IPS
Internet Draft
Document: draft-ietf-ips-iscsi-04.txt
Category: standards-track

Julian Satran
Daniel Smith
Kalman Meth
Ofer Biran
IBM

Costa Sapuntzakis
Cisco Systems

Matt Wakeley
Agilent Technologies

Luciano Dalle Ore
Quantum

Paul Von Stamwitz
Adaptec

Randy Haagens
Hewlett-Packard Co.

Efri Zeidner
SANGate

Yaron Klein
SANRAD

iSCSI

Status of this Memo

Abstract

The Small Computer Systems Interface (SCSI) is a popular family of protocols for communicating with I/O devices, especially storage devices. This memo describes a transport protocol for SCSI that operates on top of TCP. The iSCSI protocol aims to be fully compliant with the requirements laid out in the SCSI Architecture Model - 2 [SAM2] document.

Acknowledgements

Besides the authors a large group of people contributed through their review, comments and valuable insights to the creation of this document - too many to mention them all. Nevertheless, we are grateful to all of them. We are especially grateful to those that found the time and patience to participate in our weekly phone conferences and intermediate meetings in Almaden and Haifa and thus helped shape this document: Jim Hafner, John Hufferd, Prasenjit Sarkar, Meir Toledano, John Dowdy, Steve Legg, Alain Azagury (IBM), Dave Nagle (CMU), David Black (EMC), John Matze (Veritas), Mark Bakke, Steve DeGroote, Mark Shrandt (NuSpeed), Gabi Hecht (Gadzoox), Robert Snively (Brocade), Nelson Nachum (StorAge), Uri Elzur (Intel). Many more helped clean and improve this document within the IPS working group. We are especially grateful to David Robinson and Raghavendra Rao (Sun), Charles Monia, Joshua Tseng (Nishan), Somesh Gupta, Mallikarjun C., Michael Krause, Pierre Labat, Santosh Rao

(HP), Stephen Byan (Genroco), Robert Elliott (Compaq), Steve Senum

(CISCO), Barry Reinhold (Trebia Networks). Last, but not least, thanks to Ralph Weber for keeping us in-line with T10 (SCSI) standardization. Thanks also Steve Hetzler for his unwavering support and for coming up with such a good name for the protocol and Micky Rodeh, Jai Menon, Clod Barrera and Andy Bechtolsheim for helping this work happen.

Conventions used in this document

In examples, "I->" and "T->" indicate 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.

Table of Contents

1. Overview

1.1 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 to situate readers in the vocabulary of the SCSI architecture.

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 parlance, an individual I/O device is called a "logical unit" (LU).

SCSI is client-server architecture. Clients of a SCSI interface are called "initiators". Initiators issue SCSI "commands" to request service from a logical unit. The "device server" on the logical unit accepts SCSI commands and executes them.

A "SCSI transport" maps the client-server SCSI protocol to a specific interconnect. Initiators are one endpoint of a SCSI transport. The "target" is the other endpoint. A target can have multiple Logical Units (LUs) behind it. 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. The queue of tasks is managed by the target, though. 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. A response terminates a SCSI command. For performance reasons iSCSI allows a "phase-collapse" - e.g., command and its associated data may be shipped together from initiator to target and data and responses may be shipped together from targets.

Command Descriptor Blocks (CDB) is the data structure used to contain the command parameters to be handed by an initiator to a target. The CDB content and structure is defined by [SAM] and device-type specific SCSI standards.

1.2 iSCSI Concepts & Functional Overview

The iSCSI protocol is a mapping of the SCSI remote procedure invocation model on top of the TCP protocol.

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

iSCSI transfer direction is defined with regard to the initiator. Outbound or outgoing transfers are transfers from initiator to target while inbound or incoming transfers are from target to initiator.

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

1.2.1 Layers & Sessions

The following conceptual layering model is used in this document to specify initiator and target actions and how those relate to transmitted and received Protocol Data Units:

-the SCSI layer builds/receives SCSI CDBs (Command Descriptor Blocks) and relays/receives them with the remaining command execute parameters (cf. SAM-2) to/from the
-the iSCSI layer that builds/receives iSCSI PDUs and relays/receives them to/from - one or more TCP connections that form an initiator-target "session".

Communication between 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 linking an initiator with a target form a session (loosely equivalent to a SCSI I-T nexus). A session is defined by a session ID (composed of an initiator part and a target part). TCP connections can be added and removed from a session. Connections within a session are identified by a connection ID (CID).

Across all connections within a session, an initiator will see one "target image". All target identifying elements, like LUN are the same. In addition, across all connections within a session a target will see one "initiator image". Initiator identifying elements like Initiator Task Tag can be used to identify the same entity 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.

1.2.2 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 recovery in case of connection failure.

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

Normally, 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-message PDUs may be utilized to synchronize the command and status ordering counters of the target and initiator.

1.2.2.1 Command numbering and acknowledging

iSCSI supports ordered command delivery within a session. All commands (initiator-to-target) are numbered.

Any SCSI activity is related to a task (SAM-2). The task is identified by the Initiator Task Tag for the life of the task.

Commands in transit from the initiator SCSI layer to the SCSI target layer are numbered by iSCSI and the number is carried by the iSCSI PDU as CmdSN (Command-Sequence-Number). The numbering is session-wide. All iSCSI PDUs that have a task association carry this number. CmdSNs are allocated by the initiator iSCSI within a 32 bit unsigned counter (modulo 2**32). The value 0 is reserved and used to mean immediate delivery. Comparisons and arithmetic on CmdSN SHOULD use Serial Number Arithmetic as defined in [RFC1982] where SERIAL_BITS = 32.

The means by which the SCSI layer may request immediate delivery for a command or by which iSCSI will decide by itself to mark a PDU for immediate delivery are outside the scope of this document.

Using immediate delivery with some commands may have unexpected side effects. If used with Task Management commands those may get to the

SCSI task manager at the target before the tasks they where suppose to act upon.
Whenever those effects are undesirable connection allegiance or ordered delivery MAY be used.

CmdSNs are significant only during command delivery to the target. Once the device serving part of the target SCSI has received a command, CmdSN ceases to be significant. During command delivery to the target, the allocated numbers are unique session wide.

The iSCSI target layer MUST deliver the commands to the SCSI target layer in the order specified by CmdSN.

The initiator and target are assumed to have three counters that define the numbering mechanism

- CmdSN - the current command Sequence Number advanced by 1 on each command shipped
- ExpCmdSN - the next expected command by the target - acknowledges all commands up to it
- MaxCmdSN - the maximum number to be shipped - MaxCmdSN - ExpCmdSN defines the queuing capacity of the receiving iSCSI layer.

The target MUST NOT transmit a MaxCmdSN that is more than 2**31 - 1 above the last ExpCmdSN. CmdSN can take any value from ExpCmdSN to MaxCmdSN except 0. The target MUST silently ignore any command outside this range or duplicates within the range not flagged with the retry bit (the X bit in the opcode).

iSCSI initiators and target MUST support the command numbering scheme.

1.2.2.2 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.

Status numbering starts after Login. During login, there is always only one outstanding command per connection and status numbering is not needed.
The login response includes an initial value for status numbering.

To enable command recovery the target MAY maintain enough state to enable data and status recovery after a connection failure.
A target can discard all the state information maintained for recovery after the status delivery is acknowledged through ExpStatSN.
A large difference between StatSN and ExpStatSN may indicate a failed connection.

Initiators and Targets MUST support the response-numbering scheme.

1.2.2.3 Data Sequencing

Data PDUs that are transferred as part of some command execution MUST be sequenced. The DataSN field is used for data sequencing. For input (read) data PDUs DataSN will start with 0 for the first data PDU and advance by 1 for each subsequent data PDU. For output data, PDUs DataSN will start with 0 for the first data PDU of a sequence (the initial unsolicited sequence or any data PDU sequence issued to satisfy a R2T) and advance by 1 for each subsequent data PDU. Unlike command and status the data PDUs are not acknowledged except as implied by status. The DataSN field is meant to enable the initiator to detect missing data PDUs and simplify this operation at the target. 0x'ffffffff' is not a valid DataSN and MUST be skipped when counting (serial arithmetic).

1.2.3 iSCSI Login

The purpose of iSCSI login is to enable a TCP connection for iSCSI use, authenticate the parties, negotiate the session's parameters,

open a security association protocol and mark 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. If an initiator and target are connected through more than one session, each of the initiator and target perceives the other as a different entity on each session (a different I_T nexus in SAM-2 parlance).

The targets listen on a well-known TCP port for incoming connections. The initiator begins the login process by connecting to that well-known TCP port.

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

Negotiation and security associations executed before the Login Command are outside the scope of this document although they might realize a related function (e.g., establish a IPsec tunnel).

The Login Command starts the iSCSI Login Phase. Within the Login Phase, negotiation is carried on through parameters of the Login Command and Response and optionally through intervening Text Commands and Responses. The Login Response concludes the Login Phase. Once suitable authentication has occurred, the target MAY authorize the initiator to send SCSI commands. How the target chooses to authorize an initiator is beyond the scope of this document. The target indicates a successful authentication and authorization by sending a login response with "login accept". Otherwise, it sends a response with a "login reject", indicating a session is not established.

It is expected that iSCSI parameters will be negotiated after the security association protocol is established, if there is a security association.

The login message includes a session ID - composed with an initiator part ISID and a target part TSID. For a new session, the TSID is null. As part of the response, the target will generate a TSID. Session specific parameters can be specified only for the first login of a session (TSID null)(e.g., the maximum number of connections that can be used for this session). Connection specific parameters (if any) can be specified for any login. Thus, a session is operational once it has at least one connection.

Any message except login and text sent on a TCP connection before this connection gets into full feature phase at the initiator SHOULD be ignored by the initiator. Any message except login and text reaching a target on a TCP connection before the full feature phase MUST be silently ignored by the target.

1.2.4 Text mode negotiation

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

In "list" negotiation, the offering party will send a list of values for a key in its order of preference.

The responding party will answer with a value from the list.

The value "none" MUST always be used to indicate a missing function. However, none is a valid selection only if it was explicitly offered and it MAY be selected by omission (i.e. <key>=none MAY be omitted).

The general format is:

```
Offer-> <key>=<value1>,<value2>,...,<valuen>
Answer-> <key>=<valuex>
```

In "numerical" negotiations, the offering and responding party state a numerical value. The result of the negotiation is key dependent (usually the lower or the higher of the two values).

1.2.5 iSCSI Full Feature Phase

Once the initiator is authorized to do so, the iSCSI session is in iSCSI full feature phase. The initiator may send SCSI commands and data to the various LUs on the target by wrapping them in iSCSI messages that go over the established iSCSI session.

For SCSI commands that require data and/or parameter transfer, the (optional) data and the status for a command must be sent over the same TCP connection that was used to deliver the SCSI command (we call this "connection allegiance"). 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 and the target MUST return R2T, if any, and the status over the same TCP connection that was used to deliver the SCSI command.

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

Outgoing SCSI data (initiator to target - user data or command parameters) will be sent as either solicited data or unsolicited data. Solicited data are sent in response to Ready To Transfer (R2T) PDUs. Unsolicited data can be sent as part of an iSCSI command PDU ("immediate data") or in separate iSCSI data PDUs. An initiator may send unsolicited data either as immediate (up to the negotiated maximum PDU size - DataPDULength - disconnect-reconnect mode page) or in a separate PDU sequence (up to the negotiated limit - FirstBurstSize - disconnect-reconnect mode page). All subsequent data have to be solicited. The maximum size of an individual data PDU or

the immediate-part of the initial unsolicited burst as well as the initial burst size MAY be negotiated at login.

Targets operate in either solicited (R2T) data mode or unsolicited (non R2T) data mode. A target MAY separately enable immediate data without enabling the more general (separate data PDUs) form of unsolicited data.

An initiator MUST always honor an R2T data request for a valid outstanding command (i.e., carrying a valid Initiator Task Tag) and provided the command is supposed to deliver outgoing data and the R2T specifies data within the command bounds.

It is considered an error for an initiator to send unsolicited data PDUs to a target operating in R2T mode (only solicited data). It is also an error for an initiator to send more data, whether immediate or as separate PDUs, than the SCSI limit for initial burst. An initiator MAY request, at login, to send data blocks and an initial burst of any size; in this case the target MUST indicate the size of the initial burst and of the immediate and data blocks, it is ready to accept. The agreed upon limits for the initial burst as well as the maximum data PDU will be recorded (and are retrievable from) the disconnect-reconnect mode page.

A target SHOULD NOT silently discard data and request retransmission

through R2T. Initiators MUST NOT perform any score boarding for data and the residual count calculation is to be performed by the targets. Incoming data is always implicitly solicited. SCSI Data packets are matched to their corresponding SCSI commands by using Tags that are specified in the protocol.

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 those will be 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. The above mechanisms are designed to accomplish efficient data delivery and a large degree of control over the data flow.

iSCSI initiators and targets MUST also enforce some ordering rules to achieve deadlock-free operation. Unsolicited data MUST be sent on every connection in the same order in which commands were sent. If the amount of data exceeds the amount allowed for unsolicited write data, the specific connection MUST be stalled - i.e., no more unsolicited data will be sent on this connection until the specific command has finished sending all its data and has received a

response. However, new commands as well as solicited data can be sent on the stalled connection. A target receiving data out of order or observing a connection violating the above rules SHOULD terminate the session.

Each iSCSI session to a target is treated as if it originated from a different and logically independent initiator.

1.2.6 iSCSI Connection Termination

Connection termination is assumed an exceptional event.
Graceful TCP connection shutdowns are done by sending TCP FINs. Graceful connection shutdowns MUST only occur when there are no outstanding tasks that have allegiance to the connection. A target SHOULD respond rapidly to a FIN from the initiator by closing it's half of the connection after waiting for all outstanding tasks that have allegiance to the connection to conclude and send their status. Connection termination with outstanding tasks may require recovery actions.

Connection termination is also required as prelude to recovery. By terminating a connection before starting recovery, initiator and target can avoid having stale PDUs being received after recovery. In this case, the initiator will send a LOGOUT request on any of the

operational connections of a session indicating what connection should be terminated.

LOGOUT can also be issued by an initiator at the explicit request of a target (through an Asynchronous Event PDU).

1.2.7 Naming and addressing

This section provides a summary of the naming and addressing mechanisms used in iSCSI. More details are provided in a separate document [NDT].

All iSCSI initiators and targets are named. Each target or initiator is known by a World-Wide Unique Identifier (WWUI). The WWUI is independent of the location of the initiator and target, and various formats are provided for naming authorities to use when generating them. A special format of the ubiquitous internet domain name can be used as a name; it is important not to confuse this with an address. The WWUI is a UTF-8 text string, and its structure is defined in [NDT].

WWUIs are used in iSCSI to:

- Provide a target identifier for configurations that present multiple targets behind a single IP address and port.
- Provide a method to recognize multiple paths to the same device on different IP addresses and ports.
- Provide a symbolic address for source and destination targets for use in third party commands.
- Provide an identifier for initiators and targets to enable them to recognize each other regardless of IP address and port mapping on intermediary firewalls.

The initiator MUST present both its initiator WWUI and the target WWUI to which it wishes to connect during the login phase.

A target MAY also provide a canonical WWUI called "iSCSI". This is not a globally unique name. An initiator can log into this canonical target WWUI, and use a text command called "SendTargets" to retrieve a list of WWUIs that exist at that address.

To iSCSI the WWUI is an opaque object.

Beside names, iSCSI targets also have addresses. An iSCSI address

specifies a single path to an iSCSI target. The WWUI is part of the address. An iSCSI address is given in an URL-like form, such as:

```
<domain-name>[:<port>]/<wwui>
```

Where <domain-name> is one of:

- IPv4 address, in dotted decimal notation. Assumed if the name contains exactly four numbers, separated by dots (.), where each number is in the range 0..255.
- IPv6 address, in dotted decimal notation. Assumed if the name contains more than four, but at most 16 numbers, separated by dots (.), where each number is in the range 0..255.
- Fully Qualified Domain Name (FQDN - host name). Assumed if the <domain-name> is neither an IPv4 nor an IPv6 address.

and <wwui> is the WWUI of the target being addressed.

The <port> in the address is optional; it specifies the TCP port on which the target is listening for connections. If <port> is not specified, the well-known iSCSI target port is assumed.

The iSCSI address, or URL, is not generally used within normal connections between iSCSI initiators and targets; it is primarily used during discovery. Details are specified in [NDT].

Examples of Worldwide Unique Identifiers:

```
com.disk-vendor.diskarrays.sn.45678
com.gateways.yourtargets.24
com.os-vendor.plan9.cdrom.12345
com.service-provider.users.customer235.host90
```

Examples of IPv4 addresses/names:

```
10.0.0.1/com.disk-vendor.diskarrays.sn.45678
10.0.0.2/iscsi
```

Examples of IPv6 addresses/names:

```
12.5.7.10.0.0.1/com.gateways.yourtargets.24
12.5.6.10.0.0.2/iscsi
```

For management/support tools, as well as naming services, that use a

text prefix to express the protocol intended (as in http:// or ftp://) the following form MAY be used:

```
iSCSI://<domain-name>[:port][/wwui]
```

Examples:

```
iSCSI://diskfarm1.acme.com/iscsi
iSCSI://computingcenter.acme.com/com.disk-
vendor.diskarrays.sn.45678
iSCSI://computingcenter.acme.com:4002/com.gateways.yourtargets.
24
```

To provide a friendlier (to the humans!) user interface for devices containing iSCSI targets and initiators, a target or initiator may also provide an alias. This alias is a simple UTF-8 string, is not globally unique, and is never interpreted or used to identify an initiator or device within the iSCSI protocol. Its use is described in [NDT].

When a target has to act as an initiator for a third party command, it MAY use the initiator WWUI it learned during login as required by the authentication mechanism to the third party.

To address targets and logical units within a target, SCSI uses a fixed length (8 bytes) uniform addressing scheme; in this document, we call those addresses SCSI reference addresses (SRA).

To provide the target with the protocol specific addresses iSCSI relies on the SCSI aliasing mechanism (work in progress in T10). The aliasing support enables an initiator to associate protocol specific addresses with SRAs; the later can be used in subsequent commands.

For iSCSI, a protocol specific address is a TCP address and a WWUI.

An initiator may use one of a few techniques to configure and/or discover the target WWUIs to which it has access, along with their addresses. These techniques are discussed fully in [NDT].

1.2.8 Message Synchronization and Steering

1.2.8.1 Rationale

iSCSI presents a mapping of the SCSI protocol onto TCP. This encapsulation is accomplished by sending iSCSI PDUs that are of

varying length. 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 (messages are processed one after the other). In the presence of IP packet reordering (e.g. 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.

Unfortunately, when relying solely on the "message length in the iSCSI message" scheme to delineate iSCSI messages, a missing TCP segment that contains an iSCSI message header (with the message length) makes it impossible to find message boundaries in subsequent TCP segments. 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 (one of them is detailed in an Appendix). To make those schemes work iSCSI implementations have to make sure that the appropriate protocol layers are provided with enough information to implement a synchronization and/or data steering mechanism.

1.2.8.2 Synch and Steering functional model

We assume that iSCSI is implemented according to the following layering scheme:

```
+-----------------------------------+
|             SCSI                  |
+-----------------------------------+
|            iSCSI                  |
+-----------------------------------+
|     Synch and Steering            |
```

```
+-----------------------------------+
|              TCP                  |
+-----------------------------------+
|   Lower Functional Layers (LFL)   |
+-----------------------------------+
|              IP                   |
+-----------------------------------+
|             Link                  |
+-----------------------------------+
```

In this model LFL can be IPsec (a mechanism changing the IP stream and invisible to TCP). We assume that Synch and Steering operates just underneath iSCSI. Please note that an implementation may choose to place Synch and Steering somewhere else in the stack provided that it can translate the information kept by iSCSI in terms valid for the chosen layer.

According to our model of layering iSCSI considers the information it delivers (headers and payloads) as a contiguous stream of bytes mapped to the positive integers from 0 to infinity. For all practical purposes iSCSI is not supposed to have to handle infinitely long streams and the stream addressing scheme will wrap around at 2**32-1.

It is also assumed that iSCSI will deliver to the layers beneath any PDU through an indivisible (atomic) operation. If a specific implementation does PDU delivery to the Synch and Steering layer

through multiple operations it MUST bracket an operation set used to deliver a single PDU in a manner understandable to the Synch and Steering Layer.

The Synch and Steering Layer (that itself is OPTIONAL) MUST retain for every delivered iSCSI PDU the PDU end address within the stream. To enable the Synch and Steering operation to perform Steering some additional information including identifying tags, and buffer offsets MUST be retained as well for every sent PDU. Those will be required to add to every sent data item (IP packet, TCP packet or some other superstructure) enough information to enable the receiver to steer it to a memory location independent of any other piece.

If the transmission stream is built dynamically this information will be used to insert Synch and Steering information in the transmission stream (at first transmission or at re-transmission) either trough a globally accessible table or through a call-back mechanism. If the transmission stream is built statically, the Synch and Steering information is just inserted in the transmission stream.

The retained information can be released whenever the transmitted data is acknowledged by the receiver (in case of dynamically built streams by deletion from the global table or by an additional callback).

On the outgoing path, the Synch and Steering layer MUST map the outgoing stream addresses from iSCSI stream addresses to TCP stream sequence numbers.

On the incoming path, the Synch and Steering layer will extract the Synch & Steering information from the TCP stream and help deliver (steer) the data stream into its final address and/or recover iSCSI PDU boundaries when some TCP packets are lost or received out of order. The data stream seen by the receiving iSCSI layer is identical to the data stream that left the sending iSCSI layer.

1.2.8.3 Synch and Steering and other encapsulation layers

We recognize that in many environments a more appropriate layering model would be the following:

```
+-----------------------------------+
|             SCSI                  |
+-----------------------------------+
|            iSCSI                  |
+-----------------------------------+
|   Upper Functional Layers (UFL)   |
+-----------------------------------+
|     Synch and Steering            |
+-----------------------------------+
|              TCP                  |
+-----------------------------------+
```

```
|   Lower Functional Layers (LFL)  |
+----------------------------------+
|              IP                  |
+----------------------------------+
|             Link                 |
+----------------------------------+
```

In this model UFL can be TLS or some other transport conversion mechanism (a mechanism changing the TCP stream but transparent to iSCSI).

To be effective and act on reception of TCP packets out of order Synch and Steering has to be underneath UFL and Synch and Steering data have to be left out of any UFL transformation (encryption, compression, padding etc.). However, Synch and Steering MUST take into account the additional data inserted in the stream by UFL. Synch and Steering MAY also restrict the type of transformations UFL may do on the stream.

This makes implementation of Synch and Steering in the presence of otherwise opaque UFLs less attractive.

1.2.8.4 Synch/Steering and iSCSI PDU Size

When a large iSCSI message is sent, the TCP segment(s) containing the iSCSI header may be lost. The remaining TCP segment(s) up to the next iSCSI message need to be buffered (in temporary buffers), since the iSCSI header that indicates what SCSI buffers, the data is to be steered to was lost. To minimize the amount of buffering, it is

recommended that the iSCSI PDU size be restricted to a small value (perhaps a few TCP segments in length). Each end of the iSCSI session specifies during login the maximum size of an iSCSI PDU it will accept.

2. iSCSI PDU Formats

All multi-byte integers specified in formats defined in this document are to be represented in network byte order (i.e., big endian). Any bits not defined MUST be set to zero. Any reserved fields and values MUST be 0 unless specified otherwise.

2.1 iSCSI PDU length and padding

iSCSI PDUs are padded to an integer number of 4 byte words.

2.2 PDU Template, Header and Opcodes

All iSCSI PDUs begin with one or more header segments followed by 0 or 1 data segments. After the entire header segment group there MAY

be a header-digest. The data segment MAY also be followed by a data-digest.

The first segment - and in many cases the only segment - (Basic Header Segment or BHS) is a fixed-length 44-byte header segment. It may be followed by Additional Header Segments (AHS). Each segment is preceded by a 4 byte Next-Qualifier. Thus, when we have only a BHS (with no data or digests) the net size of the iSCSI PDU is 48 bytes.

The overall structure of a PDU is:

```
Byte /    0       |       1       |       2       |       3       |
   /              |               |               |               |
  |7 6 5 4 3 2 1 0|7 6 5 4 3 2 1 0|7 6 5 4 3 2 1 0|7 6 5 4 3 2 1 0|
  +---------------+---------------+---------------+---------------+
 0| WN            |WN specific fields                             |
  +---------------+---------------+---------------+---------------+
 4/ BHS                                                          /
 +/                                                              /
  +---------------+---------------+---------------+---------------+
48| WN            |WN specific fields                             |
  +---------------+---------------+---------------+---------------+
52/ AHS                                                          /
 +/                                                              /
  +---------------+---------------+---------------+---------------+
----
  +---------------+---------------+---------------+---------------+
 m/ Header-Digest (optional)                                      /
 +/                                                               /
```



```
  +---------------+---------------+---------------+---------------+
 n/ Data Segment(optional)                                        /
 +/                                                               /
  +---------------+---------------+---------------+---------------+
 m/ Data-Digest (optional)                                        /
 +/                                                               /
  +---------------+---------------+---------------+---------------+
```

All PDU segments and digests are padded to an integer number of 4 byte words.

2.2.1 What's Next (WN)

This is an encoded field indicating what is the next segment as

follows:

```
   bit 7  - 0  Next is another header segment.
      bit 6-4  Next header type code
            0  Extended CDB
            1  Bi-directional read-data transfer header
            2  Long Data Header
          3,4,5,6,7  Reserved
   bit 7  - 1  Next is a data segment or no additional segment
   (empty data segment)
      bit 6-4  Reserved
   bit 3-2  Digest info for THIS segment
            0  No digest follows THIS segment
            1  A CRC-32Q digest follows THIS segment
          2,3  Reserved
   bit 1-0  Digest info for NEXT segment
            0  No digest follows NEXT segment
            1  A CRC-32Q digest follows NEXT segment
            2  A CRC-64 digest follows NEXT segment
            3  Reserved
```

N.B. An empty data segment MUST NOT be followed by a digest.
N.B. A digest MUST NOT follow a segment that is followed by another header segment in the same PDU (i.e., only the last header segment MAY be followed by a digest).

2.2.2 WN specific fields

These fields carry information specific to the next segment type.

2.2.2.1 WN specific fields for a next Extended CDB header segment

```
   Byte /    0       |       1       |       2       |       3       |
```



```
      /              |               |               |               |
     |7 6 5 4 3 2 1 0|7 6 5 4 3 2 1 0|7 6 5 4 3 2 1 0|7 6 5 4 3 2 1 0|
     +---------------+---------------+---------------+---------------+
    0| WN            | Reserved                      | AddCDB        |
     +---------------+---------------+---------------+---------------+
```

Where AddCDB is the additional CDB length in units of 4 byte words beyond the first extension word (i.e., AddCDB 0 means a 20 byte CDB, 1 a 24 byte etc.).

2.2.2.2 WN specific fields for next Bi-directional read data header segment and Long Data Transfer Header

```
Byte /    0       |       1       |       2       |       3       |
   /              |               |               |               |
  |7 6 5 4 3 2 1 0|7 6 5 4 3 2 1 0|7 6 5 4 3 2 1 0|7 6 5 4 3 2 1 0|
  +---------------+---------------+---------------+---------------+
 0| WN            | Reserved                                      |
  +---------------+---------------+---------------+---------------+
```

2.2.2.3 WN specific fields for next Data header segment

```
Byte /    0       |       1       |       2       |       3       |
   /              |               |               |               |
  |7 6 5 4 3 2 1 0|7 6 5 4 3 2 1 0|7 6 5 4 3 2 1 0|7 6 5 4 3 2 1 0|
  +---------------+---------------+---------------+---------------+
 0| WN            | Data Length or Reserved                       |
  +---------------+---------------+---------------+---------------+
```

Whenever this is the Next-Qualifier of a Long Data Header or a Long Data Header appeared earlier, in the sequence the data length field is ignored and the data length is taken from within this long-data-header (a 32 bit field). Else, the length field is the data length. Without a Long Data Header the maximum length of a data segment is 16Mbytes.

2.2.3 Header Digest and Data Digest

Optional header and data digests protect the integrity and authenticity of header and data, respectively. The digests, if present, appear as trailers located, respectively, after the header and PDU-specific data.

The digest types are negotiated during the login phase.

The separation of the header and data digests is useful in iSCSI routing applications, where only the header changes when a message is forwarded. In this case, only the header digest should be re-calculated.

2.2.4 Basic Header Segment (BHS)

The Basic Header Segment is 44 bytes long.
The field of Opcode appears in all iSCSI PDUs. In addition, the Initiator Task Tag, Logical Unit Number, and Flags fields, when used,

always appear in the same location in the header.

```
Byte /    0       |       1       |       2       |       3       |
   /              |               |               |               |
  |7 6 5 4 3 2 1 0|7 6 5 4 3 2 1 0|7 6 5 4 3 2 1 0|7 6 5 4 3 2 1 0|
  +---------------+---------------+---------------+---------------+
 0|X| Opcode      |F| Opcode-specific fields                      |
  |               |P|                                             |
  +---------------+---------------+---------------+---------------+
 4| LUN or Opcode-specific fields                                 |
  +                                                               +
 8|                                                               |
  +---------------+---------------+---------------+---------------+
12| Initiator Task Tag or Opcode-specific fields                  |
  +---------------+---------------+---------------+---------------+
16/ Opcode-specific fields                                        /
 +/                                                               /
  +---------------+---------------+---------------+---------------+
44
```

2.2.4.1 X

The first bit of the Opcode is used as a Retry/Restart indicator

2.2.4.2 Opcode

The Opcode indicates what type of iSCSI PDU the header encapsulates. The Opcode is further encoded as follows:

```
b6   Response
b5-0 Operation
```


The opcodes are divided into two categories: initiator opcodes and target opcodes. Initiator opcodes are in PDUs sent by the initiators, and target opcodes are in PDUs sent by the target. The initiator MUST NOT send target opcodes and the target MUST NOT send initiator opcodes. Target opcodes are also called responses and are distinguished by having the Response bit (bit 6) set to 1.

Valid initiator opcodes defined in this specification are:

```
0x00 NOP-Out (from initiator to target)
```

0x01 SCSI Command (encapsulates a SCSI Command Descriptor Block)
0x02 SCSI Task Management Command
0x03 Login Command
0x04 Text Command
0x05 SCSI Data (for WRITE operation)
0x06 Logout Command
0x10 SACK Request

Valid target opcodes are:

0x40 NOP-In (from target to initiator)
0x41 SCSI Response (contains SCSI status and possibly sense information or other response information)
0x42 SCSI Task Management Response
0x43 Login Response
0x44 Text Response
0x45 SCSI Data (for READ operation)
0x46 Logout Response
0x50 Ready To Transfer (R2T - sent by target to initiator when it is ready to receive data from initiator)
0x51 Asynchronous Message (sent by target to initiator to indicate certain special conditions)
0x6f Reject

Initiator opcodes 0x30-0x3f and target opcodes 0x70-0x7f are vendor specific codes.

2.2.4.3 Opcode-specific fields

These fields have different meanings for different messages.

Bit 7 of the second byte is used as a Poll/Final bit (P/F bit) for some iSCSI PDUs and must be 0 in all other iSCSI PDUs. When used as a Poll bit it indicates that an answer is required. When used as a Final bit it indicates a Final PDU in a logical sequence (e.g., the last Data PDU of unsolicited or solicited data PDU sequence or the perceived final Request/Response of the Login Phase).

2.2.4.4 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 either is ignored or may be used for some other purpose. The LUN field is 64-bits in accordance with [SAM2]. The exact format of this field can be found in the [SAM2] document.

2.2.4.5 Initiator Task Tag

The initiator assigns a Task Tag to each iSCSI task that it issues. While a task exists this tag MUST uniquely identify it session-wide.

2.2.5 Extended CDB Additional Header Segment

```
Byte /    0       |       1       |       2       |       3       |
   /              |               |               |               |
  |7 6 5 4 3 2 1 0|7 6 5 4 3 2 1 0|7 6 5 4 3 2 1 0|7 6 5 4 3 2 1 0|
  +---------------+---------------+---------------+---------------+
 0/ Extended CDB                                                  /
 +/                                                               /
  +---------------+---------------+---------------+---------------+
```

2.2.6 Bi-directional Read Additional Header Segment

```
Byte /    0       |       1       |       2       |       3       |
   /              |               |               |               |
  |7 6 5 4 3 2 1 0|7 6 5 4 3 2 1 0|7 6 5 4 3 2 1 0|7 6 5 4 3 2 1 0|
  +---------------+---------------+---------------+---------------+
 0| Bi-directional Read Expected Data Length                      |
  +---------------+---------------+---------------+---------------+
```


2.2.7 Long Data Additional Header Segment

```
Byte /    0       |       1       |       2       |       3       |
   /              |               |               |               |
  |7 6 5 4 3 2 1 0|7 6 5 4 3 2 1 0|7 6 5 4 3 2 1 0|7 6 5 4 3 2 1 0|
  +---------------+---------------+---------------+---------------+
 0| Data Length                                                   |
  +---------------+---------------+---------------+---------------+
```

2.3 SCSI Command

```
   Byte /    0       |       1       |       2       |       3       |
      /              |               |               |               |
     |7 6 5 4 3 2 1 0|7 6 5 4 3 2 1 0|7 6 5 4 3 2 1 0|7 6 5 4 3 2 1 0|
     +---------------+---------------+---------------+---------------+
```

```
 0|X| 0x01        |F|R|W|0 0|ATTR | Reserved      | CmdRN or Rsvd |
  +---------------+---------------+---------------+---------------+
 4| Logical Unit Number (LUN)                                     |
  +                                                               +
 8|                                                               |
  +---------------+---------------+---------------+---------------+
12| Initiator Task Tag                                            |
  +---------------+---------------+---------------+---------------+
16| Expected Data Transfer Length                                 |
  +---------------+---------------+---------------+---------------+
20| CmdSN                                                         |
  +---------------+---------------+---------------+---------------+
24| ExpStatSN or EndDataSN                                        |
  +---------------+---------------+---------------+---------------+
28/ SCSI Command Descriptor Block (CDB)                           /
 +/                                                               /
  +---------------+---------------+---------------+---------------+
44
```

2.3.1 Flags & Task Attributes

The flags field for a SCSI Command is:

b7 (F) set to 1 when the immediate data that accompany the command are all the data associated with this command
b6 (R) set to 1 when input data is expected
b5 (W) set to 1 when output data is expected
b3-4 Reserved (MUST be 0)
b0-2 used to indicate Task Attributes

The Task Attributes (ATTR) can have one of the following integer values (see [SAM2] for details):

0 Untagged
1 Simple
2 Ordered
3 Head of Queue
4 ACA

2.3.2 CmdRN

SCSI command reference number - if present in the SCSI execute arguments

2.3.3 CmdSN - Command Sequence Number

Enables ordered delivery across multiple connections in a single session.

2.3.4 ExpStatSN/EndDataSN - Expected Status Sequence Number

Command responses up to ExpStatSN-1 (mod 2**32) have been received (acknowledges status) on the connection. If the command is a retry (the X bit is 1) this field will contain the last input DataSN number seen by the initiator for this command in a previous execution or 0x'ffffffff'.

2.3.5 Expected Data Transfer Length

For unidirectional operations, the Expected Data Transfer Length field states the number of bytes of data involved in this SCSI operation. For a WRITE (W flag set to 1 and R flag set to 0) operation, the initiator uses this field to specify the number of bytes of data it expects to transfer for this operation. For a READ (W flag set to 0 and R flag set to 1) operation, 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 SAM-2 byte count.

If the Expected Data transfer Length for a WRITE and the length of 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.

For bi-directional operations (both R and W flags are set to 1), this field states the number of data bytes involved in the outbound transfer. For bi-directional operations, an additional header segment MUST be present in the header sequence indicating the Expected Bi-directional Read Data Length. If this additional header segment is absent, the Expected Bi-directional Read Data Length is assumed 0.

Upon completion of a data transfer, the target will inform the initiator of how many bytes were actually processed (sent or received) by the target. This will be done through residual counts.

2.3.6 CDB - SCSI Command Descriptor Block

There are 16 bytes in the CDB field to accommodate the commonly used CDB. Whenever larger CDBs are used, the CDB spillover MAY extend beyond the 48-byte header.

2.3.7 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 (a data segment). Alternatively, user data (as from a WRITE operation) can be placed in the same PDU (both cases referred to as immediate data). Those data are governed by the general rules for solicited vs. unsolicited data.

2.4 SCSI Response

```
Byte /    0       |       1       |       2       |       3       |
   /              |               |               |               |
  |7 6 5 4 3 2 1 0|7 6 5 4 3 2 1 0|7 6 5 4 3 2 1 0|7 6 5 4 3 2 1 0|
  +---------------+---------------+---------------+---------------+
 0| 0x41          |Rsvd |S|o|u|O|U| Reserved (0)  |Status/Response|
  +---------------+---------------+---------------+---------------+
 4| Reserved (0)                                                  |
  +                                                               +
 8|                                                               |
  +---------------+---------------+---------------+---------------+
12| Initiator Task Tag                                            |
  +---------------+---------------+---------------+---------------+
16| Basic Residual Count                                          |
  +---------------+---------------+---------------+---------------+
20| StatSN                                                        |
  +---------------+---------------+---------------+---------------+
24| ExpCmdSN                                                      |
  +---------------+---------------+---------------+---------------+
28| MaxCmdSN                                                      |
  +---------------+---------------+---------------+---------------+
32| EndDataSN or Reserved (0)                                     |
  +---------------+---------------+---------------+---------------+
36| SRLength                      | Reserved (0)                  |
  +---------------+---------------+---------------+---------------+
40| Bidi-Read Residual Count                                      |
  +---------------+---------------+---------------+---------------+
44| Digests if any...                                             |
  +---------------+---------------+---------------+---------------+
  / Sense Data (optional) or Response Data                        /
 +/                                                               /
  +---------------+---------------+---------------+---------------+
```

2.4.1 Byte 1 - Flags

b0 (U) set for Residual Underflow. In this case, the Basic Residual Count indicates how many bytes were not transferred out of those expected to be transferred.
b1 (O) set for Residual Overflow. In this case, the Basic Residual Count indicates how many bytes could not be transferred because the initiator's Expected Data Transfer Length was too small.
b2 (u) same as b0 but for the read-part of a bi-directional operation

b3 (o) same as b1 but for the read-part of a bi-directional

operation
b4 (S) Status-Response selector - if 1 the response contains a valid SCSI status else a valid iSCSI Response
b5-7 Reserved

Bits O and U are mutually exclusive and so are bits o and u. For a response (S=0) b0-b3 MUST be 0.

2.4.2 Status/Response

The Status field is used to report the SCSI status of the command (as specified in [SAM2]). The Response is used to report a Service Response. The exact mapping of the iSCSI response codes to SAM service response symbols is outside the scope of this document.

If a SCSI device error is detected while data from the initiator are 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 the a Data PDU with the F bit set before sending the Response PDU.

Valid iSCSI Response codes are:

1 - Target Failure
2 - Delivery Subsystem Failure
3 - Unsolicited data rejected

2.4.3 Basic Residual Count

The Basic Residual Count field is valid only in case either the U bit or the O bit is set. If neither bit is set, the Basic Residual Count field SHOULD be zero. If the U bit is set, the Basic Residual Count indicates how many bytes were not transferred out of those expected to be transferred. If the O bit is set, the Basic Residual Count indicates how many bytes could not be transferred because the initiator's Expected Data Transfer Length was too small.

2.4.4 Bidi-Read Residual Count

The Bidi-Read Residual Count field is valid only in case either the u bit or the o bit is set. If neither bit is set, the Bidi-Read Residual Count field SHOULD be zero. If the u bit is set, the Bidi-Read Residual Count indicates how many bytes were not transferred to the initiator out of those expected to. If the o bit is set, the Bidi-Read Residual Count indicates how many bytes could not be

transferred to the initiator because the initiator's Expected Bidi-Read Transfer Length was too small.

2.4.5 SR-length

This is the length of sense data or of the response.

2.4.6 Sense or Response Data

iSCSI targets MUST support and enable autosense. If the Command Status was CHECK CONDITION (0x02), then the Sense Data field will contain sense data for the failed command.

For some iSCSI responses the response field MAY contain some response related information, e.g., for a target failure it may contain a (vendor specific) detailed description of the failure.

2.4.7 EndDataSN

One past the largest DataSN in an input (read) data PDU the target has sent for the command. 0 means no data PDUs where sent.

2.4.8 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 SACK. For responses sent because of retry the StatSN used MUST be the same as the first time the PDU was sent unless the connection was restarted since then.

2.4.9 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 counter with the same name.

2.4.10 MaxCmdSN - maximum CmdSN acceptable 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 counter with the same name.

MaxCmdSN and ExpCmdSN are processed as follows:

-if the PDU MaxCmdSN is less than the PDU ExpCmdSN (in Serial Arithmetic Sense and with a difference bounded by 2**31-1), they are both ignored
-if the PDU MaxCmdSN is less than the current MaxCmdSN (in Serial Arithmetic Sense and with a difference bounded by 2**31-1), it is ignored; else it updates MaxCmdSN
-if the PDU ExpCmdSN is less than the current ExpCmdSN (in Serial Arithmetic Sense and with a difference bounded by 2**31-1), it is ignored; else it updates ExpCmdSN

This sequence is required as updates may arrive out of order (they travel on different TCP connections).

2.5 SCSI Task Management Command

```
Byte /    0       |       1       |       2       |       3       |
   /              |               |               |               |
  |7 6 5 4 3 2 1 0|7 6 5 4 3 2 1 0|7 6 5 4 3 2 1 0|7 6 5 4 3 2 1 0|
  +---------------+---------------+---------------+---------------+
 0|X| 0x02        |0| Function    | Reserved (0)                  |
  +---------------+---------------+---------------+---------------+
 4| Logical Unit Number (LUN) or Reserved (0)                     |
  +                                                               +
 8|                                                               |
  +---------------+---------------+---------------+---------------+
12| Initiator Task Tag                                            |
  +---------------+---------------+---------------+---------------+
16| Referenced Task Tag or Reserved (0)                           |
  +---------------+---------------+---------------+---------------+
20| CmdSN                                                         |
  +---------------+---------------+---------------+---------------+
24| ExpStatSN                                                     |
  +---------------+---------------+---------------+---------------+
28/ Reserved (0)                                                  /
 +/                                                               /
  +---------------+---------------+---------------+---------------+
44
```

2.5.1 Function

The Task Management functions provide an initiator with a way to explicitly control the execution of one or more Tasks. The Task Management functions are summarized as follows (for a more detailed description see the [SAM2] document):

1 Abort Task---aborts the task identified by the Referenced Task Tag field.
2 Abort Task Set---aborts all Tasks issued by this initiator on the Logical Unit.
3 Clear ACA---clears the Auto Contingent Allegiance condition.
4 Clear Task Set---Aborts all Tasks (from all initiators) for the Logical Unit.
5 Logical Unit Reset
6 Target Warm Reset
7 Target Cold Reset

For the functions above a SCSI Task Management Response MUST be returned, using the Initiator Task Tag to identify the operation for which it is responding.

For the <Clear Task Set>, if SCSI control mode enables AE reporting, the target MUST send an Asynchronous Event to all other attached initiators to inform them that all pending tasks are cancelled and then enter the ACA state for any initiator for which it had pending tasks.

For the <Target Warm Reset> and <Target Cold Reset> functions, the target cancels all pending operations and are both equivalent to the Target Reset as specified by SAM-2. Provided that SCSI control mode enables AE reporting, the target MUST send an Asynchronous Event to all attached initiators notifying them that the target is being reset.

In addition, for the <Target Warm Reset> the target will enter the ACA state on all sessions and all LUs on which an AE was sent.

In addition, for the <Target Cold Reset> the target then MUST terminate all of its TCP connections to all initiators (all sessions are terminated). However, if the target finds that it cannot send the required response or AEN it MUST continue the reset operation and it SHOULD log the condition for later retrieval. The logging operation MUST be reported through the target MIB.

Further actions on reset functions are specified in the relevant SCSI documents for the specific class of devices.

2.5.2 Referenced Task Tag

Initiator Task Tag of the task to be aborted - for abort task

2.6 SCSI Task Management Response

```
Byte /    0       |       1       |       2       |       3       |
   /              |               |               |               |
  |7 6 5 4 3 2 1 0|7 6 5 4 3 2 1 0|7 6 5 4 3 2 1 0|7 6 5 4 3 2 1 0|
  +---------------+---------------+---------------+---------------+
 0| 0x42          |0| Reserved (0)                                |
  +---------------+---------------+---------------+---------------+
 4| Logical Unit Number (LUN)                                     |
  +                                                               +
 8|                                                               |
  +---------------+---------------+---------------+---------------+
12| Initiator Task Tag                                            |
  +---------------+---------------+---------------+---------------+
16| Referenced Task Tag or Reserved (0)                           |
  +---------------+---------------+---------------+---------------+
20| StatSN                                                        |
  +---------------+---------------+---------------+---------------+
24| ExpCmdSN                                                      |
  +---------------+---------------+---------------+---------------+
28| MaxCmdSN                                                      |
  +---------------+---------------+---------------+---------------+
32| Response      | Reserved (0)                                  |
  +---------------+---------------+---------------+---------------+
36/ Reserved (0)                                                  /
 +/                                                               /
  +---------------+---------------+---------------+---------------+
44
```

For the functions <Abort Task, Abort Task Set, Clear ACA, Clear Task Set, Logical Unit reset, Target Warm Reset>, the target performs the requested Task Management function and sends a SCSI Task Management Response back to the initiator. The target provides a Response, which may take on the following values:

```
    0    Function Complete
    1    Task was not in task set
  255    Function Rejected
```

For the <Target Cold Reset> and <Target Warm Reset> functions, the target cancels all pending operations. If SCSI control mode enables AE reporting, the target MUST send an Asynchronous Event to all attached initiators notifying them that the target has been reset.

For the <Target Cold Reset> 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 is outside the scope of this document.

2.6.1 Referenced Task Tag

Initiator Task Tag of the task not found

2.7 SCSI Data

The typical data transfer specifies the length of the data payload, the Target Transfer Tag provided by the receiver for this data transfer, and a buffer offset. The typical SCSI Data packet for WRITE (from initiator to target) has the following format:

```
Byte /    0       |       1       |       2       |       3       |
   /              |               |               |               |
  |7 6 5 4 3 2 1 0|7 6 5 4 3 2 1 0|7 6 5 4 3 2 1 0|7 6 5 4 3 2 1 0|
  +---------------+---------------+---------------+---------------+
 0|0| 0x05        |F| Reserved (0)                                |
  +---------------+---------------+---------------+---------------+
 4| LUN or Reserved (0)                                           |
  +                                                               +
 8|                                                               |
  +---------------+---------------+---------------+---------------+
12| Initiator Task Tag                                            |
  +---------------+---------------+---------------+---------------+
16| Target Transfer Tag or (0x'ffffffff')                         |
  +---------------+---------------+---------------+---------------+
20| Reserved (0)                                                  |
  +---------------+---------------+---------------+---------------+
24| ExpStatSN                                                     |
  +---------------+---------------+---------------+---------------+
28| Reserved (0)                                                  |
  +---------------+---------------+---------------+---------------+
32| DataSN                                                        |
  +---------------+---------------+---------------+---------------+
36| Buffer Offset                                                 |
  +---------------+---------------+---------------+---------------+
40| Reserved (0)                                                  |
  +---------------+---------------+---------------+---------------+
44| Digests if any...                                             |
  +---------------+---------------+---------------+---------------+
  / Payload                                                       /
 +/                                                               /
  +---------------+---------------+---------------+---------------+
```

The typical SCSI Data packet for READ (from target to initiator) has the following format:

```
Byte /    0       |       1       |       2       |       3       |
   /              |               |               |               |
  |7 6 5 4 3 2 1 0|7 6 5 4 3 2 1 0|7 6 5 4 3 2 1 0|7 6 5 4 3 2 1 0|
  +---------------+---------------+---------------+---------------+
 0| 0x45          |F|   (0) |S|O|U| Reserved (0)  |Status or Rsvd |
  +---------------+---------------+---------------+---------------+
 4| Reserved (0)                                                  |
  +                                                               +
 8|                                                               |
  +---------------+---------------+---------------+---------------+
12| Initiator Task Tag                                            |
  +---------------+---------------+---------------+---------------+
16| Reserved (0)                                                  |
  +---------------+---------------+---------------+---------------+
20| StatSN or Reserved (0)                                        |
  +---------------+---------------+---------------+---------------+
24| ExpCmdSN                                                      |
  +---------------+---------------+---------------+---------------+
28| MaxCmdSN                                                      |
  +---------------+---------------+---------------+---------------+
32| DataSN                                                        |
  +---------------+---------------+---------------+---------------+
36| Buffer Offset                                                 |
  +---------------+---------------+---------------+---------------+
40| Residual Count                                                |
  +---------------+---------------+---------------+---------------+
44| Digests if any...                                             |
  +---------------+---------------+---------------+---------------+
  / Payload                                                       /
 +/                                                               /
  +---------------+---------------+---------------+---------------+
```

2.7.1 F (Final) bit

For outgoing data, this bit is 1 for the last PDU of unsolicited data or the last PDU of a sequence answering a R2T.

For incoming data, this bit is 1 for the last input data PDU associated with the command (even if it includes the status).

2.7.2 Target Transfer Tag

On outgoing data the Target Transfer Tag is provided to the target if the transfer is honoring a R2T. In this case, the Target Transfer Tag field is a replica of the Target Transfer Tag provided with the R2T. The Target Transfer Tag values are not specified by this protocol except that the all-bits-one value (0x'ffffffff') 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 consistent with whatever was specified with the command, else the LUN field is reserved.

2.7.3 DataSN

For input (read) data PDUs, the DataSN is the data PDU number (starting with 0) within the data transfer for the command identified by the Initiator Task Tag.

For output (write) data PDUs, the DataSN is the data PDU number (starting with 0) within the current output sequence as identified by the Initiator Task Tag (for unsolicited data) or by the Target Task Tag and LUN (for data solicited through R2T).

0x'ffffffff' is not a valid DataSN and MUST be skipped when counting (serial arithmetic)

2.7.4 Buffer Offset

The Buffer Offset field contains the offset of the following data against the complete data transfer. The sum of the buffer offset and length should not exceed the expected transfer length for the command.

Input data ordering is governed by a disconnect-reconnect mode page bit (EMDP). If this bit is 1 the target MUST deliver packets in increasing buffer offset order.

Output data within a burst (initial or any data PDU sequence that fulfils a R2T) MUST be delivered in increasing buffer offset order.

2.7.5 Flags

The last SCSI Data packet sent from a target to an initiator for a

particular SCSI command that completed successfully may optionally also contain the Command Status for the data transfer. In this case Sense Data cannot be sent together with the Command Status. If the command completed with an error, then the response and sense data

must be sent in a SCSI Response packet and must not be sent in a SCSI Data packet.

b0-1 as in an SCSI Response
b2 S (status)- set to indicate that the Command Status field contains status
b3-6 not used (should be set to 0)
b7 P (poll) - set to indicate data acknowledgement is requested; b7 and b2 are mutually exclusive - if S bit is set P bit MUST be ignored

If the S bit is set to 1, then there is meaning to the extra fields in the SCSI Data packet (StatSN, Command Status, Residual Count).

2.8 Text Command

The Text Command is provided to allow 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       |
   /              |               |               |               |
  |7 6 5 4 3 2 1 0|7 6 5 4 3 2 1 0|7 6 5 4 3 2 1 0|7 6 5 4 3 2 1 0|
  +---------------+---------------+---------------+---------------+
 0|0| 0x04        |F| Reserved (0)                                |
  +---------------+---------------+---------------+---------------+
 4| Reserved (0)                                                  |
  +                                                               +
 8|                                                               |
  +---------------+---------------+---------------+---------------+
12| Initiator Task Tag                                            |
  +---------------+---------------+---------------+---------------+
16| Reserved (0)                                                  |
  +---------------+---------------+---------------+---------------+
20| CmdSN                                                         |
  +---------------+---------------+---------------+---------------+
24| ExpStatSN                                                     |
  +---------------+---------------+---------------+---------------+
28/ Reserved (0)                                                  /
 +/                                                               /
  +---------------+---------------+---------------+---------------+
44| Digests if any...                                             |
  +---------------+---------------+---------------+---------------+
  / Text                                                          /
 +/                                                               /
  +---------------+---------------+---------------+---------------+
```

2.8.1 Final (F) bit

When set to 1 it indicates that his is the last or only text command in a sequence of commands; else it indicates that more commands will follow.

2.8.2 Initiator Task Tag

The initiator assigned identifier for this Text Command.
If the command is sent as part of a sequence of commands (e.g., the Login Phase or a sequence of Text commands) the Initiator Task Tag

MUST be the same for all the commands within the sequence (similar to linked SCSI commands).

2.8.3 Text

The initiator sends the target a set of key=value or key=list pairs encoded in UTF-8 Unicode. The key and value are separated by a '=' (0x3D) delimiter. Many key=value pairs can be included in the Text block by separating them with null (0x00) delimiters. A list is a set of values separated by comma (0x2C). Large binary items can be encoded using their hexadecimal representation (e.g., 8190 is 0x1FFE).

Character strings are represented as plain text. Numeric and binary values are represented using either decimal numbers or the hexadecimal 0x'ffff' notation. The result is adjusted to the specific key.

The target responds by sending its response back to the initiator. The response text format is similar to the request text format.

Some basic key=value pairs are described in Appendix A & D. All these keys except the X- extension formatted MUST be supported by iSCSI initiators and targets.

Manufacturers may introduce new keys by prefixing them with X- followed by their (reversed) domain name, for example the company owning the domain acme.com can issue:

```
X-com.acme.bar.foo.do_something=0000000000000003
```

Any key that the target does not understand may be ignored without affecting basic function.

Text operations are usually meant for parameter setting/negotiations but can be used also to perform some active operations.

It is recommended that Text operations that will take a long time should be placed in their own Text command. If the Text Response does not contain a key that was requested, the initiator must assume that the key was not understood by the target.

Targets and initiators may limit the size of the text accepted in a text command and text response as well as the size of key=value pairs. Such limits should be indicated at login.

2.9 Text Response

The Text Response message contains the responses of the target to the initiator's Text Command. The format of the Text field matches that of the Text Command.

```
Byte /    0       |       1       |       2       |       3       |
   /              |               |               |               |
  |7 6 5 4 3 2 1 0|7 6 5 4 3 2 1 0|7 6 5 4 3 2 1 0|7 6 5 4 3 2 1 0|
  +---------------+---------------+---------------+---------------+
 0| 0x44          |F| Reserved (0)                                |
  +---------------+---------------+---------------+---------------+
 4| Reserved (0)                                                  |
  +                                                               +
 8|                                                               |
  +---------------+---------------+---------------+---------------+
12| Initiator Task Tag                                            |
  +---------------+---------------+---------------+---------------+
16| Reserved (0)                                                  |
  +---------------+---------------+---------------+---------------+
20| StatSN                                                        |
  +---------------+---------------+---------------+---------------+
24| ExpCmdSN                                                      |
  +---------------+---------------+---------------+---------------+
28| MaxCmdSN                                                      |
  +---------------+---------------+---------------+---------------+
32/ Reserved (0)                                                  /
 +/                                                               /
  +---------------+---------------+---------------+---------------+
44| Digests if any...                                             |
  +---------------+---------------+---------------+---------------+
```

```
 / Text                                                        /
+/                                                             /
 +---------------+---------------+---------------+---------------+
```

2.9.1 Final (F) bit

When set to 1 in response to a text command with the Final bit set to 1 it indicates that the target has finished it's operation. Else if set to 0 in response to a text command with the Final Bit set to 1 it indicates that the target has more work to do (invites a follow-on text command). A text response with the F bit set to 1 in response to a text command with the F bit set to 0 is a protocol error.

2.9.2 Initiator Task Tag

The Initiator Task Tag matches the tag used in the initial Text Command or the Login Initiator Task Tag.

2.9.3 Text Response

The Text Response field contains responses in the same key=value format as the Text Command. Appendix C lists some basic Text Commands and their Responses. If the Text Response does not contain a key that was requested, the initiator must assume that the key was not understood by the target or that the answer is <key>=none and the two MUST be equivalent where applicable.

2.10 Login Command

After establishing a TCP connection between an initiator and a target, the initiator MUST issue a Login Command to gain further access to the target's resources.

A Login Command MUST NOT be issued more than once on an iSCSI TCP connection.

```
Byte /    0       |       1       |       2       |       3       |
   /              |               |               |               |
  |7 6 5 4 3 2 1 0|7 6 5 4 3 2 1 0|7 6 5 4 3 2 1 0|7 6 5 4 3 2 1 0|
  +---------------+---------------+---------------+---------------+
 0|X| 0x03        |F| Reserved (0)| Version-max   | Version-min   |
  +---------------+---------------+---------------+---------------+
 4| CID                           | Reserved (0)                  |
  +---------------+---------------+---------------+---------------+
 8| ISID                          |TSID                           |
  +---------------+---------------+---------------+---------------+
12| Initiator Task Tag                                            |
  +---------------+---------------+---------------+---------------+
16| Reserved (0)                                                  |
  +---------------+---------------+---------------+---------------+
20| InitCmdSN   or   Reserved (0)                                 |
  +---------------+---------------+---------------+---------------+
24| ExpStatSN   or   Reserved (0)                                 |
  +---------------+---------------+---------------+---------------+
28/ Reserved (0)                                                  /
```

```
 +/                                                              /
  +---------------+---------------+---------------+---------------+
44/ Login Parameters in Text Command Format                       /
 +/                                                              /
  +---------------+---------------+---------------+---------------+
```

2.10.1 X - Restart

This is an attempt to reinstate a failed connection - CID does not change but logout first the old connection.

2.10.2 F - final

If set to 1 indicates that the initiator has no more parameters to set

2.10.3 Version-max

Maximum Version number supported.

2.10.4 Version-min

Minimum Version supported
The version number of the current draft is 0x1.

2.10.5 CID

This is a unique id for this connection within the session.
CIDs MUST NOT be reused during the life of a session (every connection ever used in a session MUST have a unique CID)

2.10.6 ISID

This an initiator defined session-identifier. It MUST be the same for all connections within a session.

2.10.7 InitCmdSN

Is significant only if TSID is zero and indicates the starting Command Sequence Number for this session; it SHOULD be zero for all other instances.

2.10.8 ExpStatSN

This is ExpStatSN for the old connection.

This field is valid only if the X bit is set to 1.

2.10.9 Login Parameters

The initiator MAY provide some basic parameters in order to enable the target to determine if the initiator may in fact use the target's resources and the initial text parameters for the security exchange. The format of the parameters is as specified for the Text Command. Keys and their explanations are listed in Appendixes.

2.11 Login Response

The Login Response indicates the end of the login phase. Note that if security is established, the login response is authenticated.

```
Byte /    0       |       1       |       2       |       3       |
   /              |               |               |               |
  |7 6 5 4 3 2 1 0|7 6 5 4 3 2 1 0|7 6 5 4 3 2 1 0|7 6 5 4 3 2 1 0|
  +---------------+---------------+---------------+---------------+
 0| 0x83          |F| Reserved (0)| Version-max   | Version-active|
  +---------------+---------------+---------------+---------------+
 4| Reserved (0)                                                  |
  +---------------+---------------+---------------+---------------+
 8| ISID                          |TSID                           |
  +---------------+---------------+---------------+---------------+
12| Initiator Task Tag                                            |
  +---------------+---------------+---------------+---------------+
16| Reserved (0)                                                  |
  +---------------+---------------+---------------+---------------+
20| InitStatSN                                                    |
  +---------------+---------------+---------------+---------------+
24| ExpCmdSN                                                      |
  +---------------+---------------+---------------+---------------+
28| MaxCmdSN                                                      |
  +---------------+---------------+---------------+---------------+
32| Status-Class  | Status-Detail |                               |
  +---------------+---------------+---------------+---------------+
```

```
36/ Reserved (0)                                                /
 +/                                                             /
  +---------------+---------------+---------------+---------------+
44| Digests if any...                                            |
  +---------------+---------------+---------------+---------------+
  / Login Parameters in Text Command Format                     /
 +/                                                             /
  +---------------+---------------+---------------+---------------+
```

2.11.1 Version-max

This is the highest version number supported by the target.

2.11.2 Version-active/lowest

Indicates the version supported (the highest supported by the target and initiator). If the target is not supporting a version within the

range of the initiator it will reject the login and this field will indicate the lowest version supported by the target.

2.11.3 InitStatSN

This is the starting status Sequence Number for this connection. The value is relevant only if the F bit is set to 1.

2.11.4 Status-Class and Status-Detail

The Status returned in a Login Response indicates the status of the login request. The status includes:

Status-Class
Status-Detail

The Status-Class is sufficient for a simple initiator to use when handling errors, without having to look at the Status-Detail. The Status-Detail allows finer-grained error recovery for more sophisticated initiators, as well as better information for error logging.

The status classes are as follows:

0 - Success - the iSCSI target successfully received, understood, and accepted the request.

1 - Redirection - indicates that further action must be taken by the initiator to complete the request. This is usually due to the target moving to a different address. All of the 3 status class responses MUST return one or more text key parameters of the type "TargetAddress", indicating the target's new address.

2 - Initiator Error - indicates that the initiator likely caused the error. This is MAY be due to a request for a resource for which the initiator does not have permission.

3 - Target Error - indicates that the target is incapable of fulfilling the request.

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. The allowable state of the Final (F) bit in responses with each of the codes is indicated as well.

Status	Code (hex)	F bit	Description
Accept Login	0000	1/0	Login is OK, moving to Full Feature Phase (F=1) or Operational Parameter Negotiation (F=0).
Authenticate	0001	0	The target WWUI exists and authentication proceeds.
Target Moved Temporarily	0101	1	The requested target WWUI has moved temporarily to the address provided.
Target Moved Permanently	0102	1	The requested target WWUI has moved permanently to the address provided.
Proxy Required	0103	1	The initiator must use an iSCSI proxy for this target; address is provided.
Authentication Failed	0201	1	The initiator authentication failed.

Forbidden Target	0202	1	The initiator is not allowed access to the given target.
Not Found	0203	1	The requested Target WWUI does not exist at this address.
Target Removed	0204	1	The requested target WWUI has been removed; no forwarding address provided.
Target Conflict	0205	1	Target is currently in use by another initiator, and does not support multiple initiators.
Target Error	0300	1	An error occurred in the iSCSI target (out of resources, etc.).
Service Unavailable	0301	1	The iSCSI service or target is not currently operational, usually due to maintenance.

Unsupported version	0302	1	The required version is not supported by the target.

If the Status is "accept login" (0x0000) and the F bit is 1, the initiator may proceed to issue SCSI commands. If the Status is "accept login" (0x0000) and the F bit is 0, the initiator may proceed negotiating operational parameters. The target MUST not set the Status to 0x'0000' and the F bit to 1 if the Login Command had the F bit set to 0.

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.

2.11.5 TSID

The TSID is an initiator identifying tag set by the target. A 0 in the returned TSID indicates that either the target supports only a single connection or that the ISID has already been used as a leading

ISID. In both cases, the target is rejecting the login.

2.11.6 Final bit

Final bit is set to one in the Final Login Response. A Final bit of 0 indicates a "partial" response - more negotiation needed.
TSID must be returned in the partial response and the same value must be presented with the final response.

2.12 NOP-Out

```
Byte /    0       |       1       |       2       |       3       |
   /              |               |               |               |
  |7 6 5 4 3 2 1 0|7 6 5 4 3 2 1 0|7 6 5 4 3 2 1 0|7 6 5 4 3 2 1 0|
  +---------------+---------------+---------------+---------------+
 0| 0x00          |P| Reserved (0)                                |
  +---------------+---------------+---------------+---------------+
 4| LUN or Reserved (0)                                           |
  +                                                               +
 8|                                                               |
  +---------------+---------------+---------------+---------------+
12| Initiator Task Tag or Reserved (0x'ffffffff')                 |
  +---------------+---------------+---------------+---------------+
16| Target Transfer Tag or Reserved (0x'ffffffff')                |
  +---------------+---------------+---------------+---------------+
20| CmdSN or (0)                                                  |
  +---------------+---------------+---------------+---------------+
24| ExpStatSN or (0)                                              |
  +---------------+---------------+---------------+---------------+
```

```
28/ Reserved (0)                                                /
 +/                                                             /
  +---------------+---------------+---------------+---------------+
44| Digests if any...                                            |
  +---------------+---------------+---------------+---------------+
  / Ping Data (optional)                                        /
 +/                                                             /
  +---------------+---------------+---------------+---------------+
```

The NOP-Out with the P bit set acts as a "ping command". This form of the NOP-Out can be used to verify that a connection is still active and all it's components are operational using in-order delivery or out-of-order delivery. It may be useful in the case where an initiator has been waiting a long time for the response to some command, and the initiator suspects that there is some problem with the connection. When a target receives the NOP-Out with the Ping bit set, it should respond with a Ping Response, duplicating as much as possible of the data that was provided in the NOP-Out. If the initiator does not receive the NOP-In within some time (determined by the initiator), or if the data returned by the NOP-In is different from the data that was in the NOP-Out, the initiator may conclude that there is a problem with the connection. The initiator will then close the connection and may try to establish a new connection.

The NOP-Out can be sent by an initiator because of a NOP-In with the poll bit set, in which case the Target Tag will copy the NOP-In value and the P bit will be 0.

2.12.1 P - Ping bit

Request a NOP-In

2.12.2 LUN

The LUN field MUST be set whenever the Target Transfer Tag is set.

2.12.3 Initiator Task Tag

An initiator assigned identifier for the operation.

The NOP-Out MUST have the Initiator Task Tag set only if the P bit is 1.

2.12.4 Target Transfer Tag

A target assigned identifier for the operation.

The NOP-Out MUST have the Target Tag set only if it issued in response to a NOP-In or a Data-IN with the P bit one, in which case it copies the Target Transfer Tag from the NOP-In or Data-IN PDU. When the Target Transfer Tag is set the LUN field must have the correct value for the task.

2.12.5 Ping Data

Ping data will be reflected in the Ping Response. Please note that the length of the reflected data is limited by a negotiated parameter and the initiator SHOULD avoid sending more than the negotiated limit.

2.13 NOP-In

```
Byte /    0       |       1       |       2       |       3       |
   /              |               |               |               |
  |7 6 5 4 3 2 1 0|7 6 5 4 3 2 1 0|7 6 5 4 3 2 1 0|7 6 5 4 3 2 1 0|
  +---------------+---------------+---------------+---------------+
 0| 0x80          |P| Reserved (0)                                |
  +---------------+---------------+---------------+---------------+
 4| Reserved (0)                                                  |
  +                                                               +
 8|                                                               |
  +---------------+---------------+---------------+---------------+
12| Initiator Task Tag or Reserved (0x'ffffffff')                 |
  +---------------+---------------+---------------+---------------+
16| Target Transfer Tag or Reserved (0x'ffffffff')                |
  +---------------+---------------+---------------+---------------+
20| StatSN                                                        |
```

```
     +---------------+---------------+---------------+---------------+
    24| ExpCmdSN                                                      |
     +---------------+---------------+---------------+---------------+
    28| MaxCmdSN                                                      |
     +---------------+---------------+---------------+---------------+
    36/ Reserved (0)                                                  /
     +/                                                               /
     +---------------+---------------+---------------+---------------+
    44| Digests if any...                                             |
     +---------------+---------------+---------------+---------------+
     / Return Ping Data                                               /
     +/                                                               /
     +---------------+---------------+---------------+---------------+
```

When a target receives the NOP-Out with the P bit set, it MUST respond with a NOP-In, with the same Initiator Task Tag that was provided in the Ping Command. It SHOULD also duplicate as much of the initiator provided Ping Data as allowed by a configurable target parameter. The P bit for such a response MUST be 0.

2.13.1 P bit

A target may issue a NOP-In on its own to test the connection and the state of the initiator. If the target wants to test the initiator, it will set the P bit to 1 to ask for an answer from the initiator. In this case the Initiator Task Tag MUST be 0x'ffffffff' and the Target Tag MUST be set (not 0x'ffffffff'). If the target wants only to test

the connection, the P bit will be set to 0 and both tags MUST hold the reserved value 0x'ffffffff.

Whenever the NOP-In is not issued in response to a NOP-Out the StatSN field will contain as usual the next StatSN but StatSN for this connection will not be advanced.

2.13.2 Target Transfer Tag

A target assigned identifier for the operation.

2.14 Logout Command

The Logout command is used to perform a controlled closing of a connection.

An initiator MAY use a logout command to remove a connection from a session or to close an entire session.

If an initiator intends to start recovery for a failing connection it MUST use the either the Logout command to "clean-up" the target end of a failing connection and enable recovery to start, or use the restart option of the Login command to the same effect. On sessions with a single connection, this might imply opening a second

connection with the sole purpose of cleaning-up the first.

```
Byte /    0       |       1       |       2       |       3       |
   /              |               |               |               |
  |7 6 5 4 3 2 1 0|7 6 5 4 3 2 1 0|7 6 5 4 3 2 1 0|7 6 5 4 3 2 1 0|
  +---------------+---------------+---------------+---------------+
 0| 0x06          |0| Reserved (0)                                |
  +---------------+---------------+---------------+---------------+
 4| CID                           | Reserved (0)  |Reason Code    |
  +---------------+---------------+---------------+---------------+
 8| Reserved (0)                                                  |
  +---------------+---------------+---------------+---------------+
12| Initiator Task Tag                                            |
  +---------------+---------------+---------------+---------------+
16/ Reserved (0)                                                  /
 +/                                                               /
  +---------------+---------------+---------------+---------------+
24| ExpStatSN or (0)                                              |
  +---------------+---------------+---------------+---------------+
28/ Reserved (0)                                                  /
 +/                                                               /
  +---------------+---------------+---------------+---------------+
44
```

2.14.1 CID

The connection ID of the connection to be closed (including closing the TCP stream)

2.14.2 ExpStatSN

This is the ExpStatSN for the connection to be closed.

2.14.3 Reason Code

Indicate the reason for Logout:

0 - Remove the connection because the session is closing
1 - Remove the connection for recovery
2 - Remove the connection at target's request (requested through an AEN)

2.15 Logout Response

The logout response is used by the target to indicate that the cleanup operation for the failed connection has completed.

After Logout, the TCP connection MUST be closed at both ends.

```
Byte /    0       |       1       |       2       |       3       |
   /              |               |               |               |
  |7 6 5 4 3 2 1 0|7 6 5 4 3 2 1 0|7 6 5 4 3 2 1 0|7 6 5 4 3 2 1 0|
  +---------------+---------------+---------------+---------------+
 0| 0x86          |0| Reserved (0)                                |
  +---------------+---------------+---------------+---------------+
 4| Reserved (0)                                                  |
  +                                                               +
 8|                                                               |
  +---------------+---------------+---------------+---------------+
12| Initiator Task Tag                                            |
  +---------------+---------------+---------------+---------------+
16/ Reserved (0)                                                  /
 +/                                                               /
  +---------------+---------------+---------------+---------------+
24| ExpCmdSN                                                      |
  +---------------+---------------+---------------+---------------+
28| MaxCmdSN                                                      |
  +---------------+---------------+---------------+---------------+
32| Status        | Reserved (0)                                  |
  +---------------------------------------------------------------+
36/ Reserved (0)                                                  /
 +/                                                               /
  +---------------+---------------+---------------+---------------+
44
```

2.15.1 Status

Logout ending status:

0 - connection closed successfully
1 - cleanup failed

2.16 SACK Request

```
Byte /    0       |       1       |       2       |       3       |
   /              |               |               |               |
  |7 6 5 4 3 2 1 0|7 6 5 4 3 2 1 0|7 6 5 4 3 2 1 0|7 6 5 4 3 2 1 0|
  +---------------+---------------+---------------+---------------+
 0| 0x10          | Reserved (0)|D|               | AddRuns       |
```

```
     +---------------+---------------+---------------+---------------+
    4| Reserved (0)                                                  |
     +                                                               +
    8|                                                               |
     +---------------+---------------+---------------+---------------+
   12| Initiator Task Tag or Reserved (0x'ffffffff')                 |
     +---------------+---------------+---------------+---------------+
   16| Reserved (0)                                                  |
     +---------------+---------------+---------------+---------------+
   20| BegRun                                                        |
     +---------------+---------------+---------------+---------------+
   24| RunLength                                                     |
     +---------------+---------------+---------------+---------------+
   28/ Additional Runs or Reserved (0)                               /
    +/                                                               /
     +---------------+---------------+---------------+---------------+
   44
```

SACK request is used to request retransmission of status or data PDUs from the target. It also implicitly acknowledges data or status PDUs. The SACK request indicates to the target the missed status or data runs - where a run is composed of an initial missed StatSN or DataSN and the number of additional missed Status or Data PDUs (0 means only the initial).

2.16.1 D

If 1, indicates that this is a Data SACK; else it is a status SACK. Data SACK for a command MUST precede implicit or explicit status acknowledgement for the given command.
For Data SACK the Initiator Task Tag has to be set to the Initiator Task Tag of the referenced Command.

2.16.2 AddRun

Runs are gaps in sequence numbers as perceived by the receiver and each run is characterized by a starting sequence and a length.

This field specifies the number of additional runs (0, 1 or 2 are the only valid values).

2.16.3 BegRun

First missed DataSN or StatSN

2.16.4 RunLength

Number of additional missed DataSN or StatSN (if BegRun is the only one missing RunLength MUST be 0)

2.17 Ready To Transfer (R2T)

When an initiator has submitted a SCSI Command with data passing from the initiator to the target (WRITE), the target may specify which blocks of data it is ready to receive. In general, the target may

request that the data blocks be delivered in whatever 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) message.

In order to allow write operations without R2T, the initiator and target must have agreed to do so by both sending the UseR2T=no key-pair attribute to each other (either during Login or through the Text Command/Response mechanism).

An R2T MAY be answered with one or more iSCSI Data-out PDU with a matching Target Transfer Tag. If an R2T is answered with a single Data 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 not exceed the Desired Data Length specified in 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, the last PDU should have the F bit set to 1.

The target may send several R2T PDUs and thus have a number of data transfers pending. All outstanding R2T should have different Target Transfer Tags. Outstanding R2Ts MUST be fulfilled by the initiator in the order they where received.

```
Byte /    0       |       1       |       2       |       3       |
   /              |               |               |               |
  |7 6 5 4 3 2 1 0|7 6 5 4 3 2 1 0|7 6 5 4 3 2 1 0|7 6 5 4 3 2 1 0|
  +---------------+---------------+---------------+---------------+
 0| 0x90          |0| Reserved (0)                                |
  +---------------+---------------+---------------+---------------+
 4| Reserved (0)                                                  |
  +                                                               +
 8|                                                               |
  +---------------+---------------+---------------+---------------+
12| Initiator Task Tag                                            |
  +---------------+---------------+---------------+---------------+
16| Target Transfer Tag                                           |
  +---------------+---------------+---------------+---------------+
```



```
20| Reserved (0)                                                  |
  +---------------+---------------+---------------+---------------+
24| ExpCmdSN                                                      |
  +---------------+---------------+---------------+---------------+
28| MaxCmdSN                                                      |
```

```
  +---------------+---------------+---------------+---------------+
32| Desired Data Length                                           |
  +---------------+---------------+---------------+---------------+
36| Buffer Offset                                                 |
  +---------------+---------------+---------------+---------------+
40| Reserved (0)                                                  |
  +                                                               +
  |                                                               |
  +---------------+---------------+---------------+---------------+
44
```

2.17.1 Desired Data Transfer Length and Buffer Offset

The target specifies how many bytes it wants the initiator to send because of this R2T message. 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 indicating the point at which the data transfer should begin, relative to the beginning of the total data transfer.

2.17.2 Target Transfer Tag

The target assigns its own tag to each R2T request that it sends to the initiator. This can be used by the target to easily identify data it receives. The Target Transfer Tag is copied in the outgoing data PDUs and used by the target only. There is no protocol rule about Target Transfer Tag, but it is assumed that it will be used to tag the response data to the target (alone or combination with the LUN).

2.18 Asynchronous Message

An Asynchronous Message may be sent from the target to the initiator without corresponding to a particular command. The target specifies the status for the event and sense data.

```
Byte /    0       |       1       |       2       |       3       |
   /              |               |               |               |
  |7 6 5 4 3 2 1 0|7 6 5 4 3 2 1 0|7 6 5 4 3 2 1 0|7 6 5 4 3 2 1 0|
  +---------------+---------------+---------------+---------------+
 0| 0x91          |0| Reserved (0)                                |
  +---------------+---------------+---------------+---------------+
 4| Logical Unit Number (LUN)                                     |
  +                                                               +
 8|                                                               |
  +---------------+---------------+---------------+---------------+
12/ Reserved (0)                                                  /
 +/                                                               /
  +---------------+---------------+---------------+---------------+
20| StatSN                                                        |
  +---------------+---------------+---------------+---------------+
24| ExpCmdSN                                                      |
  +---------------+---------------+---------------+---------------+
28| MaxCmdSN                                                      |
  +---------------+---------------+---------------+---------------+
32|SCSI Event     |iSCSI Event    | Parameter1 or Reserved (0)    |
  +---------------+---------------+---------------+---------------+
36| Parameter2 or Reserved (0)     | Reserved (0)                 |
  +---------------+---------------+---------------+---------------+
40| Reserved (0)                                                  |
  +---------------+---------------+---------------+---------------+
44| Digests if any...                                             |
  +---------------+---------------+---------------+---------------+
  / Sense Data                                                    /
 +/                                                               /
  +---------------+---------------+---------------+---------------+
```

Some Asynchronous Messages are strictly related to iSCSI while others are related to SCSI [SAM-2}. An Asynchronous Message may contain both types of events.

Please note that StatSN counts this PDU as an acknowledgeable event, allowing initiator and target state synchronization.

2.18.1 iSCSI Event

The codes returned for iSCSI Asynchronous Messages (Events) are:

1 Target is being reset.
2 Target requests Logout - the Parameter1 field will indicate on what CID while the Parameter2 field will indicate the minimum time to reconnect in seconds
3 Target indicates it will/has dropped the connection - the Parameter1 field will indicate on what CID while the Parameter2 field will indicate the minimum time to reconnect in seconds

2.18.2 SCSI Event

The following values are defined. (See [SAM2] for details):

1 An error condition was encountered after command completion.
2 A newly initialized device is available to this initiator.
3 Some other type of unit attention condition has occurred.
4 An asynchronous event has occurred.

Event 4 also includes the case when all Task Sets are being Reset by another Initiator.
Sense Data accompanying the report identifies the condition. The Length parameter is set to the length of the Sense Data.

For new device identification, an iSCSI target MUST support the Device Identification page.

2.19 Third Party Commands

SCSI allows every addressable entity to be either an initiator or a target. In host-to-host communication, each such entity can take on the initiator role. In typical I/O operations between a host and a peripheral subsystem, the host plays the initiator role and the peripheral subsystem plays the target role.

For EXTENDED COPY and other third party SCSI commands, that involve device-to-device communication, such as (EXTENDED) COPY and COMPARE, SCSI defines a copy-manager. The copy-manager takes on the role of initiator in the device-to-device communication. The copy-manager is the "original-target" of the command and acts as initiator for a (variable) number of the devices, called sources and destinations. Sources and destinations act as targets. The whole operation is described by one "master CDB" delivered to the copy-manager and a series of descriptor blocks; each descriptor block addresses a source and destination target and LU and a description of the work to be done in terms of blocks or bytes as required by the device types. The relevant SCSI standards do not require full support of the (EXTENDED) COPY or COMPARE nor do they provide a detailed execution model.

Enabling a FC copy-manager to support iSCSI sources and destinations is subject to coordination with T10.

2.20 Reject

```
Byte /    0       |       1       |       2       |       3       |
   /              |               |               |               |
  |7 6 5 4 3 2 1 0|7 6 5 4 3 2 1 0|7 6 5 4 3 2 1 0|7 6 5 4 3 2 1 0|
  +---------------+---------------+---------------+---------------+
 0|0| 0xef        |0| Reserved (0)                                |
  +---------------+---------------+---------------+---------------+
 4/ Reserved (0)                                                  /
 +/                                                               /
  +---------------+---------------+---------------+---------------+
36| Reason        | Reserved (0)  | First Bad Byte or Rsvd(0)     |
  +---------------+---------------+---------------+---------------+
40| Reserved (0)                                                 |
 +/                                                               /
  +---------------+---------------+---------------+---------------+
44/ Complete Header of Bad Message                                /
 +/                                                               /
  +---------------+---------------+---------------+---------------+
xx
```

It may happen that a target receives a message with a format error (inconsistent fields, reserved fields not 0, inexistent LUN etc.) or a digest error (invalid payload or header). The target returns the header of the message in error as the data of the response.

2.20.1 Reason

The reject Reason is coded as follows:

1 - Format Error
2 - Header Digest Error
3 - Payload Digest Error
4 - Data-SACK Reject
5 - Command Restart Reject
15 - Full Feature Phase Command before login

2.20.2 First Bad Byte

For a format error reject this is the offset of the first offending byte in the header.

3. SCSI mode parameters for iSCSI

This chapter describes fields and mode pages that control and report the behavior of the iSCSI protocol. All fields not described here MUST control the behavior of iSCSI devices as defined by the corresponding command set standard.

3.1 iSCSI Disconnect-Reconnect mode page

3.1.1 Enable Modify Data Pointers bit

This field is used to control incoming data ordering. Incoming data PDUs can be in any order (EMDP = 1) or have to be at continuously increasing addresses (EMDP = 0).
EMDP can also be set by a text-mode key=value pair (InDataOrder).

3.1.2 Maximum Burst Size field (16 bit)

This field is used by iSCSI to define the maximum data payload in iSCSI data PDUs or as immediate data in command PDUs in units of 512 bytes. This value can also be set by a text-mode key=value pair (DataPDULength).

3.1.3 First Burst Size field (16 bit)

This field is used by iSCSI to define the maximum of unsolicited data an iSCSI initiator is allowed to send to the target in units of 512 bytes. This value can also be set by a text-mode key=value pair (FirstBurstSize).

3.1.4 Other fields

No other fields in this page are used by iSCSI.

3.2 iSCSI Logical Unit Control mode page

3.2.1 Protocol Identifier

This field is set to the iSCSI code set by T10 (xx)

3.2.2 Enable CmdRN

When this field is set to 1 the CmdRN field is valid.
This field can also be set by a text-mode key=value pair (EnableCmdRN).

3.3 iSCSI Port Control mode page

No field in this page is used by iSCSI

4. Login phase

In the rest of this chapter whenever we mention security we mean security and/or data integrity.

The login phase establishes an iSCSI session between initiator and target. It sets the iSCSI protocol parameters, security parameters, and authenticates the initiator and target to each other.

Operational parameters MAY be negotiated within or outside (after) the login phase.

Security MUST be completely negotiated within the Login Phase or provided by external means (e.g., IPSec).

In some environments, a target or an initiator will not be interested in authenticating their counterpart. It is possible to achieve this through the Login Command and Response.

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

Authentication and a Secure Channel setup MAY be performed independent of iSCSI (as when using tunneling IPSec or some implementations of transport IPSec) in which case the Login phase can be reduced to operational parameter negotiations.

The login phase is implemented via login and text commands and responses only. The login command is sent from the initiator to the target in order to start the login phase, and the login response is sent from the target to the initiator to conclude the login phase. Text messages are used to implement negotiation, establish security and set operational parameters.

The whole login phase is considered as a single task and has a single Initiator Task Tag (very much like the linked SCSI commands).

The login phase sequence of commands and responses proceeds as follows:

- Login command (mandatory)
- Login Partial-Response (optional)
- Text Command(s) and Response(s) (optional)
- Login Final-Response (mandatory)

The Login Final-Response can come only as a response to a Login command with the F bit set to 1 or a Text Command with the F bit set to 1.

4.1 Login phase start

The login phase starts with a login request via a login command from the initiator to the target. The login request includes:

- -Protocol version supported by the initiator (currently 0x'01')
- -Session and connection Ids
- -Security/Integrity Parameters OR
- -iSCSI operational parameters

A target MAY use the Initiator WWUI as part of its access control mechanism; therefore, the Initiator WWUI must be sent before the target is required to disclose its LUs.

If the target WWUI is going to be used in determining the security mode or it is implicit part of authentication, then the target WWUI MUST be sent in the login command of the first connection of a session to identify the storage endpoint of the session. However, it is OPTIONAL for all the connections after the first (it will be ignored by the target for new connections within an existing session). If the target WWUI is going to be used only for access control it can be sent after the Security Context Complete is achieved. A unknown target can be accessed by using "iSCSI" as a placeholder for the WWUI.
The WWUIs MUST be in text command format.

The target can answer in the following ways:

- -Login Response with Login Reject (and Final bit 1). This is an immediate rejection from the target causing the session to terminate.
- -Login Response with Login Accept with session ID and iSCSI parameters and F bit set to 1. This is a valid response only if the Login Command had also the F bit set to 1. In this case, the target does not support any security or authentication mechanism and starts with the session immediately (enters full feature phase)

-Login Response with Final bit 0 indicating the start of a negotiation sequence. The response includes the protocol version supported by the target and EITHER security/integrity parameters OR iSCSI parameters (when no security/integrity mechanism is chosen) supported by the target. It also indicates what sequence is expected next (security/integrity or iSCSI parameters negotiation). The initiator MAY decide to drop the connection if the sequence is not what it expects (e.g., an initiator expecting a security/integrity sequence and getting a response indicating that iSCSI parameters negotiation is the next phase expected by the initiator).

4.2 iSCSI Security and Integrity negotiation

The security exchange sets the security mechanism and authenticates the user and the target to each other. The exchange proceeds according to the algorithms that were chosen in the negotiation phase and is conducted by the text commands key=value parameters.

The negotiable security mechanisms include the following modes:

-Initiator-target authentication - the host and the target authenticate themselves to each other. A negotiable algorithm, e.g., Kerberos, provides this feature.
-Message integrity - an integrity/authentication digest is attached to each packet. The algorithm is negotiable.

Using IPsec for encryption or authentication may eliminate the need for security negotiation at the iSCSI level (for example, ISAKMP for IPsec).

If security is established in the login phase note that:

-After the security context negotiation is complete, each iSCSI message MUST include the appropriate digest field if any.

-The iSCSI parameter negotiation (non-security parameters) SHOULD start only after security is established. This should be performed using text commands.

The negotiation proceeds as follows:

-The initiator sends a text command with an ordered list of the

options it supports for each subject (authentication algorithm, iSCSI parameters and so on). The options are listed from the most preferable (to the initiator) to the least.

-The target MUST reply with the first option in the list it supports. The parameters are encoded in Unicode - UTF8 as key=value. The initiator MAY send proprietary options as well. The "none" option, if allowed, MUST be included in the list, indicating no algorithm supported by the target. If security is to be established, the initiator MUST NOT send parameters other than security parameters in the login command. The general parameters should be negotiated only after security is established at the desired level. Any operational parameters sent before establishing a secure context MUST be reset by both the target and the initiator when establishing the security context. For a list of security, parameters see Appendix A.

-Every party in the security negotiation will indicate that it has completed building its security context (has all the required information) by sending the key=value pair:

SecurityContextComplete=Yes

The other party will either offer some more parameters or answer with the same:

SecurityContextComplete=Yes

The party that is ready will keep sending the SecurityContextComplete=Yes pair (in addition to new security parameters if required) until the handshake is complete.

If the initiator has been the last to complete the handshake it MUST NOT start sending operational parameters within the same text command; a text response including only SecurityContextComplete=Yes will conclude the security sub phase.

If the target has been the last to complete the handshake, the initiator can start the operational parameter negotiation with the next text command; the security negotiation sub phase has ended with the target text response.

All PDUs sent after the security negotiation sub phase MUST be built using the agreed security.

4.3 Operational parameter negotiation during the login phase

Operational parameter negotiation during the login MAY be done:

- starting with the Login command if the initiator does not offer any security/ integrity option
- starting immediately after the security/integrity negotiation if the initiator and target perform such a negotiation
- starting immediately after the Login response with Final bit 0 if the initiator does offer security/integrity options but the target chose none.

Operational parameter negotiation MAY involve several request-response exchanges (login and/or text) always driven by the initiator. The initiator MUST indicate its intent to terminate the negotiation by setting the F bit to 1; the target will set the F bit to 1 on the last response and that last response must be the Login Response.
If the target responds to a text or Login command with the F bit set to 1 with a text response with the F bit set to 0, or a login response with the text bit set to 0, the initiator must keep sending text command (even empty) with the F bit set to 1 until it gets the Login Response with the F bit set to 1.

A target MUST not send more than one Login Response with the F bit set to 0.

An initiator MUST send a single Login command per connection per session.

5. Operational parameter negotiation outside the login phase

Operational parameters MAY be negotiated outside (after) the login phase.

Operational parameter negotiation MAY involve several text request-response exchanges always driven by the initiator. The initiator MUST indicate its intent to terminate the negotiation by setting the F bit to 1; the target will set the F bit to 1 on the last response.
If the target responds to a text command with the F bit set to 1 with a text response with the F bit set to 0, the initiator must keep sending text command (even empty) with the F bit set to 1 until it gets the text response with the F bit set to 1.

6. iSCSI Error Handling and 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 target, incoming data (read data) MAY be kept for recovery or it can be re-read from a device server.

It is further assumed that a target will keep the "status & sense" for a command it has executed while the total number of outstanding commands and executed commands does not exceed its limit and status has not been acknowledged.

6.1 Format errors

Explicit violations of the rules stated in this document are format errors.

While a session is active, whenever a target receives an iSCSI PDU with a format error, it MUST answer with a Reject iSCSI PDU with a Reason-code of Format Error. It MUST also provide a 2-byte offset of the first offending byte in the rejected PDU.

When an initiator receives an iSCSI PDU with a format error, for which it has an outstanding task, it MUST abort the target task and report the error through an appropriate service response (e.g., Target Failure). The exact coding of the service response is outside the scope of this document.

6.2 Digest errors

When a target receives an iSCSI PDU with a header digest error or a payload digest error in an iSCSI PDU it MUST answer with a Reject iSCSI PDU with a Reason-code of Header-Digest-error or Data-Digest-Error and discard the offending PDU. If the error is a Data-Digest-

Error in a Data-PDU, the target MUST either request retransmission with a R2T or answer with a Reject iSCSI PDU and abort the task.

When an initiator receives an iSCSI PDU with a header digest error, it MUST discard it. When an initiator receives any iSCSI PDU other than a data PDU, with a Data-Digest-Error, and this PDU is part of a task (has an Initiator Task Tag set) it MUST discard the PDU and it MAY restart the task (reissue the command with the same Initiator Task Tag and the X-bit set to 1). If the reissued command is a SCSI command and it implies Read Data (Expected Data Length is not 0), the

reissued command will also include the sequence number of the Next Data Packet expected by the initiator (0 if there was no data packet yet).

When an initiator receives an iSCSI data PDU with a Data-Digest error, it must discard the PDU and it MUST either request the missing data PDUs through SACK or terminate the command with an error.

6.3 Sequence errors

When an initiator receives an iSCSI data PDU with an out-of-order DataSN or a SCSI command response PDU with an EndDataSN implying missing data PDUs it MAY request the missing data PDUs through a data SACK PDU or handle this case as a connection failure. In its turn, the target MUST either reject the SACK with a Reject PDU with a reason-code of Data-SACK-Reject or resend the data PDU.

When an initiator receives an iSCSI status PDU with an out-of-order StatSN implying missing responses, it MUST either request the missing response PDUs through a status SACK or handle this case as a connection failure. The target MUST reissue the missing responses. As a side effect of receiving the missing responses, the initiator might discover missing data PDUs. The initiator MUST NOT acknowledge (explicitly through ExpStatRN or implicitly through a status SACK) the received responses until it has completed receiving all the data PDUs of a SCSI command.

6.4 Protocol Errors

The authors recognize that mapping framed messages over a "stream" connection (like TCP) makes the proposed mechanisms vulnerable to simple software framing errors and introducing framing mechanisms may be onerous for performance and bandwidth. Command Sequence Numbers and the above mechanisms for connection drop and reestablishment will help handle this type of mapping errors.

6.5 Connection failure

iSCSI can keep a session in operation if it is able to keep/establish at least one TCP connection between the initiator and target in a timely fashion. It is assumed that targets and/or initiators will recognize a failing connection by either transport level means (TCP) or by a gap in the command or response stream that is not filled for a long time, or by a failing iSCSI NOP-ping (the later MAY be used periodically by highly reliable implementations). Initiators and targets MAY also use the keep-alive option on the TCP connection to enable early link failure detection on otherwise idle links.

At connection failure, initiator and target MUST either attempt connection recovery within the session or session recovery.

6.6 Session Errors

If all the connections of a session fail and can't 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. It will terminate all outstanding requests with a appropriate response before initiating a new session. The target will take the following actions:

- Reset the TCP connections (close the session).
- Abort all Tasks in the task set for the corresponding initiator.

6.7 Recovery levels

iSCSI enables the following levels of recovery (in increasing coverage order):

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

The recovery scenarios detailed in the rest of this part are representative rather than exclusive. In every case they detail the lowest level recovery that MAY be attempted leaving the implementer to decide under which circumstances to raise the recovery level

and/or what recovery levels to implement.

At all levels, 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 (like ACA) have to be considered.

6.7.1 Recovery within-task

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

(1)Lost data PDU - a data PDU may be lost due to a header digest error or a data digest error. In case of a data digest error, the error is recognized immediately, and the target MAY request the missing data through R2T. In case of a header digest error, the target will recognize the missing data either when receiving a subsequent piece out of sequence or by a timeout in completing a sequence (no data or partial-data-and-no-F-bit). In this case, too, the target MAY request the missing data through a R2T.

The time to timeout to be used by a target is outside the scope of this document.

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

(1)Lost data PDU - a data PDU may be lost due to a header digest error or a data digest error. In case of a data digest error, the error is recognized immediately and the initiator MAY request the missing data through SACK. In case of a header digest error, the initiator will recognize the missing data either when receiving a subsequent piece out of sequence or by a timeout in completing a sequence (no status). In this case, too, the initiator MAY request the missing data through a SACK.

The time to timeout to be used by an initiator is outside the scope of this document.

Both the iSCSI target and initiator MAY resort to a more drastic, not-within-task recovery procedure in any of these cases.

An initiator MAY reissue a command when missing data or status.

An iSCSI target MAY reject a data-SACK and terminate the command with an iSCSI error response of SACK rejected.

An iSCSI initiator MUST accept an R2T.

An iSCSI target on detecting missing data MAY terminate the command with an iSCSI error response of Delivery Subsystem Failure.

6.7.1.1 Recovery within-connection

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

(1)Lost iSCSI numbered Response recognized by either receiving it with a data digest error or receiving a Response PDU with a higher StatSN than expected. The initiator MAY request the missing responses through SACK, in which case the target MUST reissue them.
(2)Requests not acknowledged for a long time. Requests are acknowledged explicitly through ExpCmdSN or implicitly by receiving data and/or status. The initiator MAY reissue non-acknowledged commands. The reissued, non-acknowledged commands MUST carry their original CmdSN and the X (retry) flag set to 1. Please note that this is the only case in which the reissued command will carry the same CmdSN.
N.B. While the original connection for a command is still "active" (has not been logged-out or restarted), any command MUST be retried only on the original connection. After logging out the original connection, commands can be retried on a different connection, but must still carry the original CmdSN.

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

(1)Status/Response not acknowledged for a long time. The target MAY issue a NOP-IN (with or with the P bit set to 1 or 0) indicating in the StatSN field the next status number it is going to issue. This will help the initiator detect missing StatSN and issue a SACK-status.

The time to timeout by both initiator and target are outside the scope of this document.

Both the iSCSI target and initiator MAY resort to a more drastic, not-within-connection recovery procedure in any of those cases.

6.7.1.2 Recovery within-session

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

(1)TCP connection failure. The initiator MUST close the connection following which it MUST either Logout the failed connection, or Login with an implied Logout, and reissue all commands associated with the failed connection on another connection (that MAY be a newly established connection) with the X (retry) flag set to 1.

N.B. The logout function is mandatory, while a new connection establishment is mandatory only if the failed connection was the last or only connection in the session

N.B. As an alternative to Logout and reissue commands, the initiator MAY instead reset the target and terminate all outstanding commands with a service response indicating Delivery Subsystem Failure. The initiator MUST perform one of the two actions.

(2)Receiving an Asynchronous Message requiring recovery Logout. The initiator MUST handle it as a TCP connection failure for the connection referred to in the message.

At an iSCSI target, the following cases lend themselves to within-session recovery

(1)TCP connection failure. The target MUST close the connection and then, if more than one connection is available, the target SHOULD send an Asynchronous Message indicating it has dropped the connection. Following that, the target will wait for the initiator to continue recovery.

6.7.1.3 Session Recovery

Session recovery is to be performed when all other recovery attempts have failed. Very simple initiators and targets MAY perform session recovery on all iSCSI errors - and thus place the burden of recovery on the SCSI layer and above.

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

7. 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 tag mechanism was designed to enable RDMA at the iSCSI level or lower.

The guiding assumption made throughout the design of this protocol was that targets are resource constrained relative to initiators.

7.1 Multiple Network Adapters

The iSCSI protocol allows multiple connections, not all of which need 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 insure 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.

7.2 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 mandates support for autosense.

ACA helps preserving ordered command execution in presence of errors. As iSCSI can have many commands in-flight between initiator and target iSCSI mandates support for ACA.

8. 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 subnet 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 (pretending as another identity, message insertion, deletion, and modification) and MAY provide means of protection against passive attacks (eavesdropping, gaining advantage by analyzing the data sent over the line).

The following section describes the security protection modes that should
be provided by an iSCSI implementation.

Authentication and a Secure Channel setup MAY be performed independent of iSCSI (as when using tunneling IPSec or some implementations of transport IPSec).

8.1 iSCSI Security Protection Modes

8.1.1 No Security

This mode does not authenticate nor does it encrypt data. This mode should only be used in environments where the security risk is

minimal and configuration errors are improbable.

8.1.2 Initiator-Target Authentication

In this mode, the target authenticates the initiator and the initiator optionally authenticates the target. An attacker should not gain any advantage by inspecting the authentication phase messages (so, e.g., sending clear password is out of question). This mode protects against an unauthorized access to storage resources by using a false identity (SPOOFING). Once the authentication phase is completed, all messages are sent and received in clear. This mode should only be used when there is minimal risk to man-in-the-middle attacks, eavesdropping, message insertion, deletion, and modification.

8.1.3 Data Integrity and Authentication

This mode provides origin authentication and data integrity for every message that is sent after a security context is established. It protects against man-in-the-middle attacks, message insertion, deletion, and modification.

It is possible to use different authentication mechanisms for headers and data.

Every compliant iSCSI initiator and target MUST be able to provide initiator-target authentication and data integrity and authentication. This quality of protection MAY be achieved on every connection through properly configured IPSec involving only administrative (indirect) interaction with iSCSI implementations.

8.1.4 Encryption

This mode provides data privacy in addition to data integrity and authentication, and protects against eavesdropping, man-in-the-middle attacks, message insertion, deletion, and modification.

A connection or multiple connections MAY be protected end-to-end or partial-path (gateway tunneling) by using IPSec.

9. IANA Considerations

There will be a well-known port for iSCSI connections. This well known port will be registered with IANA.

10. References and Bibliography

[AC] A detailed proposal for Access Control, Jim Hafner, T10/99-245
[ALTC] Internet Draft: Alternative checksums (work in progress)
[BOOT] P. Sarkar & team draft-ietf-ips-iscsi-boot-01.txt
[CAM] ANSI X3.232-199X, Common Access Method-3 (Cam-3)
[CRC] ISO 3309, High-Level Data Link Control (CRC 32)
[FIPS-180-1] FIPS-Secure Hash Standard
[FIPS-186-2] FIPS-Digital Signature Standard
[NDT] M. Bakke & team, draft-ietf-ips-iSCSI-NamingAndDiscovery-00.txt
[PKIX-Part1] Housley, R., et al, "Internet X.509 Public Key Infrastructure, Certificate and CRL Profile", Internet Draft, draft-ietf-pkix-ipki-part1-11.txt
[RFC793] Transmission Control Protocol, RFC 793
[RFC1122] Requirements for Internet Hosts-Communication Layer RFC1122, R. Braden (editor)
[RFC-1510] J. Kohl, C. Neuman, "The Kerberos Network Authentication Service (V5)", September 1993.
[RFC1766] Alvestrand, H., "Tags for the Identification of Languages", March 1995.
[RFC1964] J. Linn, The Kerberos Version 5 GSS-API Mechanism ,

June 1996.
[RFC1982] Elz, R., Bush, R., "Serial Number Arithmetic", RFC 1982, August 1996.
[RFC2026] Bradner, S., "The Internet Standards Process -- Revision 3", RFC 2026, October 1996.
[RFC-2044] Yergeau, F., "UTF-8, a Transformation Format of Unicode and ISO 10646", October 1996.
[RFC2104] Krawczyk, H., Bellare, M., and Canetti, R., "HMAC: Keyed-Hashing for Message Authentication", February 1997
[RFC2119] Bradner, S. "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.
[RFC2144] Adams, C., "The CAST-128 Encryption Algorithm", May 1997.
[RFC2234] D. Crocker, P. Overell Augmented BNF for Syntax Specifications: ABNF
[RFC2313] B. Kaliski, PKCS #1: RSA Encryption, Version 1.5
[RFC2434] T. Narten, and H. Avestrand, "Guidelines for Writing an IANA Considerations Section in RFCs.", RFC2434, October 1998.
[RFC2440] Callas, J., et al, "OpenPGP Message Format", November 1998.
[RFC2945], Wu, T., "The SRP Authentication and Key Exhange System", September 2000.
[SAM2] ANSI X3.270-1998, SCSI-3 Architecture Model (SAM-2)

[SBC] ANSI X3.306-199X, SCSI-3 Block Commands (SBC)
[SCSI2] ANSI X3.131-1994, SCSI-2
[Schneier] Schneier, B., "Applied Cryptography Second Edition: protocols, algorithms, and source code in C", 2nd edition, John Wiley & Sons, New York, NY, 1996.
[SPC] ANSI X3.301-199X, SCSI-3 Primary Commands (SPC)
[Wolf94] J. K. Wolf et al. The Single Burst Error Detection Performance of Binary Cyclic Codes - IEEE Transactions on Communications, Vol. 42 No. 1
[Wolf88] J. K. Wolf et al. The Exact Evaluation of the Probability of Undetected Error for Certain Shortened Binary CRC Codes - Proc. MILCOM 1988 pp 15.2.1-15.2.6

11. Author's Addresses

Julian Satran
Kalman Meth
Ofer Biran
IBM, Haifa Research Lab
MATAM - Advanced Technology Center
Haifa 31905, Israel
Phone +972 4 829 6211
Email: Julian_Satran@vnet.ibm.com meth@il.ibm.com
biran@il.ibm.com

Daniel F. Smith
IBM Almaden Research Center
650 Harry Road
San Jose, CA 95120-6099, USA
Phone: +1 408 927 2072
Email: dfsmith@almaden.ibm.com

Costa Sapuntzakis
Cisco Systems, Inc.
170 W. Tasman Drive
San Jose, CA 95134, USA
Phone: +1 408 525 5497
Email: csapuntz@cisco.com

Randy Haagens
Hewlett-Packard Company
8000 Foothills Blvd.
Roseville, CA 95747-5668, USA
Phone: +1 (916) 785-4578
E-mail: Randy_Haagens@hp.com

Matt Wakeley
Agilent Technologies
1101 Creekside Ridge Drive
Suite 100, M/S RH21
Roseville, CA 95661
Phone: +1 (916) 788-5670
E-Mail: matt_wakeley@agilent.com

Efri Zeidner
SANGate
Israel
efri@sangate.com

Paul von Stamwitz
Adaptec, Inc.
691 South Milpitas Boulevard
Milpitas, CA 95035
Phone: +1(408) 957-5660
E-mail: paulv@corp.adaptec.com

Luciano Dalle Ore
Quantum Corp.
Phone: +1(408) 232 6524
E-mail: ldalleore@snapserver.com

Yaron Klein
SANRAD
24 Raul Valenberg St.
Tel-Aviv, 69719 Israel
Phone: +972-3-7659998
E-mail: klein@sanrad.com

Comments may be sent to Julian Satran

Appendix A. iSCSI Security and Integrity

01 Security keys and values

The parameters (keys) negotiated for security are:

- Digests (HeaderDigest, DataDigest)
- Authentication methods (InitAuth, TargetAuth)

Digests enable checking end-to-end data integrity (beyond the integrity checks provided by the link layers and covering the whole communication path including all elements that may change the network level PDUs - like routers, switches, proxies etc.).

The following table lists cyclic integrity checksums that can be

negotiated for the digests and MUST be implemented by every iSCSI initiator and target. Note that these digest options have only error detection significance.

Name	Description
crc-32Q	32 bit CRC
crc-64	64 bit CRC
none	no digest

The generator polynomials for those digests are:

```
crc-32Q   - x**32+x**31+x**24+x**22+x**16+x**14+x**8+x**7+
            x**5+x**7+x**5+x**3+x+1
crc-64    - TBD
```

crc-64 MUST NOT be used for HeaderDigest.
Cyclic codes are particularly well suited for hardware implementations.

Implementations MAY also negotiate digests with security significance for data authentication and integrity as detailed in the following table:

Name	Description	Definition
KRB5_MD5	the SGN_CKSUM field (8 bytes) of the GSS_GetMIC() token in GSS_KRB5_INTEG_C_QOP_MD5 QOP (partial MD5 ("MD2.5"))	RFC-1964
KRB5_DES_MD5	the SGN_CKSUM field (8 bytes) of the GSS_GetMIC() token in GSS_KRB5_INTEG_C_QOP_DES_MD5 QOP (DES MAC of MD5)	RFC-1964
KRB5_DES_MAC	the SGN_CKSUM field (8 bytes)	RFC-1964

```
|               | of the GSS_GetMIC() token in  |             |
|               | GSS_KRB5_INTEG_C_QOP_ DES_MAC |             |
|               | QOP (DES MAC)                 |             |
+-------------------------------------------------------------+
```

Note: the KRB5_* digests are allowed only when combined with KRB5 initiator authentication method (see below). I.e., the initiator may offer one of these digests only if he also offers KRB5 as InitAuth method, and the target may respond with one of these digests only if he also responds with KRB5 as the InitAuth method.

Other and proprietary algorithms MAY also be negotiated.
The none value is the only one that MUST be supported.

The following table details authentication methods:

```
+-------------------------------------------------------------+
| Name          | Description                                 |
+-------------------------------------------------------------+
| KERB5         | Kerberos V5                                   |
+-------------------------------------------------------------+
| srp           | Secure Remote Password                      |
+-------------------------------------------------------------+
| none          | No authentication                           |
+-------------------------------------------------------------+
```

KERB5 is defined in [RFC-1510] and Secure Remote Password is defined in [RFC-2945].

Note: KERB5 target authentication is allowed only when combined with KERB5 client authentication. I.e., the initiator may offer KERB5 as TargetAuth method only if he also offers KERB5 as InitAuth method, and the target may respond with KERB5 for TargetAuth only if he also responds KERB5 for InitAuth.

02 Authentication

The authentication exchange authenticates the initiator and target to each other. Authentication is not mandatory and is distinct from the data integrity exchange.

Different levels of authentication can be applied such as initiator authentication, target authentication or both.

The authentication methods to be used are KERB5, SRP or proprietary.

For Kerberos [RFC-1510], the initiator MUST use:

```
Authenticate=<blob>
```

where blob contains the KRB_AP_REQ message encoded as a number.

If the initiator has selected the mutual authentication option, the target MUST either return an error or use:

```
Authenticate=<blob>
```

Where blob contains the KRB_AP_REP message encoded as a hexadecimal string. The format of these messages is defined in [RFC1510].

For SRP [RFC2945], the initiator MUST use:

```
Authenticate=U,A
```

The target MUST either return an error or reply with:

```
Authenticate=s,B
```

The initiator MUST either abort or continue with:

```
AuthenticateNext=M1
```

The target MUST either return an error or reply with


```
AuthenticateNext=M2
```

Where U, A, s, B, M1 and M2 are numbers defined in [RFC2945].

03 Login Phase examples

In the first example, the initiator and target authenticate each other via Kerberos:

```
I-> Login InitiatorWWUI=com.os.hostid.77
TargetWWUI=com.acme.diskarray.sn.88
HeaderDigest=KRB5_MD5,KRB5_DES_MAC,crc-32Q,none
```

```
DataDigest=crc-32Q,none InitAuth=srp,KERB5,none
TargetAuth=KERB5,none

T-> Login-PR HeaderDigest=KERB5_MD5 DataDigest=crc-32Q
InitAuth=KERB5 TargetAuth=KERB5

  (Login-PR stands for Login-Partial-Response)

I-> Text Authenticate=krb_ap_req
(krb_ap_req contains the KERB5 ticket and authenticator)

If the authentication is successful, the target proceeds with:

T-> Text Authenticate=krb_ap_rep SecurityContextComplete=Yes
krb_ap_rep is the KERB5 mutual authentication reply)

If the authentication is successful, the initiator proceeds:

I-> Text SecurityContextComplete=Yes
T-> Text SecurityContextComplete=Yes

From this point on, any Text command and each PDU thereafter
will have a KERB5_MD5 digest for the header and a crc-32Q for
the data.

The initiator may proceed:

I-> Text ... iSCSI parameters
T-> Text ... iSCSI parameters

And at the end:
```



```
I-> Text optional iSCSI parameters F bit set to 1
T-> Login "login accept" TargetWWUI=com.acme.diskarray.sn.88

If the initiator authentication by the target was not
successful, the target responds with:

T-> Login "login reject"

instead of the Text krb_ap_rep message, and terminates the
connection.

If the target authentication by the initiator was not
successful, the initiator terminates the connection (without
responding to the Text krb_ap_rep message).
```

In the next example only the initiator is authenticated by the target via Kerberos:

```
I-> Login InitiatorWWUI=com.os.hostid.77
TargetWWUI=com.acme.diskarray.sn.88
HeaderDigest=KRB5_MD5,KRB5_DES_MAC,crc-32Q,none
DataDigest=crc-32Q,none InitAuth=srp,KERB5,none
T-> Login-PR HeaderDigest=KERB5_MD5 DataDigest=crc-32Q
InitAuth=KERB5

I-> Text Authenticate=krb_ap_req SecurityContextComplete=Yes
T-> Text SecurityContextComplete=Yes
```

From this point on, any Text command and each PDU thereafter must have a KERB5_MD5 digest for the header and a crc-32Q for the data.

```
I-> Text ... iSCSI parameters
T-> Text ... iSCSI parameters

. . .

T-> Login "login accept" TargetWWUI=com.acme.diskarray.sn.88
```

In the next example, the target authenticates the initiator via SRP.

```
I-> Login InitiatorWWUI=com.os.hostid.77
TargetWWUI=com.acme.diskarray.sn.88 HeaderDigest=crc-32Q,none
DataDigest=crc-32Q,crc-64, none InitAuth=KERB5,srp,none
TargetAuth=none
```



```
T-> Login-PR HeaderDigest=crc-32Q DataDigest=crc-64
InitAuth=srp
I-> Text Authenticate=U,A
T-> Text Authenticate=s,B
I-> Text AuthenticateNext=M1
```

If authentication is successful, the target proceeds:

```
T-> Text AuthenticateNext=M2 SecurityContextComplete=Yes
I-> Text SecurityContextComplete=Yes
T-> Text SecurityContextComplete=Yes
```

Where U, A, s, B, M1 and M2 are numbers defined in [RFC2945].

From this point on, any Text command and each PDU thereafter will have a crc-32Q digest for the header and a crc-64 for the data.

```
I-> Text   iSCSI parameters
T-> Text   iSCSI parameters
```

And at the end:

```
I-> Text optional iSCSI parameters and F bit set to 1
T-> Login "login accept" TargetWWUI=com.acme.diskarray.sn.88
```

If the authentication was not successful, the target responds with

```
T-> Login "login reject"
```

Instead of the T-> Text AuthenticateNext=M2 ... message and terminates the connection.

In the next example, the initiator does not offer any security/integrity parameters, so he may offer iSCSI parameters on the Login message with the F bit set to 1, and the target may respond with a final Login message immediately:

```
I-> Login InitiatorWWUI=com.os.hostid.77
TargetWWUI=com.acme.diskarray.sn.88  ... iSCSI parameters
T-> Login "login accept"
TargetWWUI=com.acme.diskarray.sn.88  ... ISCSI parameters
```


In the next example, the initiator does offer security/integrity parameters on the Login message, but the target does not choose any (i.e., chooses the "none" values):

```
I-> Login InitiatorWWUI=com.os.hostid.77
TargetWWUI=com.acme.diskarray.sn.88 HeaderDigest=crc-32Q,none
DataDigest=crc-32Q,crc-64,none InitAuth:KERB5,srp
T-> Login-PR

I-> Text ... iSCSI parameters
T-> Text ... iSCSI parameters
```

And at the end:

```
I-> Text optional iSCSI parameters F bit set to 1
T-> Login "login accept" TargetWWUI=com.acme.diskarray.sn.88
```

Note that no SecurityContextComplete=Yes is required since no security mechanism was chosen.

Appendix B. Examples

04 Read operation example

```
|Initiator Function|    Message Type       |  Target Function     |
+------------------+-----------------------+----------------------+
|  Command request |SCSI Command (READ)>>> |                      |
|  (read)          |                       |                      |
+------------------+-----------------------+----------------------+
|                  |                       | Prepare Data Transfer|
```

```
+------------------+----------------------+---------------------+
|   Receive Data   |   <<< SCSI Data      |   Send Data         |
+------------------+----------------------+---------------------+
|   Receive Data   |   <<< SCSI Data      |   Send Data         |
+------------------+----------------------+---------------------+
|   Receive Data   |   <<< SCSI Data      |   Send Data         |
+------------------+----------------------+---------------------+
|                  |   <<< SCSI Response  |Send Status and Sense |
+------------------+----------------------+---------------------+
| Command Complete |                      |                     |
+------------------+----------------------+---------------------+
```


05 Write operation example

```
+------------------+----------------------+--------------------+
|Initiator Function|    Message Type      |  Target Function   |
+------------------+----------------------+--------------------+
|  Command request |SCSI Command (WRITE)>>>| Receive command    |
```

```
|  (write)         |                        | and queue it        |
+------------------+------------------------+---------------------+
|                  |                        | Process old commands|
+------------------+------------------------+---------------------+
|                  |                        | Ready to process    |
|                  |   <<< R2T              | WRITE command       |
+------------------+------------------------+---------------------+
|   Send Data      |   SCSI Data >>>        |   Receive Data      |
+------------------+------------------------+---------------------+
|                  |   <<< R2T              |                     |
+------------------+------------------------+---------------------+
|                  |   <<< R2T              |                     |
+------------------+------------------------+---------------------+
|   Send Data      |   SCSI Data >>>        |   Receive Data      |
+------------------+------------------------+---------------------+
|   Send Data      |   SCSI Data >>>        |   Receive Data      |
+------------------+------------------------+---------------------+
|                  |   <<< SCSI Response    |Send Status and Sense|
+------------------+------------------------+---------------------+
| Command Complete |                        |                     |
+------------------+------------------------+---------------------+
```


Appendix C. Synch and Steering with Fixed Interval Markers

This appendix presents a simple scheme for synchronization (PDU boundary retrieval). It uses markers including synchronization information placed at fixed intervals in the TCP stream.

A Marker consists of:

```
Byte /    0       |       1       |       2       |       3       |
   /              |               |               |               |
  |7 6 5 4 3 2 1 0|7 6 5 4 3 2 1 0|7 6 5 4 3 2 1 0|7 6 5 4 3 2 1 0|
  +---------------+---------------+---------------+---------------+
 0| Next-iSCSI-PDU-start pointer - copy #1                        |
  +---------------+---------------+---------------+---------------+
 4| Next-iSCSI-PDU-start pointer - copy #2                        |
  +---------------+---------------+---------------+---------------+
```

The marker uses 2 copies of pointer so that a marker spanning a TCP packet boundary will leave at least one valid copy in one of the packets.

The use of markers is negotiable. Initiator and target MAY indicate their readiness to receive and/or send markers, during login, separately for each connection. The default is NO. In certain environments a sender not willing to supply markers to a receiver willing to accept markers MAY suffer from a considerable performance degradation.

06 Markers At Fixed Intervals

At fixed intervals in the TCP byte stream, a "Marker" is inserted. This Marker indicates the offset to the next iSCSI message header.

The Marker is eight bytes in length, and contains two 32-bit offset fields that indicate how many bytes to skip in the TCP stream to find the next iSCSI message header. There are two copies of the offset in the Marker to handle the case where the Marker straddles a TCP segment boundary. Each end of the iSCSI session specifies during login the interval of the Marker it will be receiving, or disables the Marker altogether. If a receiver indicates that it desires a Marker, the sender SHOULD agree (during negotiation) and provide the Marker at the desired interval.

The marker interval (and the initial marker-less interval) are counted in terms of the TCP stream data. Anything counted in the TCP sequence-number is counted for the interval and the initial marker-less interval (this specifically includes any bytes "inserted" in the TCP stream by an UFL).

When reduced to iSCSI terms markers MUST point to a 4 byte word boundary in the stream - the last 2 bits of each marker word are reserved and will be considered 0 for offset computation.

Padding iSCSI PDU payloads to 4 byte word boundaries simplifies marker manipulation.

07 Initial marker-less interval

To enable the connection setup including the login phase negotiation, the negotiated marking will be started at a negotiated boundary in the stream. The marker-less interval will not be less than 64 kbytes and the default will be 64 kbytes.

Appendix D. Login/Text miscellaneous keys

ISID and TSID form collectively the SSID (session id). A TSID of zero

indicates a leading connection. Only a leading connection login can carry session specific parameters, e.g. MaxConnections, the maximum immediate data length requested, etc..

08 MaxConnections

MaxConnections=<number-from-1-to-65535>

Default is 8.

Initiator and target negotiate the maximum number of connections requested/acceptable. The lower of the 2 numbers is selected.

09 TargetWWUI

TargetWWUI=<wwui>

Examples:

```
TargetWWUI=com.disk-vendor.diskarrays.sn.45678
TargetWWUI=eui.020000023B040506
TargetWWUI=oui.00023B.target.45
TargetWWUI=iSCSI
```

This key is provided by the initiator of the TCP connection to the remote endpoint. The Target WWUI specifies the worldwide unique name of the target. The non-unique default name "iSCSI" may be used to indicate whatever default target exists at the address to which the connection was made.

The TargetWWUI key may also be returned by the "SendTargets" text command, described in [NDT].

10 InitiatorWWUI

InitiatorWWUI=<wwui>

Examples:

```
InitiatorWWUI=com.os-vendor.plan9.cdrom.12345
InitiatorWWUI=com.service-provider.users.customer235.host90
InitiatorWWUI=iSCSI
```


The Initiator key enables the initiator to identify itself to the remote endpoint. The use of the default WWUI "iSCSI" is interpreted

as "other side of TCP connection". The target may silently ignore this key if it does not support it, and does not need to track or verify which initiators use it. A target that supports this field may use it to allow or deny access to an initiator.

11 TargetAlias

TargetAlias=<UTF-8 string>

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, it may be communicated to the initiator during a Login Response message. This string is not used as an identifier, but can be displayed by the initiator's user interface in a list of targets to which it is connected.

This key is OPTIONAL, and MAY be returned by a target within a Login Response. This field may also be returned in the response to the "SendTargets" text command.

12 InitiatorAlias

InitiatorAlias=<UTF-8 string>

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 may be communicated to the initiator during a Login Request message. If not, the host name can be used instead. This string is not used as an identifier, but can be displayed by the target's user interface in a list of initiators to which it is connected.

This key is OPTIONAL, and MAY be sent by an initiator within a Login Request.

13 TargetAddress

TargetAddress=domainname[:port]/wwui

Examples:

```
TargetAddress=10.0.0.1/com.disk-vendor.diskarrays.sn.45678
TargetAddress=12.5.7.10.0.0.1/com.gateways.yourtargets.24
TargetAddress=computingcenter.acme.com/com.disk-
vendor.diskarrays.sn.45678
```

The response to a SendTargets text command returns one or more target addresses for each target WWUI it returns. This field is used to indicate one of the known addresses of the target.

14 AccessID

AccessID=<SCSI-AccessID-value>

Deliver a SCSI AccessID to the target

15 FMarker

FMarker=<send|receive|send-receive|no>

Examples:

```
I->FMarker=send-receive
T->FMarker=send-receive
```

results in Marker being used in both directions while

```
I->FMarker=send-receive
T->FMarker=receive
```

results in Marker being used from the initiator to the target but not from the target to initiator.

16 RFMarkInt

RFMarkInt=<number-from-1-to-65535>

Indicates at what interval (in 4 byte words) the receiver wants the markers. The larger of the numbers (wanted by receiver and offered by sender) is selected. The interval is measured from the beginning of a

marker to the beginning of the next marker - e.g., a value of 1026

means 1026 words (4096 bytes of "pure" payload between markers).

Default is 2050.

17 SFMarkInt

SFMarkInt=<number-from-1-to-65535>

Indicates at what interval (in 4 byte words) the sender offers to send the markers. The larger of the numbers (wanted by receiver and offered by sender) is selected. The interval is measured from the beginning of a marker to the beginning of the next marker - e.g., a value of 1026 means 1026 words (4096 bytes of "pure" payload between markers).

Default is 2050.

18 IFMarkInt

IFMarkInt=<number-from-1-to-65535>

Indicates the initial marker-less interval required by the initiator in both directions in 4 byte words. The interval is measured from the beginning of the TCP stream to the beginning of the first marker - e.g., a value of 1024 means 1024 words (4096 bytes of "pure" payload up to the first marker).

Default is 4096.

19 UseR2T

UseR2T=<yes|no>

Examples:

```
I->UseR2T=no
T->UseR2T=no
```

The UseR2T key is used to turn off the default use of R2T, thus allowing an initiator to send data to a target without the target having sent an R2T to the initiator. The default action is that R2T is required, unless both the initiator and the target send this key-pair attribute specifying UseR2T:no. Once UseR2T has been set to 'no', it cannot be set back to 'yes'. Note than only the first

outgoing data item (either immediate data or a separate PDU) can be sent unsolicited by a R2T.

20 BidiUseR2T

BidiUseR2T=<yes|no>

Examples:

```
I->BidiUseR2T=no
T->BidiUseR2T=no
```

The BidiUseR2T key is used to turn off the default use of BiDiR2T, thus allowing an initiator to send data to a target without the target having sent an R2T to the initiator for the output data (write part) of a Bi-directional command (having both the R and the W bits set). The default action is that R2T is required, unless both the initiator and the target send this key-pair attribute specifying BidiUseR2T=no. Once BidiUseR2T has been set to 'no', it cannot be set back to 'yes'. Note than only the first outgoing data burst (immediate data or separate PDUs) can be sent unsolicited by a R2T.

21 ImmediateData

ImmediateData=<yes|no>

Initiator and target negotiate support for immediate data. Default is yes. If ImmediateData is set to yes and UseR2T is set to yes (default) then only immediate data are accepted in the first burst.

If ImmediateData is set to no and UseR2T is set to yes then the initiator MUST NOT send unsolicited data and the target MUST reject them with the corresponding response code.

22 DataPDULength

DataPDULength=<number-1-to-65535>

Initiator and target negotiate the maximum data payload supported for command or data PDUs in units of 4096 bytes. Default is 16. This parameter sets the maximum-burst-size value stored in the SCSI disconnect-reconnect mode page. The value can subsequently be retrieved with the mode sense SCSI command.

23 FirstBurstSize

FirstBurstSize=<number-from-1-to-65535>>

Initiator and target negotiate the maximum length supported for unsolicited data in units of 4096 bytes. Default is 16384 units . This parameter sets the first-burst-size value stored in the SCSI disconnect-reconnect mode page. The value can subsequently be retrieved with the mode sense SCSI command.

24 ITagLength

ITagLength=<number-from16-to-32>

Initiator and target negotiate the significant length of the initiator tag to be used. Default is 32.

25 EnableCmdRN

EnableCmdRN=<no|yes>

Default is no.

Initiator and target negotiate support for CmdRN.

If CmdRN is not supported by the target the CmdRN field is ignored. This parameter is setting the EnableCmdRN field stored in the SCSI Logical Unit Control mode page.

26 PingMaxReplyLength

PingMaxReplyLength=<number>

Initiator and target negotiate the maximum length of data contained in a ping reply. Default is DataPDULength*512. The lowest of the 2 numbers is selected.

PingMaxReplyLength cannot be larger than DataPDULength*512 and the target MUST reset PingMaxReplyLength to DataPDULength*512 whenever that becomes lower than the current PingMaxReplyLength.

27 TotalText

TotalText=<number-from-512-to-65535>

Initiator and target indicate the total text limit for any Text or Login command.

Default is DataPDULength*512.

TotalText cannot be larger than DataPDULength*512 and the target MUST reset TotalText to DataPDULength*512 whenever that becomes lower than the current TotalText.

28 KeyValueText

KeyValueText=<number-from-256-to-8192>

Initiator and target indicate the total text limit for any key=value pair including delimiter.

Default is 255.

KeyValueText MUST NOT be larger than TotalText

29 MaxOutstandingR2T

MaxOutstandingR2T=<number-from-1-to-65535>

Initiator and target negotiate the maximum number of outstanding R2Ts per task. The default is 8.

30 InDataOrder

InDataOrder=<yes|no>

No is used by iSCSI to indicate that the incoming data PDUs can be in any order (EMDP = 1) while yes is used to indicate that they have to be at continuously increasing addresses (EMDP = 0).

This sets also the Connect-Disconnect mode page EMDP bit.

The default is yes but targets MAY support no.

31 BootSession

BootSession=<no|yes>

Default is no.

BootSession MAY be set to yes by the Login Command indicating to the Target that the only purpose of this Session is boot. The target MAY

restrict the type of iSCSI requests it accepts in such a Session to Logout, NOP-out, and SCSI read commands. Accepting other commands in this type of session is vendor-dependent. A target MAY reject a boot-session.

32 The Glen-Turner vendor specific key format

X-vendor.dns.name-xxxxx=

Keys with this format will be used for vendor-specific purposes. These keys will always start with X- .

To identify the vendor it is suggested to use the DNS-name as a prefix to the key-proper.

Full Copyright Statement