iSCSI Port Provider
Design Specification

Version 0.2

June 2008

Table of Contents

1 Introduction

1.1 Overview

ISCSI Target Mode Framework (STMF) provides a modular extensible interface for implementing SCSI protocol-based services and transports. A module that implements a specific class of SCSI device is a “lun provider”. A modules that implements a type of SCSI transport is a “port provider”. Examples of STMF port providers include Fibre Channel (FC) and Serial Attached SCSI (SAS). This document describes the design for a port provider that implements iSCSI transport (iscsit).

In addition to providing iSCSI transport for STMF, iscsit will form the basis for a target-mode iSCSI Extensions for RDMA (iSER) transport. ISER imposes additional requirements on the design of iscsit – the most significant of these requirements is a modular transport capability. ISER connections are established using TCP/IP and are then switched to an RDMA-based transport. The iSCSI Data Mover (IDM) kernel module provides this modular transport capability and implements some of the functionality that would normally be part of iscsit (see the iSCSI Data Mover design spec for more details).

1.2 Current iSCSI Target

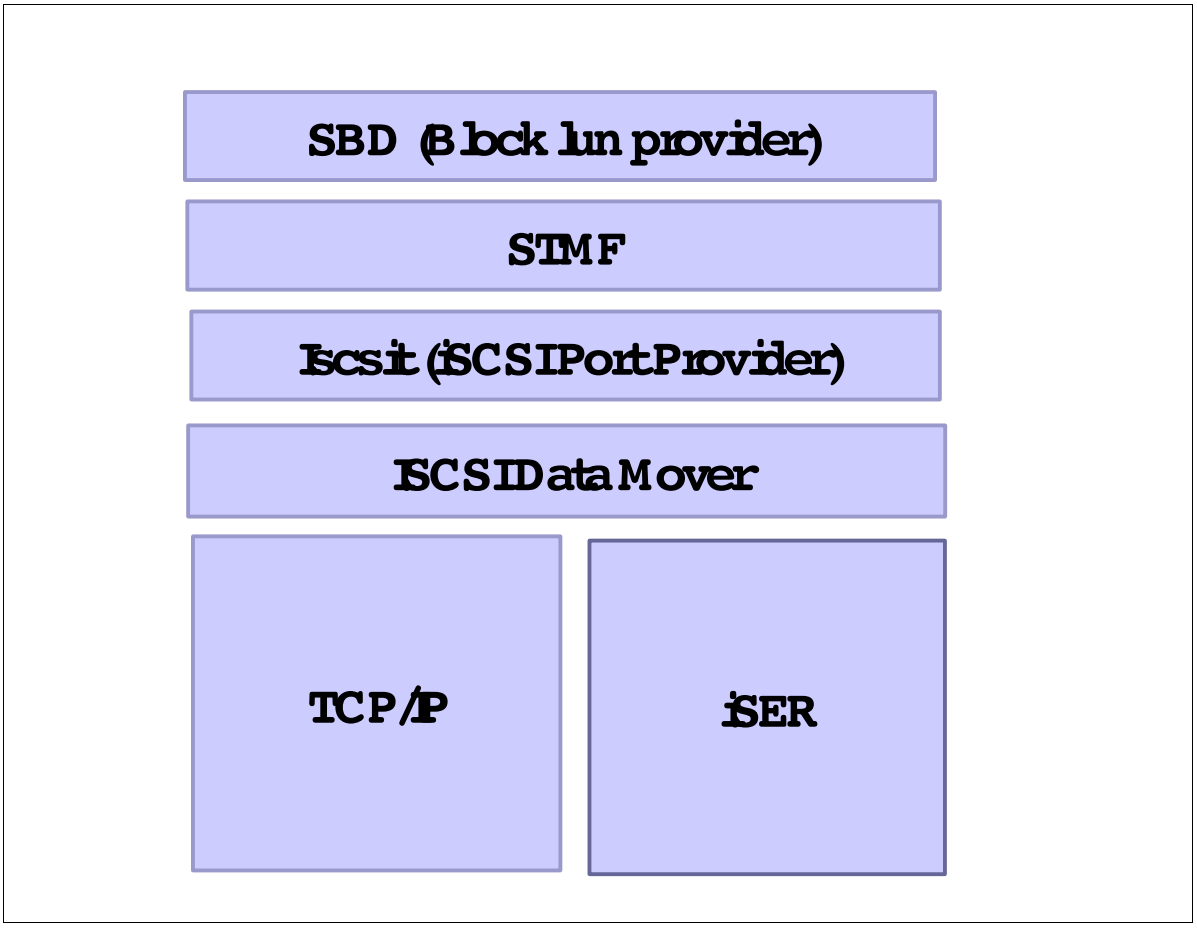
Solaris has an existing iSCSI target implementation today in the form of a user-space daemon called “iscsitgtd”. Iscsitgtd provides both the iSCSI protocol and SCSI protocol. To adapt the existing code to Comstar would require at least the following work:

- Modify the user-space code to use kernel functions including memory management, thread management, multithreaded synchronization, timers, and network access
- Remove the SCSI protocol code and interface with the STMF API
- Remove the connection code and interface with the IDM API

Since neither STMF nor IDM existed when the existing code was developed it's not surprising that the code structure doesn't lend itself to consuming the new interfaces. Consequently iscsit will not be able to directly leverage code from iscsitgtd.

2 Architecture

2.1 Block Diagram



2.2 Component Descriptions

SCSI Target Mode Framework (STMF)	See STMF Design document
SCSI Block Driver (SBD)	Block storage lun provider (See STMF design document)

ISCSI Target Port Provider (iSCSIT)	Provides the following functions: STMF port provider (see STMF design document) iSCSI login iSCSI authentication iSCSI logout iSCSI text requests (send targets) iSCSI no-op iSCSI session management iSCSI session state machine
ISCSI Data Mover (IDM)	Provides the following functions: iSCSI connection management iSCSI connection state machine iSCSI socket transport (default IDM transport) Receive and partially validate complete iSCSI PDU's using the selected IDM transport Send and partially validate complete iSCSI PDU's using the selected IDM transport Header and data digests
TCP/IP	This is the default ISCSI transport. ISCSI PDU's are sent and received using sockets over TCP/IP. ISCSI depends on the reliable delivery of TCP/IP.
iSER	Alternate iSCSI transport for RDMA capable networks (see iSER design spec for more details). Connections are established using TCP/IP and then switched to iSER transport during full-feature mode (FFM).

3 iSCSI SMF Service

The STMF framework implements the SMF service “system/stmf:default” which controls all STMF lun providers and port providers. If this service is disabled then STMF devices are not remotely accessible.

Since Iscsit listens on one or more TCP/IP ports depending on the configuration, it will implement a separate SMF service “network/iscsi/target:default”. This separate SMF service will allow administrators to easily determine whether a system is exporting iSCSI target services.

The legacy iSCSI target (/usr/sbin/iscsitgtd) service is named “svc:/system/iscsitgtd:default”. If both new new service and the old service were to be enabled at the same time the resulting behavior would be very confusing and unpredictable (likely whichever service was enabled first would receive all incoming iSCSI connections). To prevent this situation, “svc:/network/iscsi/target:default” and “svc:/system/iscsitgtd:default” will be mutually exclusive. This will require modifications to the SMF components of the current iSCSI target.

When “svc:/network/iscsi/target:default” is enabled, Iscsit will register all configured iSCSI targets with STMF. Similarly when the service is disabled, all existing connections will be cleaned up and the iSCSI targets deregistered from STMF. Consequently, iSCSI targets (local ports) will not be visible using stmfadm when the iSCSI service is disabled.

4 ISCSI Protocol Implementation

4.1 Connection Establishment

Iscsit begins listening for connections on the appropriate port by calling the following IDM services (See the IDM design specification for a complete description of these functions):

- idm_tgt_svc_create – create a new server context
- idm_tgt_svc_connect – connect the new server

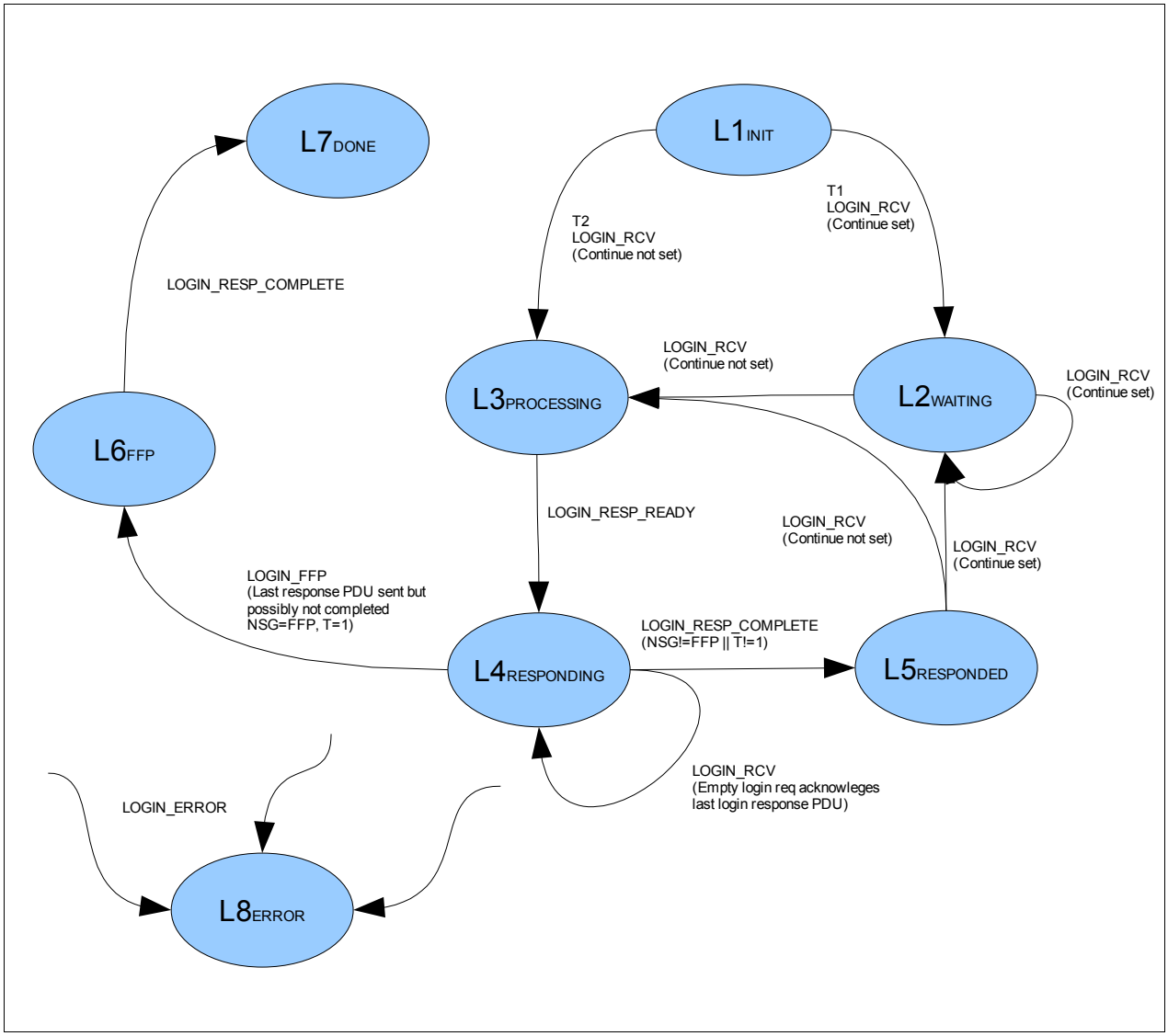
IDM handles the iSCSI connection state machine and generates notifications to iscsit for certain well defined events:

- Connection accepted
- Full Feature Mode Enabled
- Full Feature Mode Disabled
- Connection lost
- Connection destroyed

When a new connection is accepted on the listening socket, IDM will call iscsit's *client_notify* callback (iscsit_client_notify) with a handle for the new connection and a notification type of CN_CONNECT_ACCEPT. Iscsit allocates and initializes a local context representing the new connection and also initializes the login state machine (see the section on Login below). At this point iscsit is ready to receive iSCSI PDU's from the new connection.

4.2 Login

An iSCSI initiator performs a login process with an iSCSI target to authenticate itself, bind the connection to the appropriate target node and negotiate iSCSI parameters that are acceptable to both initiator and target. Although most straightforward login sequences involve a single login request and login response per login phase, the standard allows login requests and responses to be broken up over multiple PDU's. Each login PDU except the last in a sequence (request or response) has the “Continue” bit set in the BHS, and login PDU's with the “Continue” bit set must be acknowleged by an empty login PDU (RFC 3720 section 5.2). A complete login implementation that conforms to the standard requires a substantial amount of state – to address this requirement iscsit will implement a login state machine.



4.2.1 Login States

INIT	Initial state
WAITING	Received one or more login request PDU's with the C bit set and waiting for the remaining PDU's in the login sequence
PROCESSING	All PDU's in a login sequence received. Processing the key-value pairs according to the current login state
RESPONDING	Sending login response PDU(s). If the response consists of multiple PDU's then each response PDU except the last must be acknowleged by an empty LOGIN request PDU from the initiator

RESPONDED	All login response PDUs sent
FFP	The last login response PDU was submitted for transmission and the login response sequence requested a transition to full feature phase (NSG=Full feature phase, T=1)
DONE	Transmission complete for last login PDU
ERROR	An error occurred that prevented the login process from completing

4.2.2 Login Events

LOGIN_RCV	Received a login PDU
LOGIN_RESP_READY	The data payload for the login response is ready
LOGIN_FFP	Transmitted the last login response PDU of a sequence and the response requested a transition to full feature phase
LOGIN_RESP_COMPLETE	Transmit complete for login response PDU
LOGIN_ERROR	An error occurred that prevented the login process from completing

4.2.3 Keyword processing

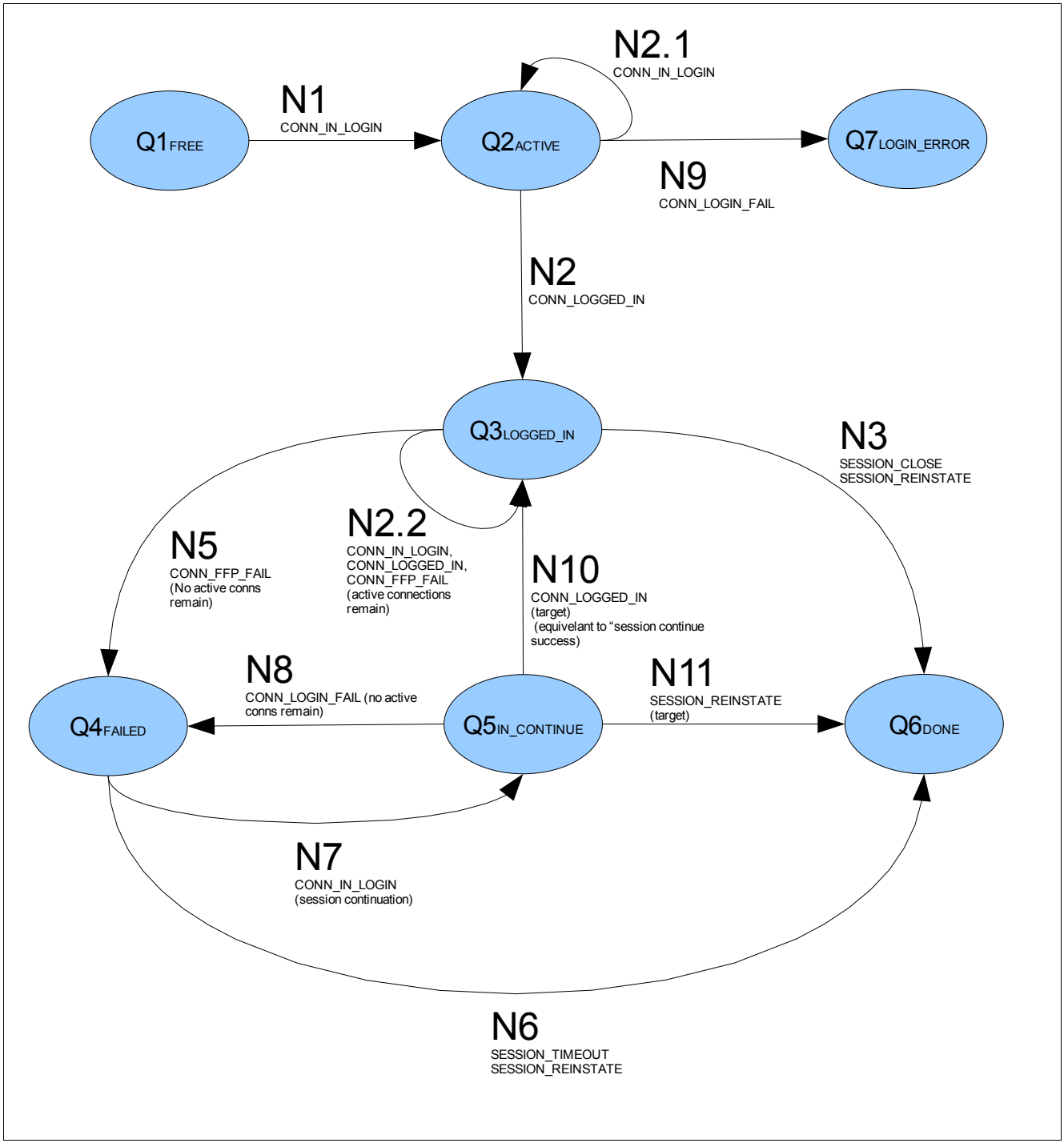
Login keyword processing takes place in the “PROCESSING” state of the login state machine. All iSCSI textbuffers will be converted to Solaris nvlist format. The login negotiation code will construct a nvlist containing the login response key-values and the response text buffer will be constructed from that login response nvlist. As the values from the login request are processed any values that are considered to have completed negotiation (declarative values or response values selected from available request values) are added to a negotiated values nvlist. All three nvlists are maintained within iscsit_conn_t (iscsit's connection context). The following pseudo-code illustrates the high-level algorithm for processing login keywords

```
login_handle_login_request(login_pdu_list)
{
       /* Decode text buffer PDU list into nvlist format */
       Login_request_nvlist = login_pdu_list_to_nvlist(login_pdu_list);

       /* Process request nvlist, building an nvlist for the login response */
       Login_response_nvlist = login_process_request_nvlist(Login_request_nvlist);

       /*
        * Based on the request, add any agreed upon values to the negotiated
        * values nvlist
        */
       Negiotated_nvlist = login_sm_negotiated_values(login_request_nvlist,
```

```
                    login_response_nvlist);

            /* Encode response nvlist into a text buffer PDU list */
            login_send_response(Login_response_nvlist)
    }
```

4.3 Session management

Iscsit will implement the RFC3720 session state machine with addition of two terminal states. “Done” represents a session that has completed normally and is otherwise equivalent to “Free”. If the leading login for a session fails then the session will go to “Login_Error” state.


N2.1
CONN_IN_LOGIN
N1
CONN_IN_LOGIN
Q1FREE
Q2ACTIVE
Q7LOGIN_ERROR
N9
CONN_LOGIN_FAIL
N2
CONN_LOGGED_IN
Q3LOGGED_IN
N3
SESSION_CLOSE
SESSION_REINSTATE
N5
CONN_FFP_FAIL
(No active conns remain)
N2.2
CONN_IN_LOGIN,
CONN_LOGGED_IN,
CONN_FFP_FAIL
(active connections remain)
N10
CONN_LOGGED_IN
(target)
(equivelant to "session continue success)
N8
CONN_LOGIN_FAIL (no active conns remain)
N11
SESSION_REINSTATE
(target)
Q4FAILED
Q5IN_CONTINUE
Q6DONE
N7
CONN_IN_LOGIN
(session continuation)
N6
SESSION_TIMEOUT
SESSION_REINSTATE

4.3.1 Session States

FREE	Initial session state
ACTIVE	Leading login connection is in “IN_LOGIN” state
LOGGED_IN	Leading login connection is in “LOGGED_IN” state
FAILED	No remaining connections in “LOGGED_IN” state
IN_CONTINUE	A new connection has been established with the same ISID as this session (if TSIH for the new connection matches the existing session then the initiator is attempting session continuation and if TSIH is zero then the initiator is attempting session reinstatement)
DONE	Terminal state, session ended normally
LOGIN_ERROR	Leading login failed to reach “LOGGED_IN” state

4.3.2 Session Events

CONN_IN_LOGIN	Connection transitioned to “IN_LOGIN” state after receiving an initial login request
CONN_LOGGED_IN	Connection transitioned to “LOGGED_IN” state after a successful login exchange
CONN_LOGIN_FAIL	Connection login failed and connection transitioned to “INIT ERROR” state
CONN_FFP_FAIL	A connection in “Full Feature Phase” failed
SESSION_CLOSE	“Session close” logout
SESSION_REINSTATE	Session was successfully reinstated
SESSION_FAIL	Session failed (See RFC3720 5.3.6)
SESSION_TIMEOUT	Session recovery timer expired
SESSION_CONTINUE	Connection reinstatement or new connection for failed session
SESSION_CONTINUE_FAIL	Connection reinstatement failed or new connection failed

4.4 Logout

IDM notices the logout PDU when it is received and generates the appropriate connection state machine transition before delivering the PDU to iscsit. Iscsit is primarily responsible for generating and sending the correct logout reponse to the PDU. The associated task and session cleanup will be driven by event notifications from the IDM connection state machine.

4.5 SCSI Command dispatch

IDM validates iSCSI command requests against the current connection state before forwarding

the PDU to Iscsit. Consequently Iscsit can assume the connection is still in “full feature phase” at the time it is called with a SCSI command. Iscsit decodes the SCSI request, allocates a task from STMF and dispatches the request to STMF for processing. To process the task, STMF may call back into Iscsit to allocate data buffers and transfer data as necessary. Once the operation associated with the task is complete including all data transfers STMF calls Iscsit to send the task status.

5 iSCSI Management Library

5.1 Overview

Iscsit is implemented as an STMF port provider. The STMF framework allows the association of an nvlist containing persistent configuration data to a port provider instance using libstmf. This configuration is then handed to the port provider kernel implementation at configuration time.

Libiscsit provides a number of services to facilitate the encoding of iSCSI configuration data to and from nvlist format. The task flow for modifying the iscsit configuration is as follows:

1. Retrieve the iscsit configuration by calling it_config_load(). it_config_t is a C representation of iscsit's STMF port provider configuration.
2. Modify the configuration by manipulating the data in the it_config_t structure. Targets, target portal groups and initiator contexts can be added or remove by using the associated create and delete functions (e.g. it_target_create and it_target_delete for targets).
3. Call it_config_commit to inform the iscsit service that the configuration has changed and commit the new configuration.

5.2 Global Configuration Services

5.2.1 it_config_t

NAME

it_config_t – Iscsit Port Provider configuration

SYNOPSIS

#include <libiscsit.h>

DESCRIPTION

This structure represents a complete configuration for the iscsit port provider. In addition to the global configuration it_config_t includes lists of child objects including targets, target portal groups and initiator contexts. Each object includes a “generation” value which is used by the iscsit kernel driver to identify changes from one configuration update to the next.

STRUCTURE MEMBERS

```
int             *config_version;
it_tgt_t        *config_tgt_list;
```

```
int             config_tgt_count;
it_tpg_t        *config_tpg_list;
int             config_tpg_count;
it_ini_t        *config_ini_list;
int             config_ini_count;
nvlist_t        *config_global_properties;
```

config_version	Version number for this configuration structure.
config_tgt_list	Linked list of target context representing the currently defined targets. Applications can add targets to or remove targets from this list using the it_tgt_create and it_target_delete functions.
config_tgt_count	The number of currently defined targets
config_tpg_list	Linked list of target portal group contexts. Applications can add target portal groups to or remove target portal groups from this list using the it_tpg_create and it_tpg_delete_functions.
config_tpg_count	The number of currently defined target portal groups
config_ini_list	Linked list of initiator contexts. Applications can add inititator contexts to or remove initiator contexts from this list using the it_ini_create and it_ini_delete functions.
config_ini_count	The number of currently defined initiator contexts
config_global_properties	Nvlist representing the current global property settings. This list can be manipulated using libnvpair(3lib).

SEE ALSO

it_tgt_t, it_tpg_t, it_ini_t, it_config_load, it_config_commit, it_config_free, it_tgt_create, it_tgt_delete, it_tpg_create, it_tpg_delete, it_ini_create, it_ini_delete

5.2.2 it_config_load, it_config_commit, it_config_free

NAME

it_config_load, it_config_commit, it_config_free

SYNOPSIS

```
#include <libiscsit.h>

int         it_config_load(it_config_t **cfg);
int         it_config_commit(it_config_t *cfg);
void        it_config_free(it_config_t *cfg);
```

PARAMETERS

`cfg`	A C representation of the current iSCSI configuration.

DESCRIPTION

it_config_load

Allocate and create an it_config_t structure representing the current iSCSI configuration. This structure is compiled using the "provider" data returned by stmfGetProviderData. If there is no provider data associated with iscsit, the it_config_t structure will be set to a default configuration.

it_config_commit

Informs the iscsit service that the configuration has changed and commits the new configuration to the persistent store by calling stmfSetProviderData. This function can be called multiple times during a configuration sequence if necessary.

it_config_free

Free any resources associated with the it_config_t structure.

RETURN_VALUES

0	Success
ENOMEM	Couldn't get resources
EINVAL	Invalid it_config_t structure

SEE ALSO

it_config_t

5.3 Target Services

5.3.1 it_tgt_t

NAME

it_tgt_t – Iscsit Target Node

SYNOPSIS

#include <libiscsit.h>

DESCRIPTION

An iSCSI target node is represented by an it_tgt_t structure. Each target node includes a list

of associated target portal group tags and a list of properties.

STRUCTURE MEMBERS

```
char            tgt_name[MAX_ISCSI_NODENAMELEN];
it_tpgt_t       *tgt_tpgt_list;
int             tgt_tpgt_count
nvlist_t        *tgt_properties;
it_tgt_t        *tgt_next;
```

tgt_name	The iSCSI target node name in either IQN or EUI format (see RFC 3720).
tgt_tpgt_list	A linked list representing the current target portal group tags associated with this target.
tgt_tpgt_count	The number of currently defined target portal group tags
tgt_properties	An nvlist representation of the properties associated with this target
tgt_next	Next target in the list of targets. If tgt_next is NULL then this is the last target in the list.

SEE ALSO

it_tpgt_t, it_tpg_t, it_tgt_create, it_tgt_delete, it_tpgt_create, it_tpgt_delete

5.3.2 it_tpgt_t

NAME

it_tpgt_t – Iscsit Target Portal Group Tag

SYNOPSIS

#include <libiscsit.h>

DESCRIPTION

A target portal group tag is a binding between a target, and target portal group along with a numerical value associated with that binding. The numerical identifier is used as the “target portal group tag” defined in RFC3720.

STRUCTURE MEMBERS

```
uint16_t        tpgt_tag;
char            tpgt_tpg_name[MAX_TPG_NAMELEN]
it_tpgt_t       *tpgt_next;
```

tpgt_tag	A numerical identifier that uniquely identifies a target portal group within the associated target node.
tpgt_tpg_name	The name of the target portal group associated with this target portal group tag
tpgt_next	Next target portal group tag in the list of target portal group tags. If tpgt_next is NULL then this is the last target portal group tag in the list.

SEE ALSO

it_tgt_t, it_tpg_t, it_tpgt_create, it_tpgt_delete

5.3.3 it_tgt_create, it_tgt_delete

NAME

it_tgt_create, it_tgt_delete

SYNOPSIS

```
#include <libiscsit.h>

int         it_tgt_create(it_config_t *cfg, it_tgt_t **tgt,
                          char *tgt_name);
void        it_tgt_delete(it_config_t *cfg, it_tgt_t *tgt);
```

PARAMETERS

`cfg`	The current iSCSI configuration obtained from it_config_load()
`tgt`	Pointer to an iSCSI target structure
`tgt_name`	The target node name for the target to be created. The name must be in either IQN or EUI format. If this value is NULL then a node name will be generated automatically in IQN format.

DESCRIPTION

it_tgt_create

Allocate and create an it_tgt_t structure representing a new iscsi target node. If tgt_name is NULL then a unique target node name will be generated automatically, otherwise the value of tgt_name will be used as the target node name. The new it_tgt_t structure is added to the target list (cfg_tgt_list) in the configuration structure and the new target will not be instantiated until the modified configuration is committed by calling it_config_commit().

it_tgt_delete

Delete target represented by “tgt” where “tgt” is an existing it_tgt_t structure within the configuration “cfg”. The target removal will not take effect until the modified configuration is committed by calling it_config_commit().

RETURN_VALUES

0	Success
ENOMEM	Couldn't get resources
EINVAL	Invalid parameter

SEE ALSO

it_tgt_t, it_tpgt_create, it_tpgt_delete

5.3.4 it_tpgt_create, it_tpgt_delete

NAME

it_tpgt_create, it_tpgt_delete

SYNOPSIS

```
#include <libiscsit.h>

int         it_tpgt_create(it_config_t *cfg, it_tgt_t *tgt,
            it_tpgt_t **tpgt, char *tpg_name, uint16_t tpgt_tag);
void        it_tpgt_delete(it_config_t *cfg, it_tgt_t *tgt,
            it_tpgt_t *tpgt);
```

PARAMETERS

`cfg`	The current iSCSI configuration obtained from it_config_load()
`tgt`	Pointer to the iSCSI target structure associated with the target portal group tag
`tpgt`	Pointer to a target portal ground tag structure
`tpg_name`	The name of the TPG to be associated with this TPGT.
`tpgt_tag`	16 bit Numerical identifier for this TPGT. The value must be unique within the associated target node.

DESCRIPTION

it_tpgt_create

Allocate and create an it_tpgt_t structure representing a new iscsi target portal group tag. The new it_tpgt_t structure is added to the target tpgt list (tgt_tpgt_list) in the it_tgt_t and the new target portal

group tag will not be instantiated until the modified configuration is committed by calling it_config_commit().

it_tpgt_delete

Delete target portal group tag represented by “tpgt” where “tpgt” is an existing it_tpgt_t structure within the target “tgt”. The target portal group tag removal will not take effect until the modified configuration is committed by calling it_config_commit().

RETURN_VALUES

0	Success
ENOMEM	Couldn't get resources
EINVAL	Invalid parameter

SEE ALSO

it_tgt_t, it_tpg_t, it_tpg_create, it_tpg_delete

5.4 Target Portal Group Services

5.4.1 it_tpg_t

NAME

it_tpg_t – Iscsi Target Portal Group

SYNOPSIS

#include <libiscsit.h>

DESCRIPTION

A portal is an IP address and TCP port and a portal group is a set of portals. Each defined portal belows to exactly one portal group. Applications can associated a target portal group with a particular target using a target portal group tag. Initiators can only connect to targets through the portals associated with the target's target portal group tags.

STRUCTURE MEMBERS

```
char            tpg_name[MAX_TPG_NAMELEN];
it_portal_t     *tpg_portal_list;
int             tpg_portal_count;
it_tpg_t        *tpg_next;
```

tpg_name	Identifier for the target portal group.
tpg_portal_list	Linked list of it_portal_t structures
tpg_portal_count	Number of it_portal_t structures in the list
tpg_next	Next target portal group in the list of target portal groups. If tpg_next is NULL then this is the last target portal group in the list.

SEE ALSO

it_portal_t, it_tpg_create, it_tpg_delete, it_portal_create, it_portal_delete

5.4.2 it_tpg_create, it_tpg_delete

NAME

it_tpg_create, it_tpg_delete

SYNOPSIS

```
#include <libiscsit.h>

int         it_tpg_create(it_config_t *cfg, it_tpg_t **tpg,
            char *tpg_name, char *portal_ip_port);
void        it_tpg_delete(it_config_t *cfg, it_tpg_t *tpg);
```

PARAMETERS

`cfg`	The current iSCSI configuration obtained from it_config_load()
`tpg`	Pointer to the it_tpg_t structure representing the target portal group
`tpg_name`	Identifier for the target portal group
`portal_ip_port`	A string containing an appropriately formatted IP address or IP address:port. Both IPv4 and IPv6 addresses are permitted. This value becomes the first portal in the TPG – applications can add additional values using it_portal_create before committing the TPG.

DESCRIPTION

it_tpg_create

Allocate and create an it_tpg_t structure representing a new iscsi target portal group. The new it_tpg_t structure is added to the global tpg list (cfg_tgt_list) in the it_config_t structure and the new target portal group will not be instantiated until the modified configuration is committed by calling it_config_commit().

it_tpg_delete

Delete target portal group represented by “tpg” where “tpg” is an existing it_tpg_t structure within the global configuration “cfg”. The target portal group removal will not take effect until the modified configuration is committed by calling it_config_commit().

RETURN_VALUES

0	Success
ENOMEM	Couldn't get resources
EINVAL	Invalid parameter

SEE ALSO

it_portal_t, it_portal_create, it_portal_delete

5.5 Portal Services

5.5.1 it_portal_t

NAME

it_portal_t – Iscsit Portal

SYNOPSIS

#include <libiscsit.h>

DESCRIPTION

A target portal is represented by an IP address and a listening TCP port

STRUCTURE MEMBERS

```
struct sockaddr_storage   portal_addr;
it_portal_t               *portal_next;
```

portal_addr	Sockaddr_storage structure representing the IPv4 or IPv6 address and TCP port associated with the portal
portal_next	Next portal in the list of portals. If portal_next is NULL then this is the last portal in the list.

SEE ALSO

it_portal_create, it_portal_delete

5.5.2 it_portal_create, it_portal_delete

NAME

it_portal_create, it_portal_delete

SYNOPSIS

```
#include <libiscsit.h>

int         it_portal_create(it_config_t *cfg, it_tpg_t *tpg,
            it_portal_t **portal, char *portal_ip_port);
void        it_portal_delete(it_config_t *cfg, it_tpg *tpg,
            it_portal_t *portal);
```

PARAMETERS

`cfg`	The current iSCSI configuration obtained from it_config_load()
`tpg`	Pointer to the it_tpg_t structure representing the target portal group
`portal`	Pointer to the it_portal_t structure representing the portal
`portal_ip_port`	A string containing an appropriately formatted IP address or IP address:port. Both IPv4 and IPv6 addresses are permitted.

DESCRIPTION

it_portal_create

Add an it_portal_t structure presenting a new portal to the specified target portal group. The change to the target portal group will not take effect until the modified configuration is committed by calling it_config_commit().

it_portal_delete

Remove the specified portal from the specified target portal group. The portal removal will not take effect until the modified configuration is committed by calling it_config_commit().

RETURN_VALUES

0	Success
ENOMEM	Couldn't get resources
EINVAL	Invalid parameter

SEE ALSO

it_portal_t, it_portal_create, it_portal_delete

5.6 Initiator Context Services

5.6.1 it_ini_t

NAME

it_ini_t – Iscsit Initiator Context

SYNOPSIS

#include <libiscsit.h>

DESCRIPTION

A context representing a remote ISCSI initiator node. The purpose of this structure is to maintain information specific to a remote initiator such as the CHAP username and CHAP secret.

STRUCTURE MEMBERS

```
char        ini_name[MAX_ISCSI_NODENAMELEN];
nvlist_t    *ini_properties
```

ini_name	The iSCSI node name of the remote initiator
ini_properties	Properties associated with the initiator context

SEE ALSO

it_ini_create, it_ini_delete

5.6.2 it_ini_create, it_ini_delete

NAME

it_ini_create, it_ini_delete

SYNOPSIS

```
#include <libiscsit.h>

int         it_ini_create(it_config_t *cfg, it_ini_t **ini,
            char *ini_node_name);
void        it_ini_delete(it_config_t *cfg, it_ini_t *ini);
```

PARAMETERS

`cfg`	The current iSCSI configuration obtained from it_config_load()
`ini`	Pointer to the it_ini_t structure representing the initiator context
`ini_node_name`	The iSCSI node name of the remote initiator

DESCRIPTION

it_ini_create

Add an initiator context to the global configuration. The new initiator context will not be instantiated until the modified configuration is committed by calling it_config_commit().

it_ini_delete

Remove the specified initiator context from the global configuration. The removal will not take effect until the modified configuration is committed by calling it_config_commit().

RETURN_VALUES

0	Success
ENOMEM	Couldn't get resources
EINVAL	Invalid parameter

SEE ALSO

it_portal_t, it_portal_create, it_portal_delete

5.7 Properties

Name	Applicable to	Format	Description
targetchapuser	it_tgt_t	String	Specifies the CHAP username for a target for use in mutual CHAP authentication. This value is only used when the initiator node is configured to use mutual CHAP authentication. If the property is not defined then the target node name is used as the username.
targetchapsecret	it_tgt_t	Base64 encoded string	The CHAP secret to send during mutual CHAP authentication.
alias	it_config_t, it_tgt_t	String	An alternate identifier associated with a

			target node. Targets can have individual unique aliases or they can inherit the global setting of this property – the system hostname is an example of a possible global alias that could be applied to all targets as a default alias.
auth	it_config_t, it_tgt_t	String	Specifies the authentication method to use for the target. Valid values are “radius”, “chap” and “none”. “chap” indicates that initiators connecting to this target must be authenticated using Challenge Handshake Authentication Protocol (CHAP). “radius” indicates initiators should also be authenticated via CHAP but the required authentication parameters should be obtained from a central RADIUS server (see “radiusserver” and “radiussecret” properties). “none” means that no authentication is required to connect to the target. Targets inherit the global setting of this property.
isns	it_config_t	Boolean	Specifies whether or not targets should be registered with the set of defined iSCSI Name Service (iSNS) servers. Targets inherit the global setting of this property
isnsserver	it_config_t	String	Defines a list of iSNS servers with which iSCSI target nodes will be registered with the “isns” property associated with the respective target is set. The format of the string is: ip_address[:port],....
radiusserver	it_config_t	String	Defines the RADIUS server to use for RADIUS-based CHAP authentication. The format of the string is: ip_address[:port]
radiussecret	it_config_t	Base64 encoded string	RADIUS Shared Secret for centralized CHAP authentication. This property is only

			used if the “auth” property for a target node is set to “radius”.
chapuser	it_ini_t	String	Specifies the CHAP username for an initiator for use in CHAP authentication. If the property is not defined then the initator node name is used as the username.
chapsecret	it_ini_t	Base64 encoded string	The CHAP secret expected from the initiator during CHAP authentication.

6 ISCSI Kernel Configuration

Iscsit receives its initial configuration from STMF via the callback it registers along with its other port provider information. Libiscsit will provide subsequent configuration updates using an ioctl with the new configuration in packed nvlist format. Iscsit will compare this updated nvlist with the current configuration and incorporate any changes. Such changes fall into three categories:

1. If an object (target, target portal group, target portal group tag or initiator context) is in the new configuration that is not in the current configuration that object will be added to the current configuration
2. If an object in the current configuration is no longer in the new configuration that object will be removed from the current configuration
3. If an object has been modified according to its generation value (see below) the existing object will be updated appropriately

6.1 Detecting modified configuration objects

To identify which objects in the configuration have changed each object includes a private "generation" value. Iscsit can compare the generation value of an object in the new configuration with the corresponding object in the current configuration. If the new generation value is higher then the object has been modified and the current object should be updated with the new configuration. Any change to the configuration including object addition or removal is reflected by a change in the global generation value.

6.2 Ioctls

Libiscsit will communicate with the Iscsit kernel code using a set of ioctls. These ioctls are project private and are intended only for use by Libiscsit.

6.2.1 ISCSIT_IOC_SET_CONFIG

NAME

ISCSIT_IOC_SET_CONFIG - Set updated iSCSI Port Provider Configuration

SYNOPSIS

#include <sys/iscsit_ioctl.h>

DESCRIPTION

Libiscsit call this ioctl after a configuration change to provide the new configuration to the iSCSI target

kernel code. Each configuration has a global generation value associated with it. Iscsit can compare the new generation value to its current generation value to ensure that no intervening configuration updates have taken place. If the new generation value is less than or equal to the current generation value then the ioctl will return EFAULT, indicating the application should update its version of the configuration and redo the operation.

INPUT STRUCTURE

```
typedef struct {
        int     set_cfg_vers;
        char    *set_cfg_pnvlist;
        int     set_cfg_pnvlist_len;
} iscsit_ioc_set_config_t
```

set_cfg_vers	Version number specifying the format of the structure.
set_cfg_pnvlist	Pointer to a buffer containing a packed nvlist representing the new configuration
set_cfg_pnvlist_len	Length of the nvlist buffer

RETURN VALUES

Upon successful completion the value returned is 0. If the “new” configuration is older than the current configuration (new generation value <= current generation value) then EFAULT is returned. If the nvlist is somehow invalid such that it cannot be translated into a valid configuration then EINVAL is returned

6.2.2 ISCSIT_IOC_GET_STATE

NAME

ISCSIT_IOC_GET_STATE – Get iSCSI Runtime State Information

SYNOPSIS

```
#include <sys/iscsit_ioctl.h>
```

DESCRIPTION

Libiscsit calls this ioctl to obtain information about the current operational state including iSCSI session information and connection information.

INPUT STRUCTURE

```
typedef struct {
```

```
        int     getst_vers;
        char    *getst_pnvlist;
        int     getst_pnvlist_len;
} iscsit_ioc_getstate_t
```

getst_vers	Version number specifying the format of the structure.
getst_pnvlist	Pointer to a destination buffer into which to copy the state information
getst_pnvlist_len	Length of the nvlist buffer. If the buffer is not large enough to hold the state information then this value will be updated with the required size and E2BIG will be returned.

RETURN VALUES

Upon successful completion the value returned is 0. If the packed nvlist containing the state information is larger than the buffer size provided then E2BIG is returned.