to high speeds by relying on a generic data placement process and RDMA technologies and products, which enable direct data placement of both in-order and out-of-order data.

This document describes iSER as a protocol extension to iSCSI, both for convenience of description and also because it is true in a very strict protocol sense. However, it is to be noted that iSER is in reality extending the connectivity of the iSCSI protocol defined in [RFC3720], and the name iSER reflects this reality.

When the iSCSI protocol as defined in [RFC3720] (i.e. without the iSER enhancements) is intended in the rest of the document, the term "Traditional iSCSI" is used to make the intention clear.

## 2.2 Architectural Goals

This section summarizes the architectural goals that guided the design of iSER.

1. Provide an RDMA data transfer model for iSCSI that enables direct in order or out of order data placement of SCSI data into pre-allocated SCSI buffers while maintaining in order data delivery.
2. Not require any major changes to SCSI Architecture Model [SAM2] and SCSI command set standards.
3. Utilize existing iSCSI infrastructure (sometimes referred to as "iSCSI ecosystem") including but not limited to MIB, bootstrapping, negotiation, naming & discovery, and security.
4. Require a session to operate in the Traditional iSCSI data transfer mode if iSER is not supported by either the initiator or the target (not require iSCSI full feature phase interoperability between an end node operating in Traditional iSCSI mode, and an end node operating in iSER-assisted mode).
5. Allow initiator and target implementations to utilize generic RDMA-Capable Controllers such as RNICs, or implement iSCSI and iSER in software (not require iSCSI or iSER specific assists in the RCaP implementation or RDMA-Capable Controller).
6. Require full and only generic RCaP functionality at both the initiator and the target.
7. Implement a light weight Datamover protocol for iSCSI with minimal state maintenance.



Consistent with the architectural goals stated in section 2.2, the iSER protocol does not require changes in the iSCSI ecosystem or any related SCSI specifications. iSER protocol defines the mapping of iSCSI PDUs to RCaP Messages in such a way that it is entirely feasible to realize iSCSI/iSER implementations that are based on generic RDMA-Capable Controllers. The iSER protocol layer requires minimal state maintenance to assist an iSCSI full feature phase connection, besides being oblivious to the notion of an iSCSI session. The crucial protocol aspects of iSER may be summarized thus:

1. iSER-assisted mode is negotiated during the iSCSI login for each session, and an entire iSCSI session can only operate in one mode (i.e. a connection in a session cannot operate in iSER-assisted mode if a different connection of the same session is already in full feature phase in the Traditional iSCSI mode).
2. Once in iSER-assisted mode, all iSCSI interactions on that connection use RCaP Messages.
3. A Send Message Type is used for carrying an iSCSI control-type PDU preceded by an iSER header. See section 7.2 for more details on iSCSI control-type PDUs.
4. RDMA Write, RDMA Read Request, and RDMA Read Response Messages are used for carrying control and all data information associated with the iSCSI data-type PDUs. See section 7.1 for more details on iSCSI data-type PDUs.
5. Target drives all data transfer (with the exception of iSCSI unsolicited data) for SCSI writes and SCSI reads, by issuing RDMA Read Requests and RDMA Writes respectively.
6. RCaP is responsible for ensuring data integrity. (For example, iWARP includes a CRC-enhanced framing layer called MPA on top of TCP; and for Infiniband, the CRCs are included in the Reliable Connection mode). For this reason, iSCSI header and data digests are negotiated to "None" for iSCSI/iSER sessions.
7. The iSCSI error recovery hierarchy defined in [RFC3720] is fully supported by iSER. (However, see section 7.3.11 on the handling of SNACK Request PDUs.)
8. iSER requires no changes to iSCSI authentication, security, and text mode negotiation mechanisms.

Note that Traditional iSCSI implementations may have to be adapted to employ iSER. It is expected that the adaptation when required is likely to be centered around the upper layer interface requirements of iSER (section 3).

## 2.4 RDMA services and iSER

iSER is designed to work with software and/or hardware protocol stacks providing the protocol services defined in RCaP documents such as [RDMAP], [IB], etc. The following subsections describe the key protocol elements of RCaP services that iSER relies on.

### 2.4.1 STag

An STag is the identifier of an I/O Buffer unique to an RDMA-Capable Controller that the iSER Layer Advertises to the remote iSCSI/iSER node in order to complete a SCSI I/O.

In iSER, Advertisement is the act of informing the target by the initiator that an I/O Buffer is available at the initiator for RDMA Read or RDMA Write access by the target. The initiator Advertises the I/O Buffer by including the STag in the header of an iSER Message containing the SCSI Command PDU to the target. The base Tagged Offset is not explicitly specified, but the target must always assume it as zero. The buffer length is as specified in the SCSI Command PDU.

The iSER Layer at the initiator Advertises the STag for the I/O Buffer of each SCSI I/O to the iSER Layer at the target in the iSER header of the SendSE Message containing the SCSI Command PDU, unless the I/O can be completely satisfied by unsolicited data alone.

The iSER Layer at the target provides the STag for the I/O Buffer that is the Data Sink of an RDMA Read Operation (section 2.4.4) to the RCaP layer on the initiator node - i.e. this is completely transparent to the iSER Layer at the initiator.

The iSER protocol is defined so that the Advertised STag is automatically invalidated upon a normal completion of the associated task. This automatic invalidation is realized via the SendInvSE Message carrying the SCSI Response PDU. There are two exceptions to this automatic invalidation - bidirectional commands, and abnormal completion of a command. The iSER Layer at the initiator is required to explicitly invalidate the STag in these cases, in addition to sanity checking the automatic invalidation even when that does happen.

Send is the RDMA Operation that is not addressed to an Advertised buffer by the sending side, and thus uses Untagged buffers on the receiving side.

The iSER Layer at the initiator uses the Send Operation to transmit any iSCSI control-type PDU to the target. As an example, the initiator uses Send Operations to transfer iSER Messages containing SCSI Command PDUs to the iSER Layer at the target.

An iSER layer at the target uses the Send Operation to transmit any iSCSI control-type PDU to the initiator. As an example, the target uses Send Operations to transfer iSER Messages containing SCSI Response PDUs to the iSER Layer at the initiator.

#### 2.4.3 RDMA Write

RDMA Write is the RDMA Operation that is used to place data into an Advertised buffer on the receiving side. The sending side addresses the Message using an STag and a Tagged Offset that are valid on the Data Sink.

The iSER Layer at the target uses the RDMA Write Operation to transfer the contents of a local I/O Buffer to an Advertised I/O Buffer at the initiator. The iSER Layer at the target uses the RDMA Write to transfer whole or part of the data required to complete a SCSI Read command.

The iSER Layer at the initiator does not employ RDMA Writes.

#### 2.4.4 RDMA Read

RDMA Read is the RDMA Operation that is used to retrieve data from an Advertised buffer on a remote node. The sending side of the RDMA Read Request addresses the Message using an STag and a Tagged Offset that are valid on the Data Source in addition to providing a valid local STag and Tagged Offset that identify the Data Sink.

The iSER Layer at the target uses the RDMA Read Operation to transfer the contents of an Advertised I/O Buffer at the initiator to a local I/O Buffer at the target. The iSER Layer at the target uses the RDMA Read to fetch whole or part of the data required to complete a SCSI Write Command.

The iSER Layer at the initiator does not employ RDMA Reads.

## 2.5 SCSI Read Overview

The iSER Layer at the initiator receives the SCSI Command PDU from the iSCSI Layer. The iSER Layer at the initiator generates an STag for the I/O Buffer of the SCSI Read and Advertises the buffer by including the STag as part of the iSER header for the PDU. The iSER Message is transferred to the target using a SendSE Message.

The iSER Layer at the target uses one or more RDMA Writes to transfer the data required to complete the SCSI Read.

The iSER Layer at the target uses a SendInvSE Message to transfer the SCSI Response PDU back to the iSER Layer at the initiator. The iSER Layer at the initiator notifies the iSCSI Layer of the availability of the SCSI Response PDU.

## 2.6 SCSI Write Overview

The iSER Layer at the initiator receives the SCSI Command PDU from the iSCSI Layer. If solicited data transfer is involved, the iSER Layer at the initiator generates an STag for the I/O Buffer of the SCSI Write and Advertises the buffer by including the STag as part of the iSER header for the PDU. The iSER Message is transferred to the target using a SendSE Message.

The iSER Layer at the initiator may optionally send one or more non-immediate unsolicited data PDUs to the target using Send Message Types.

If solicited data transfer is involved, the iSER Layer at the target uses one or more RDMA Reads to transfer the data required to complete the SCSI Write.

The iSER Layer at the target uses a SendInvSE Message to transfer the SCSI Response PDU back to the iSER Layer at the initiator. The iSER Layer at the initiator notifies the iSCSI Layer of the availability of the SCSI Response PDU.

## 2.7 iSCSI/iSER Layering

iSCSI Extensions for RDMA (iSER) is layered between the iSCSI layer and the RCaP layer. Note that the RCaP layer may be composed of one or more distinct protocol layers depending on the specifics of the RCaP. Figure 1 shows an example of the relationship between SCSI, iSCSI, iSER, and the different RCaP layers. For TCP, the RCaP is iWARP. For Infiniband, the RCaP is the Reliable Connected Transport Service. Note that the iSCSI layer as described here supports the RDMA Extensions as used in iSER.

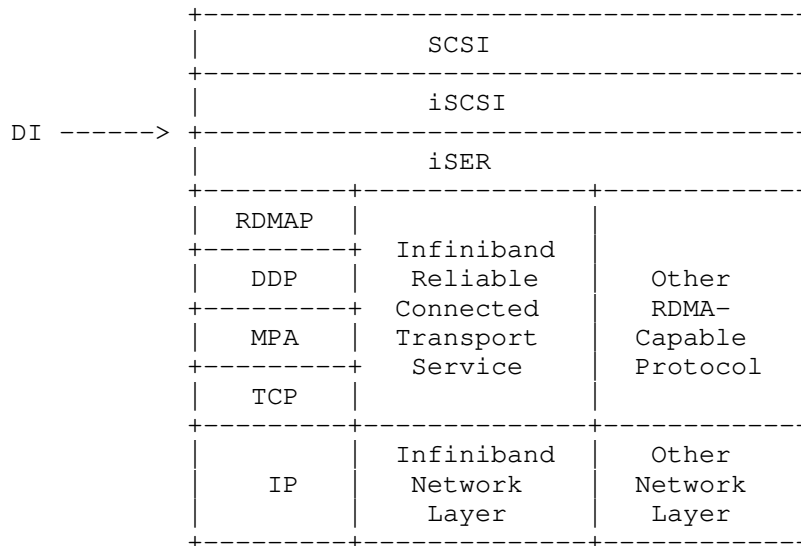


Figure 1 Example of iSCSI/iSER Layering in Full Feature Phase

## 3 Upper Layer Interface Requirements

This section discusses the upper layer interface requirements in the form of an abstract model of the required interactions between the iSCSI Layer and the iSER Layer. The abstract model used here is derived from the architectural model described in [DA]. [DA] also provides a functional overview of the interactions between the iSCSI Layer and the datamover layer as intended by the Datamover Architecture.

The interface requirements are specified by Operational Primitives. An Operational Primitive is an abstract functional interface procedure between the iSCSI Layer and the iSER Layer that requests one layer to perform a specific action on behalf of the other layer or notifies the other layer of some event. Whenever an Operational Primitive is invoked, the Connection\_Handle qualifier is used to identify a particular iSCSI connection. For some Operational Primitives, a Data\_Descriptor is used to identify the iSCSI/SCSI data buffer associated with the requested or completed operation.

The abstract model and the Operational Primitives defined in this section facilitate the description of the iSER protocol. In the rest of the iSER specification, the compliance statements related to the use of these Operational Primitives are only for the purpose of the required interactions between the iSCSI Layer and the iSER Layer. Note that the compliance statements related to the Operational Primitives in the rest of this specification only mandate functional equivalence on implementations, but do not put any requirements on the implementation specifics of the interface between the iSCSI Layer and the iSER Layer.

Each Operational Primitive is invoked with a set of qualifiers which specify the information context for performing the specific action being requested of the Operational Primitive. While the qualifiers are required, the method of realizing the qualifiers (e.g., by passing synchronously with invocation, or by retrieving from task context, or by retrieving from shared memory, etc.) is implementation dependent.

## 3.1 Operational Primitives offered by iSER

The iSER protocol layer MUST support the following Operational Primitives to be used by the iSCSI protocol layer.

### 3.1.1 Send\_Control

Input qualifiers: Connection\_Handle, BHS and AHS (if any) of the iSCSI PDU, PDU-specific qualifiers

Return results: Not specified

This is used by the iSCSI Layers at the initiator and the target to request the outbound transfer of an iSCSI control-type PDU (see section 7.2). Qualifiers that only apply for a particular control-type PDU are known as PDU-specific qualifiers, e.g., ImmediateDataSize for a SCSI Write command. For details on PDU-specific qualifiers, see section 7.3. The iSCSI Layer can only invoke the Send\_Control Operational Primitive when the connection is in iSER-assisted mode.

### 3.1.2 Put\_Data

Input qualifiers: Connection\_Handle, content of a SCSI Data-in PDU header, Data\_Descriptor, Notify\_Enable

Return results: Not specified

This is used by the iSCSI Layer at the target to request the outbound transfer of data for a SCSI Data-in PDU from the buffer identified by the Data\_Descriptor qualifier. The iSCSI Layer can only invoke the Put\_Data Operational Primitive when the connection is in iSER-assisted mode.

The Notify\_Enable qualifier is used to indicate to the iSER Layer whether or not it should generate an eventual local completion notification to the iSCSI Layer. See section 3.2.2 on Data\_Completion\_Notify for details.

### 3.1.3 Get\_Data

Input qualifiers: Connection\_Handle, content of an R2T PDU, Data\_Descriptor, Notify\_Enable

Return results: Not specified

This is used by the iSCSI Layer at the target to request the inbound transfer of solicited data requested by an R2T PDU into the buffer identified by the Data\_Descriptor qualifier. The iSCSI Layer can only invoke the Get\_Data Operational Primitive when the connection is in iSER-assisted mode.

The `Notify_Enable` qualifier is used to indicate to the iSER Layer whether or not it should generate the eventual local completion notification to the iSCSI Layer. See section 3.2.2 on `Data_Completion_Notify` for details.

#### 3.1.4 `Allocate_Connection_Resources`

Input qualifiers: `Connection_Handle`, `Resource_Descriptor` (optional)

Return results: `Status`

This is used by the iSCSI Layers at the initiator and the target to request the allocation of all connection resources necessary to support RCaP for an operational iSCSI/iSER connection. The iSCSI Layer may optionally specify the implementation-specific resource requirements for the iSCSI connection using the `Resource_Descriptor` qualifier.

A return result of `Status=success` means the invocation succeeded, and a return result of `Status=failure` means that the invocation failed. If the invocation is for a `Connection_Handle` for which an earlier invocation succeeded, the request will be ignored by the iSER Layer and the result of `Status=success` will be returned. Only one `Allocate_Connection_Resources` Operational Primitive invocation can be outstanding for a given `Connection_Handle` at any time.

#### 3.1.5 `Deallocate_Connection_Resources`

Input qualifiers: `Connection_Handle`

Return results: Not specified

This is used by the iSCSI Layers at the initiator and the target to request the deallocation of all connection resources that were allocated earlier as a result of a successful invocation of the `Allocate_Connection_Resources` Operational Primitive.

#### 3.1.6 `Enable_Datamover`

Input qualifiers: `Connection_Handle`,  
`Transport_Connection_Descriptor`, `Final_Login_Response_PDU` (optional)

Return results: Not specified



This is used by the iSCSI Layers at the initiator and the target to request that iSER-assisted mode be used for the connection. The `Transport_Connection_Descriptor` qualifier is used to identify the specific connection associated with the `Connection_Handle`. The iSCSI layer can only invoke the `Enable_Datamover` Operational Primitive when there was a corresponding prior resource allocation.

The `Final_Login_Response_PDU` input qualifier is applicable only for a target, and contains the final Login Response PDU that concludes the iSCSI Login Phase.

### 3.1.7 `Connection_Terminate`

Input qualifiers: `Connection_Handle`

Return results: Not specified

This is used by the iSCSI Layers at the initiator and the target to request that a specified iSCSI/iSER connection be terminated and all associated connection and task resources be freed. When this Operational Primitive invocation returns to the iSCSI layer, the iSCSI layer may assume full ownership of all iSCSI-level resources, e.g. I/O Buffers, associated with the connection.

### 3.1.8 `Notice_Key_Values`

Input qualifiers: `Connection_Handle`, number of keys, list of Key-Value pairs

Return results: Not specified

This is used by the iSCSI Layers at the initiator and the target to request the iSER Layer to take note of the specified Key-Value pairs which were negotiated by the iSCSI peers for the connection.

### 3.1.9 `Deallocate_Task_Resources`

Input qualifiers: `Connection_Handle`, ITT

Return results: Not specified

This is used by the iSCSI Layers at the initiator and the target to request the deallocation of all RCaP-specific resources allocated by the iSER Layer for the task identified by the ITT qualifier. The iSER Layer may require a certain number of RCaP-specific resources associated with the ITT for each new iSCSI task. In the normal course of execution, these task-level resources in the iSER Layer

are assumed to be transparently allocated on each task initiation and deallocated on the conclusion of each task as appropriate. In exception scenarios where the task does not conclude with a SCSI Response PDU, the iSER Layer needs to be notified of the individual task terminations to aid its task-level resource management. This Operational Primitive is used for this purpose, and is not needed when a SCSI Response PDU normally concludes a task. Note that RCaP-specific task resources are deallocated by the iSER Layer when a SCSI Response PDU normally concludes a task, even if the SCSI Status was not success.

### 3.2 Operational Primitives used by iSER

The iSER layer MUST use the following Operational Primitives offered by the iSCSI protocol layer when the connection is in iSER-assisted mode.

#### 3.2.1 Control\_Notify

Input qualifiers: Connection\_Handle, an iSCSI control-type PDU

Return results: Not specified

This is used by the iSER Layers at the initiator and the target to notify the iSCSI Layer of the availability of an inbound iSCSI control-type PDU. A PDU is described as "available" to the iSCSI Layer when the iSER Layer notifies the iSCSI Layer of the reception of that inbound PDU, along with an implementation-specific indication as to where the received PDU is.

#### 3.2.2 Data\_Completion\_Notify

Input qualifiers: Connection\_Handle, ITT, SN

Return results: Not specified

This is used by the iSER Layer to notify the iSCSI Layer of the completion of outbound data transfer that was requested by the iSCSI Layer only if the invocation of the Put\_Data Operational Primitive (see section 3.1.2) was qualified with Notify\_Enable set. SN refers to the DataSN associated with the SCSI Data-In PDU.

This is used by the iSER Layer to notify the iSCSI Layer of the completion of inbound data transfer that was requested by the iSCSI Layer only if the invocation of the Get\_Data Operational Primitive (see section 3.1.3) was qualified with Notify\_Enable set. SN refers to the R2TSN associated with the R2T PDU.

### 3.2.3 Data\_ACK\_Notify

Input qualifier: Connection\_Handle, ITT, DataSN

Return results: Not specified

This is used by the iSER Layer at the target to notify the iSCSI Layer of the arrival of the data acknowledgement (as defined in [RFC3720]) requested earlier by the iSCSI Layer for the outbound data transfer via an invocation of the Put\_Data Operational Primitive where the A-bit in the SCSI Data-in PDU is set to 1. See section 7.3.5. DataSN refers to the expected DataSN of the next SCSI Data-in PDU which immediately follows the SCSI Data-in PDU with the A-bit set to which this notification corresponds, with semantics as defined in [RFC3720].

### 3.2.4 Connection\_Terminate\_Notify

Input qualifiers: Connection\_Handle

Return results: Not specified

This is used by the iSER Layers at the initiator and the target to notify the iSCSI Layer of the unsolicited termination or failure of an iSCSI/iSER connection. The iSER Layer MUST deallocate the connection and task resources associated with the terminated connection before the invocation of this Operational Primitive. Note that the Connection\_Terminate\_Notify Operational Primitive is not invoked when the termination of the connection was earlier requested by the local iSCSI Layer.

## 3.3 iSCSI Protocol Usage Requirements

To operate in an iSER-assisted mode, the iSCSI Layers at both the initiator and the target MUST negotiate the RDMAExtensions key (see section 6.3) to "Yes" on the leading connection. If the RDMAExtensions key is not negotiated to "Yes", then iSER-assisted mode MUST NOT be used. If the RDMAExtensions key is negotiated to "Yes" but the invocation of the Allocate\_Connection\_Resources Operational Primitive to the iSER layer fails, the iSCSI layer MUST fail the iSCSI Login process or terminate the connection as appropriate. See section 10.1.3.1 for details.

If the RDMAExtensions key is negotiated to "Yes", the iSCSI Layer MUST satisfy the following protocol usage requirements from the iSER protocol:

1. The iSCSI Layer at the initiator MUST set ExpDataSN to 0 in Task Management Function Requests for Task Allegiance Reassignment for read/bidirectional commands, so as to cause the target to send all unacknowledged read data.
2. The iSCSI Layer at the target MUST always return the SCSI status in a separate SCSI Response PDU for read commands, i.e., there MUST NOT be a "phase collapse" in concluding a SCSI Read Command.
3. The iSCSI Layers at both the initiator and the target MUST support the keys as defined in section 6 on Login/Text Operational Keys. If used as specified, these keys MUST NOT be answered with NotUnderstood and the semantics as defined MUST be followed for each iSER-assisted connection.
4. The iSCSI Layer at the initiator MUST NOT issue SNACKs for PDUs.

#### 4.1 Interactions with the RCaP Layer

The iSER protocol layer is layered on top of an RCaP layer (see Figure 1) and the following are the key features that are assumed to be supported by any RCaP layer:

- \* The RCaP layer supports all basic RDMA operations, including RDMA Write Operation, RDMA Read Operation, Send Operation, Send with Invalidate Operation, Send with Solicited Event Operation, Send with Solicited Event & Invalidate Operation, and Terminate Operation.
- \* The RCaP layer provides reliable, in-order message delivery and direct data placement.
- \* When the iSER Layer initiates an RDMA Read Operation following an RDMA Write Operation on one RCaP Stream, the RDMA Read Response Message processing on the remote node will be started only after the preceding RDMA Write Message payload is placed in the memory of the remote node.
- \* The RCaP layer encapsulates a single iSER Message into a single RCaP Message on the Data Source side. The RCaP layer decapsulates the iSER Message before delivering it to the iSER Layer on the Data Sink side.
- \* When the iSER Layer provides the STag to be remotely invalidated to the RCaP layer for a SendInvSE Message, the RCaP layer uses this STag as the STag to be invalidated in the SendInvSE Message.
- \* The RCaP layer uses the STag and Tagged Offset provided by the iSER Layer for the RDMA Write and RDMA Read Request Messages.
- \* When the RCaP layer delivers the content of an RDMA Send Message Type to the iSER Layer, the RCaP layer provides the length of the RDMA Send message. This ensures that the iSER Layer does not have to carry a length field in the iSER header.
- \* When the RCaP layer delivers the SendSE or SendInvSE Message to the iSER Layer, it notifies the iSER Layer with the mechanism provided on that interface.
- \* When the RCaP layer delivers a SendInvSE Message to the iSER Layer, it passes the value of the STag that was invalidated.

- \* The RCaP layer propagates all status and error indications to the iSER Layer.
- \* For a transport layer that operates in byte stream mode such as TCP, the RCaP implementation supports the enabling of the RDMA mode after Connection establishment and the exchange of Login parameters in byte stream mode. For a transport layer that provides message delivery capability such as [IB], the RCaP implementation supports the use of the messaging capability by the iSCSI Layer directly for the Login phase after connection establishment before enabling iSER-assisted mode.
- \* Whenever the iSER Layer terminates the RCaP Stream, the RCaP layer terminates the associated Connection.

#### 4.2 Interactions with the Transport Layer

After the iSER connection is established, the RCaP layer and the underlying transport layer are responsible for maintaining the Connection and reporting to the iSER Layer any Connection failures.

## 5 Connection Setup and Termination

## 5.1 iSCSI/iSER Connection Setup

During connection setup, the iSCSI Layer at the initiator is responsible for establishing a connection with the target. After the connection is established, the iSCSI Layers at the initiator and the target enter the Login Phase using the same rules as outlined in [RFC3720]. The connection transitions into the iSCSI full feature phase in iSER-assisted mode following a successful login negotiation between the initiator and the target in which iSER-assisted mode is negotiated and the connection resources necessary to support RCaP have been allocated at both the initiator and the target. The same connection MUST be used for both the iSCSI Login phase and the subsequent iSER-assisted full feature phase.

iSER-assisted mode MUST be enabled only if it is negotiated on the leading connection during the LoginOperationalNegotiation Stage of the iSCSI Login Phase. iSER-assisted mode is negotiated using the RDMAExtensions=<boolean-value> key. Both the initiator and the target MUST exchange the RDMAExtensions key with the value set to "Yes" to enable iSER-assisted mode. If both the initiator and the target fail to negotiate the RDMAExtensions key set to "Yes", then the connection MUST continue with the login semantics as defined in [RFC3720]. If the RDMAExtensions key is not negotiated to Yes, then for some RCaP implementation (such as [IB]), the connection may need to be re-established in TCP capable mode. (For InfiniBand this will require an [IPoIB] type connection.)

iSER-assisted mode is defined for a Normal session only and the RDMAExtensions key MUST NOT be negotiated for a Discovery session. Discovery sessions are always conducted using the transport layer as described in [RFC3720].

An iSER enabled node is not required to initiate the RDMAExtensions key exchange if its preference is for the Traditional iSCSI mode. The RDMAExtensions key, if offered, MUST be sent in the first available Login Response or Login Request PDU in the LoginOperationalNegotiation stage. This is due to the fact that the value of some login parameters might depend on whether iSER-assisted mode is enabled or not.

iSER-assisted mode is a session-wide attribute. If both the initiator and the target negotiated RDMAExtensions="Yes" on the leading connection of a session, then all subsequent connections of the same session MUST enable iSER-assisted mode without having to exchange RDMAExtensions key during the iSCSI Login Phase.

Conversely, if both the initiator and the target failed to negotiate RDMAExtensions to "Yes" on the leading connection of a session, then the RDMAExtensions key MUST NOT be negotiated further on any additional subsequent connection of the session.

When the RDMAExtensions key is negotiated to "Yes", the HeaderDigest and the DataDigest keys MUST be negotiated to "None" on all iSCSI/iSER connections participating in that iSCSI session. This is because, for an iSCSI/iSER connection, RCaP is responsible for providing error detection that is at least as good as a 32-bit CRC for all iSER Messages. Furthermore, all SCSI Read data are sent using RDMA Write Messages instead of the SCSI Data-in PDUs, and all solicited SCSI write data are sent using RDMA Read Response Messages instead of the SCSI Data-out PDUs. HeaderDigest and DataDigest which apply to iSCSI PDUs would not be appropriate for RDMA Read and RDMA Write operations used with iSER.

#### 5.1.1 Initiator Behavior

If the outcome of the iSCSI negotiation is to enable iSER-assisted mode, then on the initiator side, prior to sending the Login Request with the T (Transit) bit set to 1 and the NSG (Next Stage) field set to FullFeaturePhase, the iSCSI Layer MUST request the iSER Layer to allocate the connection resources necessary to support RCaP by invoking the `Allocate_Connection_Resources` Operational Primitive. The connection resources required are defined by implementation and are outside the scope of this specification. The iSCSI Layer may invoke the `Notice_Key_Values` Operational Primitive before invoking the `Allocate_Connection_Resources` Operational Primitive to request the iSER Layer to take note of the negotiated values of the iSCSI keys for the Connection. The specific keys to be passed in as input qualifiers are implementation dependent. These may include, but not limited to, `MaxOutstandingR2T`, `ErrorRecoveryLevel`, etc.

To minimize the potential for a denial of service attack, the iSCSI Layer MUST NOT request the iSER Layer to allocate the connection resources necessary to support RCaP until the iSCSI layer is sufficiently far along in the iSCSI Login Phase that it is reasonably certain that the peer side is not an attacker. In particular, if the Login Phase includes a `SecurityNegotiation` stage, the iSCSI Layer MUST defer the connection resource allocation (i.e. invoking the `Allocate_Connection_Resources` Operational Primitive) to the `LoginOperationalNegotiation` stage ([RFC3720]) so that the resource allocation occurs after the authentication phase is completed.



Among the connection resources allocated at the initiator is the Inbound RDMA Read Queue Depth (IRD). As described in section 9.5.1, R2Ts are transformed by the target into RDMA Read operations. IRD limits the maximum number of simultaneously incoming outstanding RDMA Read Requests per an RCaP Stream from the target to the initiator. The required value of IRD is outside the scope of the iSER specification. The iSER Layer at the initiator MUST set IRD to 1 or higher if R2Ts are to be used in the connection. However, the iSER Layer at the initiator MAY set IRD to 0 based on implementation configuration which indicates that no R2Ts will be used on that connection. Initially, the iSER-IRD value at the initiator SHOULD be set to the IRD value at the initiator and MUST NOT be more than the IRD value.

On the other hand, the Outbound RDMA Read Queue Depth (ORD) MAY be set to 0 since the iSER Layer at the initiator does not issue RDMA Read Requests to the target.

Failure to allocate the requested connection resources locally results in a login failure and its handling is described in section 10.1.3.1.

If the iSER Layer at the initiator is successful in allocating the connection resources necessary to support RCaP, the following events MUST occur in the specified sequence:

1. The iSER Layer MUST return a success status to the iSCSI Layer in response to the Allocate\_Connection\_Resources Operational Primitive.
2. After the target returns the Login Response with the T bit set to 1 and the NSG field set to FullFeaturePhase, and a status class of 0 (Success), the iSCSI Layer MUST invoke the Enable\_Datamover Operational Primitive with the following qualifiers. (See section 10.1.4.6 for the case when the status class is not Success.):
  - a. Connection\_Handle that identifies the iSCSI connection.
  - b. Transport\_Connection\_Descriptor which identifies the specific transport connection associated with the Connection\_Handle.
3. The iSER Layer MUST send the iSER Hello Message as the first iSER Message if iSERHelloRequired is negotiated to "Yes". See Section 5.1.3 on iSER Hello Exchange.

## 5.1.2 Target Behavior

If the outcome of the iSCSI negotiation is to enable iSER-assisted mode, then on the target side, prior to sending the Login Response with the T (Transit) bit set to 1 and the NSG (Next Stage) field set to FullFeaturePhase, the iSCSI Layer MUST request the iSER Layer to allocate the resources necessary to support RCaP by invoking the `Allocate_Connection_Resources` Operational Primitive. The connection resources required are defined by implementation and are outside the scope of this specification. Optionally, the iSCSI Layer may invoke the `Notice_Key_Values` Operational Primitive before invoking the `Allocate_Connection_Resources` Operational Primitive to request the iSER Layer to take note of the negotiated values of the iSCSI keys for the Connection. The specific keys to be passed in as input qualifiers are implementation dependent. These may include, but not limited to, `MaxOutstandingR2T`, `ErrorRecoveryLevel`, etc.

To minimize the potential for a denial of service attack, the iSCSI Layer MUST NOT request the iSER Layer to allocate the connection resources necessary to support RCaP until the iSCSI layer is sufficiently far along in the iSCSI Login Phase that it is reasonably certain that the peer side is not an attacker. In particular, if the Login Phase includes a `SecurityNegotiation` stage, the iSCSI Layer MUST defer the connection resource allocation (i.e. invoking the `Allocate_Connection_Resources` Operational Primitive) to the `LoginOperationalNegotiation` stage ([RFC3720]) so that the resource allocation occurs after the authentication phase is completed.

Among the connection resources allocated at the target is the Outbound RDMA Read Queue Depth (ORD). As described in section 9.5.1, R2Ts are transformed by the target into RDMA Read operations. The ORD limits the maximum number of simultaneously outstanding RDMA Read Requests per RCaP Stream from the target to the initiator. Initially, the iSER-ORD value at the target SHOULD be set to the ORD value at the target.

On the other hand, the IRD at the target MAY be set to 0 since the iSER Layer at the target does not expect RDMA Read Requests to be issued by the initiator.

Failure to allocate the requested connection resources locally results in a login failure and its handling is described in section 10.1.3.1.

If the iSER Layer at the target is successful in allocating the connection resources necessary to support RCaP, the following events MUST occur in the specified sequence:

1. The iSER Layer MUST return a success status to the iSCSI Layer in response to the Allocate\_Connection\_Resources Operational Primitive.
2. The iSCSI Layer MUST invoke the Enable\_Datamover Operational Primitive with the following qualifiers:
  - a. Connection\_Handle that identifies the iSCSI connection.
  - b. Transport\_Connection\_Descriptor which identifies the specific transport connection associated with the Connection\_Handle.
  - c. The final transport layer (e.g. TCP) message containing the Login Response with the T bit set to 1 and the NSG field set to FullFeaturePhase
3. The iSER Layer MUST send the final Login Response PDU in the native transport mode to conclude the iSCSI Login Phase. If the underlying transport is TCP, then the iSER Layer MUST send the final Login Response PDU in byte stream mode.
4. After receiving the iSER Hello Message from the initiator, the iSER Layer MUST respond with the iSER HelloReply Message to be sent as the first iSER Message if iSERHelloRequired is negotiated to "Yes". See section 5.1.3 on iSER Hello Exchange for more details.

Note: In the above sequence, the operations as described in bullets 3 and 4 MUST be performed atomically for iWARP connections. Failure to do this may result in race conditions.

### 5.1.3 iSER Hello Exchange

If iSERHelloRequired is negotiated to "Yes", the first iSER Message sent by the iSER Layer at the initiator to the target MUST be the iSER Hello Message. The iSER Hello Message is used by the iSER Layer at the initiator to declare iSER parameters to the target. See section 9.3 on iSER Header Format for iSER Hello Message.

In response to the iSER Hello Message, the iSER Layer at the target MUST return the iSER HelloReply Message as the first iSER Message sent by the target. The iSER HelloReply Message is used by the iSER

Layer at the target to declare iSER parameters to the initiator.  
See section 9.4 on iSER Header Format for iSER HelloReply Message.

In the iSER Hello Message, the iSER Layer at the initiator declares the iSER-IRD value to the target.

Upon receiving the iSER Hello Message, the iSER Layer at the target MUST set the iSER-ORD value to the minimum of the iSER-ORD value at the target and the iSER-IRD value declared by the initiator. The iSER Layer at the target MAY adjust (lower) its ORD value to match the iSER-ORD value if the iSER-ORD value is smaller than the ORD value at the target in order to free up the unused resources.

In the iSER HelloReply Message, the iSER Layer at the target declares the iSER-ORD value to the initiator.

Upon receiving the iSER HelloReply Message, the iSER Layer at the initiator MAY adjust (lower) its IRD value to match the iSER-ORD value in order to free up the unused resources, if the iSER-ORD value declared by the target is smaller than the iSER-IRD value declared by the initiator.

It is an iSER level negotiation failure if the iSER parameters declared in the iSER Hello Message by the initiator are unacceptable to the target. This includes the following:

- \* The initiator-declared iSER-IRD value is greater than 0 and the target-declared iSER-ORD value is 0.
- \* The initiator-supported and the target-supported iSER protocol versions do not overlap.

See section 10.1.3.2 on the handling of the error situation.

If iSERHelloRequired is negotiated to "No", then the iSER layer at the initiator MUST NOT send the iSER Hello Message to the target, and if an iSER Hello Message or an iSER HelloReply Message is sent, then it is treated as an iSER protocol error. See section 10.1.3.4 for more details.

## 5.2 iSCSI/iSER Connection Termination

### 5.2.1 Normal Connection Termination at the Initiator

The iSCSI Layer at the initiator terminates an iSCSI/iSER connection normally by invoking the Send\_Control Operational Primitive qualified with the Logout Request PDU. The iSER Layer at the

initiator MUST use a SendSE Message to send the Logout Request PDU to the target. After the iSER Layer at the initiator receives the SendSE Message containing the Logout Response PDU from the target, it MUST notify the iSCSI Layer by invoking the Control\_Notify Operational Primitive qualified with the Logout Response PDU.

After the iSCSI logout process is complete, the iSCSI layer at the target is responsible for closing the iSCSI/iSER connection as described in Section 5.2.2. After the RCaP layer at the initiator reports that the Connection has been closed, the iSER Layer at the initiator MUST deallocate all connection and task resources (if any) associated with the connection, invalidate the Local Mapping(s) (if any) that associate the ITT(s) used on that connection to the local STag(s) before notifying the iSCSI Layer by invoking the Connection\_Terminate\_Notify Operational Primitive.

### 5.2.2 Normal Connection Termination at the Target

Upon receiving the SendSE Message containing the Logout Request PDU, the iSER Layer at the target MUST notify the iSCSI Layer at the target by invoking the Control\_Notify Operational Primitive qualified with the Logout Request PDU. The iSCSI Layer completes the logout process by invoking the Send\_Control Operational Primitive qualified with the Logout Response PDU. The iSER Layer at the target MUST use a SendSE Message to send the Logout Response PDU to the initiator. After the iSCSI logout process is complete, the iSCSI Layer at the target MUST request the iSER Layer at the target to terminate the RCaP Stream by invoking the Connection\_Terminate Operational Primitive.

As part of the termination process, the RCaP layer MUST close the Connection. When the RCaP layer notifies the iSER Layer after the RCaP Stream and the associated Connection are terminated, the iSER Layer MUST deallocate all connection and task resources (if any) associated with the connection, and invalidate the Local and Remote Mapping(s) (if any) that associate the ITT(s) used on that connection to the local STag(s) and the Advertised STag(s) respectively.

### 5.2.3 Termination without Logout Request/Response PDUs

#### 5.2.3.1 Connection Termination Initiated by the iSCSI Layer

The Connection\_Terminate Operational Primitive MAY be invoked by the iSCSI Layer to request the iSER Layer to terminate the RCaP Stream without having previously exchanged the Logout Request and Logout Response PDUs between the two iSCSI/iSER nodes. As part of the

termination process, the RCaP layer will close the Connection. When the RCaP layer notifies the iSER Layer after the RCaP Stream and the associated Connection are terminated, the iSER Layer MUST perform the following actions.

If the `Connection_Terminate` Operational Primitive is invoked by the iSCSI Layer at the target, then the iSER Layer at the target MUST deallocate all connection and task resources (if any) associated with the connection, and invalidate the Local and Remote Mappings (if any) that associate the ITT(s) used on the connection to the local STag(s) and the Advertised STag(s) respectively.

If the `Connection_Terminate` Operational Primitive is invoked by the iSCSI Layer at the initiator, then the iSER Layer at the initiator MUST deallocate all connection and task resources (if any) associated with the connection, and invalidate the Local Mapping(s) (if any) that associate the ITT(s) used on the connection to the local STag(s).

#### 5.2.3.2 Connection Termination Notification to the iSCSI Layer

If the iSCSI/iSER connection is terminated without the invocation of `Connection_Terminate` from the iSCSI Layer, the iSER Layer MUST notify the iSCSI Layer that the iSCSI/iSER connection has been terminated by invoking the `Connection_Terminate_Notify` Operational Primitive.

Prior to invoking `Connection_Terminate_Notify`, the iSER Layer at the target MUST deallocate all connection and task resources (if any) associated with the connection, and invalidate the Local and Remote Mappings (if any) that associate the ITT(s) used on the connection to the local STag(s) and the Advertised STag(s) respectively.

Prior to invoking `Connection_Terminate_Notify`, the iSER Layer at the initiator MUST deallocate all connection and task resources (if any) associated with the connection, and invalidate the Local Mappings (if any) that associate the ITT(s) used on the connection to the local STag(s).

If the remote iSCSI/iSER node initiated the closing of the Connection (e.g., by sending a TCP FIN or TCP RST), the iSER Layer MUST notify the iSCSI Layer after the RCaP layer reports that the Connection is closed by invoking the `Connection_Terminate_Notify` Operational Primitive.

Another example of a Connection termination without a preceding logout is when the iSCSI Layer at the initiator does an implicit logout (connection reinstatement).

## 6 Login/Text Operational Keys

Certain iSCSI login/text operational keys have restricted usage in iSER, and additional keys are used to support the iSER protocol functionality. All other keys defined in [RFC3720] and not discussed in this section may be used on iSCSI/iSER connections with the same semantics.

## 6.1 HeaderDigest and DataDigest

Irrelevant when: RDMAExtensions=Yes

Negotiations resulting in RDMAExtensions=Yes for a session implies HeaderDigest=None and DataDigest=None for all connections in that session and overrides both the default and an explicit setting.

## 6.2 MaxRecvDataSegmentLength

For an iSCSI connection belonging to a session in which RDMAExtensions=Yes was negotiated on the leading connection of the session, MaxRecvDataSegmentLength need not be declared in the Login Phase, and MUST be ignored if it is declared. Instead InitiatorRecvDataSegmentLength (as described in section 6.5) and TargetRecvDataSegmentLength (as described in section 6.4) keys are negotiated. The values of the local and remote MaxRecvDataSegmentLength are derived from the InitiatorRecvDataSegmentLength and TargetRecvDataSegmentLength keys.

In the full feature phase, the initiator MUST consider the value of its local MaxRecvDataSegmentLength (that it would have declared to the target) as having the value of InitiatorRecvDataSegmentLength, and the value of the remote MaxRecvDataSegmentLength (that would have been declared by the target) as having the value of TargetRecvDataSegmentLength. Similarly, the target MUST consider the value of its local MaxRecvDataSegmentLength (that it would have declared to the initiator) as having the value of TargetRecvDataSegmentLength, and the value of the remote MaxRecvDataSegmentLength (that would have been declared by the initiator) as having the value of InitiatorRecvDataSegmentLength.

Note that RFC 3720 requires that when a target receives a NOP-Out request with a valid Initiator Task Tag, it responds with a NOP-In with the same Initiator Task Tag that was provided in the NOP-Out request. Furthermore, it returns the first MaxRecvDataSegmentLength bytes of the initiator provided Ping Data. Since there is no MaxRecvDataSegmentLength common to the initiator and the target in



iSER, the length of the data sent with the NOP-Out request MUST not exceed InitiatorMaxRecvDataSegmentLength.

The MaxRecvDataSegmentLength key is applicable only for iSCSI control-type PDUs.

### 6.3 RDMAExtensions

Use: LO (leading only)

Senders: Initiator and Target

Scope: SW (session-wide)

RDMAExtensions=<boolean-value>

Irrelevant when: SessionType=Discovery

Default is No

Result function is AND

This key is used by the initiator and the target to negotiate the support for iSER-assisted mode. To enable the use of iSER-assisted mode, both the initiator and the target MUST exchange RDMAExtensions=Yes. iSER-assisted mode MUST NOT be used if either the initiator or the target offers RDMAExtensions=No.

An iSER-enabled node is not required to initiate the RDMAExtensions key exchange if it prefers to operate in the Traditional iSCSI mode. However, if the RDMAExtensions key is to be negotiated, an initiator MUST offer the key in the first Login Request PDU in the LoginOperationalNegotiation stage of the leading connection, and a target MUST offer the key in the first Login Response PDU with which it is allowed to do so (i.e., the first Login Response PDU issued after the first Login Request PDU with the C bit set to 0) in the LoginOperationalNegotiation stage of the leading connection. In response to the offered key=value pair of RDMAExtensions=yes, an initiator MUST respond in the next Login Request PDU with which it is allowed to do so, and a target MUST respond in the next Login Response PDU with which it is allowed to do so.

Negotiating the RDMAExtensions key first enables a node to negotiate the optimal value for other keys. Certain iSCSI keys such as MaxBurstLength, MaxOutstandingR2T, ErrorRecoveryLevel, InitialR2T, ImmediateData, etc., may be negotiated differently depending on

#### 6.4 TargetRecvDataSegmentLength

Use: IO (Initialize only)

Senders: Initiator and Target

Scope: CO (connection-only)

Irrelevant when: RDMAExtensions=No

TargetRecvDataSegmentLength=<numerical-value-512-to-(2\*\*24-1)>

Default is 8192 bytes

Result function is minimum

This key is relevant only for the iSCSI connection of an iSCSI session if RDMAExtensions=Yes was negotiated on the leading connection of the session. It is used by the initiator and the target to negotiate the maximum size of the data segment that an initiator may send to the target in an iSCSI control-type PDU in the full feature phase. For SCSI Command PDUs and SCSI Data-out PDUs containing non-immediate unsolicited data to be sent by the initiator, the initiator MUST send all non-Final PDUs with a data segment size of exactly TargetRecvDataSegmentLength whenever the PDUs constitute a data sequence whose size is larger than TargetRecvDataSegmentLength.

#### 6.5 InitiatorRecvDataSegmentLength

Use: IO (Initialize only)

Senders: Initiator and Target

Scope: CO (connection-only)

Irrelevant when: RDMAExtensions=No

InitiatorRecvDataSegmentLength=<numerical-value-512-to-(2\*\*24-1)>

Default is 8192 bytes

Result function is minimum

This key is relevant only for the iSCSI connection of an iSCSI session if RDMAExtensions=Yes was negotiated on the leading connection of the session. It is used by the initiator and the target to negotiate the maximum size of the data segment that a target may send to the initiator in an iSCSI control-type PDU in the full feature phase.

## 6.6 OFMarker and IFMarker

Irrelevant when: RDMAExtensions=Yes

Negotiations resulting in RDMAExtensions=Yes for a session implies OFMarker=No and IFMarker=No for all connections in that session and overrides both the default and an explicit setting.

## 6.7 MaxOutstandingUnexpectedPDUs

Use: LO (leading only), Declarative

Senders: Initiator and Target

Scope: SW (session-wide)

Irrelevant when: RDMAExtensions=No

MaxOutstandingUnexpectedPDUs=<numerical-value-from-2-to-(2\*\*32-1) | 0>

Default is 4

This key is used by the initiator and the target to declare the maximum number of outstanding "unexpected" iSCSI control-type PDUs that it can receive in the full feature phase. It is intended to allow the receiving side to determine the amount of buffer resources needed beyond the normal flow control mechanism available in iSCSI. An initiator or target should select a value such that it would not impose an unnecessary constraint on the iSCSI Layer under normal circumstances. The value of 0 is defined to indicate that the declarer has no limit on the maximum number of outstanding "unexpected" iSCSI control-type PDUs that it can receive. See sections 8.1.1 and 8.1.2 for the usage of this key. Note that iSER Hello and HelloReply Messages are not iSCSI control-type PDUs and are not affected by this key.

## 6.8 MaxAHSLength

Use: LO (leading only), Declarative

Scope: SW (session-wide)

Irrelevant when: RDMAExtensions=No

MaxAHSLength=<numerical-value-from-2-to-(2\*\*32-1) | 0>

Default is 256

This key is used by the initiator and target to declare the maximum size of AHS in an iSCSI control-type PDU that it can receive in the full feature phase. It is intended to allow the receiving side to determine the amount of resources needed for receive buffering. An initiator or target should select a value such that it would not impose an unnecessary constraint on the iSCSI Layer under normal circumstances. The value of 0 is defined to indicate that the declarer has no limit on the maximum size of AHS in iSCSI control-type PDUs that it can receive.

#### 6.9 WriteAddressForSolicitedDataOnly

Use: LO (leading only), Declarative

Senders: Initiator

Scope: SW (session-wide)

RDMAExtensions=<boolean-value>

Irrelevant when: RDMAExtensions=No

Default is No

This key is used by the initiator to declare to the target the origin of the Write Virtual Address used in the iSER header of an iSCSI control-type PDU. When set to No, the Write Virtual Address points to an I/O buffer that contains all the write data, including both the unsolicited and solicited data. When set to Yes, the Write Virtual Address points to an I/O buffer that only contains the solicited data.

#### 6.10 iSERHelloRequired

Use: LO (leading only), Declarative

Senders: Initiator

RDMAExtensions=<boolean-value>

Irrelevant when: RDMAExtensions=No

Default is No

This key is relevant only for the iSCSI connection of an iSCSI session if RDMAExtensions=Yes was negotiated on the leading connection of the session. It is used by the initiator to declare to the target if the iSER Hello Exchange is required. When set to Yes, the iSER layers MUST perform the iSER Hello Exchange as described in 5.1.3. When set to No, the iSER layers MUST NOT perform the iSER Hello Exchange.

## 7 iSCSI PDU Considerations

When a connection is in the iSER-assisted mode, two types of message transfers are allowed between the iSCSI Layer at the initiator and the iSCSI Layer at the target. These are known as the iSCSI data-type PDUs and the iSCSI control-type PDUs and these terms are described in the following sections.

## 7.1 iSCSI Data-Type PDU

An iSCSI data-type PDU is defined as an iSCSI PDU that causes data transfer, transparent to the remote iSCSI layer, to take place between the peer iSCSI nodes in the full feature phase of an iSCSI/iSER connection. An iSCSI data-type PDU, when requested for transmission by the iSCSI Layer in the sending node, results in the data being transferred without the participation of the iSCSI Layers at the sending and the receiving nodes. This is due to the fact that the PDU itself is not delivered as-is to the iSCSI Layer in the receiving node. Instead, the data transfer operations are transformed into the appropriate RDMA operations which are handled by the RDMA-Capable Controller. The set of iSCSI data-type PDUs consists of SCSI Data-in PDUs and R2T PDUs.

If the invocation of the Operational Primitive by the iSCSI Layer to request the iSER Layer to process an iSCSI data-type PDU is qualified with `Notify_Enable` set, then upon completing the RDMA operation, the iSER Layer at the target MUST notify the iSCSI Layer at the target by invoking the `Data_Completion_Notify` Operational Primitive qualified with `ITT` and `SN`. There is no data completion notification at the initiator since the RDMA operations are completely handled by the RDMA-Capable Controller at the initiator and the iSER Layer at the initiator is not involved with the data transfer associated with iSCSI data-type PDUs.

If the invocation of the Operational Primitive by the iSCSI Layer to request the iSER Layer to process an iSCSI data-type PDU is qualified with `Notify_Enable` cleared, then upon completing the RDMA operation, the iSER Layer at the target MUST NOT notify the iSCSI Layer at the target and MUST NOT invoke the `Data_Completion_Notify` Operational Primitive.

If an operation associated with an iSCSI data-type PDU fails for any reason, the contents of the Data Sink buffers associated with the operation are considered indeterminate.

## 7.2 iSCSI Control-Type PDU

Any iSCSI PDU that is not an iSCSI data-type PDU and also not a SCSI Data-out PDU carrying solicited data is defined as an iSCSI control-type PDU. The iSCSI Layer invokes the Send\_Control Operational Primitive to request the iSER Layer to process an iSCSI control-type PDU. iSCSI control-type PDUs are transferred using Send Message Types of RCaP. Specifically, it is to be noted that SCSI Data-Out PDUs carrying unsolicited data are defined as iSCSI control-type PDUs. See section 7.3.4 on the treatment of SCSI Data-out PDUs.

When the iSER Layer receives an iSCSI control-type PDU, it MUST notify the iSCSI Layer by invoking the Control\_Notify Operational Primitive qualified with the iSCSI control-type PDU.

## 7.3 iSCSI PDUs

This section describes the handling of each of the iSCSI PDU types by the iSER Layer. The iSCSI Layer requests the iSER Layer to process the iSCSI PDU by invoking the appropriate Operational Primitive. A Connection\_Handle MUST qualify each of these invocations. In addition, BHS and the optional AHS of the iSCSI PDU as defined in [RFC3720] MUST qualify each of the invocations. The qualifying Connection\_Handle, the BHS and the AHS are not explicitly listed in the subsequent sections.

### 7.3.1 SCSI Command

Type: control-type PDU

PDU-specific qualifiers (for SCSI Write or bidirectional command): ImmediateDataSize, UnsolicitedDataSize, DataDescriptorOut

PDU-specific qualifiers (for SCSI Read or bidirectional command): DataDescriptorIn

The iSER Layer at the initiator MUST send the SCSI command in a SendSE Message to the target.

For a SCSI Write or bidirectional command, the iSCSI Layer at the initiator MUST invoke the Send\_Control Operational Primitive as follows:

- \* If there is immediate data to be transferred for the SCSI write or bidirectional command, the qualifier ImmediateDataSize MUST be used to define the number of bytes of immediate unsolicited data

to be sent with the write or bidirectional command, and the qualifier `DataDescriptorOut` MUST be used to define the initiator's I/O Buffer containing the SCSI Write data.

- \* If there is unsolicited data to be transferred for the SCSI Write or bidirectional command, the qualifier `UnsolicitedDataSize` MUST be used to define the number of bytes of immediate and non-immediate unsolicited data for the command. The iSCSI Layer will issue one or more SCSI Data-out PDUs for the non-immediate unsolicited data. See Section 7.3.4 on SCSI Data-out.
- \* If there is solicited data to be transferred for the SCSI Write or bidirectional command, as indicated by the Expected Data Transfer Length in the SCSI Command PDU exceeding the value of `UnsolicitedDataSize`, the iSER Layer at the initiator MUST do the following:
  - a. It MUST allocate a Write STag for the I/O Buffer defined by the qualifier `DataDescriptorOut`. `DataDescriptorOut` describes the I/O buffer starting with the immediate unsolicited data (if any), followed by the non-immediate unsolicited data (if any) and solicited data. This means that the `BufferOffset` for the SCSI Data-out for this command is equal to the TO. This implies zero TO for this STag points to the beginning of this I/O Buffer.
  - b. It MUST establish a Local Mapping that associates the Initiator Task Tag (ITT) to the Write STag.
  - c. It MUST Advertise the Write STag to the target by sending it as the Write STag in the iSER header of the iSER Message (the payload of the SendSE Message of RCaP) containing the SCSI Write or bidirectional command PDU. See section 9.2 on iSER Header Format for iSCSI Control-Type PDU.

For a SCSI Read or bidirectional command, the iSCSI Layer at the initiator MUST invoke the `Send_Control Operational Primitive` qualified with `DataDescriptorIn` which defines the initiator's I/O Buffer for receiving the SCSI Read data. The iSER Layer at the initiator MUST do the following:

- a. It MUST allocate a Read STag for the I/O Buffer.
- b. It MUST establish a Local Mapping that associates the Initiator Task Tag (ITT) to the Read STag.



- c. It MUST Advertise the Read STag to the target by sending it as the Read STag in the iSER header of the iSER Message (the payload of the SendSE Message of RCaP) containing the SCSI Read or bidirectional command PDU. See section 9.2 on iSER Header Format for iSCSI Control-Type PDU.

If the amount of unsolicited data to be transferred in a SCSI Command exceeds TargetRecvDataSegmentLength, then the iSCSI Layer at the initiator MUST segment the data into multiple iSCSI control-type PDUs, with the data segment length in all PDUs generated except the last one having exactly the size TargetRecvDataSegmentLength. The data segment length of the last iSCSI control-type PDU carrying the unsolicited data can be up to TargetRecvDataSegmentLength.

When the iSER Layer at the target receives the SCSI Command, it MUST establish a Remote Mapping that associates the ITT to the Advertised Write STag and the Read STag if present in the iSER header. The Write STag is used by the iSER Layer at the target in handling the data transfer associated with the R2T PDU(s) as described in section 7.3.6. The Read STag is used in handling the SCSI Data-in PDU(s) from the iSCSI Layer at the target as described in section 7.3.5.

### 7.3.2 SCSI Response

Type: control-type PDU

PDU-specific qualifiers: DataDescriptorStatus

The iSCSI Layer at the target MUST invoke the Send\_Control Operational Primitive qualified with DataDescriptorStatus which defines the buffer containing the sense and response information. The iSCSI Layer at the target MUST always return the SCSI status for a SCSI command in a separate SCSI Response PDU. "Phase collapse" for transferring SCSI status in a SCSI Data-in PDU MUST NOT be used. The iSER Layer at the target sends the SCSI Response PDU according to the following rules:

- \* If no STags were Advertised by the initiator in the iSER Message containing the SCSI command PDU, then the iSER Layer at the target MUST send a SendSE Message containing the SCSI Response PDU.
- \* If the initiator Advertised a Read STag in the iSER Message containing the SCSI Command PDU, then the iSER Layer at the target MUST send a SendInvSE Message containing the SCSI Response PDU. The header of the SendInvSE Message MUST carry the Read STag to be invalidated at the initiator.

- \* If the initiator Advertised only the Write STag in the iSER Message containing the SCSI command PDU, then the iSER Layer at the target MUST send a SendInvSE Message containing the SCSI Response PDU. The header of the SendInvSE Message MUST carry the Write STag to be invalidated at the initiator.

When the iSCSI Layer at the target invokes the Send\_Control Operational Primitive to send the SCSI Response PDU, the iSER Layer at the target MUST invalidate the Remote Mapping that associates the ITT to the Advertised STag(s) before transferring the SCSI Response PDU to the initiator.

Upon receiving the SendInvSE Message containing the SCSI Response PDU from the target, the RCaP layer at the initiator will invalidate the STag specified in the header. The iSER Layer at the initiator MUST ensure that the correct STag is invalidated. If both the Read and the Write STags were Advertised earlier by the initiator, then the iSER Layer at the initiator MUST explicitly invalidate the Write STag upon receiving the SendInvSE Message because the header of the SendInvSE Message can only carry one STag (in this case the Read STag) to be invalidated.

The iSER Layer at the initiator MUST ensure the invalidation of the STag(s) used in a command before notifying the iSCSI Layer at the initiator by invoking the Control\_Notify Operational Primitive qualified with the SCSI Response. This precludes the possibility of using the STag(s) after the completion of the command thereby causing data corruption.

When the iSER Layer at the initiator receives the SendSE or the SendInvSE Message containing the SCSI Response PDU, it SHOULD invalidate the Local Mapping that associates the ITT to the local STag(s). The iSER Layer MUST ensure that all local STag(s) associated with the ITT are invalidated before notifying the iSCSI Layer of the SCSI Response PDU by invoking the Control\_Notify Operational Primitive qualified with the SCSI Response PDU.

### 7.3.3 Task Management Function Request/Response

Type: control-type PDU

PDU-specific qualifiers (for TMF Request): DataDescriptorOut, DataDescriptorIn

The iSER Layer MUST use a SendSE Message to send the Task Management Function Request/Response PDU.

For the Task Management Function Request with the TASK REASSIGN function, the iSER Layer at the initiator MUST do the following:

- \* It MUST use the ITT as specified in the Referenced Task Tag from the Task Management Function Request PDU to locate the existing STag(s), if any, in the Local Mapping(s) that associates the ITT to the local STag(s).
- \* It MUST invalidate the existing STag(s), if any, and the Local Mapping(s) that associates the ITT to the local STag(s).
- \* It MUST allocate a Read STag for the I/O Buffer as defined by the qualifier DataDescriptorIn if the Send\_Control Operational Primitive invocation is qualified with DataDescriptorIn.
- \* It MUST allocate a Write STag for the I/O Buffer as defined by the qualifier DataDescriptorOut if the Send\_Control Operational Primitive invocation is qualified with DataDescriptorOut.
- \* If STags are allocated, it MUST establish new Local Mapping(s) that associate the ITT to the allocated STag(s).
- \* It MUST Advertise the STags, if allocated, to the target in the iSER header of the SendSE Message carrying the iSCSI PDU, as described in section 9.2.

For the Task Management Function Request with the TASK REASSIGN function for a SCSI Read or bidirectional command, the iSCSI Layer at the initiator MUST set ExpDataSN to 0 since the data transfer and acknowledgements happen transparently to the iSCSI Layer at the initiator. This provides the flexibility to the iSCSI Layer at the target to request transmission of only the unacknowledged data as specified in [RFC3720].

When the iSER Layer at the target receives the Task Management Function Request with the TASK REASSIGN function, it MUST do the following:

- \* It MUST use the ITT as specified in the Referenced Task Tag from the Task Management Function Request PDU to locate the mappings that associate the ITT to the Advertised STag(s) and the local STag(s), if any.
- \* It MUST invalidate the local STag(s), if any, associated with the ITT.

- \* It MUST replace the Advertised STag(s) in the Remote Mapping that associates the ITT to the Advertised STag(s) with the Write STag and the Read STag if present in the iSER header. The Write STag is used in the handling of the R2T PDU(s) from the iSCSI Layer at the target as described in section 7.3.6. The Read STag is used in the handling of the SCSI Data-in PDU(s) from the iSCSI Layer at the target as described in section 7.3.5.

#### 7.3.4 SCSI Data-out

Type: control-type PDU

PDU-specific qualifiers: DataDescriptorOut

The iSCSI Layer at the initiator MUST invoke the Send\_Control Operational Primitive qualified with DataDescriptorOut which defines the initiator's I/O Buffer containing unsolicited SCSI Write data.

If the amount of unsolicited data to be transferred as SCSI Data-out exceeds TargetRecvDataSegmentLength, then the iSCSI Layer at the initiator MUST segment the data into multiple iSCSI control-type PDUs, with the DataSegmentLength having the value of TargetRecvDataSegmentLength in all PDUs generated except the last one. The DataSegmentLength of the last iSCSI control-type PDU carrying the unsolicited data can be up to TargetRecvDataSegmentLength. The iSCSI Layer at the target MUST perform the reassembly function for the unsolicited data.

For unsolicited data, if the F bit is set to 0 in a SCSI Data-out PDU, the iSER Layer at the initiator MUST use a Send Message to send the SCSI Data-out PDU. If the F bit is set to 1, the iSER Layer at the initiator MUST use a SendSE Message to send the SCSI Data-out PDU.

Note that for solicited data, the SCSI Data-out PDUs are not used since R2T PDUs are not delivered to the iSCSI layer at the initiator; instead R2T PDUs are transformed by the iSER layer at the target into RDMA Read operations. (See section 7.3.6.)

#### 7.3.5 SCSI Data-in

Type: data-type PDU

PDU-specific qualifiers: DataDescriptorIn

When the iSCSI Layer at the target is ready to return the SCSI Read data to the initiator, it MUST invoke the Put\_Data Operational

Primitive qualified with DataDescriptorIn which defines the SCSI Data-in buffer. See section 7.1 on the general requirement on the handling of iSCSI data-type PDUs. SCSI Data-in PDU(s) are used in SCSI Read data transfer as described in section 9.5.2.

The iSER Layer at the target MUST do the following for each invocation of the Put\_Data Operational Primitive:

1. It MUST use the ITT in the SCSI Data-in PDU to locate the remote Read STag in the Remote Mapping that associates the ITT to Advertised STag(s). The Remote Mapping was established earlier by the iSER Layer at the target when the SCSI Read Command was received from the initiator.
2. It MUST generate and send an RDMA Write Message containing the read data to the initiator.
  - a. It MUST use the remote Read STag as the Data Sink STag of the RDMA Write Message.
  - b. It MUST use the Buffer Offset from the SCSI Data-in PDU as the Data Sink Tagged Offset of the RDMA Write Message.
  - c. It MUST use DataSegmentLength from the SCSI Data-in PDU to determine the amount of data to be sent in the RDMA Write Message.
3. It MUST associate DataSN and ITT from the SCSI Data-in PDU with the RDMA Write operation. If the Put\_Data Operational Primitive invocation was qualified with Notify\_Enable set, then when the iSER Layer at the target receives a completion from the RCaP layer for the RDMA Write Message, the iSER Layer at the target MUST notify the iSCSI Layer by invoking the Data\_Completion\_Notify Operational Primitive qualified with DataSN and ITT. Conversely, if the Put\_Data Operational Primitive invocation was qualified with Notify\_Enable cleared, then the iSER Layer at the target MUST NOT notify the iSCSI Layer on completion and MUST NOT invoke the Data\_Completion\_Notify Operational Primitive.

When the A-bit is set to 1 in the SCSI Data-in PDU, the iSER Layer at the target MUST notify the iSCSI Layer at the target when the data transfer is complete at the initiator. To perform this additional function, the iSER Layer at the target can take advantage of the operational ErrorRecoveryLevel if previously disclosed by the iSCSI Layer via an earlier invocation of the Notice\_Key\_Values Operational Primitive. There are two approaches that can be taken:

1. If the iSER Layer at the target knows that the operational ErrorRecoveryLevel is 2, or if the iSER Layer at the target does not know the operational ErrorRecoveryLevel, then the iSER Layer at the target MUST issue a zero-length RDMA Read Request Message following the RDMA Write Message. When the iSER Layer at the target receives a completion for the RDMA Read Request Message from the RCaP layer, implying that the RDMA-Capable Controller at the initiator has completed processing the RDMA Write Message due to the completion ordering semantics of RCaP, the iSER Layer at the target MUST notify the iSCSI Layer at the target by invoking the Data\_Ack\_Notify Operational Primitive qualified with ITT and DataSN (see section 3.2.3).
2. If the iSER Layer at the target knows that the operational ErrorRecoveryLevel is 1, then the iSER Layer at the target MUST do one of the following:
  - a. It MUST notify the iSCSI Layer at the target by invoking the Data\_Ack\_Notify Operational Primitive qualified with ITT and DataSN (see section 3.2.3) when it receives the local completion from the RCaP layer for the RDMA Write Message. This is allowed since digest errors do not occur in iSER (see section 10.1.4.2) and a CRC error will cause the connection to be terminated and the task to be terminated anyway. The local RDMA Write completion from the RCaP layer guarantees that the RCaP layer will not access the I/O Buffer again to transfer the data associated with that RDMA Write operation.
  - b. Alternatively, it MUST use the same procedure for handling the data transfer completion at the initiator as for ErrorRecoveryLevel 2.

It should be noted that the iSCSI Layer at the target cannot set the A-bit to 1 if the ErrorRecoveryLevel=0.

SCSI status MUST always be returned in a separate SCSI Response PDU. The S bit in the SCSI Data-in PDU MUST always be set to 0. There MUST NOT be a "phase collapse" in the SCSI Data-in PDU.

Since the RDMA Write Message only transfers the data portion of the SCSI Data-in PDU but not the control information in the header, such as ExpCmdSN, if timely updates of such information is crucial, the iSCSI Layer at the initiator MAY issue NOP-Out PDUs to request the iSCSI Layer at the target to respond with the information using NOP-In PDUs.

Type: data-type PDU

PDU-specific qualifiers: DataDescriptorOut

In order to send an R2T PDU, the iSCSI Layer at the target MUST invoke the Get\_Data Operational Primitive qualified with DataDescriptorOut which defines the I/O Buffer for receiving the SCSI Write data from the initiator. See section 7.1 on the general requirements on the handling of iSCSI data-type PDUs.

The iSER Layer at the target MUST do the following for each invocation of the Get\_Data Operational Primitive:

1. It MUST ensure a valid local STag for the I/O Buffer and a valid Local Mapping that associates the Initiator Task Tag (ITT) to the local STag. This may involve allocating a valid local STag and establishing a Local Mapping.
2. It MUST use the ITT in the R2T to locate the remote Write STag in the Remote Mapping that associates the ITT to Advertised STag(s). The Remote Mapping was established earlier by the iSER Layer at the target when the iSER Message containing the Advertised Write STag and the SCSI Command PDU for a SCSI Write or bidirectional command was received from the initiator.
3. If the iSER-ORD value at the target is set to 0, the iSER Layer at the target MUST terminate the connection and free up the resources associated with the connection (as described in 5.2.3) if it received the R2T PDU from the iSCSI Layer at the target. Upon termination of the connection, the iSER Layer at the target MUST notify the iSCSI Layer at the target by invoking the Connection\_Terminate\_Notify Operational Primitive.
4. If the iSER-ORD value at the target is set to greater than 0, the iSER Layer at the target MUST transform the R2T PDU into an RDMA Read Request Message. While transforming the R2T PDU, the iSER Layer at the target MUST ensure that the number of outstanding RDMA Read Request Messages does not exceed iSER-ORD value. To transform the R2T PDU, the iSER Layer at the target:
  - a. MUST derive the local STag and local Tagged Offset from the DataDescriptorOut that qualified the Get\_Data invocation.
  - b. MUST use the local STag as the Data Sink STag of the RDMA Read Request Message.

- c. MUST use the local Tagged Offset as the Data Sink Tagged Offset of the RDMA Read Request Message.
  - d. MUST use the Desired Data Transfer Length from the R2T PDU as the RDMA Read Message Size of the RDMA Read Request Message.
  - e. MUST use the remote Write STag as the Data Source STag of the RDMA Read Request Message.
  - f. MUST use the Buffer Offset from the R2T PDU as the Data Source Tagged Offset of the RDMA Read Request Message.
5. It MUST associate R2TSN and ITT from the R2T PDU with the RDMA Read operation. If the Get\_Data Operational Primitive invocation was qualified with Notify\_Enable set, then when the iSER Layer at the target receives a completion from the RCaP layer for the RDMA Read operation, the iSER Layer at the target MUST notify the iSCSI Layer by invoking the Data\_Completion\_Notify Operational Primitive qualified with R2TSN and ITT. Conversely, if the Get\_Data Operational Primitive invocation was qualified with Notify\_Enable cleared, then the iSER Layer at the target MUST NOT notify the iSCSI Layer on completion and MUST NOT invoke the Data\_Completion\_Notify Operational Primitive.

When the RCaP layer at the initiator receives a valid RDMA Read Request Message, it will return an RDMA Read Response Message containing the solicited write data to the target. When the RCaP layer at target receives the RDMA Read Response Message from the initiator, it will place the solicited data in the I/O Buffer referenced by the Data Sink STag in the RDMA Read Response Message.

Since the RDMA Read Request Message from the target does not transfer the control information in the R2T PDU such as ExpCmdSN, if timely updates of such information is crucial, the iSCSI Layer at the initiator MAY issue NOP-Out PDUs to request the iSCSI Layer at the target to respond with the information using NOP-In PDUs.

Similarly, since the RDMA Read Response Message from the initiator only transfers the data but not the control information normally found in the SCSI Data-out PDU, such as ExpStatSN, if timely updates of such information is crucial, the iSCSI Layer at the target MAY issue NOP-In PDUs to request the iSCSI Layer at the initiator to respond with the information using NOP-Out PDUs.



### 7.3.7 Asynchronous Message

Type: control-type PDU

PDU-specific qualifiers: DataDescriptorSense

The iSCSI Layer MUST invoke the Send\_Control Operational Primitive qualified with DataDescriptorSense which defines the buffer containing the sense and iSCSI Event information. The iSER Layer MUST use a SendSE Message to send the Asynchronous Message PDU.

### 7.3.8 Text Request & Text Response

Type: control-type PDU

PDU-specific qualifiers: DataDescriptorTextOut (for Text Request), DataDescriptorIn (for Text Response)

The iSCSI Layer MUST invoke the Send\_Control Operational Primitive qualified with DataDescriptorTextOut (or DataDescriptorIn) which defines the Text Request (or Text Response) buffer. The iSER Layer MUST use SendSE Messages to send the Text Request (or Text Response PDUs).

### 7.3.9 Login Request & Login Response

During the login negotiation, the iSCSI Layer interacts with the transport layer directly and the iSER Layer is not involved. See section 5.1 on iSCSI/iSER Connection Setup. If the underlying transport is TCP, the Login Request PDUs and the Login Response PDUs are exchanged when the connection between the initiator and the target is still in the byte stream mode.

The iSCSI Layer MUST not send a Login Request (or a Login Response) PDU during the full feature phase. A Login Request (or a Login Response) PDU, if used, MUST be treated as an iSCSI protocol error. The iSER Layer MAY reject such a PDU from the iSCSI Layer with an appropriate error code. If a Login Request PDU is received by the iSCSI Layer at the target, it MUST respond with a Reject PDU with a reason code of "protocol error".

### 7.3.10 Logout Request & Logout Response

Type: control-type PDU

PDU-specific qualifiers: None

The iSER Layer MUST use a SendSE Message to send the Logout Request or Logout Response PDU. Section 5.2.1 and 5.2.2 describe the handling of the Logout Request and the Logout Response at the initiator and the target and the interactions between the initiator and the target to terminate a connection.

#### 7.3.11 SNACK Request

Since HeaderDigest and DataDigest must be negotiated to "None", there are no digest errors when the connection is in iSER-assisted mode. Also since RCaP delivers all messages in the order they were sent, there are no sequence errors when the connection is in iSER-assisted mode. Therefore the iSCSI Layer MUST NOT send SNACK Request PDUs. A SNACK Request PDU, if used, MUST be treated as an iSCSI protocol error. The iSER Layer MAY reject such a PDU from the iSCSI Layer with an appropriate error code. If a SNACK Request PDU is received by the iSCSI Layer at the target, it MUST respond with a Reject PDU with a reason code of "protocol error".

#### 7.3.12 Reject

Type: control-type PDU

PDU-specific qualifiers: DataDescriptorReject

The iSCSI Layer MUST invoke the Send\_Control Operational Primitive qualified with DataDescriptorReject which defines the Reject buffer. The iSER Layer MUST use a SendSE Message to send the Reject PDU.

#### 7.3.13 NOP-Out & NOP-In

Type: control-type PDU

PDU-specific qualifiers: DataDescriptorNOPOut (for NOP-Out),  
DataDescriptorNOPIn (for NOP-In)

The iSCSI Layer MUST invoke the Send\_Control Operational Primitive qualified with DataDescriptorNOPOut (or DataDescriptorNOPIn) which defines the Ping (or Return Ping) data buffer. The iSER Layer MUST use SendSE Messages to send the NOP-Out (or NOP-In) PDU.

## 8 Flow Control and STag Management

## 8.1 Flow Control for RDMA Send Message Types

Send Message Types in RCaP are used by the iSER Layer to transfer iSCSI control-type PDUs. Each Send Message Type in RCaP consumes an Untagged Buffer at the Data Sink. However, neither the RCaP layer nor the iSER Layer provides an explicit flow control mechanism for the Send Message Types. Therefore, the iSER Layer SHOULD provision enough Untagged buffers for handling incoming Send Message Types to prevent buffer exhaustion at the RCaP layer. If buffer exhaustion occurs, it may result in the termination of the connection.

An implementation may choose to satisfy the buffer requirement by using a common buffer pool shared across multiple connections, with usage limits on a per connection basis and usage limits on the buffer pool itself. In such an implementation, exceeding the buffer usage limit for a connection or the buffer pool itself may trigger interventions from the iSER Layer to replenish the buffer pool and/or to isolate the connection causing the problem.

iSER also provides the MaxOutstandingUnexpectedPDUs key to be used by the initiator and the target to declare the maximum number of outstanding "unexpected" control-type PDUs that it can receive. It is intended to allow the receiving side to determine the amount of buffer resources needed beyond the normal flow control mechanism available in iSCSI.

The buffer resources required at both the initiator and the target as a result of control-type PDUs sent by the initiator is described in section 8.1.1. The buffer resources required at both the initiator and target as a result of control-type PDUs sent by the target is described in section 8.1.2.

## 8.1.1 Flow Control for Control-Type PDUs from the Initiator

The control-type PDUs that can be sent by an initiator to a target can be grouped into the following categories:

1. **Regulated:** Control-type PDUs in this category are regulated by the iSCSI CmdSN window mechanism and the immediate flag is not set.
2. **Unregulated but Expected:** Control-type PDUs in this category are not regulated by the iSCSI CmdSN window mechanism but are expected by the target.

3. Unregulated and Unexpected: Control-type PDUs in this category are not regulated by the iSCSI CmdSN window mechanism and are "unexpected" by the target.

#### 8.1.1.1 Control-Type PDUs from the Initiator in the Regulated Category

Control-type PDUs that can be sent by the initiator in this category are regulated by the iSCSI CmdSN window mechanism and the immediate flag is not set.

The queuing capacity required of the iSCSI layer at the target is described in section 3.2.2.1 of [RFC3720]. For each of the control-type PDUs that can be sent by the initiator in this category, the initiator MUST provision for the buffer resources required for the corresponding control-type PDU sent as a response from the target. The following is a list of the PDUs that can be sent by the initiator and the PDUs that are sent by the target in response:

- a. When an initiator sends a SCSI Command PDU, it expects a SCSI Response PDU from the target.
- b. When the initiator sends a Task Management Function Request PDU, it expects a Task Management Function Response PDU from the target.
- c. When the initiator sends a Text Request PDU, it expects a Text Response PDU from the target.
- d. When the initiator sends a Logout Request PDU, it expects a Logout Response PDU from the target.
- e. When the initiator sends a NOP-Out PDU as a ping request with ITT != 0xffffffff and TTT = 0xffffffff, it expects a NOP-In PDU from the target with the same ITT and TTT as in the ping request.

The response from the target for any of the PDUs enumerated here may alternatively be in the form of a Reject PDU sent instead before the task is active, as described in section 6.3 of [RFC3720].

#### 8.1.1.2 Control-Type PDUs from the Initiator in the Unregulated but Expected Category

For the control-type PDUs in the Unregulated but Expected category, the amount of buffering resources required at the target can be predetermined. The following is a list of the PDUs in this category:

- a. SCSI Data-out PDUs are used by the initiator to send unsolicited data. The amount of buffer resources required by the target can be determined using FirstBurstLength. Note that SCSI Data-out PDUs are not used for solicited data since the R2T PDU which is used for solicitation is transformed into RDMA Read operations by the iSER layer at the target. See section 7.3.4.
- b. A NOP-Out PDU with TTT != 0xffffffff is sent as a ping response by the initiator to the NOP-In PDU sent as a ping request by the target.

#### 8.1.1.3 Control-Type PDUs from the Initiator in the Unregulated and Unexpected Category

PDUs in the Unregulated and Unexpected category are PDUs with the immediate flag set. The number of PDUs in this category which can be sent by an initiator is controlled by the value of MaxOutstandingUnexpectedPDUs declared by the target. (See section 6.7.) After a PDU in this category is sent by the initiator, it is outstanding until it is retired. At any time, the number of outstanding unexpected PDUs MUST not exceed the value of MaxOutstandingUnexpectedPDUs declared by the target.

The target uses the value of MaxOutstandingUnexpectedPDUs that it declared to determine the amount of buffer resources required for control-type PDUs in this category that can be sent by an initiator. For the initiator, for each of the control-type PDUs that can be sent in this category, the initiator MUST provision for the buffer resources if required for the corresponding control-type PDU that can be sent as a response from the target.

An outstanding PDU in this category is retired as follows. If the CmdSN of the PDU sent by the initiator in this category is x, the PDU is outstanding until the initiator sends a non-immediate control-type PDU on the same connection with CmdSN = y (where y is at least x) and the target responds with a control-type PDU on any connection where ExpCmdSN is at least y+1.

When the number of outstanding unexpected control-type PDUs equals MaxOutstandingUnexpectedPDUs, the iSCSI Layer at the initiator MUST NOT generate any unexpected PDUs which otherwise it would have generated, even if it is intended for immediate delivery.

## 8.1.2 Flow Control for Control-Type PDUs from the Target

Control-type PDUs that can be sent by a target and are expected by the initiator are listed in the Regulated category. (See section 8.1.1.1.)

For the control-type PDUs that can be sent by a target and are unexpected by the initiator, the number is controlled by `MaxOutstandingUnexpectedPDUs` declared by the initiator. (See section 6.7.) After a PDU in this category is sent by a target, it is outstanding until it is retired. At any time, the number of outstanding unexpected PDUs MUST not exceed the value of `MaxOutstandingUnexpectedPDUs` declared by the initiator. The initiator uses the value of `MaxOutstandingUnexpectedPDUs` that it declared to determine the amount of buffer resources required for control-type PDUs in this category that can be sent by a target. The following is a list of the PDUs in this category and the conditions for retiring the outstanding PDU:

- a. For an Asynchronous Message PDU with `StatSN = x`, the PDU is outstanding until the initiator sends a control-type PDU with `ExpStatSN` set to at least `x+1`.
- b. For a Reject PDU with `StatSN = x` which is sent after a task is active, the PDU is outstanding until the initiator sends a control-type PDU with `ExpStatSN` set to at least `x+1`.
- c. For a NOP-In PDU with `ITT = 0xffffffff` and `StatSN = x`, the PDU is outstanding until the initiator responds with a control-type PDU on the same connection where `ExpStatSN` is at least `x+1`. But if the NOP-In PDU is sent as a ping request with `TTT != 0xffffffff`, the PDU can also be retired when the initiator sends a NOP-Out PDU with the same `ITT` and `TTT` as in the ping request. Note that when a target sends a NOP-In PDU as a ping request, it must provision a buffer for the NOP-Out PDU sent as a ping response from the initiator.

When the number of outstanding unexpected control-type PDUs equals `MaxOutstandingUnexpectedPDUs`, the iSCSI Layer at the target MUST NOT generate any unexpected PDUs which otherwise it would have generated, even if its intent is to indicate an iSCSI error condition (e.g., Asynchronous Message, Reject). Task timeouts as in the initiator waiting for a command completion or other connection and session level exceptions will ensure that correct operational behavior will result in these cases despite not generating the PDU. This rule overrides any other requirements elsewhere which require that a Reject PDU MUST be sent.

(Implementation note: SCSI task timeout and recovery can be a lengthy process and hence SHOULD be avoided by proper provisioning of resources.)

(Implementation note: To ensure that the initiator has a means to inform the target that outstanding PDUs have been retired, the target should reserve the last unexpected control-type PDU allowable by the value of MaxOutstandingUnexpectedPDUs declared by the initiator for sending a NOP-In ping request with TTT != 0xffffffff to allow the initiator to return the NOP-Out ping response with the current ExpStatSN.)

## 8.2 Flow Control for RDMA Read Resources

If iSERHelloRequired is negotiated to "Yes", then the total number of RDMA Read operations that can be active simultaneously on an iSCSI/iSER connection depends on the amount of resources allocated as declared in the iSER Hello exchange described in section 5.1.3. Exceeding the number of RDMA Read operations allowed on a connection will result in the connection being terminated by the RCaP layer. The iSER Layer at the target maintains the iSER-ORD to keep track of the maximum number of RDMA Read Requests that can be issued by the iSER Layer on a particular RCaP Stream.

During connection setup (see section 5.1), iSER-IRD is known at the initiator and iSER-ORD is known at the target after the iSER Layers at the initiator and the target have respectively allocated the connection resources necessary to support RCaP, as directed by the Allocate\_Connection\_Resources Operational Primitive from the iSCSI Layer before the end of the iSCSI Login Phase. In the full feature phase, if iSERHelloRequired is negotiated to "Yes", then the first message sent by the initiator is the iSER Hello Message (see section 9.3) which contains the value of iSER-IRD. In response to the iSER Hello Message, the target sends the iSER HelloReply Message (see section 9.4) which contains the value of iSER-ORD. The iSER Layer at both the initiator and the target MAY adjust (lower) the resources associated with iSER-IRD and iSER-ORD respectively to match the iSER-ORD value declared in the HelloReply Message. The iSER Layer at the target MUST flow control the RDMA Read Request Messages to not exceed the iSER-ORD value at the target.

## 8.3 StTag Management

An StTag, as defined in [RDMA], is an identifier of a Tagged Buffer used in an RDMA operation. The allocation and the subsequent invalidation of the StTags are specified in this document if the

STags are exposed on the wire by being Advertised in the iSER header or declared in the header of an RCaP Message.

### 8.3.1 Allocation of STags

When the iSCSI Layer at the initiator invokes the Send\_Control Operational Primitive to request the iSER Layer at the initiator to process a SCSI Command, zero, one, or two STags may be allocated by the iSER Layer. See section 7.3.1 for details. The number of STags allocated depends on whether the command is unidirectional or bidirectional and whether solicited write data transfer is involved or not.

When the iSCSI Layer at the initiator invokes the Send\_Control Operational Primitive to request the iSER Layer at the initiator to process a Task Management Function Request with the TASK REASSIGN function, besides allocating zero, one, or two STags, the iSER Layer MUST invalidate the existing STags, if any, associated with the ITT. See section 7.3.3 for details.

The iSER Layer at the target allocates a local Data Sink STag when the iSCSI Layer at the target invokes the Get\_Data Operational Primitive to request the iSER Layer to process an R2T PDU. See section 7.3.6 for details.

### 8.3.2 Invalidation of STags

The invalidation of the STags at the initiator at the completion of a unidirectional or bidirectional command when the associated SCSI Response PDU is sent by the target is described in section 7.3.2.

When a unidirectional or bidirectional command concludes without the associated SCSI Response PDU being sent by the target, the iSCSI Layer at the initiator MUST request the iSER Layer at the initiator to invalidate the STags by invoking the Deallocate\_Task\_Resources Operational Primitive qualified with ITT. In response, the iSER Layer at the initiator MUST locate the STag(s) (if any) in the Local Mapping that associates the ITT to the local STag(s). The iSER Layer at the initiator MUST invalidate the STag(s) (if any) and the Local Mapping.

For an RDMA Read operation used to realize a SCSI Write data transfer, the iSER Layer at the target SHOULD invalidate the Data Sink STag at the conclusion of the RDMA Read operation referencing the Data Sink STag (to permit the immediate reuse of buffer resources).



For an RDMA Write operation used to realize a SCSI Read data transfer, the Data Source STag at the target is not declared to the initiator and is not exposed on the wire. Invalidation of the STag is thus not specified.

When a unidirectional or bidirectional command concludes without the associated SCSI Response PDU being sent by the target, the iSCSI Layer at the target MUST request the iSER Layer at the target to invalidate the STags by invoking the Deallocate\_Task\_Resources Operational Primitive qualified with ITT. In response, the iSER Layer at the target MUST locate the local STag(s) (if any) in the Local Mapping that associates the ITT to the local STag(s). The iSER Layer at the target MUST invalidate the local STag(s) (if any) and the mapping.

## 9 iSER Control and Data Transfer

For iSCSI data-type PDUs (see section 7.1), the iSER Layer uses RDMA Read and RDMA Write operations to transfer the solicited data. For iSCSI control-type PDUs (see section 7.2), the iSER Layer uses Send Message Types of RCaP.

## 9.1 iSER Header Format

An iSER header MUST be present in every Send Message Type of RCaP. The iSER header is located in the first 12 bytes of the message payload of the Send Message Type of RCaP, as shown in Figure 2.

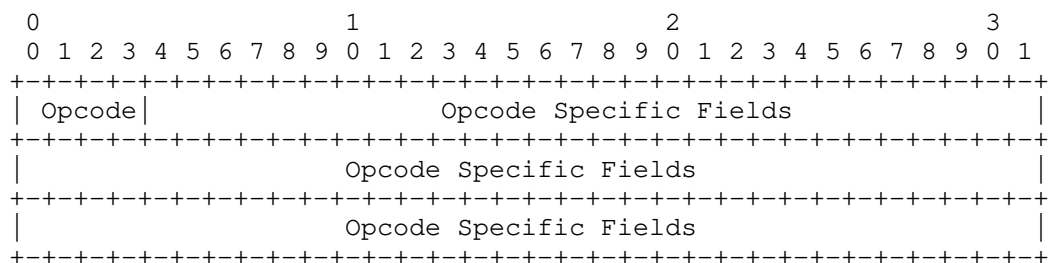


Figure 2 iSER Header Format

Opcode - Operation Code: 4 bits

The Opcode field identifies the type of iSER Messages:

0001b = iSCSI control-type PDU

0010b = iSER Hello Message

0011b = iSER HelloReply Message

All other opcodes are reserved.

## 9.2 iSER Header Format for iSCSI Control-Type PDU

The iSER Layer uses Send Message Types of RCaP to transfer iSCSI control-type PDUs (see section 7.2). The message payload of each of the Send Message Types of RCaP used for transferring an iSER Message contains an iSER Header followed by an iSCSI control-type PDU.

The iSER header in a Send Message Type of RCaP carrying an iSCSI control-type PDU MUST have the format as described in Figure 3.

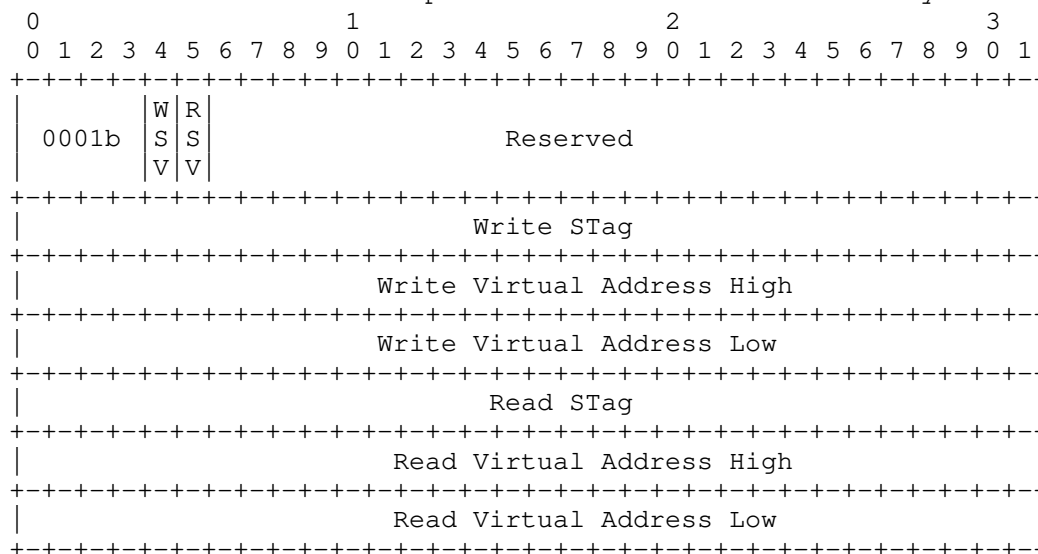


Figure 3 iSER Header Format for iSCSI Control-Type PDU

WSV - Write STag Valid flag: 1 bit

This flag indicates the validity of the Write STag field of the iSER Header. If set to one, the Write STag field in this iSER Header is valid. If set to zero, the Write STag field in this iSER Header MUST be ignored at the receiver. The Write STag Valid flag is set to one when there is solicited data to be transferred for a SCSI Write or bidirectional command, or when there are non-immediate unsolicited and solicited data to be transferred for the referenced task specified in a Task Management Function Request with the TASK REASSIGN function.

RSV - Read STag Valid flag: 1 bit

This flag indicates the validity of the Read STag field of the iSER Header. If set to one, the Read STag field in this iSER Header is valid. If set to zero, the Read STag field in this iSER Header MUST be ignored at the receiver. The Read STag Valid flag is set to one for a SCSI Read or bidirectional command, or a Task Management Function Request with the TASK REASSIGN function.

Write STag - Write Steering Tag: 32 bits

This field contains the Write STag when the Write STag Valid flag is set to one. For a SCSI Write or bidirectional command,

the Write STag is used to Advertise the initiator's I/O Buffer containing the solicited data. For a Task Management Function Request with the TASK REASSIGN function, the Write STag is used to Advertise the initiator's I/O Buffer containing the non-immediate unsolicited data and solicited data. This Write STag is used as the Data Source STag in the resultant RDMA Read operation(s). When the Write STag Valid flag is set to zero, this field MUST be set to zero and ignored on receive.

#### Write Virtual Address High: 32 bits

This field contains the high order bits of the Virtual Address for the SCSI Write command when the Write STag Valid flag is set to one. When the Write STag Valid flag is set to zero, this field MUST be set to zero and ignored on receive.

#### Write Virtual Address Low: 32 bits

This field contains the low order bits of the Virtual Address for the SCSI Write command when the Write STag Valid flag is set to one. When the Write STag Valid flag is set to zero, this field MUST be set to zero and ignored on receive.

#### Read STag - Read Steering Tag: 32 bits

This field contains the Read STag when the Read STag Valid flag is set to one. The Read STag is used to Advertise the initiator's Read I/O Buffer of a SCSI Read or bidirectional command, or a Task Management Function Request with the TASK REASSIGN function. This Read STag is used as the Data Sink STag in the resultant RDMA Write operation(s). When the Read STag Valid flag is zero, this field MUST be set to zero and ignored on receive.

#### Read Virtual Address High: 32 bits

This field contains the high order bits of the Virtual Address for the SCSI Read command when the Read STag Valid flag is set to one. When the Read STag Valid flag is set to zero, this field MUST be set to zero and ignored on receive.

#### Read Virtual Address Low: 32 bits

This field contains the low order bits of the Virtual Address for the SCSI Read command when the Read STag Valid flag is set to one. When the read STag Valid flag is set to zero, this field MUST be set to zero and ignored on receive.

Reserved fields MUST be set to zero on transmit and MUST be ignored on receive.

9.3 iSER Header Format for iSER Hello Message

An iSER Hello Message MUST only contain the iSER header which MUST have the format as described in Figure 4. If iSERHelloRequired is negotiated to "Yes", then iSER Hello Message is the first iSER Message sent on the RCaP Stream from the iSER Layer at the initiator to the iSER Layer at the target.

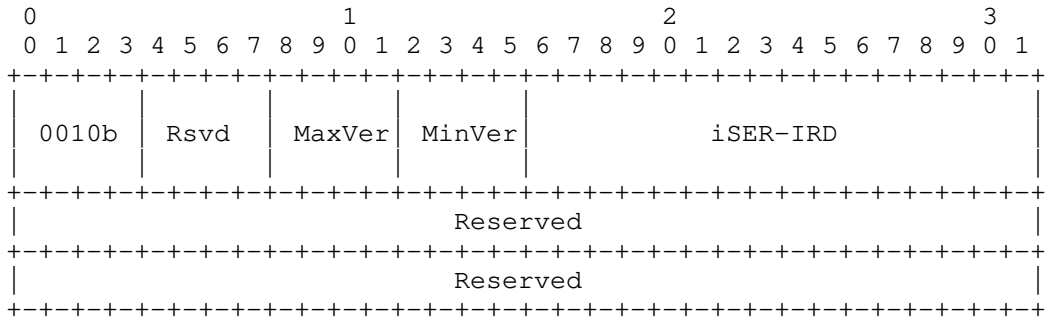


Figure 4 iSER Header Format for iSER Hello Message

MaxVer - Maximum Version: 4 bits

This field specifies the maximum version of the iSER protocol supported. It MUST be set to 10 to indicate the version of the specification described in this document.

MinVer - Minimum Version: 4 bits

This field specifies the minimum version of the iSER protocol supported. It MUST be set to 10 to indicate the version of the specification described in this document.

iSER-IRD: 16 bits

This field contains the value of the iSER-IRD at the initiator.

Reserved (Rsvd):

Reserved fields MUST be set to zero on transmit, and MUST be ignored on receive.

## 9.4 iSER Header Format for iSER HelloReply Message

An iSER HelloReply Message MUST only contain the iSER header which MUST have the format as described in Figure 5. If iSERHelloRequired is negotiated to "Yes", then the iSER HelloReply Message is the first iSER Message sent on the RCaP Stream from the iSER Layer at the target to the iSER Layer at the initiator.

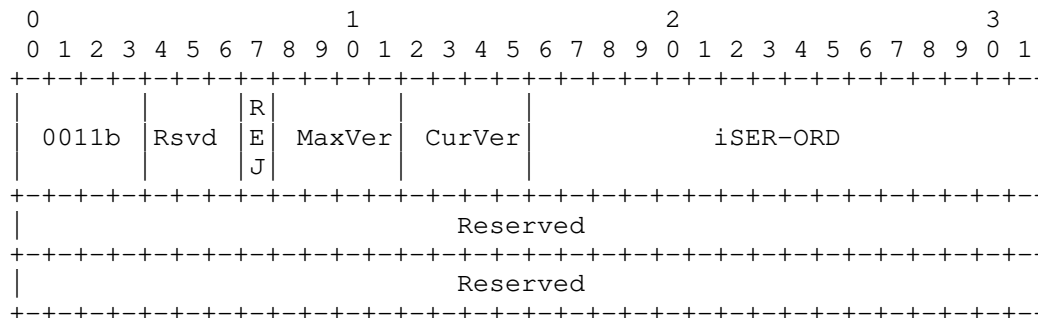


Figure 5 iSER Header Format for iSER HelloReply Message

REJ - Reject flag: 1 bit

This flag indicates whether the target is rejecting this connection. If set to one, the target is rejecting the connection.

MaxVer - Maximum Version: 4 bits

This field specifies the maximum version of the iSER protocol supported. It MUST be set to 10 to indicate the version of the specification described in this document.

CurVer - Current Version: 4 bits

This field specifies the current version of the iSER protocol supported. It MUST be set to 10 to indicate the version of the specification described in this document.

iSER-ORD: 16 bits

This field contains the value of the iSER-ORD at the target.

Reserved (Rsvd):

Reserved fields MUST be set to zero on transmit, and MUST be ignored on receive.

## 9.5 SCSI Data Transfer Operations

The iSER Layer at the initiator and the iSER Layer at the target handle each SCSI Write, SCSI Read, and bidirectional operation as described below.

## 9.5.1 SCSI Write Operation

The iSCSI Layer at the initiator MUST invoke the Send\_Control Operational Primitive to request the iSER Layer at the initiator to send the SCSI Write Command. The iSER Layer at the initiator MUST request the RCaP layer to transmit a SendSE Message with the message payload consisting of the iSER header followed by the SCSI Command PDU and immediate data (if any). If there is solicited data, the iSER Layer MUST Advertise the Write STag in the iSER header of the SendSE Message, as described in section 9.2. Upon receiving the SendSE Message, the iSER Layer at the target MUST notify the iSCSI Layer at the target by invoking the Control\_Notify Operational Primitive qualified with the SCSI Command PDU. See section 7.3.1 for details on the handling of the SCSI Write Command.

For the non-immediate unsolicited data, the iSCSI Layer at the initiator MUST invoke a Send\_Control Operational Primitive qualified with the SCSI Data-out PDU. Upon receiving each Send or SendSE Message containing the non-immediate unsolicited data, the iSER Layer at the target MUST notify the iSCSI Layer at the target by invoking the Control\_Notify Operational Primitive qualified with the SCSI Data-out PDU. See section 7.3.4 for details on the handling of the SCSI Data-out PDU.

For the solicited data, when the iSCSI Layer at the target has an I/O Buffer available, it MUST invoke the Get\_Data Operational Primitive qualified with the R2T PDU. See section 7.3.6 for details on the handling of the R2T PDU.

When the data transfer associated with this SCSI Write operation is complete, the iSCSI Layer at the target MUST invoke the Send\_Control Operational Primitive when it is ready to send the SCSI Response PDU. Upon receiving a SendSE or SendInvSE Message containing the SCSI Response PDU, the iSER Layer at the initiator MUST notify the iSCSI Layer at the initiator by invoking the Control\_Notify Operational Primitive qualified with the SCSI Response PDU. See section 7.3.2 for details on the handling of the SCSI Response PDU.

## 9.5.2 SCSI Read Operation

The iSCSI Layer at the initiator MUST invoke the Send\_Control Operational Primitive to request the iSER Layer at the initiator to send the SCSI Read Command. The iSER Layer at the initiator MUST request the RCaP layer to transmit a SendSE Message with the message payload consisting of the iSER header followed by the SCSI Command PDU. The iSER Layer at the initiator MUST Advertise the Read STag in the iSER header of the SendSE Message, as described in section 9.2. Upon receiving the SendSE Message, the iSER Layer at the target MUST notify the iSCSI Layer at the target by invoking the Control\_Notify Operational Primitive qualified with the SCSI Command PDU. See section 7.3.1 for details on the handling of the SCSI Read Command.

When the requested SCSI data is available in the I/O Buffer, the iSCSI Layer at the target MUST invoke the Put\_Data Operational Primitive qualified with the SCSI Data-in PDU. See section 7.3.5 for details on the handling of the SCSI Data-in PDU.

When the data transfer associated with this SCSI Read operation is complete, the iSCSI Layer at the target MUST invoke the Send\_Control Operational Primitive when it is ready to send the SCSI Response PDU. Upon receiving the SendInvSE Message containing the SCSI Response PDU, the iSER Layer at the initiator MUST notify the iSCSI Layer at the initiator by invoking the Control\_Notify Operational Primitive qualified with the SCSI Response PDU. See section 7.3.2 for details on the handling of the SCSI Response PDU.

## 9.5.3 Bidirectional Operation

The initiator and the target handle the SCSI Write and the SCSI Read portions of this bidirectional operation the same as described in Section 9.5.1 and Section 9.5.2 respectively.



## 10 iSER Error Handling and Recovery

RCaP provides the iSER Layer with reliable in-order delivery. Therefore, the error management needs of an iSER-assisted connection are somewhat different than those of a Traditional iSCSI connection.

## 10.1 Error Handling

iSER error handling is described in the following sections, classified loosely based on the sources of errors:

1. Those originating at the transport layer (e.g., TCP).
2. Those originating at the RCaP layer.
3. Those originating at the iSER Layer.
4. Those originating at the iSCSI Layer.

## 10.1.1 Errors in the Transport Layer

If the transport layer is TCP, then TCP packets with detected errors are silently dropped by the TCP layer and result in retransmission at the TCP layer. This has no impact on the iSER Layer. However, connection loss (e.g., link failure) and unexpected termination (e.g., TCP graceful or abnormal close without the iSCSI Logout exchanges) at the transport layer will cause the iSCSI/iSER connection to be terminated as well.

## 10.1.1.1 Failure in the Transport Layer Before RCaP Mode is Enabled

If the Connection is lost or terminated before the iSCSI Layer invokes the `Allocate_Connection_Resources` Operational Primitive, the login process is terminated and no further action is required.

If the Connection is lost or terminated after the iSCSI Layer has invoked the `Allocate_Connection_Resources` Operational Primitive, then the iSCSI Layer MUST request the iSER Layer to deallocate all connection resources by invoking the `Deallocate_Connection_Resources` Operational Primitive.

## 10.1.1.2 Failure in the Transport Layer After RCaP Mode is Enabled

If the Connection is lost or terminated after the iSCSI Layer has invoked the `Enable_Datamover` Operational Primitive, the iSER Layer MUST notify the iSCSI Layer of the connection loss by invoking the `Connection_Terminate_Notify` Operational Primitive. Prior to

#### 10.1.2 Errors in the RCaP Layer

The RCaP layer does not have error recovery operations built in. If errors are detected at the RCaP layer, the RCaP layer will terminate the RCaP Stream and the associated Connection.

##### 10.1.2.1 Errors Detected in the Local RCaP Layer

If an error is encountered at the local RCaP layer, the RCaP layer MAY send a Terminate Message to the Remote Peer to report the error if possible. (For iWARP, see [RDMAP] for the list of errors where a Terminate Message is sent.) The RCaP layer is responsible for terminating the Connection. After the RCaP layer notifies the iSER Layer that the Connection is terminated, the iSER Layer MUST notify the iSCSI Layer by invoking the Connection\_Terminate\_Notify Operational Primitive. Prior to invoking the Connection\_Terminate\_Notify Operational Primitive, the iSER layer MUST perform the actions described in Section 5.2.3.2.

##### 10.1.2.2 Errors Detected in the RCaP Layer at the Remote Peer

If an error is encountered at the RCaP layer at the Remote Peer, the RCaP layer at the Remote Peer may send a Terminate Message to report the error if possible. If it is unable to send the Terminate Message, the Connection is terminated. This is treated the same as a failure in the transport layer after RDMA is enabled as described in section 10.1.1.2.

If an error is encountered at the RCaP layer at the Remote Peer and it is able to send a Terminate Message, the RCaP layer at the Remote Peer is responsible for terminating the connection. After the local RCaP layer notifies the iSER Layer that the Connection is terminated, the iSER Layer MUST notify the iSCSI Layer by invoking the Connection\_Terminate\_Notify Operational Primitive. Prior to invoking the Connection\_Terminate\_Notify Operational Primitive, the iSER layer MUST perform the actions described in Section 5.2.3.2.

#### 10.1.3 Errors in the iSER Layer

The error handling due to errors at the iSER Layer is described in the following sections.

### 10.1.3.1 Insufficient Connection Resources to Support RCaP at Connection Setup

After the iSCSI Layer at the initiator invokes the `Allocate_Connection_Resources` Operational Primitive during the iSCSI login negotiation phase, if the iSER Layer at the initiator fails to allocate the connection resources necessary to support RCaP, it MUST return a status of failure to the iSCSI Layer at the initiator. The iSCSI Layer at the initiator MUST terminate the Connection as described in Section 5.2.3.1.

After the iSCSI Layer at the target invokes the `Allocate_Connection_Resources` Operational Primitive during the iSCSI login negotiation phase, if the iSER Layer at the target fails to allocate the connection resources necessary to support RCaP, it MUST return a status of failure to the iSCSI Layer at the target. The iSCSI Layer at the target MUST send a Login Response with a status class of 3 (Target Error), and a status code of "0302" (Out of Resources). The iSCSI Layers at the initiator and the target MUST terminate the Connection as described in Section 5.2.3.1.

### 10.1.3.2 iSER Negotiation Failures

If `iSERHelloRequired` is negotiated to "Yes" and the RCaP or iSER related parameters declared by the initiator in the iSER Hello Message is unacceptable to the iSER Layer at the target, the iSER Layer at the target MUST set the Reject (REJ) flag, as described in section 9.4, in the iSER HelloReply Message. The following are the cases when the iSER Layer MUST set the REJ flag to 1 in the HelloReply Message:

- \* The initiator-declared iSER-IRD value is greater than 0 and the target-declared iSER-ORD value is 0.
- \* The initiator-supported and the target-supported iSER protocol versions do not overlap.

After requesting the RCaP layer to send the iSER HelloReply Message, the handling of the error situation is the same as that for iSER format errors as described in section 10.1.3.3.

### 10.1.3.3 iSER Format Errors

The following types of errors in an iSER header are considered format errors:

- \* Illegal contents of any iSER header field

- \* Inconsistent field contents in an iSER header
- \* Length error for an iSER Hello or HelloReply Message (see section 9.3 and 9.4)

When a format error is detected, the following events MUST occur in the specified sequence:

1. The iSER Layer MUST request the RCaP layer to terminate the RCaP Stream. The RCaP layer MUST terminate the associated Connection.
2. The iSER Layer MUST notify the iSCSI Layer of the connection termination by invoking the Connection\_Terminate\_Notify Operational Primitive. Prior to invoking the Connection\_Terminate\_Notify Operational Primitive, the iSER layer MUST perform the actions described in Section 5.2.3.2.

#### 10.1.3.4 iSER Protocol Errors

If iSERHelloRequired is negotiated to "Yes", then the first iSER Message sent by the iSER Layer at the initiator MUST be the iSER Hello Message (see section 9.3). In this case the first iSER Message sent by the iSER Layer at the target MUST be the iSER HelloReply Message (see section 9.4). Failure to send the iSER Hello or HelloReply Message, as indicated by the wrong Opcode in the iSER header, is a protocol error. The handling of this error situation is the same as that for iSER format errors as described in section 10.1.3.3.

If the sending side of an iSER-enabled connection acts in a manner not permitted by the negotiated or declared login/text operational key values as described in section 6, this is a protocol error and the receiving side MAY handle this the same as for iSER format errors as described in section 10.1.3.3.

#### 10.1.4 Errors in the iSCSI Layer

The error handling due to errors at the iSCSI Layer is described in the following sections. For error recovery, see section 10.2.

##### 10.1.4.1 iSCSI Format Errors

When an iSCSI format error is detected, the iSCSI Layer MUST request the iSER Layer to terminate the RCaP Stream by invoking the Connection\_Terminate Operational Primitive. For more details on the connection termination, see Section 5.2.3.1.

#### 10.1.4.2 iSCSI Digest Errors

In the iSER-assisted mode, the iSCSI Layer will not see any digest error because both the HeaderDigest and the DataDigest keys are negotiated to "None".

#### 10.1.4.3 iSCSI Sequence Errors

For Traditional iSCSI, sequence errors are caused by dropped PDUs due to header or data digest errors. Since digests are not used in iSER-assisted mode and the RCaP layer will deliver all messages in the order they were sent, sequence errors will not occur in iSER-assisted mode.

#### 10.1.4.4 iSCSI Protocol Error

When the iSCSI Layer handles certain protocol errors by dropping the connection, the error handling is the same as that for iSCSI format errors as described in section 10.1.4.1.

When the iSCSI Layer uses the iSCSI Reject PDU and response codes to handle certain other protocol errors, no special handling at the iSER Layer is required.

#### 10.1.4.5 SCSI Timeouts and Session Errors

This is handled at the iSCSI Layer and no special handling at the iSER Layer is required.

#### 10.1.4.6 iSCSI Negotiation Failures

For negotiation failures that happen during the Login Phase at the initiator after the iSCSI Layer has invoked the `Allocate_Connection_Resources` Operational Primitive and before the `Enable_Datamover` Operational Primitive has been invoked, the iSCSI Layer MUST request the iSER Layer to deallocate all connection resources by invoking the `Deallocate_Connection_Resources` Operational Primitive. The iSCSI Layer at the initiator MUST terminate the Connection.

For negotiation failures during the Login Phase at the target, the iSCSI Layer can use a Login Response with a status class other than 0 (success) to terminate the Login Phase. If the iSCSI Layer has invoked the `Allocate_Connection_Resources` Operational Primitive and before the `Enable_Datamover` Operational Primitive has been invoked, the iSCSI Layer at the target MUST request the iSER Layer at the target to deallocate all connection resources by invoking the

Deallocate\_Connection\_Resources Operational Primitive. The iSCSI Layer at both the initiator and the target MUST terminate the Connection.

During the iSCSI Login Phase, if the iSCSI Layer at the initiator receives a Login Response from the target with a status class other than 0 (Success) after the iSCSI Layer at the initiator has invoked the Allocate\_Connection\_Resources Operational Primitive, the iSCSI Layer MUST request the iSER Layer to deallocate all connection resources by invoking the Deallocate\_Connection\_Resources Operational Primitive. The iSCSI Layer MUST terminate the Connection in this case.

For negotiation failures during the full feature phase, the error handling is left to the iSCSI Layer and no special handling at the iSER Layer is required.

## 10.2 Error Recovery

Error recovery requirements of iSCSI/iSER are the same as that of Traditional iSCSI. All three ErrorRecoveryLevels as defined in [RFC3720] are supported in iSCSI/iSER.

- \* For ErrorRecoveryLevel 0, session recovery is handled by iSCSI and no special handling by the iSER Layer is required.
- \* For ErrorRecoveryLevel 1, see section 10.2.1 on PDU Recovery.
- \* For ErrorRecoveryLevel 2, see section 10.2.2 on Connection Recovery.

The iSCSI Layer may invoke the Notice\_Key\_Values Operational Primitive during connection setup to request the iSER Layer to take note of the value of the operational ErrorRecoveryLevel, as described in sections 5.1.1 and 5.1.2.

### 10.2.1 PDU Recovery

As described in sections 10.1.4.2 and 10.1.4.3, digest and sequence errors will not occur in the iSER-assisted mode. If the RCaP layer detects an error, it will close the iSCSI/iSER connection, as described in section 10.1.2. Therefore, PDU recovery is not useful in the iSER-assisted mode.

The iSCSI Layer at the initiator SHOULD disable iSCSI timeout-driven PDU retransmissions.

## 10.2.2 Connection Recovery

The iSCSI Layer at the initiator MAY reassign connection allegiance for non-immediate commands which are still in progress and are associated with the failed connection by using a Task Management Function Request with the TASK REASSIGN function. See section 7.3.3 for more details.

When the iSCSI Layer at the initiator does a task reassignment for a SCSI Write command, it MUST qualify the Send\_Control Operational Primitive invocation with DataDescriptorOut which defines the I/O Buffer for both the non-immediate unsolicited data and the solicited data. This allows the iSCSI Layer at the target to use recovery R2Ts to request for data originally sent as unsolicited and solicited from the initiator.

When the iSCSI Layer at the target accepts a reassignment request for a SCSI Read command, it MUST request the iSER Layer to process SCSI Data-in for all unacknowledged data by invoking the Put\_Data Operational Primitive. See section 7.3.5 on the handling of SCSI Data-in.

When the iSCSI Layer at the target accepts a reassignment request for a SCSI Write command, it MUST request the iSER Layer to process a recovery R2T for any non-immediate unsolicited data and any solicited data sequences that have not been received by invoking the Get\_Data Operational Primitive. See section 7.3.6 on the handling of Ready To Transfer (R2T).

The iSCSI Layer at the target MUST NOT issue recovery R2Ts on an iSCSI/iSER connection for a task for which the connection allegiance was never reassigned. The iSER Layer at the target MAY reject such a recovery R2T received via the Get\_Data Operational Primitive invocation from the iSCSI Layer at the target, with an appropriate error code.

The iSER Layer at the target will process the requests invoked by the Put\_Data and Get\_Data Operational Primitives for a reassigned task in the same way as for the original commands.

## 11 Security Considerations

When iSER is layered on top of an RCaP layer and provides the RDMA extensions to the iSCSI protocol, the security considerations of iSER are the same as that of the underlying RCaP layer. For iWARP, this is described in [RDMAP] and [RDDPSEC].

Since iSER-assisted iSCSI protocol is still functionally iSCSI from a security considerations perspective, all of the iSCSI security requirements as described in [RFC3720] and [RFC3723] apply. If iSER is layered on top of a non-IP based RCaP layer, all the security protocol mechanisms applicable to that RCaP layer is also applicable to an iSCSI/iSER connection. If iSER is layered on top of a non-IP protocol, the IPsec mechanism as specified in [RFC3720] MUST be implemented at any point where the iSER protocol enters the IP network (e.g., via gateways), and the non-IP protocol SHOULD implement (optional to use) a packet by packet security protocol equal in strength to the IPsec mechanism specified by [RFC3720].

To minimize the potential for a denial of service attack, the iSCSI Layer MUST NOT request the iSER Layer to allocate the connection resources necessary to support RCaP until the iSCSI layer is sufficiently far along in the iSCSI Login Phase that it is reasonably certain that the peer side is not an attacker, as described in sections 5.1.1 and 5.1.2.



This document has no actions for IANA.

### 13.1 Normative References

- [RFC5046] M. Ko et al., "iSCSI Extensions for Remote Direct Memory Access", RFC 5046, October 2007
- [RFC3720] J. Satran et al., "iSCSI", RFC 3720, April 2004
- [RFC3723] B. Aboba et al., "Securing Block Storage Protocols over IP", RFC 3723, April 2004.
- [RDMAP] R. Recio et al., "An RDMA Protocol Specification", RFC 5040, October 2007
- [DDP] H. Shah et al., "Direct Data Placement over Reliable Transports", RFC 5041, October 2007
- [MPA] P. Culley et al., "Marker PDU Aligned Framing for TCP Specification", RFC 5044, October 2007
- [RDDPSEC] J. Pinkerton et al., "DDP/RDMAP Security", RFC 5042, October 2007
- [TCP] Postel, J., "Transmission Control Protocol", STD 7, RFC 793, September 1981
- [RFC2119] Bradner, S., "Key Words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997

### 13.2 Informative References

- [SAM2] T10/1157D, SCSI Architecture Model - 2 (SAM-2)
- [DA] M. Chadalapaka et al., "Datamover Architecture for iSCSI", RFC 5047, October 2007
- [VERBS] J. Hilland et al., "RDMA Protocol Verbs Specification", RDMAC Consortium Draft Specification draft-hilland-iwarp-verbs-v1.0-RDMAC, April 2003
- [IPSEC] S. Kent et al., "Security Architecture for the Internet Protocol", RFC 2401, November 1998
- [IB] InfiniBand Architecture Specification Volume 1 Release 1.2, October 2004



1. Removed the requirement that a connection be opened in "normal" TCP mode and transitioned to zero-copy mode. This allows the spec to conform to existing implementation for both Infiniband and iWARP. Changes were made in sections 2, 3.1.6, 4.2, 5.1, 5.1.1, 5.1.2, 5.1.3, 10.1.3.4, and 11.
2. Added a clause in section 6.2 to clarify that MaxRecvDataSegmentLength must be ignored if it is declared in the Login Phase.
3. Added a clause in section 6.2 to clarify that the initiator must not send more than InitiatorMaxRecvDataSegmentLength worth of data when a NOP-Out request is sent with a valid Initiator Task Tag. Since InitiatorMaxRecvDataSegmentLength can be smaller than TargetMaxRecvDataSegmentLength, returning the original data in the NOP-Out request in this situation can overflow the receive buffer unless the length of the data sent with the NOP-Out request is less than InitiatorMaxRecvDataSegmentLength.
4. Changed the default of MaxOutstandingUnexpectedPDUs from 0 (unlimited) to 4.
5. Added MaxAHSLength key in section 6.8 to set a limit on the AHS Length. This is useful when posting receive buffers in knowing what the maximum possible message length is in a PDU which contains AHS.
6. Added WriteAddressForSolicitedDataOnly key in section 6.9 to indicate how the memory region will be used. An initiator can treat the memory regions intended for unsolicited and solicited data differently, and can use different registration modes. In contrast, RFC 5046 treats the memory occupied by the data as a contiguous (or virtually contiguous, by means of scatter-gather mechanisms) and homogenous region. Adding a new key will allow different memory models to be accommodated.
7. Added iSERHelloRequired key in section 6.10 to make it optional to use iSER Hello messages. iSER Hello messages are required for certain RCaP implementations such as iWARP but can cause problems for others such as InfiniBand. Changes were made in sections 5.1.1, 5.1.2, 5.1.3, 8.2, 9.3, 9.4, 10.1.3.2 and 10.1.3.4.
8. Added two 64-bit fields in iSER header in section 9.2 for the Read Virtual Address and the Write Virtual Address to conform to the Infiniband format. This allows one implementation such as the

Internet-Draft

iSER Specification

July 2011

OFED stack to be used in both the Infiniband and the iWARP  
environment.

## 15 Appendix B: Message Format for iSER

This section is for information only and is NOT part of the standard.

## 15.1 iWARP Message Format for iSER Hello Message

The following figure depicts an iSER Hello Message encapsulated in an iWARP SendSE Message.

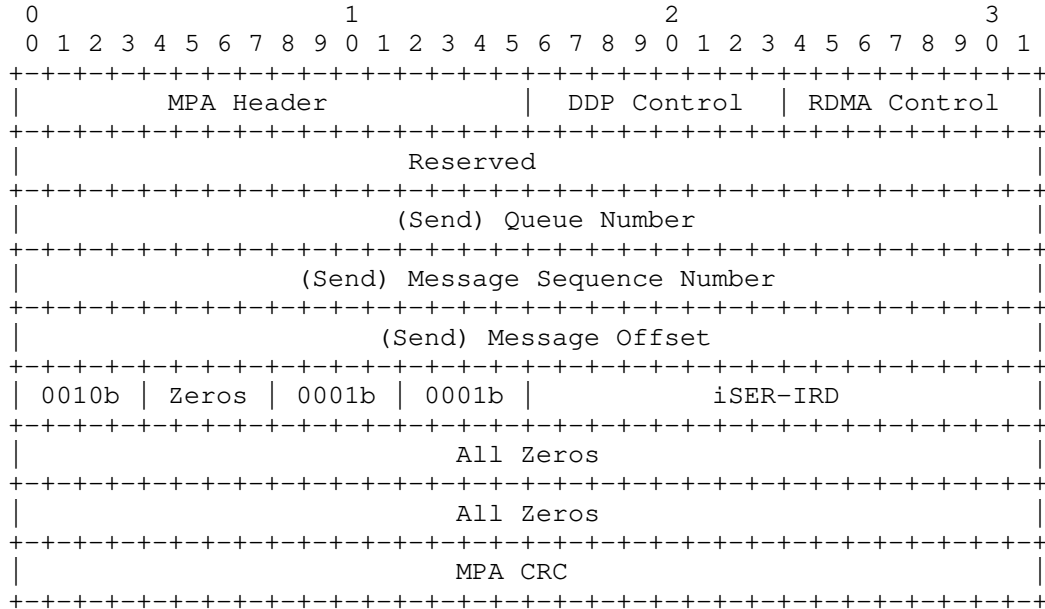


Figure 6 SendSE Message containing an iSER Hello Message

## 15.2 iWARP Message Format for iSER HelloReply Message

The following figure depicts an iSER HelloReply Message encapsulated in an iWARP SendSE Message. The Reject (REJ) flag is set to 0.

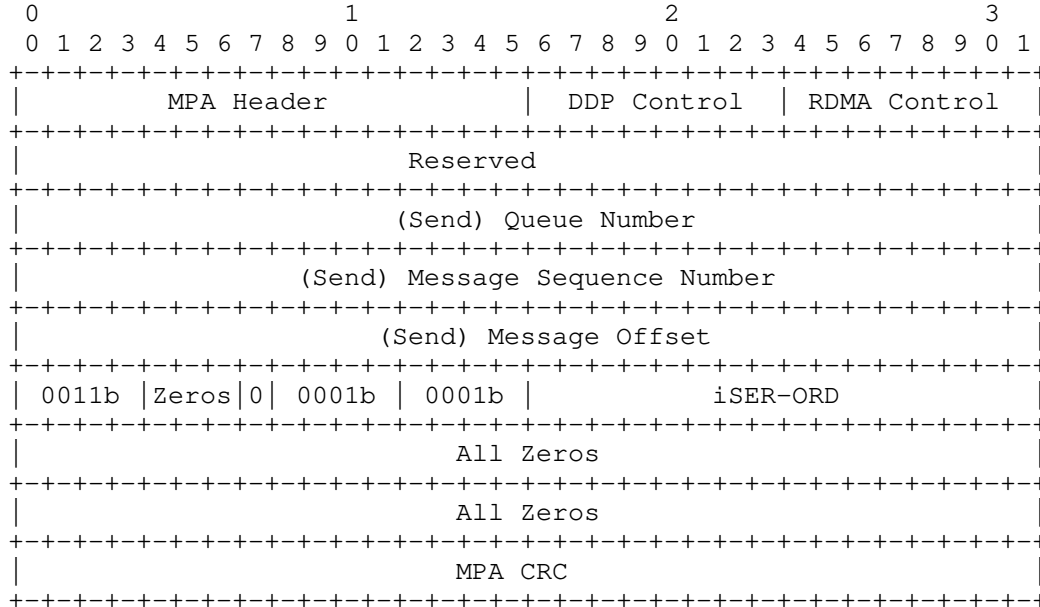


Figure 7 SendSE Message containing an iSER HelloReply Message

## 15.3 iWARP Message Format for SCSI Read Command PDU

The following figure depicts a SCSI Read Command PDU embedded in an iSER Message encapsulated in an iWARP SendSE Message. For this particular example, in the iSER header, the Write STag Valid flag is set to zero, the Read STag Valid flag is set to one, the Write STag field is set to all zeros, and the Read STag field contains a valid Read STag.

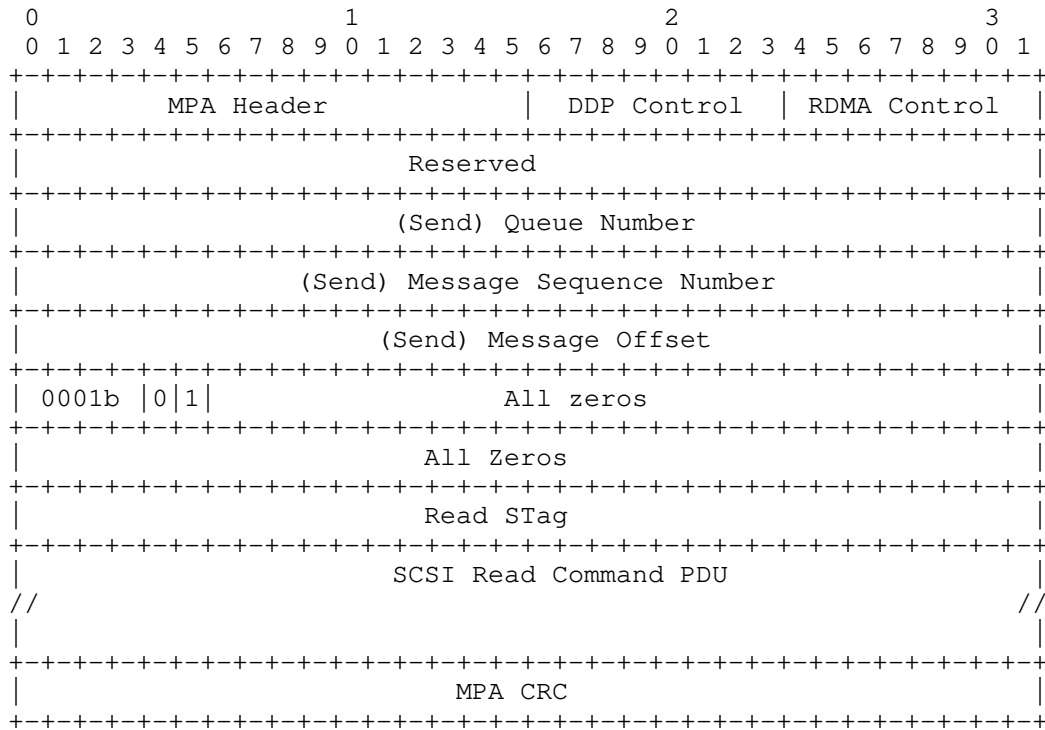


Figure 8 SendSE Message containing a SCSI Read Command PDU



## 15.4 iWARP Message Format for SCSI Read Data

The following figure depicts an iWARP RDMA Write Message carrying SCSI Read data in the payload:

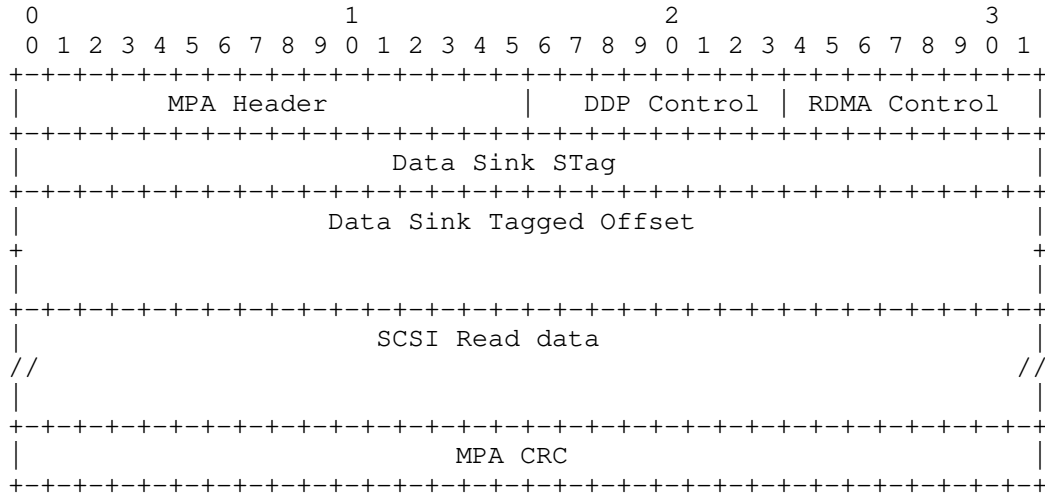


Figure 9 RDMA Write Message containing SCSI Read Data

## 15.5 iWARP Message Format for SCSI Write Command PDU

The following figure depicts a SCSI Write Command PDU embedded in an iSER Message encapsulated in an iWARP SendSE Message. For this particular example, in the iSER header, the Write STag Valid flag is set to one, the Read STag Valid flag is set to zero, the Write STag field contains a valid Write STag, and the Read STag field is set to all zeros since it is not used.

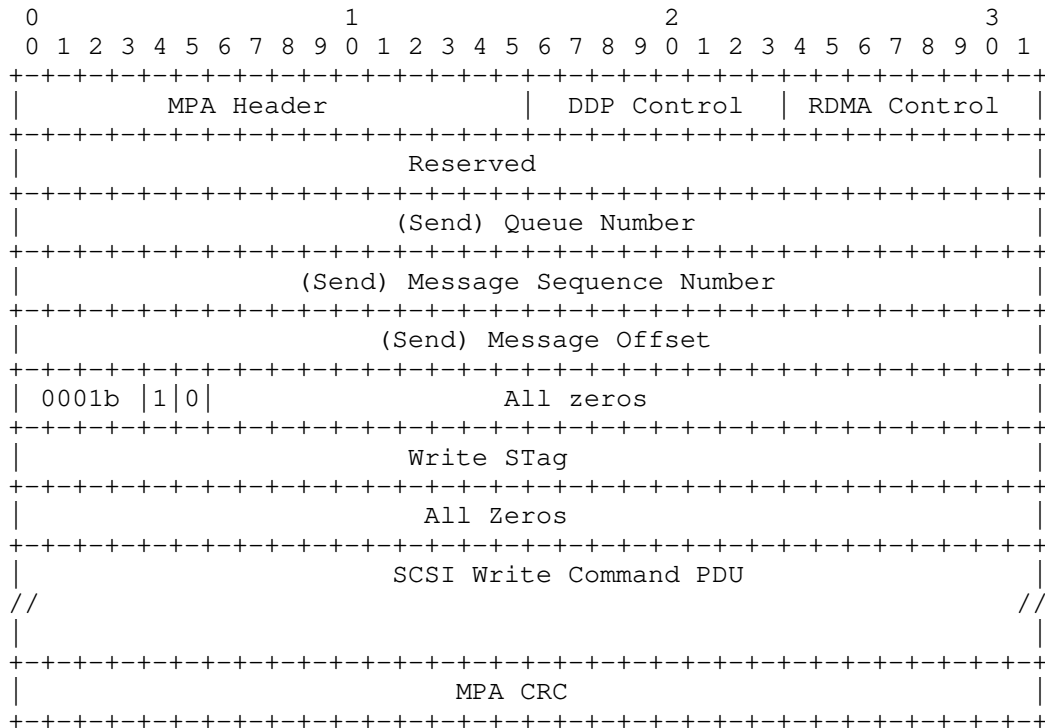


Figure 10 SendSE Message containing a SCSI Write Command PDU

## 15.6 iWARP Message Format for RDMA Read Request

An iSCSI R2T is transformed into an iWARP RDMA Read Request Message. The following figure depicts an iWARP RDMA Read Request Message:

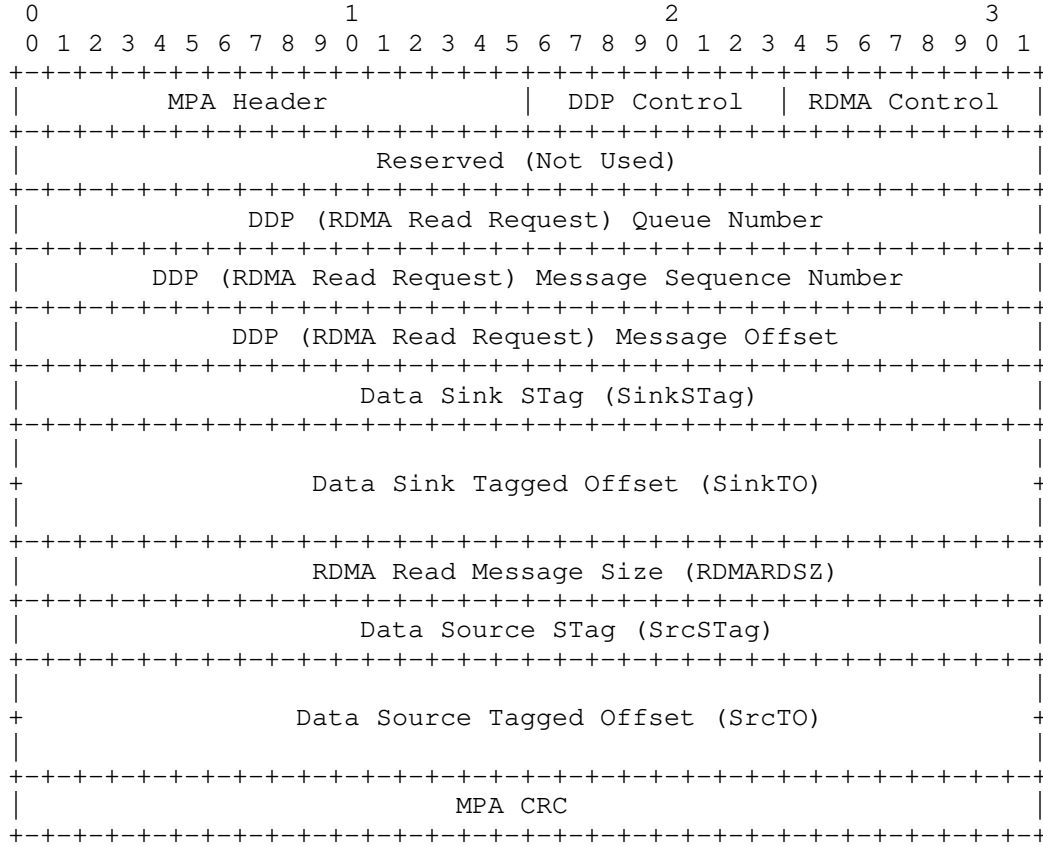


Figure 11 RDMA Read Request Message

## 15.7 iWARP Message Format for Solicited SCSI Write Data

The following figure depicts an iWARP RDMA Read Response Message carrying the solicited SCSI Write data in the payload:

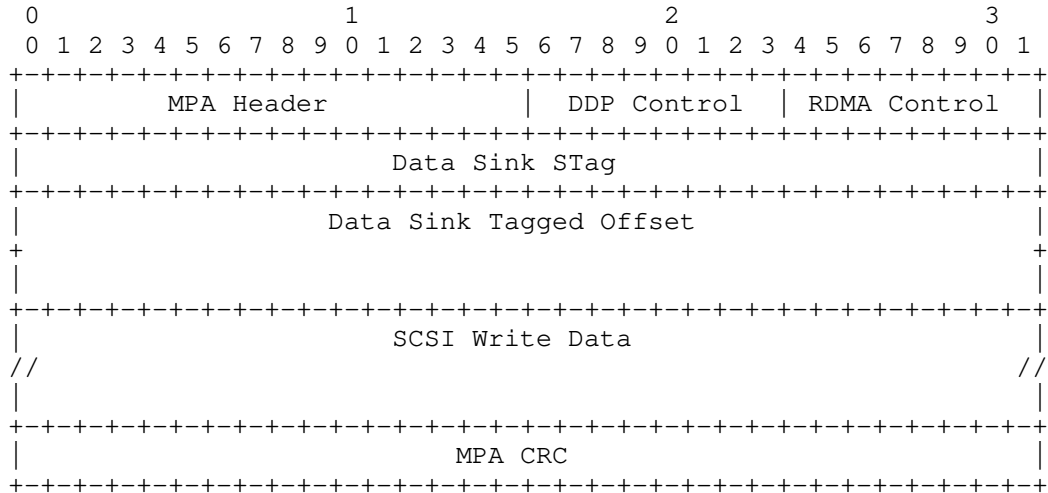


Figure 12 RDMA Read Response Message containing SCSI Write Data

## 15.8 iWARP Message Format for SCSI Response PDU

The following figure depicts a SCSI Response PDU embedded in an iSER Message encapsulated in an iWARP SendInvSE Message:

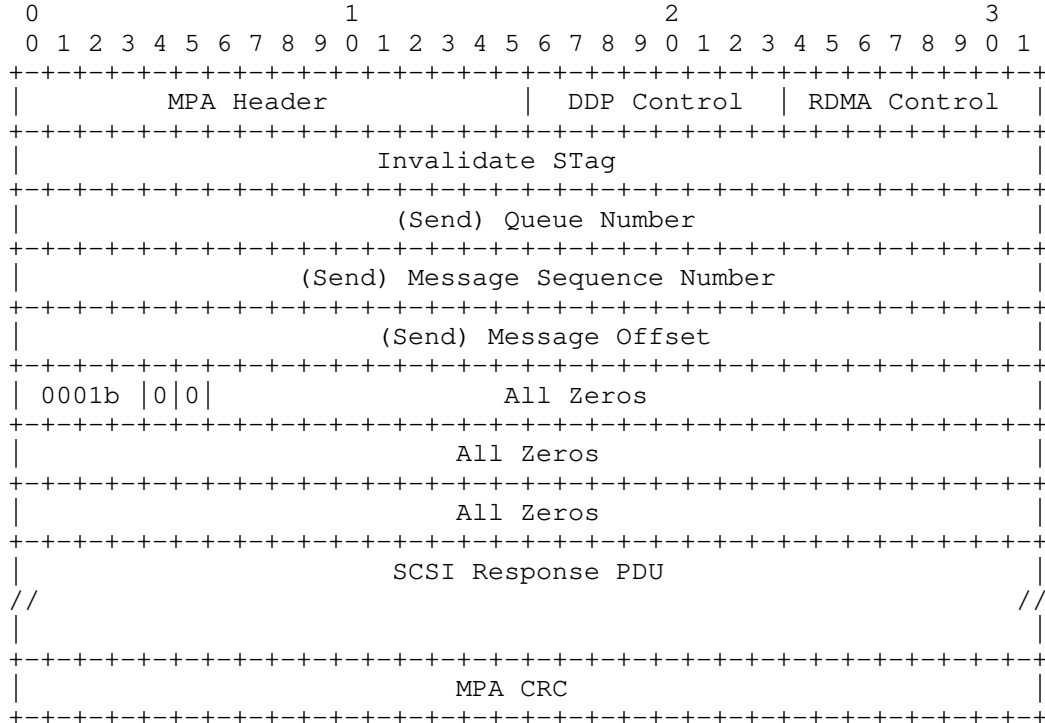


Figure 13 SendInvSE Message containing SCSI Response PDU

## 16 Appendix C: Architectural discussion of iSER over InfiniBand

This section explains how an InfiniBand network (with Gateways) would be structured. It is informational only and is intended to provide insight on how iSER is used in an InfiniBand environment.

## 16.1 Host side of iSCSI &amp; iSER connections in Infiniband

Figure 14 defines the topologies in which iSCSI and iSER will be able to operate on an InfiniBand Network.

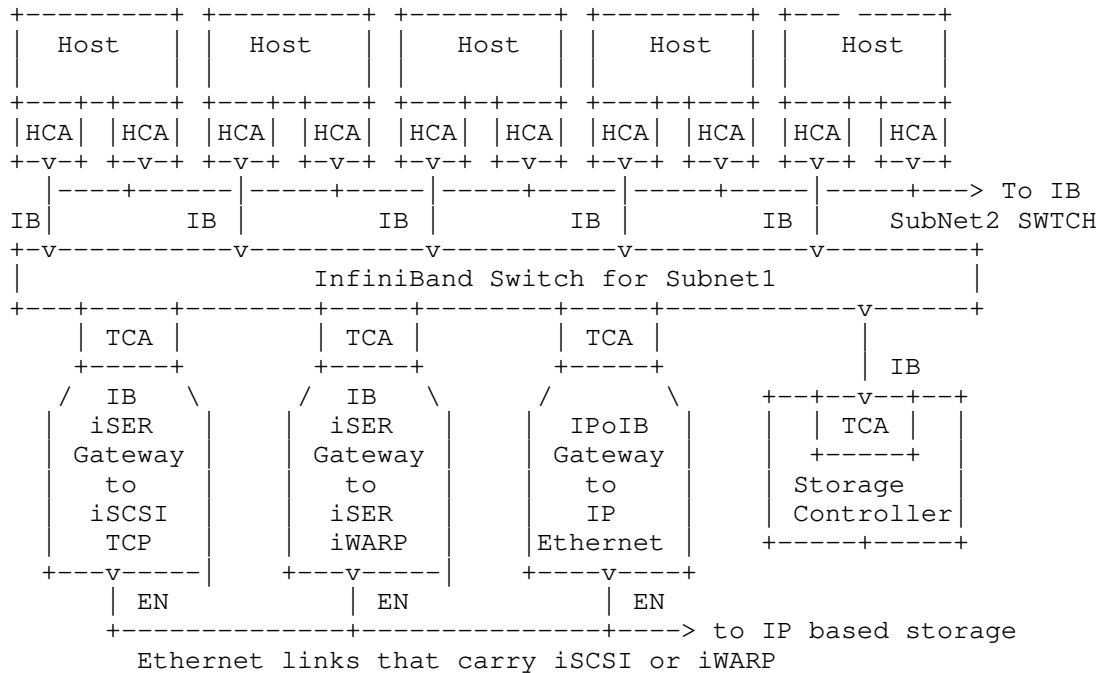


Figure 14 iSCSI and iSER on IB

In Figure 14, the Host systems are connected via the InfiniBand Host Channel Adapters (HCAs) to the InfiniBand links. With the use of IB switch(es), the InfiniBand links connect the HCA to InfiniBand Target Channel Adapters (TCAs) located in gateways or Storage Controllers. An iSER-capable IB-IP Gateway converts the iSER Messages encapsulated in IB protocols to either standard iSCSI, or iSER Messages for iWARP. An [IPoIB] Gateway converts the InfiniBand [IPoIB] protocol to IP protocol, and in the iSCSI case, permits iSCSI to be operated on an IB Network between the Hosts and the [IPoIB] Gateway.

## 16.2 Storage side of iSCSI &amp; iSER mixed network environment

Figure 15 shows a storage controller that has three different portal groups: one supporting only iSCSI (TPG-4), one supporting iSER/iWARP or iSCSI (TPG-2), and one supporting iSER/IB (TPG-1).

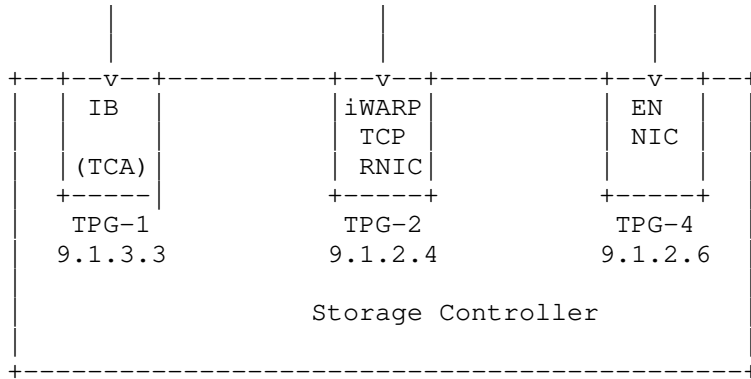


Figure 15 Storage Controller with TCP, iWARP, and IB Connections

The normal iSCSI portal group advertising processes (via SLP, iSNS, or SendTargets) are available to a Storage Controller.

## 16.3 Discovery processes for an InfiniBand Host

An InfiniBand Host system can gather portal group IP address from SLP, iSNS, or the SendTargets discovery processes by using TCP/IP via [IPoIB]. After obtaining one or more remote portal IP addresses, the Initiator uses the standard IP mechanisms to resolve the IP address to a local outgoing interface and the destination hardware address (Ethernet MAC or IB GID of the target or a gateway leading to the target). If the resolved interface is an [IPoIB] network interface, then the target portal can be reached through an InfiniBand fabric. In this case the Initiator can establish an iSCSI/TCP or iSCSI/iSER session with the Target over that InfiniBand interface, using the Hardware Address (InfiniBand GID) obtained through the standard Address Resolution (ARP) processes.

If more than one IP address are obtained through the discovery process, the Initiator should select a Target IP address that is on the same IP subnet as the Initiator if one exists. This will avoid a potential overhead of going through a gateway when a direct path exists.

In addition a user can configure manual static IP route entries if a particular path to the target is preferred.

#### 16.4 IBTA Connection specifications

It is outside the scope of this document, but it is expected that the InfiniBand Trade Association (IBTA) has or will define:

- \* The iSER ServiceID
- \* A Means for permitting a Host to establish a connection with a peer InfiniBand end-node, and that peer indicating when that end-node supports iSER, so the Host would be able to fall back to iSCSI/TCP over [IPoIB].
- \* A Means for permitting the Host to establish connections with IB iSER connections on storage controllers or IB iSER connected Gateways in preference to [IPoIB] connected Gateways/Bridges or connections to Target Storage Controllers that also accept iSCSI via [IPoIB].
- \* A Means for combining the IB ServiceID for iSER and the IP port number such that the IB Host can use normal IB connection processes, yet ensure that the iSER target peer can actually connect to the required IP port number.



Credit goes to the authors of [RFC5046], M. Ko, M. Chadalapaka, J. Hufferd, U. Elzur, H. Shah, and P. Thaler for coming up with the first version of the iSER specification.

#### Author's Address

Michael Ko  
Huawei Symantec  
20245 Stevens Creek Blvd.  
Cupertino, CA 95014, USA  
Phone: +1-408-510-7465  
Email: michael@huaweisymantec.com

Alexander Nezhinsky  
Mellanox Technologies  
13 Zarchin St.  
Raanana 43662, Israel  
Phone: +972-74-712-9000  
Email: alexandern@mellanox.com, nezhinsky@gmail.com

#### Copyright Notice

Copyright (c) 2011 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<http://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Storage Maintenance (storm) Working Group  
Internet Draft  
Intended status: Standards Track  
Expires: September 2011

Hemal Shah  
Broadcom Corporation  
Felix Marti  
Wael Nouredine  
Asgeir Eiriksson  
Chelsio Communications, Inc.  
Robert Sharp  
Intel Corporation  
March 7, 2011

RDMA Protocol Extensions  
draft-ietf-storm-rdmap-ext-00.txt

Status of this Memo

This Internet-Draft is submitted to IETF in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at <http://datatracker.ietf.org/drafts/current>.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on September 7, 2011.

Copyright Notice

Copyright (c) 2011 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<http://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in

Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

## Abstract

This document specifies extensions to the IETF Remote Direct Memory Access Protocol (RDMA [RFC5040]). RDMA provides read and write services directly to applications and enables data to be transferred directly into Upper Layer Protocol (ULP) Buffers without intermediate data copies. The extensions specified in this document provide the following capabilities and/or improvements: Atomic Operations and Immediate Data.

## Table of Contents

1. Introduction.....	3
2. Requirements Language.....	3
3. Glossary.....	3
4. Header Format changes from RFC 5040.....	5
4.1. RDMA Control and Invalidate STag Fields.....	5
4.2. RDMA Message Definitions.....	6
5. Atomic Operations.....	7
5.1. Atomic Operation Details.....	8
5.1.1. FetchAdd.....	8
5.1.2. Swap.....	9
5.1.3. CmpSwap.....	10
5.2. Atomic Operations.....	11
5.2.1. Atomic Operation Request Message.....	11
5.2.2. Atomic Operation Response Message.....	15
5.3. Atomicity Guarantees.....	16
5.4. Atomic Operations Ordering and Completion Rules.....	16
6. Immediate Data.....	17
6.1. RDMA Interactions with the ULP for Immediate Data Operations.....	17
6.2. Immediate Data Header Format.....	18
6.3. Immediate Data or Immediate Data with SE Message.....	19
6.4. Ordering and Completions.....	19
7. Ordering and Completions Table.....	19
8. Error Processing.....	23
8.1. Errors Detected at the Local Peer.....	23
8.2. Errors Detected at the Remote Peer.....	23
9. Security Considerations.....	24

10. IANA Considerations.....	24
11. References.....	24
11.1. Normative References.....	24
11.2. Informative References.....	24
12. Acknowledgments.....	24
Appendix A. DDP Segment Formats for RDMA Messages.....	25
A.1. DDP Segment for Atomic Operation Request.....	25
A.2. DDP Segment for Atomic Response.....	27
A.3. DDP Segment for Immediate Data and Immediate Data with SE27	

## 1. Introduction

The RDMA Protocol [RFC5040] provides capabilities for zero copy and kernel bypass data communications. This document specifies the following extensions to the RDMA Protocol standard:

- o Atomic operations on remote memory locations. Support for atomic operation enhances the usability of RDMA in distributed shared memory environments.
- o Immediate Data messages allow the ULP at the sender to provide a small amount of data following an RDMA Write payload.

Other RDMA transport protocols define the functionality added by these extensions leading to differences in RDMA applications and/or Upper Layer Protocols. Removing these differences in the transport protocols simplifies these applications and ULPs and that is the main motivation for the extensions specified in this document.

## 2. Requirements Language

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in RFC-2119 [RFC2119].

## 3. Glossary

This document is an extension of [RFC5040] and key words are defined in the glossary of the referenced document.

Atomic Operation - is an operation that results in an execution of a 64-bit operation at a specific address on a remote node. The consumer can use atomic operations to read, modify and write at the destination address while at the same time guarantee that no other

read or write operation will occur across any other RDMAP/DDP Streams on an RNIC at the Data Sink.

Atomic Operation Request - An RDMA Message used by the Data Source to perform an atomic operation at the Data Sink.

Atomic Operation Response - An RDMA Message used by the Data Sink to describe the completion of an atomic operation at the Data Sink.

CmpSwap - is an Atomic Operation that is used to compare and swap a value at a specific address on a remote node.

FetchAdd - is an Atomic Operation that is used to atomically increment a value at a specific address on a remote node.

Immediate Data - a small fixed size portion of data sent from the Data Source to a Data Sink

Immediate Data Message - An RDMA Message used by the Data Source to send Immediate Data to the Data Sink

Immediate Data with Solicited Event (SE) Message - An RDMA Message used by the Data Source to send Immediate Data with Solicited Event to the Data Sink

Requester - the sender of an RDMA atomic operation request.

Responder - the receiver of an RDMA atomic operation request.

Swap - is an Atomic Operation that is used to swap a value at a specific address on a remote node.

#### 4. Header Format changes from RFC 5040

The control information of RDMA Messages is included in DDP protocol defined header fields, with the following new formats:

- . Four new RDMA Messages carry additional RDMAP headers. The Immediate Data operation and Immediate Data with Solicited Event operation include 8 bytes of data following the DDP header. Atomic Operations include Atomic Request or Atomic Response headers following the DDP header.

##### 4.1. RDMAP Control and Invalidate STag Fields

Figure 1 depicts the format of the DDP Control and RDMAP Control fields, in the style and convention of [RFC5040]:

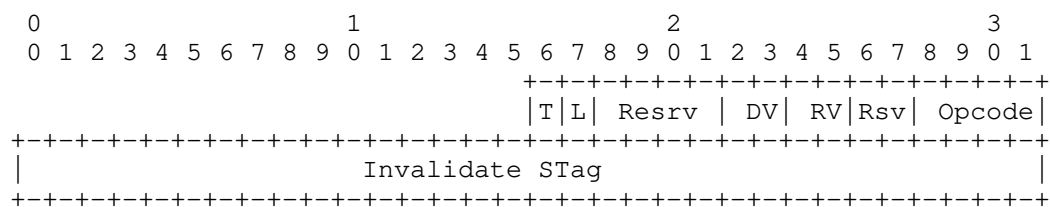


Figure 1 DDP Control and RDMAP Control Fields

The RDMAP Version (RV) field in the RDMAP Control Field when the set of extensions specified in this document is implemented MUST be 01b.

Additionally new RDMA Message Operation Codes are added for the Atomic and Immediate Data operations as shown in Figure 2.

RDMA Message OpCode	Message Type	Tagged Flag	STag and TO	Queue Number	Invalidate STag	Message Length Communicated between DDP and RDMAP
1000b	Immediate Data	0	N/A	0	N/A	Yes
1001b	Immediate Data with SE	0	N/A	1	N/A	Yes
1010b	Atomic Request	0	N/A	1	N/A	Yes
1011b	Atomic Response	0	N/A	1	N/A	Yes

Figure 2 Additional RDMA Usage of DDP Fields

Note: N/A means Not Applicable.

All other DDP and RDMAP control fields MUST be set as described in RFC5040 [RFC5040].

#### 4.2. RDMA Message Definitions

The following figure defines which RDMA Headers MUST be used on each new RDMA Message and which new RDMA Messages are allowed to carry ULP payload:

RDMA Message OpCode	Message Type	RDMA Header Used	ULP Message allowed in the RDMA Message
1000b	Immediate Data	Immediate Data Header	No
1001b	Immediate Data with SE	Immediate Data Header	No
1010b	Atomic Request	Atomic Request Header	No
1011b	Atomic Response	Atomic Response Header	No

Figure 3 RDMA Message Definitions

## 5. Atomic Operations

The RDMA Protocol Specification in [RFC4050] does not include support for atomic operations which are an important building block for implementing distributed shared memory.

This document extends the RDMA Protocol specification with a set of basic atomic operations, and specifies their resource and ordering rules.

Atomic operations as specified in this document execute a 64-bit operation at a specified destination address on a remote node. The operations atomically read, modify and write back the contents of the destination address and guarantee that atomic operations on this address by other Queue Pairs (QPs) on the same RNIC do not occur between the read and the write. Atomic operations as specified in this document MAY be implemented. The discovery of whether the atomic operations are implemented or not is outside the scope of this specification and it should be handled by the ULPs or applications.



Implementation note: It is recommended that the applications do not use the buffer addresses used for atomic operations for other RDMA operations.

Atomic operations use the same remote addressing mechanism as RDMA Reads and Writes. The buffer address specified in the request is in the address space of the Remote Peer that the atomic operation is targeted at.

## 5.1. Atomic Operation Details

The following sub-sections describe the atomic operations in more details.

### 5.1.1. FetchAdd

The FetchAdd atomic operation requests the responder to read a 64-bit Original Remote Data value at a naturally aligned buffer address in the responder's memory, to perform FetchAdd operation on multiple fields of selectable length specified by 64-bit "Add Mask", and write the result back to the same virtual address. The Atomic addition is performed independently on each one of these fields. A bit set in the Add Mask field specifies the field boundary. The FetchAdd atomic operation result is unknown when the buffer address is not naturally aligned. The setting of "Add Mask" field to 0x0000000000000000 results in Atomic Add of 64-bit Original Remote Data Value and 64-bit "Add Data".

The pseudo code below describes masked FetchAdd atomic operation.

```
bit_location = 1

carry = 0

Remote Data Value = 0

for bit = 0 to 63
{
    if (bit != 0 ) bit_location = bit_location << 1

    val1 = !(!(Original Remote Data Value & bit_location))

    val2 = !(!(Add Data & bit_location))
```

```
    sum = carry + val1 + val2

    carry = !(sum & 2)

    sum = sum & 1

    if (sum)

        Remote Data Value |= bit_location

    carry = ((carry) && (!(Add Mask & bit_location)))

}
```

The FetchAdd operation is performed in the endian format of the target memory. The "Original Remote Data" is converted from the endian format of the target memory for return and returned to the requester. The fields are in big-endian format on the wire.

The requester specifies:

- o Remote STag
- o Remote Tagged Offset
- o Add Data
- o Add Mask

The responder returns:

- o Original Remote Data

#### 5.1.2. Swap

The Swap Atomic Operation requires the responder to read a 64-bit value at a naturally aligned buffer address in the responder's memory, then to write the "Swap Data" fields into the same buffer address. The "Original Remote Data" is converted from the endian format of the target memory for return and returned to the requester. The fields are in big-endian format on the wire.

The requester specifies:

- o Remote STag

- o Remote Tagged Offset
- o Swap Data

The responder returns:

- o Original Remote Data

After the successful completion of Swap operation, the responder's memory at the specified buffer address contains the "Swap Data" field in the header. The Swap atomic operation result is unknown when the buffer address is not naturally aligned.

#### 5.1.3. CmpSwap

The CmpSwap Atomic Operation requires the responder to read a 64-bit value at a naturally aligned buffer address in the responder's memory, to perform an AND logical operation using the 64 bit "Compare Mask" field in the atomic operation Request header, then to compare it with the result of a logical AND operation of the "Compare Mask" and the "Compare Data" fields in the header, and, if the two values are equal, to swap masked bits in the same buffer address with the masked Swap Data. If the two masked compare values are not equal, the contents of the responder's memory are not changed. In either case, the original value read from the buffer address is converted from the endian format of the target memory for return and returned to the requester. The fields are in big-endian format on the wire.

The requester specifies:

- o Remote STag
- o Remote Tagged Offset
- o Swap Data
- o Swap Mask
- o Compare Data
- o Compare Mask

The responder returns:

- o Original Remote Data Value

The following pseudo code describes the masked CmpSwap operation result.

```
if (!((Compare Data ^ Original Remote Data value) & Compare Mask)
then
    Remote Data Value =
        (Original Remote Data Value & ~(Swap Mask))
        | (Swap Data & Swap Mask)
else
    Remote Data Value = Original Remote Data Value
```

After the operation, the remote data buffer SHALL contain the "Original Remote Data Value" (if comparison did not match) or the masked "Swap Data" (if the comparison did match). The CmpSwap atomic operation result is unknown when the buffer address is not naturally aligned.

## 5.2. Atomic Operations

The Atomic Operation Request and Response are RDMA Messages. An Atomic Operation makes use of the DDP Untagged Buffer Model. Atomic Operations use the same Queue Number as RDMA Read Requests (QN=1). Reusing the same Queue Number allows the Atomic Operations to reuse the same infrastructure (e.g. ORD/IRD flow control) as defined for RDMA Read Requests.

The RDMA Message OpCode for an Atomic Request Message is 1010b. The RDMA Message OpCode for an Atomic Response Message is 1011b.

### 5.2.1. Atomic Operation Request Message

The Atomic Operation Request Message carries an Atomic Operation Header that describes the buffer address in the responder's memory. The Atomic Operation Request header immediately follows the DDP header. The RDMAP layer passes to the DDP layer a RDMAP Control Field. The following figure depicts the Atomic Operation Request Header that MUST be used for all Atomic Operation Request Messages:

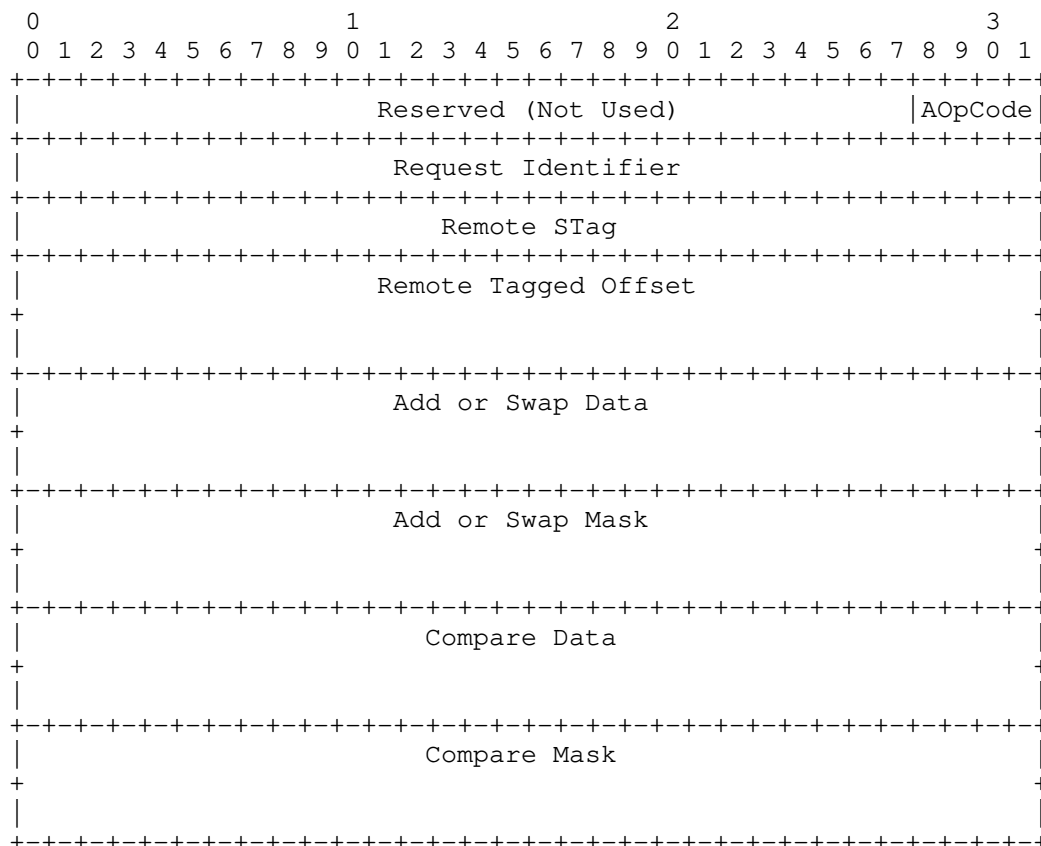


Figure 4 Atomic Operation Request Header

Reserved (Not Used): 28 bits

This field MUST be set to zero on transmit, ignored on receive.

Atomic Operation Code (AOpCode): 4 bits.

See Figure below.

Request Identifier: 32 bits.

The Request Identifier specifies a number that is used to identify Atomic Operation Request Message. The use of this field is implementation dependent and outside the scope of this specification.

Remote STag: 32 bits.

The Remote STag identifies the Remote Peer's Tagged Buffer targeted by the atomic operation. The Remote STag is associated with the RDMA Stream through a mechanism that is outside the scope of the RDMA specification.

Remote Tagged Offset: 64 bits.

The Remote Tagged Offset specifies the starting offset, in octets, from the base of the Remote Peer's Tagged Buffer targeted by the atomic operation. The Remote Tagged Offset MAY start at an arbitrary offset.

Add or Swap Data: 64 bits.

The Add or Swap Data field specifies the 64-bit "Add Data" value in an Atomic FetchAdd Operation or the 64-bit "Swap Data" value in an Atomic Swap or CmpSwap Operation.

Add or Swap Mask: 64 bits

This field is used in masked atomic operations (FetchAdd and CmpSwap) to perform a bitwise logical AND operation as specified in the definition of these operations. For non-masked atomic operations (Swap), this field MUST be set to ffffffff on transmit and ignored by the receiver.

Compare Data: 64 bits.

The Compare Data field specifies the 64-bit "Compare Data" value in an Atomic CmpSwap Operation. For Atomic FetchAdd and Atomic Swap operation, the Compare Data field MUST be set to zero on transmit and ignored by the receiver.

Compare Mask: 64 bits

This field is used in masked atomic operation CmpSwap to perform a bitwise logical AND operation as specified in the definition of these operations. For atomic operations

FetchAndAdd and Swap, this field MUST be set to ffffffffh on transmit and ignored by the receiver.

Atomic Operation OpCode	Atomic Operation	Add or Swap Data	Add or Swap Mask	Compare Data	Compare Mask
0000b	FetchAdd	Add Data	Add Mask	N/A	N/A
0001b	Swap	Swap Data	N/A	N/A	N/A
0010b	CmpSwap	Swap Data	Swap Mask	Valid	Valid
0011b to 1111b	Reserved	Not Specified			

Figure 5 Atomic Operation Message Definitions

The Atomic Operation Request Message has the following semantics:

1. An Atomic Operation Request Message MUST reference an Untagged Buffer. That is, the Local Peer's RDMA layer MUST request that the DDP mark the Message as Untagged.
2. One Atomic Operation Request Message MUST consume one Untagged Buffer.
3. The Remote Peer's RDMA layer MUST process an Atomic Operation Request Message. A valid Atomic Operation Request Message MUST NOT be delivered to the Data Sink's ULP (i.e., it is processed by the RDMA layer).
4. At the Remote Peer, when an invalid Atomic Operation Request Message is delivered to the Remote Peer's RDMA layer, an error is surfaced.
5. An Atomic Operation Request Message MUST reference the RDMA Read Request Queue. That is, the Local Peer's RDMA layer MUST request that the DDP layer set the Queue Number field to one.

6. The Local Peer MUST pass to the DDP layer Atomic Operation Request Messages in the order they were submitted by the ULP.
7. The Remote Peer MUST process the Atomic Operation Request Messages in the order they were sent.
8. If the Data Source receives a valid Atomic Operation Request Message, it MUST respond with a valid Atomic Operation Response Message.

#### 5.2.2. Atomic Operation Response Message

The Atomic Operation Response Message carries an Atomic Operation Response Header that contains the "Original Request Identifier" and "Original Remote Data Value". The Atomic Operation Response Header immediately follows the DDP header. The RDMAP layer passes to the DDP layer a RDMAP Control Field. The following figure depicts the Atomic Operation Response header that MUST be used for all Atomic Operation Response Messages:

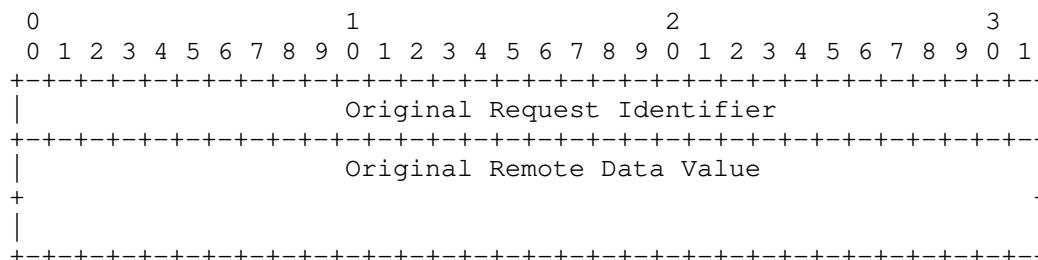


Figure 6 Atomic Operation Response Header

Original Request Identifier: 32 bits.

The Original Request Identifier MUST be set to the value specified in the Request Identifier field that was originally provided in the corresponding Atomic Operation Request Message.

Original Remote Data Value: 64 bits.

The Original Remote Value specifies the original 64-bit value stored at the buffer address targeted by the atomic operation.

The Atomic Operation Response Message has the following semantics:



1. The Atomic Operation Response Message for the associated Atomic Operation Request Message travels in the opposite direction.
2. An Atomic Operation Response Message MUST consume an Untagged Buffer. That is, the Data Source RDMAP layer MUST request that the DDP mark the Message as Untagged.
3. An Atomic Operation Response Message MUST reference the Queue Number 3. That is, the Local Peer's RDMAP layer MUST request that the DDP layer set the Queue Number field to 3.
4. The Data Source MUST ensure that a sufficient number of Untagged Buffers are available on the RDMA Read Request Queue (Queue with DDP Queue Number 1) to support the maximum number of Atomic Operation Requests negotiated by the ULP.
5. The RDMAP layer MUST Deliver the Atomic Operation Response Message to the ULP.
6. At the Remote Peer, when an invalid Atomic Operation Response Message is delivered to the Remote Peer's RDMAP layer, an error is surfaced.
7. The Data Source RDMAP layer MUST pass Atomic Operation Response Messages to the DDP layer, in the order that the Atomic Operation Request Messages were received by the RDMAP layer, at the Data Source.

### 5.3. Atomicity Guarantees

Atomicity of the RMW on the responder's node by the Atomic Operation SHALL be assured in the presence of concurrent atomic accesses by other QPs on the same RNIC.

### 5.4. Atomic Operations Ordering and Completion Rules

In addition to the ordering and completion rules described in RFC5040 [RFC5040], the following rules apply to implementations of the Atomic operations.

1. For an Atomic operation, the contents of the Tagged Buffer at the Data Sink MAY be indeterminate until the Atomic Operation Response Message has been Delivered at the Local Peer.

2. Atomic Operation Request Messages MUST NOT start processing at the Remote Peer until they have been Delivered to RDMAP by DDP.
  3. Atomic Operation Response Messages MAY be generated at the Remote Peer after subsequent RDMA Write Messages or Send Messages have been Placed or Delivered.
  4. Atomic Operation Response Message processing at the Remote Peer MUST be started only after the Atomic Operation Request Message has been Delivered by the DDP layer (thus, all previous RDMA Messages have been properly submitted for ordered Placement).
  5. Send Messages MAY be Completed at the Remote Peer (Data Sink) before prior incoming Atomic Operation Request Messages have completed their response processing.
  6. An Atomic Operation MUST NOT be Completed at the Local Peer until the DDP layer Delivers the associated incoming Atomic Operation Response Message.
  7. If more than one outstanding Atomic Request Messages are supported by both peers, the Atomic Operation Request Messages MUST be processed in the order they were delivered by the DDP layer on the Remote Peer. Atomic Operation Response Messages MUST be submitted to the DDP layer on the Remote Peer in the order the Atomic Operation Request Messages were Delivered by DDP.
6. Immediate Data

The Immediate Data operation is used in conjunction with an RDMA Write operation to improve ULP processing efficiency by allowing 8 bytes of immediate data which are placed in a Completion Queue Entry (CQE) after the previous operation has been delivered at the remote peer.

#### 6.1. RDMAP Interactions with the ULP for Immediate Data Operations

For Immediate Data operations, the following are the interactions between the RDMAP Layer and the ULP:

- . At the Data Source:
  - . The ULP passes to the RDMAP Layer the following:
    - . Eight bytes of ULP Immediate Data

- . When the Immediate Data operation Completes, an indication of the Completion results.
- . At the Data Sink:
  - . If the Immediate Data operation is Completed successfully, the RDMAP Layer passes the following information to the ULP Layer:
    - . Eight bytes of Immediate Data
    - . An Event, if the Data Sink is configured to generate an Event and the RDMA Message Opcode indicates Message Type Immediate Data with Solicited Event.
  - . If the Immediate Data operation is Completed in error, the Data Sink RDMAP Layer will pass up the corresponding error information to the Data Sink ULP and send a Terminate Message to the Data Source RDMAP Layer. The Data Source RDMAP Layer will then pass up the Terminate Message to the ULP.

## 6.2. Immediate Data Header Format

The Immediate Data and Immediate Data with SE Messages carry immediate data as shown in Figure 7. The RDMAP layer passes to the DDP layer an RDMAP Control Field and 8 bytes of Immediate Data. The first 8 bytes of the data following the DDP header contains the Immediate Data. See section A.3. for the DDP segment format of an Immediate Data or Immediate Data with SE Message.

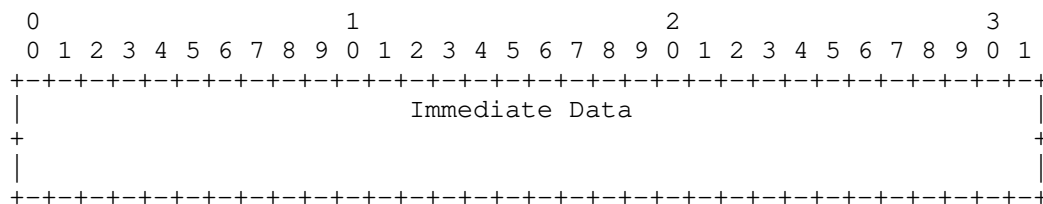


Figure 7 Immediate Data or Immediate Data with SE Message Header

Immediate Data: 64 bits.

Eight bytes of data transferred from the Requester to an untagged buffer at the Responder.

### 6.3. Immediate Data or Immediate Data with SE Message

The Immediate Data or Immediate Data with SE Message uses the DDP Untagged Buffer Model to transfer Immediate data from the Data Source to the Data Sink.

- . An Immediate Data or Immediate Data with SE Message MUST reference an Untagged Buffer. That is, the Local Peer's RDMAP Layer MUST request that the DDP layer mark the Message as Untagged.
- . One Immediate Data or Immediate Data with SE Message MUST consume one Untagged Buffer.
- . At the Remote Peer, the Immediate Data or Immediate Data with SE Message MUST be Delivered to the Remote Peer's ULP in the order they were sent.
- . For an Immediate Data or Immediate Data with SE Message, the Local Peer's RDMAP Layer MUST request that the DDP layer set the Queue Number field to zero.
- . For an Immediate Data or Immediate Data with SE Message, the Local Peer's RDMAP Layer MUST request that the DDP layer transmit 8 bytes of data.
- . The Local Peer MUST issue Immediate Data and Immediate Data with SE Messages in the order they were submitted by the ULP.
- . The Remote Peer MUST check that Immediate Data and Immediate Data with SE Messages include exactly 8 bytes of data from the DDP layer.

### 6.4. Ordering and Completions

Ordering and completion rules for Immediate Data are the same as those for a Send operation as described in section 5.5 of RFC 5040.

### 7. Ordering and Completions Table

The following table summarizes the ordering relationships for Atomic and Immediate Data operations from the standpoint of local Peer issuing the Operations. Note that in the table that follows, Send includes

Send, Send with Invalidate, Send with Solicited Event, and Send with Solicited Event and Invalidate. Also note that in the table below, Immediate Data includes Immediate Data and Immediate Data with Solicited Event.

First Operation	Second Operation	Placement Guarantee at Remote Peer	Placement Guarantee at Local Peer	Ordering Guarantee at Remote Peer
Immediate Data	Send	No Placement Guarantee between Send Payload and Immediate Data	Not Applicable	Completed in Order
Immediate Data	RDMA Write	No Placement Guarantee between RDMA Write Payload and Immediate Data	Not Applicable	Not Applicable
Immediate Data	RDMA Read	No Placement Guarantee between Immediate Data and RDMA Read Request	RDMA Read Response will not be Placed until Immediate Data is Placed at Remote Peer	RDMA Read Response Message will not be generated until Immediate Data has been Completed
Immediate Data	Atomic	No Placement Guarantee between Immediate Data and Atomic Request	Atomic Response will not be Placed until Immediate Data is Placed at Remote Peer	Atomic Response Message will not be generated until Immediate Data has been Completed

Immediate Data or Send	Immediate Data	No Placement Guarantee	Not Applicable	Completed in Order
RDMA Write	Immediate Data	No Placement Guarantee	Not Applicable	Immediate Data is Completed after RDMA Write is Placed and Delivered
RDMA Read	Immediate Data	No Placement Guarantee between Immediate Data and RDMA Read Request	Immediate Data may be Placed before RDMA Read Response is generated	Not Applicable
Atomic	Immediate Data	No Placement Guarantee between Immediate Data and Atomic Request	Immediate Data may be Placed before Atomic Response is generated	Not Applicable
Atomic	Send	No Placement Guarantee between Send Payload and Atomic Request	Send Payload may be Placed before Atomic Response is generated	Not Applicable
Atomic	RDMA Write	No Placement Guarantee between RDMA Write Payload and Atomic Request	RDMA Write Payload may be Placed before Atomic Response is generated	Not Applicable
Atomic	RDMA Read	No Placement Guarantee	No Placement Guarantee	RDMA Read Response

		between Atomic Request and RDMA Read Request	between Atomic Response and RDMA Read Response	Message will not be generated until Atomic Response Message has been generated
Atomic	Atomic	No Placement Guarantee between two Atomic Requests	No Placement Guarantee between two Atomic Responses	Second Atomic Response Message will not be generated until first Atomic Response has been generated
Send	Atomic	No Placement Guarantee between Send Payload and Atomic Request	Atomic Response will not be Placed at the Local Peer Until Send Payload is Placed at the Remote Peer	Atomic Response Message will not be generated until Send has been Completed
RDMA Write	Atomic	No Placement Guarantee between RDMA Write Payload and Atomic Request	Atomic Response will not be Placed at the Local Peer Until Send Payload is Placed at the Remote Peer	Not Applicable
RDMA Read	Atomic	No Placement Guarantee between Atomic Request and	No Placement Guarantee between Atomic Response	Atomic Response Message will not be generated until RDMA Read Response

		RDMA Read Request	and RDMA Read Response	has been generated
-----+-----+-----+-----+-----				

## 8. Error Processing

In addition to error processing described in section 7 of RFC 5040, the following rules apply for the new RDMA Messages defined in this specification.

### 8.1. Errors Detected at the Local Peer

The Local Peer **MUST** send a Terminate Message for each of the following cases:

1. For errors detected while creating an Atomic Request, Atomic Response, Immediate Data, or Immediate Data with SE Message, or other reasons not directly associated with an incoming Message, the Terminate Message and Error code are sent instead of the Message. In this case, the Error Type and Error Code fields are included in the Terminate Message, but the Terminated DDP Header and Terminated RDMA Header fields are set to zero.
2. For errors detected on an incoming Atomic Request, Atomic Response, Immediate Data, or Immediate Data with Solicited Event (after the Message has been Delivered by DDP), the Terminate Message is sent at the earliest possible opportunity, preferably in the next outgoing RDMA Message. In this case, the Error Type, Error Code, and Terminated DDP Header fields are included in the Terminate Message, but the Terminated RDMA Header field is set to zero.

### 8.2. Errors Detected at the Remote Peer

On incoming Atomic Requests, Atomic Responses, Immediate Data, and Immediate Data with Solicited Event, the following must be validated:

1. The DDP layer **MUST** validate all DDP Segment fields.
2. The RDMA OpCode **MUST** be valid.
3. The RDMA Version **MUST** be valid.



## 9. Security Considerations

This document specifies extensions to the RDMA Protocol specification in [RFC5040], and as such the Security Considerations discussed in Section 8 of [RFC5040] apply.

## 10. IANA Considerations

This document requests no direct action from IANA.

## 11. References

### 11.1. Normative References

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.

[RFC5040] Recio, R. et al., "A Remote Direct Memory Access Protocol Specification", RFC 5040, October 2007.

[RFC5041] Shah, H. et al., "Direct Data Placement over Reliable Transports", RFC 5041, October 2007.

### 11.2. Informative References

## 12. Acknowledgments

The authors would like to acknowledge the following contributors who provided valuable comments and suggestions.

- o Steve Wise.

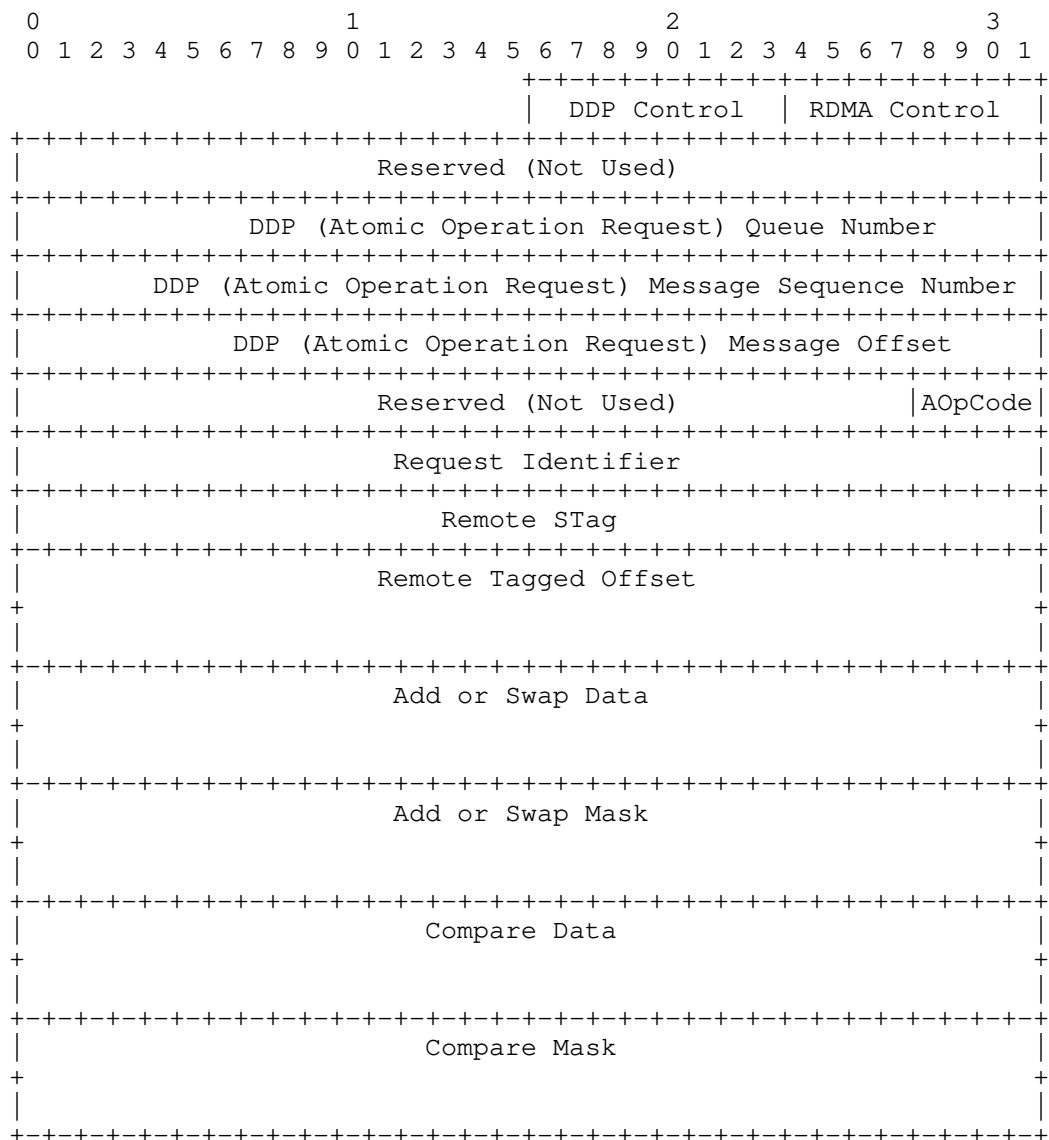
This document was prepared using 2-Word-v2.0.template.dot.

## Appendix A. DDP Segment Formats for RDMA Messages

This appendix is for information only and is NOT part of the standard. It simply depicts the DDP Segment format for the various RDMA Messages.

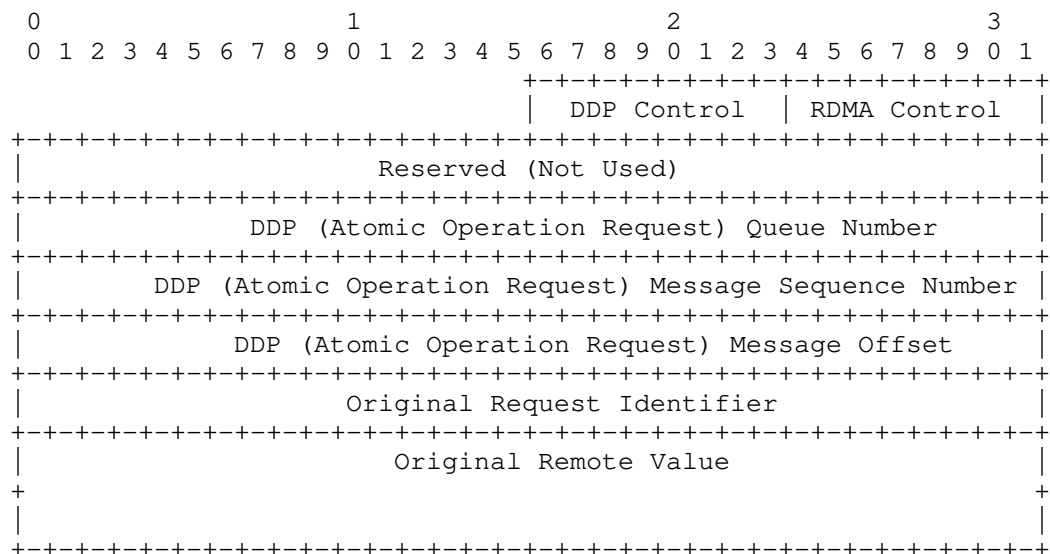
### A.1. DDP Segment for Atomic Operation Request

The following figure depicts an Atomic Operation Request, DDP Segment:



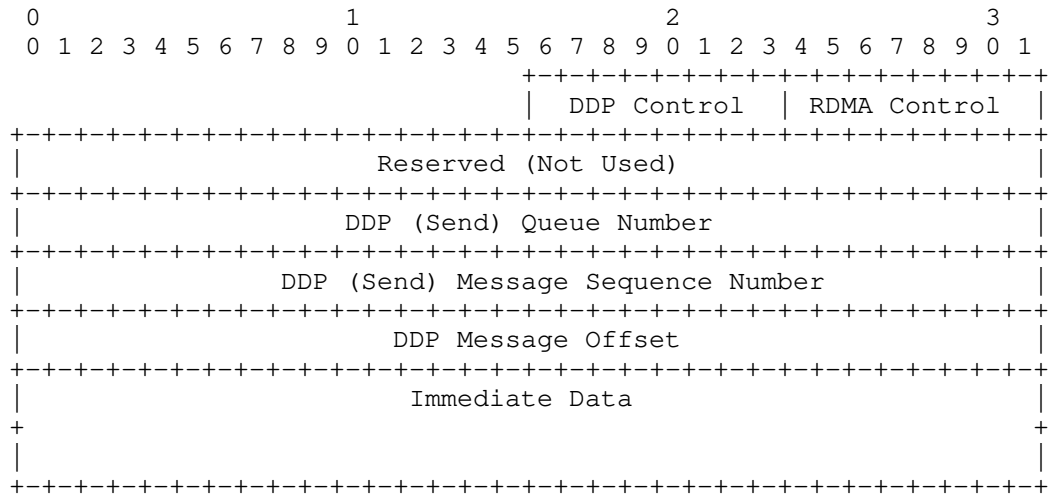
## A.2. DDP Segment for Atomic Response

The following figure depicts an Atomic Operation Response, DDP Segment:



## A.3. DDP Segment for Immediate Data and Immediate Data with SE

The following figure depicts an Immediate Data or Immediate data with SE, DDP Segment:



## Authors' Addresses

Hemal Shah  
Broadcom Corporation  
5300 California Avenue  
Irvine, CA 92617  
Phone: 1-949-926-6941  
Email: hemal@broadcom.com

Felix Marti  
Chelsio Communications, Inc.  
370 San Aleso Ave.  
Sunnyvale, CA 94085  
Phone: 1-408-962-3600  
Email: felix@chelsio.com

Asgeir Eiriksson  
Chelsio Communications, Inc.  
370 San Aleso Ave.  
Sunnyvale, CA 94085  
Phone: 1-408-962-3600  
Email: asgeir@chelsio.com

Wael Nouredidine  
Chelsio Communications, Inc.  
370 San Aleso Ave.  
Sunnyvale, CA 94085  
Phone: 1-408-962-3600  
Email: wael@chelsio.com

Robert Sharp  
Intel Corporation  
1501 South Mopac, Suite 400, Mailstop: AN1-WTR1  
Austin, TX 78746  
Phone: 1-512-493-3242  
Email: robert.o.sharp@intel.com

