libstmf

Library interface specification for the SCSI Target Mode Framework (STMF)

Document Revision: 0.8

Table of Contents

1 Introduction

libstmf provides a C programming interface to the SCSI Target Mode Framework allowing clients to manage the provisioning of logical units and targets to the initiator clients of the framework. Please refer to <SCSI Target Mode Framework Specification Document> for more information on the SCSI Target Mode Framework.

1.1 Terms

The terms that are used in this specification are defined in this section.

Term	Definition
Target Port	The system component that receives SCSI commands
Logical Unit	The system component that executes SCSI commands
Logical Unit Number	The SCSI identifier of a logical unit within a target
Logical Unit Access List	A list of initiator ports that have access rights for a specific logical unit
Map/Unmap	Process of assigning a logical unit number for a specified host group
Host Group	A list of initiator ports that can be used to apply access control to a logical unit
Target Group	A set of targets that can be used to provide availability for a logical unit
View	A list of logical units exposed to a list of initiator ports through a list of targets
View Entry	A single entry of a view comprised of a target group, an host group and a logical unit number for a given logical unit.

1.2 Library Objects

1.2.1 Logical Unit

A Logical Unit object is provided to the SCSI Target Mode Framework for the purposes of executing SCSI commands. Library clients can manage a Logical Unit object's accessibility to one or more SCSI initiator clients. Library clients cannot add or remove Logical Unit objects from the system. Every Logical Unit object within the SCSI Target Mode Framework is owned by a logical unit provider whose identity is available via the properties on the Logical Unit object.

1.2.2 Target port

A Target port object is provided to the SCSI Target Mode Framework for the purposes of receiving SCSI commands on a particular logical unit. Library clients can manage a Logical Unit object's availability to one or more Target port objects. Library clients cannot add or remove Target objects from the system. Every Target port object within the SCSI Target Mode Framework is owned by a Local Port provider whose identity is available via the properties on the Target port object.

1.2.3 View Entry

A View Entry object defines the association of an host group, a target group and a logical unit number with a specified logical unit. When a view entry is created for a logical unit, a caller can assign all targets and/or all initiator ports to the logical unit thus making the logical unit accessible to all target ports and/or all initiator ports. A logical unit may have one or more view entries associated with it. Any two view entries are considered to be in conflict when an attempt is made to duplicate the association of any given initiator port, target port and logical unit. Attempting this will result in an error returned from the call to stmfAddViewEntry.

1.2.4 Target Port Group

A Target Port Group is a set of one or more Target ports that are combined together for the purposes of applying availability to a Logical Unit object. A Target Port Group may be applied to any given Logical Unit via a view entry. Target ports may not be a member in more than one Target Port Group. Target Port Group names are unique within the framework.

1.2.5 Initiator Port Group

An Initiator Group is a set of one or more initiator ports that are combined together for the purposes of applying access controls to a Logical Unit object and assigning a logical unit number to the Logical Unit. The assigned logical unit number will be reported to the members of that Initiator Port Group via the SCSI REPORT LUN command. Initiator Port Groups may contain initiator ports that are not visible to the SCSI Target Mode Framework.

Initiator ports may not be a member in more than one group. An Initiator Port Groups may be associated with a given Logical Unit via a view entry. Initiator Port Group names are unique within the framework.

2 Constants and Types

The following section defines the constants and types used by this API. These are available in /usr/include/libstmf.h.

2.1 stmf status

Macros

STMF_SUCCESS(return_value)

A macro that returns B_TRUE when the API call was successful and B_FALSE if there was an error.

STMF_ERROR(return_value)

A macro that returns B_TRUE when the API call returned an error and B_FALSE if the API call was successful.

Successful Return Status

STMF_STATUS_SUCCESS

The API call was successful

Error Return Status

STMF_ERROR_BUSY

The API call was not successful because a resource required to complete the operation was busy.

STMF_ERROR_OBJECT_NOT_FOUND

The OID used as the input argument to the API call was not found in the system.

STMF_ERROR_OBJECT_NOT_VALID

The OID used as the input argument to the API call was of an unknown object type or the object type is not valid for this API call.

STMF_ERROR_PERM

The caller did not have the correct permissions for this API call to succeed.

STMF_ERROR_NOMEM

The library was unable to allocate sufficient memory to complete the API call.

STMF_ERROR_BUSY

A resource required to complete the API call was busy.

STMF_ERROR_GROUP_NOT_FOUND

The specified group name was not found.

STMF_ERROR_MEMBER_NOT_FOUND

The specified group member was not found.

STMF_ERROR_INVALID_ARG

A specified argument was not valid.

STMF_ERROR_VIEW_ENTRY_CONFLICT

A conflict would be caused by this API call.

STMF_ERROR_EXISTS

The specified entry already exists in the system.

2.2 stmfProtocol

Format

```
typedef enum _stmfProtocol
{
        STMF_PROTOCOL_FIBRE_CHANNEL =   0,
        STMF_PROTOCOL_ISCSI =           1
} stmfProtocol;
```

Fields

STMF_PROTOCOL_FIBRE_CHANNEL

Protocol type for fibre channel whose value is 0.

STMF_PROTOCOL_ISCSI

Protocol type for iSCSI whose value is 1.

2.3 stmfDevid

Format

```
typedef struct _stmfDevid
{
        uint8_t identLength;
        uint8_t ident[256];
} stmfDevid;
```

Fields

identLength

Set to the length of the identifier in the identifier field.

ident

The device identifier for this device. STMF supports an identifier field according to the SCSI name string identifier format specified in ANSI INCITS 408-2005 (SPC-3).

Notes

This structure is used to represent a target port or an initiator port. APIs are available that assist in converting a fibre channel/SAS world-wide name and iSCSI name to an stmfDevid structure. Use stmfDevidFromWWN() to convert a fibre channel /SAS world-wide name to an stmfDevid structure. Use stmfDevidFromIscsiName() to convert an iSCSI name to an stmfDevid structure.

2.4 stmfGroupName

Format

```
typedef wchar_t stmfGroupName[256];
```

Fields

stmfGroupName

This type is used to represent the name of an initiator or target port group. The name shall be null terminated.

2.5 stmfGroupNameList

Format

```
typedef struct _stmfGroupList
{
        uint32_t         cnt;
        stmfGroupName    name[1];
} stmfGroupList;
```

Fields

cnt

The count of group names in the groupName array.

name

This is an array of one or more group names.

2.6 stmfViewEntry

Format

```
typedef struct _stmfViewEntry
{
        boolean_t       veIndexValid;
        uint32_t        veIndex;
        boolean_t       allHosts;
        stmfGroupName   hostGroup;
        boolean_t       allTargets;
        stmfGroupName   targetGroup;
        boolean_t       luNbrValid;
        uint8_t         luNbr[8];
} stmfViewEntry;
```

Fields

veIndexValid

A boolean indicating if the field veIndex contains a valid view entry index. This value must be set to B_FALSE on input when adding a view entry to a logical unit using stmfAddViewEntry.

veIndex

An unsigned 32-bit integer which specifies the view entry index. This index, along with the logical unit identifier, can be used to uniquely refer to a view entry.

allHosts

A boolean indicating whether this view entry encompasses all initiator ports or a single group of initiator ports that login to the framework. When set to B_TRUE, the value in hostGroup will be ignored.

hostGroup

The host group name to be used for this view entry.

allTargets

A boolean indicating whether this view entry encompasses all target ports or a single group of target ports within the framework. When set to B_TRUE, the value in targetGroup will be ignored.

targetGroup

The target port group name to be used for this view entry.

luNbrValid

A boolean indicating if the field luNbr contains a valid logical unit number.

luNbr

An array that represents the 8-byte logical unit number structure defined by the SCSI Architecture Model (ANSI 402-2005 SAM-3).

2.7 stmfGroupProperties

Format

```
typedef struct _stmfGroupProperties
{
        uint32_t         cnt;
        stmfDevid        name[1];
} stmfGroupProperties;
```

Fields

cnt

The count of group members in the groupMember array.

name

This is an array of 0 devids representing the group members.

2.8 stmfTargetProperties

Format

```
typedef struct _stmfTargetProperties
{
        stmfDevid        devid;
        stmfProtocol     protocol;
        uint16_t         status;
        stmfProviderName providerName;
} stmfTargetProperties;
```

Fields

devid

The device identifier for this target port.

protocol

The protocol type for this target port.

status

A value indicating the status of this target port. Can be one of the following values:

STMF_TARGET_PORT_OFFLINE

STMF_TARGET_PORT_ONLINE

providerName

The name of the owning provider for this target.

2.9 stmfLogicalUnitProperties

Format

```
typedef struct _stmfLogicalUnitProperties
{
        char             alias[256];
        uchar_t          vendor[8];
        uchar_t          product[16];
        uchar_t          revision[4];
        uint32_t         status;
        uchar_t          luid[16];
        stmfProviderName providerName[256];
} stmfLogicalUnitProperties;
```

Fields

alias

A user defined name for this logical unit.

vendor

The vendor name as defined by the SCSI standard inquiry page.

product

The product name as defined by the SCSI standard inquiry page.

revision

The revision data as defined by the SCSI standard inquiry page.

status

A value indicating the status of the logical unit. Can be one of the following values:

STMF_LOGICAL_UNIT_OFFLINE

STMF_LOGICAL_UNIT_ONLINE

luid

The 16-byte logical unit identifier for this logical unit. The luid is a value represented by the SCSI VPD page 0x83 NAA Registered Extended identifier format.

providerName

The name of the owning provider for this logical unit

3 APIs

This section defines the API calls for libstmf.

3.1 stmfAddToHostGroup

Description

Adds an initiator port to an existing host group.

Prototype

```
int stmfAddToHostGroup(
          stmfGroupName  hostGroupName,
          stmfDevid      initiatorName
);
```

Parameters

hostGroupName

The name of the host group to which the specified initiatorName is added.

initiatorName

The device identifier of the initiator port to add to the specified host group.

Return Values

STMF_STATUS_SUCCESS

The API call was successful.

STMF_ERROR_GROUP_NOT_FOUND

The specified hostGroupName was not found in the system.

3.2 stmfAddToTargetGroup

Description

Add a target to an existing target group.

Prototype

```
int stmfAddToTargetGroup(
          stmfGroupName  targetGroupName,
          stmfDevid      targetName
);
```

Parameters

targetGroupName

The name of the target port group to which the specified targetName is added.

targetName

The device identifier of the target port to add to the specified target group.

Return Values

STMF_STATUS_SUCCESS

The API call was successful.

STMF_ERROR_GROUP_NOT_FOUND

The specified targetGroupName was not found in the system.

3.3 stmfAddViewEntry

Description

Adds a view entry for a given logical unit.

Prototype

```
int stmfAddViewEntry(
          stmfGuid           logicalUnit,
          stmfViewEntry      *viewEntry
);
```

Parameters

logicalUnit

The identifier of the logical unit to which this view entry is being added.

viewEntry

The view entry to add to the specified logical unit identifier.

Return Values

STMF_STATUS_SUCCESS

The API call was successful.

STMF_ERROR_NOT_FOUND

The id specified for logicalUnit was not found in the system.

STMF_INVALID_ARGUMENT

The value specified for viewEntry was not valid.

STMF_VIEW_ENTRY_CONFLICT

Adding this view entry is in conflict with one or more existing view entries.

API Notes

If luNbrValid in the stmfViewEntry structure is set to B_FALSE, the framework will assign a logical unit number for this view entry. veIndexValid must be set to B_FALSE when adding a view entry. On successful return, veIndexValid will be set to B_TRUE and veIndex will contain the view entry index assigned to this view entry by the framework.

3.4 stmfCreateHostGroup

Description

Create a new host group.

Prototype

```
int stmfCreateHostGroup(
          stmfGroupName  hostGroupName
);
```

Parameters

hostGroupName

The name of the host group to be created.

Return Values

STMF_STATUS_SUCCESS

The API call was successful.

STMF_GROUP_NAME_EXISTS

The value specified for hostGroupName already exists in the system.

STMF_INVALID_ARGUMENT

The value specified for hostGroupName was not valid.

3.5 stmfCreateTargetGroup

Description

Create a new target port group.

Prototype

```
int stmfCreateTargetGroup(
          stmfGroupName  targetGroupName
);
```

Parameters

targetGroupName

The name of the target port group to be created.

Return Values

STMF_STATUS_SUCCESS

The API call was successful.

STMF_GROUP_NAME_EXISTS

The value specified for targetGroupName already exists in the system.

STMF_INVALID_ARGUMENT

The value specified for targetGroupName was not valid.

3.6 stmfDeleteHostGroup

Description

Delete an existing host group.

Prototype

```
int stmfDeleteHostGroup(
          stmfGroupName hostGroupName
);
```

Parameters

hostGroupName

The name of the host group being deleted.

Return Values

STMF_STATUS_SUCCESS

The API call was successful.

STMF_ERROR_NOT_FOUND_NOT_FOUND

The specified hostGroupName was not found in the system.

3.7 stmfDeleteTargetGroup

Description

Delete an existing target port group.

Prototype

```
int stmfDeleteTargetGroup(
          stmfGroupName targetGroupName
);
```

Parameters

targetGroupName

The name of the target port group being deleted.

Return Values

STMF_STATUS_SUCCESS

The API call was successful.

STMF_ERROR_NOT_FOUND

The specified targetGroupName was not found in the system.

3.8 stmfDevidFromIscsiName

Description

Convert an iSCSI name to a stmfDevid structure.

Prototype

```
int stmfDevidFromIscsiName(
          wchar_t        iscsiName[224],
          stmfDevid      *devid
);
```

Parameters

iscsiName

A 224-byte array of UTF-8 encoded Unicode characters representing the iSCSI name terminated with the Unicode nul character.

devid

A pointer to a stmfDevid structure allocated by the caller. On successful return, this will contain the converted device identifier. On error, the value of this parameter is undefined.

Return Values

STMF_STATUS_SUCCESS

The API call was successful.

STMF_INVALID_ARGUMENT

The value of iscsiName was not valid iSCSI name.

API Notes

This API will return the devid as a SCSI name string identifier.

3.9 stmfDevidFromWwn

Description

Convert a WWN to a stmfDevid structure.

Prototype

```
int stmfDevidFromWWN(
          uchar_t        wwn[8],
          stmfDevid      *devid
);
```

Parameters

wwn

The 8-byte WWN identifier.

devid

A pointer to a stmfDevid structure allocated by the caller. On successful return, this will contain the converted device identifier. On error, the value of this parameter is undefined.

Return Values

STMF_STATUS_SUCCESS

The API call was successful.

STMF_INVALID_ARGUMENT

The value of wwn was not valid WWN identifier.

API Notes

This API will return the devid as a SCSI name string.

3.10 stmfFreeMemory

Description

Frees memory allocated by this library.

Prototype

```
void stmfFreeMemory(
          void *stmfMemory
);
```

Parameters

memory

A pointer to memory that was previously allocated by this library. If stmfMemory is equal to NULL, the call will return successfully.

Return Values

None

3.11 stmfGetHostGroupList

Description

Retrieves the list of host groups.

Prototype

```
int stmfGetInitiatorGroupList(
          stmfGroupList   **hostGroupList
);
```

Parameters

hostGroupList

A pointer to a pointer to an stmfGroupList structure. On successful return, this will contain a list of host groups.

Return Values

STMF_STATUS_SUCCESS

The API call was successful.

STMF_ERROR_NOMEM

The library was unable to allocate sufficient memory for hostGroupList.

3.12 stmfGetHostGroupMembers

Description

Retrieves the properties of the specified host group.

Prototype

```
int stmfGetHostGroupMembers(
          stmfGroupName       hostGroupName,
          stmfGroupProperties **groupProperties
);
```

Parameters

initiatorGroupName

The name of the host group whose member list is being retrieved.

groupProperties

A pointer to a pointer to an stmfGroupProperties structure. On successful return, this will contain the properties for the specified initiatorGroupName.

Return Values

STMF_STATUS_SUCCESS

The API call was successful.

STMF_ERROR_NOT_FOUND

The specified hostGroupName was not found in the system.

API Notes

The caller should call stmfFreeMemory when this list is no longer needed.

3.13 stmfGetTargetGroupMembers

Description

Retrieves the properties of the specified target port group.

Prototype

```
int stmfGetTargetGroupMembers(
          stmfGroupName       targetGroupName,
          stmfGroupProperties **groupProperties
);
```

Parameters

targetGroupName

The name of the target port group whose member list is being retrieved.

groupProperties

A pointer to a pointer to an stmfGroupProperties structure. On successful return, this will contain the properties for the specified targetGroupName.

Return Values

STMF_STATUS_SUCCESS

The API call was successful.

STMF_ERROR_NOT_FOUND

The specified targetGroupName was not found in the system.

API Notes

The caller should call stmfFreeMemory when this list is no longer needed.

3.14 stmfGetLocalPortProviderList

Description

Retrieves the list of local port providers.

Prototype

```
int stmfGetLocalPortProviderList(
          stmfProviderList     **localPortProviderList
);
```

Parameters

localPortProviderList

A pointer to a pointer to an stmfProviderList structure. On successful return, this will contain a list of local port providers in the system.

Return Values

STMF_STATUS_SUCCESS

The API call was successful.

STMF_ERROR_NOMEM

The library was unable to allocate sufficient memory for localPortProviderList.

API Notes

The caller should call stmfFreeMemory when this list is no longer needed.

3.15 stmfGetLocalPortProviderProperties

Description

Retrieve the properties of the specified local port provider.

Prototype

```
int stmfGetProviderProperties(
          stmfProviderName *providerName,
          stmfLocalPortProviderProperties
               *providerProperties
);
```

Parameters

providerName

The name of the local port provider whose properties are being retrieved.

providerProperties

A pointer to an stmfLocalPortProviderProperties structure. On successful return, this will contain the properties for the specified localPortProviderOid.

Return Values

STMF_STATUS_SUCCESS

The API call was successful.

STMF_ERROR_NOT_FOUND

The specified providerName was not found in the system.

3.16 stmfGetLogicalUnitList

Description

Retrieves the list of logical units.

Prototype

```
int stmfGetLogicalUnitList(
          stmfGuidList    **logicalUnitList
);
```

Parameters

logicalUnitList

A pointer to a pointer to an stmfGuidList structure. On successful return, this will contain a list of logical units in the system.

Return Values

STMF_STATUS_SUCCESS

The API call was successful.

STMF_ERROR_NOMEM

The library was unable to allocate sufficient memory for logicalUnitList.

API Notes

The caller should call stmfFreeMemory when this list is no longer needed.

3.17 stmfGetLogicalUnitProperties

Description

Retrieve the properties of the specified logical unit.

Prototype

```
int stmfGetLogicalUnitProperties(
          stmfGuid                      logicalUnit,
          stmfLogicalUnitProperties     *logicalUnitProps
);
```

Parameters

logicalUnit

The identifier of the logical unit whose properties are being retrieved.

logicalUnitProps

A pointer to an stmfLogicalUnitProperties structure. On successful return, this will contain the properties for the specified logicalUnitOid.

Return Values

STMF_STATUS_SUCCESS

The API call was successful.

STMF_ERROR_OBJECT_NOT_VALID

logicalUnitOid does not refer to a logical unit .

STMF_ERROR_OBJECT_NOT_FOUND

The specified logicalUnitOid was not found in the system.

3.18 stmfGetLogicalUnitProviderList

Description

Retrieves the list of logical unit providers.

Prototype

```
int stmfGetLogicalUnitProviderList(
          stmfProviderList     **logicalUnitProviderList
);
```

Parameters

logicalUnitProviderList

A pointer to a pointer to an stmfProviderList structure. On successful return, this will contain a list of logical unit providers in the system.

Return Values

STMF_STATUS_SUCCESS

The API call was successful.

STMF_ERROR_NOMEM

The library was unable to allocate sufficient memory for logicalUnitProviderList.

API Notes

The caller should call stmfFreeMemory when this list is no longer needed.

3.19 stmfGetLogicalUnitProviderProperties

Description

Retrieve the properties of the specified logical unit provider.

Prototype

```
int stmfGetLogicalUnitProperties(
          stmfProviderName    providerName,
          stmfLogicalUnitProviderProperties
             *providerProperties
);
```

Parameters

providerName

The name of the logical unit provider whose properties are being retrieved.

providerProperties

A pointer to an stmfLogicalUnitProviderProperties structure. On successful return, this will contain the properties for the specified providerName.

Return Values

STMF_STATUS_SUCCESS

The API call was successful.

STMF_ERROR_NOT_FOUND

The specified providerName was not found in the system.

3.20 stmfGetTargetGroupList

Description

Retrieves the list of target port groups.

Prototype

```
int stmfGetTargetGroupList(
          stmfGroupList   **targetGroupList
);
```

Parameters

targetGroupList

A pointer to a pointer to an stmfGroupList structure. On successful return, this will contain a list of target port group object identifiers.

Return Values

STMF_STATUS_SUCCESS

The API call was successful.

STMF_ERROR_NOMEM

The library was unable to allocate sufficient memory for targetGroupList.

3.21 stmfGetTargetList

Description

Retrieves the list of target ports.

Prototype

```
int stmfGetTargetList(
          stmfDevidList   **targetList
);
```

Parameters

targetList

A pointer to a pointer to an stmfDevidList structure. On successful return, this will contain a list of target ports in the system.

Return Values

STMF_STATUS_SUCCESS

The API call was successful.

STMF_ERROR_NOMEM

The library was unable to allocate sufficient memory for targetList.

API Notes

The caller should call stmfFreeMemory when this list is no longer needed.

3.22 stmfGetTargetProperties

Description

Retrieve the properties of the specified target port.

Prototype

```
int stmfGetTargetProperties(
          stmfDevid                target,
          stmfTargetProperties     *targetProperties
);
```

Parameters

target

The identifier of the target port whose properties are being retrieved.

targetProperties

A pointer to an stmfTargetProperties structure allocated by the caller. On successful return, the structure will contain the properties for the specified targetOid.

Return Values

STMF_STATUS_SUCCESS

The API call was successful.

STMF_ERROR_NOT_FOUND

The specified target was not found in the system.

3.23 stmfGetViewEntryList

Description

Retrieves the list of view entries for a specified logical unit.

Prototype

```
int stmfGetViewEntryList(
          stmfGuid            logicalUnit,
          stmfViewEntryList   **viewEntryList
);
```

Parameters

logicalUnit

The identifier of the logical unit for which to retrieve the list of view entries.

viewEntryList

A pointer to a pointer to an stmfViewEntryList structure. On successful return, this will contain a list of view entries for logicalUnit.

Return Values

STMF_STATUS_SUCCESS

The API call was successful.

STMF_ERROR_NOMEM

The library was unable to allocate sufficient memory for viewEntryList.

API Notes

The caller should call stmfFreeMemory when this list is no longer needed.

3.24 stmfOfflineLogicalUnit

Description

Offline of a logical unit that is currently in the online state. Once in the offline state, the logical unit will no longer be capable of servicing requests in the system.

Prototype

```
int stmfOfflineLogicalUnit(
          stmfDevid *logicalUnit,
          boolean_t force
);
```

Parameters

logicalUnit

The identifier of the logical unit to offline.

force

When set to B_TRUE, attempts to offline the device even when it is busy.

Return Values

STMF_STATUS_SUCCESS

The API call was successful.

STMF_ERROR_BUSY

The device is currently busy.

API notes

This API call can be used to offline a logical unit for servicing. Once the logical unit is offline, an initiator port that attempts to issue any SCSI commands to the offlined logical unit will receive a check condition. For purposes of the REPORT LUNS command, the logical unit will no longer appear in the logical unit inventory for any initiator ports to which it is currently mapped via one or more view entries

3.25 stmfOfflineTarget

Description

Offline of a target port that is currently in the online state. Once in the offline state, the target port will no longer be capable of servicing requests in the system.

Prototype

```
int stmfOfflineTarget(
          stmfDevid *target,
          boolean_t force
);
```

Parameters

target

The identifier of the target port to offline.

force

When set to B_TRUE, attempts to offline the device even when it is busy.

Return Values

STMF_STATUS_SUCCESS

The API call was successful.

STMF_ERROR_BUSY

The device is currently busy.

API notes

This API call can be used to offline a target port device for servicing. Once the target port is offline, it will no longer be available to any entities outside of the SCSI Target Mode Framework. Any initiator ports that currently have sessions established via the offlined target port will be logged out.

3.26 stmfOnlineLogicalUnit

Description

Online of a logical unit that is currently in the offline state.

Prototype

```
int stmfOnlineDevice(
          stmfDevid *logicalUnit
);
```

Parameters

logicalUnit

The identifier of the logical unit to online.

Return Values

STMF_STATUS_SUCCESS

The API call was successful.

3.27 stmfOnlineTarget

Description

Online of a target port that is currently in the offline state.

Prototype

```
int stmfOnlineDevice(
          stmfDevid *target
);
```

Parameters

target

The identifier of the target port to online.

Return Values

STMF_STATUS_SUCCESS

The API call was successful.

3.28 stmfRemoveFromHostGroup

Description

Removes an initiator port from an host group.

Prototype

```
int stmfRemoveFromHostGroup(
          stmfGroupName  hostGroupName,
          stmfDevid      initiatorPortName
);
```

Parameters

hostGroupName

The name of the host group from which the specified hostGroupName is being removed.

initiatorPortName

The device identifier of the initiator port to remove from the specified host group.

Return Values

STMF_STATUS_SUCCESS

The API call was successful.

STMF_ERROR_GROUP_NOT_FOUND

The specified hostGroupName was not found in the system.

STMF_ERROR_MEMBER_NOT_FOUND

The specified initiatorPortName was not found in the system.

3.29 stmfRemoveFromTargetGroup

Description

Removes an target port from an target port group.

Prototype

```
int stmfRemoveFromTargetGroup(
          stmfGroupName  targetGroupName,
          stmfDevid      targetName
);
```

Parameters

targetGroupName

The name of the target port group from which the specified targetGroupName is being removed.

targetName

The device identifier of the target port to remove from the specified target port group.

Return Values

STMF_STATUS_SUCCESS

The API call was successful.

STMF_ERROR_GROUP_NOT_FOUND

The specified targetGroupName was not found in the system.

STMF_ERROR_MEMBER_NOT_FOUND

The specified targetName was not found in the system.

3.30 stmfRemoveViewEntry

Description

Remove a view entry from the system.

Prototype

```
int stmfRemoveViewEntry(
          stmfGuid          *logicalUnit,
          stmfViewEntryName viewEntryName
);
```

Parameters

logicalUnit

The identifier of the logical unit for the view entry being removed.

viewEntryName

The name of the view entry to be removed.

Return Values

STMF_STATUS_SUCCESS

The API call was successful.

STMF_ERROR_NOT_FOUND

The specified logicalUnit/viewEntryName was not found in the system.

4 Coding Examples

The following section has examples of how to use some of the API calls.

4.1 Example of how to create a host group with one member

```
void
main() {
        int stmfStatus;
        stmfDevid initiatorPortDevid;
        uchar_t wwn[8] = {0x20, 0x01, 0x00, 0xe0, 0x8b,
                0xb0, 0x92, 0x21};

        stmfStatus = stmfCreateHostGroup(L"Host-A");
        if (STMF_ERROR(stmfStatus)) {
                printf("Error occurred on initiator port”
                    “ group create: %d\n", stmfStatus);
                exit(1);
        }
        stmfStatus = stmfDevidFromWwn(wwn, &initiatorPortDevid);
        if (STMF_ERROR(stmfStatus)){
            printf("Error occurred on initiator port group”
                “ create: %d\n", stmfStatus);
                exit(1);
        }
        stmfStatus =
            stmfAddToHostGroup(L“Host-A”,initiatorPortDevid);
        if (STMF_ERROR(stmfStatus)) {
                printf("Error occurred on initiator port group”
                    “ create: %d\n", stmfStatus);
                exit(1);
        }
}
```

4.2 Example of how to retrieve the list of logical units

```
void
main() {
        int stmfStatus;
        int i;
        stmfGuidList *logicalUnitList;
        stmfLogicalUnitProperties logicalUnitProps;

        stmfStatus =
            stmfGetLogicalUnitList(&logicalUnitList);
        if (STMF_ERROR(stmfStatus)) {
                printf("Error occurred on logical unit list”
                    “ get: %d\n", stmfStatus);
                exit(1);
        }

        for (i = 0; i < logicalUnitList->cnt; i++) {
            stmfStatus = stmfGetLogicalUnitProperties(
                    logicalUnitList->devid[i],
                    &logicalUnitProps);
            if (STMF_ERROR(stmfStatus)) {
                printf("Error occurred on logical unit prop”
                    “ get: %d\n", stmfStatus);
                    exit(1);
            }
            printf("Logical Unit Properties:\n");
            printf("   Alias: %s\n", logicalUnitProps.alias);
            printf("   Vendor: %s\n",
                logicalUnitProps.vendor);
            printf("   Product: %s\n",
                logicalUnitProps.product);
        }
}
```

4.3 Example of how to add a view entry to a logical unit

```
void
main() {
        int status;
        stmfGuid logicalUnit;
        stmfViewEntry viewEntry;
        stmfGroupName hostGroup
        int ret;

        viewEntry.allHosts = B_FALSE;

        /* set host group */
        bcopy(hostGroup, viewEntry.hostGroup, sizeof (stmfGroupName));

        /* make available to all target ports */
        viewEntry.allTargetPorts = B_TRUE;

        /* let the framework select a logical unit number */
        viewEntry.luNbrValid = B_FALSE;

        /* add the view to the logical unit */
        status = stmfAddViewEntry(logicalUnit, &viewEntry);
        if (status != STMF_STATUS_SUCCESS) {
                printf("Error adding view entry\n");
                exit(1);
        }
}

static int
findInitiatorGroupOid(char *groupName, stmfOid *groupOid)
{
        stmfGroupProperties *initiatorPortGroupProperties;
        stmfOidList *initiatorPortGroupList;
        int i;
        int ret = 1;
        int status;

        status =
            stmfGetInitiatorGroupList(
                  &initiatorPortGroupList);
        if (status != STMF_STATUS_SUCCESS) {
                printf("Error retrieving initiator port”
                    “ group list\n");
                exit(1);
        }

        /* Find requested initiator port group name */
        for (i = 0; i <
             initiatorPortGroupList->oidCnt; i++){
                status =
                    stmfGetGroupProperties(
                        initiatorPortGroupList->oid[i],
```

```
                        &initiatorPortGroupProperties);
                if (status != STMF_STATUS_SUCCESS) {
                        printf("Error retrieving group"
                            " properties\n");
                        exit(1);
                }
                if (strcmp(
                        initiatorPortGroupProperties->groupName,
                            groupName) == 0) {
                        bcopy(initiatorPortGroupList->oid[i],
                            groupOid, sizeof (stmfOid));
                        stmfFreeMemory(
                            initiatorPortGroupProperties);
                        ret = 0;
                        break;
                }
                stmfFreeMemory(initiatorPortGroupProperties);
        }
        stmfFreeMemory(initiatorPortGroupList);
        return (ret);
}
```