The Storage Resource Manager
Functional Interface Specification

Version 3.0
ReleaseCandidate.1

15 March 2006

Collaboration Web: http://sdm.lbl.gov/srm-wg
Document Location: http://sdm.lbl.gov/srm-wg/doc/SRM.v3.0.rc1.pdf

Editors:

Arie Shoshani	Lawrence Berkeley National Laboratory
Alex Sim	Lawrence Berkeley National Laboratory

Contributors:

Peter Kunszt	EGEE Project (Enabling Grids for E-sciencE), CERN
Don Petravick Timur Perelmutov	Fermi National Accelerator Laboratory (FNAL)
Junmin Gu	Lawrence Berkeley National Laboratory (LBNL)
Jean-Philippe Baud James Casey	LHC Computing Project (LCG), CERN
Jens Jensen Owen Synge	Rutherford Appleton Laboratory (RAL), England
Michael Haddox-Schatz Bryan Hess Andy Kowalski Chip Watson	Thomas Jefferson National Accelerator Facility (TJNAF)

Copyright Notice

This manuscript has been supported by the Office of Energy Research, Office of Computational and Technology Research, Division of Mathematical, Information, and

Computational Sciences, of the U.S. Department of Energy under Contract No. DE-AC03-76SF00098.

Table of Contents

I. Introduction

This document contains the functional interface specification of SRM 3.0. It is designed to support the functionality of previous SRM versions (specifically, v1.1 and v2.1.1) but is organized to support the functionality by "core features", and "advanced features" functions.

Storage Resource Managers (SRMs) are middleware components whose function is to provide dynamic space allocation and file management of shared storage components on the Grid. Introductory information about SRM concepts and the design of their functionality can be found in http://sdm.lbl.gov/srm-wg/papers/SRM.book.chapter.pdf.

In this introduction we describe the organization of SRM v3.0 specification. We start with description of how to represent "core features" and advanced features" in the specification. Specifically, we address the issue of having functions that may be involved in multiple features. This is followed file a section on space and file types (referred to as "volatile", durable" and "permanent"), and the discovery of which features and types are supported by a certain SRM implementation. Next, are sections that describe the semantics of releasing and removing files, as well as reserving and releasing spaces. Finally, we describe the behavior of directory management when multiple spaces are managed for the client.

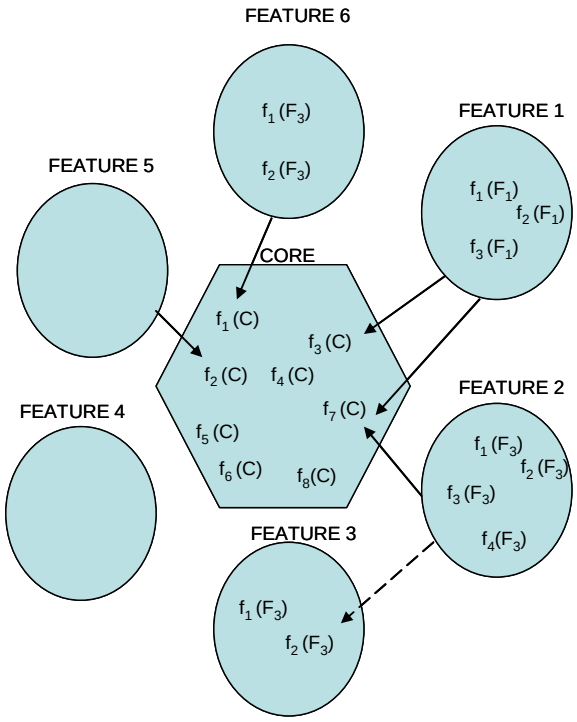
Following the introductory section, we included a section that describes the evolution of SRM versions and the relationship between functional specifications and operational specifications. The detailed specification of SRM v3.0 follows.

I.1. Extending parameters according to features

The functional design of SRM v3.0 calls for having "core" functions that all SRM implementations must support, and "advanced Features" functions that are optional. Thus, it is inevitable that some of the core functions will have additional functionality if some advanced feature is supported by the SRM. For example, a "srmPrepareToGet" as a core function does not have to specify a space_token, but if the "space reservation" feature is supported, the client may want to specify the space that the files will go into, and thus the parameter for the space_token must exist.

The advanced features supported in this version include: space management, directory management, authorization management, remote accessfunctions, and request administrative functions.
If an SRM supports an advanced feature, then all the functions in that feature must be supported.

The relationship between core functions and advanced feature functions is shown in the figure below. The core functions in the center can be affected by certain features. For example, the figure shows that Feature 6 effects function 1 of core, and Feature 1 effects functions 3 and 7 of core. Note that two or more features can affect the same function, as is the case with function 7 of core. In addition, it is also possible, although not very common, that a feature can affect the function of another feature. This is shown with the broken line from Feature 2 that effects function 2 of Feature 3.



We represent this in this specification document is by using "behavior" sub-sections. For the core functions there are two subsections: "core behavior", and "behavior with advanced feature". Under "behavior with advanced features" we have subsections for each of the features that affect particular functions.

I.2. Space and File Types

I.2.a. File Types

The concepts of permanent and temporary file types are familiar concepts, but when dealing with shared storage on a Grid, temporary files require additional functionality. Temporary files on shared Grid spaces cannot be removed arbitrarily. Some minimal amount of time must be guaranteed by the SRM for the client to rely on. This is the reason for a lifetime of a file. This feature of a lifetime for a file is associated with each client accessing a file. That is, a lifetime is associated with a file for a client when the file is made available to the client. If another client requests the same file later, the file gets a fresh lifetime associated with that client (regardless of whether the SRM chooses to replicate the file to a separate space or not). We refer to a file that is temporary in nature, but has a lifetime guarantee as a *volatile* file. The concept of volatile files is very useful for dynamic space usage, automatic garbage collection, and sharing of files on a temporary basis. In contrast, Permanent files have no lifetime associated with them, and can only be removed by the owner of the file

For grid applications, one needs another type of a file that has properties of both permanent and volatile files. A *durable* file has the behavior of a volatile file in that it has a lifetime associated

with it, but also the behavior of a permanent file in that when the lifetime expires the file is not automatically eligible for removal. Instead, the SRM may advise the client or an administrator that the lifetime expired or take some other action. For example, the SRM may move the file to a permanent space, release the durable space, and notify the client. A durable file type is necessary in order to provide temporary space for files (such as files generated by large simulations), which need to be eventually archived. However, the archiving process is usually too slow, thus slowing down and wasting the computational resources. By storing durable files into available shared disk caches, the simulation can continue efficiently, yet it is guaranteed that the files will not be removed prematurely, and the files can be moved to the archive as a secondary task. Similar to volatile files, durable files can be released by the client as soon as the files have been moved to a permanent location, or as soon as the client has no need for the files.

Permanent, durable, and volatile files are file types that provide the flexibility needed to manage files on the Grid for various use cases. Permanent files are files that need to be stored for the long term, such as the results of a long running simulation. Durable files are useful for dumping files as soon as possible to disk caches in order not to slow down computation resources. Volatile files are useful in analysis use cases, where files have to be replicated temporarily but a lifetime duration is guaranteed. Another advantage of volatile files is that they can be shared at the discretion of the SRM.

I.2.b. Space types

SRMs manage shared spaces. If the "space reservation" feature is supported, then clients can negotiate and acquire space that the SRM assigns to them. Otherwise, the SRM assigns a default space size that depends on its policy. Similar to file types, the spaces assigned by the SRM have types. Thus, Volatile and Durable space types have a lifetime associated with them. Permanent space has unlimited lifetime.

If the "space reservation" feature is supported, a client can acquire multiple spaces of the same type. For this reason, spaces are assigned a space_token. Generally, when a request is made to bring a file into a space managed by the SRM, a space token is expected. If it is not provided, and the client has only one space of that type, the file is put into that space.

Files of a certain type can only be assigned to a space of the same type. Furthermore, the lifetime of a file cannot exceed the lifetime of the space it is assigned to.

I.2.c. The support of file types by core and advanced features

An SRM can support any combination of the three space (and file) types: Volatile, Durable, and Permanent. Given that an SRM supports a particular combination of space types, it may choose to support any subset of that combination for the core functions and a different subset for any of its advanced features functions. However, if certain space types are supported for some feature, then all of the functions of that feature must support that space type.

For example, an SRM that supports the space types of Volatile and Permanent, and the advance feature "space reservation", may choose to support Volatile and Permanent for the core functions, and only Volatile for "space reservation". This flexibility is necessary in order to allow various SRMs to be developed for different types of storage systems.

The types of files supported for each feature must be discoverable when invoking the feature discovery function. For the above example, the discovery function will return: {core [volatile, permanent], space_reservation [volatile]}.

I.3. Releasing and removing files

srmPrepareToGet and srmPrepareToPut put or get local files only. In order to get files from remote sites or put files into remote sites, the srmRemoteCopy function has to be used. From the client's point of view files brought in by the SRM as a result of srmPrepareToGet, srmPrepareToPut, and srmRemoteCopy requests are always brought into one or more local spaces that are assigned to the client. The spaces can be assigned by default, as in the "core" feature, or acquired explicitely if the "space reservation" feature is supported. Consequently, releasing and removing files refers to local spaces only.

We use the term "releasing a file" to mean that the file is marked as eligible for removal. The "release a file" action takes place when a client no longer needs the file. The "release" action takes place either by a direct request to release the file using the release command, or implicitly when abort or cleanup commands are issued. Released files are removed by the SRM only when space is needed or when the removal of the files is explicitly requested by the client. We explain next the behavior of release, cleanup and abort functions.

srmReleaseFiles (Request_Token, SURLs) can only be used for files previously pinned as a result of srmPrepareToGet, srmPrepareToPut, or srmRemoteCopy. The files designated by SURLs are marked as released, and are removed only when space is needed. For example, if additional files need to be brought into the space the SRM will remove one or more of the released files to make space for the additional files. To remove files, the srmRm function has to be used.

srmCleanupFilesFromSpace (Space_Token) can be used to release files regardless of the request that brought them in. It has a remove-parameter (flag) that can be set to remove the files from the space, rather than only releasing them.

Aborting files is possible at the request-level – for aborting the entire request, and at the file-level – for aborting a specified subset of the files in the request. Abort functions release files that are in spaces, and remove files from the queue of files that were not brought into the space yet. Both file-level and request-level aborts have a remove-parameter (flag) that can be set if the client wishes the files to be removed, not only released.

Aborting an srmPrepareToGet and srmPrepareToPut has only a local effect. However, aborting an srmRemoteCopy has to be propagated to the remote site. In the case of aborting an srmPrepareToGet and srmPrepareToPut the files released are files that were brought into the local space (assigned to the client) as a result of that function. If the remove-flag is set, the files that were brought into local space are removed.

Aborting an srmRemoteCopy request has two cases to consider. When the srmRemoteCopy function is from the remote SRM to local SRM (also referred to as a "copy-pull"), and vice versa, if the srmRemoteCopy command is from local SRM to a remote SRM (also referred to as a "copy-push"). For copy-pull, the local SRM issues an srmPrepareToGet request to the remote SRM, and for copy-push an srmPrepareToPut request is issued. When the srmRemoteCopy is aborted, it is propagated to the remote site by aborting the srmPrepareToGet or the srmPrepareToPut request as well. Furthermore, if the abort function has the remove-flag set, then the propagated abort should have this flag set, too. In the case of copy-push, the srmPrepareToPut gets aborted with the remove-flag set, which has the effect of removing the copied files from the remote SRM. In the case of a copy-pull, the srmPrepareToGet to the remote site is aborted, but the remove-flag effects only the local site.

I.4. Reserving and releasing spaces

I.4.a. Reserving spaces

Given that an SRM support the Space Management feature for certain space types, it is possible to make space reservation requests for spaces of these types. The space reservation request can specify a minimum "guaranteed" space, and a "total" space (which is larger than the "guaranteed"). The difference between total and guaranteed is referred to as "best effort". The SRM can honor the request, or return lower values than requested. At this point the SRM assumes that the offer is accepted. The client can refuse the space offer, by simply releasing that space (see below).

Different space reservation requests are needed for each space requested. It is possible to make reservations to multiple spaces of each type, as long as the total does not exceed the SRM allocation. The SRM allocation per client may be an internal feature, or may be dictated by the virtual organization (VO) that owns the space. Currently, there are no mechanisms (interface) in place to communicate the VO's allocation per client.

Space reservations to volatile and durable spaces are made for a lifetime. The lifetime is subject to the SRM granting it. Here again, the SRM may return a shorter lifetime for a space. If the client wishes to refuse the modified lifetime, the space should be released, otherwise it is considered as granted.

There is no way of consolidating spaces. However, one can request to increase or shrink an existing space as well as change the lifetime of a space with the srmUpdateSpace function.

As mentioned above all files assigned to a space must have the same file type as the space, and the lifetime of a file put in a space must be shorter than the lifetime of the space it is put into.

For core functionality, space reservation is performed by default. That is, space is allocated to the client by the SRM according to its internal policies. The policy for default space allocation and space types can be found in the discovery function. The default space amount can be expressed as "guaranteed" and "total". As, above, the difference between total and guaranteed is referred to as "best effort". The actual amount of space that was assigned can be found

dynamically with the srmGetSpaceMetadata function. The same default behavior applied in the case that the Space Management feature is supported, but no space request was made.

I.4.b. Releasing spaces

Releasing a space requires a space token and of course only the owner of the space can release it. Releasing a space, that has no files in it, is straightforward – it has the action that the space is no longer available. It is not possible to request re-use of a space that was released. Instead, a new request must be made.

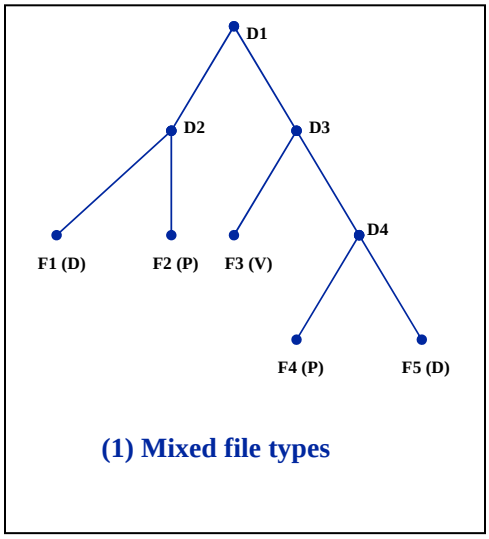
If the release space has files in it, the space that is not occupied by files is released immediately. However, the action regarding the files that are in the space depends on the type of space. We describe the action for each next.

- For volatile space, the files that were released prior to the space release request, will be removed from the space. Files that were not released, will be left in the space till their lifetime expires or till they are released, and the space they occupy is released. The space not occupied by files will be released. Therefore, it is good to "cleanup" the space before releasing it.
- For Durable file, the files that were released prior to the space release request, will be removed from the space. Files that were not released, will be left in the space till their lifetime expires, and then the files archived (and client is notified), and only then the space they occupy is released.
- For permanent files, the files must be removed or released before the space they occupy is released. Otherwise, the files stay in the space and the space they occupy is not released.

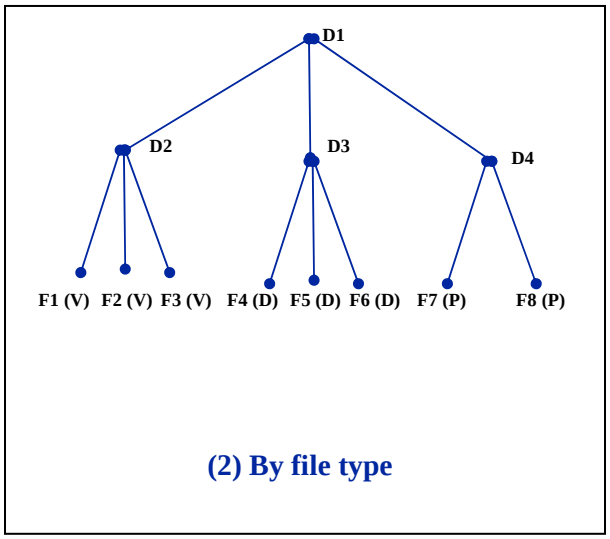
I.5. Directory Management

The directory management feature is intended to provide the client the capability to organize files managed by the SRM in directories and to assign names to the files that reside in the directories. The SRM supports the usual functions: srmLs, srmMkdir, srmRmdir, srmRm, srmMv, and srmCp. srmLs and srmRm are considered core functions. No "link" functions are supported.

In the case that the SRM supports a single space for for each client, the behavior of the above functions is straightforward. However, what is the behavior when there are multiple space types and/or multiple spaces per type? There are two options: to support a directory for each space, or a single directory over all the spaces. SRMs support the second option. Thus, an SRM can have files of different types in a single directory, while these files may belong to different spaces. This is illustrated in the diagram below in part (1) where (D), (V), and (P) represent file types. The spaces that files belong to are not shown in the diagam, but are known to the SRM. Note that this choice of a single directory over all the spaces, also gives the client the flexibility of how to organize the directories. For example, the diagram in part (2) below is a choice of having each sub-directory D2, D3, and D4, contain different file types.



(1) Mixed file types



(2) By file type

The advantage of a single directory for all space and file types, is that it is possible to move files between spaces and even change their type without changing the directory structure. Thus, this permits the SRM to manage all the spaces virtually, regardless of where they are physically placed. The issue of whether files are physically copied when they are moved between spaces or when their type changes, is an implementation choice. The same behavior of files assigned to spaces according to space type is visible to the clients regardless of the implementation choice.

I.6. Acknowledgements

While the initial ideas of SRMs were first introduced by people from the Lawrence Berkeley National Laboratory (LBNL), many of the ideas and concepts described in this paper were developed over time and suggested by various people involved in collaboration meetings and discussions. In particular, we would like to acknowledge the contributions of people from Fermi National Accelerator Laboratory (FNAL), Thomas Jefferson National Accelerator Facility (Jlab), European Organization for Nuclear Research (CERN, including European Data Grid, LCG and EGEE), and Rutherford Appleton Laboratory (RAL). Finally, the people who participated in meetings and discussions over the last few years also have contributed to the development and refinement of the concepts described here. We apologize for any names that we might have overlooked in the list of contributors.

I.7. The history of SRM versions

This document contains the functional interface specification of SRM 3.0. It is designed to support the functionality of SRM 2.1.1 and SRM 1.1 (see http://sdm.lbl.gov/srm-wg/documents.html), but but is organized to support the functionality by "core features", and "advanced features" functions.

I.7.a. The relationship between functional and operational specifications

The functional specification document is to have an independent functional interface specification that operational specifications may be derived from this document. Figure 1 shows the relationship between specifications.

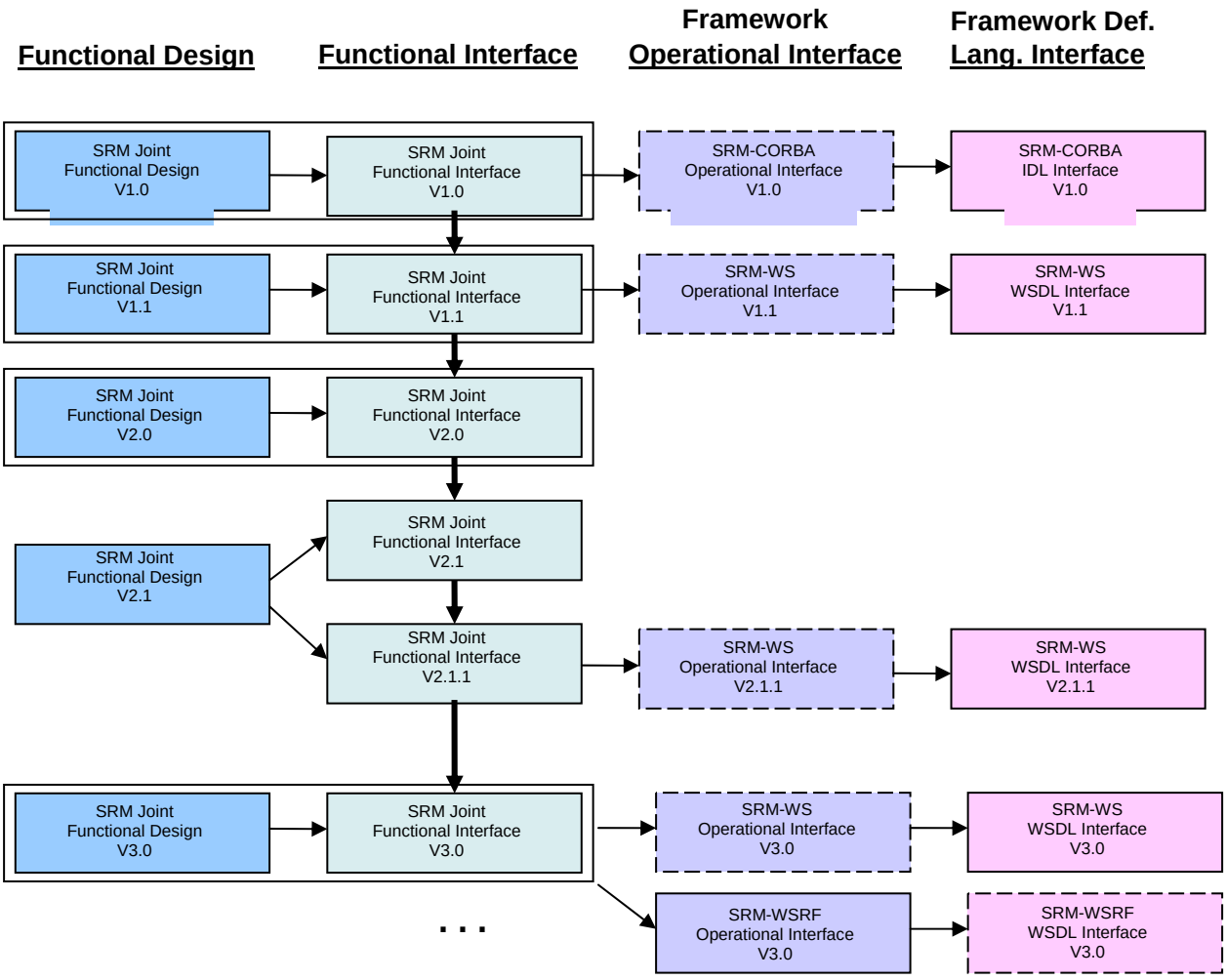


Figure 1 Relationship betweeen specifications

II. Notes

SRM's "local" storage and "local" files mean that the storage spaces and files are under the SRM's administrative control.

Core function may support File Storage Type of volatile and permanent. Advanced Space Function may support durable file storage type.

When an advanced feature function is not supported and an option for the advanced feature is provided to the server, an error must be returned from the server

SiteURL(SURL) or StorageFileName(StFN) must be *anyURI* in the operational specification.

- The definition of the type "*anyURI*" used below is compliant with the XML standard. See http://www.w3.org/TR/xmlschema-2/#anyURI. It is defined as: "The lexical space of *anyURI* is finite-length character sequences which, when the algorithm defined in Section 5.4 of [XML Linking Language] is applied to them, result in strings which are legal URIs according to [RFC 2396], as amended by [RFC 2732]".
- The definition *SURL* is used for both site URL and the "Storage File Name" (stFN). This was done in order to simplify the notation. Recall that stFN is the file path/name of the intended storage location when a file is put (or copied) into an SRM controlled space. Thus, a stFN can be thought of a special case of an *SURL*, where the protocol is "srm" and the machine:port is local to the SRM. For example, when the request srmRemoteCopy is made, the source file is specified by a site URL, and the target location can be optionally specified as a stFN. By considering the stFN a special case of an SURL, an srmRemoteCopy takes *SURL*s as both the source and target parameters.

Transfer URL (TURL) must be *anyURI* in the operational specification.

Request tokens and space tokens are strings, assigned by SRM, and are unique and immutable (non-reusable). For example, if the date:time is part of the request reference it will be immutable.

User ID is in any form of user authentication information in the operational specification.
Authorization ID is in any form of user authorization information in the operational specification.

- Authorization ID : from the SASL RFC 2222
  During the authentication protocol exchange, the mechanism performs authentication, transmits an authorization identity (frequently known as a user id) from the client to server…. The transmitted authorization identity may be different than the identity in the client's authentication credentials. This permits agents such as proxy servers to authenticate using their own credentials, yet request the access privileges of the identity for which they are proxying. With any mechanism, transmitting an authorization identity of the empty string directs the server to derive an authorization identity from the client's authentication credentials.

Storage system info is string.

- Storage system info may contain but is not limited to the following: storage device, storage login ID, storage login authorization.
- Storage system info may be a string that contains the login and password required by the storage system. For example, it might have a form of login:passwd@hostname, where ":" is a reserved separator between login and passwd. If hostname is not provided, it is defaulted to what's in the accompanying site URL or the host of SRM.
- Storage system info is added in the parameters of functions where storage access is needed. This is to simplify the case so that all files sent to the request share the same storage system info. If storage system info is different for each file, SRM needs a different request for those files.

Regarding file sharing by the SRM, it is an implementation choice. An SRM can choose to share files by providing multiple clients access to the same physical file, or by copying a file into another space. Either way, if an SRM chooses to share a file (that is, to avoid reading a file over again from the source site) the SRM should check with the source site whether the client has a read/write permission. Only if permission is granted, the file can be shared.

III. Terminology and Notation

The keywords "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].

This specification uses a notational convention, refered to as “Pseudo-schemas”. A Pseudo-schema uses a BNF-style convention to describe attributes and elements:

- element … a named variable that has a type: string, integer, boolean, or enumerated.
  e.g. “string SURL”
- item … a single element, a tuple, a list, or a set
- {item, item, …} … tuple of items
- item() … list of items (ordered collection)
- item[] … set of items (unordered collection, array)
- _ … underline as required (as opposed to optional)
- ::= … definition operator
- {item | item | …} … choice operator
- {item} … a way of enclosing a complex item for the list or set notation.
  e.g. {MY_TYPE()}[] is used to denote a set of MY_TYPE lists
- String … set of characters
- Enum … enumerated strings
- Int … number (to be defined in the operational specification, e.g. long or unsigned long)
- Boolean … true (1) or false (0)

Underlined attributes are REQUIRED. If a set is not required but some of attirubutes are underlined, then those underlined attributes are required when the set is provided: For example, ReturnSURLStatus { String SURL, EnumStatusCode statusCode, string explanation } represents that ReturnSURLStatus is not required, but when it is provided, SURL and statusCode are required to be provided.

1. Common Type Definitions

Notes:

- The following definitions are not data type definitions nor bound to any programming languages. When defined in operational specifications, the exact operational data type name shall be different.
- Here only repeating definitions are listed. Those definitions that appeared only once are listed under the associated function.

Namespace SRM

EnumFileStorageType ::= VOLATILE |
PERMANENT |
DURABLE

- Volatile file has an expiration time and the storage may delete all traces of the file when it expires.
- Permanent file has no expiration time.
- Durable file has an expiration time, but the storage may not delete the file, and should raise error condition instead.

EnumFileType ::= FILE |
DIRECTORY |
LINK

EnumRetentionQualityMode ::= REPLICA |
OUTPUT |
CUSTODIAL

- Quality of Retention is a kind of Quality of Service. It refers to the probability that the storage system lose a file. Numeric probabilities are self-assigned.
    - Replica quality has the highest probablity of loss, but is appropriate for data that can be replace because other copies can be accessed in a timely fasion.
    - Output quality is an intermediate level and refers to the data which can be replace by lenghty or effort-full processes.
    - Custodial quality provides low proabality of loss.

EnumAccessLatencyMode ::= ONLINE |
NEARLINE

- Files may be Online, Nearline or Offline. These terms are used to describe how latency to access a file is improvable. Latency is improved by storage systems replicating a file such that its access latency is online.
    - The ONLINE cache of a storage system is the part of the storage system which provides file with online latencies.
    - For the SRM we only keep ONLINE and NEARLINE.
    - For completeness, we also describe OFFLINE here.

- ONLINE has the lowest latency possible. No further latency improvements are applied to online files.
- NEARLINE file can have their latency improved to online latency automatically by staging the file to online cache.
- OFFLINE files need a human to be involved to achieve online latency.

EnumOverwriteMode ::= NEVER |
ALWAYS |
WHENFILESAREDIFFERENT

EnumPermissionMode ::= NONE | X | W | WX | R | RX | RW | RWX
EnumPermissionType ::= ADD | REMOVE | CHANGE

EnumRequestType ::= PREPARE_TO_GET |
PREPARE_TO_PUT |
REMOTE_COPY |
CHANGE_FILE_STORAGE_TYPE |
CLEANUP_FILES_FROM_SPACE |
LS |
LS_DETAILS |
RM |
CP |
MV

- EnumRequestType is for the srmGetRequestSummary and srmGetRequestTokens, indicating the type of request.

UTCtime ::= date and time in Coordinated Universal Time
(UTC, formerly GMT) with no local time extention

EnumStatusCode ::= SRM_SUCCESS |
SRM_FAILURE |
SRM_PARTIAL_SUCCESS |
SRM_REQUEST_QUEUED |
SRM_REQUEST_INPROGRESS |
SRM_REQUEST_FINISHED |
SRM_REQUEST_SUSPENDED |
SRM_ABORTED |
SRM_RELEASED |
SRM_FILE_PINNED |
SRM_FILE_IN_SPACE |
SRM_FILE_IN_CACHE |
SRM_FILE_IN_USE |
SRM_FILE_REMOVED |
SRM_CUSTOM_STATUS |
SRM_SPACE_AVAILABLE |
SRM_LOWER_SPACE_GRANTED

EnumErrorCode ::= SRM_NOT_SUPPORTED |
SRM_INVALID_REQUEST |
SRM_INVALID_PATH |
SRM_INVALID_REQUEST_TOKEN |
SRM_INVALID_SPACE_TOKEN |
SRM_FILE_LIFETIME_EXPIRED |
SRM_EXCEED_ALLOCATION |
SRM_NO_USER_SPACE |
SRM_NO_FREE_SPACE |
SRM_DUPLICATION_ERROR |
SRM_TOO_MANY_RESULTS |
SRM_INTERNAL_ERROR |
SRM_FATAL_INTERNAL_ERROR |
SRM_AUTHENTICATION_FAILURE |
SRM_AUTHORIZATION_FAILURE |
SRM_SPACE_LIFETIME_EXPIRED |
SRM_NON_EMPTY_DIRECTORY

- The SRM status codes and error codes are explained in Appendix A.

2. Core Functions

summary:

srmAbortRequest
srmAbortRequestedFiles
srmChangeFileStorageType
srmChangeFileStorageTypeStatus
srmExtendFileLifetime
srmExtendRequestedFileLifetime
srmGetFeatures
srmGetRequestSummary
srmGetRequestTokens
srmGetSRMStorageInfo
srmGetTransferProtocols
srmLs
srmLsStatus
srmPrepareToGet
srmPrepareToPut
srmPutFileDone
srmPutRequestDone
srmReleaseFiles
srmReleaseRequestedFiles
srmRm
srmRmStatus
srmStatusOfGetRequest
srmStatusOfPutRequest

details:

2.1. srmAbortRequest

2.1.1. NAME

srmAbortRequest - terminates the previously submitted request

2.1.2. SYNOPSIS

In	Out	
String userID	ReturnRequestStatus {	
String authorizationID	EnumStatusCode	statusCode,
String storageSystemInfo	string	explanation,
String requestToken	EnumErrorCode	errorCode
Boolean remove	} returnStatus	

2.1.3. DESCRIPTION

srmAbortRequest() allows clients to prematurely terminate asynchronous requests of any types. It may involve data transfer requests initiated by a call to *srmPrepareToGet*(), *srmPrepareToPut*() or *srmRemoteCopy*() as well as calls such as *srmLs*(). The effect of *srmAbortRequest*() depends on the type of request. For data transfer request, the SRM will attempt a complete cleanup of running transfers and files in intermediate state. See 2.1.6.

- String *userID* (required)
  User authentication identifier. See Section II for notes.
- String *authorizationID*
  User authorization information. The *authorizationID* may be NULL. See Section II for notes.
- String *storageSystemInfo*
  String containining information specific to the underlying storage system. The *storageSystemInfo* may be NULL. See Section II for notes.
- String *requestToken* (required)
  A token associated with the request to be aborted. The *requestToken* was returned by the function initiating the request (e.g. *srmPrepareToGet*()).
- Boolean *remove*
  Files eligible for removal as the result of the cleanup must be removed immediately. Default value is false.
- ReturnRequestStatus *returnStatus* (required)
  Output parameter reporting the success or failure of the request. In case of failure, the associated *errorCode* is returned.

2.1.4. RETURN CODE

On successful abort, the *statusCode* is set to SRM_SUCCESS.
On failure, the *statusCode* is set to SRM_FAILURE and the *errorCode* is set.

2.1.5. ERROR CODE

When status is failure, *errorCode* is set to one of the following:

SRM_AUTHENTICATION_FAILURE
- SRM server failed to authenticate the client

SRM_AUTHORIZATION_FAILURE
- client is not authorized to abort the request specified by the *requestToken*

SRM_INVALID_REQUEST_TOKEN
- *requestToken* does not refer to an existing request

2.1.6. NOTES on the Core Behavior

a) Terminate all files in the request regardless of the file state. Remove files from the queue, and release cached files if applicable. Expired files are released, too.

b) When remove flag is true, all files that were copied into the client's space as a result of the request must be removed. If the copy generated an SURL, the SURL will be removed as well.

c) Abort must be allowed to all requests with *requestToken*, including requests such as *srmLs* or *srmRemoteCopy*.
d) If any file transfers are active, SRM should attempt to stop the transfer. If not possible, abort takes place after the pending transfers are completed.

2.1.7. NOTES on the Advanced Behavior with Remote Access Feature

a) *srmAbortRequest* with the *remove* flag on must be propagated for any calls to remote SRMs.

2.1.8. SEE ALSO

srmAbortRequestedFiles, srmPrepareToGet, srmPrepareToPut, srmRemoteCopy, srmLs, srmLsDetails

2.2. srmAbortRequestedFiles

2.2.1. NAME

srmAbortRequestedFiles - aborts selected files from the request.

2.2.2. SYNOPSIS

In	Out
String userID	ReturnRequestStatus {
String authorizationID	EnumStatusCode statusCode,
String storageSystemInfo	string explanation,
String requestToken	EnumErrorCode errorCode
String SURL[]	} returnStatus
Boolean remove	ReturnSURLStatus {
	String SURL,
	EnumStatusCode statusCode,
	string explanation,
	EnumErrorCode errorCode
	} returnSURLStatus[]

2.2.3. DESCRIPTION

srmAbortRequestedFiles() allows clients to abort selective file requests from the asynchronous requests of any type. It may include data transfer requests initiated by a call to srmPrepareToGet(), srmPrepareToPut(), or srmRemoteCopy(). The effect of a srmAbortRequestedFiles() depends on the type of the request. See 2.2.6.

- String *userID* (required)
  User authentication identifier. See Section II for notes.
- String *authorizationID*
  User authorization information. The *authorizationID* may be NULL. See Section II for notes.
- String *storageSystemInfo*
  String containining information specific to the underlying storage system. The *storageSystemInfo* may be NULL. See Section II for notes.
- String *requestToken* (required)

A token associated with the request to be aborted. The requestToken was returned by the function initiating the request (e.g. *srmPrepareToGet*()).

- String *SURL*[] (required)
  SURLs associated with the request token need to be provided.
- Boolean *remove*
  Files eligible for removal as the result of the cleanup must be removed immediately. Default value is false.
- ReturnRequestStatus *returnStatus* (required)
  Output parameter reporting the success or failure of the request. In case of failure, the associated *errorCode* is returned.
- ReturnSURLStatus returnSURLStatus []
  Output parameter reporting the success or failure of the each SURL requests. *returnSURLStatusL*[] may be empty and NULL. If returned to the client, *SURL* and its *statusCode* are required to return.
  - String *SURL* (required)
    SURL that client has requested to abort.

2.2.4. RETURN CODE

On successful abort, the *statusCode* is set to SRM_SUCCESS.
On failure, the *statusCode* is set to SRM_FAILURE and the *errorCode* is set.
If only partial files were successful, the *statusCode* is set to SRM_PARTIAL_SUCCESS, and the *restunSURLStatus* should explain on those failed files.

2.2.5. ERROR CODE

When status is failure, *errorCode* is set to one of the followings:

For request level resturnStatus,

SRM_AUTHENTICATION_FAILURE
- SRM server failed to authenticate the client

SRM_AUTHORIZATION_FAILURE
- client is not authorized to abort files in the request specified by the *requestToken*

SRM_INVALID_REQUEST_TOKEN
- *requestToken* does not refer to an existing request

SRM_INVALID_REQUEST
- *SURL* is empty.

For file level returnStatus,

SRM_INVALID_PATH
- *SURL* does not refer to an existing file request that is associated with the request token

2.2.6. NOTES on the Core Behavior

a) Abort all previously requested files in the request regardless of the file status. Files must be removed from the queue, and cached files must be released, when applicable.

b) If *remove* is true, then cached files must be removed.

2.2.7. NOTES on the advanced behavior with Remote Access Feature

a) *srmAbortRequestedFiles* with the *remove* flag on must be propagated for any calls to remote SRMs.

2.2.8. SEE ALSO

srmAbortRequest, srmPrepareToGet, srmPrepareToPut, srmRemoteCopy, srmLs, srmLsDetails

2.3. srmChangeFileStorageType

2.3.1. NAME

srmChangeFileStorageType - changes file storage types.

2.3.2. SYNOPSIS

In	Out
String userID	String requestToken
String authorizationID	ReturnRequestStatus {
String storageSystemInfo	EnumStatusCode statusCode,
String SURL[]	string explanation,
EnumFileStorageType desiredFileStorageType	EnumErrorCode errorCode
Int offset	} returnStatus
Int count	ReturnSURLStatuswLifetime{
String spaceToken	String SURL,
	Int newLifetime,
	EnumStatusCode statusCode,
	string explanation,
	EnumErrorCode errorCode
	} returnSURLStatus[]
	Boolean partialList
	Int totalFilesInTheRequest

2.3.3. DESCRIPTION

srmChangeFileStorageType allows clients to switch file storage type from one to another.

- String *userID* (required)
  User authentication identifier. See Section II for notes.
- String *authorizationID*
  User authorization information. The *authorizationID* may be NULL. See Section II for notes.
- String *storageSystemInfo*
  String containining information specific to the underlying storage system. The *storageSystemInfo* may be NULL. See Section II for notes.
- String *SURL*[] (required)
  *SURL*s to change the file storage type.

- EnumFileStorageType *desiredFileStorageType* (required)
  Desired file storage type that the file storage type of *SURL* is changed into.
- Int *offset*
  Integer indicator for long listed responses that the listing starts from the *offset*. Default is 0 for the first entry.
- Int *count*
  Integer indicator for long listed responses that the listing contains the number of *count*. Default is to return all entries of the list.
- String *spaceToken* (advanced option for space management feature)
  A token associated with the space if a particular space in file storage type is to be used. The *spaceToken* is acquired separately (e.g. *srmReserveSpace*).
- String *requestToken*
  Output parameter string token is associated with the request for the later asynchronous status request. *requestToken* may be NULL, in case *srmChangeFileStorageType* is processed without delay.
- ReturnRequestStatus *returnStatus* (required)
  Output parameter reporting the success or failure of the request. In case of failure, the associated *errorCode* is returned.
- ReturnSURLStatusLifetime *returnSURLStatus*[]
  Output parameter reporting the status of each file in the request. In case of failure, the associated *errorCode* is returned. *returnSURLStatus*[] may be empty and NULL. If returned to the client, *SURL* and its *statusCode* are required to return.
  - String *SURL* (required)
    SURL that client has requested to change the file storage type.
  - Int *newLifetime*
    Integer lifetime in seconds that is newly assigned to the *SURL*.
- Boolean partialList
  Output parameter indicating if return values are partial or not. If true, *requestToken* must be returned. Default is false.
- Int totalFilesInTheRequest
  Output parameter indicating how many files in the request for a hint to the asynchronous status cals.

2.3.4. RETURN CODE

On successful request, the *statusCode* is set to SRM_SUCCESS.
On failure, the *statusCode* is set to SRM_FAILURE and the *errorCode* is set.
If only partial files were successful, the *statusCode* is set to SRM_PARTIAL_SUCCESS, and the *restunSURLStatus* should explain on those failed files.

2.3.5. ERROR CODE

When status is failure, *errorCode* is set to one of the following:

For request level returnStatus,

SRM_AUTHENTICATION_FAILURE
- SRM server failed to authenticate the client

SRM_AUTHORIZATION_FAILURE

- client is not authorized to change the file types

SRM_INVALID_REQUEST
- *SURL* is empty.

SRM_INVALID_SPACE_TOKEN
- *spaceToken* does not refer to an existing space.

For file level returnStatus,

SRM_INVALID_PATH
- *SURL* does not refer to an existing file request

SRM_AUTHORIZATION_FAILURE
- client is not authorized to change the file types that is associated with the *SURL*

2.3.6. NOTES on the Core Behavior

a) *SURL* must be a reference to a local file only.

2.3.7. NOTES on the Advanced Behavior with Directory Management Feature

a) *SURL* must be applied to both directories and files.

b) If *SURL* refers to a directory, then the effect is recursive for all the files under the directory.

2.3.8. NOTES on the Advanced Behavior with Space Management Feature

a) Space allocation and de-allocation may be involved.

2.3.9. SEE ALSO

srmChangeFileStorageTypeStatus, srmPrepareToPut, srmRemoteCopy, srmExtendFileLifetime, srmReserveSpace

2.4. srmChangeFileStorageTypeStatus

2.4.1. NAME

srmChangeFileStorageTypeStatus - gets the response to the srmChangeFileStorageType.

2.4.2. SYNOPSIS

In	**Out**
String userID	ReturnRequestStatus {
String authorizationID	EnumStatusCode statusCode,
String requestToken	string explanation,
Int offset	EnumErrorCode errorCode
Int count	} returnStatus
	ReturnSURLStatuswLifetime{
	String SURL,
	Int newLifetime,
	EnumStatusCode statusCode,
	string explanation,
	EnumErrorCode errorCode
	} returnSURLStatus[]

2.4.3. DESCRIPTION

srmChangFileStorageTypeStatus() allows the asynchronous status request for the previous srmChangeFileStorageType(). *requestToken* from the response to the previous srmChangeFileStorageType() is required.

- String *userID* (required)
  User authentication identifier. See Section II for notes.
- String *authorizationID*
  User authorization information. The *authorizationID* may be NULL. See Section II for notes.
- String *requestToken* (required)
  A token associated with the previously submitted request to change file storage types. The *requestToken* was returned by the function initiating the request (e.g. *srmChangeFileStorageType*()).
- Int *offset*
  Integer indicator for long listed responses that the listing starts from the *offset*. Default is 0 for the first entry.
- Int *count*
  Integer indicator for long listed responses that the listing contains the number of *count.* Default is to return all entries of the list.
- ReturnRequestStatus *returnStatus* (required)
  Output parameter reporting the success or failure of the request. In case of failure, the associated *errorCode* is returned.
- ReturnSURLStatuswLifetime *returnSURLStatus*[]
  Output parameter reporting the status of each file in the request. In case of failure, the associated *errorCode* is returned. *returnSURLStatus*[] may be empty and NULL. If returned to the client, *SURL* and its *statusCode* are required to return.
  - String *SURL* (required)
    SURL that client has requested to change the file storage type.
  - Int *newLifetime*
    Integer lifetime in seconds if it is newly assigned to the *SURL*.

2.4.4. RETURN CODE

On successful abort, the *statusCode* is set to SRM_SUCCESS.
On failure, the *statusCode* is set to SRM_FAILURE and the *errorCode* is set.
If only partial files were successful, the *statusCode* is set to SRM_PARTIAL_SUCCESS, and the *restunSURLStatus* should explain on those failed files.

2.4.5. ERROR CODE

When status is failure, *errorCode* is set to one of the following:

For request level returnStatus,

SRM_AUTHENTICATION_FAILURE
- SRM server failed to authenticate the client

SRM_AUTHORIZATION_FAILURE

- client is not authorized to change file types
- client is not authorized to call the request specified by the *requestToken*

SRM_INVALID_REQUEST_TOKEN

- *requestToken* does not refer to an existing request

For file level returnStatus,

SRM_INVALID_PATH

- *SURL* does not refer to an existing file request

SRM_AUTHORIZATION_FAILURE

- client is not authorized to change file types of the *SURL*

2.4.6. SEE ALSO

srmChangeFileStorageType

2.5. srmExtendFileLifetime

2.5.1. NAME

srmExtendFileLifetime - allows clients to request extention of lifetime for volatile and durable files.

2.5.2. SYNOPSIS

In	Out

String userID
String authorizationID
String storageSystemInfo
String SURL
Int newLifetime
	ReturnRequestStatus {
	EnumStatusCode statusCode,
	string explanation,
	EnumErrorCode errorCode
	} returnStatus
	ReturnSURLStatuswLifetime{
	String SURL,
	Int newLifetime,
	EnumStatusCode statusCode,
	string explanation,
	EnumErrorCode errorCode
	} returnSURLStatus

2.5.3. DESCRIPTION

srmExtendFileLifetime() allows clients to extend lifetime of existing *SURL*s of volatile and durable file storage types.

- String *userID* (required)
  User authentication identifier. See Section II for notes.
- String *authorizationID*
  User authorization information. The *authorizationID* may be NULL. See Section II for notes.
- String *storageSystemInfo*

String containining information specific to the underlying storage system. The *storageSystemInfo* may be NULL. See Section II for notes.

- String *SURL* (required)
  SURL to extend the file lifetime.
- Int newLifetime
  Desired lifetime in seconds for the *SURL*. If not provided, SRM will assign default to extend.
- ReturnRequestStatus *returnStatus* (required)
  Output parameter reporting the success or failure of the request. In case of failure, the associated *errorCode* is returned.
- ReturnSURLStatuswLifetime *returnSURLStatus*
  Output parameter reporting the status of each file in the request. In case of failure, the associated *errorCode* is returned. *returnSURLStatus*[] may be empty and NULL. If returned to the client, *SURL* and its *statusCode* are required to return.
  - String *SURL* (required)
    SURL that client has requested to extend the lifetime.
  - Int *newLifetime*
    Integer lifetime in seconds that is newly assigned to the *SURL*.

2.5.4. RETURN CODE

On successful abort, the *statusCode* is set to SRM_SUCCESS.
On failure, the *statusCode* is set to SRM_FAILURE and the *errorCode* is set.

2.5.5. ERROR CODE

When status is failure, *errorCode* is set to one of the following:

For request level returnStatus,

SRM_AUTHENTICATION_FAILURE
- SRM server failed to authenticate the client

SRM_AUTHORIZATION_FAILURE
- client is not authorized to extend fiel lifetime

SRM_INVALID_REQUEST
- *SURL* is NULL.

For file level returnStatus,

SRM_INVALID_PATH
- *SURL* does not refer to an existing file request

SRM_AUTHORIZATION_FAILURE
- client is not authorized to extend fiel lifetime of *SURL*

2.5.6. NOTES on the Core Behavior

a) *newLifetime* is relative to the calling time.
b) *newLifetime* is the lifetime duration requested.
c) SRM may refuse the request, and failure error code is returned.
d) The number of lifetime extensions may be limited by SRM according to its policies.

e) If remaining lifetime is longer than the requested one, then the requested one will be assigned.
f) If *newLifetime* is not specified, the SRM may use its default to assign the *newLifetime*.
g) The life time of expired files may be extended, if the file is still in the space.
h) The life time of released files must not be extended. If needed again, it should be requested again.

2.5.7. SEE ALSO
srmExtendRequestedFileLifetime, srmPrepareToPut, srmLs

2.6. srmExtendRequestedFileLifetime

2.6.1. NAME
srmExtendRequestedFileLifetime – allows clients to request extention of expiration time on Transfer URL for the requested files.

2.6.2. SYNOPSIS

In	Out
String userID	ReturnRequestStatus {
String authorizationID	EnumStatusCode statusCode,
String storageSystemInfo	string explanation,
String requestToken	EnumErrorCode errorCode
String SURL	} returnStatus
Int newLifetime	ReturnSURLStatuswTURL{
	String SURL,
	String TURL,
	UTCtime newExpirationTime,
	EnumStatusCode statusCode,
	string explanation,
	EnumErrorCode errorCode
	} returnSURLStatus

2.6.3. DESCRIPTION
srmExtendRequestedFileLifetime allows clients to extend expiration time of TURL for those previously requested files, if TURL is still available.

- String *userID* (required)
  User authentication identifier. See Section II for notes.
- String *authorizationID*
  User authorization information. The *authorizationID* may be NULL. See Section II for notes.
- String *storageSystemInfo*
  String containining information specific to the underlying storage system. The *storageSystemInfo* may be NULL. See Section II for notes.
- String *requestToken* (required)

A token associated with the previously submitted request (e.g. *srmPrepareToGet*(), *srmStatusOfGetRequest*()).

- String *SURL* (required)
  SURL to extend the file expiration time.
- Int newLifetime
  Desired integer lifetime in seconds to extend the *SURL*. Default will be assigned by SRM if not provided. Default varies by the SRM system.
- ReturnRequestStatus *returnStatus* (required)
  Output parameter reporting the success or failure of the request. In case of failure, the associated *errorCode* is returned.
- ReturnSURLStatuswTURL *returnSURLStatus*
  Output parameter reporting the status of each file in the request. In case of failure, the associated *errorCode* is returned. *returnSURLStatus*[] may be empty and NULL. If returned to the client, *SURL, TURL* and its *statusCode* are required to return.
  - String *SURL* (required)
    SURL that client has requested to extend the expiration time.
  - UTCtime *newExpirationTime*
    New expiration time in UTCtime that is newly assigned to the *TURL* associated with the SURL.

2.6.4. RETURN CODE

On successful abort, the *statusCode* is set to SRM_SUCCESS.
On failure, the *statusCode* is set to SRM_FAILURE and the *errorCode* is set.

2.6.5. ERROR CODE

When status is failure, *errorCode* is set to one of the following:

For request level resturnStatus,

SRM_AUTHENTICATION_FAILURE
- SRM server failed to authenticate the client

SRM_AUTHORIZATION_FAILURE
- client is not authorized to extend the file expiration time on SURLs the request specified by the *requestToken*

SRM_INVALID_REQUEST_TOKEN
- *requestToken* does not refer to an existing request

SRM_INVALID_REQUEST
- *SURL* is NULL.

For file level returnStatus,

SRM_INVALID_PATH
- *SURL* does not refer to an existing file any more

SRM_AUTHORIZATION_FAILURE
- client is not authorized to extend the file expiration time on *SURL*

2.6.6. NOTES on the Core Behavior

a) *newLifetime* is relative to the calling time.
b) *newLifetime* is the lifetime duration requested.
c) SRM may refuse the request, and failure error code is returned.
d) The number of lifetime extensions may be limited by SRM according to its policies.
e) If remaining lifetime is longer than the requested one, then the requested one will be assigned.
f) If *newLifetime* is not specified, the SRM may use its default to assign the *newLifetime*.
g) The life time of expired files may be extended, if the SURL is still valid in the space.
h) The life time of released files must not be extended. If needed again, it should be requested again.

2.6.7. SEE ALSO
srmExtendFileLifetime, srmPrepareToGet, srmPrepareToPut, srmStatusOfGetRequest, srmStatusOfPutRequest

2.7. srmGetFeatures

2.7.1. NAME
srmGetFeatures - is a discovery function for clients to find out which features an SRM supports.

2.7.2. SYNOPSIS

In	Out
String userID String authorizationID	ReturnRequestStatus { EnumStatusCode statusCode, string explanation, EnumErrorCode errorCode } returnStatus EnumSRMFeatures supportedFeature[] EnumFileStorageType supportedType[] EnumRetentionQualityMode supportedQuality[] EnumAccessLatencyMode supportedLatency[]

EnumSRMFeatures := SRM_CORE |
SRM_REMOTE_ACCESS |
SRM_SPACE_MANAGEMENT |
SRM_DIRECTORY_MANAGEMENT |
SRM_AUTHORIZATION |
SRM_ADMINISTRATION |
SRM_ACCOUNTING

2.7.3. DESCRIPTION
srmGetFeatures is a discovery function for clients to find out which features an SRM supports. This function returns the list of each feature that SRM supports.

- String *userID* (required)
  User authentication identifier. See Section II for notes.

- String *authorizationID*
  User authorization information. The *authorizationID* may be NULL. See Section II for notes.
- ReturnRequestStatus *returnStatus* (required)
  Output parameter reporting the success or failure of the request. In case of failure, the associated *errorCode* is returned.
- EnumSRMFeatures *supportedFeature*[] (required)
  Output parameter returning the list of features that the SRM supports.

2.7.4. RETURN CODE
On successful abort, the *statusCode* is set to SRM_SUCCESS.
On failure, the *statusCode* is set to SRM_FAILURE and the *errorCode* is set.

2.7.5. ERROR CODE
When status is failure, *errorCode* is set to one of the following:

SRM_AUTHENTICATION_FAILURE
- SRM server failed to authenticate the client

SRM_AUTHORIZATION_FAILURE
- Client is not authorized to request SRM features

2.7.6. SEE ALSO
srmGetRequestSummary, srmGetRequestTokens, srmGetSRMStorageInfo, srmGetTransferProtocols, srmGetSpaceTokens

2.8. srmGetRequestSummary

2.8.1. NAME
srmGetRequestSummary - is to retrieve a summary of the submitted request.

2.8.2. SYNOPSIS

In	**Out**
String userID String authorizationID String requestToken	ReturnRequestStatus { EnumStatusCode statusCode, string explanation, EnumErrorCode errorCode } returnStatus EnumRequestType requestType Int totalFilesInTheRequest Int numFilesQueued Int numFilesReleased Int numFilesExpired Int numFilesFailed Int numFilesInProgress Boolean suspended

2.8.3. DESCRIPTION

srmGetRequestSummary is to retrieve a summary of the previously submitted request.

- String *userID* (required)
  User authentication identifier. See Section II for notes.
- String *authorizationID*
  User authorization information. The *authorizationID* may be NULL. See Section II for notes.
- String *requestToken* (required)
  A token associated with the previously submitted request. The *requestToken* was returned by the function initiating the request (e.g. *srmPrepareToGet*()).
- ReturnRequestStatus *returnStatus* (required)
  Output parameter reporting the success or failure of the request. In case of failure, the associated *errorCode* is returned.
- EnumRequestType requestType
  Output parameter reporting the type of the request (e.g. PREPARE_TO_GET)
- Int totalFilesInTheRequest
  Output parameter reporting the total number of files in the request
- Int numFilesQueued
  Output parameter reporting the number of queued files in the request
- Int numFilesReleased
  Output parameter reporting the number of released files in the request
- Int numFilesExpired
  Output parameter reporting the number of expired files in the request
- Int numFilesFailed
  Output parameter reporting the number of failed files in the request. It refers to failure to get the file such as file not found or file transfer failed.
- Int numFilesInProgress
  Output parameter reporting the number of files in progress in the request. It refers to files that are in transfer, or in cache but not released yet.
- Boolean suspended (advanced option)
  An advanced option when Request Administration feature is supported. Output parameter reporting if the rquest has been suspended or not

2.8.4. RETURN CODE

On successful abort, the *statusCode* is set to SRM_SUCCESS.
On failure, the *statusCode* is set to SRM_FAILURE and the *errorCode* is set.
If only partial files were successful, the *statusCode* is set to SRM_PARTIAL_SUCCESS, and the *explanation* should explain what it is. Upon receiving SRM_PARTIAL_SUCCESS, client may submit status request for the request.

2.8.5. ERROR CODE

When status is failure, *errorCode* is set to one of the following:

SRM_AUTHENTICATION_FAILURE
- SRM server failed to authenticate the client

SRM_AUTHORIZATION_FAILURE

- client is not authorized to get summary of the request specified by the *requestToken*

SRM_INVALID_REQUEST_TOKEN

- *requestToken* does not refer to an existing request

2.8.6. SEE ALSO

srmGetRequestTokens, srmStatusOfGetRequest, srmStatusOfPutRequest, srmStatusOfRemoteCopyRequest, srmGetSpaceTokens.

2.9. srmGetRequestTokens

2.9.1. NAME

srmGetRequestTokens - retrieves request tokens for the previously submitted requests by the client .

2.9.2. SYNOPSIS

In	Out

String userID
String authorizationID
String userRequestDescription
	ReturnRequestStatus {
	EnumStatusCode statusCode,
	string explanation,
	EnumErrorCode errorCode
	} returnStatus
	ReturnToken {
	String requestToken,
	EnumRequestType requestType,
	UTCtime submittedTime
	} returnTokens[]

2.9.3. DESCRIPTION

srmGetRequestTokens retrieves request tokens for the client's requests, given client provided request description. This is to accommodate lost request tokens. This can also be used for getting all request tokens.

- String *userID* (required)
  User authentication identifier. See Section II for notes.
- String *authorizationID*
  User authorization information. The *authorizationID* may be NULL. See Section II for notes.
- String *userRequestDescription*
  String containining description of the request. The description was given by the client at the time of the request (e.g. *srmPrepareToGet*). The *userRequestDescription* may be NULL.
- ReturnRequestStatus *returnStatus* (required)
  Output parameter reporting the success or failure of the request. In case of failure, the associated *errorCode* is returned.
- ReturnToken *returnTokens*[]

Output parameter returning the request tokens owned by the client. *returnTokens*[] may be NULL and empty. If returned to the client, *requestToken* and *requestType* are required to be filled.

2.9.4. RETURN CODE

On successful abort, the *statusCode* is set to SRM_SUCCESS.
On failure, the *statusCode* is set to SRM_FAILURE and the *errorCode* is set.

2.9.5. ERROR CODE

When status is failure, *errorCode* is set to one of the following:

SRM_AUTHENTICATION_FAILURE
- SRM server failed to authenticate the client

SRM_AUTHORIZATION_FAILURE
- client is notauthorized to get request tokens specified by the userRequestDescription

SRM_INVALID_REQUEST
- *userRequestDescription* does not refer to any existing requests

2.9.6. NOTES on Core Behavior

a) If *userRequestDescription* is not provided, all requests that belong to the client are returned.
b) If the client assigned the same request description to multiple requests, multiple request tokens each with the time the request was made are returned.

2.9.7. SEE ALSO

srmPrepareToGet, srmPrepareToPut, srmRemoteCopy

2.10. srmGetSRMStorageInfo

2.10.1. NAME

srmGetSRMStorageInfo - retrieves SRM storage information, such as storage capacity, client quota, default lifetime, etc.

2.10.2. SYNOPSIS

In	**Out**

String userID
String authorizationID
EnumStorageAttributes desiredAttributes[]
	ReturnRequestStatus {
	EnumStatusCode statusCode,
	string explanation,
	EnumErrorCode errorCode
	} returnStatus
	SupportedAttributes {
	EnumStorageAttributes storageAttr,
	String value,
	String valueType
	} storageInfo[]

EnumStorageAttributes := SRM_FILE_STORAGE_TYPE |
SRM_STORAGE_CAPACITY |
SRM_STORAGE_ACCESS_LATENCY_TYPE |
SRM_USER_STORAGE_MAX |
SRM_USER_STORAGE_MIN |
SRM_USER_STORAGE_DEFAULT_LIFETIME |
SRM_DEFAULT_FILE_LIFETIME |
SRM_DEFAULT_FILE_STORAGE_TYPE |
SRM_DEFAULT_TURL_EXPIRATION_TIME |
SRM_REQUEST_INFO_AVAILABILITY |
SRM_DEFAULT_RETURN_COUNT

2.10.3. DESCRIPTION

srmGetSRMStorageInfo retrieves SRM storage information, such as storage capacity, client quota, default lifetime, etc.

- ReturnRequestStatus *returnStatus* (required)
  Output parameter reporting the success or failure of the request. In case of failure, the associated *errorCode* is returned.
- SupportedAttributes *storageInfo*[]
  Output parameter reporting the storage information. *storageInfo*[] may be empty and NULL. If returned to the client, *storageAttr* and its *value* are required to return.
  - EnumStorageAttributes *storageAttr* (required)
    One of EnumStorageAttributes that client has requested to find out the storage system info.
  - String *value*
    A value of the *storageAttr*.
  - String *valueType*
    Data type of the value for storageAttr. For example, literal int, long, string, boolean.

2.10.4. RETURN CODE

On successful abort, the *statusCode* is set to SRM_SUCCESS.
On failure, the *statusCode* is set to SRM_FAILURE and the *errorCode* is set.

2.10.5. ERROR CODE

When status is failure, *errorCode* is set to one of the following:

SRM_AUTHENTICATION_FAILURE
- SRM server failed to authenticate the client

SRM_AUTHORIZATION_FAILURE
- client is not authorized to request storage information

SRM_NOT_SUPPORTED
- *storageAttr* is not supported in the SRM

2.10.6. NOTES on the Core Behavior

a) SRM_REQUEST_INFO_AVAILABILITY describes how long SRM will keep the status of the request after completion or termination.

b) SRM_DEFAULT_RETURN_COUNT shows the default number of entries returned starting from the first available entry in case the result is too large.

2.10.7. SEE ALSO

srmGetTransferProtocol, srmGetFeatures

2.11. srmGetTransferProtocols

2.11.1. NAME

srmGetTransferProtocols - returns the supported file transfer protocols in the SRM.

2.11.2. SYNOPSIS

In	Out

String userID
String authorizationID
	ReturnRequestStatus {
	EnumStatusCode statusCode,
	string explanation,
	EnumErrorCode errorCode
	} returnStatus
	SupportedTransferPrototols {
	String transferProtocol,
	String attributes
	} transferProtocols[]

2.11.3. DESCRIPTION

srmGetTransferProtocols() returns the supported file transfer protocols in the SRM.

- ReturnRequestStatus *returnStatus* (required)
  Output parameter reporting the success or failure of the request. In case of failure, the associated *errorCode* is returned.
- SupportedTransferPrototols *transferProtocols*[]
  Output parameter reporting the supported file transfer protocol list.
  - String *transferProtocol* (required)
    Supported transfer protocol. For example, http.
  - String *attributes*
    Informational hints for the paired transfer protocol, such how many number of parallel streams can be used, desired buffer size, etc. Recommended to use the key/value pairs as a string list, separated by comma (,).

2.11.4. RETURN CODE

On successful abort, the *statusCode* is set to SRM_SUCCESS.
On failure, the *statusCode* is set to SRM_FAILURE and the *errorCode* is set.

2.11.5. ERROR CODE

When status is failure, *errorCode* is set to one of the following:

SRM_AUTHENTICATION_FAILURE
- SRM server failed to authenticate the client

SRM_AUTHORIZATION_FAILURE
- client is not authorized to get the list of the file transfer protocols that SRM server supports

2.11.6. SEE ALSO
srmGetSRMStorageInfo, srmGetFeatures

2.12. srmLs

2.12.1. NAME
srmLs - returns a list of files with a basic information.

2.12.2. SYNOPSIS

In	Out

String userID
String authorizationID
String storageSystemInfo
{ String SURL[] |
EnumFileStorageType fileStorageType |
String spaceToken }
Boolean directoriesOnly
Boolean allLevelRecursive
Int numOfLevels
Int offset
Int count
	String requestToken
	ReturnRequestStatus {
	EnumStatusCode statusCode,
	string explanation,
	EnumErrorCode errorCode
	} returnStatus
	SURLMetaData {
	String SURL,
	EnumStatusCode statusCode,
	String explanation,
	EnumErrorCode errorCode,
	Int size,
	UTCtime expirationTime,
	EnumFileType fileType,
	EnumAccessLatencyMode latencyMode,
	String spaceToken,
	SURLMetaData subpath[]
	} returnSURLInfo[]
	Boolean partialList
	Int totalFilesInTheRequest

2.12.3. DESCRIPTION
srmLs() returns a list of files with a basic information.

- String *userID* (required)
  User authentication identifier. See Section II for notes.
- String *authorizationID*

User authorization information. The *authorizationID* may be NULL. See Section II for notes.

- String *storageSystemInfo*
  String containining information specific to the underlying storage system. The *storageSystemInfo* may be NULL. See Section II for notes.
- String *SURL*[], or
  EnumFileStorageType *fileStorageType*, or
  String *spaceToken* (advanced option when Space Management feature is supported)
  Only one of these three parameters are required. *SURL* refers to files only as the core function. As an advanced behavior, *SURL* may refer to directories as well when Directory Management Feature is supported.
- Boolean *directoriesOnly*
  An advanced option when Directory Management feature is supported. Boolean indicator for directory listing only. Default is false.
- Boolean *allLevelRecursive*
  An advanced option when Directory Management feature is supported. Boolean indicator for recursive directory listing. Default is false.
- Int *numOfLevels*
  An advanced option when Directory Management feature is supported. Positive integer indicator for how many levels of the recursive directory listing to be performed. Default is 0 for the single level of directory listing (no recursive).
- Int *offset*
  Integer indicator for long listed responses that the listing starts from the *offset*. Default is 0 for the first entry.
- Int *count*
  Integer indicator for long listed responses that the listing contains the number of *count.* Default 0 (zero) indicates to return all entries of the list.
- String *requestToken*
  Output parameter string token is associated with the request for the later asynchronous status request. *requestToken* may be NULL, in case *srmLs* is processed without delay.
- ReturnRequestStatus *returnStatus* (required)
  Output parameter reporting the success or failure of the request. In case of failure, the associated *errorCode* is returned.
- SURLMetaData *returnSURLInfo*[]
  Output parameter reporting the information of each file in the request. In case of failure, the associated *errorCode* is returned. *returnSURLInfo*[] may be empty and NULL. If returned to the client, *SURL, size,* and its *fileType* are required to return.
  - String *SURL* (required)
    SURL that client has requested for the info.
  - EnumStatusCode *statusCode*
    Status code of the *SURL*.
  - String *explanation*
    String explanation of the *statusCode* for the *SURL* in case of not successful.
  - EnumErrorCode *errorCode*
    In case of failure, *errorCode* is expected to be returned for the *SURL*.
  - Int *size* (required)

Size of the *SURL*. In case of *SURL* being a directory for advanced features, *size* is expected to be 0.

- UTCtime *expirationTime*
  Expiration time that is associated with the *SURL*.
- EnumFileType *fileType* (required)
  File type associated with the *SURL*.
- EnumAccessLatencyMode *latencyMode*
  Access latency mode that is associated with the *SURL*.
- String *spaceToken* (advanced option)
  An advanced option when Space Management feature is supported. *spaceToken* is associated with the *SURL*.
- SURLMetaData *subpath*[]
  An advanced option when Directory Management feature is supported. In case of the recursive directory *SURL* listing, *subpath* indicates sub-directory that contains files or directoties.

- Boolean *partialList*
  Output parameter reporting if the returned *returnSURLInfo*[] is a partial list. Default is false. If the *returnSURLList*[] is a partial list and *partialList* is true, *requestToken* is required to be returned, and client is expected to request its status again with *srmLsStatus*().
- Int totalFilesInTheRequest
  Output parameter indicating how many files in the request for a hint to the asynchronous status cals.

2.12.4. RETURN CODE

On successful abort, the *statusCode* is set to SRM_SUCCESS.
On failure, the *statusCode* is set to SRM_FAILURE and the *errorCode* is set.
If only partial files were successful, the *statusCode* is set to SRM_PARTIAL_SUCCESS, and the *restunSURLInfo* should explain on those failed files.

2.12.5. ERROR CODE

When status is failure, *errorCode* is set to one of the following:

For request level resturnStatus,

SRM_AUTHENTICATION_FAILURE
- SRM server failed to authenticate the client

SRM_AUTHORIZATION_FAILURE
- client is not authorized to request information

SRM_INVALID_SPACE_TOKEN
- *spaceToken* does not refer to an existing space in SRM

SRM_TOO_MANY_RESULTS
- srmLs request has generated too many results that SRM cannot handle. In most cases, it needs to be narrowed down with offset and count by the client.

SRM_NOT_SUPPORTED
- Requested *fileStorageType* is not supported in SRM
- Directory operation is not supported in SRM

For file level returnStatus,

SRM_INVALID_PATH

- *SURL* does not refer to an existing file path

SRM_AUTHORIZATION_FAILURE

- client is not authorized to view the *SURL* or to access the directory or sub-directories

2.12.6. NOTES on the Core Behavior

a) Either *SURLs*, *spaceToken* or *fileStorageType* is needed.
b) *SURL* refers to files only.
c) If output parameter *partialList* is true, it indicates that only part of the result was returned. In this case, a *requestToken* must be returned.
d) If the entire result is returned, then output parameter *requestToken* is optional.
e) *statusCode* for the *SURL* in *returnSURLInfo* is recommended to be for the client. For example, another client requested the same *SURL* and the file is pinned in the SRM cache. The SRM must not return *SRM_FILE_PINNED* for a client requesting *srmLs* on the same *SURL* unless the client requested the *SURL* previously. Instead *SRM_FILE_IN_CACHE* may be appropriate.
f) When listing for a particular type specified by *fileStorageType*, all the files of that type must be returned.

2.12.7. NOTES on the Advanced Behavior with Directory Management Feature

a) *SURL*s may refer to either directory or file.
b) Files are returned in width first order.
c) List of directories come before list of files in the return order.
d) If *SURL* refers to a directory, the returned value *size* must be 0.
e) If *allLevelRecursive* is "true", it dominates, i.e. ignore *numOfLevels*.
f) If *allLevelRecursive* is "false" or missing, then *numOfLevels* must be honored. If *numOfLevels* is "0" (zero), a single level is assumed.
g) Directory names are returned even if they are empty.

2.12.8. NOTES on the Advanced Behavior with Space Management Feature

a) If *spaceToken* is provided, all files that belong to the *spaceToken* are returned.
b) If *spaceToken* is not provided and *fileStorageType* is provided, then all the files with *fileStorageType* must be returned.

2.12.9. SEE ALSO

srmLsStatus, srmLsDetails, srmLsDetailsStatus, srmGetSpaceMetaData, srmGetSpaceTokens

2.13. srmLsStatus

2.13.1. NAME

srmLsStatus - is an asynchronous call for srmLs that is large.

2.13.2. SYNOPSIS

In	Out

String userID
String authorizationID
String requestToken
Int offset
Int count
	ReturnRequestStatus {
	EnumStatusCode statusCode,
	string explanation,
	EnumErrorCode errorCode
	} returnStatus
	SURLMetaData {
	String SURL,
	EnumStatusCode statusCode,
	String explanation,
	EnumErrorCode errorCode,
	Int size,
	UTCtime expirationTime,
	EnumFileType fileType,
	EnumAccessLatencyMode latencyMode,
	String spaceToken,
	SURLMetaData subpath[]
	} returnSURLInfo[]

2.13.3. DESCRIPTION

srmLsStatus() retrieves the status of the previously requested srmLs. See also 2.12.3 for more descriptions.

- String *requestToken* (required)
  A token associated with the previously submitted srmLs request. The requestToken was returned by the function initiating the request (e.g. *srmLs*()).
- Int *offset*
  Integer indicator for long listed responses that the listing starts from the *offset*. Default is 0 for the first entry.
- Int *count*
  Integer indicator for long listed responses that the listing contains the number of *count.* Default 0 (zero) indicates to return all entries of the list.
- ReturnRequestStatus *returnStatus* (required)
  Output parameter reporting the success or failure of the request. In case of failure, the associated *errorCode* is returned.
- SURLMetaData *returnSURLInfo*[]
  Output parameter reporting the information of each file in the request. In case of failure, the associated *errorCode* is returned. *returnSURLInfo*[] may be empty and NULL. If returned to the client, *SURL, size,* and its *fileType* are required to return.
  - String *SURL* (required)
    SURL that client has requested for the info.
  - EnumStatusCode *statusCode*
    Status code of the *SURL*.
  - String *explanation*
    String explanation of the *statusCode* for the *SURL* in case of not successful.
  - EnumErrorCode *errorCode*

In case of failure, *errorCode* is expected to be returned for the *SURL*.

- Int *size* (required)
  Size of the *SURL*. In case of *SURL* being a directory for advanced features, *size* is expected to be 0.
- UTCtime *expirationTime*
  Expiration time that is associated with the *SURL*.
- EnumFileType *fileType* (required)
  File type associated with the *SURL*.
- EnumAccessLatencyMode *latencyMode*
  Access latency mode that is associated with the *SURL*.
- String *spaceToken* (advanced option)
  An advanced option when Space Management feature is supported. *spaceToken* is associated with the *SURL*.
- SURLMetaData *subpath*[] (advanced option)
  An advanced option when Directory Management feature is supported. In case of the recursive directory *SURL* listing, *subpath* indicates sub-directory that contains files or directoties.

2.13.4. RETURN CODE

On successful abort, the *statusCode* is set to SRM_SUCCESS. On failure, the *statusCode* is set to SRM_FAILURE and the *errorCode* is set. See also 2.12.4.

2.13.5. ERROR CODE

When status is failure, *errorCode* is set to one of the following:

SRM_AUTHENTICATION_FAILURE
- SRM server failed to authenticate the client

SRM_AUTHORIZATION_FAILURE
- client is not authorized to get the status the request specified by the *requestToken*

SRM_INVALID_REQUEST_TOKEN
- *requestToken* does not refer to an existing request

See also 2.12.5.

2.13.6. SEE ALSO

srmLs, srmLsDetails, srmLsDetailsStatus, srmGetSpaceMetaData, srmGetRequestTokens, srmGetSpaceTokens

2.14. srmPrepareToGet

2.14.1. NAME

srmPrepareToGet - gets files from the local space of SRM.

2.14.2. SYNOPSIS

In	Out
String userID	String requestToken

String authorizationID
String transferProtocols[]
String userRequestDescription
Int totalRetryTime
Boolean streaming
EnumOverwriteMode overwriteOption,
GetFileRequest {
String fromSURL,
int desiredLifetime,
String fromStorageSystemInfo,
Int knownSizeOfFile,
Boolean isSourceADirectory,
Boolean allLevelRecursive,
int numOfLevels
} getRequests[]
Int desiredLifetime,
EnumFileStorageType fromFileStorageType,
String fromStorageSystemInfo,
Boolean performChecksum

ReturnRequestStatus {
EnumStatusCode statusCode,
string explanation,
EnumErrorCode errorCode
} returnStatus

2.14.3. DESCRIPTION

srmPrepareToGet() allows clients to retrieve files from the local space of SRM.

- String *userID* (required)
  User authentication identifier. See Section II for notes.
- String *authorizationID*
  User authorization information. The *authorizationID* may be NULL. See Section II for notes.
- String *transferProtocols*[]
  List of the transfer protocols that client can support.
- String *userRequestDescription*
  String description of the request. It may be used for later retrieval of the rquest token.
- Int *totalRetryTime*
  totalRetryTime is the amount of time in seconds that SRM should try to transfer failed files, if some file transfers fail. Default is 0 that SRM assigns by default.
- Boolean *streaming*
  Boolean indication of *streaming* mode. When *streaming* mode is on, full space does not have to be prepared to hold all files in the request. Default is false.
- EnumOverwriteMode *overwriteOption*
  In case SRM stages requested files for the user, SRM needs to stage the file according to the *overwriteOption* on the target that SRM brings the files into.
- GetFileRequest *getRequests*[] (required)
  Input parameter g*etRequest* consists of SURL information that client wants to retrieve.
  - String *fromSURL* (required)
    SURL that client desires to retrieve
  - Int *desiredLifetime*

Desired life time of the file in seconds once when the file is ready. Default is 0, and SRM assigns default lifetime. In the status call, expiration time in UTCtime on Transfer URL is returned.

- String *fromStorageSystemInfo*
  String containining information specific to the underlying storage system associated with the *fromSURL*. The *fromStorageSystemInfo* may be NULL. See Section II for notes
- Int *knownSizeOfFile*
  File size of the SURL if known. Default is 0, indicating unknown.
- Boolean *isSourceADirectory*
  An advanced option when Directory Management feature is supported. Boolean indicator if SURL is a directory. Default is false.
- Boolean *allLevelRecursive*
  An advanced option when Directory Management feature is supported. Boolean indicator if SURL is a directory and all files under the SURL must be retrieved. Defautl is false.
- Int *numOfLevels*
  An advanced option when Directory Management feature is supported. Positive integer indicator for how many levels of the recursive directory must be browsed and all files in those directories to be retrieved. Default is 0 for the single level of directory (no recursive).

- Int *desiredLifetime*
  Desired life time of the file in seconds once when the file is ready. Default is 0, and SRM assigns default lifetime. This is request level option. If file level option (*desiredLifetime*) in the *getRequests* is set, it takes the priority.
- EnumFileStorageType *fromFileStorageType*
  fromFileStorageType indicates which type of *fromSURL*s are.
- String *fromStorageSystemInfo*
  String containining information specific to the underlying storage system that is associated with the source SURLs or *fromFileStorageType*. The *fromStorageSystemInfo* may be NULL. See Section II for notes.
- Boolean *performChecksum*
  performChecksum indicates that checksum calculation for all files in the request needs to be performed when SRM server brings files into its space. Default is false.
- String *requestToken* (required)
  Output parameter string token is associated with the request for the later asynchronous status request.
- ReturnRequestStatus *returnStatus* (required)
  Output parameter reporting the success or failure of the request. In case of failure, the associated *errorCode* is returned.

2.14.4. RETURN CODE

On successful abort, the *statusCode* is set to SRM_SUCCESS.
On failure, the *statusCode* is set to SRM_FAILURE and the *errorCode* is set.

2.14.5. ERROR CODE

When status is failure, *errorCode* is set to one of the following:

SRM_AUTHENTICATION_FAILURE
- SRM server failed to authenticate the client

SRM_AUTHORIZATION_FAILURE
- client is not authorized to submit the request

SRM_INVALID_REQUEST
- input parameters do not conform the SRM. For example, client requested negative *totalRetryTime* and SRM cannot honor the number.

SRM_NOT_SUPPORTED
- SRM server does not support the given input parameters. For example, client requested bbftp for the only transfer protocol, but SRM cannot support that.

2.14.6. NOTES on the Core Behavior

a) This is an asynchronous (non-blocking) call. To get status and results, separate calls may be made with the *requestToken*.

b) *fromSURL* must be local to SRM.

c) The *userRequestDescription* is a client designated name for the request. It can be used in the *srmGetRequestToken*() function to get back the request tokens for requests made by the client.

d) SRM assigns the *requestToken* for the request.

e) *fromStorageSystemInfo*, *fromFileStorageType* or *desiredLifetime* may be provided at the request level and the file level. In that case, SRM uses the one provided at the file level.

f) After the file is brought into the cache, it is pinned, the lifetime clock starts at that time, and the expiration time on Transfer URL gets returned.

g) When the *streaming* option is true, and only part of the files fit into the space, the SRM wait until files are released and continues to bring files in. When the *streaming* option is false, and only part of the files fit into the space, SRM must return failure for the request.

h) If some file transfers fail, and all the other files transferred successfully, then *totalRetryTime* is the amount of time in seconds that SRM should try to transfer failed files.

i) In case that the retries fail, the return values in status call include an explanation of why the retries failed.

j) *srmReleaseFile*() or *srmReleaseRequestedFiles*() is expected for files that are no longer needed. Otherwise, file lifetime expires. If lifetime of all the files in the space expires, then the request is terminated and the status of the request is set to SRM_REQUEST_FINISHED.

2.14.7. NOTES on the Advanced Behavior with Directory Management Feature

a) *numOfLevels* must be a valid input parameter, and must be a positive integer. Default is 0 for the single level of directory (no recursive).

b) The default value for *allLevelResursive* is false.

2.14.8. SEE ALSO

srmStatusOfGetRequest, srmPrepareToPut, srmRemoteCopy

2.15. srmPrepareToPut

2.15.1. NAME

srmPrepareToPut - allows SRM to prepare local space for files in the request, so that the client may push files to the allocated space. It allocates a local space, and return TransferURL (TURL) for each file to the client.

2.15.2. SYNOPSIS

In	Out
String userID	String requestToken
String authorizationID	ReturnRequestStatus {
String transferProtocols[]	EnumStatusCode statusCode,
String userRequestDescription	string explanation,
Boolean streaming	EnumErrorCode errorCode
EnumOverwriteMode overwriteOption	} returnStatus
PutFileRequest {	
String toSURL,	
int desiredLifetime,	
EnumFileStorageType toFileStorageType,	
String toStorageSystemInfo,	
Int knownSizeOfFile	
String spaceToken	
} putRequests[]	
Int desiredLifetime,	
EnumFileStorageType toFileStorageType	
String toStorageSystemInfo	
Boolean performChecksum	
String spaceToken	

2.15.3. DESCRIPTION

srmPrepareToPut allows SRM to prepare local space for files in the request, so that the client may push files to the allocated space. It allocates a local space, and return TransferURL (TURL) for each file to the client.

- String *userID* (required)
  User authentication identifier. See Section II for notes.
- String *authorizationID*
  User authorization information. The *authorizationID* may be NULL. See Section II for notes.
- String *transferProtocols*[]
  List of the transfer protocols that client can support.
- String *userRequestDescription*
  String description of the request. It may be used for later retrieval of the rquest token.
- Int *totalRetryTime*
  totalRetryTime is the amount of time in seconds that SRM should try to transfer failed files, if some file transfers fail. Default is 0 that SRM assigns by default.
- Boolean *streaming*

Boolean indication of *streaming* mode. When *streaming* mode is on, full space does not have to be prepared to hold all files in the request. Default is false.

- EnumOverwriteMode *overwriteOption*
  SRM prepares the space for a requested file according to the *overwriteOption*.
- PutFileRequest *putRequests*[]
  Input parameter *putRequests* consists of SURL information that client wants to store. If NULL, SRM will allocate a space for one file with a default life time in a default storage space with a default allocated size.
  - String *toSURL*
    SURL that client targets to store a file
  - Int *desiredLifetime*
    Desired life time of the file in seconds once when the file is ready. Default is 0, and SRM assigns default lifetime.
  - String *toStorageSystemInfo*
    String containining information specific to the underlying storage system associated with the *toSURL* or *toFileStorageType*. The *toStorageSystemInfo* may be NULL. See Section II for notes
  - Int *knownSizeOfFile*
    Desired File size if known. Default is 0, indicating unknown. In such case, SRM assigns a default space allocation for each file request.
  - String *spaceToken*
    An advanced option when Space Management feature is supported. SRM will use the specific space when allocating space for files in the request .
- Int *desiredLifetime*
  Desired life time of the file in seconds once when the fiel is ready. Default is 0, and SRM assigns default lifetime. This is request level option. If file level option (*desiredLifetime*) in the *putRequests* is set, it takes the priority.
- EnumFileStorageType *toFileStorageType*
  toFileStorageType indicates which type of space allocation needs to be performed by SRM server.
- String *toStorageSystemInfo*
  String containining information specific to the underlying storage system that is associated with the *toSURL* or *toFileStorageType*. The *toStorageSystemInfo* may be NULL. See Section II for notes.
- Boolean *performChecksum*
  performChecksum indicates that checksum calculation for all files in the request needs to be performed when files gets into its designated space. Default is false.
- String *spaceToken*
  An advanced option when Space Management feature is supported. SRM will use the specific space when allocating space for files in the request. This is request level option. If file level option (*spaceToken*) in the *putRequests* is set, it takes the priority.
- String *requestToken* (required)
  Output parameter string token is associated with the request for the later asynchronous status request.
- ReturnRequestStatus *returnStatus* (required)

Output parameter reporting the success or failure of the request. In case of failure, the associated *errorCode* is returned.

2.15.4. RETURN CODE

On successful abort, the *statusCode* is set to SRM_SUCCESS.
On failure, the *statusCode* is set to SRM_FAILURE and the *errorCode* is set.

2.15.5. ERROR CODE

When status is failure, *errorCode* is set to one of the following:

SRM_AUTHENTICATION_FAILURE
- SRM server failed to authenticate the client

SRM_AUTHORIZATION_FAILURE
- client is not authorized to submit the request

SRM_INVALID_REQUEST
- input parameters do not conform the SRM. For example, *knownSizeOfFile* is negative integer and SRM cannot honor the number.

SRM_NOT_SUPPORTED
- SRM server does not support the given input parameters. For example, client requested bbftp for the only transfer protocol, but SRM cannot support that.

2.15.6. NOTES on the Core Behavior

a) *toSURL*s must be limited to the local files.
b) Only push mode from the client to SRM is supported.
c) When the input parameter *toSURL* is not specified, the SRM names it automatically. The SURL is returned as *referenceSURL* along with the *transferURL* (*TURL*) in the *srmStatusOfPutRequest*() call.
d) *srmPutFileDone*() is expected from the client after each file is "put" into the allocated space. The *srmPutRequestDone*() tells the SRM that the *srmPrepareToPut*() is completed.
e) When the *streaming* option is true, and the allocated space is full, the SRM waits until files in the space are released, and then continues. When the *streaming* flag is false, and if there is not enough space to accommodate the entire request, the SRM returns a failure code.
f) The lifetime of the file starts as soon as the SRM receives *srmPutFileDone*() or *srmPutRequestDone*(). If *srmPutFileDone*() or *srmPutRequestDone*() is not provided, then the files in the space are subject to removal when the space lifetime expires or when the expiration time on Transfer URL is reached.
g) *toStorageSystemInfo*, *toFileStorageType* or *desiredLifetime* may be provided at the request level and the file level. In that case, SRM uses the one provided at the file level.

2.15.7. NOTES on the Advanced Behavior with Space Management Feature

a) If *spaceToken* is provided at the request level and the file level, the SRM uses the one provided at the file level.
b) The default value of "lifetime" for volatile or durable files must be equal to or less than the lifetime left in the space of the corresponding file type. The default value of fileType is "volatile".

2.15.8. SEE ALSO
srmPrepareToGet, srmStatusOfPutRequest, srmRemoteCopy

2.16. srmPutFileDone

2.16.1. NAME
srmPutFileDone - is used to notify the SRM that the client completed a file transfer to the TransferURL in the allocated space. This call should normally follow srmPrepareToPut.

2.16.2. SYNOPSIS

In	Out	
String userID	ReturnRequestStatus {	
String authorizationID	EnumStatusCode	statusCode,
String storageSystemInfo	string	explanation,
String requestToken	EnumErrorCode	errorCode
String SURL[]	} returnStatus	
	ReturnSURLStatus {	
	String	SURL,
	EnumStatusCode	statusCode,
	string	explanation,
	EnumErrorCode	errorCode
	} returnSURLStatus[]	

2.16.3. DESCRIPTION
srmPutFileDone() is used to notify the SRM that the client completed a file transfer to the TransferURL in the allocated space. This call should normally follow srmPrepareToPut.

- String *userID* (required)
  User authentication identifier. See Section II for notes.
- String *authorizationID*
  User authorization information. The *authorizationID* may be NULL. See Section II for notes.
- String *storageSystemInfo*
  String containining information specific to the underlying storage system. The *storageSystemInfo* may be NULL. See Section II for notes.
- String *requestToken* (required)
  A token associated with the previously submitted request for *srmPrepareToPut*. The requestToken was returned by the function initiating the request.
- String *SURL*[] (required)
  SURLs to indicate the completed file transfer initiated by the client.

2.16.4. RETURN CODE
On successful abort, the *statusCode* is set to SRM_SUCCESS.
On failure, the *statusCode* is set to SRM_FAILURE and the *errorCode* is set.
If only partial files were successful, the *statusCode* is set to SRM_PARTIAL_SUCCESS, and the *restunSURLStatus* should explain on those failed files.

2.16.5. ERROR CODE

When status is failure, *errorCode* is set to one of the following:

For request level resturnStatus,

SRM_AUTHENTICATION_FAILURE
- SRM server failed to authenticate the client

SRM_AUTHORIZATION_FAILURE
- client is not authorized to call the request specified by the *requestToken*

SRM_INVALID_REQUEST_TOKEN
- *requestToken* does not refer to an existing request

SRM_INVALID_REQUEST
- *SURL* is empty.

For file level returnStatus,

SRM_INVALID_PATH
- *SURL* does not refer to an existing file request

SRM_AUTHORIZATION_FAILURE
- client is not authorized to call the request *srmPutFileDone*() on the *SURL*

2.16.6. SEE ALSO

srmPrepareToPut, srmPutRequestDone, srmStatusOfPutRequest

2.17. srmPutRequestDone

2.17.1. NAME

srmPutRequestDone - is used to notify the SRM that the client completed all file transfers in the request. All file transfers associated with the requestToken are considered to be completed. If srmPutFileDone is issued for all files, this call is not needed.

2.17.2. SYNOPSIS

In	Out
String userID	ReturnRequestStatus {
String authorizationID	EnumStatusCode statusCode,
String storageSystemInfo	string explanation,
String requestToken	EnumErrorCode errorCode
	} returnStatus
	ReturnSURLStatus {
	String SURL,
	EnumStatusCode statusCode,
	string explanation,
	EnumErrorCode errorCode
	} returnSURLStatus[]

2.17.3. DESCRIPTION

srmPutRequestDone is used to notify the SRM that the client completed all file transfers in the request. All file transfers associated with the *requestToken* are considered to be completed. If srmPutFileDone is issued for all files, this call is not needed. See also 2.16.3 for srmPutFileDone.

- String *userID* (required)
  User authentication identifier. See Section II for notes.
- String *authorizationID*
  User authorization information. The *authorizationID* may be NULL. See Section II for notes.
- String *storageSystemInfo*
  String containining information specific to the underlying storage system. The *storageSystemInfo* may be NULL. See Section II for notes.
- String *requestToken* (required)
  A token associated with the previously submitted request for *srmPrepareToPut*. The *requestToken* was returned by the function initiating the request.

2.17.4. RETURN CODE
On successful abort, the *statusCode* is set to SRM_SUCCESS.
On failure, the *statusCode* is set to SRM_FAILURE and the *errorCode* is set.
If only partial files were successful, the *statusCode* is set to SRM_PARTIAL_SUCCESS, and the *restunSURLStatus* should explain on those failed files.

2.17.5. ERROR CODE
When status is failure, *errorCode* is set to one of the following:

SRM_AUTHENTICATION_FAILURE
- SRM server failed to authenticate the client

SRM_AUTHORIZATION_FAILURE
- client is not authorized to call the request specified by the *requestToken*

SRM_INVALID_REQUEST_TOKEN
- *requestToken* does not refer to an existing request

2.17.6. SEE ALSO
srmPrepareToPut, srmPutFileDone, srmStatusOfPutRequest

2.18. srmReleaseFiles

2.18.1. NAME
srmReleaseFiles - releases the files in a space. The file is not removed, but the space occupied by the released file is eligible for removal if space is needed.

2.18.2. SYNOPSIS

In	Out

String userID
String authorizationID
String storageSystemInfo
String SURL[]
Boolean releaseAllCurrentlyPinnedFiles
	ReturnRequestStatus {
	EnumStatusCode statusCode,
	string explanation,
	EnumErrorCode errorCode
	} returnStatus
	ReturnSURLStatus {
	String SURL,
	EnumStatusCode statusCode,
	string explanation,
	EnumErrorCode errorCode
	} returnSURLStatus[]

2.18.3. DESCRIPTION

srmReleaseFiles releases the files in a space. The file is not removed, but the space occupied by the released file is eligible for removal if space is needed.

- String *userID* (required)
  User authentication identifier. See Section II for notes.
- String *authorizationID*
  User authorization information. The *authorizationID* may be NULL. See Section II for notes.
- String *storageSystemInfo*
  String containining information specific to the underlying storage system. The *storageSystemInfo* may be NULL. See Section II for notes.
- String *SURL*[] (required)
  SURLs to be released.
- Boolean *releaseAllCurrentlyPinnedFiles*
  releaseAllCurrentlyPinnedFiles indicates if all currently pinned files for the client are to be released. Default is false.
- ReturnRequestStatus *returnStatus* (required)
  Output parameter reporting the success or failure of the request. In case of failure, the associated *errorCode* is returned.
- ReturnSURLStatus *returnSURLStatus*[]
  Output parameter reporting the status of each file in the request. In case of failure, the associated *errorCode* is returned. *returnSURLStatus*[] may be empty and NULL. If returned to the client, *SURL* and its *statusCode* are required to return.
  - String *SURL* (required)
    SURL that is released from the client space.

2.18.4. RETURN CODE

On successful abort, the *statusCode* is set to SRM_SUCCESS.
On failure, the *statusCode* is set to SRM_FAILURE and the *errorCode* is set.
If only partial files were successful, the *statusCode* is set to SRM_PARTIAL_SUCCESS, and the *restunSURLStatus* should explain on those failed files.

2.18.5. ERROR CODE

When status is failure, *errorCode* is set to one of the following:

For request level resturnStatus,

SRM_AUTHENTICATION_FAILURE
- SRM server failed to authenticate the client

SRM_AUTHORIZATION_FAILURE
- client is not authorized to release files

SRM_INVALID_REQUEST
- *releaseAllCurrentlyPinnedFiles* is false and input *SURL*[] is empty.

For file level returnStatus,

SRM_INVALID_PATH
- *SURL* does not refer to an existing file request

SRM_AUTHORIZATION_FAILURE
- client is not authorized to release *SURL*

2.18.6. NOTES on the Core Behavior

a) The default value of *releaseAllCurrentlyPinnedFiles* is false.

b) If *releaseAllCurrentlyPinnedFiles* is true, then none of *SURL* must be provided, or all of *SURL*s must be provided. Otherwise SRM_INVALID_REQUEST will be returned.

c) If none of *SURL*s are provided and *releaseAllCurrentlyPinnedFiles* is true, then all currently pinned files in the client's space are released.

2.18.7. NOTES on the Advanced Behavior with Directory Management Feature

a) The *SURL* may be a directory. In this case, all files under the directory are released.

2.18.8. SEE ALSO

srmPrepareToGet, srmPrepareToPut, srmRemoteCopy, srmPutFileDone, srmPutRequestDone, srmStatusOfGetRequest, srmStatusOfPutRequest, srmReleaseRequestedFiles, srmReleaseSpace

2.19. srmReleaseRequestedFiles

2.19.1. NAME

srmReleaseRequestedFiles - releases the files under the *requestToken* from a space. Optionally the cached file (on Transfer URL) may be removed explicitly. If the file is not removed, the space occupied by the released file is eligible for removal if space is needed.

2.19.2. SYNOPSIS

In	Out

String userID
String authorizationID
String storageSystemInfo
String requestToken
String SURL[]
Boolean releaseAllCurrentlyPinnedFiles
Boolean remove
	ReturnRequestStatus {
	EnumStatusCode statusCode,
	string explanation,
	EnumErrorCode errorCode
	} returnStatus
	ReturnSURLStatus {
	String SURL,
	EnumStatusCode statusCode,
	string explanation,
	EnumErrorCode errorCode
	} returnSURLStatus[]

2.19.3. DESCRIPTION

srmReleaseRequestedFiles releases the files under the *requestToken* from a space. Optionally the cached file (on Transfer URL) may be removed explicitly. If the file is not removed, the space occupied by the released file is eligible for removal if space is needed.

- String *userID* (required)
  User authentication identifier. See Section II for notes.
- String *authorizationID*
  User authorization information. The *authorizationID* may be NULL. See Section II for notes.
- String *storageSystemInfo*
  String containining information specific to the underlying storage system. The *storageSystemInfo* may be NULL. See Section II for notes.
- String *requestToken* (required)
  A token associated with the previously submitted request. The requestToken was returned by the function initiating the request (e.g. *srmPrepareToGet*()).
- String *SURL*[]
  SURLs to be released.
- Boolean *releaseAllCurrentlyPinnedFiles*
  releaseAllCurrentlyPinnedFiles indicates if all currently pinned files in the request associated with the *requestToken* are to be released. Default is false.
- Boolean *remove*
  remove indicates if all currently pinned files in the request are to be released AND the Transfer URL is to be removed. Default is false.
- ReturnRequestStatus *returnStatus* (required)
  Output parameter reporting the success or failure of the request. In case of failure, the associated *errorCode* is returned.
- ReturnSURLStatus *returnSURLStatus*[]
  Output parameter reporting the status of each file in the request. In case of failure, the associated *errorCode* is returned. *returnSURLStatus*[] may be empty and NULL. If returned to the client, *SURL* and its *statusCode* are required to return.
  - String *SURL* (required)
    SURL that is released.

2.19.4. RETURN CODE

On successful abort, the *statusCode* is set to SRM_SUCCESS.
On failure, the *statusCode* is set to SRM_FAILURE and the *errorCode* is set.
If only partial files were successful, the *statusCode* is set to SRM_PARTIAL_SUCCESS, and the *restunSURLStatus* should explain on those failed files.

2.19.5. ERROR CODE

When status is failure, *errorCode* is set to one of the following:

For request level resturnStatus,

SRM_AUTHENTICATION_FAILURE
- SRM server failed to authenticate the client

SRM_AUTHORIZATION_FAILURE
- client is not authorized to call the request specified by the *requestToken*

SRM_INVALID_REQUEST_TOKEN
- *requestToken* does not refer to an existing file request

SRM_INVALID_REQUEST
- *releaseAllCurrentlyPinnedFiles* is false and *SURL*[] is empty

For file level returnStatus,

SRM_INVALID_PATH
- *SURL* does not refer to an existing file request

SRM_AUTHORIZATION_FAILURE
- client is not authorized to release the specified *SURL*

2.19.6. NOTES on the Core Behavior

a) *releaseAllCurrentlyPinnedFiles* is valid within the *requestToken*.
b) The default value of *releaseAllCurrentlyPinnedFiles* is false.
c) If *releaseAllCurrentlyPinnedFiles* is true, then none of *SURL* must be provided, or all of *SURL*s in the *requestToken* must be provided. Otherwise SRM_INVALID_REQUEST will be returned.
d) If none of *SURL*s are provided and *releaseAllCurrentlyPinnedFiles* is true, then all currently pinned files in the client's request are released.
e) Once a file is released, its lifetime cannot be extended with the *srmExtendFileLifetime*() function. If the released file is needed again, it should be requested again.
f) *Remove* flag is false by default. If *remove* flag is on, then it releases the TURL and remove from the SRM cache.

2.19.7. NOTES on the Advanced Behavior with Directory Management Feature

a) The *SURL* may be a directory. In this case, all files under the directory are released.

2.19.8. SEE ALSO

srmPrepareToGet, srmPrepareToPut, srmRemoteCopy, srmPutFileDone, srmPutRequestDone, srmStatusOfGetRequest, srmStatusOfPutRequest, srmReleaseFiles, srmReleaseSpace

2.20. srmRm

2.20.1. NAME
srmRm - removes local files regardless of the requests that got the files into the space.

2.20.2. SYNOPSIS

In	Out
String userID	String requestToken
String authorizationID	ReturnRequestStatus {
String storageSystemInfo	EnumStatusCode statusCode,
String SURL[]	string explanation,
Boolean recursiveForce	EnumErrorCode errorCode
Int offset	} returnStatus
Int count	ReturnSURLStatus {
	String SURL,
	EnumStatusCode statusCode,
	string explanation,
	EnumErrorCode errorCode
	} returnSURLStatus []
	Boolean partialList
	Int totalFilesInTheRequest

2.20.3. DESCRIPTION
srmRm removes local files regardless of the requests that got the files into the space.

- String *userID* (required)
  User authentication identifier. See Section II for notes.
- String *authorizationID*
  User authorization information. The *authorizationID* may be NULL. See Section II for notes.
- String *storageSystemInfo*
  String containining information specific to the underlying storage system. The *storageSystemInfo* may be NULL. See Section II for notes.
- String *SURL*[] (required)
  SURLs to be removed.
- Boolean *recursiveForce* (advanced option)
  An advanced option when Directory Management feature is supported. *recursiveForce* indicates if all files under *SURL*s are to be removed recursively. Default is false.
- Int *offset*
  Integer indicator for long listed responses that the listing starts from the *offset*. Default is 0 for the first entry.
- Int *count*
  Integer indicator for long listed responses that the listing contains the number of *count.* Default 0 (zero) indicates to return all entries of the list.
- String *requestToken*
  Output parameter string token is associated with the request for the later asynchronous status request. *requestToken* may be NULL, in case *srmRm* is processed without delay.

- ReturnRequestStatus *returnStatus* (required)
  Output parameter reporting the success or failure of the request. In case of failure, the associated *errorCode* is returned.
- ReturnSURLStatus *returnSURLStatus*[]
  Output parameter reporting the status of each file in the request. In case of failure, the associated *errorCode* is returned. *returnSURLStatus*[] may be empty and NULL. If returned to the client, *SURL* and its *statusCode* are required to return.
  - String *SURL* (required)
    SURL that is removed.
- Boolean partialList
  Output parameter indicating if return values are partial or not. If true, *requestToken* must be returned. Default is false.
- Int totalFilesInTheRequest
  Output parameter indicating how many files in the request for a hint to the asynchronous status cals.

2.20.4. RETURN CODE

On successful abort, the *statusCode* is set to SRM_SUCCESS.
On failure, the *statusCode* is set to SRM_FAILURE and the *errorCode* is set.
If only partial files were successful, the *statusCode* is set to SRM_PARTIAL_SUCCESS, and the *restunSURLStatus* should explain on those failed files.

2.20.5. ERROR CODE

When status is failure, *errorCode* is set to one of the following:

For request level resturnStatus,

SRM_AUTHENTICATION_FAILURE
- SRM server failed to authenticate the client

SRM_AUTHORIZATION_FAILURE
- client is not authorized to remove any files

SRM_NOT_SUPPORTED
- *resursiveForce* is not supported when *recursiveForce* is set to true

SRM_TOO_MANY_RESULTS
- Request produced too many results that SRM server cannot handle, and *offset* and *count* cannot fit the results to return.

For file level returnStatus,

SRM_INVALID_PATH
- *SURL* does not refer to an existing file request

SRM_AUTHORIZATION_FAILURE
- client is not authorized to remove *SURL*

2.20.6. NOTES on the Core Behavior

a) If file removal is processed without delay, the *requestToken* may not be returned.

2.20.7. NOTES on the Advanced Behavior with Directory Management Feature

a) To remove empty directories, *srmRmdir*() must to be used.
b) *SURL*s can be either directories or files.
c) Boolean *recursiveForce* is false by default.
d) When *SURL* is a directory, all files under the directory and the directory itself will be removed, only if *recursiveForce* is true.

2.20.8. NOTES on the Advanced Behavior with Authorization Feature

a) Authorization checks are performed on *SURL*s before files can be removed.

2.20.9. SEE ALSO

srmRmStatus, srmLs, srmLsDetails

2.21. srmRmStatus

2.21.1. NAME

srmRmStatus - is a status call for srmRm in case that the srmRm could not be performed synchronously.

2.21.2. SYNOPSIS

In	Out	
String userID	ReturnRequestStatus {	
String authorizationID	EnumStatusCode	statusCode,
String requestToken	string	explanation,
Int offset	EnumErrorCode	errorCode
Int count	} returnStatus	
	ReturnSURLStatus {	
	String	SURL,
	EnumStatusCode	statusCode,
	string	explanation,
	EnumErrorCode	errorCode
	} returnSURLStatus[]	

2.21.3. DESCRIPTION

srmRmStatus is a status call for srmRm in case that the srmRm could not be performed synchronously.

- String *userID* (required)
  User authentication identifier. See Section II for notes.
- String *authorizationID*
  User authorization information. The *authorizationID* may be NULL. See Section II for notes.
- String *requestToken* (required)
  A token associated with the previously submitted *srmRm* request. The *requestToken* was returned by the function initiating the request.
- Int *offset*

Integer indicator for long listed responses that the listing starts from the *offset*. Default is 0 for the first entry.

- Int *count*
  Integer indicator for long listed responses that the listing contains the number of *count*. Default 0 (zero) indicates to return all entries of the list.

- ReturnRequestStatus *returnStatus* (required)
  Output parameter reporting the success or failure of the request. In case of failure, the associated *errorCode* is returned.
- ReturnSURLStatus *returnSURLStatus*[]
  Output parameter reporting the status of each file in the request. In case of failure, the associated *errorCode* is returned. *returnSURLStatus*[] may be empty and NULL. If returned to the client, *SURL* and its *statusCode* are required to return.
  - String *SURL* (required)
    SURL that is removed.

2.21.4. RETURN CODE

On successful abort, the *statusCode* is set to SRM_SUCCESS.
On failure, the *statusCode* is set to SRM_FAILURE and the *errorCode* is set.
If only partial files were successful, the *statusCode* is set to SRM_PARTIAL_SUCCESS, and the *restunSURLStatus* should explain on those failed files.

2.21.5. ERROR CODE

When status is failure, *errorCode* is set to one of the following:

For request level resturnStatus,

SRM_AUTHENTICATION_FAILURE
- SRM server failed to authenticate the client

SRM_AUTHORIZATION_FAILURE
- client is not authorized to check the status of the previously requested *srmRm*

SRM_TOO_MANY_RESULTS
- Request produced too many results that SRM server cannot handle, and *offset* and *count* cannot fit the results to return.

For file level returnStatus,

SRM_INVALID_PATH
- *SURL* does not refer to an existing file request

SRM_AUTHORIZATION_FAILURE
- client is not authorized to remove *SURL*

2.21.6. SEE ALSO

srmRm, srmLs, srmLsDetails, srmGetRequestTokens

2.22. srmStatusOfGetRequest

2.22.1. NAME

srmStatusOfGetRequest - is the status call for srmPrepareToGet.

2.22.2. SYNOPSIS

In	Out

String userID
String authorizationID
String requestToken
String fromSURL[]
Int offset
Int count
	ReturnRequestStatus {
	EnumStatusCode statusCode,
	string explanation,
	EnumErrorCode errorCode
	} returnStatus
	ReturnStatusForGet {
	String fromSURL,
	String transferURL,
	EnumStatusCode statusCode,
	String explanation,
	EnumErrorCode errorCode,
	int fileSize,
	UTCtime transferExpirationTime,
	int remainintLifetime,
	EnumFileStorageType fileStorageType,
	String checksumType,
	String checksumValue,
	String spaceToken
	} returnGetStatus[]
	Boolean partialList
	Int totalFilesInTheRequest

2.22.3. DESCRIPTION

srmStatusOfGetRequest is the status call for srmPrepareToGet.

- String *userID* (required)
  User authentication identifier. See Section II for notes.
- String *authorizationID*
  User authorization information. The *authorizationID* may be NULL. See Section II for notes.
- String *requestToken* (required)
  A token associated with the previously submitted *srmPrepareToGet* request. The *requestToken* was returned by the function initiating the request.
- String *fromSURL*[]
  SURLs to request their status.
- Int *offset*
  Integer indicator for long listed responses that the listing starts from the *offset*. Default is 0 for the first entry.
- Int *count*
  Integer indicator for long listed responses that the listing contains the number of *count*. Default 0 (zero) indicates to return all entries of the list.
- ReturnRequestStatus *returnStatus* (required)

Output parameter reporting the success or failure of the request. In case of failure, the associated *errorCode* is returned.

- ReturnStatusForGet *returnGetStatus*[]
  Output parameter reporting the status of each file in the request. In case of failure, the associated *errorCode* is returned. *returnGetStatus*[] may be empty and NULL. If returned to the client, *fromSURL* and its *statusCode* are required to return.
  - String *fromSURL* (required)
    SURL that client has requested.
  - String *transferURL*
    Transfer URL that SRM server prepared for client to transfer for the corresponding *fromSURL*.
  - Int *fileSize*
    File size in bytes for the *fromSURL*.
  - UTCtime *transferExpirationTime*
    UTCtime for file transfer expiration time that is assigned to the *transferURL*.
  - Int *remainingLifetime*
    Integer life time in seconds that is remained on the *SURL*.
  - EnumFileStorageType *fileStorageType*
    File storage type of *SURL*.
  - String *checksumType*
    Checksum type if check is performed. For example, MD5.
  - String *checksumValue*
    Checksum value if check is performed.
  - String *spaceToken*
    An advanced option when Space Management feature is supported. A token associated with the space if a particular space in file storage type is to be used for *SURL*.
- Boolean partialList
  Output parameter indicating if return values are partial or not. If true, *requestToken* must be returned. Default is false.
- Int totalFilesInTheRequest
  Output parameter indicating how many files in the request for a hint to the asynchronous status cals.

2.22.4. RETURN CODE

On successful abort, the *statusCode* is set to SRM_SUCCESS.
On failure, the *statusCode* is set to SRM_FAILURE and the *errorCode* is set.
If only partial files were successful, the *statusCode* is set to SRM_PARTIAL_SUCCESS, and the *restunGetStatus* should explain on those failed files.

2.22.5. ERROR CODE

When status is failure, *errorCode* is set to one of the following:

For request level resturnStatus,
SRM_AUTHENTICATION_FAILURE
- SRM server failed to authenticate the client

SRM_AUTHORIZATION_FAILURE
- client is not authorized to request the status of the request that is associated with *requestToken*

SRM_TOO_MANY_RESULTS
- Request produced too many results that SRM server cannot handle, and *offset* and *count* cannot fit the results to return.

SRM_INVALID_REQUEST_TOKEN
- *requestToken* does not refer to an existing request

For file level returnStatus,

SRM_INVALID_PATH
- *SURL* does not refer to an existing file request

SRM_AUTHORIZATION_FAILURE
- Client is not authorized to retrieve *SURL*

SRM_FILE_LIFETIME_EXPIRED
- *SURL* is expired

2.22.6. NOTES on the Core Behavior

a) If *fromSURL* is not provided, the status for all the files associated with the *requestToken* is returned.

b) Output parameter *fromSURL* is the same as the *fromSURL* that client provided with *srmPrepareToGet*().

2.22.7. NOTES on the Advanced Behavior with Space Management Feature

a) *spaceToken* may be returned for each file in the request, since a single request can have files in multiple spaces.

2.22.8. SEE ALSO

srmPrepareToGet, srmPrepareToPut, srmRemoteCopy, srmLs, srmLsDetails, srmReleaseFiles, srmReleaseRequestedFiles, srmRm, srmRmStatus, srmGetRequestTokens

2.23. srmStatusOfPutRequest

2.23.1. NAME

srmStatusPutRequest - is the status call for srmPrepareToPut.

2.23.2. SYNOPSIS

In	Out

String userID
String authorizationID
String requestToken
String toSURL[]
Int offset
Int count
	ReturnRequestStatus {
	EnumStatusCode statusCode,
	string explanation,
	EnumErrorCode errorCode
	} returnStatus
	ReturnStatusForPut {
	String referenceSURL,
	String transferURL,
	int fileSize,
	EnumStatusCode statusCode,
	String explanation,
	EnumErrorCode errorCode,
	UTCtime transferExpirationTime,
	int remainingLifetime,
	EnumFileStorageType fileStorageType,
	String checksumType,
	String checksumValue,
	String spaceToken
	} returnPutStatus[]
	Boolean partialList,
	Int totalFilesInTheRequest

2.23.3. DESCRIPTION

srmStatusPutRequest is the status call for srmPrepareToPut.

- String *userID* (required)
  User authentication identifier. See Section II for notes.
- String *authorizationID*
  User authorization information. The *authorizationID* may be NULL. See Section II for notes.
- String *requestToken* (required)
  A token associated with the previously submitted *srmPrepareToPut* request. The *requestToken* was returned by the function initiating the request.
- String *toSURL*[]
  *toSURL*s to check the status, if client submitted in *srmPrepareToPut*.
- Int *offset*
  Integer indicator for long listed responses that the listing starts from the *offset*. Default is 0 for the first entry.
- Int *count*
  Integer indicator for long listed responses that the listing contains the number of *count*. Default 0 (zero) indicates to return all entries of the list.
- ReturnRequestStatus *returnStatus* (required)
  Output parameter reporting the success or failure of the request. In case of failure, the associated *errorCode* is returned.
- ReturnStatusForPut *returnPutStatus*[]

Output parameter reporting the status of each file in the request. In case of failure, the associated *errorCode* is returned. *returnPutStatus*[] may be empty and NULL. If returned to the client, *statusCode* are required to return.

- String *referenceSURL*
  Reference SURL is an SURL that SRM server and client can refer to the file that client would put. If client provided *toSURL* in the request, *referenceSURL* must be the same as the *toSURL*.
- String *transferURL*
  Transfer URL that SRM server prepared for client to transfer a file into. It may correspond to the *referenceSURL*.
- Int *fileSize*
  File size in bytes that SRM server allocated for the *transferURL*.
- UTCtime *transferExpirationTime*
  UTCtime for file transfer expiration time that is assigned to the *transferURL*.
- Int *remainingLifetime*
  Integer lifetime in seconds that is assigned to the *SURL* that file transfer is completed.
- EnumFileStorageType *fileStorageType*
  File storage type of space allocation for the file request.
- String *checksumType*
  Checksum type if check is performed. For example, MD5.
- String *checksumValue*
  Checksum value if check is performed.
- String *spaceToken* (advanced option)
  An advanced option when Space Management feature is supported. A token associated with the space if a particular space in file storage type is to be used for file request.

- Boolean partialList
  Output parameter indicating if return values are partial or not. Default is false.
- Int totalFilesInTheRequest
  Output parameter indicating how many files in the request for a hint to the asynchronous status cals.

2.23.4. RETURN CODE

For request level resturnStatus,

On successful request, the *statusCode* is set to SRM_SUCCESS.
On failure, the *statusCode* is set to SRM_FAILURE and the *errorCode* is set.
If only partial files were successful, the *statusCode* is set to SRM_PARTIAL_SUCCESS, and the *restunPutStatus* should explain on those failed files.

For file level returnStatus,

SRM_SPACE_AVAILABLE

- Space is ready for client to transfer a file into the Transfer URL

2.23.5. ERROR CODE

When status is failure, *errorCode* is set to one of the following:

For request level resturnStatus,

SRM_AUTHENTICATION_FAILURE
- SRM server failed to authenticate the client

SRM_AUTHORIZATION_FAILURE
- client is not authorized to request the status of the request that is associated with *requestToken*

SRM_INVALID_REQUEST_TOKEN
- *requestToken* does not refer to an existing request

SRM_TOO_MANY_RESULTS
- Request produced too many results that SRM server cannot handle, and *offset* and *count* cannot fit the results to return.

For file level returnStatus,

SRM_INVALID_PATH
- *toSURL* does not refer to an existing file request

SRM_AUTHORIZATION_FAILURE
- Client is not authorized to put data into *toSURL* that client provided in *srmPrepareToPut*
- Client is not authorized to put data into space that client provided with *spaceToken* in *srmPrepareToPut*

SRM_INVALID_SPACE_TOKEN
- *spaceToken* that client provided does not refer to an existing space

2.23.6. NOTES on the Core Behavior

a) If input parameter *toSURL* is empty, it returns status for all files associated with the *requestToken*.

b) The output parameter *referenceSURL* must be the input parameter *toSURL* of *srmPrepareToPut* if the client provided it. Otherwise, it is an SURL that SRM assigned as the reference.

c) When the space to put a file is allocated by SRM, *referenceSURL*, *transferURL*, *fileStorageType*, *fileSize* and *statusCode* must be returned.

d) Returned *fileSize* is initially the default allocated space size, unless a specific size was requested for the file. After the *srmPutFileDone*() or *srmPutRequestDone*() is issued, the file size is set to the actual file size that occupies the location. If the actual file size is more than the allocated space size, then it is up to the SRM server to decide the behavior.

e) While the *srmPreparetToPut* is processed, the output parameter *remainingLifetime* is NULL until the *srmPutFileDone* (or *srmPutRequestDone*) is issued by the client. After an *srmPutFileDone* is issued, the *remainingLifetime* indicates the remaining lifetime of the file.

2.23.7. NOTES on the Advanced Behavior with Space Management Feature

a) *spacetToken* must be a valid output parameter.

2.23.8. SEE ALSO

srmPrepareToGet, srmPrepareToPut, srmRemoteCopy, srmLs, srmLsDetails, srmReleaseFiles, srmReleaseRequestedFiles, srmRm, srmRmStatus, srmGetRequestTokens

3. Advanced feature set 1 : Remote Access Functions

summary:

srmRemoteCopy
srmStatusOfRemoteCopyRequest

details:

3.1. srmRemoteCopy

3.1.1. NAME

srmRemoteCopy - replicates files from one site to another.

3.1.2. SYNOPSIS

In	Out
String userID String authorizationID String fromStorageSystemInfo String toStorageSystemInfo String userRequestDescription Int totalRetryTime Boolean streaming EnumOverwriteMode overwriteOption Boolean removeSourceFiles Boolean performChecksum CopyFileRequest { String fromSURL, String toSURL, Int knownSizeOfFile, int fileLifetime, EnumFileStorageType toFileStorageType, String fromStorageSystemInfo, String toStorageSystemInfo, String spaceToken, Boolean isSourceADirectory, Boolean allLevelRecursive, int numOfLevels } copyRequests[] String spaceToken EnumFileStorageType toFileStorageType	String requestToken ReturnRequestStatus { EnumStatusCode statusCode, string explanation, EnumErrorCode errorCode } returnStatus

3.1.3. DESCRIPTION

srmRemoteCopy replicates files from one site to another.

- String *userID* (required)

User authentication identifier. See Section II for notes.

- String *authorizationID*
  User authorization information. The *authorizationID* may be NULL. See Section II for notes.
- String *fromStorageSystemInfo*
  String containining information specific to the source storage system that is associated with the *fromSURL*s. The *fromStorageSystemInfo* may be NULL. See Section II for notes.
- String *toStorageSystemInfo*
  String containining information specific to the target storage system that is associated with the *toSURL*s or *toFileStorageType*. The *toStorageSystemInfo* may be NULL. See Section II for notes.
- String *userRequestDescription*
  String description of the request. It may be used for later retrieval of the rquest token.
- Int *totalRetryTime*
  totalRetryTime is the amount of time in seconds that SRM should try to transfer failed files, if some file transfers fail. Default is 0 that SRM assigns by default.
- Boolean *streaming*
  Boolean indication of *streaming* mode. When *streaming* mode is on, full space at the target storage does not have to be prepared to hold all files in the request. Default is false.
- EnumOverwriteMode *overwriteOption*
  SRM needs to replicate the file according to the *overwriteOption* on the target that SRM brings the files into.
- Boolean *removeSourceFiles*
  Boolean indication of file removal at the source (*fromSURL*) after the copy is performed. Default is false.
- Boolean *performChecksum*
  performChecksum indicates that checksum calculation for all files in the request needs to be performed when files get into its designated target space. Default is false.
- CopyFileRequest *copyRequests*[] (required)
  Input parameter copy*Request* consists of SURL information that client wants to copy from one site to another.
  - String *fromSURL* (required)
    Source SURL
  - String *toSURL*
    Target SURL
  - Int *knownSizeOfFile*
    File size of the SURL if known. Default is 0, indicating unknown.
  - Int *fileLifetime*
    Desired life time of the file in seconds once when it's in the target SRM. Default is 0, and SRM assigns default lifetime.
  - EnumFileStorageType *toFileStorageType*
    toFileStorageType indicates which type of storage that *fromSURL*s are copied into in target SRM.
  - String *fromStorageSystemInfo*

String containining information specific to the underlying storage system associated with the *fromSURL*. The *fromStorageSystemInfo* may be NULL. See Section II for notes

- String *toStorageSystemInfo*
  String containining information specific to the underlying storage system associated with the *toSURL, toFileStorageType*, or the space that target SRM will bring *fromSURL* into. The *toStorageSystemInfo* may be NULL. See Section II for notes
- String *spaceToken* (advanced option)
  An advanced option when Space Management feature is supported. A token associated with the space if a particular space in file storage type is to be used. The spaceToken is acquired separately (e.g. *srmReserveSpace*).
- Boolean *isSourceADirectory* (advanced option)
  An advanced option when Directory Management feature is supported. Boolean indicator if *fromSURL* is a directory. Default is false.
- Boolean *allLevelRecursive* (advanced option)
  An advanced option when Directory Management feature is supported. Boolean indicator if *fromSURL* is a directory and all files under the *fromSURL* must be retrieved. The corresponding target directory structure must be hierachically created according to the source directory structure. Defautl is false.
- Int *numOfLevels* (advanced option)
  An advanced option when Directory Management feature is supported. Positive integer indicator for how many levels of the recursive directory must be browsed and all files in those directories to be retrieved. Default is 0 for the single level of directory (no recursive).

- String *spaceToken* (advanced option)
  An advanced option when Space Management feature is supported. A token associated with the space if a particular space in file storage type is to be used. The *spaceToken* is acquired separately (e.g. *srmReserveSpace*). This is request level option. If file level option (*spaceToken*) in the *copytRequests* is set, it takes the priority.
- EnumFileStorageType *toFileStorageType*
  toFileStorageType indicates which type of storage that *fromSURL*s are copied into in target SRM. This is request level option. If file level option (*toFileStorageType*) in the *copytRequests* is set, it takes the priority.
- String *requestToken* (required)
  Output parameter string token is associated with the request for the later asynchronous status request.
- ReturnRequestStatus *returnStatus* (required)
  Output parameter reporting the success or failure of the request. In case of failure, the associated *errorCode* is returned.

3.1.4. RETURN CODE

On successful abort, the *statusCode* is set to SRM_SUCCESS.
On failure, the *statusCode* is set to SRM_FAILURE and the *errorCode* is set.

3.1.5. ERROR CODE

When status is failure, *errorCode* is set to one of the following:

SRM_AUTHENTICATION_FAILURE
- SRM server failed to authenticate the client

SRM_AUTHORIZATION_FAILURE
- client is not authorized to submit the *srmRemoteCopy* request

SRM_INVALID_REQUEST
- input parameters do not conform the SRM. For example, client requested negative *totalRetryTime* and SRM cannot honor the number.
- Both *fromSURL* and *toSURL* are remote SURLs, representing 3rd party copy.
- Both *fromSURL* and *toSURL* are local SURLs, representing local copy.

SRM_NOT_SUPPORTED
- SRM server does not support the given input parameters. For example, client requested to use *spaceToken*, but SRM does not support the space management feature.

3.1.6. NOTES on the Behavior

a) Third party copy are not supported, from a remote location to another remote location. Either source or target must be local to the SRM where the request is submitted.

b) *srmRemoteCopy* can be done in a pull of a push mode. Pull mode is to copy from remote location to local SRM. Push mode is to copy from local SRM to remote location. The mode is determined by the source and target SURLs.

c) SRM server always releases files on the source after the copy is done.

d) When *removeSourceFiles* is true, SRM removes the source files on behalf of the client after the copy is done.

e) The client may release the file local to the SRM after the copy is completed in push mode. If the client releases a file being copied to another location before the transfer is completed, then the release fails.

f) When the streaming option is true, and the target space is full, the copy operation is suspended till more space is made available. When the streaming option is false, and if there is not enough space to accommodate the entire request, the SRM returns failure.

g) There is no protocol negotiation with the client.

h) *totalRetryTime* represents the length of time in seconds that the SRM will try to copy file whose transfer previously failed. This action takes place after all the other file transfers for the request completed.

i) In case that retries fail, the return should include an explanation of why the retries failed.

j) *srmRemoteCopy* performs a copy from or to remote sites only. Thus, when both *fromSURL* and *toSURL* are local, an error SRM_INVALID_REQUEST is returned. A copy of a local file to another must be done by the *srmCp* function if the "directory management feature" is supported.

3.1.7. NOTES on the Advanced Behavior with Space Management Feature

a) The default value of "lifetime" for volatile or durable file types must be equal to or less than the lifetime left in the space of the corresponding *spaceToken*. The default value of *toFileStorageType* is "volatile".

3.1.8. NOTES on the Advanced Behavior with Directory Management Feature

a) Empty directories must be copied as well.

3.1.9. SEE ALSO

srmPrepareToGet, srmPrepareToPut, srmStatusOfRemoteCopyRequest, srmGetRequestTokens

3.2. srmStatusOfRemoteCopyRequest

3.2.1. NAME

srmStatusOfRemoteCopyRequest - is the status call for srmRemoteCopy.

3.2.2. SYNOPSIS

In	Out

String userID
String authorizationID
String requestToken
String fromSURL[]
Int offset
Int count
	ReturnRequestStatus {
	EnumStatusCode statusCode,
	string explanation,
	EnumErrorCode errorCode
	} returnStatus
	RequestStatusForCopy {
	String fromSURL,
	String toSURL,
	EnumFileStorageType toFileStorageType,
	Int fileSize,
	EnumFileType fileType,
	Int fileLifetime,
	EnumStatusCode statusCode,
	EnumErrorCode errorCode,
	String explanation,
	String checksumType,
	String checksumValue,
	String spaceToken
	} returnCopyStatus[]
	Boolean partialList
	Int totalFilesInTheRequest

3.2.3. DESCRIPTION

srmStatusOfRemoteCopyRequest is the status call for srmRemoteCopy.

- String *userID* (required)
  User authentication identifier. See Section II for notes.
- String *authorizationID*
  User authorization information. The *authorizationID* may be NULL. See Section II for notes.
- String *requestToken* (required)
  A token associated with the previously submitted *srmRemoteCopy* request. The *requestToken* was returned by the function initiating the request.

- String *fromSURL*[]
  Selective *fromSURL*s to check the status.
- Int *offset*
  Integer indicator for long listed responses that the listing starts from the *offset*. Default is 0 for the first entry.
- Int *count*
  Integer indicator for long listed responses that the listing contains the number of *count*. Default 0 (zero) indicates to return all entries of the list.
- ReturnRequestStatus *returnStatus* (required)
  Output parameter reporting the success or failure of the request. In case of failure, the associated *errorCode* is returned.
- ReturnStatusForCopy *returnCopyStatus*[]
  Output parameter reporting the status of each file in the request. In case of failure, the associated *errorCode* is returned. *returnCopyStatus*[] may be empty and NULL. If returned to the client, *statusCode* are required to return.
  - String *fromSURL* (required)
    Source SURL.
  - String *toSURL* (required)
    Target SURL that SRM server copied the *fromSURL* into.
  - EnumFileStorageType *toFileStorageType*
    File storage type of target SURL (*toSURL*).
  - Int *fileSize*
    File size in bytes for *toSURL*.
  - EnumFileType *fileType*
    File type of target SURL (*toSURL*).
  - Int *fileLifetime*
    Integer lifetime in seconds that is assigned to the *toSURL*.
  - String *checksumType*
    Checksum type if check is performed. For example, MD5.
  - String *checksumValue*
    Checksum value if check is performed.
  - String *spaceToken*
    An advanced option when Space Management feature is supported. A token associated with the space if a particular space in file storage type is to be used for *toSURL*.
- Boolean partialList
  Output parameter indicating if return values are partial or not. If true, *requestToken* must be returned. Default is false.
- Int totalFilesInTheRequest
  Output parameter indicating how many files in the request for a hint to the asynchronous status cals.

3.2.4. RETURN CODE

For request level resturnStatus,

On successful abort, the *statusCode* is set to SRM_SUCCESS.

On failure, the *statusCode* is set to SRM_FAILURE and the *errorCode* is set.
If only partial files were successful, the *statusCode* is set to SRM_PARTIAL_SUCCESS, and the *restunCopyStatus* should explain on those failed files.

For file level returnStatus,

SRM_REQUEST_QUEUED

- File copy request is on the queue

3.2.5. ERROR CODE

When status is failure, *errorCode* is set to one of the following:

For request level returnStatus,

SRM_AUTHENTICATION_FAILURE

- SRM server failed to authenticate the client

SRM_AUTHORIZATION_FAILURE

- client is not authorized to request the status of the request that is associated with *requestToken*

SRM_TOO_MANY_RESULTS

- Request produced too many results that SRM server cannot handle, and *offset* and *count* cannot fit the results to return.

SRM_INVALID_REQUEST_TOKEN

- *requestToken* does not refer to an existing request

SRM_INVALID_SPACE_TOKEN

- *spaceToken* does not refer to an existing space

SRM_NOT_SUPPORTED

- Directory or space management feature is not supported

For file level returnStatus,

SRM_INVALID_PATH

- *fromSURL* does not refer to an existing file request

SRM_AUTHORIZATION_FAILURE

- Client is not authorized to copy files from *fromSURL*
- Client is not authorized to copy files into *toSURL*
- Client is not authorized to copy files into the space that client provided with *spaceToken* or *toFileStorageType* in *srmRemoteCopy*

SRM_INVALID_SPACE_TOKEN

- *spaceToken* does not refer to an existing space

SRM_NOT_SUPPORTED

- Directory management feature is not supported

3.2.6. NOTES on the Behavior

a) If any *fromSURL* is not provided, the status for all the files associated with the *requestToken* is returned.

4. Advanced feature set 2 : Space Management Functions

summary:

srmCleanupFilesFromSpace
srmGetSpaceMetaData
srmGetSpaceTokens
srmReleaseSpace
srmReserveSpace
srmStatusOfCleanupFilesFromSpace
srmUpdateSpace

details:

4.1. srmCleanupFilesFromSpace

4.1.1. NAME

srmCleanupFilesFromSpace - releases all files from the space associated with the space token. The space of released files can be used when space is needed. The space is not released.

4.1.2. SYNOPSIS

In	Out	
String userID	String requestToken	
String authorizationID	ReturnRequestStatus {	
String storageSystemInfo	EnumStatusCode	statusCode,
String spaceToken	string	explanation,
Boolean remove	EnumErrorCode	errorCode
Int offset	} returnStatus	
Int count	ReturnSURLStatus {	
	String	SURL,
	EnumStatusCode	statusCode,
	string	explanation,
	EnumErrorCode	errorCode
	} returnSURLStatus []	
	Boolean partialList	
	Int totalFilesInTheRequest	

4.1.3. DESCRIPTION

srmCleanupFilesFromSpace releases all files from the space associated with the space token. The space of released files can be used when space is needed. The space is not released.

- String *userID* (required)
  User authentication identifier. See Section II for notes.
- String *authorizationID*
  User authorization information. The *authorizationID* may be NULL. See Section II for notes.

- String *storageSystemInfo*
  String containining information specific to the underlying storage system. The *storageSystemInfo* may be NULL. See Section II for notes.
- String *spaceToken* (required)
  A token associated with the space to clean up. The *spaceToken* is acquired separately (e.g. *srmReserveSpace*).
- Boolean *remove*
  remove indicates to remove all files in the space that is associated with *spaceToken* after the clean-up. Default is false.
- Int *offset*
  Integer indicator for long listed responses that the listing starts from the *offset*. Default is 0 for the first entry.
- Int *count*
  Integer indicator for long listed responses that the listing contains the number of *count.* Default 0 (zero) indicates to return all entries of the list.
- String *requestToken*
  Output parameter string token is associated with the request for the later asynchronous status request. *requestToken* may be NULL, in case *srmCleanupFilesFromSpace* is processed without delay.
- ReturnRequestStatus *returnStatus* (required)
  Output parameter reporting the success or failure of the request. In case of failure, the associated *errorCode* is returned.
- ReturnSURLStatus *returnSURLStatus*[]
  Output parameter reporting the status of each file in the request. In case of failure, the associated *errorCode* is returned. *returnSURLStatus*[] may be empty and NULL. If returned to the client, *SURL* and its *statusCode* are required to return.
  - String *SURL* (required)
    SURL that client has requested to clean up.
- Boolean partialList
  Output parameter indicating if return values are partial or not. If true, *requestToken* must be returned. Default is false.
- Int totalFilesInTheRequest
  Output parameter indicating how many files in the request for a hint to the asynchronous status cals.

4.1.4. RETURN CODE

On successful abort, the *statusCode* is set to SRM_SUCCESS.
On failure, the *statusCode* is set to SRM_FAILURE and the *errorCode* is set.
If only partial files were successful, the *statusCode* is set to SRM_PARTIAL_SUCCESS, and the *restunSURLStatus* should explain on those failed files.

4.1.5. ERROR CODE

When status is failure, *errorCode* is set to one of the following:

For request level resturnStatus,
SRM_AUTHENTICATION_FAILURE

- SRM server failed to authenticate the client

SRM_AUTHORIZATION_FAILURE

- client is not authorized to clean up the space that is associated with *spaceToken*

SRM_TOO_MANY_RESULTS

- Request produced too many results that SRM server cannot handle, and *offset* and *count* cannot fit the results to return.

For file level returnStatus,

SRM_AUTHORIZATION_FAILURE

- Client is not authorized to clean up *SURL* in the space that is associated with *spaceToken*

4.1.6. NOTES on the Behavior

a) The default value of the *remove* parameter is false. If *remove* flag is true, then all files in the *spaceToken* or *SURLs* are removed.

b) If any of the files in the space associated with the *spaceToken* cannot be removed, then *ReturnSURLStatus* must be returned for the status and the explanation.

4.1.7. SEE ALSO

srmReleaseFiles, srmReleaseRequestedFiles, srmReleaseSpace, srmUpdateSpace, srmGetSpaceTokens, srmGetSpaceMetaData

4.2. srmGetSpaceMetaData

4.2.1. NAME

srmGetSpaceMetaData - is used to retrieve meta information of a space.

4.2.2. SYNOPSIS

In	Out

String userID
String authorizationID
String spaceToken[]
	ReturnRequestStatus {
	EnumStatusCode statusCode,
	string explanation,
	EnumErrorCode errorCode
	} returnStatus
	SpaceMetaData {
	String spaceToken,
	EnumStatusCode statusCode,
	String explanation,
	EnumErrorCode errorCode,
	EnumRetentionQualityMode retentionMode,
	Boolean expired,
	String ownerID,
	Int totalSize,
	Int guaranteedSize,
	Int unusedSize,
	Int lifetimeAssigned,
	Int lifetimeLeft
	} returnSpaceInfo[]

4.2.3. DESCRIPTION

srmGetSpaceMetaData is used to retrieve meta information of a space.

- String *userID* (required)
  User authentication identifier. See Section II for notes.
- String *authorizationID*
  User authorization information. The *authorizationID* may be NULL. See Section II for notes.
- String *spaceToken[]* (required)
  A token associated with the space. The spaceToken is acquired separately (e.g. srmReserveSpace).
- ReturnRequestStatus *returnStatus* (required)
  Output parameter reporting the success or failure of the request. In case of failure, the associated *errorCode* is returned.
- SpaceMetaData *returnSpaceInfo*[]
  Output parameter reporting the space information in the request. In case of failure, the associated *errorCode* is returned. *returnSpaceInfo*[] may be empty and NULL. If returned to the client, *spaceToken* is required to return.
  - String *spaceToken* (required)
    A token associated with the space in the request.
  - EnumRetentionQualityMode *retentionMode*
    Retention quality type of the space.
  - Boolean *expired*
    expired indicates if the space has been expired.
  - String *ownerID*
    Space owner.

- Int *totalSize*
  Total space size in bytes.
- Int *guaranteedSize*
  Guaranteed space size in bytes.
- Int *unusedSize*
  unused space size in bytes.
- Int *lifetimeAssigned*
  Life time of the space that is initially assigned, in seconds.
- Int *lifetimeLeft*
  Remaining life time of the space in seconds.

4.2.4. RETURN CODE
On successful abort, the *statusCode* is set to SRM_SUCCESS.
On failure, the *statusCode* is set to SRM_FAILURE and the *errorCode* is set.
If only partial space information is available, the *statusCode* is set to SRM_PARTIAL_SUCCESS, and the *returnSpaceInfo* should explain on those failed spaces.

4.2.5. ERROR CODE
When status is failure, *errorCode* is set to one of the following:

For request level resturnStatus,

SRM_AUTHENTICATION_FAILURE
- SRM server failed to authenticate the client

SRM_AUTHORIZATION_FAILURE
- client is not authorized to request space information

SRM_TOO_MANY_RESULTS
- Request produced too many results that SRM server cannot handle, and *offset* and *count* cannot fit the results to return.

For space level returnStatus,

SRM_AUTHORIZATION_FAILURE
- client is not authorized to request information on the space that is associated with the *spaceToken*

SRM_INVALID_SPACE_TOKEN
- *spaceToken* does not refer to an existing space

4.2.6. NOTES on the Behavior
a) The returned size does not include the extra space needed to hold the directory structures.
b) If multiple spaces per space type exist, the returned metadata is for each space using the space token.

4.2.7. SEE ALSO
srmgetSpaceTokens. srmReserveSpace, srmUpdateSpace

4.3. srmGetSpaceTokens

4.3.1. NAME
srmGetSpaceToken - returns space tokens for currently allocated spaces.

4.3.2. SYNOPSIS

In	Out
String userID	ReturnRequestStatus {
String authorizationID	EnumStatusCode statusCode,
String userSpaceDescription	string explanation,
	EnumErrorCode errorCode
	} returnStatus
	String spaceTokens[]

4.3.3. DESCRIPTION
srmGetSpaceToken() returns space tokens for currently allocated spaces.

- String *userID* (required)
  User authentication identifier. See Section II for notes.
- String *authorizationID*
  User authorization information. The *authorizationID* may be NULL. See Section II for notes.
- String *userSpaceDescription*
  String containining description of the space. The description was given by the client at the time of the request (e.g. *srmReserveSpace*). The *userSpaceDescription* may be NULL.
- ReturnRequestStatus *returnStatus* (required)
  Output parameter reporting the success or failure of the request. In case of failure, the associated *errorCode* is returned.
- String *spaceTokens*[]
  Output parameter returning the space tokens owned by the client. *spaceTokens*[] may be NULL and empty.

4.3.4. RETURN CODE
On successful abort, the *statusCode* is set to SRM_SUCCESS.
On failure, the *statusCode* is set to SRM_FAILURE and the *errorCode* is set.

4.3.5. ERROR CODE
When status is failure, *errorCode* is set to one of the following:

SRM_AUTHENTICATION_FAILURE
- SRM server failed to authenticate the client

SRM_AUTHORIZATION_FAILURE
- client is not authorized to request *spaceTokens* associated with the *userSpaceDescription*

SRM_INVALID_REQUEST
- *userSpaceDescription* does not refer to an existing space

4.3.6. NOTES on the Behavior

a) If *userSpaceDescription* is null, the SRM returns all *spaceTokens* that the client owns.

b) If the client assigned the same name to multiple space reservations, the client will get back multiple space tokens.

4.3.7. SEE ALSO

srmGetSpaceMetaData, srmGetRequestTokens, srmLs, srmLsDetails

4.4. srmReleaseSpace

4.4.1. NAME

srmReleaseSpace - releases an occupied space.

4.4.2. SYNOPSIS

In	Out	
String userID	ReturnRequestStatus {	
String authorizationID	EnumStatusCode	statusCode,
String storageSystemInfo	string	explanation,
String spaceToken	EnumErrorCode	errorCode
Boolean forceFileRelease	} returnStatus	

4.4.3. DESCRIPTION

srmReleaseSpace releases an occupied space.

- String *userID* (required)
  User authentication identifier. See Section II for notes.
- String *authorizationID*
  User authorization information. The *authorizationID* may be NULL. See Section II for notes.
- String *storageSystemInfo*
  String containining information specific to the underlying storage system. The *storageSystemInfo* may be NULL. See Section II for notes.
- String *spaceToken* (required)
  A token associated with the space to release. The *spaceToken* is acquired separately (e.g. *srmReserveSpace*).
- Boolean *forceFileRelease*
  forceFileRelease indicates that the space must be released regardless of all files that it contains and their status. Default is false.
- ReturnRequestStatus *returnStatus* (required)
  Output parameter reporting the success or failure of the request. In case of failure, the associated *errorCode* is returned.

4.4.4. RETURN CODE

On successful abort, the *statusCode* is set to SRM_SUCCESS.
On failure, the *statusCode* is set to SRM_FAILURE and the *errorCode* is set.

4.4.5. ERROR CODE

When status is failure, *errorCode* is set to one of the following:

SRM_AUTHENTICATION_FAILURE
- SRM server failed to authenticate the client

SRM_AUTHORIZATION_FAILURE
- client is not authorized to release the space that is associated with the *spaceToken*

SRM_INVALID_SPACE_TOKEN
- *spaceToken* does not refer to an existing space

SRM_NOT_SUPPORTED
- *forceFileRelease* is not supported

4.4.6. NOTES on the Behavior

a) The parameter *forceFileRelease* is false by default. This means that the space will not be released if it has files that are still pinned in the space. To release the space regardless of the files it contains and their status, *forceFileRelease* must be specified as true.

b) All files must be released in the specified space before the space can be released, unless the *forceFileRelease* is set.

c) A request to release a reserved space that has on-going file transfers will be postponed until after the transfers complete (if the transfers cannot be interrupted). The files will then be released, and the space released.

d) When space is releasable and *forceFileRelease* is true, all the files in the space are released.

e) *srmReleaseSpace* may not complete right away because of the lifetime of files in the space. When space is released, the files in that space are treated according to their types: If file storage types are permanent, keep them until further operation such as srmRm is issued by the client. If file storage types are durable, perform necessary actions at the end of their lifetime. If file storage types are volatile, release those files at the end of their lifetime.

4.4.7. SEE ALSO

srmCleanupFilesFromSpace, srmReserveSpace, srmGetSpaceMetaData, srmGetSpaceTokens, srmUpdateSpace

4.5. srmReserveSpace

4.5.1. NAME

srmReserveSpace - facilitates negotiation of space reservation.

4.5.2. SYNOPSIS

In	Out

String userID String authorizationID String storageSystemInfo EnumRetentionQualityMode retentionMode Int expectedFileSize[] String userSpaceDescription Int sizeOfTotalSpaceDesired Int sizeOfGuaranteedSpaceDesired Int lifetimeOfSpaceToReserve	ReturnRequestStatus { EnumStatusCode statusCode, string explanation, EnumErrorCode errorCode } returnStatus String spaceToken EnumRetentionQualityMode retentionMode Int sizeOfTotalReservedSpace Int sizeOfGuaranteedReservedSpace Int lifetimeOfReservedSpace

4.5.3. DESCRIPTION

srmReserveSpace facilitates negotiation of space reservation.

- String *userID* (required)
  User authentication identifier. See Section II for notes.
- String *authorizationID*
  User authorization information. The *authorizationID* may be NULL. See Section II for notes.
- String *storageSystemInfo*
  String containining information specific to the underlying storage system. The *storageSystemInfo* may be NULL. See Section II for notes.
- EnumRetentionQualityMode *retentionMode* (required)
  Type of space to reserve in retention quality.
- Int *expectedFileSize[]*
  A hint that SRM server can use to reserve consecutive storage sizes for the request.
- String *userSpaceDescription*
  String containining description of the space. The description will be used to retrieve *spaceTokens* later with *srmGetSpaceTokens*(). The *userSpaceDescription* may be NULL.
- Int *sizeOfTotalSpaceDesired*
  Desired total space size in bytes. Default is 0 for space size that SRM assigns by default.
- Int *sizeOfGuaranteedSpaceDesired*
  The guaranteed space size in bytes that client needs to work at the minimum. Default is 0 (zero) for the space size that SRM guarantees by default .
- Int *lifetimeOfSpaceToReserve*
  Desired life time of the space in seconds. Default is 0 (zero) for the lifetime that SRM assigned by default.
- ReturnRequestStatus *returnStatus* (required)
  Output parameter reporting the success or failure of the request. In case of failure, the associated *errorCode* is returned.
- String *spaceToken*
  Output parameter string token is associated with the request for the later asynchronous space related request. *spaceToken* may be NULL, in case space reservation is failed for the client.
- EnumRetentionQualityMode *retentionMode*

Output parameter reporting the space type in retention quality that SRM server reserved upon the successful request.

- Int *sizeOfTotalReservedSpace*
  Output parameter reporting the size of the total space that SRM server reserved upon the successful request.
- Int *sizeOfGuaranteedReservedSpace*
  Output parameter reporting the size of the guaranteed space that SRM server reserved upon the successful request.
- Int *lifetimeOfReservedSpace*
  Output parameter reporting the life time of the space that SRM server reserved upon the successful request.

4.5.4. RETURN CODE

On successful abort, the *statusCode* is set to SRM_SUCCESS.
On failure, the *statusCode* is set to SRM_FAILURE and the *errorCode* is set.

4.5.5. ERROR CODE

When status is failure, *errorCode* is set to one of the following:

SRM_AUTHENTICATION_FAILURE
- SRM server failed to authenticate the client

SRM_AUTHORIZATION_FAILURE
- client is not authorized to reserve space

SRM_NO_USER_SPACE
- SRM server does not have enough space for the client to fulfill the request

SRM_NO_FREE_SPACE
- SRM server does not have enough free space to fulfill the request

4.5.6. NOTES on the Behavior

a) *lifetimeOfSpaceToReserve* is not needed when requesting permanent space. It is ignored for permanent space.
b) An SRM may provide default size and lifetime if not supplied.
c) *storageSystemInfo* is optional and used for the case that the storage system requires an additional security check.
d) If *sizeOfTotalSpaceDesired* is not specified, the SRM must return its default quota.
e) The difference between *sizeOfTotalReservedSpace* and *sizeOfGuaranteedReservedSpac* is on best effort basis.

4.5.7. SEE ALSO

srmCleanupFilesFromSpace, srmGetSpaceMetaData, srmGetSpaceTokens, srmReleaseSpace, srmUpdateSpace

4.6. srmStatusOfCleanupFilesFromSpace

4.6.1. NAME

srmStatusOfCleanupFilesFromSpace – is the status call for srmCleanupFilesFromSpace.

4.6.2. SYNOPSIS

In	Out
String userID	String spaceToken
String authorizationID	ReturnRequestStatus {
String requestToken	EnumStatusCode statusCode,
Int offset	string explanation,
Int count	EnumErrorCode errorCode
	} returnStatus
	ReturnSURLStatus {
	String SURL,
	EnumStatusCode statusCode,
	string explanation,
	EnumErrorCode errorCode
	} returnSURLStatus []

4.6.3. DESCRIPTION

srmStatufOfCleanupFilesFromSpace is the status call for srmCleanupFilesFromSpace. srmCleanupFilesFromSpace releases all files from the space associated with the space token. The spaces of released files can be used when space is needed. The space is not released. See also 4.1.3.

- String *userID* (required)
  User authentication identifier. See Section II for notes.
- String *authorizationID*
  User authorization information. The *authorizationID* may be NULL. See Section II for notes.
- String request*Token* (required)
  A token associated with the previously submitted *srmCleanupFilesFromSpace* request. The *requestToken* was returned by the function initiating the request.
- Int *offset*
  Integer indicator for long listed responses that the listing starts from the *offset*. Default is 0 for the first entry.
- Int *count*
  Integer indicator for long listed responses that the listing contains the number of *count*. Default 0 (zero) indicates to return all entries of the list.
- String *spaceToken* (required)
  Output parameter string token is associated with the space that client requested to clean up the space.
- ReturnRequestStatus *returnStatus* (required)
  Output parameter reporting the success or failure of the request. In case of failure, the associated *errorCode* is returned.
- ReturnSURLStatus *returnSURLStatus*[]
  Output parameter reporting the status of each file in the request. In case of failure, the associated *errorCode* is returned. *returnSURLStatus*[] may be empty and NULL. If returned to the client, *SURL* and its *statusCode* are required to return.

- String *SURL* (required)
  SURL that client has requested to clean up.

4.6.4. RETURN CODE
On successful abort, the *statusCode* is set to SRM_SUCCESS.
On failure, the *statusCode* is set to SRM_FAILURE and the *errorCode* is set.

4.6.5. ERROR CODE
When status is failure, *errorCode* is set to one of the following:

For request level resturnStatus,
SRM_AUTHENTICATION_FAILURE
- SRM server failed to authenticate the client

SRM_AUTHORIZATION_FAILURE
- client is not authorized to request the status of the request that is associated with *requestToken*

SRM_TOO_MANY_RESULTS
- Request produced too many results that SRM server cannot handle, and *offset* and *count* cannot fit the results to return.

For file level returnStatus,
SRM_AUTHORIZATION_FAILURE
- Client is not authorized to clean up *SURL* in the space that is associated with *spaceToken*
- Client is not authorized to remove SURL after the clean up, if *remove* option was set.

4.6.6. SEE ALSO
srmCleanupFilesFromSpace, srmReleaseFiles, srmReleaseRequestedFiles, srmReleaseSpace, srmUpdateSpace, srmGetSpaceTokens, srmGetSpaceMetaData

4.7. srmUpdateSpace

4.7.1. NAME
srmUpdateSpace - is to resize the space and/or extend the lifetime of a space.

4.7.2. SYNOPSIS

In	**Out**
String userID	ReturnRequestStatus {
String authorizationID	EnumStatusCode statusCode,
String storageSystemInfo	string explanation,
String spaceToken	EnumErrorCode errorCode
Int newSizeOfTotalSpaceDesired	} returnStatus
Int newSizeOfGuaranteedSpaceDesired	String spaceToken
Int newLifetime	EnumRetentionQualityMode retentionMode
	Int sizeOfTotalSpace

	Int sizeOfGuaranteedSpace Int lifetimeGranted

4.7.3. DESCRIPTION

srmUpdateSpace is to resize the space and/or extend the lifetime of a space.

- String *userID* (required)
  User authentication identifier. See Section II for notes.
- String *authorizationID*
  User authorization information. The *authorizationID* may be NULL. See Section II for notes.
- String *storageSystemInfo*
  String containining information specific to the underlying storage system. The *storageSystemInfo* may be NULL. See Section II for notes.
- String *spaceToken* (required)
  A token associated with the space to update. The *spaceToken* is acquired separately (e.g. *srmReserveSpace*).
- Int *newSizeOfTotalSpaceDesired*
  Desired total space size in bytes. Default is 0 for space size that SRM assigns by default.
- Int *newSizeOfGuaranteedSpaceDesired*
  The guaranteed space size in bytes that client needs to work at the minimum. Default is 0 (zero) for the space size that SRM guarantees by default .
- Int *newLifetime*
  Desired life time of the space in seconds. Default is 0 (zero) for the lifetime that SRM assigned by default.
- ReturnRequestStatus *returnStatus* (required)
  Output parameter reporting the success or failure of the request. In case of failure, the associated *errorCode* is returned.
- String *spaceToken*
  Output parameter string token is associated with the space request for the later asynchronous space related request. *spaceToken* may be NULL, and it is assumed that the output parameter *spaceToken* is the same as the input parameter *spaceToken*. *spaceToken* may be the same as the input parameter *spaceToken*. Some SRM server may require to assign a new *spaceToken*.
- EnumRetentionQualityMode *retentionMode*
  Output parameter reporting the space type in retention quality mode that SRM server reserved upon the successful update request.
- Int *sizeOfTotalSpace*
  Output parameter reporting the new size of the total space that SRM server has upon the successful update request.
- Int *sizeOfGuaranteedSpace*

Output parameter reporting the new size of the guaranteed space that SRM server has upon the successful update request.

- Int *lifetimeGranted*

  Output parameter reporting the new life time of the space that SRM server has upon the successful update request.

4.7.4. RETURN CODE

On successful abort, the *statusCode* is set to SRM_SUCCESS.
On failure, the *statusCode* is set to SRM_FAILURE and the *errorCode* is set.

4.7.5. ERROR CODE

When status is failure, *errorCode* is set to one of the following:

SRM_AUTHENTICATION_FAILURE
- SRM server failed to authenticate the client

SRM_AUTHORIZATION_FAILURE
- client is not authorized to reserve space

SRM_INVALID_REQUEST
- new size or lifetime is not requested

SRM_NO_USER_SPACE
- SRM server does not have enough space for the client to fulfill the request

SRM_NO_FREE_SPACE
- SRM server does not have enough free space to fulfill the request

4.7.6. NOTES on the Behavior

a) Function call must include size and/or lifetime.
b) If neither size nor lifetime is supplied in the input, then return will be an error.
c) New size is the new actual size of the space, and it must be positive.
d) *newLifetime* is the new lifetime requested regardless of the previous lifetime, and it must be positive. It may even be shorter than the remaining lifetime at the time of the call.

5. Advanced feature set 3 : Directory Management Functions

summary:

srmCp
srmCpStatus
srmLsDetails
srmLsDetailsStatus
srmMkdir
srmMkdirStatus
srmMv
srmMvStatus
srmRmdir

details:

5.1. srmCp

5.1.1. NAME

srmCp - is to copy SRM's local file to another space in the same SRM.

5.1.2. SYNOPSIS

In	Out
String userID String authorizationID String fromStorageSystemInfo String toStorageSystemInfo String userRequestDescription Int totalRetryTime EnumOverwriteMode overwriteOption Boolean performChecksum CpFileRequest { String fromSURL, String toSURL, Int knownSize int fileLifetime, EnumFileStorageType toFileStorageType, String fromStorageSystemInfo, String toStorageSystemInfo, String spaceToken, Boolean isSourceADirectory, Boolean allLevelRecursive, int numOfLevels } cpRequests[] String spaceToken EnumFileStorageType toFileStorageType	String requestToken ReturnRequestStatus { EnumStatusCode statusCode, string explanation, EnumErrorCode errorCode } returnStatus

5.1.3. DESCRIPTION

srmCp is to copy SRM's local file to another space in the same SRM.

- String *userID* (required)
  User authentication identifier. See Section II for notes.
- String *authorizationID*
  User authorization information. The *authorizationID* may be NULL. See Section II for notes.
- String *fromStorageSystemInfo*
  String containining information specific to the source storage system that is associated with the *fromSURL*s. The *fromStorageSystemInfo* may be NULL. See Section II for notes.
- String *toStorageSystemInfo*
  String containining information specific to the target storage system that is associated with the *toSURL*s or *toFileStorageType*. The *toStorageSystemInfo* may be NULL. See Section II for notes.
- String *userRequestDescription*
  String description of the request. It may be used for later retrieval of the rquest token.
- Int *totalRetryTime*
  totalRetryTime represents the length of time in seconds that the SRM will try to copy file whose transfer previously failed. This action takes place after all the other file transfers for the request completed. Default is 0 that SRM assigns by default.
- EnumOverwriteMode *overwriteOption*
  SRM needs to copy the files according to the *overwriteOption* on the target that SRM copies files into.
- Boolean *performChecksum*
  performChecksum indicates that checksum calculation for all files in the request needs to be performed when files get copied into its designated target space. Default is false.
- CpFileRequest *cpRequests*[] (required)
  Input parameter copy*Request* consists of SURL information that client wants to copy from one site to another.
  - String *fromSURL* (required)
    Source SURL
  - String *toSURL* (required)
    Target SURL
  - Int *knownSize*
    File size of the SURL if known. Default is 0, indicating unknown.
  - Int *fileLifetime*
    Desired life time of the file in seconds once when it's in the target SRM. Default is 0, and SRM assigns default lifetime.
  - EnumFileStorageType *toFileStorageType*
    toFileStorageType indicates which type of storage that *fromSURL*s are copied into the *toSURL*.
  - String *fromStorageSystemInfo*

String containining information specific to the underlying storage system associated with the *fromSURL*. The *fromStorageSystemInfo* may be NULL. See Section II for notes

- String *toStorageSystemInfo*
  String containining information specific to the underlying storage system associated with the *toSURL* or *toFileStorageType*. The *toStorageSystemInfo* may be NULL. See Section II for notes
- String *spaceToken*
  An advanced option when Space Management feature is supported. A token associated with the space if a particular space in file storage type is to be used. The spaceToken is acquired separately (e.g. *srmReserveSpace*).
- Boolean *isSourceADirectory* (advanced option)
  Boolean indicator if *fromSURL* is a directory. Default is false.
- Boolean *allLevelRecursive* (advanced option)
  Boolean indicator if *fromSURL* is a directory and all files under the *fromSURL* must be retrieved. The corresponding target directory structure must be hierachically created according to the source directory structure. Defautl is false.
- Int *numOfLevels* (advanced option)
  Positive integer indicator for how many levels of the recursive directory must be browsed and all files in those directories to be retrieved. Default is 0 for the single level of directory (no recursive).

- String *spaceToken*
  An advanced option when Space Management feature is supported. A token associated with the space if a particular space in file storage type is to be used. The *spaceToken* is acquired separately (e.g. *srmReserveSpace*). This is request level option. If file level option (*spaceToken*) in the *cpRequests* is set, it takes the priority.
- EnumFileStorageType *toFileStorageType*
  toFileStorageType indicates which type of storage that *fromSURL*s are copied into. This is request level option. If file level option (*toFileStorageType*) in the *cpRequests* is set, it takes the priority.
- String *requestToken*
  Output parameter string token is associated with the request for the later asynchronous status request. *requestToken* may be NULL, in case *srmCp* is processed without delay.
- ReturnRequestStatus *returnStatus* (required)
  Output parameter reporting the success or failure of the request. In case of failure, the associated *errorCode* is returned.

5.1.4. RETURN CODE

On successful abort, the *statusCode* is set to SRM_SUCCESS.
On failure, the *statusCode* is set to SRM_FAILURE and the *errorCode* is set.

5.1.5. ERROR CODE

When status is failure, *errorCode* is set to one of the following:

SRM_AUTHENTICATION_FAILURE

- SRM server failed to authenticate the client

SRM_AUTHORIZATION_FAILURE
- client is not authorized to submit the *srmRemoteCopy* request

SRM_INVALID_REQUEST
- input parameters do not conform the SRM. For example, client requested 10,000 *totalRetryTime* and SRM cannot honor the number.

SRM_NOT_SUPPORTED
- SRM server does not support the given input parameters. For example, client requested to use *allLevelRecursive*, but SRM cannot support the directory management feature.

5.1.6. NOTES on the Behavior

a) Output parameter *requestToken* is optional, when the copying files are small enough to be handled at once.
b) *srmCp* does not support remote copy, but only local copies; from a local location to another local location.
c) Same amount of space as the source files is required to copy to the target.
d) There is no protocol negotiation with the client for the request.
e) *totalRetryTime* means that if all the file transfers for the request are complete, then try previously failed transfers for a time period of *totalRetryTime*.
f) In case that the retries fail, the return should include an explanation of why the retries failed.
g) Empty directories are copied as well.
h) The default value of lifetime for volatile or durable file types must be equal to or less than the lifetime left in the space of the corresponding *spaceToken*. The default value of fileType is "volatile".

5.1.7. NOTES on the Advanced Behavior with Space Management Feature

a) *spaceToken* must be a valid input parameter.
b) *spaceToken* or *toFileStorageType* or both must be provided. When *spaceToken* is not provided and *toFileStorageType* is provided, space allocation is needed in the *toFileStorageType*.

5.1.8. SEE ALSO

srmRemoteCopy

5.2. srmCpStatus

5.2.1. NAME

srmCpStatus - is a status call for srmCp.

5.2.2. SYNOPSIS

In	Out	
String userID	ReturnRequestStatus {	
String authorizationID	EnumStatusCode	statusCode,
String requestToken	string	explanation,
Int offset	EnumErrorCode	errorCode
Int count	} returnStatus	

RequestStatusForCopy {
String fromSURL,
String toSURL,
EnumFileStorageType toFileStorageType,
Int fileSize,
EnumFileType fileType,
Int fileLifetime,
EnumStatusCode statusCode,
String explanation,
EnumErrorCode errorCode,
String spaceToken
} returnCpStatus[]
Boolean partialList
Int totalFilesInTheRequest

5.2.3. DESCRIPTION
srmCpStatus is a status call for srmCp.

- String *userID* (required)
  User authentication identifier. See Section II for notes.
- String *authorizationID*
  User authorization information. The *authorizationID* may be NULL. See Section II for notes.
- String *requestToken* (required)
  A token associated with the previously submitted *srmCp* request. The *requestToken* was returned by the function initiating the request.
- Int *offset*
  Integer indicator for long listed responses that the listing starts from the *offset*. Default is 0 for the first entry.
- Int *count*
  Integer indicator for long listed responses that the listing contains the number of *count*. Default 0 (zero) indicates to return all entries of the list.
- ReturnRequestStatus *returnStatus* (required)
  Output parameter reporting the success or failure of the request. In case of failure, the associated *errorCode* is returned.
- ReturnStatusForCopy *returnCpStatus*[]
  Output parameter reporting the status of each file in the request. In case of failure, the associated *errorCode* is returned. *returnCpStatus*[] may be empty and NULL. If returned to the client, *statusCode, fromSURL, toSURL* and *toFileStorageType* are required to return.
  - String *fromSURL* (required)
    Source SURL.
  - String *toSURL* (required)
    Target SURL that SRM server copied the *fromSURL* into.
  - EnumFileStorageType *toFileStorageType*
    File storage type of *toSURL*.

- Int *fileSize*
  File size in bytes for *toSURL*.
- EnumFileType *fileType*
  File type of *toSURL*.
- Int *fileLifetime*
  Integer lifetime in seconds that is assigned to the *toSURL*.
- String *checksumType*
  Checksum type if check is performed. For example, MD5.
- String *checksumValue*
  Checksum value if check is performed.
- String *spaceToken*
  An advanced option when Space Management feature is supported. A token associated with the space if a particular space in file storage type is to be used for *toSURL*.

- Boolean partialList
  Output parameter indicating if return values are partial or not. If true, *requestToken* must be returned. Default is false.
- Int totalFilesInTheRequest
  Output parameter indicating how many files in the request for a hint to the asynchronous status cals.

5.2.4. RETURN CODE

On successful abort, the *statusCode* is set to SRM_SUCCESS.
On failure, the *statusCode* is set to SRM_FAILURE and the *errorCode* is set.
If only partial files were successful, the *statusCode* is set to SRM_PARTIAL_SUCCESS, and the *restunCpStatus* should explain on those failed files.

5.2.5. ERROR CODE

When status is failure, *errorCode* is set to one of the following:

For request level resturnStatus,

SRM_AUTHENTICATION_FAILURE
- SRM server failed to authenticate the client

SRM_AUTHORIZATION_FAILURE
- client is not authorized to request the status of the request that is associated with *requestToken*

SRM_TOO_MANY_RESULTS
- Request produced too many results that SRM server cannot handle, and *offset* and *count* cannot fit the results to return.

SRM_INVALID_REQUEST_TOKEN
- *requestToken* does not refer to an existing request

SRM_INVALID_SPACE_TOKEN
- *spaceToken* does not refer to an existing space

For file level returnStatus,

SRM_INVALID_PATH

- *fromSURL* does not refer to an existing file request

SRM_AUTHORIZATION_FAILURE

- Client is not authorized to copy files from *fromSURL*
- Client is not authorized to copy files into *toSURL*
- Client is not authorized to copy files into space that client provided with *spaceToken* or *toFileStorageType* in *srmCp*

SRM_INVALID_SPACE_TOKEN

- *spaceToken* does not refer to an existing space that is associated with the *toSURL*

5.2.6. NOTES on the Advanced Behavior with Space Management Feature

a) *spaceToken* must be valid.

5.2.7. SEE ALSO

srmRemoteCopy, srmCp, srmGetRequestTokens

5.3. srmLsDetails

5.3.1. NAME

srmLsDetails - returns a list of files in the space with detailed information.

5.3.2. SYNOPSIS

In	**Out**

String userID
String authorizationID
String storageSystemInfo
{ String SURL[] |
EnumFileStorageType fileStorageType |
String spaceToken }
Boolean directoriesOnly
Boolean fullDetailedList
Boolean allLevelRecursive
Int numOfLevels
Int offset
Int count
	String requestToken
	ReturnRequestStatus {
	EnumStatusCode statusCode,
	string explanation,
	EnumErrorCode errorCode
	} returnStatus
	SURLMetaDataDetails {
	String SURL,
	EnumStatusCode statusCode,
	String explanation,
	EnumErrorCode errorCode,
	Int size,
	UTCtime expirationTime,
	UTCtime lastModificationTime,
	EnumFileType fileType,
	EnumAccessLatencyMode latencyMode,
	String checksumType,
	String checksumValue,
	String ownerID,
	EnumPermissionMode ownerPermission,
	UserPermission {
	String userID,
	EnumPermissionMode permission

```
    } userPermission[]
    GroupPermission {
        String groupID,
        EnumPermissionMode permission
    } groupPermission[]
    EnumPermissionMode otherPermission,
    SURLMetaData[] subpath
} returnSURLInfo[]
Boolean partialList
Int totalFileInTheRequest
```

5.3.3. DESCRIPTION

srmLsDetails returns a list of files in the space with detailed information.

- String *userID* (required)
  User authentication identifier. See Section II for notes.
- String *authorizationID*
  User authorization information. The *authorizationID* may be NULL. See Section II for notes.
- String *storageSystemInfo*
  String containining information specific to the underlying storage system. The *storageSystemInfo* may be NULL. See Section II for notes.
- String *SURL*[], or
  EnumFileStorageType *fileStorageType*, or
  String *spaceToken* (advanced option when Space Management feature is supported)
  Only one of these three parameters are required. *SURL* refers to files only. As an advanced behavior when space management feature is supported, *spaceToken* may be provided.
- Boolean *directoriesOnly*
  Boolean indicator for directory listing only. Default is false.
- Boolean *fullDetailedList*
  Boolean indicator for all possible information about the request to be returned. Default is false.
- Boolean *allLevelRecursive*
  Boolean indicator for recursive directory listing. Default is false.
- Int *numOfLevels*
  Positive integer indicator for how many levels of the recursive directory listing to be performed. Default is 0 for the single level of directory listing (no recursive).
- Int *offset*
  Integer indicator for long listed responses that the listing starts from the *offset*. Default is 0 for the first entry.
- Int *count*
  Integer indicator for long listed responses that the listing contains the number of *count*. Default 0 (zero) indicates to return all entries of the list.
- String *requestToken*

Output parameter string token is associated with the request for the later asynchronous status request. *requestToken* may be NULL, in case *srmLsDetails* is processed without delay.

- ReturnRequestStatus *returnStatus* (required)
  Output parameter reporting the success or failure of the request. In case of failure, the associated *errorCode* is returned.
- SURLMetaDataDetails *returnSURLInfo*[]
  Output parameter reporting the information of each file in the request. In case of failure, the associated *errorCode* is returned. *returnSURLInfo*[] may be empty and NULL. If returned to the client, *SURL, size,* and its *fileType* are required to return.
  - String *SURL* (required)
    SURL that client has requested for the info.
  - EnumStatusCode *statusCode*
    statusCode of the *SURL*.
  - String *explanation*
    String explanation of the *statusCode* for the *SURL* in case of not successful.
  - EnumErrorCode *errorCode*
    In case of failure, *errorCode* is expected to be returned for the *SURL*.
  - Int *size* (required)
    Size of the *SURL*. In case of SURL being a directory, *size* is expected to be 0.
  - UTCtime *expirationTime*
    Expiration time that is associated with the *SURL*.
  - UTCtime *lastModificationTime*
    Last modified time that is associated with the *SURL*.
  - EnumFileType *fileType* (required)
    File type associated with the *SURL*.
  - EnumAccessLatencyMode *latencyMode*
    Access latency mode that is associated with the *SURL*.
  - String *checksumType*
    Checksum type if check is performed. For example, MD5.
  - String *checksumValue*
    Checksum value if check is performed.
  - String *ownerID*
    ID of the file owner that is associated with the *SURL*.
  - EnumPermissionMode *ownerPermission*
    Owner permissions that is associated with the *SURL*.
  - UserPermission *userPermission*[]
    User permissions that is associated with the *SURL*.
    - String *userID*
      User ID that is associated with the *SURL*.
    - EnumPermissionMode p*ermission*
      Permissions for the *userID* on *SURL*.
  - GroupPermission *groupPermission*[]
    Group permissions that is associated with the *SURL*.
    - String *groupID*
      Group ID that is associated with the *SURL*.

        - EnumPermissionMode p*ermission*
          Permissions for the *groupID* on *SURL*.
    - EnumPermissionMode *otherPermission*
      Other permissions that is associated with the *SURL*.
    - SURLMetaData *subpath*[]
      In case of the recursive directory *SURL* listing, *subpath* indicates sub-directory that contains files or directoties.
- Boolean *partialList*
  Output parameter reporting if the returned *returnSURLInfo*[] is a partial list. Default is false. If the *returnSURLList*[] is a partial list and *partialList* is true, *requestToken* is required to be returned, and client is expected to request its status again with *srmLsDetailsStatus*().
- Int totalFilesInTheRequest
  Output parameter indicating how many files in the request for a hint to the asynchronous status cals.

5.3.4. RETURN CODE

On successful abort, the *statusCode* is set to SRM_SUCCESS.
On failure, the *statusCode* is set to SRM_FAILURE and the *errorCode* is set.
If only partial files were successful, the *statusCode* is set to SRM_PARTIAL_SUCCESS, and the *restunSURLInfo* should explain on those failed files.

5.3.5. ERROR CODE

When status is failure, *errorCode* is set to one of the following:

For request level resturnStatus,

SRM_AUTHENTICATION_FAILURE
- SRM server failed to authenticate the client

SRM_AUTHORIZATION_FAILURE
- client is not authorized to retrieve detailed information

SRM_INVALID_SPACE_TOKEN
- *spaceToken* does not refer to an existing space in SRM

SRM_TOO_MANY_RESULTS
- srmLsDetails request has generated too many results that SRM cannot handle. In most cases, it needs to be narrowed down with offset and count by the client.

For file level returnStatus,

SRM_INVALID_PATH
- *SURL* does not refer to an existing file path

SRM_AUTHORIZATION_FAILURE
- client is not authorized to view the *SURL* or to access the directory or sub-directories

5.3.6. NOTES on the Behavior

a) Either *SURL*s, *spaceToken* or *fileStorageType* is needed.

b) If output parameter *partialList* is true, it indicates that only part of the result was returned. In this case, a *requestToken* is returned.
c) If the entire result is returned, then output parameter *requestToken* is optional.
d) *fullDetailedList* is false by default. In this case, *SURL*, *size* and *fileType* are returned in the status.
e) If *fullDetailedList* is true, additional items may be returned by the SRM.
f) When listing for a particular type specified by *fileStorageType*, all the files of that type must be returned.
g) *SURL*s may refer to either directory or file.
h) Files are returned in width first order.
i) List of directories come before list of files in the return order.
j) If *SURL* refers to a directory, the returned value size must be 0.
k) If *allLevelRecursive* is true, it dominates, i.e. ignore numOfLevels.
l) If *allLevelRecursive* is false or missing, then *numOfLevels* must be honored. If numOfLevels is 0 (zero) or missing, a single level is assumed.
m) Directory names are returned even if they are empty.

5.3.7. NOTES on the Advanced Behavior with Space Management Feature

a) If *spaceToken* is provided, all files that belong to the *spaceToken* are returned.
b) If *spaceToken* is not provided and *fileStorageType* is provided, then all the files in each space of that type must be returned.

5.3.8. SEE ALSO

srmLs, srmLsStatus, srmLsDetailsStatus

5.4. srmLsDetailsStatus

5.4.1. NAME

srmLsDetailsStatus - is an asynchronous call for srmLsDetails that is large.

5.4.2. SYNOPSIS

In	Out

String userID
String authorizationID
String requestToken
Int offset
Int count
	ReturnRequestStatus {
	EnumStatusCode statusCode,
	string explanation,
	EnumErrorCode errorCode
	} returnStatus
	SURLMetaData {
	String SURL,
	EnumStatusCode statusCode,
	String explanation,
	EnumErrorCode errorCode,
	Int size,
	UTCtime expirationTime,
	UTCtime lastModificationTime,
	EnumFileType fileType,

	EnumAccessLatencyMode latencyMode,
	String checksumType,
	String checksumValue,
	String ownerID,
	EnumPermissionMode ownerPermission,
	UserPermission {
	String userID,
	EnumPermissionMode userPermission
	}[]
	GroupPermission {
	String groupID,
	EnumPermissionMode groupPermission
	}[]
	EnumPermissionMode otherPermission,
	MetaDataDetails[] subpath
	} returnSURLInfo[]

5.4.3. DESCRIPTION

srmLsDetailsStatus is an asynchronous call for srmLsDetails that is large.

- String *userID* (required)
  User authentication identifier. See Section II for notes.
- String *authorizationID*
  User authorization information. The *authorizationID* may be NULL. See Section II for notes.
- String *requestToken* (required)
  A token associated with the previously submitted *srmLsDetails* request.
- Int *offset*
  Integer indicator for long listed responses that the listing starts from the *offset*. Default is 0 for the first entry.
- Int *count*
  Integer indicator for long listed responses that the listing contains the number of *count*. Default 0 (zero) indicates to return all entries of the list.
- ReturnRequestStatus *returnStatus* (required)
  Output parameter reporting the success or failure of the request. In case of failure, the associated *errorCode* is returned.
- SURLMetaDataDetails *returnSURLInfo*[]
  Output parameter reporting the information of each file in the request. In case of failure, the associated *errorCode* is returned. *returnSURLInfo*[] may be empty and NULL. If returned to the client, *SURL*, *size*, and its *fileType* are required to return.
  - String *SURL* (required)
    SURL that client has requested for the info.
  - EnumStatusCode *statusCode*
    statusCode of the *SURL*.
  - String *explanation*
    String explanation of the *statusCode* for the *SURL* in case of not successful.

- EnumErrorCode *errorCode*
  In case of failure, *errorCode* is expected to be returned for the *SURL*.
- Int *size* (required)
  Size of the *SURL*. In case of SURL being a directory, *size* is expected to be 0.
- UTCtime *expirationTime*
  Expiration time that is associated with the *SURL*.
- UTCtime *lastModificationTime*
  Last modified time that is associated with the *SURL*.
- EnumFileType *fileType* (required)
  File type associated with the *SURL*.
- EnumAccessLatencyMode *latencyMode*
  Access latency mode that is associated with the *SURL*.
- String *checksumType*
  Checksum type if check is performed. For example, MD5.
- String *checksumValue*
  Checksum value if check is performed.
- String *ownerID*
  ID of the file owner that is associated with the *SURL*.
- EnumPermissionMode *ownerPermission*
  Owner permissions that is associated with the *SURL*.
- UserPermission *userPermission*[]
  User permissions that is associated with the *SURL*.
  - String *userID*
    User ID that is associated with the *SURL*.
  - EnumPermissionMode p*ermission*
    Permissions for the *userID* on *SURL*.
- GroupPermission *groupPermission*[]
  Group permissions that is associated with the *SURL*.
  - String *groupID*
    Group ID that is associated with the *SURL*.
  - EnumPermissionMode p*ermission*
    Permissions for the *groupID* on *SURL*.
- EnumPermissionMode *otherPermission*
  Other permissions that is associated with the *SURL*.
- SURLMetaData *subpath*[]
  In case of the recursive directory *SURL* listing, *subpath* indicates sub-directory that contains files or directoties.

5.4.4. RETURN CODE

On successful abort, the *statusCode* is set to SRM_SUCCESS.
On failure, the *statusCode* is set to SRM_FAILURE and the *errorCode* is set.
If only partial files were successful, the *statusCode* is set to SRM_PARTIAL_SUCCESS, and the *restunSURLInfo* should explain on those failed files.

5.4.5. ERROR CODE

When status is failure, *errorCode* is set to one of the following:

For request level resturnStatus,

SRM_AUTHENTICATION_FAILURE
- SRM server failed to authenticate the client

SRM_AUTHORIZATION_FAILURE
- client is not authorized to retrieve detailed information on the request that is associated with the *requestToken*

SRM_INVALID_REQUEST_TOKEN
- *requestToken* does not refer to an existing request

SRM_TOO_MANY_RESULTS
- srmLsDetails request has generated too many results that SRM cannot handle. In most cases, it needs to be narrowed down with offset and count by the client.

For file level returnStatus,

SRM_INVALID_PATH
- *SURL* does not refer to an existing file path

SRM_AUTHORIZATION_FAILURE
- client is not authorized to view the *SURL* or to access the directory or sub-directories

5.4.6. SEE ALSO

srmLs, srmLsStatus, srmLsDetails

5.5. srmMkdir

5.5.1. NAME

srmMkdir - create a directory in a local SRM space.

5.5.2. SYNOPSIS

In	Out
String userID String authorizationID String storageSystemInfo String SURL	ReturnRequestStatus { EnumStatusCode statusCode, string explanation, EnumErrorCode errorCode } returnStatus

5.5.3. DESCRIPTION

srmMkdir create a directory in a local SRM space.

- String *userID* (required)
  User authentication identifier. See Section II for notes.
- String *authorizationID*
  User authorization information. The *authorizationID* may be NULL. See Section II for notes.
- String *storageSystemInfo*

String containining information specific to the underlying storage system. The *storageSystemInfo* may be NULL. See Section II for notes.

- String *SURL* (required)
  SURL to create as a directory.
- ReturnRequestStatus *returnStatus* (required)
  Output parameter reporting the success or failure of the request. In case of failure, the associated *errorCode* is returned.

5.5.4. RETURN CODE

On successful abort, the *statusCode* is set to SRM_SUCCESS.
On failure, the *statusCode* is set to SRM_FAILURE and the *errorCode* is set.

5.5.5. ERROR CODE

When status is failure, *errorCode* is set to one of the following:

SRM_AUTHENTICATION_FAILURE
- SRM server failed to authenticate the client

SRM_AUTHORIZATION_FAILURE
- client is not authorized to create a directory as *SURL*

SRM_INVALID_PATH
- *SURL* does not refer to a valid path

5.5.6. NOTES on the Behavior

a) Recursive creation of directories is not supported.

5.5.7. SEE ALSO

srmLs, srmLsDetails, srmRm, srmRmdir

5.6. srmMv

5.6.1. NAME

srmMv - is to move a file from one local directory to another, or from one space token to another space token.

5.6.2. SYNOPSIS

In	**Out**	
String userID	String requestToken	
String authorizationID	ReturnRequestStatus {	
String fromStorageSystemInfo	EnumStatusCode	statusCode,
String toStorageSystemInfo	string	explanation,
String fromSURL	EnumErrorCode	errorCode
{ String toSURL |	} returnStatus	
String toFileStorageType }		
String toSpaceToken		

5.6.3. DESCRIPTION

srmMv is to move a file from one local directory to another, or from one space token to another space token.

- String *userID* (required)
  User authentication identifier. See Section II for notes.
- String *authorizationID*
  User authorization information. The *authorizationID* may be NULL. See Section II for notes.
- String *fromStorageSystemInfo*
  String containining information specific to the source storage system that is associated with the *fromSURL*. The *fromStorageSystemInfo* may be NULL. See Section II for ntoes.
- String *toStorageSystemInfo*
  String containining information specific to the target storage system that is associated with the *toSURL* or *toSpaceToken*. The *toStorageSystemInfo* may be NULL. See Section II for ntoes.
- String *fromSURL* (required)
  SURL to move from.
- String *toSURL* or
  String *toFileStorageType*
  SURL or target file storage type to move *fromSURL* to. Either *toSURL* or *toFileStorageType* is needed.
- String *toSpaceToken*
  Space token that is associated with the *toSURL*
- String *requestToken*
  Output parameter string token is associated with the request for the later asynchronous status request. *requestToken* may be NULL, in case *srmMv* is processed without delay.
- ReturnRequestStatus *returnStatus* (required)
  Output parameter reporting the success or failure of the request. In case of failure, the associated *errorCode* is returned.

5.6.4. RETURN CODE

On successful abort, the *statusCode* is set to SRM_SUCCESS.
On failure, the *statusCode* is set to SRM_FAILURE and the *errorCode* is set.

5.6.5. ERROR CODE

When status is failure, *errorCode* is set to one of the following:

SRM_AUTHENTICATION_FAILURE
- SRM server failed to authenticate the client

SRM_AUTHORIZATION_FAILURE
- client is not authorized to move *fromSURL*.
- Client is not authorized to move a file into *toSURL*

SRM_INVALID_PATH
- *fromSURL* or *toSURL* does not refer to an existing path

SRM_NO_USER_SPACE

- client owned destination space cannot accommodate *toSURL*

SRM_NO_FREE_SPACE
- *toSURL* cannot be moved into the destination space where SRM server can allocate freely

SRM_INVALID_SPACE_TOKEN
- *toSpaceToken* does not refer to an existing space

5.6.6. NOTES on the Behavior

a) The output parameter *requestToken* is optional, when moving the file is fast enough to be returned at once.
b) *SURL* may be applied to both directory and file.
c) When a file is moved from one directory to another, *toSURL* must be provided.

5.6.7. NOTES on the Advanced Behavior with Space Management Feature

a) When a file is moved from one space token to another, *toSURL* is not required, but *toSpaceToken* must be provided.

5.6.8. NOTES on the Advanced Behavior with AuthorizationFeature

a) Authorization checks need to be performed on both *fromSURL* and *toSURL*.

5.6.9. SEE ALSO

srmLs, srmLsDetails, srmRm, srmCp, srmMvStatus, srmMkdir, srmRmdir

5.7. srmMvStatus

5.7.1. NAME

srmMvStatus - is the status call for srmMv.

5.7.2. SYNOPSIS

In	**Out**
String userID String authorizationID String requestToken	ReturnRequestStatus { EnumStatusCode statusCode, string explanation, EnumErrorCode errorCode } returnStatus

5.7.3. DESCRIPTION

srmMvStatus is the status call for srmMv.

- String *userID* (required)
  User authentication identifier. See Section II for notes.
- String *authorizationID*
  User authorization information. The *authorizationID* may be NULL. See Section II for notes.
- String *requestToken* (required)
  A token associated with the previously submitted *srmMv* request.

- ReturnRequestStatus *returnStatus* (required)
  Output parameter reporting the success or failure of the request. In case of failure, the associated *errorCode* is returned.

5.7.4. RETURN CODE
On successful abort, the *statusCode* is set to SRM_SUCCESS.
On failure, the *statusCode* is set to SRM_FAILURE and the *errorCode* is set.

5.7.5. ERROR CODE
When status is failure, *errorCode* is set to one of the following:

SRM_AUTHENTICATION_FAILURE
- SRM server failed to authenticate the client

SRM_AUTHORIZATION_FAILURE
- client is not authorized to move *fromSURL*.
- Client is not authorized to move a file into *toSURL*

SRM_INVALID_EQUEST_TOKEN
- *requestToken* does not refer to an existing request

SRM_INVALID_PATH
- *fromSURL* or *toSURL* does not refer to an existing path

SRM_NO_USER_SPACE
- client owned destination space cannot accommodate *toSURL*

SRM_NO_FREE_SPACE
- *toSURL* cannot be moved into the destination space where SRM server can allocate freely

5.7.6. SEE ALSO
srmLs, srmLsDetails, srmRm, srmCp, srmMv, srmMkdir, srmRmdir

5.8. srmRmdir

5.8.1. NAME
srmRmdir - removes a local empty directory.

5.8.2. SYNOPSIS

In	Out
String userID	ReturnRequestStatus {
String authorizationID	EnumStatusCode statusCode,
String storageSystemInfo	string explanation,
String SURL	EnumErrorCode errorCode
	} returnStatus

5.8.3. DESCRIPTION
srmRmdir removes an empty directory in a local SRM space.

- String *userID* (required)

User authentication identifier. See Section II for notes.

- String *authorizationID*
  User authorization information. The *authorizationID* may be NULL. See Section II for notes.
- String *storageSystemInfo*
  String containining information specific to the underlying storage system. The *storageSystemInfo* may be NULL. See Section II for notes.
- String *SURL* (required)
  SURL to remove as a directory.
- ReturnRequestStatus *returnStatus* (required)
  Output parameter reporting the success or failure of the request. In case of failure, the associated *errorCode* is returned.

5.8.4. RETURN CODE

On successful abort, the *statusCode* is set to SRM_SUCCESS.
On failure, the *statusCode* is set to SRM_FAILURE and the *errorCode* is set.

5.8.5. ERROR CODE

When status is failure, *errorCode* is set to one of the following:

SRM_AUTHENTICATION_FAILURE
- SRM server failed to authenticate the client

SRM_AUTHORIZATION_FAILURE
- client is not authorized to remove a directory as *SURL*

SRM_INVALID_PATH
- *SURL* does not refer to a valid path

SRM_NOT_EMPTY_DIRECTORY
- *SURL* directory is not empty

5.8.6. NOTES on the Behavior

a) *SURL* must be an empty directory only.

5.8.7. SEE ALSO

srmLs, srmLsDetails, srmRm, srmMkdir

6. Advanced feature set 4 : Authorization Functions

summary:

srmCheckPermission
srmSetPermission

details:

6.1. srmCheckPermission

6.1.1. NAME

srmCheckPermission - is used to check the client permissions on the SURLs.

6.1.2. SYNOPSIS

In	Out

String userID
String authorizationID
String storageSystemInfo
String SURL[]
Boolean localCheckOnly
	ReturnRequestStatus {
	EnumStatusCode statusCode,
	string explanation,
	EnumErrorCode errorCode
	} returnStatus
	CheckedPermission {
	String SURL,
	EnumStatusCode statusCode,
	String explanation,
	EnumErrorCode errorCode,
	EnumPermissionMode clientPermission
	} returnPermissionInfo[]

6.1.3. DESCRIPTION

srmCheckPermission is used to check the client permissions on the SURLs. It only checks for the client for authorization on the *SURL*.

- String *userID* (required)
  User authentication identifier. See Section II for notes.
- String *authorizationID*
  User authorization information. The *authorizationID* may be NULL. See Section II for notes.
- String *storageSystemInfo*
  String containining information specific to the underlying storage system. The *storageSystemInfo* may be NULL. See Section II for notes.
- String *SURL[]* (required)
  SURLs to check their permission.
- Boolean *localCheckOnly*
  Boolean indicator for checking local authorization permissions. Default is true.
- ReturnRequestStatus *returnStatus* (required)

Output parameter reporting the success or failure of the request. In case of failure, the associated *errorCode* is returned.

- CheckedPermission *returnPermissionInfo*[]
  Output parameter reporting the permission of each file in the request. In case of failure, the associated *errorCode* is returned. *returnPermissionInfo*[] may be empty and NULL. If returned to the client, *SURL* and its *statusCode* are required to return.
  - String *SURL* (required)
    SURL that client has requested to check the permission.
  - EnumPermissionMode *clientPermission*
    Client's permission information on the *SURL*.

6.1.4. RETURN CODE

On successful abort, the *statusCode* is set to SRM_SUCCESS.
On failure, the *statusCode* is set to SRM_FAILURE and the *errorCode* is set.
If only partial files were successful, the *statusCode* is set to SRM_PARTIAL_SUCCESS, and the *returnPermissionInfo* should explain on those failed files.

6.1.5. ERROR CODE

When status is failure, *errorCode* is set to one of the following:

For request level resturnStatus,

SRM_AUTHENTICATION_FAILURE
- SRM server failed to authenticate the client

SRM_AUTHORIZATION_FAILURE
- client is not authorizaed to request permission information

For file level returnStatus,

SRM_INVALID_PATH
- *SURL* does not refer to an existing path

SRM_AUTHORIZATION_FAILURE
- client is not authorizaed to request permission information on the *SURL*

6.1.6. NOTES on the Behavior

a) The default value of *localCheckOnly* is true, and SRM only checks files in its local space. Otherwise, if a file is not in its local space, then SRM goes to the *SURL* location to check the client permission.

b) If *localCheckOnly* is false, SRM may choose to always check the *SURL* for client permission of each file. It may be okay if SRM choose to check its local cache first.

6.1.7. SEE ALSO

srmLs, srmLsDetails

6.2. srmSetPermission

6.2.1. NAME

srmSetPermission - is to set permission on local SURLs. This is similar to unix style permissions.

6.2.2. SYNOPSIS

In	Out
String userID	ReturnRequestStatus {
String authorizationID	EnumStatusCode statusCode,
String storageSystemInfo	String explanation,
String SURL[]	EnumErrorCode errorCode
EnumPermissionType permissionType	} returnStatus
EnumPermissionTarget permissionTarget	RequestSURLStatus {
EnumPermissionMode permissionMode	String SURL,
String targetID[]	EnumStatusCode statusCode,
Boolean recursive	String explanation.
	EnumErrorCode errorCode
	} returnSURLStatus []

EnumPermissionTarget := OWNER |
USER |
GROUP |
OTHERS

6.2.3. DESCRIPTION

srmSetPermission is to set permission on local SURLs. This is similar to unix style permissions.

- String *userID* (required)
  User authentication identifier. See Section II for notes.
- String *authorizationID*
  User authorization information. The *authorizationID* may be NULL. See Section II for notes.
- String *storageSystemInfo*
  String containining information specific to the underlying storage system. The *storageSystemInfo* may be NULL. See Section II for notes.
- String *SURL*[] (required)
  SURLs to set the permissions.
- EnumPermissionType *permissionType* (required)
  Permission type information. Either *ADD*, *REMOVE*, or *CHANGE* as defined in section 1.
- EnumPermissionTarget *permissionTarget*
  Permission target information. Either *OWNER*, *USER*, *GROUP* or *OTHERS* as defined above.
- EnumPermissionMode *permissionMode*
  Permission mode information.
- String *targetID[]*
  ID for the updated permissions.
- Boolean *recursive*

recursive indicates if the permission updates should be performed recursively on all files under the sub-directories.

- ReturnRequestStatus *returnStatus* (required)
  Output parameter reporting the success or failure of the request. In case of failure, the associated *errorCode* is returned.
- ReturnSURLStatus *returnSURLStatus* []
  Output parameter reporting the status of each file in the request. In case of failure, the associated *errorCode* is returned. *returnSURLStatus* [] may be empty and NULL. If returned to the client, *SURL* and its *statusCode* are required to return.
  - String *SURL* (required)
    SURL that client has requested to set the permissions.

6.2.4. RETURN CODE

On successful abort, the *statusCode* is set to SRM_SUCCESS.
On failure, the *statusCode* is set to SRM_FAILURE and the *errorCode* is set.
If only partial files were successful, the *statusCode* is set to SRM_PARTIAL_SUCCESS, and the *restunSURLStatus* should explain on those failed files.

6.2.5. ERROR CODE

When status is failure, *errorCode* is set to one of the following:

For request level resturnStatus,

SRM_AUTHENTICATION_FAILURE
- SRM server failed to authenticate the client

SRM_AUTHORIZATION_FAILURE
- client is not authorized to set permissions

For file level returnStatus,

SRM_INVALID_PATH
- *SURL* does not refer to an existing path

SRM_AUTHORIZATION_FAILURE
- client is not authorized to set permissions on the *SURL*

6.2.6. NOTES on the Behavior

a) *permissionTarget* refers to who the permission setting is for.
b) When *permissionTarget* is either USER or GROUP, *targetID* is needed.
c) EnumPermissionMode is similar to Unix permission modes.
d) User permissions are not supported in this version for dynamic user-level permission assignment similar to Access Control Lists (ACLs).
e) Permissions must be assigned to a single owner and a single group, similar to unix permission.
f) SRMs do not provide any group operations (setup, modify, remove, etc.).
g) Groups must be assumed to be set up separately, before *srmSetPermission* is used. The owner must be a member of the group.

h) If *EnumPermissionType* is ADD or CHANGE, and *EnumPermissionMode* is null, then it must be assumed that *EnumPermissionMode* is READ only.
i) If *EnumPermissionType* is REMOVE, then the *EnumPermissionMode* is ignored.

6.2.7. NOTES on the Advanced Behavior with Directory Management Feature

a) *SURL* may be either directory or file.
b) When *SURL* is a directory, all files in the directory is set to the new permission.

7. Advanced feature set 5 : Request Administration Functions

summary:

srmResumeRequest
srmSuspendRequest

details:

7.1. srmResumeRequest

7.1.1. NAME

srmResumeRequest - is to resume previously suspended requests.

7.1.2. SYNOPSIS

In	Out
String userID	ReturnRequestStatus {
String authorizationID	EnumStatusCode statusCode,
String requestToken	string explanation,
	EnumErrorCode errorCode
	} returnStatus

7.1.3. DESCRIPTION

srmResumeRequest is to resume previously suspended requests.

- String *userID* (required)
  User authentication identifier. See Section II for notes.
- String *authorizationID*
  User authorization information. The *authorizationID* may be NULL. See Section II for notes.
- String *requestToken* (required)
  A token associated with the previously submitted request to resume its activities. The requestToken was returned by the function initiating the request (e.g. *srmPrepareToGet*()).
- ReturnRequestStatus *returnStatus* (required)
  Output parameter reporting the success or failure of the request. In case of failure, the associated *errorCode* is returned.

7.1.4. RETURN CODE

On successful abort, the *statusCode* is set to SRM_SUCCESS.
On failure, the *statusCode* is set to SRM_FAILURE and the *errorCode* is set.

7.1.5. ERROR CODE

When status is failure, *errorCode* is set to one of the following:

SRM_AUTHENTICATION_FAILURE
- SRM server failed to authenticate the client

SRM_AUTHORIZATION_FAILURE
- client is not authorized to resume the request specified by the *requestToken*

SRM_INVALID_REQUEST_TOKEN
- *requestToken* does not refer to an existing request

7.1.6. NOTES on the Behavior

a) Resume the previously suspended files that belong to the request associated with the *requestToken*.

7.1.7. SEE ALSO

srmGetRequestTokens, srmSuspendRequest

7.2. srmSuspendRequest

7.2.1. NAME

srmSuspendedRequest - is to suspend a previously submitted active request.

7.2.2. SYNOPSIS

In	Out
String userID String authorizationID String requestToken	ReturnRequestStatus { EnumStatusCode statusCode, string explanation, EnumErrorCode errorCode } returnStatus

7.2.3. DESCRIPTION

srmSuspendedRequest - is to suspend a previously submitted active request.

- String *userID* (required)
  User authentication identifier. See Section II for notes.
- String *authorizationID*
  User authorization information. The *authorizationID* may be NULL. See Section II for notes.
- String *requestToken* (required)
  A token associated with the previously submitted request to suspend its activities. The requestToken was returned by the function initiating the request (e.g. *srmPrepareToGet*()).
- ReturnRequestStatus *returnStatus* (required)
  Output parameter reporting the success or failure of the request. In case of failure, the associated *errorCode* is returned.

7.2.4. RETURN CODE

On successful abort, the *statusCode* is set to SRM_SUCCESS.
On failure, the *statusCode* is set to SRM_FAILURE and the *errorCode* is set.

7.2.5. ERROR CODE

When status is failure, *errorCode* is set to one of the following:

SRM_AUTHENTICATION_FAILURE
- SRM server failed to authenticate the client

SRM_AUTHORIZATION_FAILURE
- client is not authorized to suspend the request specified by the *requestToken*

SRM_INVALID_REQUEST_TOKEN
- *requestToken* does not refer to an existing request

7.2.6. NOTES on the Behavior

a) Suspend all files in the request until *srmResumeRequest* is issued. Local policy may be enforced for the duration of suspended time period. If the suspended time period expires and there has been no action from the client on the request, the SRM may choose to abort the request.

b) In order to avoid space charges on pinned files, the client must release those pinned files before or after suspending the request.

c) Lifetime of files not released in will continue until its expiration.

d) If lifetime of files expires for the suspended request, then those files will be put back on the queue for the client.

e) File release requests are performed even if the request is suspended. Releasing a request can be done after suspending the request because some files may be brought into the cache between release and suspend.

f) Upon suspending the request, new files must not be brought into the SRM space for the request .

g) *srmAbortRequest* may be performed to end the suspended request.

7.2.7. SEE ALSO

srmGetRequestTokens, srmResumeRequest

8. Appendix

8.1. Appendix A : StatusCode Specification

Note:

- Status codes represent errors, warnings and status.

Status code Explanation

SRM_SUCCESS:

- SRM request was successful

Errors:

SRM_FAILURE:

- Requested operation failed for unspecified reason, and additional info is in the explanation string.

SRM_AUTHENTICATION_FAILURE:

- Requester has an invalid authentication information.

SRM_AUTHORIZATION_FAILURE:

- Requester has no permissions for the operation (although the user could have a valid authentication information).

SRM_INVALID_REQUEST:

- The request is invalid, and additional information may be provided in the explanation string. For example,
  - The request token is invalid
  - The requested life time of a file is longer than the lifetime of the space.

SRM_INVALID_PATH:

- The requested file/directory path or SURL is invalid.

SRM_INVALID_REQUEST_TOKEN:

- The request token is invalid.

SRM_INVALID_SPACE_TOKEN:

- The space token is invalid.

SRM_FILE_LIFETIME_EXPIRED:

- The life time on the pinned file has expired

SRM_SPACE_LIFETIME_EXPIRED:

- The life time on the reserved space has expired

SRM_EXCEED_ALLOCATION:

- Requester exceeded allocation (number of requests, files or spaces), and the request cannot be placed.

SRM_NO_USER_SPACE:

- The requester does not have enough space to put the file into that space.

SRM_NO_FREE_SPACE:

- SRM has not more space.

SRM_DUPLICATION_ERROR :

- Requester tried to create a new file or directory that already exists.

SRM_NON_EMPTY_DIRECTORY:

- Requester tried to remove a non-empty directory without the recursive option set.

SRM_TOO_MANY_RESULTS:

- The request produced too many results; for example, as a result of srmLs. The term "too many" is determined by each SRM , and the detailed information, such as the supported max number of results can be returned in the explanation string.

SRM_INTERNAL_ERROR:

- SRM has an internal error temporarily. Client may try again.

SRM_FATAL_INTERNAL_ERROR:

- SRM has a severe internal error that cannot be recovered for an extended period of time.

SRM_NOT_SUPPORTED:

- SRM implementation does not support this functionality that client requested.

Status:

SRM_REQUEST_QUEUED
SRM_REQUEST_INPROGRESS
SRM_REQUEST_FINISHED
SRM_REQUEST_SUSPENDEND
SRM _ABORTED
SRM _RELEASED

SRM_FILE_PINNED

- The requested file is pinned

SRM_FILE_IN_CACHE

- The file is in cache, but not pinned

SRM_FILE_IN_SPACE

- The file is in space. Will be used with srmPutFileDone() or srmPutRequestDone().

SRM_SPACE_AVAILABLE

- The requested space is reserved and ready to be used

SRM_LOWER_SPACE_GRANTED

- The requested space is not ready, but lower sized space is granted.

SRM_CUSTOM_STATUS:

- SRM has a site specific status information. The details are described in the explanation string.

References and Related Papers

[1] Storage Resource Management working group: http://sdm.lbl.gov/srm-wg

[2] GGF Grid Storage Management working group: http://sdm.lbl.gov/gsm

[3] StorageManager.org: http://www.storagemanager.org

[4] Storage Resource Managers: Middleware Components for Grid Storage, *Arie Shoshani, Alex Sim, Junmin Gu,* Nineteenth IEEE Symposium on Mass Storage Systems, 2002 (MSS '02)

[5] Disk Cache Replacement Algorithm for Storage Resource Managers in Data Grids, *Ekow J. Otoo, Frank Olken and Arie Shoshani*, Supercomputing Conference, 2002.

[6] The Anatomy of the Grid: Enabling Scalable Virtual Organizations. *I. Foster, C. Kesselman, S. Tuecke*. International J. Supercomputer Applications, 15(3), 2001.

[7] Best-effort versus reservations: A simple comparative analysis, *Lee Breslau and Scott Shenker*, ACM Computer Communication Review, 28(4):3--16, September 1998.

[8] Concurrency Control and Recovery In Database Systems, *Philip A. Bernstein, Vassos Hadzilacos, and Nathan Goodman*, Addison-Wesley Longman, 1987.

[9] Transaction Processing: Techniques and Concepts, *J. Gray and A. Reuter*, Morgan Kaufmann, San Mateo, CA, 1994.

[10] DataMover: Robust Terabyte-Scale Multi-file Replication over Wide-Area Networks, *Alex Sim, Junmin Gu, Arie Shoshani, Vijaya Natarajan,* Proceedings of the 16th International Conference on Scientific and Statistical Database Management (SSDBM 2004), Greece

[11] Storage Resource Managers: Essential Components for the Grid, *Arie Shoshani, Alexander Sim, and Junmin Gu*, in Grid Resource Management: State of the Art and Future Trends, Edited by Jarek Nabrzyski, Jennifer M. Schopf, Jan weglarz, Kluwer Academic Publishers, 2003