The Storage Resource Manager
Functional Design Specification

Version 3.0
Draft 2006.2

25 August 2006

Collaboration Web: http://sdm.lbl.gov/srm-wg
Document Location: http://sdm.lbl.gov/srm-wg/doc/SRM.v3.0.2006.2.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)
Olof Barring Jean-Philippe Baud James Casey	LHC Computing Project (LCG), CERN
Jens Jensen Shaun De Witt	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

Table of Contents

I. Introduction

This document contains the functional design specification of SRM 3.0. It is designed to support the functionality of previous SRM versions (specifically, v1.1, v2.1.2 and v2.2) 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 storage 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 the SRM v3.0 specification, as well as some basic concepts We start with a description of how to represent "core features" and "advanced features" in the specification. Core features are functions that all SRM implementations should support. Advanced features are functions that are optional, such as "space reservations". However, some core functions may be effected by advanced features. We first address the issue of having functions that may be involved in multiple features. This is followed with a section on storage properties ("retention policies", and "access latency") and file types (referred to as "volatile", durable" and "permanent"). 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.

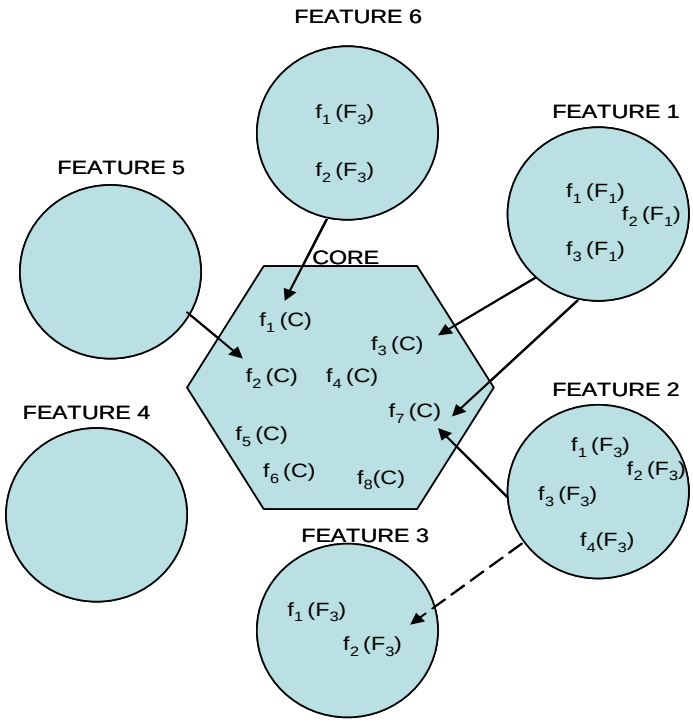
I.1. Extending parameters according to features

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 1 effects functions 3 and 7 of core, and Feature 6 effects function 1 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. Storage Properties and File Storage Types

I.2.a. File Storage 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 storage 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 (or

can be thought of as having unlimited lifetime), 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 an archival space, release the space that the *durable* file occupies, and notify the client. A *durable* file type is necessary in order to allocate temporarily on-line 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 will 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 another, usually archival 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 computational resources. *Volatile* files are useful in analysis use cases, where files can 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.

To summarize, there are three file storage types which represent the lifetime of a file managed by SRM, and the removal policy when the lifetime expires. They have the following semantics:

- ***Volatile***: This has an expiration time and the storage may delete it either when the file is "released" or when the lifetime expires.
- ***Durable****:* This has an expiration time, but the SRM may not delete the file; instead, the SRM should take some action when the lifetime expires, including raising an error condition and/or moving the file to an archive till a response form the owner can be obtained as to the viability of the file.
- ***Permanent****:* This has no expiration time.

I.2.b. Storage and Space Properties

SRMs manage shared storage spaces. If the "space reservation" feature is supported, then clients can negotiate and acquire storage space that the SRM assigns to them. Otherwise, the SRM assigns a default space size that depends on its policy. Similar to files the storage spaces assigned by the SRM have a lifetime associated with them. In addition, when making a space reservation, the request can specify the storage properties of that space. In version 3.0 two storage properties can be specified: *Retention Policy*, and *Access Latency*. These are described below.

Retention Policies

Quality of Retention is a kind of Quality of Service. It acts as a notification to the storage system of the intention of the user regarding the "preciousness" of the file, and what would/could be done if the storage system lost the data.
There are three retention policies with the following semantics:

- ***Replica**:* Replica quality is appropriate for data that can be replaced since other copies can be accessed in a timely fashion.
- ***Output**:* Output quality is an intermediate level and is used for data which can be replaced by length or effort-full processes.
- ***Custodial**:* Custodial quality indicates that the data is in some sense unique, and all attempts should be made to ensure it is kept.

Access Latency

This refers to how latency to access a file is improvable. Latency is changed by storage systems replicating a file internally on a storage area of different access latency.
Three access latency modes are defined, but in practice only the first two are used:

- ***Online**:* This is the mode with the lowest latency, implying that there exists a copy of the file in the cache and available for the client
- ***Nearline**:* This represents file stored on a high latency medium, such as tape, which can have their latency automatically improved to *ONLINE* by a staging operation
- ***Offline**:* A file in *OFFLINE* requires human intervention to achieve low latency. SRMs do not manage offline storage spaces.

If the "space reservation" feature is supported by an SRM, a client can acquire multiple storage spaces regardless of the storage properties. 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 a single default space.

The lifetime of a file cannot exceed the lifetime of the space it is assigned to.

I.2.c. Copying and Moving files between storage spaces

An SRM can support any combination of storage types based on their properties. For example, an SRM can support an two types of *online* storage, one with *replica* quality and one with *custodial* quality. The quality level implementation depends on the underlying storage system. For example, an *online custodial* quality may mean support of multiple disk copies or RAID storage to ensure that the probablility of their retention is very high.

The properties of a storage space can be found through a discovery function. Given a space-token, the SRM will return its retention policy, access latency, size (guranteed, and best-effort), and lifetime.

A very common use is getting copies of files that are stored in *nearline* storage space to an *online* storage space. For this reason, v3.0 has the function *BringOnline***.** In this case, the file brought online is still referred to with the same SURL, and a TURL is not returned. If the online space token is not provided, a default online space is assigned. Note that in this case, the original files is not removed from the nearline space.

For all other functions of moving files from one space to another the function *ChangeSpaceForFiles* is used. In this case, the target space token is mandatory, the file retains the same SURL, and the the original file is removed from it source space.

Other functions of copying and moving files are *Cp* and *Mv*. These are functions used to move files between directory structures, not change storage space. An important difference between these functions and *ChangeSpaceForFiles* is that when using these functions, a new target SURL is assigned. The following table summarizes the effects expected in the use of these various SRM functions.

properties → / Functions ↓	Source SURL provided?	Target SURL provided?	Retain same SURL?	Return TURL?	space-token Mandatory?	Retain file in Source space?
PrepareToGet	Yes	No	Yes	Yes	No	Yes
PrepareToPut	No	Optional	NA	Yes	No	NA
BringOnline	Yes	No	Yes	No	No	Yes
ChangeSpace	Yes	No	Yes	No	Yes	No
Cp	Yes	Yes	No	No	No	Yes
Mv	Yes	Yes	No	No	Yes	No

I.3. Releasing and removing files

PrepareToGet and *PrepareToPut* put or get local files only. In order to get files from remote sites or put files into remote sites, the advanced function *RemoteCopy* function has to be used. From the client's point of view files brought in by the SRM as a result of *PrepareToGet*, *PrepareToPut*, and *RemoteCopy* requests are always brought into a local space that is assigned to the client. The spaces can be assigned by default, as in the "core" feature, or acquired explicitly 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. The *Rm* function takes an SURL as a parameter, and removes all copies of the file with this SURL.

ReleaseFiles with Request Token can only be used for files previously pinned as a result of *PrepareToGet* and *BringOnline*. 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 *Rm* function has to be used.

CleanupFilesFromSpace with Space Token can be used to release files regardless of the request that brought them in. It has an input parameter *remove* 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 an input parameter *remove* that can be set if the client wishes the files to be removed, not only released.

Aborting an *PrepareToGet* and *PrepareToPut* has only a local effect. However, aborting an *RemoteCopy* has to be propagated to the remote site. In the case of aborting an *PrepareToGet* and *PrepareToPut* 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 *RemoteCopy* request has two cases to consider. When the *RemoteCopy* function is from the remote SRM to local SRM (also referred to as a "copy-pull"), and vice versa, if the *RemoteCopy* command is from local SRM to a remote SRM (also referred to as a "copy-push"). For copy-pull, the local SRM issues an *PrepareToGet* request to the remote SRM, and for copy-push an *PrepareToPut* request is issued. When the *RemoteCopy* is aborted, it is propagated to the remote site by aborting the *PrepareToGet* or the *PrepareToPut* request, correspondingly. 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 *PrepareToPut* 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 *PrepareToGet* 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 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 *UpdateSpace* function.

As mentioned above all files assigned to a space must have a lifetime that is 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 *GetSpaceMetadata* 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 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.

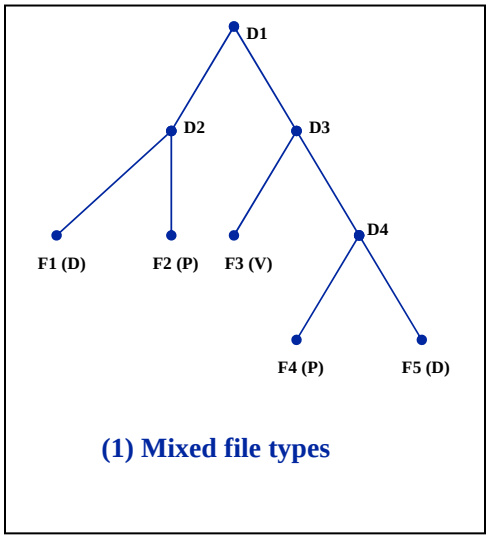
If the released space has files in it, the space that is not occupied by files is released immediately. However, the action regarding the files that are in the space depends on the type of files. We describe the action for each next.

- For *volatile* files, 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* files, 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.

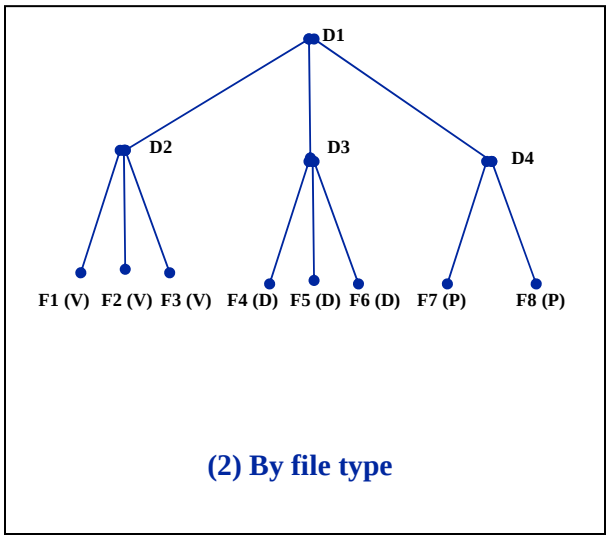
I.5. Directory Management

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: *Ls*, *Mkdir*, *Rmdir*, *Rm*, *Mv*, and *Cp*. *Ls* and *Rm* are considered core functions. No "link" functions are supported.

In the case that the SRM supports a single space for each client, the behavior of the above functions is straightforward. However, what is the behavior when there are multiple spaces? 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 storage 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 storage 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 storage 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 storage 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 storage type changes, is an implementation choice. The same behavior of files assigned to spaces 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 interface specification of SRM 3.0. It is designed to support the functionality of SRM 2.2, SRM 2.1.1 and SRM 1.1 (see http://sdm.lbl.gov/srm-wg/documents.html), but is organized to separate the functionality by "core features", and "advanced features" functions.

I.7.a. The relationship between functional design and operational interface specification

The purpose of the functional design document is to have an independent functional description from the operational interface specifications that are tailored to a particular framework, such as web-services. Figure 1 shows the relationship and history of the SRM specifications. Note that a single functional design can be used to generate multiple operational interface specifications for multiple frameworks. So far, we only anticipate such a situation for v3.0 where we first have an operational interface specification for WS, and later expect to have an interface specification for WSRF.

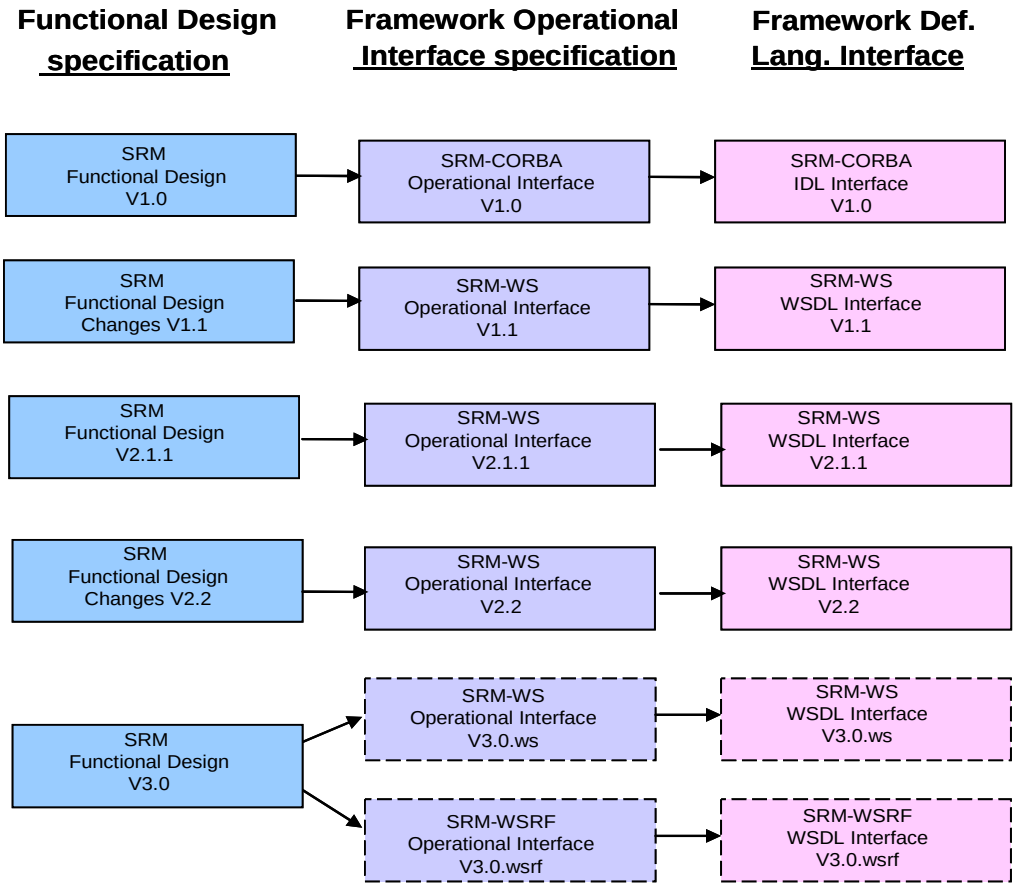


Figure 1. Relationship betweeen specifications

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, SURL Status { SURL, Status information, explanation } represents that SURL Status is not required, but when it is provided, SURL and Status information are required to be provided.

1. Description of Common 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.

1.1. Namespace SRM

1.2. Meaning of Local

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

1.3. Unsupported Features

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

1.4. User Authentication and Authorization

The Authentication and Authorization for all the SRM functions are implemented in accordance with capabilities of the authentication mechanism and messaging technology or protocol selected and specified by the particular incarnation of the functionality depending on the operational framework. It is recommended that a strong authentication mechanism like x509 based TSL, GSI or Kerberos is used. Authorization is based on site-specific authorization policy but VO based authorization is recommended for implementations suitable for GRID deployment. For example, in SASL RFC 2222, the authorization information is described as follows: "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."

1.4.a. Additional Storage System Authentication and Authorization

The storage system may require additional information of user authentication/authorization for the underlying storage system which cannot be satisfied with the user authentication/authorization information submitted with the request. We refer to this as "Storage System Info", which has the following characteristics:

- Storage System Info may contain but is not limited to the following information: storage device, storage login ID, storage login authorization, storage project id, storage file 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.

1.5. Transfer Protocol Negotiation

When making a request to an SRM, the client needs to end up with a protocol for the transfer of the file that the storage system can support. In general, systems may be able to support multiple protocols and clients may be able to use multiple protocols depending on the system they are running on, such as GridFTP, HTTP, BBFTP, etc. The SRM design addresses this issue by allowing the client to provide an ordered protocol list, and the SRM selects the highest possible matching protocol. The transfer protocol negotiation can be uses in the case of srmPrepareToGet and srmPrepareToPut. During the execution of the SRM Server constructs a transfer URL for each File Description using the highest possible protocol on the ordered protocol list.

1.6. Lifetime

1.6.a. Request Lifetime

Once created, each file request must be assigned a user specified or policy based lifetime. There must be a way to discover the remaining lifetime of the file request, with a well-specified protocol. The operational spec may select methods to modify the lifetime of the request, depending on a specific messaging protocol. If such functionality is specified, it must be optional for the implementers. There must be a way to terminate the request and free up the resources occupied by the request either upon successful transfer of the data or prematurely before the request was properly executed. For example in WSRF the lifetime may be managed by the functions specified in WS-Lifetime specification. In case of pure SOAP based implementation this could be a set of functions called SRM_Release, SRM_Modify_Lifetime, etc, taking the request token and lifetime as arguments.

1.6.b. File Lifetime

All files in the storage have a property of lifetime. Specially VOLATILE and DURABLE files have a limited lifetime that SRM may react on the expired files. The PERMANENT files are considered to have an infinite lifetime.

1.6.c. Expiration Time

Once a file is brought online as a result of srmPrepareToGet or srmBringOnline, the resulting Transfer URL (TURL) has a property of expiration time. Also, when a space is allocated for a file to be transferred into the online cache as a result of srmPrepareToPut, the resulting TURL has a property of expiration time. At the end of these expiration time, the TURL may be no longer available for the client.

1.7. Site URLs and Transfer URLs

The terminology for file names used in this functional specification is as follows:

LFN- is a logical file name that is globally unique for a given dataset. Thus, the dataset name is usually the first part of the LFN. It is the choice of the dataset designer how to assign these names. If the files are organized in directories then the directory names are part of the LFN. The dataset name and directory names are separated by "/". An example of an LFN is: "CERN-dataset-7/run17/part1/file-123".

SFN- is a file name assigned by a site to a file. Normally, the site file name will consist of a "machine:port/directory/LFN", but the site can choose to use another string instead of the LFN. An example of an SFN is: "sleepy.lbl.gov:4000/tmp/foo-3000". In this example we used the simple file name "foo-3000" to simplify the example.

SURL – is a "site URL" which consists of "protocol://SFN". The protocol for communicating with an SRM is simply "srm". An example of an SURL for a file managed by SRM is: "srm://sleepy.lbl.gov:4000/tmp/foo-3000".

TFN – is the "transfer" file name of the actual physical location of a file that needs to be transferred. It has a format similar to an SFN.

TURL – is the "transfer URL" that an SRM returns to a client for the client to "get" or "put" a file in that location. It consists of "protocol://TFN", where the protocol must be a specific transfer protocol selected by SRM from the list of protocols provided by the client (see section 2.2). If the physical storage location matches the one provided by the SURL, then only the protocol is replaced in the TURL. For the above SURL example, and assuming the protocol is "gridftp", the TURL will be: "gridftp://sleepy.lbl.gov:4000/tmp/foo-3000". However, the physical file location can be anywhere at that site, giving the freedom for the site manager to change the physical locations of files without having to change the SURL or update the replica catalog. If for the above example the physical location of the file is on another machine (e.g. "dm.lbl.gov", another path (e.g. "/home /level1"), and even another file-name (e.g. "abc-3000") then the TURL will be: "gridftp://dm.lbl.gov:4000/home /level1/abc-3000".

In the web services operational specification, these URLs may be *anyURI*. 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]".

1.8. Request Handle and User Request Description

Request handles or Tokens are needed for follow up on the status of a request or for terminating a request. All subsequent calls about a request (such as "status" or "abort") have to use the request handle. When a new call is made, the SRM assigns a request handle. However, since the request handle can be lost, the client can provide the SRM with an optional User Request Description parameter, containing textual description of the request.
SRM_Get_Request_[Tokens/Handles/EPR etc] can be later used to discover the handle on basis the description. Note that a client is allowed to assign the same request description to multiple

requests. In this case, the SRM will return back all the request handles associated with the request description along with the time the requests were submitted.

For example, in case of WSRF based operational specification, the handle to the request could be WS-Resource compliant End Point Reference (EPR); in case of the messaging protocol that does not support state, like SOAP, the request could be identifiable by some type of a request token. Each file request can be a property of the request or can be an independent entity having its own identification (ERP).

1.9. File Type

- FILE
- DIRECTORY
- LINK

1.10. Overwrite Mode

- NEVER
- ALWAYS
- WHENFILESAREDIFFERENT

1.11. Permission Mode

- NONE | X | W | WX | R | RX | RW | RWX

1.12. Permission Type

- ADD | REMOVE | CHANGE

1.13. Request Type

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

- PREPARE_TO_GET
- PREPARE_TO_PUT
- REMOTE_COPY
- BRING_ONLINE
- RESERVE_SPACE
- UPDATE_SPACE
- CHANGE_SPACE_FOR_FILES
- LS
- LS_DETAILS
- RM
- CP
- MV

1.14. File Locality

Files may be located online, nearline or both. This indicates if the file is online or not, or if the file reached to nearline or not. It also indicates if there are online and nearline copies of the file.

- ONLINE: The ONLINE indicates that there is a file on online cache of a storage system which is the part of the storage system, and the file may be accessed with online latencies.
- NEARLINE: The NEARLINE indicates that the file is located on nearline storage system, and the file may be accessed with nearline latencies.
- ONLINE_AND_NEARLINE: The ONLINE_AND_NEARLINE indicates that the file is located on online cache of a storage system as well as on nearline storage system.
- LOST: The LOST indicates when the file is lost because of the permanent hardware failure.
- NONE: The NONE value shall be used if the file is empty (zero size).
- UNAVAILABLE: The UNAVAILABLE indicates that the file is unavailable due to the temporary hardware failure.

The type will be used to describe a file property that indicates the current location or status in the storage system.

1.15. Access Pattern

Access Pattern will be passed as an input parameter to the srmPrepareToGet and srmBringOnline functions. It will make a hint from the client to SRM how the Transfer URL (TURL) produced by SRM is going to be used. If the parameter value is "ProcessingMode", the system may expect that client application will perform some processing of the partially read data, followed by more partial reads and a frequent use of the protocol specific "seek" operation. This will allow optimizations by allocating files on disks with small buffer sizes. If the value is "TransferMode" the file will be read at the highest speed allowed by the connection between the server and a client.

- TRANSFER_MODE
- PROCESSING_MODE

1.16. Connection Type

Connection Type indicates if the client is connected though a local or wide area network. SRM may optimize the access parameters to achieve maximum throughput for the connection type. This will be passed as an input to the srmPrepareToGet, srmPrepareToPut and srmBringOnline functions

- WAN
- LAN

1.17. UTC time

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

1.18. Status information and access to the status of a request

SRM functions lead to the creation of the requests on the server. There must be a way for the discovery of the state of each request and each file request within request with various level of granularity, with a well-specified messaging mechanism. For example in case of a WSRF Based implementation, this may be a standard *getResouceProperty*() for the resource identified by EPR; in SOAP this may be separately specified RPC calls.

Status of the file requests must contain original, unmodified SURL and *Status Information* which may include *Status Codes*, its *explanation* and *Error Codes* (Status codes are described in implementation neutral form in later section). Status of the file request may also contain the following information: *Transfer URL* in one of protocols specified by the client, Size of the file, Remaining lifetime of the File Request, Expiration Time by which the Transfer URL becomes invalid, Checksum type and Checksum value, etc.

1.18.a. Status codes

It represents the status of the requests or files. The SRM status codes are explained in Appendix A.

1.18.b. Error codes

It represents the errors on the requests or files. The SRM status codes are explained in Appendix B.

1.19. File Sharing

File sharing by the SRM is an implementation choice. An SRM can choose to share files by users by providing multiple clients access to the same physical file, or by copying a file into separate storage spaces. Either way, if an SRM chooses to share a file (in order 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.

1.20. Transfer Information

Transfer Information is used where Transfer Protocols was used previously in SRM v2.1. It may include information about Access Pattern, Connection Type, Client Networks, and Transfer Protocols. In srmPrepareToGet, AccessPattern may conflict with the characteristics of the online disk of the target space associated with target space token if provided. In this case, Access Pattern must be ignored. File transfer protocols are specified in a preferred order on all SRM transfer functions. Client Networks is a hint of the client IPs that some SRM can use for optimization of its internal storage systems based on the client's accessible IP addresses.

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. AbortRequest

2.1.1. NAME

AbortRequest - terminates the previously submitted request

2.1.2. Functional Description

AbortRequest() allows clients to prematurely terminate asynchronous requests of any types. It may involve data transfer requests initiated by a call to *PrepareToGet*(), *BringOnline*(), *PrepareToPut*() or *RemoteCopy*() as well as calls such as *Ls*(). The effect of *AbortRequest*() 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.

2.1.3. Parameter Description

In	Out

User ID
Authorization ID
Request Token
	Request Status {
	Status information
	}

Input parameters

- *User ID* (required)
  User authentication identifier. See Section 1.4 for notes.
- *Authorization ID*
  User authorization information. The *Authorization ID* may be NULL.
- *Request Token* (required)
  A handle associated with the request to be aborted. The *Request Token* was returned by the function initiating the request (e.g. *PrepareToGet*()).

Output parameters

- *Return Status* (required)
  Output reporting the success or failure of the request. Textual description of explanation for what happened to the request, specially in case of any failures, would help client diagnose the case. *Status information* is required and it is recommended to return as an enumerated codes separated as status code and error code, so that when client is a non-human program, response to the status information can be pre-programmed.

2.1.4. 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) Abort must be allowed to all requests with *Request Token*, including requests such as *srmLs* or *srmRemoteCopy*.

c) Those files that are brought online with unlimited lifetime will remain in the space where they are brought in, and are not removed. Clients need to remove explicitly through *Rm* or *PurgeFromSpace*.

2.1.5. Status information returned upon request execution

On successful request, the *status information* returns SUCCESS.
On failure, the *status information* returns FAILURE and the detailed failure information along with the textual explanation in a human readable form would help the client diagnose the failed request.
When detailed failure can be set to one of the followings:

SRM_SUCCESS
- successful request completion. Request is aborted successfully.

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 *Request Token*

SRM_INVALID_REQUEST
- *Request Token* does not refer to an existing known request in the SRM server.

SRM_INTERNAL_ERROR
- SRM has an internal transient error, and client may try again.

SRM_FAILURE
- any other request failure. *Explanation* needs to be filled for details.

SRM_NOT_SUPPORTED
- *function* is not supported in the SRM

2.1.6. SEE ALSO
AbortRequestedFiles, PrepareToGet, PrepareToPut, RemoteCopy, Ls, LsDetails

2.2. AbortRequestedFiles

2.2.1. NAME
AbortRequestedFiles - aborts selected files from the request.

2.2.2. Functional Description
AbortRequestedFiles() 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 PrepareToGet(), BringOnline(), PrepareToPut(), or RemoteCopy(). The effect of a AbortRequestedFiles() depends on the type of the request.

2.2.3. Parameter Description

In	Out
User ID	Request Status {
Authorization ID	Status information
Request Token	}
SURLs	SURL Statuses {
	SURL,
	Status information
	}

Input parameters

- *User ID* (required)
  User authentication identifier.
- *Authorization ID*
  User authorization information. The *Authorization ID* may be NULL.
- *Request Token* (required)
  A handle associated with the request to be aborted. The *Request Token* was returned by the function initiating the request (e.g. *PrepareToGet*()).
- *SURL*s (required)
  SURLs associated with the request token need to be provided.

Output parameters

- *Return Status* (required)
  Output reporting the success or failure of the request. Textual description of explanation for what happened to the request, specially in case of any failures, would help client diagnose the case. *Status information* is required and it is recommended to return as an enumerated codes separated as status code and error code, so that when client is a non-human program, response to the status information can be pre-programmed
- SURL Statuses
  Output reporting the success or failure of the each SURL requests. *SURL Statuses* may be empty and NULL. If returned to the client, *SURL* and its *status information* are required to return.
  - *SURL* (required)
    SURL that client has requested to abort.

2.2.4. 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.

2.2.5. Status information returned upon request execution

On successful abort, the *status information* returns SRM_SUCCESS.
On failure, the *status information* returns SRM_FAILURE and the detailed failure information along with the textual explanation in a human readable form would help the client diagnose the failed request. If only partial files were successful, the *status information* returns SRM_PARTIAL_SUCCESS, and the *SURL Statuses* should explain on those failed files.
When detailed failure can be set to one of the followings:

For request level Status,

SRM_SUCCESS
- successful request completion. All *SURLs* are aborted successfully.

SRM_PARTIAL_SUCCESS
- All requests are completed. Some *SURLs* ares successfully aborted, and some *SURLs* are failed. Details are on the files status.

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 *Request Token*

SRM_INVALID_REQUEST
- *SURLs* is empty.
- *Request Token* is empty.
- *Request Token* does not refer to an existing known request in the SRM server.

SRM_INTERNAL_ERROR
- SRM has an internal transient error, and client may try again.

SRM_FAILURE

- All files requests are failed.
- any other request failure. *Explanation* needs to be filled for details.

SRM_NOT_SUPPORTED
- *function* is not supported in the SRM

For file level Status,

SRM_SUCCESS
- successful request completion for the *SURL*. *SURL* is aborted successfully.

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

SRM_FAILURE
- any other request failure. *Explanation* needs to be filled for details.

2.2.6. SEE ALSO
AbortRequest, PrepareToGet, PrepareToPut, RemoteCopy, Ls, LsDetails

2.3. BringOnline

2.3.1. NAME
This function is used to bring files online upon the client's request so that client can make certain data readily available for future access. In hierarchical storage systems, it is expected to "stage" files to the top hierarchy and make sure that the files stay online for a certain period of time. When client specifies target space token which must be referred to an online space, the files will be brought online using the space associated with the space token. It is an asynchronous operation, and request token must be returned if asynchronous operation is necessary in SRM. The status may be checked through *BringOnlineRequestStatus* with the returned request token. This function is similar to *PrepareToGet*, but it does not return Transfer URL (TURL).

2.3.2. Functional Description
This function is used to bring files online upon the client's request so that client can make certain data readily available for future access. In hierarchical storage systems, it is expected to "stage" files to the top hierarchy and make sure that the files stay online for a certain period of time. When client specifies target space token which must be referred to an online space, the files will be brought online using the space associated with the space token. It is an asynchronous operation, and request token must be returned if asynchronous operation is necessary in SRM. The status may be checked through *BringOnlineRequestStatus* with the returned request token.

Files in the request
BringOnline function receives an unordered set of File Descriptions describing the files that client is interested in accessing from the Storage System managed by SRM. Each of the File Description data structure must contain SURL – Site URL of the file being accessed. Each File Description consists of parameters described in the "parameters description" section.

BringOnline execution

BringOnline function execution leads to the creation of the request on the server. The request consists of the collection of the file requests, one for each File Description in the request. The state of each file requests persists on the server at least for the duration of the lifetime of the request. The client should be given the handle (or collection of handles), specific to a messaging mechanism, allowing the client to refer to the request as a whole or file requests individually. The *BringOnline* function must be an asynchronous non-blocking call returning the handle of the request back to the client as soon as possible.

Once the Server receives the request to prepare certain files to bring online, it can make files available for the access in one of the specified protocols. Once the file is made available online and the file is made available to the client. When the client needs the file to transfer out from the online storage, *PrepareToGet* needs to be requested.

2.3.3. Parameter Description

In	Out
User ID	Request Token
Authorization ID	Request Status {
User Request Description	Status information
Total Request Time	}
Overwrite Mode,	Remaining Total Request Time
Get File Requests {	Remaining Deferred Start Time
Source SURL,	SURL statuses {
Directory Option	Source SURL,
}	Status information
Desired Pin Lifetime,	File Size,
Target Space Token,	Estimated Wait Time,
Target File Retention Policy Info,	Remaining Pin Time
Transfer Information,	}
Target File Storage Type,	
Target Storage System Info,	
Flag For Checksum,	
Deferred Start Time	

Input parameters

- *User ID* (required)
  User authentication identifier.
- *Authorization ID*
  User authorization information. The *Authorization ID* may be NULL.
- *Transfer Information*
  Information about the transfer protocols that client can support. It may include Access Pattern, Connection Types, the list of Client Networks, and the Transfer Protocol Info.
- *User Request Description*
  Textual description of the request. It may be used for later retrieval of the rquest token.
- *Total Request Time*

Total Request Time means that all the file transfer for this request must be complete within this *Total Request Time*. Otherwise, an error, SRM_REQUEST_TIMED_OUT must be returned as the request status information with individual file status of SRM_FAILURE with an appropriate explanation.

- *Overwrite Mode*
  In case SRM stages requested files for the user, SRM needs to stage the file according to the *Overwrite Mode* on the target that SRM brings the files into.
- Get File Requests (required)
  It consists of SURL information that client wants to retrieve, in an unordered set.
  - *Source SURL* (required)
    SURL that client desires to retrieve
  - *Directory Option*
    An advanced option when Directory Management feature is supported. It includes *Flag for Source Directory, Flag for All Level Recursive* and *Number Of Recursive Levels*: *Flag for Source Directory* is a boolean indicator if *Source SURL* is a directory. *Flag for All Level Recursive* is a boolean indicator if *Source SURL* is a directory and all files under the SURL must be retrieved. *Number Of Recursive Levels* is a positive integer indicator for how many levels of the recursive directory must be browsed and all files in those directories to be retrieved.
- *Desired Pin Lifetime*
  Desired Pin Lifetime is for a client preferred lifetime (expiration time) on the file in the online storage. Desired pin lifetime of the file is in UTC time once when the file is ready. SRM may assign a default lifetime based on the policy.
- *Target Space Token*
  Target Space Token indicates which spaces would be used when *Source SURL*s are brought in online. *Target Space Token* must refer to online space. All requested files will be prepared into the target space.
- *Target File Retention Policy Info*
  Target File Retention Policy Info indicates which space information would be used when *Source SURL*s are brought in online. It is to specify the desired retention policy information on the file when the file is prepared online.
- *Target File Storage Type*
  Target File Storage Type indicates which type of file storage is when it is brought online.
- *Target Storage System Info*
  Information specific to the underlying storage system that is associated with the *Source SURLs*. The *Target Storage System Info* may be NULL.
- *Flag for Checksum*
  Flag for Checksum indicates that checksum calculation for all files in the request needs to be performed when SRM server brings files into the space.
- *Deferred Start Time*
  Deferred Start Time is in seconds to support CE-SE resource co-allocation and tape mounting efficiency. It means that client does not intent to use the files before that time. If SRM decides not to bring any files until *Deferred Start Time* is reached, SRM_REQUEST_QUEUED must be returned.

Output parameters

- *Request Token* (required)
  Output parameter request handle is associated with the request for the later asynchronous status request.
- *Return Status* (required)
  Output reporting the success or failure of the request. Textual description of explanation for what happened to the request, specially in case of any failures, would help client diagnose the case. *Status information* is required and it is recommended to return as an enumerated codes separated as status code and error code, so that when client is a non-human program, response to the status information can be pre-programmed.
- *Remaining Total Request Time*
  Output parameter indicates the remaining total request time.
- *RemainingDeferred Start Time*
  Output parameter indicates how long the *Deferred Start Time* is left, if supported.
- SURL Statuses
  Output parameter reporting the status of the file in the request. *SURL Statuses* may be empty and NULL. If returned to the client, *SURL* and its *status information* are required to return.
  - *Source SURL* (required)
    Source SURL that client has requested to bring online.
  - *Estimated Wait Time*
    Time estimation on the file to prepare online
  - *Remaining Pin Time*
    When the *Source SURL* is brought online, the file has a property of Pin Lifetime. It indicates the remaining pin lifetime on the online copy of the *Source SURL*.

2.3.4. NOTES on the Core Behavior

a) This call may be an asynchronous (non-blocking) call, and SRM assigns the *Request Token* when the request is valid and accepted. The returned status code should be SRM_REQUEST_QUEUED. To get subsequent status and results, separate calls should be made through srmStatusOfBringOnline.

b) Input parameter *Target File Retention Policy Info* is to specify the desired retention policy information on the file when the file is brought online.

c) If both input parameters *Target Space Token* and *Target File Retention Policy Info* are provided, then their types must match exactly. Otherwise, the request may be rejected, and SRM_INVALID_REQUEST must be returned.

d) *Access Pattern* may conflict with the type of the target space associated with target space token, when both provided. In this case, *Access Pattern* in the input parameter *Transfer Information* must be ignored.

e) If the transfer protocol hints are not specified, default is assumed to be processing mode and LAN access for the site.

f) Access latency must be ONLINE always, when provided.

g) It is up to the SRM implementation to decide Connection Type if not provided.

h) The *User Request Description* is a client designated name for the request. It can be used in the *GetRequestToken*() function to get back the request tokens for requests made by the client.

i) Input parameter *Desired Pin Lifetime* is for a client preferred lifetime (expiration time) on the file "copies (or "states") of the SURLs that will be "brought online" into the target space that is associated with the *Target Space Token*.

j) The returned request token should be valid until all files in the request are released, removed or aborted.

k) When *AbortRequest* is requested for *BringOnline* request, the request gets aborted, but those files that are brought online will remain in the space where they are brought in, and are not removed. Clients need to remove those files through *PurgeFromSpace* or *Rm.*

l) If *Total Request Time* is unspecified as NULL, the request will be retried for a duration which is dependent on the SRM implementation. If *Total Request Time* is 0 (zero), each file request will be tried at least once.

2.3.5. NOTES on the Advanced Behavior with Directory Management Feature

a) *Number Of Levels* in *Directory Option* must be a valid input parameter, and must be a positive integer.

2.3.6. Status information returned upon request execution

On successful request, the *status information returns* SUCCESS.

On failure, the *status information returns* FAILURE and the detailed failure information along with the textual explanation in a human readable form would help the client diagnose the failed request.

When detailed failure can be set to one of the followings:

For request level status,

SRM_REQUEST_QUEUED
- successful request submission and acceptance. Request token must be returned.

SRM_REQUEST_INPROGRESS
- some files are completed, and some files are not completed yet. Request token must be returned.

SRM_SUCCESS
- All requests are successfully completed. All *Source SURLs* are successfully brought online.

SRM_PARTIAL_SUCCESS
- All requests are completed. Some files are successfully brought online, and some files are failed. Details are on the files status.

SRM_AUTHENTICATION_FAILURE
- SRM server failed to authenticate the client

SRM_AUTHORIZATION_FAILURE
- client is not authorized to submit the request

SRM_INVALID_REQUEST
- *Get File Request* is empty
- Access latency refers to something other than ONLINE.

- If both input parameters *Target Space Token* and *Target File Retention Policy Info* are provided, then their types must match exactly.
- *Target Space Token* does not refer to an existing known space in the SRM server.

SRM_SPACE_LIFETIME_EXPIRED
- space associated with the *Target Space Token* is expired.

SRM_EXCEED_ALLOCATION
- space associated with the *Target Space Token* is not enough to hold all requested SURLs.

SRM_NO_USER_SPACE
- user space is not enough to hold all requested *Source SURLs*.

SRM_NO_FREE_SPACE
- SRM space is not enough to hold all requested *Source SURLs* for free.

SRM_INTERNAL_ERROR
- SRM has an internal transient error, and client may try again.

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. Client requested *Desired File Storage Type* that is not supported by the SRM server.
- *Target File Retention Policy Info* does not refer to a supported retention policy in the SRM server.
- *Deferred Start Time* is not supported in the SRM server.
- Directory operation is not supported in the SRM server.
- Recursive directory operation is not supported in the SRM server.
- None of the file transfer protocols are supported in the SRM server.

SRM_FAILURE
- All files requests are failed.
- any other request failure. *Explanation* needs to be filled for details.

For file level return status,

SRM_SUCCESS
- successful request completion for the *SURL*. *SURL* is successfully brought online.

SRM_REQUEST_QUEUED
- file request is still on the queue.

SRM_REQUEST_INPROGRESS
- file request is being served.

SRM_AUTHORIZATION_FAILURE
- client is not authorized to retrieve the file that is associated with the *Source SURL*

SRM_ABORTED
- The requested file has been aborted.

SRM_RELEASED
- The requested file has been released.

SRM_FILE_LOST

- the requested file is permanently lost.

SRM_FILE_BUSY
- client requests for files which there is an active *srmPrepareToPut* (no *srmPutDone* is not yet called) for.

SRM_FILE_UNAVAILABLE
- the requested file is temporarily unavailable.

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

SRM_FILE_LIFETIME_EXPIRED
- *Source SURL* is expired
- pin lifetime has expired, but the file is still in the cache

SRM_FAILURE
- any other request failure. *Explanation* needs to be filled for details.
- The file request would not be able to be completed within the *Total Request Time*.
- The requested file has been suspended because the request has timed out.

2.3.7. SEE ALSO
BringOnlineRequestStatus, PrepareToGet, PrepareToPut, RemoteCopy

2.4. BringOnlineRequestStatus

2.4.1. NAME
BringOnlineRequestStatus - is the status call for BringOnline.

2.4.2. Functional Description
This function is used to check the status of the previous request to *BringOnline*, when asynchronous operation is necessary in the SRM. Request token must have been provided in response to the *BringOnline*.

2.4.3. Parameter Description

In	Out
User ID	Request Status {
Authorization ID	Status information
Request Token	}
Source SURLs	SURL Statuses {
offset	Source SURL,
count	Status information
	File Size,
	Estimated Wait Time,
	Remaining Pin Lifetime,
	}
	Remaining Total Request Time
	Remaining Deferred Start Time

Input parameters

- *User ID* (required)
  User authentication identifier.
- *Authorization ID*
  User authorization information. The *Authorization ID* may be NULL.
- *Request Token* (required)
  A request handle associated with the previously submitted *srmBringOnline* request. The *Request Token* was returned by the function initiating the request.
- *Source SURLs*
  SURLs to request their status.
- *offset*
  Integer indicator for long listed responses that the listing starts from the *offset*.
- *count*
  Integer indicator for long listed responses that the listing contains the number of *count.*

Output parameters

- *Request Status* (required)
  Output reporting the success or failure of the request. Textual description of explanation for what happened to the request, specially in case of any failures, would help client diagnose the case. *Status information* is required and it is recommended to return as an enumerated codes separated as status code and error code, so that when client is a non-human program, response to the status information can be pre-programmed.
- *SURL Statuses*
  Output parameter reporting the status of each file in the request. *SURL Statuses* may be empty and NULL. If returned to the client, *Source SURL* and its *status information* are required to return.
  - *Source SURL* (required)
    SURL that client has requested.
  - *Transfer URL*
    Transfer URL that SRM server prepared for client to transfer for the corresponding *Source SURL*.
  - *File Size*
    File size in bytes for the *Source SURL*.
  - *Estimated Wait Time*
    Time estimation on the file to bring online.
  - *Remaining Pin Lifetime*
    Life time in seconds that is remained on the online file.
- *Remaing Total Request Time*
  Output parameter indicating how much time left from the *Total Request Time*.
- *Remaing Deferred Start Time*
  Output parameter indicates how long the *Deferred Start Time* is left, if supported.

2.4.4. NOTES on the Core Behavior

a) If *Source SURLs* are not provided, the status for all the files associated with the *Request Token* is returned.
b) Output parameter *Source SURLs* are the same as the *Source SURL* that client provided with *BringOnline*().
c) When the file is ready online, the return status code should be SRM_FILE_IN_CACHE.
d) When the file is ready for the client, the file is implicitly pinned in the cache and pin lifetime will be enforced, subject to the policies associated with the underlying storage.
e) If any of the request files is temporarily unavailable, SRM_FILE_UNAVAILABLE must be returned for the file.
f) If any of the request files is permanently lost, SRM_FILE_LOST must be returned for the file.
g) The file request must fail with an error SRM_FILE_BUSY if *BringOnline* requests for files which there is an active *PrepareToPut* (no *PutDone* is not yet called) for.
h) SRM must fail (SRM_FAILURE) only if all files in the request failed.
i) If SRM decides not to bring any files until input parameter *Deferred Start Time* is reached, SRM_REQUEST_QUEUED must be returned.

2.4.5. NOTES on the Advanced Behavior with Space Management Feature

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

2.4.6. Status information returned upon request execution

On successful request, the *status information* returns SRM_SUCCESS.
On failure, the *status information* returns SRM_FAILURE and the detailed failure information along with the textual explanation in a human readable form would help the client diagnose the failed request. If only partial files were successful, the *status information* returns SRM_PARTIAL_SUCCESS, and the *SURL Statuses* should explain on those failed files.
When detailed failure can be set to one of the followings:

For request level status,

SRM_SUCCESS
- All requests are successfully completed. All *Source SURLs* are successfully brought online.

SRM_REQUEST_QUEUED
- successful request submission and all files request is still on the queue

SRM_REQUEST_INPROGRESS
- some files are completed, and some files are not completed yet.

SRM_PARTIAL_SUCCESS
- All requests are completed. Some files are successfully brought online, and some files are failed. Details are on the files status.

SRM_AUTHENTICATION_FAILURE
- SRM server failed to authenticate the client

SRM_AUTHORIZATION_FAILURE
- client is not authorized to submit the request

SRM_INVALID_REQUEST
- *Request Token* does not refer to an existing known request in the SRM server.

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. Client requested *Desired File Storage Type* that is not supported by the SRM server.
- *Target File Retention Policy Info* does not refer to a supported retention policy in the SRM server.
- *Deferred Start Time* is not supported in the SRM server.
- Directory operation is not supported in the SRM server.
- Recursive directory operation is not supported in the SRM server.
- None of the file transfer protocols are supported in the SRM server.

SRM_SPACE_LIFETIME_EXPIRED
- space associated with the *Target Space Token* is expired.

SRM_EXCEED_ALLOCATION
- space associated with the *Target Space Token* is not enough to hold all requested SURLs.

SRM_NO_USER_SPACE
- user space is not enough to hold all requested SURLs.

SRM_NO_FREE_SPACE
- SRM space is not enough to hold all requested SURLs for free.

SRM_ABORTED
- The request has been aborted.

SRM_REQUEST_TIMED_OUT
- Total request time is over and the rest of the request is failed.

SRM_INTERNAL_ERROR
- SRM has an internal transient error, and client may try again.

SRM_FAILURE
- All files requests are failed.
- any other request failure. *Explanation* needs to be filled for details.

For file level status,

SRM_SUCCESS
- successful request completion for the *SURL*. *Source SURL* is successfully brought online.

SRM_REQUEST_QUEUED
- file request is still on the queue.

SRM_REQUEST_INPROGRESS
- file request is being served.

SRM_AUTHORIZATION_FAILURE
- client is not authorized to retrieve the file that is associated with the *Source SURL*

SRM_ABORTED
- The requested file has been aborted.

SRM_RELEASED
- The requested file has been released.

SRM_FILE_LOST

- the requested file is permanently lost.

SRM_FILE_BUSY
- client requests for files which there is an active *srmPrepareToPut* (no *srmPutDone* is not yet called) for.

SRM_FILE_UNAVAILABLE
- the requested file is temporarily unavailable.

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

SRM_FILE_LIFETIME_EXPIRED
- *SURL* is expired
- pin lifetime has expired, but the file is still in the cache

SRM_FAILURE
- any other request failure. *Explanation* needs to be filled for details.
- The file request would not be able to be completed within the *Total Request Time*.
- The requested file has been suspended because the request has timed out.

2.4.7. SEE ALSO
PrepareToGet, BringOnline, PrepareToPut, RemoteCopy, Ls, LsDetails, ReleaseFiles, Rm, GetRequestTokens

2.5. ChangeFileStorageType

2.5.1. NAME
ChangeFileStorageType - changes file storage types.

2.5.2. Functional Description
ChangeFileStorageType allows clients to switch file storage type from one to another.

2.5.3. Parameter Description

In	Out
User ID	Request Token
Authorization ID	Request Status {
Storage System Info	Status information
SURLs	}
Desired File Storage Type	SURL Statuses{
Offset	SURL,
Count	New Lifetime,
Space Token	status information
	}
	Flag for partial list
	Total number of Files In The Request

Input parameters

- *User ID* (required)
  User authentication identifier. See Section II for notes.
- *Authorization ID*
  User authorization information. The *Authorization ID* may be NULL.
- *Storage System Info*
  Information specific to the underlying storage system. The *Storage System Info* may be NULL.
- *SURLs* (required)
  *SURL*s to change the file storage type.
- *Desired File Storage Type* (required)
  Desired file storage type that the file storage type of *SURL* is changed into.
- *offset*
  Integer indicator for long listed responses that the listing starts from the *offset*. Default is 0 for the first entry.
- *count*
  Integer indicator for long listed responses that the listing contains the number of *count.* Default is to return all entries of the list.
- *Space Token* (advanced option for space management feature)
  A handle associated with the space if a particular space in file storage type is to be used. The *Space Token* is acquired separately (e.g. *srmReserveSpace*).

Output parameters

- *Request Token*
  Output request handle that is associated with the request for the later asynchronous status request. *Request Token* may be NULL, in case *srmChangeFileStorageType* is processed without delay.
- *Request Status* (required)
  Output reporting the success or failure of the request. Textual description of explanation for what happened to the request, specially in case of any failures, would help client diagnose the case. *Status information* is required and it is recommended to return as an enumerated codes separated as status code and error code, so that when client is a non-human program, response to the status information can be pre-programmed.
- *SURL Statuses*
  Output reporting the success or failure of the each SURL requests. *SURL Statuses* may be empty and NULL. If returned to the client, *SURL* and its *status information* are required to return.
  - *SURL* (required)
    SURL that client has requested to change the file storage type.
  - *New Lifetime*
    lifetime in seconds that is newly assigned to the *SURL*.
- Flag for partial list
  Output parameter indicating if return values are partial or not. If true, *Request Token* must be returned. Default is false.
- Total number of Files In The Request

Output parameter indicating how many files in the request for a hint to the asynchronous status calls.

2.5.4. NOTES on the Core Behavior

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

2.5.5. 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.

2.5.6. NOTES on the Advanced Behavior with Space Management Feature

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

2.5.7. Status information returned upon request execution

On successful request, the *status information* returns SRM_SUCCESS.
On failure, the *status information* returns SRM_FAILURE and the detailed failure information along with the textual explanation in a human readable form would help the client diagnose the failed request. If only partial files were successful, the *status information* returns SRM_PARTIAL_SUCCESS, and the *SURL Statuses* should explain on those failed files.
When detailed failure can be set to one of the followings:

For request level status,

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
- *Space Token* does not refer to an existing space.

For file level status,

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*

2.5.8. SEE ALSO

ChangeFileStorageTypeStatus, PrepareToPut, RemoteCopy, ExtendFileLifetime, ReserveSpace

2.6. ChangeFileStorageTypeRequestStatus

2.6.1. NAME

ChangeFileStorageTypeRequestStatus - gets the response to the ChangeFileStorageType.

2.6.2. Functional Description

ChangFileStorageTypeRequestStatus() allows the asynchronous status request for the previously requested ChangeFileStorageType(). *Request Token* from the response to the previous ChangeFileStorageType() is required.

2.6.3. Parameter Description

In	Out

User ID
Authorization ID
Request Token
offset
count
	Request Status {
	Status information
	}
	SURL Statuses{
	SURL,
	New Lifetime,
	status information
	}

Input parameters

- *User ID* (required)
  User authentication identifier.
- *Authorization ID*
  User authorization information. The *authorizationID* may be NULL.
- *Request Token* (required)
  A handle associated with the previously submitted request to change file storage types. The *Request Token* was returned by the function initiating the request (e.g. *ChangeFileStorageType*()).
- *offset*
  Integer indicator for long listed responses that the listing starts from the *offset*. Default is 0 for the first entry.
- *count*
  Integer indicator for long listed responses that the listing contains the number of *count*. Default is to return all entries of the list.

Output parameters

- *Request Status* (required)
  Output reporting the success or failure of the request. Textual description of explanation for what happened to the request, specially in case of any failures, would help client diagnose the case. *Status information* is required and it is recommended to return as an enumerated codes separated as status code and error code, so that when client is a non-human program, response to the status information can be pre-programmed.
- *SURL Statuses*
  Output reporting the success or failure of the each SURL requests. *SURL Statuses* may be empty and NULL. If returned to the client, *SURL* and its *status information* are required to return.
  - *SURL* (required)

SURL that client has requested to change the file storage type.

- *New Lifetime*
  lifetime in seconds if it is newly assigned to the *SURL*.

2.6.4. Status information returned upon request execution

On successful request, the *status information* returns SRM_SUCCESS.
On failure, the *status information* returns SRM_FAILURE and the detailed failure information along with the textual explanation in a human readable form would help the client diagnose the failed request. If only partial files were successful, the *status information* returns SRM_PARTIAL_SUCCESS, and the *SURL Statuses* should explain on those failed files.
When detailed failure can be set to one of the followings:

For request level status,

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 status,

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*

2.6.5. SEE ALSO

ChangeFileStorageType

2.7. ExtendFileLifetime

2.7.1. NAME

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

2.7.2. Functional Description

ExtendFileLifetime() allows clients to extend lifetime of existing *SURL*s of volatile and durable file storage types or pinning lifetime of *TURLs*. Those *TURLs* are of the results of *PrepareToGet* and *PrepareToPut*.

2.7.3. Parameter Description

In	Out
User ID	Request Status {
Authorization ID	Status information
Request Token	}

SURL
New File Lifetime
New Pin Lifetime
	SURL Status {
	SURL,
	TURL,
	New File Lifetime,
	New Expiration Time,
	Status information
	}

Input parameters

- *User ID* (required)
  User authentication identifier.
- *Authorization ID*
  User authorization information. The *Authorization ID* may be NULL.
- *Request Token*
  A handle associated with the previously submitted request (e.g. *PrepareToGet, PrepareToGetRequestStatus).*
- *SURL* (required)
  SURL to extend the file lifetime.
- *New File Lifetime*
  Desired lifetime in seconds to extend the *SURL*. Default will be assigned by SRM if not provided. Default varies by the SRM system.
- *New Pin Lifetime*
  Desired lifetime in seconds to extend the *TURL*. Default will be assigned by SRM if not provided. Default varies by the SRM system.

Output parameters

- *Request Status* (required)
  Output reporting the success or failure of the request. Textual description of explanation for what happened to the request, specially in case of any failures, would help client diagnose the case. *Status information* is required and it is recommended to return as an enumerated codes separated as status code and error code, so that when client is a non-human program, response to the status information can be pre-programmed.
- SURL Status
  Output parameter reporting the status of the file in the request. *SURL Status* may be empty and NULL. If returned to the client, *SURL* and its *status information* are required to return.
  - *SURL* (required)
    SURL that client has requested to extend the expiration time.
  - *New File Lifetime*
    New lifetime in secondse that is newly assigned to the *SURL*.
  - *New Expiration Time*
    New expiration time in UTC time that is newly assigned to the *TURL*.

2.7.4. NOTES on the Core Behavior

a) *New Pin Lifetime* and *New File Lifetime* are relative to the calling time. Lifetime will be set from the calling time for the specified period.
b) *New File Lifetime* is the lifetime duration requested.
c) When extending pinning lifetime of *TURLs* with *New Pin Lifetime*, *Request Token* and *SURLs* must be provided.
d) When extending lifetime of *SURLs* with *New File Lifetime*, *Request Token* is optional.
e) The number of lifetime extensions may be limited by SRM according to its policies.
f) If original lifetime is longer than the requested one, then the requested one will be assigned.
g) If *New Pin Lifetime* or *New File Lifetime* is not specified, the SRM can use its default to assign the *New Pin Lifetime* or *New File Lifetime*
h) Lifetime cannot be extended on the released files, aborted files, expired files, and suspended files.
i) Extending file lifetime on SURL is similar to *ExtendFileLifetimeInSpace*.

2.7.5. Status information returned upon request execution

On successful request, the *status information* returns SRM_SUCCESS.
On failure, the *status information* returns SRM_FAILURE and the detailed failure information along with the textual explanation in a human readable form would help the client diagnose the failed request.
When detailed failure can be set to one of the followings:

For request level status,

SRM_SUCCESS
- All requests are successfully completed. All *SURLs* or *TURLs* associated with *SURLs* in the specified request have an extended lifetime. Details are on the files status.

SRM_PARTIAL_SUCCESS
- All requests are completed. Lifetimes on some *SURLs* or *TURLs* are successfully extended, and lifetimes on some *SURLs* or *TURLs* are failed to be extended. Details are on the files status.

SRM_AUTHENTICATION_FAILURE
- SRM server failed to authenticate the client

SRM_AUTHORIZATION_FAILURE
- client is not authorized to extend file lifetime

SRM_INVALID_REQUEST
- *Request Token* does not refer to an existing known request in the SRM server.
- *Request Token* is not provided, and extending pinning lifetime of *TURLs* associated with *SURLs* require *Request Token*.

SRM_INTERNAL_ERROR
- SRM has an internal transient error, and client may try again.

SRM_FAILURE
- All files requests are failed.
- any other request failure. *Explanation* needs to be filled for details.

SRM_NOT_SUPPORTED
- *function* is not supported in the SRM

For file level status,

SRM_SUCCESS
- successful request completion for the *SURL*. *SURL* or *TURL* associated with the *SURL* in the request has an extended lifetime.

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

SRM_FILE_LIFETIME_EXPIRED
- Lifetime on *SURL* is expired already.

SRM_ABORTED
- The requested file has been aborted.

SRM_RELEASED
- The requested file has been released.

SRM_FAILURE
- The requested file has been suspended because the request has timed out.
- any other request failure. *Explanation* needs to be filled for details.

2.7.6. SEE ALSO
ExtendFileLifetimeInSpace, PrepareToPut, Ls

2.8. GetFeatures

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

2.8.2. Functional Description
GetFeatures 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.

2.8.3. Parameter Description

In	Out
User ID	Request Status {
Authorization ID	Status information
	}
	Supported SRM Features
	Supported File Storage Types
	Supported Retention Policy Modes
	Supported Access Latency Modes

SRM Features := SRM_CORE |
SRM_REMOTE_ACCESS |
SRM_SPACE_MANAGEMENT |
SRM_DIRECTORY_MANAGEMENT |
SRM_AUTHORIZATION |
SRM_ADMINISTRATION |

SRM_ACCOUNTING

Input parameters

- *User ID* (required)
  User authentication identifier.
- *Authorization ID*
  User authorization information. The A*uthorization ID* may be NULL.

Output parameters

- Request Status (required)
  Output reporting the success or failure of the request. Textual description of explanation for what happened to the request, specially in case of any failures, would help client diagnose the case. *Status information* is required and it is recommended to return as an enumerated codes separated as status code and error code, so that when client is a non-human program, response to the status information can be pre-programmed.
- Supported SRM Features (required in a successful case)
  Output parameter returning the list of features that the SRM supports.
- Supported File Storage Types
  Output parameter returning the list of File Storage Types that the SRM supports.
- Supported Retention Policy Modes
  Output parameter returning the list of Retention Policy Modes that the SRM supports.
- Supported Access Latency Modes
  Output parameter returning the list of Access Latency Modes that the SRM supports.

2.8.4. Status information returned upon request execution

On successful request, the *status information* returns SRM_SUCCESS.
On failure, the *status information* returns SRM_FAILURE and the detailed failure information along with the textual explanation in a human readable form would help the client diagnose the failed request.
When detailed failure can be set to one of the followings:

SRM_AUTHENTICATION_FAILURE
- SRM server failed to authenticate the client

SRM_AUTHORIZATION_FAILURE
- Client is not authorized to request SRM features

2.8.5. SEE ALSO

GetRequestSummary, GetRequestTokens, GetSRMStorageInfo, GetTransferProtocols, GetSpaceTokens

2.9. GetRequestSummary

2.9.1. NAME

GetRequestSummary - is to retrieve a summary of the submitted request.

2.9.2. Functional Description

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

2.9.3. Parameter Description

In	Out

User ID
Authorization ID
Request Tokens
	Request Status {
	Status information
	}
	Request Summaries {
	Request Token,
	Status information
	Request Type
	Total Number Of Files In The Request,
	Number Files Completed,
	Number Files Waiting,
	Number Files Failed
	}

Input parameters

- *User ID* (required)
  User authentication identifier.
- *Authorization ID*
  User authorization information. The *Authorization ID* may be NULL.
- *Request Tokens* (required)
  A handle associated with the previously submitted request. The *Request Token* was returned by the function initiating the request (e.g. *PrepareToGet*()).

Output parameters

- Request Status (required)
  Output reporting the success or failure of the request. Textual description of explanation for what happened to the request, specially in case of any failures, would help client diagnose the case. *Status information* is required and it is recommended to return as an enumerated codes separated as status code and error code, so that when client is a non-human program, response to the status information can be pre-programmed.
- Request Summaries
  Output reporting the success or failure of the each request. *Request Summaries* may be empty and NULL. If returned to the client, *Request Token* and its *status information* are required to return.
  - *Request Token* (required)
    Request handle that client has requested to get the summary.
  - *Request Type*
    Output parameter reporting the type of the request (e.g. PREPARE_TO_GET).

- *Total Number Of Files In The Request*
  Output parameter reporting the total number of files in the request.
- *Number Files Completed*
  Output parameter reporting the total number of completed files in the request.
- *Number Files Waiting*
  Output parameter reporting the total number of files on the queue in the request.
- *Number Files Failed*
  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.

2.9.4. Status information returned upon request execution

On successful request, the *status information* returns SRM_SUCCESS.
On failure, the *status information* returns SRM_FAILURE and the detailed failure information along with the textual explanation in a human readable form would help the client diagnose the failed request. If only partial files were successful, the *status information* returns SRM_PARTIAL_SUCCESS, and the *SURL Statuses* should explain on those failed files.
When detailed failure can be set to one of the followings:

For request interface level status,

SRM_SUCCESS
- All requests are successfully completed. All requests summaries are checked and returned successfully. Details are on the request status.

SRM_PARTIAL_SUCCESS
- All requests are completed. Summaries of some requests are successfully checked and returned, but some requests summaries are failed. Details are on the request status.

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 *Request Token*

SRM_INVALID_REQUEST
- *Request Tokens* is empty.

SRM_INTERNAL_ERROR
- SRM has an internal transient error, and client may try again.

SRM_NOT_SUPPORTED
- *function* is not supported in the SRM

SRM_FAILURE
- SRM failed to get summaries of all requests that are associated with request tokens.
- any other request failure. *Explanation* needs to be filled for details.

For request level status,

SRM_INVALID_REQUEST
- *Request Token* does not refer to an existing known request in the SRM server.

SRM_SUCCESS

- The request has been completed successfully.

SRM_REQUEST_QUEUED
- successful request submission and all files request is still on the queue

SRM_REQUEST_INPROGRESS
- some files are completed, and some files are still on the queue

SRM_REQUEST_TIMED_OUT
- Total request time is over and the request is failed.

SRM_REQUEST_SUSPENDED
- The request has been suspended.

SRM_ABORTED
- The request has been aborted.

SRM_PARTIAL_SUCCESS
- All requests are completed. Some request is successfully completed, and some request is failed.

SRM_FAILURE
- The request is failed. *Explanation* needs to be filled for details.

2.9.5. SEE ALSO
GetRequestTokens, PrepareToGetRequestStatus, PrepareToPutRequestStatus, RemoteCopyRequestStatus, GetSpaceTokens.

2.10. GetRequestTokens

2.10.1. NAME
GetRequestTokens - retrieves active request tokens for the previously submitted requests by the client.

2.10.2. Functional Description
GetRequestTokens retrieves active 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.

2.10.3. Parameter Description

In	Out

User ID
Authorization ID
User Request Description
	Request Status {
	Status information
	}
	Return Request Tokens {
	Request Token,
	Submitted Time
	}

Input parameters

- *User ID* (required)
  User authentication identifier.
- *Authorization ID*
  User authorization information. The *Authorization ID* may be NULL.
- *User Request Description*
  Description of the request. The description was given by the client at the time of the request (e.g. *srmPrepareToGet*). The *User Request Description* may be NULL.

Output parameters

- *Return Status* (required)
  Output reporting the success or failure of the request. Textual description of explanation for what happened to the request, specially in case of any failures, would help client diagnose the case. *Status information* is required and it is recommended to return as an enumerated codes separated as status code and error code, so that when client is a non-human program, response to the status information can be pre-programmed.
- Return Request Tokens
  Output parameter returning the request handles owned by the client. Return Request Tokens may be NULL and empty. If returned to the client, *Request Token* is required to be returned.

2.10.4. NOTES on Core Behavior

a) If *User Request Description* is not provided, all active 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.

2.10.5. Status information returned upon request execution

On successful request, the *status information* returns SRM_SUCCESS.
On failure, the *status information* returns SRM_FAILURE and the detailed failure information along with the textual explanation in a human readable form would help the client diagnose the failed request.
When detailed failure can be set to one of the followings:

SRM_SUCCESS
- successful request completion. Request tokens are returned successfully.

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
- *User Request Description* does not refer to any existing known requests

SRM_INTERNAL_ERROR
- SRM has an internal transient error, and client may try again.

SRM_FAILURE

- any other request failure. *Explanation* needs to be filled for details.

SRM_NOT_SUPPORTED

- *function* is not supported in the SRM

2.10.6. SEE ALSO

PrepareToGet, PrepareToPut, RemoteCopy

2.11. GetSRMStorageInfo

2.11.1. NAME

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

2.11.2. Functional Description

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

2.11.3. Parameter Description

In	Out

User ID
Authorization ID
Desired Storage Attributes
	Request Status {
	Status information
	}
	Supported Attributes {
	Storage Attribute,
	Storage Attribute Value
	}

Storage Attributes := 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

Input parameters

- *Use rID* (required)
  User authentication identifier.
- *Authorization ID*
  User authorization information. The *Authorization ID* may be NULL.

- *Desired Storage Attributes*
  The storage attributes that the client needs to find out. Examples of storage sttributes are listed above, but not limited to those.

Output parameters

- Request Status (required)
  Output reporting the success or failure of the request. Textual description of explanation for what happened to the request, specially in case of any failures, would help client diagnose the case. *Status information* is required and it is recommended to return as an enumerated codes separated as status code and error code, so that when client is a non-human program, response to the status information can be pre-programmed.
- Supported Storage Attributes
  Output parameter reporting the storage information. *Supported Storage Attributes* may be empty and NULL. If returned to the client, *Storage Attribute* and its *value* are required to return.
  - *Storage Attribute* (required)
    One of Storage Attributes that client has requested to find out the storage system info.
  - *Storage Attribute Value*
    A value of the *Storage Attribute*. It may include the type of the value.

2.11.4. 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.

2.11.5. Status information returned upon request execution

On successful request, the *status information* returns SRM_SUCCESS.
On failure, the *status information* returns SRM_FAILURE and the detailed failure information along with the textual explanation in a human readable form would help the client diagnose the failed request.
When detailed failure can be set to one of the followings:

SRM_AUTHENTICATION_FAILURE
- SRM server failed to authenticate the client

SRM_AUTHORIZATION_FAILURE
- client is not authorized to request storage information

SRM_NOT_SUPPORTED
- *Storage Attribute* is not supported in the SRM

2.11.6. SEE ALSO

GetTransferInfo, GetFeatures

2.12. GetTransferInfo

2.12.1. NAME
GetTransferInfo - returns information on the supported file transfer protocols in the SRM.

2.12.2. Functional Description
GetTransferInfo returns information on the supported file transfer protocols in the SRM with any additional information.

2.12.3. Parameter Description

In	Out

User ID
Authorization ID
	Request Status {
	Status information
	}
	Transfer Infos {
	Transfer Protocol,
	attributes,
	Access Pattern,
	Connection Type,
	Supported Client Network
	}

Input parameters

- *Use rID* (required)
  User authentication identifier.
- *Authorization ID*
  User authorization information. The *Authorization ID* may be NULL.

Output parameters

- Request Status (required)
  Output reporting the success or failure of the request. Textual description of explanation for what happened to the request, specially in case of any failures, would help client diagnose the case. *Status information* is required and it is recommended to return as an enumerated codes separated as status code and error code, so that when client is a non-human program, response to the status information can be pre-programmed.
- Transfer Infos
  Output parameter reporting the supported file transfer protocol list.
  - *Transfer Protocol* (required)
    Supported transfer protocol. For example, http.
  - *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.

2.12.4. Status information returned upon request execution

On successful request, the *status information* returns SRM_SUCCESS.
On failure, the *status information* returns SRM_FAILURE and the detailed failure information along with the textual explanation in a human readable form would help the client diagnose the failed request.
When detailed failure can be set to one of the followings:

SRM_SUCCESS
- successful request completion. List of supported transfer protocols are returned successfully.

SRM_AUTHENTICATION_FAILURE
- SRM server failed to authenticate the client

SRM_AUTHORIZATION_FAILURE
- client is not authorized to request storage information

SRM_INTERNAL_ERROR
- SRM has an internal transient error, and client may try again.

SRM_NOT_SUPPORTED
- *function* is not supported in the SRM

SRM_FAILURE
- any other request failure. *Explanation* needs to be filled for details.

2.12.5. SEE ALSO

GetSRMStorageInfo, GetFeatures, Ping

2.13. Ls

2.13.1. NAME

Ls - returns a list of files with a basic information.

2.13.2. Functional Description

Ls returns a list of files with a basic information. This operation may be asynchronous, and in such case, *Request Token* must be returned.

2.13.3. Parameter Description

In	**Out**
User ID	Request Token
Authorization ID	Request Status {
Storage System Info	Status information
SURLs	}
Directory Option	SURL Meta Data Info {
offset	SURL,
count	Status information
	File size,
	Last Modification Time,

Lifetime Left,
File Type,
File Locality,
SURL Meta Data Info Subpaths
}
Total Number of Files In The Request

Input parameters

- *User ID* (required)
  User authentication identifier.
- *Authorization ID*
  User authorization information. The *Authorization ID* may be NULL.
- *Storage System Info*
  Information specific to the underlying storage system. The *Storage System Info* may be NULL.
- *SURLs*
  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.
- *Directory Option*
  An advanced option when Directory Management feature is supported. It includes *Flag for Directory Listing Only, Flag for All Level Recursive* and *Number Of Recursive Levels*: *Flag for Directory Listing Only* is a boolean indicator if directory listing only under the *Source SURL* must be returned. *Flag for All Level Recursive* is a boolean indicator if *Source SURL* is a directory and all files under the SURL must be retrieved. *Number Of Recursive Levels* is a positive integer indicator for how many levels of the recursive directory must be browsed and all files in those directories to be retrieved.
- *offset*
  Integer indicator for long listed responses that the listing starts from the *offset*. Default is 0 for the first entry.
- *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.

Output parameters

- *Rrequest Token*
  Output parameter request token is associated with the request for the later asynchronous status request. *Request Token* may be NULL, in case *srmLs* is processed without delay.
- *Request Status* (required)
  Output reporting the success or failure of the request. Textual description of explanation for what happened to the request, specially in case of any failures, would help client diagnose the case. *Status information* is required and it is recommended to return as an enumerated codes separated as status code and error code, so that when client is a non-human program, response to the status information can be pre-programmed.

- *SURL Meta Data Info*
  Output parameter reporting the information of each file in the request. *SURL Meta Data Info* may be empty and NULL. If returned to the client, *SURL, File Size*, and its *File Type* are required to return.
  - *SURL* (required)
    SURL that client has requested for the info.
  - *Status information*
    Status information on the *SURL*.
  - *File size* (required)
    Size of the *SURL*. In case of *SURL* being a directory for advanced features, *size* is expected to be 0.
  - *Last Modification Time*
    Last mofified time that is associated with the *SURL*.
  - *Lifetime Left*
    Remaining lifetime that is associated with the *SURL*.
  - *File Type* (required)
    File type associated with the *SURL*.
  - *File Locality*
    Locality information that is associated with the *SURL*.
  - *SURL Meta Data Info subpaths*
    An advanced option when Directory Management feature is supported. In case of the recursive directory *SURL* listing, *SURL Meta Data Info subpaths* indicate sub-directories that contain files or directoties.
- Total Number Of Files In The Request
  Output parameter indicating how many files in the request for a hint to the asynchronous status cals.

2.13.4. NOTES on the Core Behavior

a) *SURL* refers to files only.
b) When only part of the result was returned, a *Request Token* must be returned.
c) If the entire result is returned, then output parameter *Request Token* is optional.
d) *Status information* for the *SURL* in *SURL Meta Data Info* 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.

2.13.5. NOTES on the Advanced Behavior with Directory Management Feature

a) *SURL*s may refer to either directory or file.
b) If *SURL* refers to a directory, the returned value *size* must be 0.
c) If *Flag for all Level Recursive* is "true", it dominates, i.e. ignore *Number Of Recursive Levels*.
d) If *Flag for all Level Recursive* is "false" or missing, then *Number Of Recursive Levels* must be honored. If *Number Of Recursive Levels* is "0" (zero), a single level is assumed.
e) Directory names are returned even if they are empty.

2.13.6. Status information returned upon request execution

On successful request, the *status information* returns SRM_SUCCESS.
On failure, the *status information* returns SRM_FAILURE and the detailed failure information along with the textual explanation in a human readable form would help the client diagnose the failed request. If only partial files were successful, the *status information* returns SRM_PARTIAL_SUCCESS, and the *SURL Statuses* should explain on those failed files.
When detailed failure can be set to one of the followings:

For request level Status,

SRM_SUCCESS
- All requests are successfully completed. All *SURLs* are checked and the information for all *SURLs* is returned successfully.

SRM_PARTIAL_SUCCESS
- All requests are completed. Some *SURL* request is successfully completed, and some *SURL* request is failed. Details are on the files status.

SRM_REQUEST_QUEUED
- successful request submission and acceptance. Request token must be returned.

SRM_AUTHENTICATION_FAILURE
- SRM server failed to authenticate the client

SRM_AUTHORIZATION_FAILURE
- client is not authorized to request information

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_INTERNAL_ERROR
- SRM has an internal transient error, and client may try again.

SRM_INVALID_REQUEST
- Negative values for *Number of Recursive Levels*, *offset* and *count* are provided.

SRM_NOT_SUPPORTED
- Directory operation is not supported in SRM

SRM_FAILURE
- All files requests are failed.
- any other request failure. *Explanation* needs to be filled for details.

For file level status,

SRM_SUCCESS
- successful request completion for the *SURL*. The information for the *SURL* is checked and returned successfully.

SRM_FILE_IN_CACHE
- lifetime on *SURL* has expired, but the file is still in the cache.

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

SRM_AUTHORIZATION_FAILURE

- client is not authorized to receive the information of the *SURL* or to access the directory or sub-directories

SRM_FAILURE

- any other request failure. *Explanation* needs to be filled for details.

2.13.7. SEE ALSO

LsRequestStatus, LsDetails, GetSpaceMetaData, GetSpaceTokens

2.14. LsRequestStatus

2.14.1. NAME

LsRequestStatus - is an asynchronous call for Ls that is large.

2.14.2. Functional Description

LsRequestStatus() retrieves the status of the previously requested srmLs. See also 2.12 for more descriptions.

2.14.3. Parameter Description

In	Out
User ID	Request Status {
Authorization ID	Status information
Request Token	}
offset	SURL Meta Data Info {
count	SURL,
	status information
	File size,
	Last Modification Time,
	File Type,
	File Locality,
	SURL Meta Data Info Subpaths
	}

Input parameters

- *User ID* (required)
  User authentication identifier.
- *Authorization ID*
  User authorization information. The *Authorization ID* may be NULL.
- *Request Token* (required)
  A handle associated with the previously submitted Ls request. The *Request Token* was returned by the function initiating the request (e.g. *Ls*()).
- *offset*
  Integer indicator for long listed responses that the listing starts from the *offset*. Default is 0 for the first entry.
- *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.

Output parameters

- *Request Status* (required)
  Output reporting the success or failure of the request. Textual description of explanation for what happened to the request, specially in case of any failures, would help client diagnose the case. *Status information* is required and it is recommended to return as an enumerated codes separated as status code and error code, so that when client is a non-human program, response to the status information can be pre-programmed.
- *SURL Meta Data Info*
  Output parameter reporting the information of each file in the request. *SURL Meta Data Info* may be empty and NULL. If returned to the client, *SURL*, *File Size*, and its *File Type* are required to return.
  - *SURL* (required)
    SURL that client has requested for the info.
  - *Status information*
    Status information of the *SURL*.
  - *File Size* (required)
    Size of the *SURL*. In case of *SURL* being a directory for advanced features, *size* is expected to be 0.
  - *Last Modification Time*
    Last modified time that is associated with the *SURL*.
  - *File Type* (required)
    File type associated with the *SURL*.
  - *File Locality*
    Locality information that is associated with the *SURL*.
  - *SURL Meta Data Info subpaths*
    An advanced option when Directory Management feature is supported. In case of the recursive directory *SURL* listing, *SURL Meta Data Info subpaths* indicate sub-directories that contain files or directoties.

2.14.4. Status information returned upon request execution

On successful request, the *status information* returns SRM_SUCCESS.
On failure, the *status information* returns SRM_FAILURE and the detailed failure information along with the textual explanation in a human readable form would help the client diagnose the failed request. If only partial files were successful, the *status information* returns SRM_PARTIAL_SUCCESS, and the *SURL Statuses* should explain on those failed files. See also 2.12.4.
When detailed failure can be set to one of the followings:

For request level Status,
SRM_SUCCESS
- All requests are successfully completed. All *SURLs* are checked and the information for all *SURLs* is returned successfully.

SRM_PARTIAL_SUCCESS
- All requests are completed. Some *SURL* request is successfully completed, and some *SURL* request is failed. Details are on the files status.

SRM_REQUEST_QUEUED
- successful request submission and acceptance. Request token must be returned.

SRM_AUTHENTICATION_FAILURE
- SRM server failed to authenticate the client

SRM_AUTHORIZATION_FAILURE
- client is not authorized to request information

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_INTERNAL_ERROR
- SRM has an internal transient error, and client may try again.

SRM_INVALID_REQUEST
- Negative values for *Number of Recursive Levels*, *offset* and *count* are provided.

SRM_NOT_SUPPORTED
- Directory operation is not supported in SRM

SRM_FAILURE
- All files requests are failed.
- any other request failure. *Explanation* needs to be filled for details.

For file level status,

SRM_SUCCESS
- successful request completion for the *SURL*. The information for the *SURL* is checked and returned successfully.

SRM_FILE_IN_CACHE
- lifetime on *SURL* has expired, but the file is still in the cache.

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

SRM_AUTHORIZATION_FAILURE
- client is not authorized to receive the information of the *SURL* or to access the directory or sub-directories

SRM_FAILURE
- any other request failure. *Explanation* needs to be filled for details.

2.14.5. SEE ALSO
Ls, LsDetails, GetSpaceMetaData, GetRequestTokens, GetSpaceTokens

2.15. Ping

2.15.1. NAME
ping - is to check the state of the SRM.

2.15.2. Functional Description
This function is used to check the state of the SRM. It works as an "are you alive" type of call.

2.15.3. Parameter Description

In	Out
User ID	Version Info
Authorization ID	Other Info

Input parameters

- *User ID* (required)
  User authentication identifier.
- *Authorization ID*
  User authorization information. The *Authorization ID* may be NULL.

Output parameters

- *Version Info* (required)
  Output string containing SRM v3.0 version number as a minimal "up and running" information. For this particular SRM v3.0 version, it must be "**v3.0**". Other versions may have "**v1.1**", "**v2.2**", and so on..
- *Other Info*
  Any additional information about the SRM can be provided.

2.15.4. SEE ALSO
GetSRMStorage Info, GetTransferInfo

2.16. PrepareToGet

2.16.1. NAME
This function is used to bring files to the local online storage upon the client's request and assign TURL so that client can access the file. The term "local" storage and "local" files are used to mean that the storage spaces and files are under the SRM's administrative control. Lifetime (pinning expiration time) is assigned on the TURL. When specified target space token which must be referred to an online space, the files will be prepared using the space associated with the space token. It is an asynchronous operation, and request token must be returned if request is valid and accepted. The status must be checked through *PrepareToGetRequestStatus* with the returned request token.

2.16.2. Functional Description
The *PrepareToGet* function allows clients to request the storage system to prepare a file or set of files in the storage system for a transfer using the highest possible protocol specified transfer protocol list. The function has the effect of pinning each file that is already in an on-line cache with a specific lifetime (that can be negotiated). If files are not in on-line cache, they will be

copied from near-line storage to on-line cache, and assigned a lifetime. The number of files pinned in cache depends on the storage allocation assigned to the client and/or the local policy. If there is not enough space allocated and "streaming" option is supported by the SRM, only some subset of the files will be placed in cache and pinned (the order of which file to be placed first is the SRM's choice). Upon performing *Release* of a file, the freed space will be used to bring additional files.

Files in the request
PrepareToGet function receives an unordered set of File Descriptions describing the files that client is interested in transferring from the Storage System managed by SRM. Each of the File Description data structure must contain SURL – Site URL of the file being accessed. Each File Description consists of parameters described in the "parameters description" section.

srmPrepareToGet execution
PrepareToGet function execution leads to the creation of the request on the server. The request consists of the collection of the file requests, one for each File Description in the request. The state of each file requests persists on the server at least for the duration of the lifetime of the request. The client should be given the handle (or collection of handles), specific to a messaging mechanism, allowing the client to refer to the request as a whole or file requests individually. The srmPrepareToGet function must be an asynchronous non-blocking call returning the handle of the request back to the client as soon as possible.

Once the Server receives the request to prepare certain files to get, it can make files available for the transfer in one of the specified protocols. Depending on the architecture of the storage system it might be as simple as just an assignment of the Transfer URL or as complex as getting the file copied from the near-line storage like tape robot into an online storage like disk cache and starting a new server process that will be serving the file in the specified protocol. Once the file is made available and the Transfer URL is assigned and made available to the client, the execution of the request is complete.

2.16.3. Parameter Description

In	Out
User ID	Request Token

Authorization ID
User Request Description
Total Request Time
Flag For Streaming
Overwrite Mode,
Get File Requests {
Source SURL,
Directory Option
}
Desired Pin Lifetime,
Target Space Token,
Target File Retention Policy Info,
Transfer Information,
Target File Storage Type,
Source Storage System Info,
Flag For Checksum
	Request Status {
	Status information
	}
	Remaining Total Request Time
	SURL statuses {
	Source SURL,
	Status information,
	File Size,
	Estimated Wait Time,
	Remaining Pin Time,
	Transfer URL,
	Transfer Protocol Info
	}

Input parameters

- *User ID* (required)
  User authentication identifier.
- *Authorization ID*
  User authorization information. The *Authorization ID* may be NULL.
- *Transfer Information*
  Information about the transfer protocols that client can support. It may include Access Pattern, Connection Types, the list of Client Networks, and the Transfer Protocol Info.
- *User Request Description*
  Textual description of the request. It may be used for later retrieval of the rquest token.
- *Total Request Time*
  Total Request Time means that all the file transfer for this request must be complete within this *Total Request Time*. Otherwise, an error, SRM_REQUEST_TIMED_OUT must be returned as the request status information with individual file status of SRM_FAILURE with an appropriate explanation.
- *Flag for 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.
- *Overwrite Mode*
  In case SRM stages requested files for the user, SRM needs to stage the file according to the *Overwrite Mode* on the target that SRM brings the files into.
- Get File Requests (required)
  It consists of SURL information that client wants to retrieve, in an unordered set.
  - *Source SURL* (required)
    SURL that client desires to retrieve
  - *Directory Option*
    An advanced option when Directory Management feature is supported. It includes *Flag for Source Directory, Flag for All Level Recursive* and *Number*

Of Recursive Levels: *Flag for Source Directory* is a boolean indicator if *Source SURL* is a directory. *Flag for All Level Recursive* is a boolean indicator if *Source SURL* is a directory and all files under the SURL must be retrieved. *Number Of Recursive Levels* is a positive integer indicator for how many levels of the recursive directory must be browsed and all files in those directories to be retrieved.

- *Desired Pin Lifetime*
  Desired Pin Lifetime is for a client preferred lifetime (expiration time) on the prepared TURL. Desired pin life time of the file is in seconds once when the file is ready. SRM may assign a default lifetime based on the policy. In the status call, expiration time in UTCtime on Transfer URL is returned.
- *Target Space Token*
  Target Space Token indicates which spaces would be used when *Source SURL*s are brought in online. *Target Space Token* must refer to online space. All requested files will be prepared into the target space.
- *Target File Retention Policy Info*
  Target File Retention Policy Info indicates which space information would be used when *Source SURL*s are brought in online. It is to specify the desired retention policy information on the file when the file is prepared online.
- *Target File Storage Type*
  Target File Storage Type indicates which type of file storage is used when the file is prepared online.
- *Source Storage System Info*
  Information specific to the underlying storage system that is associated with the *Source SURLs*. The *Source Storage System Info* may be NULL.
- *Flag for Checksum*
  Flag for Checksum indicates that checksum calculation for all files in the request needs to be performed when SRM server brings files into its space.

Output parameters

- *Request Token* (required)
  Output parameter request handle is associated with the request for the later asynchronous status request.
- *Return Status* (required)
  Output reporting the success or failure of the request. Textual description of explanation for what happened to the request, specially in case of any failures, would help client diagnose the case. *Status information* is required and it is recommended to return as an enumerated codes separated as status code and error code, so that when client is a non-human program, response to the status information can be pre-programmed.
- *Remaining Total Request Time*
  Output parameter indicates the remaining total request time.
- SURL Statuses
  Output parameter reporting the status of the file in the request. *SURL Statuses* may be empty and NULL. If returned to the client, *SURL* and its *status information* are required to return.

- *Source SURL* (required)
  Source SURL that client has requested to bring online.
- *Estimated Wait Time*
  Time estimation on the file to prepare online
- *Remaining Pin Time*
  When the *Source SURL* is ready, Transfer URL is prepared with Pin Lifetime. It indicates the remaining pin lifetime (expiration time) in UTC time on the TURL.

2.16.4. 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 *Request Token*. If the submitted request is accepted, SRM assigns the *Request Token* for asynchronous status checking. In such case, the returned status information should include SRM_REQUEST_QUEUED.

b) The default value of "lifetime" for Volatile or Durable files will be the lifetime left in the space of the corresponding file type. The default value of *File Storage Type* is Volatile.

c) If input parameter *Target Space Token* is provided, then the target space token must refer to online space. All requested files will be prepared into the target space.

d) Input parameter *Target File Retention Policy* is to specify the desired retention policy info on the file when the file is prepared online.

e) If both input parameters *Target Space Token* and *Target File Retention Policy Info* are provided, then their types must match exactly. Otherwise, the request may be rejected with SRM_INVALID_REQUEST.

f) Access latency must be ONLINE always, when provided.

g) *Access Pattern* may conflict with the type of the target space associated with target space token, when both provided. In this case, *Access Pattern* in the input parameter *Transfer Information* must be ignored.

h) *Source SURL* must be local to SRM.

i) The *User Request Description* 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.

j) SRM assigns the *Request Token* for the request.

k) The returned *Request Token* should be valid until all files in the request are released or removed.

l) Only pull mode is supported for file transfers that client must pull the files from the TURL within the expiration time (*Remaining Pin Time*).

m) ClientNetworks is a hint of the client IPs that some SRMs can use for optimization of its internal storage systems based on the client's accessible IP addresses.

n) File transfer protocols are specified in a preferred order on all SRM transfer functions.

o) *Source Storage System Info*, *Source File Storage Type* or *Desired Lifetime* may be provided at the request level and the file level. In such case, SRM uses the one provided at the file level.

p) *Total Request Time* means that all the file transfer for this request must be completed within this time. Otherwise SRM_REQUEST_TIMEOUT must be returned as the request status information with individual file status of SRM_FAILURE with an appropriate explanation. If *Total Request Time* is unspecified as NULL, the request will be retried for a duration which

is dependent on the SRM implementation. If *Total Request Time* is 0 (zero), each file request will be tried at least once.

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

r) 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.

s) *ReleaseFile* 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.

2.16.5. NOTES on the Advanced Behavior with Directory Management Feature

a) *Number Of Recusrive Levels* must be a valid input parameter, and must be a positive integer.

2.16.6. Status information returned upon request execution

On successful request, the *status information returns* SUCCESS.

On failure, the *status information returns* FAILURE and the detailed failure information along with the textual explanation in a human readable form would help the client diagnose the failed request.

When detailed failure can be set to one of the followings:

For request level status,

SRM_REQUEST_QUEUED
- successful request submission and acceptance. Request token must be returned.

SRM_REQUEST_INPROGRESS
- some files are completed, and some files are still on the queue. Request token must be returned.

SRM_SUCCESS
- all file requests are successfully completed. All *Source SURLs* are successfully pinned. For *TURLs*, file level status needs to be checked.

SRM_PARTIAL_SUCCESS
- All requests are completed. Some file request is successfully pinned, and some file request is failed. Details are on the files status.

SRM_AUTHENTICATION_FAILURE
- SRM server failed to authenticate the client

SRM_AUTHORIZATION_FAILURE
- client is not authorized to submit the request

SRM_INVALID_REQUEST
- *Get File Request* is empty
- If both input parameters *Target Space Token* and *Target File Retention Policy Info* are provided, then their types must match exactly.
- Access latency is something other than ONLINE.
- *Target Space Token* does not refer to an existing known space in the SRM server.

SRM_SPACE_LIFETIME_EXPIRED

- space associated with the *Target Space Token* is expired.

SRM_EXCEED_ALLOCATION
- space associated with the *Target Space Token* is not enough to hold all requested *SURLs*.

SRM_NO_USER_SPACE
- user space is not enough to hold all requested *SURLs*.

SRM_NO_FREE_SPACE
- SRM space is not enough to hold all requested *SURLs* for free. When client does not specify the *Target Space Token*, SRM uses a default space. The default space is not sufficient to accommodate the request.

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. Client requested *Desired File Storage Type* that is not supported by the SRM server.
- *Target File Retention Policy Info* does not refer to a supported retention policy in the SRM server.
- Directory operation is not supported in the SRM server.
- Recursive directory operation is not supported in the SRM server.
- None of the file transfer protocols are supported in the SRM server.

SRM_INTERNAL_ERROR
- SRM has an internal transient error, and client may try again.

SRM_FAILURE
- All files requests are failed.
- any other request failure. *Explanation* needs to be filled for details.

For file level return status,

SRM_FILE_PINNED
- successful request completion for the *Source SURL*. *Source SURL* is successfully pinned, and *TURL* is available for access.

SRM_REQUEST_QUEUED
- file request is still on the queue.

SRM_REQUEST_INPROGRESS
- file request is being served.

SRM_ABORTED
- The requested file has been aborted.

SRM_RELEASED
- The requested file has been released.

SRM_FILE_LOST
- the requested file is permanently lost.

SRM_FILE_BUSY
- client requests for files which there is an active *PrepareToPut* (no *PutDone* is not yet called) for.

SRM_FILE_UNAVAILABLE
- the requested file is temporarily unavailable.

SRM_INVALID_PATH

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

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

SRM_FILE_LIFETIME_EXPIRED
- *SURL* is expired
- *TUR*L is expired
- pin lifetime on TURL has expired, but the file is still in the cache.

SRM_FAILURE
- any other request failure. *Explanation* needs to be filled for details.
- The file request would not be able to be completed within the *total Request Time*.
- The requested file has been suspended because the request has timed out.

2.16.7. SEE ALSO
PrepareToGetRequestStatus, PrepareToPut, RemoteCopy

2.17. PrepareToGetRequestStatus

2.17.1. NAME
PrepareToGetRequestStatus - is the status call for *PrepareToGet*.

2.17.2. Functional Description
PrepareToGetRequestStatus is the status call for *PrepareToGet*. Request token from *PrepareToGet* must be provided.

2.17.3. Parameter Description

In	Out

User ID
Authorization ID
Request Token
Source SURLs
offset
count
	Request Status {
	Status information
	}
	SURL Statuses {
	Source SURL,
	Transfer URL,
	Status information
	File Size,
	Estimated Wait Time,
	Remaining Pin Lifetime,
	Checksum Type,
	Checksum Value,
	}
	Transfer Information
	Remaining Total Request Time

Input parameters

- *User ID* (required)
  User authentication identifier.
- *Authorization ID*
  User authorization information. The *Authorization ID* may be NULL.
- *Request Token* (required)
  A request handle associated with the previously submitted *srmPrepareToGet* request. The *Request Token* was returned by the function initiating the request.
- *Source SURLs*
  SURLs to request their status.
- *offset*
  Integer indicator for long listed responses that the listing starts from the *offset*.
- *count*
  Integer indicator for long listed responses that the listing contains the number of *count.*

Output parameters

- *Request Status* (required)
  Output reporting the success or failure of the request. Textual description of explanation for what happened to the request, specially in case of any failures, would help client diagnose the case. *Status information* is required and it is recommended to return as an enumerated codes separated as status code and error code, so that when client is a non-human program, response to the status information can be pre-programmed.
- *SURL Statuses*
  Output parameter reporting the status of each file in the request. *SURL Statuses* may be empty and NULL. If returned to the client, *Source SURL* and its *status information* are required to return.
  - *Source SURL* (required)
    SURL that client has requested.
  - *Transfer URL*
    Transfer URL that SRM server prepared for client to transfer for the corresponding *Source SURL*.
  - *File Size*
    File size in bytes for the *Source SURL*.
  - *Estimated Wait Time*
    Time estimation on the file to prepare online.
  - *Remaining Pin Lifetime*
    Life time in seconds that is remained on the prepared *TURL*.
  - *Checksum Type*
    Checksum type if check is performed. For example, MD5.
  - *Checksum Value*
    Checksum value if check is performed.
- *Transfer Information*

This output parameter can be used to provide more information about the transfer protocol so that client can access the TURL efficiently.

- *Remaing Total Request Time*
  Output parameter indicating how much time left from the *Total Request Time*.

2.17.4. NOTES on the Core Behavior

a) If *Source SURL* is not provided, the status for all the files associated with the *Request Token* is returned.

b) Output parameter *Source SURL* is the same as the *Source SURL* that client provided with *PrepareToGet*().

c) When the file is ready and TURL is prepared, the return status code should be SRM_FILE_PINNED.

d) When the file is ready for the client, the file is implicitly pinned in the cache and lifetime will be enforced, subject to the policies associated with the underlying storage.

e) If any of the request files is temporarily unavailable, SRM_FILE_UNAVAILABLE must be returned for the file.

f) If any of the request files is permanently lost, SRM_FILE_LOST must be returned for the file.

g) The file request must fail with an error SRM_FILE_BUSY if *PrepareToGet* requests for files which there is an active srmPrepareToPut (no srmPutDone is not yet called) for.

h) SRM must fail (SRM_FAILURE) only if all files in the request failed.

i) *Total Request Time* means: All the file transfer for this request must be complete within this *Total Request Time*. Otherwise, SRM_REQUEST_TIMED_OUT must be returned as the request status code with individual file status of SRM_FAILURE with an appropriate explanation. If *Total Request Time* is unspecified as NULL, the request will be retried for a duration which is dependent on the SRM implementation. If *Total Request Time* is 0 (zero), each file request will be tried at least once.

2.17.5. NOTES on the Advanced Behavior with Space Management Feature

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

2.17.6. Status information returned upon request execution

On successful request, the *status information* returns SRM_SUCCESS.
On failure, the *status information* returns SRM_FAILURE and the detailed failure information along with the textual explanation in a human readable form would help the client diagnose the failed request. If only partial files were successful, the *status information* returns SRM_PARTIAL_SUCCESS, and the *SURL Statuses* should explain on those failed files.
When detailed failure can be set to one of the followings:

For request level status,

SRM_SUCCESS

- all file requests are successfully completed. All *Source SURLs* are successfully pinned. For *TURLs*, file level status needs to be checked.

SRM_REQUEST_QUEUED

- successful request submission and all files request is still on the queue

SRM_REQUEST_INPROGRESS
- some files are completed, and some files are still on the queue

SRM_PARTIAL_SUCCESS
- All requests are completed. Some file request is successfully pinned, and some file request is failed. Details are on the files status.

SRM_AUTHENTICATION_FAILURE
- SRM server failed to authenticate the client

SRM_AUTHORIZATION_FAILURE
- client is not authorized to submit the request

SRM_INVALID_REQUEST
- *Request Token* does not refer to an existing known request in the SRM server.

SRM_SPACE_LIFETIME_EXPIRED
- space associated with the *Target Space Token* is expired.

SRM_EXCEED_ALLOCATION
- space associated with the *Target Space Token* is not enough to hold all requested SURLs.

SRM_NO_USER_SPACE
- user space is not enough to hold all requested *SURLs*.

SRM_NO_FREE_SPACE
- SRM space is not enough to hold all requested *SURLs* for free.

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. Client requested *Desired File Storage Type* that is not supported by the SRM server.
- *Target File Retention Policy Info* does not refer to a supported retention policy in the SRM server.
- Directory operation is not supported in the SRM server.
- Recursive directory operation is not supported in the SRM server.
- None of the file transfer protocols are supported in the SRM server.

SRM_ABORTED
- The request has been aborted.

SRM_REQUEST_TIMED_OUT
- Total request time is over and the rest of the request is failed.

SRM_INTERNAL_ERROR
- SRM has an internal transient error, and client may try again.

SRM_FAILURE
- All files requests are failed.
- any other request failure. *Explanation* needs to be filled for details.

For file level status,

SRM_FILE_PINNED
- successful request completion for the *SURL*. *SURL* is successfully pinned, and *TURL* is available for access.

SRM_REQUEST_QUEUED
- file request is still on the queue.

SRM_REQUEST_INPROGRESS
- file request is being served.

SRM_ABORTED
- The requested file has been aborted.

SRM_RELEASED
- The requested file has been released.

SRM_FILE_LOST
- the requested file is permanently lost.

SRM_FILE_BUSY
- client requests for files which there is an active *PrepareToPut* (no *PutDone* is not yet called) for.

SRM_FILE_UNAVAILABLE
- the requested file is temporarily unavailable.

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

SRM_AUTHORIZATION_FAILURE
- client is not authorized to retrieve the file that is associated with the *Source SURL*

SRM_FILE_LIFETIME_EXPIRED
- *SURL* is expired
- *TUR*L is expired
- pin lifetime on TURL has expired, but the file is still in the cache

SRM_FAILURE
- any other request failure. *Explanation* needs to be filled for details.
- The file request would not be able to be completed within the *Total Request Time*.
- The requested file has been suspended because the request has timed out.

2.17.7. SEE ALSO
PrepareToGet, BringOnline, PrepareToPut, RemoteCopy, Ls, LsDetails, ReleaseFiles, Rm, GetRequestTokens

2.18. PrepareToPut

2.18.1. NAME
PrepareToPut - 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.

2.18.2. Functional Description
The PrepareToPut function allows clients to request the storage system to prepare a local space for a file or set of files in the underlying storage system for a transfer using the highest possible protocol specified transfer protocol list. The function has the effect of reserving each file space in the underlying storage system with a specific lifetime that can be negotiated. The number of

file spaces reserved in the storage system depends on the storage allocation assigned to the client and/or the local policy. If there is not enough space allocated and "streaming" option is supported by the SRM, only some subset of the files will be placed in the storage system. If the target destination of the file is in the near-line storage, then an on-line cache may be allocated for a temporary file holder with an expiration time, and the Transfer URL is assigned and made available to the client. Once files are transferred into the reserved location in the underlying storage, and upon performing *PutDone* of a file, the file request is completed. If the transferred location is on a temporary on-line space, then the file may be copied from the on-line cache to the target destination, and assigned a lifetime, and the on-line cache may be used to bring additional files.

Files in the request
PrepareToPut function receives an unordered set of File Descriptions describing the files that client is interested in transferring into the storage system managed by SRM. Each File Description consists of parameters described in the "parameters description" section.

srmPrepareToPut execution
PrepareToPut function execution leads to the creation of the request on the server. The request consists of the collection of the file requests, one for each File Description in the request. The state of each file requests persists on the server at least for the duration of the lifetime of the request. The client should be given the handle (or collection of handles), specific to a messaging mechanism, allowing the client to refer to the request as a whole or file requests individually. The *PrepareToPut* function must be an asynchronous non-blocking call returning the handle of the request back to the client as soon as possible.

Once the Server receives the request to prepare files to put, it can make spaces available for the transfer in one of the specified protocols. Depending on the architecture of the storage system it might be as simple as just an assignment of the Transfer URL or as complex as setting a quota on the on-line or near-line storage, and starting a new server process that will be serving the file in the specified protocol. Once the space or path for a file is made available, the Transfer URL is assigned with an expiration time and made available to the client. When the client completes the file transfer into the Transfer URL and issues *PutDone* function, or the lifetime on the Transfer URL is expired, the execution of the request is complete.

2.18.3. Parameter Description

In	Out
User ID	Request Token
Authorization ID	Request Status {
User Request Description	Status information
Flag for streaming	}
Overwrite Mode	Remaining Total Request Time,
Put File Requests {	SURL Statuses {
Target SURL,	SURL,
Expected File Size	Status information,
}	File Size,
Desired File Lifetime,	Estimated Wait Time,

Desired Pin Lifetime, Target File Storage Type, Target Storage System Info, Target Space Token, Target File Retention Policy Info, Desired Total Request Time, Transfer Information	Remaining Pin Lifetime, Remaining File Lifetime, Transfer URL, Transfer Information }

Input parameters

- *User ID* (required)
  User authentication identifier.
- *Authorization ID*
  User authorization information. The *Authorization ID* may be NULL.
- *User Request Description*
  Description of the request. It may be used for later retrieval of the request token.
- *Flag for 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.
- *Overwrite Mode*
  SRM prepares the space for a requested file according to the *Overwrite Mode*.
- *Put File Requests*
  Input parameter *Put File Requests* 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.
  - *Target SURL*
    SURL that client targets to store a file. If the optional *Target SURL* is not provided, then the reference SURL is generated by the SRM. Specific SRM implementation may require *Target SURL* as an input parameter.
  - *Expected File Size*
    Desired file size to be allocated for a file if known. SRM may assign a default space allocation for each file request.
- *Desired File Lifetime*
  Desired life time of the SURL in seconds once when the file is reached the destination. SRM may assign a default lifetime.
- *Desired Pin Lifetime*
  Desired life time of the TURL in seconds once when the TURL is ready for the client to transfer a file into. SRM may assign a default lifetime.
- *Target File Storage Type*
  Target File Storage Type indicates which type of space allocation needs to be performed by SRM server.
- *Target Storage System Info*
  String containining information specific to the underlying storage system that is associated with the *Target SURL* or *Target File Storage Type*. The *Target Storage System Info* may be NULL.
- *Target Space Token*

An advanced option when Space Management feature is supported. SRM will use the specific space when allocating space for files in the request.

- *Target File Retention Policy Info*
  An advanced option when Space Management feature is supported. SRM will use the specific retention policy information when allocating space for files in the request.
- *Desired Total Request Time*
  Total Request Time is that All the file transfer for this request must be complete within this *totalRequestTime*. Otherwise, SRM_REQUEST_TIMED_OUT must be returned as the request status code with individual file status of SRM_FAILURE with an appropriate explanation.
- *Transfer Information*
  It includes the list of the transfer protocols that client can support.

Output parameters

- *Request Token* (required)
  Output parameter request handle is associated with the request for the later asynchronous status request.
- *Request Status* (required)
  Output reporting the success or failure of the request. Textual description of explanation for what happened to the request, specially in case of any failures, would help client diagnose the case. *Status information* is required and it is recommended to return as an enumerated codes separated as status code and error code, so that when client is a non-human program, response to the status information can be pre-programmed.
- *Remaining Total Request Time*
  Output parameter indicates the remaining total request time.
- *SURL Statuses*
  Output reporting the success or failure of the each SURL requests. *SURL Statuses* may be empty and NULL. If returned to the client, *SURL* and its *status information* are required to return.
  - *SURL* (required)
    SURL that client has requested to change the file storage type.
  - *Remaining Pin Lifetime*
    It indicates the remaining pin lifetime on the TURL when the space allocation is ready for the client to put a file into.
  - *Remaining File Lifetime*
    It indicates the remaining file lifetime on the SURL when the file is in its destination target. File lifetime is assigned after srmPutDone is called on the file.
  - *Transfer Information*
    It gives clients more information about the prepared transfer protocol so that client may use the information to make an efficient access to the prepared TURL through the transfer protocol.

2.18.4. NOTES on the Core Behavior

a) *Target SURL*s must be limited to the local files.

b) TURL returned by the *PrepareToPut* may not be used for read access with any protocol. An explicit *PrepareToGet* or *BringOnline* is required.

c) *Access Pattern* may conflict with the type of the target space associated with target space token, when both provided. In this case, *Access Pattern* in the input parameter *Transfer Information* must be ignored.

d) Input parameter *Target Space Token* is provided at the request-level, and all files in the request will end up in the space that is associated with the target space token if the space is large enough for all files.

e) Input parameter *Target File Retention Policy Info* is to specify the desired retention policy information on the file when the file is written into the target storage system.

f) If both input parameters *Target Space Token* and *Target File Retention Policy Info* are provided, then their types must match exactly. Otherwise, the request may be rejected and SRM_INVALID_REQUEST must be returned.

g) Only push mode is supported for file transfers that client must "push" the file to the prepared TURL.

h) Input parameter *Target SURL* has to be local to SRM. If *Target SURL* is not specified, SRM will make a reference for the file request automatically and put it in the specified user space if provided. This reference SURL will be returned along with the "Transfer URL". Some SRM implementation may require *Target SURL*.

i) *PutDone* is expected after each file is "put" into the prepared TURL.

j) 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.

k) Input parameter *Desired Pin Lifetime* is the lifetime (expiration time) on the TURL when the Transfer URL is prepared. It does not refer to the lifetime of the SURL.

l) Input parameter *Desired File Lifetime* is the lifetime of the SURL when the file is put into the storage system. It does not refer to the lifetime (expiration time) of the TURL.

m) The lifetime of the SURL starts as soon as SRM receives the *PutDone*(). If *PutDone*() is not provided, then the files in that space are subject to removal when the lifetime on the TURL expires or the lifetime on the space expires. The lifetime on the TURL can be found in the status of the file request as output parameter *Remaining Pin Time* in *SURL statuses*.

n) If request is accepted, SRM assigns the *Request Token* for asynchronous status checking. In such case, the returned status code should be SRM_REQUEST_QUEUED.

o) Input parameter *Total Request Time* means: All the file transfer for this request must be complete within this *Total Request Time*. Otherwise, SRM_REQUEST_TIMED_OUT must be returned as the request status code with individual file status of SRM_FAILURE with an appropriate explanation.

p) If *Total Request Time* is unspecified as NULL, the request will be retried for a duration which is dependent on the SRM implementation. If *Total Request Time* is 0 (zero), each file request will be tried at least once.

2.18.5. NOTES on the Advanced Behavior with Space Management Feature

a) 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 *File Type* is "volatile".

2.18.6. Status information returned upon request execution

On successful request, the *status information* returns SRM_SUCCESS.
On failure, the *status information* returns SRM_FAILURE and the detailed failure information along with the textual explanation in a human readable form would help the client diagnose the failed request.
When detailed failure can be set to one of the followings:

For request level status,

SRM_REQUEST_QUEUED
- successful request submission and acceptance. Request token must be returned.

SRM_REQUEST_INPROGRESS
- some files are completed, and some files are still on the queue. Request token must be returned.

SRM_SUCCESS
- All requests are successfully completed. For all *SURLs*, spaces are allocated, and *TURLs* are prepared.

SRM_PARTIAL_SUCCESS
- All requests are completed. For some file requests, the spaces are allocated and *TURLs* are prepared, but for some file requests, it is failed. Details are on the files status.

SRM_AUTHENTICATION_FAILURE
- SRM server failed to authenticate the client

SRM_AUTHORIZATION_FAILURE
- client is not authorized to submit the request

SRM_INVALID_REQUEST
- If both input parameters *Target Space Token* and *Target File Retention Policy Info* are provided, then their types must match exactly.
- *Target Space Token* does not refer to an existing known space in the SRM server.

SRM_SPACE_LIFETIME_EXPIRED
- space associated with the *Target Space Token* is expired.

SRM_EXCEED_ALLOCATION
- space associated with the *Target Space Token* is not enough to hold all requested SURLs.

SRM_NO_USER_SPACE
- user space is not enough to hold all requested *SURLs*.

SRM_NO_FREE_SPACE
- SRM space is not enough to hold all requested *SURLs* for free.

SRM_INTERNAL_ERROR
- SRM has an internal transient error, and client may try again.

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. Client requested *Desired File Storage Type* that is not supported by the SRM server.

- *Target File Retention Policy Info* does not refer to a supported retention policy in the SRM server.
- None of the file transfer protocols are supported in the SRM server.

SRM_FAILURE
- All files requests are failed.
- any other request failure. *Explanation* needs to be filled for details.

For file level return status,

SRM_SPACE_AVAILABLE
- successful request completion for the *"put"* request. The space is allocated, and *TURL* is prepared.

SRM_REQUEST_QUEUED
- file request is still on the queue.

SRM_REQUEST_INPROGRESS
- file request is being served.

SRM_SUCCESS
- Client's file transfer into *TURL* is completed, and *srmPutDone* on the *Target SURL* is completed. The file is now in the cache and lifetime on the *Target SURL* is started.

SRM_FILE_IN_CACHE
- lifetime on *SURL* has expired, but the file is still in the cache.

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

SRM_DUPLICATION_ERROR
- *Target SURL* refers to an existing SURL without no overwriting option.

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

SRM_ABORTED
- The requested file has been aborted.

SRM_RELEASED
- The requested file has been released.

SRM_FAILURE
- any other request failure. *Explanation* needs to be filled for details.
- The file request would not be able to be completed within the *Total Request Time*.
- The requested file has been suspended because the request has timed out.

2.18.7. SEE ALSO
PrepareToGet, PrepareToPutRequestStatus, RemoteCopy

2.19. PrepareToPutRequestStatus

2.19.1. NAME
PrepareToPutRequestStatus - is the status call for *PrepareToPut*.

2.19.2. Functional Description

This function is used to check the status of the previously requested *PrepareToPut*. Request token from *PrepareToPut* must be provided.

2.19.3. Parameter Description

In	Out

User ID
Authorization ID
Request Token
Target SURLs
	Request Statuses {
	Status information
	}
	Remaining Total Request Time,
	SURL Statuses {
	SURL,
	Status information,
	File Size,
	Estimated Wait Time,
	Remaining Pin Lifetime,
	Remaining File Lifetime,
	Transfer URL,
	Transfer Information
	}
	Transfer Information

Input parameters

- *User ID* (required)
  User authentication identifier.
- *Authorization ID*
  User authorization information. The *Authorization ID* may be NULL.
- *Request Token* (required)
  A request handle associated with the previously submitted *PrepareToPut* request. The *Request Token* was returned by the function initiating the request.
- *Target SURLs*
  *Target SURL*s to check the status, if client submitted in *PrepareToPut*.

Output parameters

- *Request Status* (required)
  Output reporting the success or failure of the request. Textual description of explanation for what happened to the request, specially in case of any failures, would help client diagnose the case. *Status information* is required and it is recommended to return as an enumerated codes separated as status code and error code, so that when client is a non-human program, response to the status information can be pre-programmed.
- *Remaining Total Request Time*
  Output parameter indicates the remaining total request time.
- *SURL Statuses*

Output reporting the success or failure of the each SURL requests. *SURL Statuses* may be empty and NULL. If returned to the client, *SURL* and its *status information* are required to return.

- *SURL* (required)
  SURL that client has requested to change the file storage type. If the client did not provide SURL at the time of request, SRM server generates a reference SURL SRM server and client can refer to the file later.
- *Remaining Pin Lifetime*
  It indicates the remaining pin lifetime on the TURL when the space allocation is ready for the client to put a file into.
- *Remaining File Lifetime*
  It indicates the remaining file lifetime on the SURL when the file is in its destination target. File lifetime is assigned after *PutDone* is called on the file.
- *Transfer Information*
  It gives clients more information about the prepared transfer protocol so that client may use the information to make an efficient access to the prepared TURL through the transfer protocol.

- *Transfer Information*
  It gives clients more information about the prepared transfer protocol so that client may use the information to make an efficient access to the prepared TURL through the transfer protocol. This is a request level information that all the files have the same information. If any transfer information at the file level, they have the priority that the file specific information applies to the file.

2.19.4. NOTES on the Core Behavior

a) If input parameter *Target SURLs* is empty, it returns status for all files associated with the *Request Token*.

b) The output parameter *SURL* must be the input parameter *Target SURL* of *PrepareToPut* 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, *SURL, Transfer URL, File Size* and *Status information* must be returned.

d) Returned *File Size* is initially the default allocated space size, unless a specific size was requested for the file. After the *srmPutFileDone*() 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 *Remaining File Lifetime* is NULL until the *srmPutFileDone* is issued by the client. After an *srmPutFileDone* is issued, the *Remaining File Lifetime* indicates the remaining lifetime of the file.

f) When the space is ready for client to "put" data and TURL is prepared, the return status code should be SRM_SPACE_AVAILABLE.

g) When the file space is ready for the client, the TURL is available in the cache and pin lifetime on the TURL will be enforced.

h) If a Target SURL is provided with some directory structure, the directory structure must exist, and SRM will not create the directory structure for the Target SURL. In such case, SRM_INVALID_PATH must be returned. srmMkdir may be used to create the directory structure if Directory Management Feature is supported.

i) If the space for the requested files is full, and TURL cannot be returned, then SRM_EXCEED_ALLOCATION, SRM_NO_USER_SPACE, or SRM_NO_FREE_SPACE must be returned for the files.

j) SRM must fail (SRM_FAILURE) only if all files in the request failed.

k) Input parameter *Total Request Time* means: All the file transfer for this request must be complete within this *Total Request Time*. Otherwise, SRM_REQUEST_TIMED_OUT must be returned as the request status code with individual file status of SRM_FAILURE with an appropriate explanation. If *Total Request Time* is unspecified as NULL, the request will be retried for a duration which is dependent on the SRM implementation. If *Total Request Time* is 0 (zero), each file request will be tried at least once.

2.19.5. NOTES on the Advanced Behavior with Space Management Feature

a) The default value of "lifetime" for Volatile or Durable files will be the lifetime left in the space of the corresponding file type. The default value of *File Storage Type* is Volatile.

2.19.6. Status information returned upon request execution

On successful request, the *status information* returns SRM_SUCCESS.
On failure, the *status information* returns SRM_FAILURE and the detailed failure information along with the textual explanation in a human readable form would help the client diagnose the failed request. If only partial files were successful, the *status information* returns SRM_PARTIAL_SUCCESS, and the *SURL Statuses* should explain on those failed files.
When detailed failure can be set to one of the followings:

For request level status,

SRM_SUCCESS
- All requests are successfully completed. For all *SURLs*, spaces are allocated, and *TURLs* are prepared.

SRM_REQUEST_QUEUED
- successful request submission and all files request is still on the queue

SRM_REQUEST_INPROGRESS
- some files are completed, and some files are still on the queue

SRM_PARTIAL_SUCCESS
- All requests are completed. For some file requests, the spaces are allocated and *TURLs* are prepared, but for some file requests, it is failed. Details are on the files status.

SRM_AUTHENTICATION_FAILURE
- SRM server failed to authenticate the client

SRM_AUTHORIZATION_FAILURE
- client is not authorized to submit the request

SRM_INVALID_REQUEST
- *Request Token* does not refer to an existing known request in the SRM server.
- *Target Space Token* that client provided does not refer to an existing space in the SRM server.

SRM_SPACE_LIFETIME_EXPIRED
- space associated with the *Target Space Token* is expired.

SRM_EXCEED_ALLOCATION

- space associated with the *Target Space Token* is not enough to hold all requested *SURLs*.

SRM_NO_USER_SPACE
- user space is not enough to hold all requested *SURLs*.

SRM_NO_FREE_SPACE
- SRM space is not enough to hold all requested *SURLs* for free.

SRM_REQUEST_TIMED_OUT
- Total request time is over and the rest of the request is failed.

SRM_INTERNAL_ERROR
- SRM has an internal transient error, and client may try again.

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. Client requested *Desired File Storage Type* that is not supported by the SRM server.
- *Target File Retention Policy Info* does not refer to a supported retention policy in the SRM server.
- None of the file transfer protocols are supported in the SRM server.

SRM_FAILURE
- All files requests are failed.
- any other request failure. *Explanation* needs to be filled for details.

For file level status,

SRM_SPACE_AVAILABLE
- successful request completion for the *"put"* request. The space is allocated, and *TURL* is prepared.

SRM_REQUEST_QUEUED
- file request is still on the queue.

SRM_REQUEST_INPROGRESS
- file request is being served.

SRM_SUCCESS
- Client's file transfer into *TURL* is completed, and *srmPutDone* on the *Target SURL* is completed. The file is now in the cache and lifetime on the *Target SURL* is started.

SRM_FILE_IN_CACHE
- lifetime on *SURL* has expired, but the file is still in the cache.

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

SRM_DUPLICATION_ERROR
- *Target SURL* refers to an existing SURL without no overwriting option.

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

SRM_ABORTED
- The requested file has been aborted.

SRM_RELEASED
- The requested file has been released.

SRM_FAILURE

- any other request failure. *Explanation* needs to be filled for details.
- The file request would not be able to be completed within the *Total Request Time*.
- The requested file has been suspended because the request has timed out.

2.19.7. SEE ALSO

PrepareToGet, PrepareToPut, RemoteCopy, Ls, LsDetails, ReleaseFiles, Rm, GetRequestTokens

2.20. PutFileDone

2.20.1. NAME

PutFileDone - 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 *PrepareToPut*.

2.20.2. Functional Description

PutDone is used to notify the SRM that the client completed a file transfer to the Transfer URL in the allocated space. This call should normally follow *PrepareToPut*.

2.20.3. Parameter Description

In	Out
User ID	Request Status {
Authorization ID	Status information
Request Token	}
SURLs	SURL Statuses {
	SURL,
	Status information
	}

Input parameters

- *User ID* (required)
  User authentication identifier.
- *Authorization ID*
  User authorization information. The *Authorization ID* may be NULL.
- *Request Token* (required)
  A token associated with the previously submitted request for *srmPrepareToPut*. The *Request Token* was returned by the function initiating the request.
- *SURLs* (required)
  SURLs to indicate the completed file transfer initiated by the client.

Output parameters

- *Request Status* (required)

Output reporting the success or failure of the request. Textual description of explanation for what happened to the request, specially in case of any failures, would help client diagnose the case. *Status information* is required and it is recommended to return as an enumerated codes separated as status code and error code, so that when client is a non-human program, response to the status information can be pre-programmed.

- SURL Statuses

  Output parameter reporting the status of the file in the request. *SURL Statuses* may be empty and NULL. If returned to the client, *SURL* and its *status information* are required to return.

  - *SURL* (required)

    SURL that client has requested to notify the completion of a file transfer.

2.20.4. NOTES on the Core Behavior

a) Called by client after *PrepareToPut* prepares the TURL and the client completes the file transfer into the prepared TURL.

2.20.5. Status information returned upon request execution

On successful request, the *status information* returns SRM_SUCCESS.
On failure, the *status information* returns SRM_FAILURE and the detailed failure information along with the textual explanation in a human readable form would help the client diagnose the failed request. If only partial files were successful, the *status information* returns SRM_PARTIAL_SUCCESS, and the *SURL Statuses* should explain on those failed files.
When detailed failure can be set to one of the followings:

For request level status,

SRM_SUCCESS
- All requests are successfully completed. *TURLs* contain data, and file lifetimes on the *SURLs* start.

SRM_PARTIAL_SUCCESS
- All requests are completed. Some file requests are successfully completed, and some file requests are failed. Details are on the files status.

SRM_AUTHENTICATION_FAILURE
- SRM server failed to authenticate the client

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

SRM_INVALID_REQUEST
- *SURLs* is empty.
- *Request Token* is empty.
- *Request Token* does not refer to an existing known request in the SRM server.

SRM_REQUEST_TIMED_OUT
- Total request time is over and the request is failed.

SRM_ABORTED
- The request has been aborted.

SRM_INTERNAL_ERROR
- SRM has an internal transient error, and client may try again.

SRM_FAILURE

- All files requests are failed.
- any other request failure. *Explanation* needs to be filled for details.

SRM_NOT_SUPPORTED
- *function* is not supported in the SRM

For file level status,

SRM_SUCCESS
- successful request completion of the "put done" for the *Target SURL*

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

SRM_AUTHORIZATION_FAILURE
- client is not authorized to call the request *PutDone* on the *SURL*

SRM_FILE_LIFETIME_EXPIRED
- *Target SURL* has an expired *TURL*.

SRM_SPACE_LIFETIME_EXPIRED
- *Target SURL* has an expired space allocation.

SRM_ABORTED
- The requested SURL file has been aborted.

SRM_FAILURE
- any other request failure. *Explanation* needs to be filled for details.

2.20.6. SEE ALSO
PrepareToPut, PutRequestDone, prepareToPutRequestStatus

2.21. PutRequestDone

2.21.1. NAME
PutRequestDone - 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 *PutFileDone* is issued for all files, this call is not needed.

2.21.2. Functional Description
PutRequestDone is used to notify the SRM that the client completed all file transfers in the request. All file transfers associated with the *Request Token* are considered to be completed. If *PutFileDone* is issued for all files, this call is not needed. See also 2.16.3 for *PutFileDone*.

2.21.3. Parameter Description

In	Out

User ID
Authorization ID
Request Token
	Request Status {
	Status information
	}
	SURL Statuses {
	SURL,
	Status information
	}

Input parameters

- *User ID* (required)
  User authentication identifier.
- *Authorization ID*
  User authorization information. The *Authorization ID* may be NULL.
- *Request Token* (required)
  A token associated with the previously submitted request for *PrepareToPut*. The *Request Token* was returned by the function initiating the request.

Output parameters

- *Request Status* (required)
  Output reporting the success or failure of the request. Textual description of explanation for what happened to the request, specially in case of any failures, would help client diagnose the case. *Status information* is required and it is recommended to return as an enumerated codes separated as status code and error code, so that when client is a non-human program, response to the status information can be pre-programmed.
- SURL Statuses
  Output parameter reporting the status of the file in the request. *SURL Statuses* may be empty and NULL. If returned to the client, *SURL* and its *status information* are required to return.
  - *SURL* (required)
    SURL that client has requested to notify the completion of a file transfer.

2.21.4. NOTES on the Core Behavior

a) Called by client for all files in the request after *PrepareToPut* prepares the TURL and the client completes the file transfer into the prepared TURL.

2.21.5. Status information returned upon request execution

On successful request, the *status information* returns SRM_SUCCESS.
On failure, the *status information* returns SRM_FAILURE and the detailed failure information along with the textual explanation in a human readable form would help the client diagnose the failed request. If only partial files were successful, the *status information* returns SRM_PARTIAL_SUCCESS, and the *SURL Statuses* should explain on those failed files.
When detailed failure can be set to one of the followings:

For request level status,

SRM_SUCCESS
- All requests are successfully completed. *TURLs* contain data, and file lifetimes on the *SURLs* start.

SRM_PARTIAL_SUCCESS
- All requests are completed. Some file requests are successfully completed, and some file requests are failed. Details are on the files status.

SRM_AUTHENTICATION_FAILURE
- SRM server failed to authenticate the client

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

SRM_INVALID_REQUEST
- *Request Token* is empty.
- *Request Token* does not refer to an existing known request in the SRM server.

SRM_REQUEST_TIMED_OUT
- Total request time is over and the request is failed.

SRM_ABORTED
- The request has been aborted.

SRM_INTERNAL_ERROR
- SRM has an internal transient error, and client may try again.

SRM_FAILURE
- All files requests are failed.
- any other request failure. *Explanation* needs to be filled for details.

SRM_NOT_SUPPORTED
- *function* is not supported in the SRM

For file level status,

SRM_SUCCESS
- successful request completion of the "put done" for the *Target SURL*

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

SRM_AUTHORIZATION_FAILURE
- client is not authorized to call the request *PutDone* on the *SURL*

SRM_FILE_LIFETIME_EXPIRED
- *Target SURL* has an expired *TURL*.

SRM_SPACE_LIFETIME_EXPIRED
- *Target SURL* has an expired space allocation.

SRM_ABORTED
- The requested SURL file has been aborted.

SRM_FAILURE
- any other request failure. *Explanation* needs to be filled for details.

2.21.6. SEE ALSO
PrepareToPut, PutFileDone, PrepareToPutRequestStatus

2.22. ReleaseFiles

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

2.22.2. Functional Description
This function is used to release pins on the previously requested "copies" (or "state") of the SURL. This function normally follows *PrepareToGet* or *BringOnline* functions.

2.22.3. Parameter Description

In	Out
User ID	Request Status {
Authorization ID	Status information
Request Token	}
SURLs	SURL Statuses {
Flag for Releasing All Currently Pinned Files	SURL,
Flag for removal	Status information
	}

Input parameters

- *User ID* (required)
  User authentication identifier.
- *Authorization ID*
  User authorization information. The *Authorization ID* may be NULL.
- *Request Token*
  A handle associated with the previously submitted request. The *Request Token* was returned by the function initiating the request (e.g. *PrepareToGet*()).
- *SURLs* (required)
  SURLs to be released.
- *Flag for releasing All Currently Pinned Files*
  Flag for releasing All Currently Pinned Files indicates if all currently pinned files in the request associated with the *Request Token* are to be released.
- *Flag for removal*
  Flag for removal indicates if the requested files are to be released and the Transfer URLs are to be removed.

Output parameters

- *Request Status* (required)
  Output reporting the success or failure of the request. Textual description of explanation for what happened to the request, specially in case of any failures, would help client diagnose the case. *Status information* is required and it is recommended to return as an enumerated codes separated as status code and error code, so that when client is a non-human program, response to the status information can be pre-programmed.
- *SURL Statuses*
  Output parameter reporting the status of the file in the request. *SURL Statuses* may be empty and NULL. If returned to the client, *SURL* and its *status information* are required to return.
  - *SURL* (required)
    SURL that is released.

2.22.4. NOTES on the Core Behavior

a) *Flag for releasing All Currently Pinned Files* is valid only within the *Request Token*.

b) If none of *SURL*s are provided and *Flag for Releasing All Currently Pinned Files* is true, then all currently pinned files in the client's space are released.
c) If *Flag for Releasing All Currently Pinned Files* is true, then none of *SURL* must be provided, or all of *SURL*s in the *Request Token* must be provided. Otherwise SRM_INVALID_REQUEST will be returned.
d) If *Request Token* is not provided, then the SRM will release all the files specified by the SURLs owned by the client.
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) *Flag for Removal* flag is false by default. If *Flag for Removal* flag is true, the pin on the TURL is released, the "copy" or "state" is removed, and SRM may release the resource.

2.22.5. 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.

2.22.6. Status information returned upon request execution

On successful request, the *status information* returns SRM_SUCCESS.
On failure, the *status information* returns SRM_FAILURE and the detailed failure information along with the textual explanation in a human readable form would help the client diagnose the failed request. If only partial files were successful, the *status information* returns SRM_PARTIAL_SUCCESS, and the *SURL Statuses* should explain on those failed files.
When detailed failure can be set to one of the followings:

For request level status,

SRM_SUCCESS
- All requests are successfully completed. All *SURLs* are released successfully.

SRM_PARTIAL_SUCCESS
- All requests are completed. Some *SURLs* are successfully released, and some *SURLs* are failed. Details are on the files status.

SRM_AUTHENTICATION_FAILURE
- SRM server failed to authenticate the client

SRM_AUTHORIZATION_FAILURE
- client is not authorized to release files

SRM_INVALID_REQUEST
- *SURLs* is empty.
- *Request Token* does not refer to an existing known request of *PrepareToGet* or *BringOnline* in the SRM server.

SRM_INTERNAL_ERROR
- SRM has an internal transient error, and client may try again.

SRM_FAILURE
- All files requests are failed.
- any other request failure. *Explanation* needs to be filled for details.

SRM_NOT_SUPPORTED
- *function* is not supported in the SRM
- input parameter *Flag for removal* is not supported in the SRM. *srmRm* must be used.

For file level status,

SRM_SUCCESS
- successful request completion for the *SURL*. *SURL* is released successfully.

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

SRM_AUTHORIZATION_FAILURE
- client is not authorized to release *SURL*

SRM_LAST_COPY
- *SURL* is the last copy when *Flag for removal* flag is on

SRM_FILE_LIFETIME_EXPIRED
- *SURL* is expired already.

SRM_ABORTED
- The requested file has been aborted.

SRM_FAILURE
- any other request failure. *Explanation* needs to be filled for details.

2.22.7. SEE ALSO
PrepareToGet, PrepareToPut, RemoteCopy, PutFileDone, PrepareToGetRequestStatus, PrepareToPutRequestStatus, ReleaseSpace

2.23. Rm

2.23.1. NAME
Rm - removes local files regardless of the requests that got the files into the space.

2.23.2. Functional Description
This function will remove SURLs (the name space entries) in the storage system. Difference from *PurgeFromSpace* is that *PurgeFromSpace* removes only previously requested "copies" (or "state") of the SURL in a particular space, and *PurgeFromSpace* shall not remove SURLs or the name space entries.

2.23.3. Parameter Description

In	Out
User ID	Request Status {
Authorization ID	Status information
Storage System Info	}
SURLs	SURL Statuses {
	SURL,
	Status information
	}

Input parameters

- *User ID* (required)

User authentication identifier.

- *Authorization ID*
  User authorization information. The *Authorization ID* may be NULL.
- *Storage System Info*
  String containining information specific to the underlying storage system. The *storage System Info* may be NULL.
- *SURLs* (required)
  SURLs to be removed.

Output parameters

- *Request Token*
  Output parameter request handle is associated with the request for the later asynchronous status request. *Request Token* may be NULL, in case *srmRm* is processed without delay.
- *Request Status* (required)
  Output reporting the success or failure of the request. Textual description of explanation for what happened to the request, specially in case of any failures, would help client diagnose the case. *Status information* is required and it is recommended to return as an enumerated codes separated as status code and error code, so that when client is a non-human program, response to the status information can be pre-programmed.
- *SURL Statuses*
  Output reporting the success or failure of the each SURL requests. *SURL Statuses* may be empty and NULL. If returned to the client, *SURL* and its *status information* are required to return.
  - *SURL* (required)
    SURL that is removed.

2.23.4. NOTES on the Core Behavior

a) *Rm* removes all copies or states on the storage, and removes the entry from the name space.
b) When an SURL is removed, all associated pinned TURLs are all released and removed as well.
c) *Ls, PrepareToGet* or *BringOnline* will not find these removed files any more. It must set file requests on SURL from *PrepareToGet* as SRM_ABORTED.
d) *Rm* aborts the SURLs from *PrepareToPut* requests not yet in SRM_PUT_DONE state, and must set its file status as SRM_ABORTED.

2.23.5. NOTES on the Advanced Behavior with Directory Management Feature

a) To distinguish from *Rmdir*(), this function applies to files only.
b) To remove empty directories, *Rmdir*() must to be used.

2.23.6. NOTES on the Advanced Behavior with Authorization Feature

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

2.23.7. Status information returned upon request execution

On successful request, the *status information* returns SRM_SUCCESS.

On failure, the *status information* returns SRM_FAILURE and the detailed failure information along with the textual explanation in a human readable form would help the client diagnose the failed request. If only partial files were successful, the *status information* returns SRM_PARTIAL_SUCCESS, and the *SURL Statuses* should explain on those failed files.
When detailed failure can be set to one of the followings:

For request level status,

SRM_SUCCESS
- All requests are successfully completed. All *SURLs* are removed.

SRM_PARTIAL_SUCCESS
- All requests are completed. Some *SURLs* are successfully removed, and some *SURLs* are failed to be removed. Details are on the files status.

SRM_AUTHENTICATION_FAILURE
- SRM server failed to authenticate the client

SRM_AUTHORIZATION_FAILURE
- client is not authorized to remove any files

SRM_INVALID_REQUEST
- *SURLs* are empty.

SRM_INTERNAL_ERROR
- SRM has an internal transient error, and client may try again.

SRM_NOT_SUPPORTED
- *function* is not supported in the SRM

SRM_FAILURE
- All files requests are failed.
- any other request failure. *Explanation* needs to be filled for details.

For file level status,

SRM_SUCCESS
- successful request completion for the *SURL*. *SURL* is removed.

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

SRM_AUTHORIZATION_FAILURE
- client is not authorized to remove *SURL*

SRM_FILE_LOST
- the request file is permanently lost.

SRM_FILE_UNAVAILABLE
- the request file is temporarily unavailable.

SRM_FAILURE
- any other request failure. *Explanation* needs to be filled for details.

2.23.8. SEE ALSO

Ls, LsDetails

3. Advanced feature set 1 : Remote Access Functions

summary:

RemoteCopy
RemoteCopyRequestStatus

details:

3.1. RemoteCopy

3.1.1. NAME

RemoteCopy - replicates files from one site to another.

3.1.2. Functional Description

This function is used to copy files from source storage sites into the target storage sites. The source storage site or the target storage site needs to be the SRM itself that the client makes the *RemoteCopy* request. If both source and target are local to the SRM, it performed a local copy. There are two cases for remote copies: 1. Target SRM is where client makes a *RemoteCopy* request (PULL case), 2. Source SRM is where client makes a *RemoteCopy* request (PUSH case).

1. PULL case: Upon the client's *RemoteCopy* request, the target SRM makes a space at the target storage, and makes a request *PrepareToGet* to the source SRM. When TURL is ready at the source SRM, the target SRM transfers the file from the source TURL into the prepared target storage. After the file transfer completes, *ReleaseFiles* is issued to the source SRM.
2. PUSH case: Upon the client's *RemoteCopy* request, the source SRM prepares a file to be transferred out to the target SRM, and makes a request *PrepareToPut* to the target SRM. When TURL is ready at the target SRM, the source SRM transfers the file from the prepared source into the prepared Target TURL. After the file transfer completes, *PutDone* is issued to the target SRM.

When specified target space token is provided, the files will be located finally in the targeted space associated with the space token. It is an asynchronous operation, and request token must be returned. The status may be checked through *RemoteCopyRequestStatus* with the returned request token.

3.1.3. Parameter Description

In	Out
User ID	Request Token

Authorization ID
Copy File Requests {
Source SURL,
Target SURL,
Directory Option
}
User Request Description
Source Storage System Info
Target Storage System Info
Desired Total Request Time
Desired Target SURL Lifetime
Target File StorageType
Target Space Token
Target File Retention Polify Info
Flag for streaming
Overwrite Mode
Flag for Removing Source Files
Flag for Checksum
	Request Status {
	Status information
	}
	Remaining Total Request Time
	SURL Statuses {
	Source SURL,
	Target SURL,
	Status information,
	File Size,
	Estimated Wait Time,
	Remaining File Lifetime
	}

Input parameters

- *User ID* (required)
  User authentication identifier.
- *Authorization ID*
  User authorization information. The *Authorization ID* may be NULL.
- *Source Storage System Info*
  Information specific to the source storage system that is associated with the *Source SURL*s. The *Source Storage System Info* may be NULL.
- *Target Storage System Info*
  Information specific to the target storage system that is associated with the *Target SURL*s or *Target File Storage Type*. The *Target Storage System Info* may be NULL.
- *User Request Description*
  Description of the request. It may be used for later retrieval of the rquest token.
- *Desired Total Request Time*
  Desired Total Request Time means: if all the file transfer for this request must be complete in this *Desired Total Request Time*. Otherwise, the request is returned as failed at the end of the *Desired Total Request Time*, and SRM_REQUEST_TIMED_OUT must be returned as the request status code with individual file status of SRM_FAILURE with an appropriate explanation. All completed files must not be removed, but status of the files must be returned to the client.
- *Desired Target SURL Lifetime*
  Desired life time of the Target SURL in seconds once when it's in the target SRM. SRM may assign a default lifetime, if not provided.
- *Target File Storage Type*

Target File Storage Type indicates which type of storage that *Target SURL*s are copied into in target SRM.

- *Target Space Token* (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 *Space Token* is acquired separately (e.g. *ReserveSpace*).
- *Flag for 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.
- *Overwrite Mode*
  SRM needs to replicate the file according to the *Overwrite Mode* on the target that SRM brings the files into.
- *Flag for Removing Source Files*
  Boolean indication of file removal at the source (*Source SURL*) after the copy is performed.
- *Flag for Checksum*
  Flag for Checksum indicates that checksum calculation for all files in the request needs to be performed when files get into its designated target space.
- *Copy File Requests* (required)
  Input parameter *Copy File Requests* consists of SURL information that client wants to copy from one site to another.
  - *Source SURL* (required)
    Source SURL
  - *Target SURL*
    Target SURL
  - *Directory Option* (advanced option)
    An advanced option when Directory Management feature is supported. It includes *Flag for Source Directory, Flag for All Level Recursive* and *Number Of Recursive Levels*: *Flag for Source Directory* is a boolean indicator if *Source SURL* is a directory. *Flag for All Level Recursive* is a boolean indicator if *Source SURL* is a directory and all files under the SURL must be retrieved. *Number Of Recursive Levels* is a positive integer indicator for how many levels of the recursive directory must be browsed and all files in those directories to be retrieved.

Output parameters

- *Request Token* (required)
  Output parameter string token is associated with the request for the later asynchronous status request.
- *Request Status* (required)
  Output reporting the success or failure of the request. Textual description of explanation for what happened to the request, specially in case of any failures, would help client diagnose the case. *Status information* is required and it is recommended to return as an enumerated codes separated as status code and error code, so that when client is a non-human program, response to the status information can be pre-programmed.

- *Remaining Total Request Time*
  Output parameter indicates the remaining total request time.
- *SURL Statuses*
  Output reporting the success or failure of the each SURL requests. *SURL Statuses* may be empty and NULL. If returned to the client, *SURL* and its *Status information* are required to return.
  - *Source SURL* (required)
    SURL that client has requested to copy the file from.
  - *Target SURL* (required)
    SURL that client has requested to copy the file to. If the client did not provide SURL at the time of request, SRM server generates a reference SURL that SRM server and client can refer to the file.
  - *Remaining File Lifetime*
    It indicates the remaining file lifetime on the SURL when the file is copied into its destination target. File lifetime is assigned after file copy is completed.

3.1.4. NOTES on the Behavior

a) When aborted, *Target SURLs* need to be provided.

b) Input parameter *Target Space Token* is provided at the request-level, and all files in the request will end up in the space that is associated with the target space token.

c) Input parameter *Target File Retention Policy Info* is to specify the desired retention policy information on the file when the file is written into the target storage system.

d) If both input parameters *Target Space Token* and *Target File Retention Policy Info* are provided, then their types must match exactly. Otherwise, the request may be rejected, and SRM_INVALID_REQUEST must be returned.

e) If request is accepted, SRM assigns the *Request Token* for asynchronous status checking. In such case, the returned status code should be SRM_REQUEST_QUEUED.

f) *RemoteCopy* 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.

g) Always release files through *ReleaseFiles* from the source after copy is done, if source is an SRM and PULL mode was performed.

h) Always issue *PutDone* to the target after copy is done, if target is an SRM and PUSH mode was performed.

i) Note there is no transfer protocol negotiation with the client for this request.

j) Input parameter *Desired Total Request Time* means: if all the file transfer for this request must be complete in this *Desired Total Request Time*. Otherwise, the request is returned as failed at the end of the *Desired Total Request Time*, and SRM_REQUEST_TIMED_OUT must be returned as the request status code with individual file status of SRM_FAILURE with an appropriate explanation. All completed files must not be removed, but status of the files must be returned to the client. If *Total Request Time* is unspecified as NULL, the request will be retried for a duration which is dependent on the SRM implementation. If *Total Request Time* is 0 (zero), each file request will be tried at least once.

k) 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.

l) When *Flag for Removing Source Files* is true, SRM removes the source files on behalf of the client after the copy is done.

m) 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 must fail.

n) When the streaming option is true, and the target space is full, the copy operation is suspended till more space is made available. The duration of suspension depends on the local SRM policy. When the streaming option is false, and if there is not enough space to accommodate the entire request, the SRM returns failure.

o) *RemoteCopy* performs a copy from or to remote sites only. Thus, when both *Source SURL* and *Target SURL* are local, an error SRM_INVALID_REQUEST is returned. A copy of a local file to another must be done by the *Cp* function if the Directory Management Feature is supported.

3.1.5. 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 *Space Token*.

3.1.6. NOTES on the Advanced Behavior with Directory Management Feature

a) Empty directories must be copied as well.

b) If a Target SURL is provided with some directory structure, the directory structure must exist, and SRM will not create the directory structure for the Target SURL. In such case, SRM_INVALID_PATH must be returned. srmMkdir may be used to create the directory structure.

c) If the Source SURL and Target SURL are provided as directories (copying directories) when SRM implementation supports, then all sub directories will be copied over from the source to the target, and complete sub-directory structure will be created only if *Direction Option* indicates them.

3.1.7. Status information returned upon request execution

On successful request, the *status information* returns SRM_SUCCESS.
On failure, the *Status information* returns SRM_FAILURE and the detailed failure information along with the textual explanation in a human readable form would help the client diagnose the failed request. If only partial files were successful, the *Status information* returns SRM_PARTIAL_SUCCESS, and the *SURL Statuses* should explain on those failed files.
When detailed failure can be set to one of the followings:

For request level status,

SRM_REQUEST_QUEUED

- successful request submission and acceptance. Request token must be returned.

SRM_REQUEST_INPROGRESS

- Some files are completed, and some files are still on the queue. Details are on the files status. Request token must be returned.

SRM_SUCCESS

- All requests are successfully completed. All *Source SURLs* are copied into the target destination successfully.

SRM_PARTIAL_SUCCESS
- All requests are completed. Some file request is successfully copied into the target destination, and some file request is failed. Details are on the files status.

SRM_AUTHENTICATION_FAILURE
- SRM server failed to authenticate the client

SRM_AUTHORIZATION_FAILURE
- client is not authorized to submit the request
- Client is not authorized to copy files into the space that client provided with *Target Space Token* or *Target File Retention Policy Info*

SRM_INVALID_REQUEST
- If both input parameters *Target Space Token* and *Target File Retention Policy Info* are provided, then their types must match exactly.
- *Target Space Token* does not refer to an existing known space in the SRM server.

SRM_SPACE_LIFETIME_EXPIRED
- space associated with the *Target Space Token* is expired.

SRM_EXCEED_ALLOCATION
- space associated with the *Target Space Token* is not enough to hold all requested SURLs.

SRM_NO_USER_SPACE
- user space is not enough to hold all requested SURLs.

SRM_NO_FREE_SPACE
- SRM space is not enough to hold all requested SURLs for free.

SRM_INTERNAL_ERROR
- SRM has an internal transient error, and client may try again.

SRM_NOT_SUPPORTED
- SRM server does not support the given input parameters. For example, client requested *Desired File Storage Type* that is not supported by the SRM server.
- *Target File Retention Policy Info* does not refer to a supported retention policy in the SRM server.
- Directory operation is not supported in the SRM server.
- Recursive directory operation is not supported in the SRM server.
- any input parameter is not supported in the SRM server
- a particular type of an input parameter is not supported in the SRM server
- *function* is not supported in the SRM server

SRM_FAILURE
- all files requests are failed.
- any other request failure. *Explanation* needs to be filled for details.

For file level return status,

SRM_SUCCESS

- successful request completion for the file. The source *SURL* is copied into the target destination *Target SURL* successfully, and lifetime on the *Target SURL* is started.

SRM_REQUEST_QUEUED
- file request is still on the queue.

SRM_REQUEST_INPROGRESS
- file request is being served.

SRM_FILE_LOST
- the request file (*Source SURL*) is permanently lost.

SRM_FILE_BUSY
- client requests for files at the source (*Source SURL*) which there is an active *PrepareToPut* (no *PutDone* is not yet called) for.

SRM_FILE_UNAVAILABLE
- the request file (*Source SURL*) is temporarily unavailable.

SRM_FILE_LIFETIME_EXPIRED
- lifetime on *Target SURL* has expired, but the file is still in the cache.

SRM_INVALID_PATH
- *Source SURL* does not exist
- *Target SURL* does not refer to a valid path.

SRM_DUPLICATION_ERROR
- *Target SURL* refers to an existing SURL without no overwriting option.

SRM_AUTHORIZATION_FAILURE
- Client is not authorized to copy files from *Source SURL*
- Client is not authorized to copy files into *Target SURL*
- Client is not authorized to copy files into the space that client provided with *Target Space Token* or *Target File Retention Policy Info*

SRM_ABORTED
- The requested file has been aborted.

SRM_RELEASED
- The requested file has been released.

SRM_FAILURE
- any other request failure. *Explanation* needs to be filled for details.
- The file request would not be able to be completed within the *Total Request Time*.
- The requested file has been suspended because the request has timed out.

3.1.8. SEE ALSO
PrepareToGet, PrepareToPut, RemoteCopyRequestStatus, GetRequestTokens

3.2. RemoteCopyRequestStatus

3.2.1. NAME
RemoteCopyRequestStatus - is the status call for *RemoteCopy*.

3.2.2. Functional Description

This function is used to check the status of the previously requested *RemoteCopy*. Request token from *RemoteCopy* must be provided.

3.2.3. Parameter Description

In	Out
User ID	Request Status {
Authorization ID	Status information
Request Token	}
Source SURLs	Remaining Total Request Time
Target SURLs	SURL Statuses {
offset	Source SURL,
count	Target SURL,
	Status information,
	File Size,
	Estimated Wait Time,
	Remaining File Lifetime
	}

Input parameters

- *User ID* (required)
  User authentication identifier.
- *Authorization ID*
  User authorization information. The *Authorization ID* may be NULL.
- *Request Token* (required)
  A handle associated with the previously submitted *RemoteCopy* request. The *Request Token* was returned by the function initiating the request.
- *Source SURLs*
  Selective *Source SURL*s to check the status.
- *Target SURLs*
  Selective *Target SURL*s to check the status.

Output parameters

- *Requestn Status* (required)
  Output reporting the success or failure of the request. Textual description of explanation for what happened to the request, specially in case of any failures, would help client diagnose the case. *Status information* is required and it is recommended to return as an enumerated codes separated as status code and error code, so that when client is a non-human program, response to the status information can be pre-programmed.
- *Remaining Total Request Time*
  Output parameter indicates the remaining total request time.
- *SURL Statuses*
  Output reporting the success or failure of the each SURL requests. *SURL Statuses* may be empty and NULL. If returned to the client, *SURL* and its *status information* are required to return.

- *Source SURL* (required)
  SURL that client has requested to copy the file from.
- *Target SURL* (required)
  SURL that client has requested to copy the file into. If the client did not provide SURL at the time of request, SRM server generates a reference SURL that SRM server and client can refer to the file.
- *Remaining File Lifetime*
  It indicates the remaining file lifetime on the Target SURL when the file is in its destination. File lifetime is assigned after file copy is completed.

3.2.4. NOTES on the Behavior

a) If any *Source SURLs* or *Target SURLs* are not provided, the status for all the files associated with the *Request Token* is returned.

b) If the target space for the requested files is full, then SRM_EXCEED_ALLOCATION, SRM_NO_USER_SPACE, or SRM_NO_FREE_SPACE must be returned.

c) SRM must fail (SRM_FAILURE) only if all files in the request failed.

d) *Total Request Time* means: All the file transfer for this request must be complete within this *Total Request Time*. Otherwise, SRM_REQUEST_TIMED_OUT must be returned as the request status code with individual file status of SRM_FAILURE with an appropriate explanation. If *Total Request Time* is unspecified as NULL, the request will be retried for a duration which is dependent on the SRM implementation. If *Total Request Time* is 0 (zero), each file request will be tried at least once.

3.2.5. Status information returned upon request execution

On successful request, the *status information* returns SRM_SUCCESS.
On failure, the *status information* returns SRM_FAILURE and the detailed failure information along with the textual explanation in a human readable form would help the client diagnose the failed request. If only partial files were successful, the *status information* returns SRM_PARTIAL_SUCCESS, and the *SURL Statuses* should explain on those failed files.
When detailed failure can be set to one of the followings:

For request level status,

SRM_SUCCESS
- All requests are successfully completed. All *Source SURLs* are copied into the target destination successfully.

SRM_REQUEST_QUEUED
- successful request submission and all files request is still on the queue

SRM_REQUEST_INPROGRESS
- Some files are completed, and some files are still on the queue. Details are on the files status.

SRM_PARTIAL_SUCCESS
- All requests are completed. Some file request is successfully copied into the target destination, and some file request is failed. Details are on the files status.

SRM_AUTHENTICATION_FAILURE
- SRM server failed to authenticate the client

SRM_AUTHORIZATION_FAILURE
- client is not authorized to submit the request

SRM_INVALID_REQUEST
- *Request Token* does not refer to an existing known request in the SRM server.
- *Target Space Token* does not refer to an existing known space in the SRM server.

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

SRM_REQUEST_TIMED_OUT
- Total request time is over and the rest of the request is failed.

SRM_SPACE_LIFETIME_EXPIRED
- space associated with the *Target Space Token* is expired.

SRM_EXCEED_ALLOCATION
- space associated with the *Target Space Token* is not enough to hold all requested SURLs.

SRM_NO_USER_SPACE
- Insufficient space left in the space that is associated with spaceToken.

SRM_NO_FREE_SPACE
- When client does not specify the Space Token, SRM uses a default space. The default space is insufficient to accommodate the request.

SRM_ABORTED
- The request has been aborted.

SRM_INTERNAL_ERROR
- SRM has an internal transient error, and client may try again.

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. Client requested *Desired File Storage Type* that is not supported by the SRM server.
- *Target File Retention Policy Info* does not refer to a supported retention policy in the SRM server.
- Overwrite option is not supported in the SRM server.
- Directory operation is not supported in the SRM server.
- Recursive directory operation is not supported in the SRM server.
- any input parameter is not supported in the SRM server
- a particular type of an input parameter is not supported in the SRM server
- *function* is not supported in the SRM server

SRM_FAILURE
- all files requests are failed.
- any other request failure. *Explanation* needs to be filled for details.

For file level status,

SRM_SUCCESS

- successful request completion for the file. The *Source SURL* is copied into the target destination *Target SURL* successfully, and lifetime on the *Target SURL* is started.

SRM_REQUEST_QUEUED
- file request is still on the queue.

SRM_REQUEST_INPROGRESS
- file request is being served.

SRM_FILE_LOST
- the request file (*Source SURL*) is permanently lost.

SRM_FILE_BUSY
- client requests for files at the source (*Source SURL*) which there is an active srmPrepareToPut (no srmPutDone is not yet called) for.

SRM_FILE_UNAVAILABLE
- the request file (*Source SURL*) is temporarily unavailable.

SRM_FILE_LIFETIME_EXPIRED
- lifetime on *Target SURL* has expired, but the file is still in the cache.

SRM_INVALID_PATH
- *Source SURL* does not exist
- *Target SURL* does not refer to a valid path.

SRM_DUPLICATION_ERROR
- *Target SURL* refers to an existing SURL without no overwriting option.

SRM_AUTHORIZATION_FAILURE
- Client is not authorized to copy files from *Source SURL*
- Client is not authorized to copy files into *Target SURL*
- Client is not authorized to copy files into the space that client provided with *Target Space Token* or *Target File Retention Policy Info*

SRM_ABORTED
- The requested file has been aborted.

SRM_RELEASED
- The requested file has been released.

SRM_FAILURE
- any other request failure. *Explanation* needs to be filled for details.
- The file request would not be able to be completed within the *Total Request Time*.
- The requested file has been suspended because the request has timed out.

3.2.6. SEE ALSO

PrepareToGet, PrepareToPut, RemoteCopy, GetRequestTokens

4. Advanced feature set 2 : Space Management Functions

summary:

ChanceSpaceForFiles
ExtendFileLifeTimeInSpace
GetSpaceMetaData
GetSpaceTokens
PurgeFromSpace
ReleaseSpace
ReserveSpace
StatusOfChanceSpaceForFilesRequest
StatusOfReserveSpaceRequest
StatusOfUpdateSpaceRequest
UpdateSpace

details:

4.1. ChangeSpaceForFiles

4.1.1. NAME

ChangeSpaceForFiles – is used to change the space property of files to another space property.

4.1.2. Functional Description

This function is used to change the space property of files to another space property by specifying target space tokens. All files specified by SURLs will have a new space token. SURLs must not be changed. New space token may be acquired from *ReserveSpace*. Asynchronous operation may be necessary for some SRMs, and in such case, request token is returned for later status inquiry. There is no default behavior when target space token is not provided. In such case, the request will be rejected, and the return status must be SRM_INVALID_REQUEST.

4.1.3. Parameter Description

In	Out
User ID	Request Status {
Authorization ID	Status information
Storage System Info	}
SURLs	Request Token
Target Space Token	Estimated Processing Time
	SURL Statuses {
	SURL,
	Status information
	}

Input parameters

- *User ID* (required)
  User authentication identifier.
- *Authorization ID*
  User authorization information. The *Authorization ID* may be NULL.
- *Storage System Info*
  Information specific to the underlying storage system. The *Storage System Info* may be NULL.
- *SURLs* (required)
  SURLs to move to another space..
- *Target Space Token* (required)
  A space handle to move *SURLs* to. The *Space Token* must be acquired separately (e.g. *ReserveSpace*).

Output parameters

- *Request Status* (required)
  Output reporting the success or failure of the request. Textual description of explanation for what happened to the request, specially in case of any failures, would help client diagnose the case. *Status information* is required and it is recommended to return as an enumerated codes separated as status code and error code, so that when client is a non-human program, response to the status information can be pre-programmed.
- *Request Token*
  Output parameter request handle is associated with the request for the later asynchronous status request. *Request Token* may be NULL, in case *ChangeSpaceForFiles* is processed without delay.
- *Estimated Processing Time*
  Output parameter in seconds described how long the space reservation request may take in estimation, in case *ChangeSpaceForFiles* is processed with delay. It is used to indicate the estimation time to complete the request, when known.
- *SURL Statuses*
  Output reporting the status of each file in the request. *SURL Statuses* may be empty and NULL. If returned to the client, *SURL* and its *status information* are required to return.
  - *SURL* (required)
    SURL that client has requested to clean up.

4.1.4. NOTES on the Behavior

a) When space transition is completed successfully, SRM_SUCCESS must be returned for each SURL.

b) For any forbidden transition by the SRM implementation, SRM_INVALID_REQUEST must be returned.

c) Asynchronous operation may be necessary for some SRMs to serve many concurrent requests. In such case, request token must be returned. If the request can be completed immediately, request token must not be returned.

d) When asynchronous operation is necessary, the returned status information should include SRM_REQUEST_QUEUED, and *SURL Statuses* may not be filled and returned.

e) All files specified in *SURLs* will be moved to the space associated with *Target Space Token.*

f) When the *Target Space Token* is used, space allocation for a new space token must be done explicitly by the client before using this function.
g) Space de-allocation may be necessary in some cases, and it must be done by the client explicitly after this operation completes. The status can be checked by *ChangeSpaceForFilesRequestStatus*.
h) When a space is successfully changed for a file from one space to another, it will either retain its remaining lifetime, or the lifetime will be reduced to that of the target space, whichever is the lesser.
i) If the target space is only large enough to transfer a subset of the files, the request will continue taking place until the target space cannot hold any more files, and the request must be failed. The status information of the request must return an error of SRM_EXCEED_ALLOCATION in such case.

4.1.5. NOTES on the Advanced Behavior with Directory Management Feature

a) If a directory path is provided, then the effect is recursive for all files in the directory.

4.1.6. Status information returned upon request execution

On successful request, the *status information* returns SRM_SUCCESS.
On failure, the *status information* returns SRM_FAILURE and the detailed failure information along with the textual explanation in a human readable form would help the client diagnose the failed request. If only partial files were successful, the *status information* returns SRM_PARTIAL_SUCCESS, and the *SURL Statuses* should explain on those failed files.
When detailed failure can be set to one of the followings:

For request level status,

SRM_SUCCESS
- All file requests are successfully completed. All *SURLs* have new *Target Space Token*.

SRM_PARTIAL_SUCCESS
- All requests are completed. Some SURL requests have new *Target Space Token*, and some SURL requests are failed to have new *Target Space Token*. Details are on the files status.

SRM_ REQUEST_QUEUED
- request is submitted and accepted. *Request Token* must be returned.
- The status can be checked by *ChangeSpaceForFilesRequestStatus*.

SRM_ REQUEST_INPROGRESS
- The request is being processed. Some files are still queued, and some files are completed in space transition.

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.
- *Target Space Token* is empty.
- *Target Space Token* does not refer to an existing space in the SRM server.

- *Target Space Token* refers to a forbidden transition by the SRM implementation.

SRM_SPACE_LIFETIME_EXPIRED
- target space that is associated with *Target Space Token* has an expired lifetime.

SRM_EXCEED_ALLOCATION
- target space that is associated with *Target Space Token* is not enough to hold all *SURLs*.

SRM_INTERNAL_ERROR
- SRM has an internal transient error, and client may try again.

SRM_FAILURE
- any other request failure. *Explanation* needs to be filled for details.

SRM_NOT_SUPPORTED
- *function* is not supported in the SRM
- any input parameter is not supported in the SRM server
- a particular type of an input parameter is not supported in the SRM server

For file level status,

SRM_SUCCESS
- successful request completion for the *SURL*. The *SURL* has a new *Target Space Token.*

SRM_ REQUEST_QUEUED
- file request is on the queue.

SRM_ REQUEST_INPROGRESS
- file request is being processed.

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

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

SRM_INVALID_REQUEST
- *Target Space Token* refers to a forbidden transition for the particular SURL by the SRM implementation.

SRM_EXCEED_ALLOCATION
- target space that is associated with *Target Space Token* is not enough to hold *SURL*.

SRM_FILE_LOST
- the requested file with the SURL is permanently lost.

SRM_FILE_BUSY
- client requests for files which there is an active *srmPrepareToPut* (no *PutDone* is not yet called) for.
- The requested file with the SURL is being used by other clients.

SRM_FILE_UNAVAILABLE
- the requested file with the SURL is temporarily unavailable.

SRM_FAILURE
- All file requests are failed.

- any other request failure. *Explanation* needs to be filled for details.

4.1.7. SEE ALSO
ReleaseFiles, ReleaseSpace, UpdateSpace, GetSpaceTokens, GetSpaceMetaData

4.2. ChangeSpaceForFilesRequestStatus

4.2.1. NAME
ChangeSpaceForFilesRequestStatus – is used to check the previously submitted *ChangeSpaceForFiles* request.

4.2.2. Functional Description
This function is used to check the status of the previous request to *ChangeSpaceForFiles,* when asynchronous operation was necessary in the SRM. Request token must have been provided in response to the *ChangeSpaceForFiles*.

4.2.3. Parameter Description

In	Out

User ID
Authorization ID
Request Token
	Request Status {
	Status information
	}
	Estimated Processing Time
	SURL Statuses {
	SURL,
	Status information
	}

Input parameters

- *User ID* (required)
  User authentication identifier.
- *Authorization ID*
  User authorization information. The *Authorization ID* may be NULL.
- *Request Token* (required)
  A handle associated with the request to change the space properties on *SURLs*. The *Request Token* was returned by the function initiating the request (e.g. *ChangeSpaceForFiles*).

Output parameters

- *Request Status* (required)
  Output reporting the success or failure of the request. Textual description of explanation for what happened to the request, specially in case of any failures, would help client diagnose the case. *Status information* is required and it is recommended to return as an

enumerated codes separated as status code and error code, so that when client is a non-human program, response to the status information can be pre-programmed.

- *Estimated Processing Time*

  Output parameter in seconds described how long the space reservation request may take in estimation, in case *ChangeSpaceForFiles* is processed with delay. It is used to indicate the estimation time to complete the request, when known.

- *SURL Statuses*

  Output reporting the status of each file in the request. *SURL Statuses* may be empty and NULL. If returned to the client, *SURL* and its *status information* are required to return.

  - *SURL* (required)

    SURL that client has requested to clean up.

4.2.4. NOTES on the Behavior

a) When space transition is completed successfully, SRM_SUCCESS must be returned for each SURL.

b) If changing space is not completed, *Estimate Processing Time* is returned when known.

c) If all files are still in the queue and none of the files are completed in changing space, the returned status information should include SRM_REQUEST_QUEUED.

d) If some files are queued, and some files are completed in changing space, SRM_REQUEST_INPROGRESS must be returned as the return status information. Each file should have its own status information.

e) If the target space is only large enough to transfer a subset of the files, the request will continue taking place until the target space cannot hold any more files, and the request must be failed. The status information of the request must include an error of SRM_EXCEED_ALLOCATION in such case.

4.2.5. Status information returned upon request execution

On successful request, the *status information* returns SRM_SUCCESS.
On failure, the *status information* returns SRM_FAILURE and the detailed failure information along with the textual explanation in a human readable form would help the client diagnose the failed request. If only partial files were successful, the *status information* returns SRM_PARTIAL_SUCCESS, and the *SURL Statuses* should explain on those failed files.
When detailed failure can be set to one of the followings:

For request level status,

SRM_SUCCESS

- All file requests are successfully completed. All *SURLs* have new *Target Space Token.*

SRM_PARTIAL_SUCCESS

- All requests are completed. Some SURL requests have new *Target Space Token,* and some SURL requests are failed to have new *Target Space Token*. Details are on the files status.

SRM_ REQUEST_QUEUED

- Request submission was successful and the entire request is still on the queue.

SRM_ REQUEST_INPROGRESS

- Some files are still queued, and some files are completed in space transition.

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
- *Request Token* does not refer to an existing known request in the SRM server.
- *Target Space Token* refers to a forbidden transition by the SRM implementation.

SRM_SPACE_LIFETIME_EXPIRED
- target space that is associated with *Target Space Token* has an expired lifetime.

SRM_EXCEED_ALLOCATION
- target space that is associated with *Target Space Token* is not enough to hold *SURLs*.

SRM_INTERNAL_ERROR
- SRM has an internal transient error, and client may try again.

SRM_FAILURE
- All file requests are failed.
- any other request failure. *Explanation* needs to be filled for details.

SRM_NOT_SUPPORTED
- *function* is not supported in the SRM
- any input parameter is not supported in the SRM server
- a particular type of an input parameter is not supported in the SRM server

For file level status,

SRM_SUCCESS
- successful request completion for the *SURL*. The *SURL* has a new *Target Space Token.*

SRM_ REQUEST_QUEUED
- file request is on the queue.

SRM_ REQUEST_INPROGRESS
- file request is being processed.

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

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

SRM_INVALID_REQUEST
- *Target Space Token* refers to a forbidden transition for the particular SURL by the SRM implementation.

SRM_EXCEED_ALLOCATION
- target space that is associated with *Target Space Token* is not enough to hold *SURL*.

SRM_FILE_LOST
- the requested file with the SURL is permanently lost.

SRM_FILE_BUSY

- client requests for files which there is an active srmPrepareToPut (no srmPutDone is not yet called) for.
- The requested file with the SURL is being used by other clients.

SRM_FILE_UNAVAILABLE
- the requested file with the SURL is temporarily unavailable.

SRM_FAILURE
- any other request failure. *Explanation* needs to be filled for details.

4.2.6. SEE ALSO
ChangeSpaceForFiles, ReleaseFiles, ReleaseSpace, UpdateSpace, GetSpaceTokens, GetSpaceMetaData

4.3. ExtendFileLifeTimeInSpace

4.3.1. NAME
This function is used to extend lifetime of the files (SURLs) in a space.

4.3.2. Functional Description
ExtendFileLifeTimeInSpace is used to extend lifetime of the files (SURLs) in a space.

4.3.3. Parameter Description

In	Out
User ID	Request Status {
Authorization ID	Status information
Space Token	}
SURLs	SURL Statuses {
New Lifetime	SURL,
	Status information
	New Lifetime Updated
	}

Input parameters

- *User ID* (required)
  User authentication identifier.
- *Authorization ID*
  User authorization information. The *Authorization ID* may be NULL.
- *Spac Token* (required)
  A handle associated with the space to update SURLs in the spce. The *Space Token* is acquired separately (e.g. *ReserveSpace, GetSpaceTokens*).
- *New Lifetime*
  New Lifetime is the desired lifetime to be updated on each SURL in the space.

Output parameters

- *Request Status* (required)
  Output reporting the success or failure of the request. Textual description of explanation for what happened to the request, specially in case of any failures, would help client diagnose the case. *Status information* is required and it is recommended to return as an enumerated codes separated as status code and error code, so that when client is a non-human program, response to the status information can be pre-programmed.
- *SURL Statuses*
  Output reporting the status of each file in the request. *SURL Statuses* may be empty and NULL. If returned to the client, *SURL* and its *status information* are required to return.
  - *SURL* (required)
    SURL that client has requested to clean up.
  - *New Lifetime Updated*
    New lifetime that is updated on the *SURL*. It must not exceed the remaining lifetime of the space.

4.3.4. NOTES on the Behavior

a) When *Space Token* is provided, the lifetime of the files (*SURLs*) in the space associated with the space token will be updated..
b) *New Lifetime* is relative to the calling time. Lifetime will be set from the calling time for the specified period.
c) The new file lifetime, *New LifetTime Updated* must not exceed the remaining lifetime of the space.
d) The number of lifetime extensions maybe limited by SRM according to its policies.
e) If original lifetime is longer than the requested one, then the requested one will be assigned.
f) If *New Lifetime* is not specified, the SRM may use its default to assign the *New Lifetime*.

4.3.5. Status information returned upon request execution

On successful request, the *status information* returns SRM_SUCCESS.
On failure, the *status information* returns SRM_FAILURE and the detailed failure information along with the textual explanation in a human readable form would help the client diagnose the failed request. If only partial files were successful, the *status information* returns SRM_PARTIAL_SUCCESS, and the *SURL Statuses* should explain on those failed files.
When detailed failure can be set to one of the followings:

For request level status,

SRM_SUCCESS
- All requests are successfully completed. All *SURLs* have a new extended lifetime.

SRM_PARTIAL_SUCCESS
- All requests are completed. Some *SURLs* have a new extended lifetime, and some *SURLS* have failed. Details are on the files status.

SRM_AUTHENTICATION_FAILURE
- SRM server failed to authenticate the client

SRM_AUTHORIZATION_FAILURE

- client is not authorized to extend lifetime of files in the space specified by the space token.

SRM_INVALID_REQUEST
- *Space Token* is empty.
- *Space Token* does not refer to an existing known space in the SRM server.

SRM_INTERNAL_ERROR
- SRM has an internal transient error, and client may try again.

SRM_FAILURE
- All file requests updating lifetimes in a space are failed.
- any other request failure. *Explanation* needs to be filled for details.

SRM_NOT_SUPPORTED
- *function* is not supported in the SRM server
- any input parameter is not supported in the SRM server
- a particular type of an input parameter is not supported in the SRM server

For file level status,

SRM_SUCCESS
- successful request completion for the *SURL*. The *SURL* has a new extended lifetime.

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

SRM_AUTHORIZATION_FAILURE
- client is not authorized to extend the lifetime for the file that is associated with the *SURL*

SRM_FILE_LOST
- the requested file is permanently lost.

SRM_FILE_UNAVAILABLE
- the requested file is temporarily unavailable.

SRM_FILE_LIFETIME_EXPIRED
- the requested file is expired already.

SRM_FAILURE
- any other request failure. *Explanation* needs to be filled for details.

4.3.6. SEE ALSO
ReleaseFiles, ReleaseSpace, UpdateSpace, GetSpaceTokens, GetSpaceMetaData

4.4. GetSpaceMetaData

4.4.1. NAME
GetSpaceMetaData - is used to retrieve meta information of a space.

4.4.2. Functional Description

This function is used to get information of a space. Space token must be provided, and space tokens are returned upon a completion of a space reservation through *ReserveSpace* or *ReserveSpaceRequestStatus*.

4.4.3. Parameter Description

In	Out

User ID
Authorization ID
Space Tokens
	Request Status {
	Status information
	}
	Space Meta Data Infos{
	Space Token,
	Status formation
	Retention Policy Mode,
	Access Latency Mode,
	Owner ID,
	Total Size,
	Guaranteed Size,
	Unused Size,
	Lifetime Assigned,
	Lifetime Left
	}

Input parameters

- *User ID* (required)
  User authentication identifier.
- *Authorization ID*
  User authorization information. The *Authorization ID* may be NULL.
- *Space Tokens* (required)
  Handles associated with the space. The *Space Tokens* are acquired separately (e.g. srmReserveSpace).

Output parameters

- *Request Status* (required)
  Output reporting the success or failure of the request. Textual description of explanation for what happened to the request, specially in case of any failures, would help client diagnose the case. *Status information* is required and it is recommended to return as an enumerated codes separated as status code and error code, so that when client is a non-human program, response to the status information can be pre-programmed.
- *Space Meta Data Infos*
  Output parameter reporting the space information in the request. *Space Meta Data Infos* may be empty and NULL. If returned to the client, *Space Token* and its *status information* are required to return. This refers to a single space with retention policy. It does not include the extra space needed to hold the directory structures, if there is any.
  - *Space Token* (required)

A handle associated with the space in the request.

- *Retention Policy Mode*
  Retention Policy type of the space. It is requested and assigned at the time of space reservation through *ReserveSpace* and *ReserveSpaceRequestStatus.*
- *Access Latency Mode*
  Access latency type of the space.
- *Owner ID*
  Space owner.
- *Total Size*
  Total space size in bytes.
- *Guaranteed Size*
  Guaranteed space size in bytes.
- *Unused Size*
  unused space size in bytes.
- *Lifetime Assigned*
  Life time of the space that is initially assigned, in seconds.
- *Lifetime Left*
  Remaining life time of the space in seconds.

4.4.4. 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.

4.4.5. Status information returned upon request execution

On successful request, the *status information* returns SRM_SUCCESS.
On failure, the *status information* returns SRM_FAILURE and the detailed failure information along with the textual explanation in a human readable form would help the client diagnose the failed request. If only partial files were successful, the *status information* returns SRM_PARTIAL_SUCCESS, and the *SURL Statuses* should explain on those failed files.
When detailed failure can be set to one of the followings:

For request level status,

SRM_SUCCESS
- successful request completion. Information of all requested spaces are returned successfully.

SRM_PARTIAL_SUCCESS
- Request is completed. Information of some requested spaces are returned successfully, and some are failed to be returned.

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.

SRM_INVALID_REQUEST

- *Space Tokens* are empty.

SRM_INTERNAL_ERROR
- SRM has an internal transient error, and client may try again.

SRM_FAILURE
- All space requests are failed.
- any other request failure. *Explanation* needs to be filled for details.

SRM_NOT_SUPPORTED
- *function* is not supported in the SRM server

For space level status,

SRM_SUCCESS
- successful request completion for the *Space Token.* Space information is successfully returned.

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

SRM_INVALID_REQUEST
- *Space Token* does not refer to an existing known space in the SRM server.

SRM_SPACE_LIFETIME_EXPIRED
- The life time on the space that is associated with the *spaceToken* has expired

SRM_FAILURE
- any other request failure. *Explanation* needs to be filled for details.

4.4.6. SEE ALSO
GetSpaceTokens, ReserveSpace, UpdateSpace

4.5. GetSpaceTokens

4.5.1. NAME
GetSpaceToken - returns space tokens for currently allocated spaces.

4.5.2. Functional Description
GetSpaceToken returns space tokens for currently allocated spaces.

4.5.3. Parameter Description

In	Out
User ID	Request Status {
Authorization ID	Status information
User Space Description	}
	Space Tokens

Input parameters

- *User ID* (required)
  User authentication identifier.
- *Authorization ID*

User authorization information. The *Authorization ID* may be NULL.

- *User Space Description*
  String containining description of the space. The description was given by the client at the time of the request (e.g. *ReserveSpace*). The *User Space Description* may be NULL.

Output parameters

- *Request Status* (required)
  Output reporting the success or failure of the request. Textual description of explanation for what happened to the request, specially in case of any failures, would help client diagnose the case. *Status information* is required and it is recommended to return as an enumerated codes separated as status code and error code, so that when client is a non-human program, response to the status information can be pre-programmed.
- *Space Tokens*
  Output parameter returning the space tokens owned by the client. *Space Tokens* may be NULL and empty.

4.5.4. NOTES on the Behavior

a) If *User Space Description* is null, the SRM returns all *Space Tokens* that the client owns.
b) If the client assigned the same name to multiple space reservations, the client will get back multiple space tokens.

4.5.5. Status information returned upon request execution

On successful request, the *status information* returns SRM_SUCCESS.
On failure, the *status information* returns SRM_FAILURE and the detailed failure information along with the textual explanation in a human readable form would help the client diagnose the failed request.
When detailed failure can be set to one of the followings:

SRM_SUCCESS
- All requests are successfully completed. Space tokens are returned successfully.

SRM_AUTHENTICATION_FAILURE
- SRM server failed to authenticate the client

SRM_AUTHORIZATION_FAILURE
- client is not authorized to request *Space Tokens* associated with the *User Space Description*

SRM_INVALID_REQUEST
- *User Space Description* does not refer to an existing space description.

SRM_INTERNAL_ERROR
- SRM has an internal transient error, and client may try again.

SRM_FAILURE
- any other request failure. *Explanation* needs to be filled for details.

SRM_NOT_SUPPORTED
- *function* is not supported in the SRM server

4.5.6. SEE ALSO
GetSpaceMetaData, GetRequestTokens, Ls, LsDetails

4.6. PurgeFromSpace

4.6.1. NAME
PurgeFromSpace – is used to remove files from the give space.

4.6.2. Functional Description
This function is used when removing files from the given space is needed. Difference from *ReleaseFiles* and *AbortFiles* is that *PurgeFromSpace* is not associated with a request. This function must not remove the SURLs, but only the "copies" or "states" of the SURLs. *Rm* must be used to remove SURLs.

4.6.3. Parameter Description

In	Out
User ID	Request Status {
Authorization ID	Status information
Storage System Info	}
SURLs	SURL Statuses {
Space Token	SURL,
	Status information
	}

Input parameters

- *User ID* (required)
  User authentication identifier.
- *Authorization ID*
  User authorization information. The *Authorization ID* may be NULL.
- *SURLs* (required)
  SURLs to remove from the requested space.
- *Space Token* (required)
  A space handle associated with the space to remove all SURLs in the space.
- *Storage System Info*
  Information specific to the underlying storage system that is associated with the *Space Token*.

Output parameters

- *Request Status* (required)
  Output reporting the success or failure of the request. Textual description of explanation for what happened to the request, specially in case of any failures, would help client diagnose the case. *Status information* is required and it is recommended to return as an

enumerated codes separated as status code and error code, so that when client is a non-human program, response to the status information can be pre-programmed.

- *SURL Statuses*

  Output parameter reporting the status of each file in the request. *SURL Statuses* may be empty and NULL. If returned to the client, *SURL* and its *status information* are required to return.

  - *SURL* (required)

    SURL that client has requested to clean up.

4.6.4. NOTES on the Behavior

a) If the specified *SURL* is the only remaining copy of the file in the storage system, SRM_LAST_COPY must be returned in the status information. To remove the last copy of the *SURL*, *Rm* may be used.

b) If the client has an administers role that SRM server can accept in an understandable form, this request will forcefully release the pins owned by the group, and remove the "copy" (or "state") of the file.

c) In most cases, all pins on files that are associated with the client will be released. In such cases, files may still be pinned by others and SRM_FILE_BUSY will be returned in the status information.

d) SRM will remove only the "copies" (or "state") of the *SURLs* associated with the space token.

4.6.5. Status information returned upon request execution

On successful request, the *status information* returns SRM_SUCCESS.
On failure, the *status information* returns SRM_FAILURE and the detailed failure information along with the textual explanation in a human readable form would help the client diagnose the failed request.
When detailed failure can be set to one of the followings:

For request level status,

SRM_SUCCESS

- All requests are successfully completed. All *SURLs* are purged from the space specified by the *Space Token*.

SRM_PARTIAL_SUCCESS

- All requests are completed. Some *SURLs* are successfully purged from the space specified by the *Space Token*, and some *SURLs* are failed to be purged from the space specified by the *Space Token*. Details are on the files status.

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 *Space Token*

SRM_INVALID_REQUEST

- *SURLs* is empty.
- *Space Token* is empty.
- *Space Token* does not refer to an existing known space in the SRM server.

SRM_INTERNAL_ERROR
- SRM has an internal transient error, and client may try again.

SRM_FAILURE
- All file requests are failed.
- any other request failure. *Explanation* needs to be filled for details.

SRM_NOT_SUPPORTED
- *function* is not supported in the SRM server

For file level status,

SRM_SUCCESS
- successful request completion for the *SURL*. *SURL* is purged from the space specified by the *Space Token*.

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

SRM_AUTHORIZATION_FAILURE
- Client is not authorized to purge *SURL* in the space that is associated with *Space Token*

SRM_FILE_LOST
- the request file is permanently lost.

SRM_FILE_BUSY
- client requests for files which there is an active *PrepareToPut* (no *PutDone* is not yet called) for.
- The requested file is used by other clients.

SRM_FILE_UNAVAILABLE
- the requested file is temporarily unavailable.

SRM_LAST_COPY
- the requested file is the last copy and will not be purged from the space. *Rm* must be used to remove the last copy.

SRM_FAILURE
- any other request failure. *Explanation* needs to be filled for details.

4.6.6. SEE ALSO
PurgeFromSpace, ReleaseFiles, ReleaseSpace, UpdateSpace, GetSpaceTokens, GetSpaceMetaData

4.7. ReleaseSpace

4.7.1. NAME
ReleaseSpace - releases an occupied space.

4.7.2. Functional Description
ReleaseSpace releases an occupied space.

4.7.3. Parameter Description

In	Out

User ID
Authorization ID
Storage System Info
Space Token
Flag for Forceful File Release
	Request Status {
	Status information
	}

Input parameters

- *User ID* (required)
  User authentication identifier..
- *Authorization ID*
  User authorization information. The *Authorization ID* may be NULL.
- *Storage System Info*
  Information specific to the underlying storage system. The *Storage System Info* may be NULL.
- *Space Token* (required)
  A handle associated with the space to release. The *Space Token* is acquired separately (e.g. *ReserveSpace*).
- *Flag for Forceful File Release*
  Flag for Forceful File Release indicates that the space must be released regardless of all files that it contains and their status.

Output parameters

- *Request Status* (required)
  Output reporting the success or failure of the request. Textual description of explanation for what happened to the request, specially in case of any failures, would help client diagnose the case. *Status information* is required and it is recommended to return as an enumerated codes separated as status code and error code, so that when client is a non-human program, response to the status information can be pre-programmed.

4.7.4. NOTES on the Behavior

a) When *Flag for Forceful File Release* is false, 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, *Flag for Forceful File Release* must be set to true.

b) When space is releasable and *Flag for Forceful File Release* is true, all the files in the space are released, even in durable or permanent space.

c) *ReleaseSpace* 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 *Rm* 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.

4.7.5. Status information returned upon request execution

On successful request, the *status information* returns SRM_SUCCESS.

On failure, the *status information* returns SRM_FAILURE and the detailed failure information along with the textual explanation in a human readable form would help the client diagnose the failed request.
When detailed failure can be set to one of the followings:

SRM_SUCCESS
- successful request completion. Space is successfully released.

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 *Space Token*

SRM_INVALID_REQUEST
- *Space Token* does not refer to an existing known space in the SRM server.

SRM_INTERNAL_ERROR
- SRM has an internal transient error, and client may try again.

SRM_NOT_SUPPORTED
- *Flag for Forceful File Release* is not supported
- *function* is not supported

SRM_FAILURE
- space still contains pinned files.
- space associated with space is already released.
- any other request failure. *Explanation* needs to be filled for details.

4.7.6. SEE ALSO
PurgeFromSpace, ReserveSpace, GetSpaceMetaData, GetSpaceTokens, UpdateSpace

4.8. ReserveSpace

4.8.1. NAME
ReserveSpace - facilitates negotiation of space reservation.

4.8.2. Functional Description
ReserveSpace facilitates negotiation of space reservation. This function is used to reserve a space in advance for the upcoming requests to get some guarantee on the file management. Asynchronous space reservation may be necessary for some SRMs to serve many concurrent requests.

4.8.3. Parameter Description

In	Out

User ID	Request Status {
Authorization ID	Status information
Storage System Info	}
Retention Policy Mode	Request Token
Access Latency Mode	Estimated Processing Time
Expected File Sizes	Space Token
User Space Description	Retention Policy Mode
Desired Size Of Total Space	Access Latency Mode
Desired Size Of Guaranteed Space	Size Of Total Reserved Space
Lifetime Of Reserved Space	Size Of Guaranteed Reserved Space
Transfer Information	Lifetime Of Reserved Space

Input parameters

- *User ID* (required)
  User authentication identifier.
- *Authorization ID*
  User authorization information. The *Authorization ID* may be NULL.
- *Storage System Info*
  Information specific to the underlying storage system. The *Storage System Info* may be NULL.
- *Retention Policy Mode* (required)
  Type of space to reserve in Retention Policy.
- *Access Latency Mode*
  Type of space to reserve with Access Latency Mode.
- *Expected File Sizes*
  A hint that SRM server can use to reserve consecutive storage sizes for the request.
- *User Space Description*
  Description of the space. The description will be used to retrieve *Space Token* later with *GetSpaceTokens*(). The *User Space Description* may be NULL.
- *Desired Size Of Total Space*
  Desired total space size in bytes. SRM may assign a default size if not provided.
- Desired S*ize Of Guaranteed Space* (required)
  The guaranteed space size in bytes that client needs to work at the minimum.
- *Lifetime Of Reserved Space*
  Desired life time of the space in seconds. SRM may assign a default size if not provided.
- *Transfer Information*
  Expected transfer information that the clients would be interested in using with the reserved space.

Output parameters

- *Request Status* (required)
  Output reporting the success or failure of the request. Textual description of explanation for what happened to the request, specially in case of any failures, would help client diagnose the case. *Status information* is required and it is recommended to return as an

enumerated codes separated as status code and error code, so that when client is a non-human program, response to the status information can be pre-programmed.

- *Request Token*
  Output parameter request handle is associated with the request for the later asynchronous status request. *Request Token* may be NULL, in case *srmReserveSpace* is processed without delay.
- *Estimated Processing Time*
  Output parameter in seconds described how long the space reservation request may take in estimation, in case *ReserveSpace* is processed with delay. It is used to indicate the estimation time to complete the space reservation request, when known.
- *Space Token*
  Output parameter space handle is associated with the request for the later asynchronous space related request. It is a reference handle of the reserved space. *Space Token* may be NULL, in case space reservation is failed for the client.
- *Retention Policy Mode*
  Output parameter reporting the space type in Retention Policy that SRM server reserved upon the successful request.
- *Access Latency Mode*
  Output parameter reporting the space type in Access Latency Mode that SRM server reserved upon the successful request.
- *Size Of Total Reserved Space*
  Output parameter reporting the size of the total space that SRM server reserved upon the successful request. It is in the best effort.
- *Size Of Guaranteed Reserved Space*
  Output parameter reporting the size of the guaranteed space that SRM server reserved upon the successful request.
- *Lifetime Of Reserved Space*
  Output parameter reporting the life time of the space that SRM server reserved upon the successful request.

4.8.4. NOTES on the Behavior

a) *Lifetime Of Reserved Space* is not needed when requesting permanent space. It is ignored for permanent space. If the input parameter *Lifetime Of Reserved Space* is not provided, the lifetime of the reserved space is set to "infinite".

b) SRM may return its default space size and lifetime if not requested by the client. SRM may return an error SRM_INVALID_REQUEST if SRM does not support default space sizes.

c) If input parameter *Desired Size Of Total Space* is not specified, the SRM will return its default space size.

d) *Storage System Info* is optional and used for the case that the storage system requires an additional security information.

e) If *Desired* S*ize Of Total Space* is not specified, the SRM may return its default space size.

f) The difference between *Size Of Total Reserved Space* and *Size Of Guaranteed Reserved Space* is on best effort basis.

g) If the input parameter *Retention Policy Mode* cannot be satisfied by the SRM server, an error SRM_INVALID_REQUEST must be returned.

h) Asynchronous space reservation may be necessary for some SRMs to serve many concurrent requests. In such case, request token must be returned, and space token must not be assigned and returned until space reservation is completed, to prevent the usage of the space token in other interfaces before the space reservation is completed. If the space reservation can be done immediately, request token must not be returned.

i) When asynchronous space reservation is necessary, the returned status information must include SRM_REQUEST_QUEUED.

j) Input parameter *Expected File Sizes* is a hint that SRM server can use to reserve consecutive storage sizes for the request. At the time of space reservation, if space accounting is done only at the level of the total size, this hint would not help. In such case, the expected file size at the time of srmPrepareToPut will describe how much consecutive storage size is needed for the file. However, some SRMs may get benefits from these hints to make a decision to allocate some blocks in some specific devices.

k) Output parameter *Size Of Total Reserved Space* is in best effort bases. For guaranteed space size, *Size Of Guaranteed Reserved Space* should be checked. These two numbers may match, depending on the storage systems.

l) If an operation is successful, *Size Of Guaranteed Reserved Space*, *Lifetime Of Reserved Space* and *Space Token* are required to return to the client.

4.8.5. Status information returned upon request execution

On successful request, the *status information* returns SRM_SUCCESS.

On failure, the *status information* returns SRM_FAILURE and the detailed failure information along with the textual explanation in a human readable form would help the client diagnose the failed request.

When detailed failure can be set to one of the followings:

SRM_SUCCESS
- successful request completion. Space is reserved successfully as the client requested.

SRM_ REQUEST_QUEUED
- successful request submission and acceptance. Request token must be returned, and space token must not be assigned and returned.

SRM_LOWER_SPACE_GRANTED
- successful request completion, but lower space size is allocated than what the client requested

SRM_AUTHENTICATION_FAILURE
- SRM server failed to authenticate the client

SRM_AUTHORIZATION_FAILURE
- client is not authorized to reserve space

SRM_INVALID_REQUEST
- the input parameter *Retention Policy Mode* cannot be satisfied by the SRM server.
- If space size or lifetime is not requested by the client, and SRM does not support default values for space size or lifetime.
- input parameters are invalid.

SRM_NO_USER_SPACE

- SRM server does not have enough user space for the client for client to request to reserve.

SRM_NO_FREE_SPACE
- SRM server does not have enough free space for client to request to reserve.

SRM_EXCEED_ALLOCATION
- SRM server does not have enough space for the client to fulfill the request because the client request needs more than the allocated space quota for the client.

SRM_INTERNAL_ERROR
- SRM has an internal transient error, and client may try again.

SRM_FAILURE
- any other request failure. *Explanation* needs to be filled for details.

SRM_NOT_SUPPORTED
- *function* is not supported in the SRM server
- any input parameter is not supported in the SRM server
- a particular type of an input parameter is not supported in the SRM server

4.8.6. SEE ALSO
PurgeFromSpace, GetSpaceMetaData, GetSpaceTokens, ReleaseSpace, UpdateSpace, ReservedSpaceRequestStatus

4.9. ReserveSpaceRequestStatus

4.9.1. NAME
ReserveSpaceRequestStatus – checks the request status of the space reservation.

4.9.2. Functional Description
This function is used to check the status of the previous request to *ReserveSpace*, when asynchronous space reservation was necessary with the SRM. Request token must have been provided in response to the *ReserveSpace*.

4.9.3. Parameter Description

In	Out
User ID	Request Status {
Authorization ID	Status information
Request Token	}
	Estimated Processing Time
	Space Token
	Retention Policy Mode
	Access Latency Mode
	Size Of Total Reserved Space
	Size Of Guaranteed Reserved Space
	Lifetime Of Reserved Space

Input parameters

- *User ID* (required)
  User authentication identifier.
- *Authorization ID*
  User authorization information. The *Authorization ID* may be NULL.
- *Request Token* (required)
  A request handle associated with the previously submitted *srmReserveSpace* request. The *Request Token* was returned by the function initiating the request.

Output parameters

- *Request Status* (required)
  Output reporting the success or failure of the request. Textual description of explanation for what happened to the request, specially in case of any failures, would help client diagnose the case. *Status information* is required and it is recommended to return as an enumerated codes separated as status code and error code, so that when client is a non-human program, response to the status information can be pre-programmed.
- *Estimated Processing Time*
  Output parameter in seconds described how long the space reservation request may take in estimation, in case *ReserveSpace* is processed with delay. It is used to indicate the estimation time to complete the space reservation request, when known.
- *Space Token*
  Output parameter space handle is associated with the request for the later asynchronous space related request. It is a reference handle of the reserved space. *Space Token* may be NULL, in case space reservation is failed for the client.
- *Retention Policy Mode*
  Output parameter reporting the space type in Retention Policy that SRM server reserved upon the successful request.
- *Access Latency Mode*
  Output parameter reporting the space type in Access Latency Mode that SRM server reserved upon the successful request.
- *Size Of Total Reserved Space*
  Output parameter reporting the size of the total space that SRM server reserved upon the successful request.
- *Size Of Guaranteed Reserved Space*
  Output parameter reporting the size of the guaranteed space that SRM server reserved upon the successful request.
- *Lifetime Of Reserved Space*
  Output parameter reporting the life time of the space that SRM server reserved upon the successful request.

4.9.4. NOTES on the Behavior

a) If the space reservation is not completed yet, *Estimate Processing Time* is returned when known. The returned status information in such case should include SRM_REQUEST_QUEUED.

b) See notes for *ReserveSpace* for descriptions for output parameters.

c) If an operation is successful, *Size Of Guaranteed Reserved Space*, *Lifetime Of Reserved Space* and *Space Token* are required to return to the client.

4.9.5. Status information returned upon request execution

On successful request, the *status information* returns SRM_SUCCESS.
On failure, the *status information* returns SRM_FAILURE and the detailed failure information along with the textual explanation in a human readable form would help the client diagnose the failed request.
When detailed failure can be set to one of the followings:

SRM_REQUEST_QUEUED
- successful request submission and the request is still on the queue to be served.

SRM_REQUEST_INPROGRESS
- the request is being processed.

SRM_LOWER_SPACE_GRANTED
- successful request completion, but lower space size is allocated than what the client requested

SRM_SUCCESS
- successful request completion. Space is reserved successfully as the client requested.

SRM_AUTHENTICATION_FAILURE
- SRM server failed to authenticate the client

SRM_AUTHORIZATION_FAILURE
- client is not authorized to reserve space

SRM_INVALID_REQUEST
- *Request Token* does not refer to an existing known request in the SRM server.

SRM_EXCEED_ALLOCATION
- SRM server does not have enough space for the client to fulfill the request because the client request needs more than the allocated space for the client.

SRM_NO_USER_SPACE
- SRM server does not have enough user space for the client for the client for client to request to reserve.

SRM_NO_FREE_SPACE
- SRM server does not have enough free space for the client for client to request to reserve.

SRM_INTERNAL_ERROR
- SRM has an internal transient error, and client may try again.

SRM_FAILURE
- any other request failure. *Explanation* needs to be filled for details.

SRM_NOT_SUPPORTED
- *function* is not supported in the SRM server
- any input parameter is not supported in the SRM server
- a particular type of an input parameter is not supported in the SRM server

4.9.6. SEE ALSO

ReserveSpace, GetSpaceMetaData, GetSpaceTokens, ReleaseSpace, UpdateSpace

4.10. UpdateSpace

4.10.1. NAME
UpdateSpace - is to resize the space and/or extend the lifetime of a space.

4.10.2. Functional Description
UpdateSpace is to resize the space and/or extend the lifetime of a space. Asynchronous operation may be necessary for some SRMs to serve many concurrent requests.

4.10.3. Parameter Description

In	Out
User ID	Request Status {
Authorization ID	Status information
Storage System Info	}
Space Token	Request Token
Desired New Size Of Tota Space	Size Of Total Space
Desired New Size Of Guaranteed Space	Size Of Guaranteed Space
Desired New Lifetime	Lifetime Granted

Input parameters

- *User ID* (required)
  User authentication identifier.
- *Authorization ID*
  User authorization information. The *Authorization ID* may be NULL.
- *Storage System Info*
  Information specific to the underlying storage system. The *Storage System Info* may be NULL.
- *Space Token* (required)
  A space handle associated with the space to update. The *Space Token* is acquired separately (e.g. *ReserveSpace*).
- *Desired New Size Of Total Space*
  Desired total space size in bytes. SRM may assign a default if not provided
- *Desired New Size Of Guaranteed Space*
  The guaranteed space size in bytes that client needs to work at the minimum. SRM may assign a default if not provided.
- *Desired New Lifetime*
  Desired life time of the space in seconds. SRM may assign a default if not provided

Output parameters

- *Request Status* (required)

Output reporting the success or failure of the request. Textual description of explanation for what happened to the request, specially in case of any failures, would help client diagnose the case. *Status information* is required and it is recommended to return as an enumerated codes separated as status code and error code, so that when client is a non-human program, response to the status information can be pre-programmed.

- *Request Token*
  Output parameter request handle is associated with the request for the later asynchronous status request. *Request Token* may be NULL, in case *UpdateSpace* is processed without delay.
- *Size Of Total Space*
  Output parameter reporting the new size of the total space that SRM server has upon the successful update request.
- *Size Of Guaranteed Space*
  Output parameter reporting the new size of the guaranteed space that SRM server has upon the successful update request.
- *Lifetime Granted*
  Output parameter reporting the new life time of the space that SRM server has upon the successful update request.

4.10.4. 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 the request must be failed.
c) New size is the actual size of the space.
d) *Desired New Lifetime* is the new lifetime requested regardless of the previous lifetime. It may even be shorter than the remaining lifetime at the time of the call. It is relative to the calling time. Lifetime will be set from the calling time for the specified period.

4.10.5. Status information returned upon request execution

On successful request, the *status information* returns SRM_SUCCESS.
On failure, the *status information* returns SRM_FAILURE and the detailed failure information along with the textual explanation in a human readable form would help the client diagnose the failed request.
When detailed failure can be set to one of the followings:

SRM_SUCCESS
- successful request completion. Space is successfully updated as the client requested.

SRM_ REQUEST_QUEUED
- successful request submission and acceptance. Request token must be returned.

SRM_LOWER_SPACE_GRANTED
- successful request completion, but lower space size is allocated than what the client requested

SRM_AUTHENTICATION_FAILURE
- SRM server failed to authenticate the client

SRM_AUTHORIZATION_FAILURE

- client is not authorized to update the space that is associated with the *Space Token*

SRM_SPACE_LIFETIME_EXPIRED
- lifetime of the space that is associated with the *Space Token* is already expired.

SRM_INVALID_REQUEST
- *Space Token* does not refer to an existing known space in the SRM server.
- input parameter size or time is not provided.

SRM_EXCEED_ALLOCATION
- SRM server does not have enough space for the client to fulfill the request because the client request has more than the allocated space for the client.

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

SRM_INTERNAL_ERROR
- SRM has an internal transient error, and client may try again.

SRM_FAILURE
- New requested size is less than currently used space.
- any other request failure. *Explanation* needs to be filled for details.

SRM_NOT_SUPPORTED
- *function* is not supported

4.10.6. SEE ALSO
UpdateSpaceRequestStatus, ReserveSpace, ReleaseFiles, ReleaseSpace, UpdateSpace, GetSpaceTokens, GetSpaceMetaData

4.11. UpdateSpaceRequestStatus

4.11.1. NAME
UpdateSpaceRequestStatus - is to check the previously submitted request of *UpdateSpace*.

4.11.2. Functional Description
This function is used to check the status of the previous request to *UpdateSpace*, when asynchronous space update was necessary with the SRM. Request token must have been provided in response to the *UpdateSpace*.

4.11.3. Parameter Description

In	Out

User ID
Authorization ID
Request Token
	Request Status {
	Status information
	}
	Size Of Total Space
	Size Of Guaranteed Space
	Lifetime Granted

Input parameters

- *User ID* (required)
  User authentication identifier.
- *Authorization ID*
  User authorization information. The *Authorization ID* may be NULL.
- *Request Token* (required)
  A request handle associated with the previously submitted request. The *Request Token* was returned by the function initiating the request (e.g. *UpdateSpace*).

Output parameters

- *Request Status* (required)
  Output reporting the success or failure of the request. Textual description of explanation for what happened to the request, specially in case of any failures, would help client diagnose the case. *Status information* is required and it is recommended to return as an enumerated codes separated as status code and error code, so that when client is a non-human program, response to the status information can be pre-programmed.
- *Size Of Total Space*
  Output parameter reporting the new size of the total space that SRM server has upon the successful update request.
- *Size Of Guaranteed Space*
  Output parameter reporting the new size of the guaranteed space that SRM server has upon the successful update request.
- *Lifetime Granted*
  Output parameter reporting the new life time of the space that SRM server has upon the successful update request.

4.11.4. NOTES on the Behavior

a) Output parameters for new sizes are the new actual sizes of the space.
b) Output parameter, *Lifetime Granted* is the new lifetime assigned regardless of the previous lifetime. It might even be shorter than the previous lifetime. It is relative to the calling time.

4.11.5. Status information returned upon request execution

On successful request, the *status information* returns SRM_SUCCESS.
On failure, the *status information* returns SRM_FAILURE and the detailed failure information along with the textual explanation in a human readable form would help the client diagnose the failed request.
When detailed failure can be set to one of the followings:

SRM_REQUEST_QUEUED
- successful request submission and the request is still on the queue to be served.

SRM_REQUEST_INPROGRESS
- the request is being processed.

SRM_SUCCESS

- successful request completion. Space is successfully updated as the client requested.

SRM_LOWER_SPACE_GRANTED

- successful request completion, but lower space size is allocated than what the client requested

SRM_AUTHENTICATION_FAILURE

- SRM server failed to authenticate the client

SRM_AUTHORIZATION_FAILURE

- client is not authorized to update the space that is associated with the *Space Token*

SRM_SPACE_LIFETIME_EXPIRED

- lifetime of the space that is associated with the *Space Token* is already expired.

SRM_INVALID_REQUEST

- *Space Token* does not refer to an existing known space in the SRM server.
- input parameter size or time is not provided.

SRM_EXCEED_ALLOCATION

- SRM server does not have enough space for the client to fulfill the request because the client request has more than the allocated space for the client.

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

SRM_INTERNAL_ERROR

- SRM has an internal transient error, and client may try again.

SRM_FAILURE

- New requested size is less than currently used space.
- any other request failure. *Explanation* needs to be filled for details.

SRM_NOT_SUPPORTED

- *function* is not supported

4.11.6. SEE ALSO

UpdateSpaceRequestStatus, ReserveSpace, ReleaseFiles, ReleaseSpace, UpdateSpace, GetSpaceTokens, GetSpaceMetaData

5. Advanced feature set 3 : Directory Management Functions

summary:

Cp
LsDetails
Mkdir
Mv
Rmdir
CpRequestStatus
LsDetailsRequestStatus

details:

5.1. Cp

5.1.1. NAME

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

5.1.2. Functional Description

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

5.1.3. Parameter Description

In	Out
User ID	Request Token
Authorization ID	Request Status {
User Request Description	Status information
Total Request Time	}
Overwrite Mode	Remaining Total Request Time
Flag for Checksum	SURL Statuses {
Cp File Requests {	Source SURL,
Source SURL,	Target SURL,
Target SURL,	Status information,
Directory Option	File Size,
}	Estimated Wait Time,
Desired Target SURL Lifetime	Remaining File Lifetime
Target File Storage Type	}
Source Storage System Info	
Target Storage System Info	
Target Space Token	
Target File Retention Policy Info	

Input parameters

- *User ID* (required)
  User authentication identifier.

- *Authorization ID*
  User authorization information. The *Authorization ID* may be NULL.
- *Source Storage System Info*
  Information specific to the source storage system that is associated with the *Source SURL*s. The *Source Storage System Info* may be NULL.
- *Target Storage System Info*
  Information specific to the target storage system that is associated with the *Target SURL*s or *Target File Storage Type*. The *Target Storage System Info* may be NULL.
- *User Request Description*
  Description of the request. It may be used for later retrieval of the rquest token.
- *Total Request Time*
  Total Request Time means: if all the file transfer for this request must be complete in this *Total Request Time*. Otherwise, the request is returned as failed at the end of the *Total Request Time*, and SRM_REQUEST_TIMED_OUT must be returned as the request status code with individual file status of SRM_FAILURE with an appropriate explanation. All completed files must not be removed, but status of the files must be returned to the client.
- *Overwrite Mode*
  SRM needs to copy the files according to the *Overwrite Mode* on the target that SRM copies files into.
- *Flag for Checksum*
  Flag for Checksum indicates that checksum calculation for all files in the request needs to be performed when files get copied into its designated target space.
- *Cp File Requests* (required)
  Input parameter *Cp File Requests* consists of SURL information that client wants to copy from one location to another.
  - *Source SURL* (required)
    Source SURL
  - *Target SURL* (required)
    Target SURL
  - *Directory Option* (advanced option)
    An advanced option when Directory Management feature is supported. It includes *Flag for Source Directory, Flag for All Level Recursive* and *Number Of Recursive Levels*: *Flag for Source Directory* is a boolean indicator if *Source SURL* is a directory. *Flag for All Level Recursive* is a boolean indicator if *Source SURL* is a directory and all files under the SURL must be retrieved. *Number Of Recursive Levels* is a positive integer indicator for how many levels of the recursive directory must be browsed and all files in those directories to be retrieved.
- *Desired Target SURL Lifetime*
  Desired life time of the *Target SURLs* in seconds once when it's in the target SRM. SRM may assign a default lifetime, if not provided.
- *Target File Storage Type*
  Target File Storage Type indicates which type of storage that *Target SURL*s are copied into in target SRM.
- *Target Space Token* (advanced option)

An advanced option when Space Management feature is supported. A handle associated with the space if a particular space in file storage type is to be used. The *Space Token* is acquired separately (e.g. *ReserveSpace*). This is request level option. If file level option (*Space Token*) in the *Cp File Requests* is set, it takes the priority.

- *Target File Retention Policy Info*
  An advanced option when Space Management feature is supported. SRM will use the specific retention policy information when allocating space for files in the request.

Output parameters

- *Request Token*
  Output parameter request handle is associated with the request for the later asynchronous status request. *Request Token* may be NULL, in case *srmCp* is processed without delay.
- *Request Status* (required)
  Output reporting the success or failure of the request. Textual description of explanation for what happened to the request, specially in case of any failures, would help client diagnose the case. *Status information* is required and it is recommended to return as an enumerated codes separated as status code and error code, so that when client is a non-human program, response to the status information can be pre-programmed.
- *Remaining Total Request Time*
  Output parameter indicates the remaining total request time.
- *SURL Statuses*
  Output reporting the success or failure of the each SURL requests. *SURL Statuses* may be empty and NULL. If returned to the client, *SURL* and its *Status information* are required to return.
  - *Source SURL* (required)
    SURL that client has requested to copy the file from.
  - *Target SURL* (required)
    SURL that client has requested to copy the file to. If the client did not provide SURL at the time of request, SRM server generates a reference SURL that SRM server and client can refer to the file.
  - *Remaining File Lifetime*
    It indicates the remaining file lifetime on the SURL when the file is copied into its destination target. File lifetime is assigned after file copy is completed.

5.1.4. NOTES on the Behavior

a) Output parameter *Request Token* is optional, if the request is completed without delay.
b) *Cp* 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) Empty directories are copied as well.
f) 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 *Space Token*. The default value of fileType is “volatile”.

5.1.5. NOTES on the Advanced Behavior with Space Management Feature

a) *Space Token* must be a valid input parameter.

b) *Space Token* or *Target File Storage Type* or both must be provided. When *Space Token* is not provided and *Target File Storage Type* is provided, space allocation is needed in the *Target File Storage Type*.

5.1.6. Status information returned upon request execution

On successful request, the *status information* returns SRM_SUCCESS.
On failure, the *status information* returns SRM_FAILURE and the detailed failure information along with the textual explanation in a human readable form would help the client diagnose the failed request.
When detailed failure can be set to one of the followings:

For request level status,

SRM_REQUEST_QUEUED
- successful request submission and acceptance. Request token must be returned.

SRM_REQUEST_INPROGRESS
- Some files are completed, and some files are still on the queue. Details are on the files status. Request token must be returned.

SRM_SUCCESS
- All requests are successfully completed. All *Source SURLs* are copied into the target destination successfully.

SRM_PARTIAL_SUCCESS
- All requests are completed. Some file request is successfully copied into the target destination, and some file request is failed. Details are on the files status.

SRM_AUTHENTICATION_FAILURE
- SRM server failed to authenticate the client

SRM_AUTHORIZATION_FAILURE
- client is not authorized to submit the request
- Client is not authorized to copy files into the space that client provided with *Target Space Token* or *Target File Retention Policy Info*

SRM_INVALID_REQUEST
- If both input parameters *Target Space Token* and *Target File Retention Policy Info* are provided, then their types must match exactly.
- *Target Space Token* does not refer to an existing known space in the SRM server.

SRM_SPACE_LIFETIME_EXPIRED
- space associated with the *Target Space Token* is expired.

SRM_EXCEED_ALLOCATION
- space associated with the *Target Space Token* is not enough to hold all requested SURLs.

SRM_NO_USER_SPACE
- user space is not enough to hold all requested SURLs.

SRM_NO_FREE_SPACE
- SRM space is not enough to hold all requested SURLs for free.

SRM_INTERNAL_ERROR
- SRM has an internal transient error, and client may try again.

SRM_NOT_SUPPORTED
- SRM server does not support the given input parameters. For example, client requested *Desired File Storage Type* that is not supported by the SRM server.
- *Target File Retention Policy Info* does not refer to a supported retention policy in the SRM server.
- Directory operation is not supported in the SRM server.
- Recursive directory operation is not supported in the SRM server.
- any input parameter is not supported in the SRM server
- a particular type of an input parameter is not supported in the SRM server
- *function* is not supported in the SRM server

SRM_FAILURE
- all files requests are failed.
- any other request failure. *Explanation* needs to be filled for details.

For file level return status,

SRM_SUCCESS
- successful request completion for the file. The source *SURL* is copied into the target destination *Target SURL* successfully, and lifetime on the *Target SURL* is started.

SRM_REQUEST_QUEUED
- file request is still on the queue.

SRM_REQUEST_INPROGRESS
- file request is being served.

SRM_FILE_LOST
- the request file (*Source SURL*) is permanently lost.

SRM_FILE_BUSY
- client requests for files at the source (*Source SURL*) which there is an active *PrepareToPut* (no *PutDone* is not yet called) for.

SRM_FILE_UNAVAILABLE
- the request file (*Source SURL*) is temporarily unavailable.

SRM_FILE_LIFETIME_EXPIRED
- lifetime on *Target SURL* has expired, but the file is still in the cache.

SRM_INVALID_PATH
- *Source SURL* does not exist
- *Target SURL* does not refer to a valid path.

SRM_DUPLICATION_ERROR
- *Target SURL* refers to an existing SURL without no overwriting option.

SRM_AUTHORIZATION_FAILURE
- Client is not authorized to copy files from *Source SURL*
- Client is not authorized to copy files into *Target SURL*
- Client is not authorized to copy files into the space that client provided with *Target Space Token* or *Target File Retention Policy Info*

SRM_ABORTED
- The requested file has been aborted.

SRM_RELEASED
- The requested file has been released.

SRM_FAILURE
- any other request failure. *Explanation* needs to be filled for details.
- The file request would not be able to be completed within the *Total Request Time*.
- The requested file has been suspended because the request has timed out.

5.1.7. SEE ALSO
RemoteCopy

5.2. CpRequestStatus

5.2.1. NAME
CpRequestStatus - is a status call for *Cp*.

5.2.2. Functional Description
CpRequestStatus is a status call for *Cp*.

5.2.3. Parameter Description

In	Out
User ID	Request Status {
Authorization ID	Status information,
Request Token	explanation
offset	}
count	SURL Statuses {
	Source SURL,
	Target SURL,
	Status information,
	File Size,
	Estimated Wait Time,
	Remaining File Lifetime
	}
	Remaining Total Request Time

Input parameters

- *User ID* (required)
  User authentication identifier.
- *Authorization ID*
  User authorization information. The *Authorization ID* may be NULL.
- *Request Token* (required)

A request handle associated with the previously submitted *Cp* request. The *Request Token* was returned by the function initiating the request.

- *offset*
  Integer indicator for long listed responses that the listing starts from the *offset*.
- *count*
  Integer indicator for long listed responses that the listing contains the number of *count.*

Output parameters

- *Request Status* (required)
  Output reporting the success or failure of the request. Textual description of explanation for what happened to the request, specially in case of any failures, would help client diagnose the case. *Status information* is required and it is recommended to return as an enumerated codes separated as status code and error code, so that when client is a non-human program, response to the status information can be pre-programmed.
- *Remaining Total Request Time*
  Output parameter indicates the remaining total request time.
- *SURL Statuses*
  Output reporting the success or failure of the each SURL requests. *SURL Statuses* may be empty and NULL. If returned to the client, *SURL* and its *Status information* are required to return.
  - *Source SURL* (required)
    SURL that client has requested to copy the file from.
  - *Target SURL* (required)
    SURL that client has requested to copy the file to. If the client did not provide SURL at the time of request, SRM server generates a reference SURL that SRM server and client can refer to the file.
  - *Remaining File Lifetime*
    It indicates the remaining file lifetime on the SURL when the file is copied into its destination target. File lifetime is assigned after file copy is completed.

5.2.4. Status information returned upon request execution

On successful request, the *status information* returns SRM_SUCCESS.
On failure, the *status information* returns SRM_FAILURE and the detailed failure information along with the textual explanation in a human readable form would help the client diagnose the failed request. If only partial files were successful, the *status information* returns SRM_PARTIAL_SUCCESS, and the *SURL Statuses* should explain on those failed files.
When detailed failure can be set to one of the followings:

For request level status,

SRM_REQUEST_QUEUED
- successful request submission and acceptance. Request token must be returned.

SRM_REQUEST_INPROGRESS
- Some files are completed, and some files are still on the queue. Details are on the files status. Request token must be returned.

SRM_SUCCESS

- All requests are successfully completed. All *Source SURLs* are copied into the target destination successfully.

SRM_PARTIAL_SUCCESS

- All requests are completed. Some file request is successfully copied into the target destination, and some file request is failed. Details are on the files status.

SRM_AUTHENTICATION_FAILURE

- SRM server failed to authenticate the client

SRM_AUTHORIZATION_FAILURE

- client is not authorized to submit the request
- Client is not authorized to copy files into the space that client provided with *Target Space Token* or *Target File Retention Policy Info*

SRM_INVALID_REQUEST

- If both input parameters *Target Space Token* and *Target File Retention Policy Info* are provided, then their types must match exactly.
- *Target Space Token* does not refer to an existing known space in the SRM server.

SRM_SPACE_LIFETIME_EXPIRED

- space associated with the *Target Space Token* is expired.

SRM_EXCEED_ALLOCATION

- space associated with the *Target Space Token* is not enough to hold all requested SURLs.

SRM_NO_USER_SPACE

- user space is not enough to hold all requested SURLs.

SRM_NO_FREE_SPACE

- SRM space is not enough to hold all requested SURLs for free.

SRM_INTERNAL_ERROR

- SRM has an internal transient error, and client may try again.

SRM_NOT_SUPPORTED

- SRM server does not support the given input parameters. For example, client requested *Desired File Storage Type* that is not supported by the SRM server.
- *Target File Retention Policy Info* does not refer to a supported retention policy in the SRM server.
- Directory operation is not supported in the SRM server.
- Recursive directory operation is not supported in the SRM server.
- any input parameter is not supported in the SRM server
- a particular type of an input parameter is not supported in the SRM server
- *function* is not supported in the SRM server

SRM_FAILURE

- all files requests are failed.
- any other request failure. *Explanation* needs to be filled for details.

For file level return status,

SRM_SUCCESS

- successful request completion for the file. The source *SURL* is copied into the target destination *Target SURL* successfully, and lifetime on the *Target SURL* is started.

SRM_REQUEST_QUEUED
- file request is still on the queue.

SRM_REQUEST_INPROGRESS
- file request is being served.

SRM_FILE_LOST
- the request file (*Source SURL*) is permanently lost.

SRM_FILE_BUSY
- client requests for files at the source (*Source SURL*) which there is an active *PrepareToPut* (no *PutDone* is not yet called) for.

SRM_FILE_UNAVAILABLE
- the request file (*Source SURL*) is temporarily unavailable.

SRM_FILE_LIFETIME_EXPIRED
- lifetime on *Target SURL* has expired, but the file is still in the cache.

SRM_INVALID_PATH
- *Source SURL* does not exist
- *Target SURL* does not refer to a valid path.

SRM_DUPLICATION_ERROR
- *Target SURL* refers to an existing SURL without no overwriting option.

SRM_AUTHORIZATION_FAILURE
- Client is not authorized to copy files from *Source SURL*
- Client is not authorized to copy files into *Target SURL*
- Client is not authorized to copy files into the space that client provided with *Target Space Token* or *Target File Retention Policy Info*

SRM_ABORTED
- The requested file has been aborted.

SRM_RELEASED
- The requested file has been released.

SRM_FAILURE
- any other request failure. *Explanation* needs to be filled for details.
- The file request would not be able to be completed within the *Total Request Time*.
- The requested file has been suspended because the request has timed out.

5.2.5. SEE ALSO
RemoteCopy, Cp, GetRequestTokens

5.3. LsDetails

5.3.1. NAME
LsDetails - returns a list of files in the space with detailed information.

5.3.2. Functional Description

LsDetails returns a list of files in the space with detailed information. This operation may be asynchronous, and in such case, request handle must be returned.

5.3.3. Parameter Description

In	Out

User ID
Authorization ID
Storage System Info
SURLs
Directory Option
offset
count
	Request Token
	Request Status {
	Status information
	}
	SURL Meta Data Details Infos {
	SURL,
	Status information
	size,
	Creation Time,
	Last Modification Time,
	File Storage Type,
	File Type,
	Retention Policy Mode,
	Access Latency Mode,
	File Locality,
	Space Tokens,
	Lifetime Assigned,
	Lifetime Remained,
	Checksum Type,
	Checksum Value,
	Owner Permission,
	User Permissions,
	Group Permissions,
	Other Permission,
	SURL Meta Data Defails Infos On Subpaths
	}
	Total Number Of File In The Request

Input parameters

- *User ID* (required)
  User authentication identifier..
- *Authorization ID*
  User authorization information. The *Authorization ID* may be NULL.
- *Storage System Info*
  Information specific to the underlying storage system. The *Storage System Info* may be NULL.
- *SURLs*
  SURLs refer to files only.
- *Directory Option*

An advanced option when Directory Management feature is supported. It includes *Flag for Source Directory*, *Flag for All Level Recursive* and *Number Of Recursive Levels*: *Flag for Source Directory* is a boolean indicator if *Source SURL* is a directory. *Flag for All Level Recursive* is a boolean indicator if *Source SURL* is a directory and all files under the SURL must be retrieved. *Number Of Recursive Levels* is a positive integer indicator for how many levels of the recursive directory must be browsed and all files in those directories to be retrieved.

- *offset*
  Integer indicator for long listed responses that the listing starts from the *offset*.
- *count*
  Integer indicator for long listed responses that the listing contains the number of *count.*

Output parameters

- *Request Token*
  Output parameter request handle is associated with the request for the later asynchronous status request. *Request Token* may be NULL, in case *srmLsDetails* is processed without delay.
- *Request Status* (required)
  Output reporting the success or failure of the request. Textual description of explanation for what happened to the request, specially in case of any failures, would help client diagnose the case. *Status information* is required and it is recommended to return as an enumerated codes separated as status code and error code, so that when client is a non-human program, response to the status information can be pre-programmed.
- *SURL Meta Data Details Infos*
  Output parameter reporting the information of each file in the request. *SURL Meta Data Details Infos* may be empty and NULL. If returned to the client, *SURL*, *size*, and its *File Type* are required to return.
  - *SURL* (required)
    SURL that client has requested for the info.
  - *Status information*
    Status information on the *SURL*.
  - *explanation*
    explanation of the *Status information* for the *SURL* in case of not successful.
  - *size* (required)
    Size of the *SURL*. In case of SURL being a directory, *size* is expected to be 0.
  - *Creation Time*
    Creation time in UTC time that is associated with the *SURL*.
  - *Last Modification Time*
    Last modified time in UTC time that is associated with the *SURL*.
  - *File Storage Type*
    File storage type associated with the *SURL*.
  - *File Type* (required)
    File type associated with the *SURL*.
  - *Retention Policy Mode*
    Retention policy mode that is associated with the *SURL*.

- *Access Latency Mode*
  Access latency mode that is associated with the *SURL*.
- *File Locality*
  File Locality information that is associated with the *SURL*. It indicates where the file is located currently in the system.
- *Space Tokens*
  Space tokens that is associated with the *SURL*. It indicates where the file is currently located for the client. Only space tokens that the client has authorized to access to read the file must be returned.
- *Lifetime Assigned*
  Lifetime assigned to the *SURL*.
- *Lifetime Remained*
  Lifetime remained on the *SURL*.
- *Checksum Type*
  Checksum type if check is performed. For example, MD5.
- *Checksum Value*
  Checksum value if check is performed.
- *Owner Permission Mode*
  Owner permission information that is associated with the *SURL*. It may include *Owner ID* and *Owner Permission Mode*. Permissions on the SURL represent unix-like permissions: e.g. rwxr--r--. For ACL-like permissions, *GetPermission* must be used.
- User Permissions
  User permissions information that is associated with the *SURL*. It may include *User IDs* and their *Permission Modes*. Permissions on the SURL represent unix-like permissions: e.g. rwxr--r--. For ACL-like permissions, *GetPermission* must be used
- *Group Permission*s
  Group permissions information that is associated with the *SURL*. It may include *Group IDs* and their *Permission Modes*. Permissions on the SURL represent unix-like permissions: e.g. rwxr--r--. For ACL-like permissions, *GetPermission* must be used
- *Other Permission*
  Other permission that is associated with the *SURL*. Permissions on the SURL represent unix-like permissions: e.g. rwxr--r--. For ACL-like permissions, *GetPermission* must be used.
- *SURL Meta Data Defails Infos On Subpaths*
  In case of the recursive directory *SURL* listing, *SURL Meta Data Defails Infos On Subpaths* indicates sub-directory that contains files or directoties.

- Total Number Of Files In The Request
  Output parameter indicating how many files in the request for a hint to the asynchronous status cals.

5.3.4. NOTES on the Behavior

a) If output parameter *Flag for Partial List* is true, it indicates that only part of the result was returned. In this case, a *Request Token* must be returned.

b) If the entire result is returned, then output parameter *Request Token* is optional.
c) *SURL*s may refer to either directory or file.
d) If *Flag for All Level Recursive* is true, file lists of all level below current will be returned.
e) If *Flag for All Level Recursive* is true, it dominates, i.e. ignore *Number Of Recursive Levels*. If *Flag for All Level Recursive* is false or missing, then *Number Of Recursive Levels* must be honored. If *Number Of Recursive Levels* is 0 (zero) or missing, a single level is assumed.
f) If *Number Of Recursive Levels* is 0, then information about directory itself is returned.
g) If *Number Of Recursive Level* is 1, then information about files in the directory is returned.
h) Empty directories will be returned.
i) For non-existing file or directory, SRM_INVALID_PATH must be returned.

5.3.5. Status information returned upon request execution

On successful request, the *status information* returns SRM_SUCCESS.
On failure, the *status information* returns SRM_FAILURE and the detailed failure information along with the textual explanation in a human readable form would help the client diagnose the failed request. If only partial files were successful, the *status information* returns SRM_PARTIAL_SUCCESS, and the *SURL Meta Data Details Infos* should explain on those failed files.
When detailed failure can be set to one of the followings:

For request level status,

SRM_SUCCESS
- All requests are successfully completed. All *SURLs* are checked and the information for all *SURLs* is returned successfully.

SRM_PARTIAL_SUCCESS
- All requests are completed. Some *SURL* request is successfully completed, and some *SURL* request is failed. Details are on the files status.

SRM_REQUEST_QUEUED
- successful request submission and acceptance. Request token must be returned.

SRM_AUTHENTICATION_FAILURE
- SRM server failed to authenticate the client

SRM_AUTHORIZATION_FAILURE
- client is not authorized to request information

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_INTERNAL_ERROR
- SRM has an internal transient error, and client may try again.

SRM_INVALID_REQUEST
- Negative values for *Number Of Recursive Levels*, *offset* and *count* are provided.

SRM_NOT_SUPPORTED
- Directory operation (directory SURL, Flag for All Level Recursive and Number Of Recursive Levels) is not supported in SRM

SRM_FAILURE

- All files requests are failed.
- any other request failure. *Explanation* needs to be filled for details.

For file level status,

SRM_SUCCESS
- successful request completion for the *SURL*. The information for the *SURL* is checked and returned successfully.

SRM_FILE_IN_CACHE
- lifetime on *SURL* has expired, but the file is still in the cache.

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

SRM_AUTHORIZATION_FAILURE
- client is not authorized to receive the information of the *SURL* or to access the directory or sub-directories

SRM_FAILURE
- any other request failure. *Explanation* needs to be filled for details.

5.3.6. SEE ALSO
Ls, LsRequestStatus, LsDetailsRequestStatus

5.4. LsDetailsRequestStatus

5.4.1. NAME
LsDetailsRequestStatus - is an asynchronous call for *LsDetails* that is large.

5.4.2. Functional Description
LsDetailsRequestStatus is an asynchronous call for *LsDetails* that is large.

5.4.3. Parameter Description

In	Out
User ID	Request Status {
Authorization ID	Status information,
Request Token	explanation
offset	}
count	SURL Meta Data Details Infos {
	SURL,
	Status information
	size,
	Creation Time,
	Last Modification Time,
	File Storage Type,
	File Type,
	Retention Policy Mode,

	Access Latency Mode, File Locality, Space Tokens, Lifetime Assigned, Lifetime Remained, Checksum Type, Checksum Value, Owner Permission, User Permissions, Group Permissions, Other Permission, SURL Meta Data Defails Infos On Subpaths }

Input parameters

- *User ID* (required)
  User authentication identifier.
- *Authorization ID*
  User authorization information. The *Authorization ID* may be NULL.
- *Request Token* (required)
  A request handle associated with the previously submitted *LsDetails* request.
- *offset*
  Integer indicator for long listed responses that the listing starts from the *offset*.
- *count*
  Integer indicator for long listed responses that the listing contains the number of *count.*

Output parameters

- *Request Status* (required)
  Output reporting the success or failure of the request. Textual description of explanation for what happened to the request, specially in case of any failures, would help client diagnose the case. *Status information* is required and it is recommended to return as an enumerated codes separated as status code and error code, so that when client is a non-human program, response to the status information can be pre-programmed.
- *SURL Meta Data Details Infos*
  Output parameter reporting the information of each file in the request. *SURL Meta Data Details Infos* may be empty and NULL. If returned to the client, *SURL*, *size,* and its *File Type* are required to return.
  - *SURL* (required)
    SURL that client has requested for the info.
  - *Status information*
    Status information on the *SURL*.
  - *explanation*
    explanation of the *Status information* for the *SURL* in case of not successful.
  - *size* (required)

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

- *Creation Time*
  Creation time in UTC time that is associated with the *SURL*.
- *Last Modification Time*
  Last modified time in UTC time that is associated with the *SURL*.
- *File Storage Type*
  File storage type associated with the *SURL*.
- *File Type* (required)
  File type associated with the *SURL*.
- *Retention Policy Mode*
  Retention policy mode that is associated with the *SURL*.
- *Access Latency Mode*
  Access latency mode that is associated with the *SURL*.
- *File Locality*
  File Locality information that is associated with the *SURL*. It indicates where the file is located currently in the system.
- *Space Tokens*
  Space tokens that is associated with the *SURL*. It indicates where the file is currently located for the client. Only space tokens that the client has authorized to access to read the file must be returned.
- *Lifetime Assigned*
  Lifetime assigned to the *SURL*.
- *Lifetime Remained*
  Lifetime remained on the *SURL*.
- *Checksum Type*
  Checksum type if check is performed. For example, MD5.
- *Checksum Value*
  Checksum value if check is performed.
- *Owner Permission Mode*
  Owner permission information that is associated with the *SURL*. It may include *Owner ID* and *Owner Permission Mode*. Permissions on the SURL represent unix-like permissions: e.g. rwxr--r--. For ACL-like permissions, *GetPermission* must be used.
- User Permissions
  User permissions information that is associated with the *SURL*. It may include *User IDs* and their *Permission Modes*. Permissions on the SURL represent unix-like permissions: e.g. rwxr--r--. For ACL-like permissions, *GetPermission* must be used
- *Group Permission*s
  Group permissions information that is associated with the *SURL*. It may include *Group IDs* and their *Permission Modes*. Permissions on the SURL represent unix-like permissions: e.g. rwxr--r--. For ACL-like permissions, *GetPermission* must be used
- *Other Permission*

Other permission that is associated with the *SURL*. Permissions on the SURL represent unix-like permissions: e.g. rwxr--r--. For ACL-like permissions, *GetPermission* must be used.

- *SURL Meta Data Defails Infos On Subpaths*
  In case of the recursive directory *SURL* listing, *SURL Meta Data Defails Infos On Subpaths* indicates sub-directory that contains files or directoties.

5.4.4. NOTES on the Behavior

a) Empty directories will be returned.
b) For non-existing file or directory, SRM_INVALID_PATH must be returned.

5.4.5. Status information returned upon request execution

On successful request, the *status information* returns SRM_SUCCESS.
On failure, the *status information* returns SRM_FAILURE and the detailed failure information along with the textual explanation in a human readable form would help the client diagnose the failed request. If only partial files were successful, the *status information* returns SRM_PARTIAL_SUCCESS, and the *SURL Meta Data Details Infos* should explain on those failed files.
When detailed failure can be set to one of the followings:

For request level status,

SRM_SUCCESS
- All requests are successfully completed. All *SURLs* are checked and the information for all *SURLs* is returned successfully.

SRM_PARTIAL_SUCCESS
- All requests are completed. Some *SURL* request is successfully completed, and some *SURL* request is failed. Details are on the files status.

SRM_REQUEST_QUEUED
- successful request submission and acceptance. Request token must be returned.

SRM_AUTHENTICATION_FAILURE
- SRM server failed to authenticate the client

SRM_AUTHORIZATION_FAILURE
- client is not authorized to request information

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_INTERNAL_ERROR
- SRM has an internal transient error, and client may try again.

SRM_INVALID_REQUEST
- Negative values for *Number Of Recursive Levels*, *offset* and *count* are provided.

SRM_NOT_SUPPORTED
- Directory operation (directory SURL, Flag for All Level Recursive and Number Of Recursive Levels) is not supported in SRM

SRM_FAILURE

- All files requests are failed.
- any other request failure. *Explanation* needs to be filled for details.

For file level status,

SRM_SUCCESS
- successful request completion for the *SURL*. The information for the *SURL* is checked and returned successfully.

SRM_FILE_IN_CACHE
- lifetime on *SURL* has expired, but the file is still in the cache.

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

SRM_AUTHORIZATION_FAILURE
- client is not authorized to receive the information of the *SURL* or to access the directory or sub-directories

SRM_FAILURE
- any other request failure. *Explanation* needs to be filled for details.

5.4.6. SEE ALSO
Ls, LsRequestStatus, LsDetails

5.5. Mkdir

5.5.1. NAME
Mkdir - create a directory in a local SRM space.

5.5.2. Functional Description
Mkdir create a directory in a local SRM space.

5.5.3. Parameter Description

In	Out
User ID	Request Status {
Authorization ID	Status information
Storage System Info	}
SURL	

Input parameters

- *User ID* (required)
  User authentication identifier.
- *Authorization ID*
  User authorization information. The *Authorization ID* may be NULL.
- *Storage System Info*
  String containining information specific to the underlying storage system. The *Storage System Info* may be NULL.
- *SURL* (required)

SURL to create as a directory.

Output parameters

- *Request Status* (required)
  Output reporting the success or failure of the request. Textual description of explanation for what happened to the request, specially in case of any failures, would help client diagnose the case. *Status information* is required and it is recommended to return as an enumerated codes separated as status code and error code, so that when client is a non-human program, response to the status information can be pre-programmed.

5.5.4. NOTES on the Behavior

a) Recursive creation of directories is not supported.
b) *SURL* may include paths, as long as all directory hierarchy exists.

5.5.5. Status information returned upon request execution

On successful request, the *status information* returns SRM_SUCCESS.
On failure, the *status information* returns SRM_FAILURE and the detailed failure information along with the textual explanation in a human readable form would help the client diagnose the failed request.
When detailed failure can be set to one of the followings:

SRM_SUCCESS
- All requests are successfully completed. *SURL* is created.

SRM_AUTHENTICATION_FAILURE
- SRM server failed to authenticate the client

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

SRM_INVALID_PATH
- *SURL* does not refer to a valid path
- component of *SURL* does not refer to an existing path

SRM_DUPLICATION_ERROR
- *SURL* exists already

SRM_INTERNAL_ERROR
- SRM has an internal transient error, and client may try again.

SRM_FAILURE
- any other request failure. *Explanation* needs to be filled for details.

SRM_NOT_SUPPORTED
- *function* is not supported in the SRM server

5.5.6. SEE ALSO

Ls, LsDetails, Rm, Rmdir

5.6. Mv

5.6.1. NAME
Mv - is to move a file from one local directory to another.

5.6.2. Functional Description
Mv is to move a file from one local directory to another.

5.6.3. Parameter Description

In	Out
userID	Request Status {
authorizationID	Status information
Storage System Info	}
Source SURL	
Target SURL	

Input parameters

- *User ID* (required)
  User authentication identifier.
- *Authorization ID*
  User authorization information. The *Authorization ID* may be NULL.
- *Storage System Info*
  Information specific to the underlying storage system that is associated with the *Source* and *Target SURL*. The *Storage System Info* may be NULL.
- *Source SURL* (required)
  SURL to move from.
- *Target SURL*
  SURL to move *Source SURL* to.

Output parameters

- *Request Token*
  Output parameter request handle is associated with the request for the later asynchronous status request. *Request Token* may be NULL, in case *srmMv* is processed without delay.
- *Request Status* (required)
  Output reporting the success or failure of the request. Textual description of explanation for what happened to the request, specially in case of any failures, would help client diagnose the case. *Status information* is required and it is recommended to return as an enumerated codes separated as status code and error code, so that when client is a non-human program, response to the status information can be pre-programmed.

5.6.4. NOTES on the Behavior
a) *SURL* may be applied to both directory and file.

5.6.5. NOTES on the Advanced Behavior with AuthorizationFeature
a) Authorization checks need to be performed on both *Source SURL* and *Target SURL*.

5.6.6. Status information returned upon request execution

On successful request, the *status information* returns SRM_SUCCESS.
On failure, the *status information* returns SRM_FAILURE and the detailed failure information along with the textual explanation in a human readable form would help the client diagnose the failed request.
When detailed failure can be set to one of the followings:

SRM_SUCCESS
- All requests are successfully completed. *SURL* is moved successfully from one local path to another local path.

SRM_AUTHENTICATION_FAILURE
- SRM server failed to authenticate the client

SRM_AUTHORIZATION_FAILURE
- Client is not authorized to move *Source SURL*.
- Client is not authorized to move a file into *Target SURL*

SRM_INVALID_PATH
- *Source SURL* does not refer to an existing known path
- *Target SURL* does not refer to a valid path

SRM_DUPLICATION_ERROR
- *Target SURL* exists already.

SRM_FILE_LOST
- the requested file is permanently lost.

SRM_FILE_BUSY
- client requests for files which there is an active srmPrepareToPut (no srmPutDone is not yet called) for.
- The requested file is being used by other clients.

SRM_FILE_UNAVAILABLE
- the requested file is temporarily unavailable.

SRM_INTERNAL_ERROR
- SRM has an internal transient error, and client may try again.

SRM_FAILURE
- any other request failure. *Explanation* needs to be filled for details.

SRM_NOT_SUPPORTED
- *function* is not supported in the SRM server

5.6.7. SEE ALSO

Ls, LsDetails, Rm, Cp, Mkdir, Rmdir

5.7. Rmdir

5.7.1. NAME

Rmdir - removes a local empty directory.

5.7.2. Functional Description

Rmdir removes an empty directory in a local SRM space.

5.7.3. Parameter Description

In	Out
User ID	Request Status {
Authorization ID	Status information
Storage System Info	}
SURL	
Flag for Recursive Removal	

Input parameters

- *User ID* (required)
  User authentication identifier.
- *Authorization nID*
  User authorization information. The *Authorization ID* may be NULL.
- *Storage System Info*
  String containining information specific to the underlying storage system. The *Storage System Info* may be NULL.
- *SURL* (required)
  SURL to remove as a directory.
- *Flag for Recursive Removal*
  Boolean indicator to remove as a directory hierarchy if *SURL* represents a multiple levels of directories.

Output parameters

- *Request Status* (required)
  Output reporting the success or failure of the request. Textual description of explanation for what happened to the request, specially in case of any failures, would help client diagnose the case. *Status information* is required and it is recommended to return as an enumerated codes separated as status code and error code, so that when client is a non-human program, response to the status information can be pre-programmed.

5.7.4. NOTES on the Behavior

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

5.7.5. Status information returned upon request execution

On successful request, the *status information* returns SRM_SUCCESS.
On failure, the *status information* returns SRM_FAILURE and the detailed failure information along with the textual explanation in a human readable form would help the client diagnose the failed request.
When detailed failure can be set to one of the followings:

SRM_SUCCESS

- All requests are successfully completed. *SURL* is removed.

SRM_AUTHENTICATION_FAILURE
- SRM server failed to authenticate the client

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

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

SRM_NON_EMPTY_DIRECTORY
- *SURL* is not empty

SRM_INTERNAL_ERROR
- SRM has an internal transient error, and client may try again.

SRM_FAILURE
- any other request failure. *Explanation* needs to be filled for details.

SRM_NOT_SUPPORTED
- *function* is not supported in the SRM server
- input parameter *Flag for Recursive Removal* is not supported in the SRM server

5.7.6. SEE ALSO
Ls, LsDetails, Rm, Mkdir

6. Advanced feature set 4 : Authorization Functions

summary:

CheckPermission
GetPermission
SetPermission

details:

6.1. CheckPermission

6.1.1. NAME

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

6.1.2. Functional Description

CheckPermission is used to check the client permissions on the SURLs. It only checks for the requesting client for authorization on the *SURLs* in the local storage.

6.1.3. Parameter Description

In	Out
User ID	Request Status {
Authorization ID	Status information
Storage System Info	}
SURLs	Permission Infos {
	SURL,
	Status information
	Client Permission Mode
	}

Input parameters

- *User ID* (required)
  User authentication identifier.
- *Authorization ID*
  User authorization information. The *Authorization ID* may be NULL.
- *Storage System Info*
  Information specific to the underlying storage system. The *Storage System Info* may be NULL.
- *SURLs* (required)
  SURLs to check their permission.

Output parameters

- *Request Status* (required)

Output reporting the success or failure of the request. Textual description of explanation for what happened to the request, specially in case of any failures, would help client diagnose the case. *Status information* is required and it is recommended to return as an enumerated codes separated as status code and error code, so that when client is a non-human program, response to the status information can be pre-programmed.

- *Permission Infos*

  Output parameter reporting the permission of each file in the request. *Permission Infos* may be empty and NULL. If returned to the client, *SURL* and its *Status information* are required to return.

  - *SURL* (required)

    SURL that client has requested to check the permission.
  - *Client Permission Mode*

    Client's permission information on the *SURL*.

6.1.4. NOTES on the Behavior

a) SRM will check files in its local online and nearline storage.

6.1.5. Status information returned upon request execution

On successful request, the *status information* returns SRM_SUCCESS.
On failure, the *status information* returns SRM_FAILURE and the detailed failure information along with the textual explanation in a human readable form would help the client diagnose the failed request. If only partial files were successful, the *status information* returns SRM_PARTIAL_SUCCESS, and the *Permission Infos* should explain on those failed files.
When detailed failure can be set to one of the followings:

For request level status,

SRM_SUCCESS
- All requests are successfully completed. Permissions on *SURLs* are checked and returned.

SRM_PARTIAL_SUCCESS
- All requests are completed. Permissions of some SURLs are successfully checked and returned, but some permission of some *SURLs* are failed to be checked. Details are on the files status.

SRM_AUTHENTICATION_FAILURE
- SRM server failed to authenticate the client

SRM_AUTHORIZATION_FAILURE
- client is not authorized to request permission information

SRM_INVALID_REQUEST
- *SURLs* are empty.

SRM_INTERNAL_ERROR
- SRM has an internal transient error, and client may try again.

SRM_FAILURE
- All files requests are failed.
- any other request failure. *Explanation* needs to be filled for details.

SRM_NOT_SUPPORTED
- *function* is not supported in the SRM server

For file level status,

SRM_SUCCESS

- successful request completion for the *SURL*. Permissions on *SURL* are checked and returned.

SRM_INVALID_PATH

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

SRM_AUTHORIZATION_FAILURE

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

SRM_FAILURE

- any other request failure. *Explanation* needs to be filled for details.

6.1.6. SEE ALSO

Ls, LsDetails, GetPermission

6.2. GetPermission

6.2.1. NAME

GetPermission - is used to get the permission information on the SURLs.

6.2.2. Functional Description

GetPermission is used to get the permissions on the SURLs. It only checks for authorization on the *SURLs* in the local storage.

6.2.3. Parameter Description

In	**Out**

User ID
Authorization ID
Storage System Info
SURLs
	Request Status {
	Status information
	}
	Permission Infos {
	SURL,
	Status information
	Owner Permission
	User Permissions
	Group Permissions
	Other Permission
	}

Input parameters

- *User ID* (required)
  User authentication identifier.
- *Authorization ID*
  User authorization information. The *Authorization ID* may be NULL.
- *Storage System Info*

Information specific to the underlying storage system. The *Storage System Info* may be NULL.

- *SURLs* (required)
  SURLs to get their permission information

Output parameters

- *Request Status* (required)
  Output reporting the success or failure of the request. Textual description of explanation for what happened to the request, specially in case of any failures, would help client diagnose the case. *Status information* is required and it is recommended to return as an enumerated codes separated as status code and error code, so that when client is a non-human program, response to the status information can be pre-programmed.
- *Permission Infos*
  Output parameter reporting the permission of each file in the request. *Permission Infos* may be empty and NULL. If returned to the client, *SURL* and its *Status information* are required to return.
  - *SURL* (required)
    SURL that client has requested to check the permission.
  - *Owner Permission*
    Owner permission information on the *SURL*. It may include Owner ID and Owner Permission Mode.
  - *User Permissions*
    User permission information on the *SURL*. It may include User IDs and their Permission Modes.
  - *Group Permissions*
    Group permission information on the *SURL*. It may include Group IDs and their Permission Modes.
  - *Other Permission*
    Other permission information on the *SURL*. It may include Other Permission Mode.

6.2.4. NOTES on the Behavior

a) SRM will check files in its local online and nearline storage.

6.2.5. Status information returned upon request execution

On successful request, the *status information* returns SRM_SUCCESS.
On failure, the *status information* returns SRM_FAILURE and the detailed failure information along with the textual explanation in a human readable form would help the client diagnose the failed request. If only partial files were successful, the *status information* returns SRM_PARTIAL_SUCCESS, and the *Permission Infos* should explain on those failed files.
When detailed failure can be set to one of the followings:

For request level status,

SRM_SUCCESS

- All requests are successfully completed. Permissions on *SURLs* are returned.

SRM_PARTIAL_SUCCESS
- All requests are completed. Permissions of some SURLs are successfully returned, but some permission of some *SURLs* are failed to be returned. Details are on the files status.

SRM_AUTHENTICATION_FAILURE
- SRM server failed to authenticate the client

SRM_AUTHORIZATION_FAILURE
- client is not authorized to request permission information

SRM_INVALID_REQUEST
- *SURLs* are empty.

SRM_INTERNAL_ERROR
- SRM has an internal transient error, and client may try again.

SRM_FAILURE
- All files requests are failed.
- any other request failure. *Explanation* needs to be filled for details.

SRM_NOT_SUPPORTED
- *function* is not supported in the SRM server

For file level status,

SRM_SUCCESS
- successful request completion for the *SURL*. Permissions on *SURL* are returned.

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

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

SRM_FAILURE
- any other request failure. *Explanation* needs to be filled for details.

6.2.6. SEE ALSO
Ls, LsDetails, CheckPermission

6.3. SetPermission

6.3.1. NAME
SetPermission - is to set permission on local SURLs.

6.3.2. Functional Description
SetPermission is to set permission on local SURLs.

6.3.3. Parameter Description

In	Out

User ID
Authorization ID
Storage System Info
SURLs
Permission Type
Permission Target
Permission Mode
Target IDs
Flag for Recursive
	Request Status {
	Status information
	}

Permission Target := OWNER |
USER |
GROUP |
OTHERS

Input parameters

- *User ID* (required)
  User authentication identifier.
- *Authorization ID*
  User authorization information. The *Authorization ID* may be NULL.
- *Storage System Info*
  String containining information specific to the underlying storage system. The *Storage System Info* may be NULL.
- *SURLs* (required)
  SURLs to set the permissions.
- *Permission Type* (required)
  Permission type information. It is either *ADD, REMOVE,* or *CHANGE*.
- *Permission Target*
  Permission target information. It is either *OWNER, USER, GROUP* or *OTHERS*.
- *Permission Mode*
  Permission mode information.
- *Target IDs*
  IDs for the updated permissions.
- *Flag for Recursive*
  Flag for Recursive indicates if the permission updates should be performed recursively on all files under the sub-directories.

Output parameters

- *Request Status* (required)
  Output reporting the success or failure of the request. Textual description of explanation for what happened to the request, specially in case of any failures, would help client diagnose the case. *Status information* is required and it is recommended to return as an enumerated codes separated as status code and error code, so that when client is a non-human program, response to the status information can be pre-programmed.

6.3.4. NOTES on the Behavior

a) *Permission Target* refers to who the permission setting is for.
b) When *Permission Target* is either USER or GROUP, *Target ID* is needed.
c) *Permission Mode* is similar to Unix permission modes.
d) User permissions are supported for dynamic user-level permission assignment similar to Access Control Lists (ACLs).
e) Permissions can be assigned to set of users and sets of groups, but only a single owner.
f) SRMs do not provide any group operations (setup, modify, remove, etc.). roups are assumed to be set up separately, before *SetPermission* is used. The owner must be a member of the group.
g) If *Permission Type* is ADD or CHANGE, and *Permission Mode* is null, then it must be assumed that *Permission Mode* is READ only.
h) If *Permission Type* is REMOVE, then the *Permission Mode* is ignored.
i) If *Permission Type* is CHANGE, but it is being applied to a [user|group] which currently does not have permissions set up for it, then the request works as ADD. It follows the setfacl: Adds one or more new ACL entries to the file, and/or modifies one or more existing ACL entries on the file. If an entry already exists for a specified uid or gid, the specified permissions will replace the current permissions. If an entry does not exist for the specified uid or gid, an entry will be created.

6.3.5. 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.

6.3.6. Status information returned upon request execution

On successful request, the *status information* returns SRM_SUCCESS.
On failure, the *status information* returns SRM_FAILURE and the detailed failure information along with the textual explanation in a human readable form would help the client diagnose the failed request. If only partial files were successful, the *status information* returns SRM_PARTIAL_SUCCESS, and the *SURL Statuses* should explain on those failed files.
When detailed failure can be set to one of the followings:

For request level status,

SRM_SUCCESS
- successful request completion. *SURL* has a new permission.

SRM_AUTHENTICATION_FAILURE
- SRM server failed to authenticate the client

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

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

SRM_INVALID_REQUEST
- Permissions are provided incorrectly

SRM_INTERNAL_ERROR

- SRM has an internal transient error, and client may try again.

SRM_FAILURE

- any other request failure. *Explanation* needs to be filled for details.

SRM_NOT_SUPPORTED

- *function* is not supported in the SRM server
- any input parameter is not supported in the SRM server
- a particular type of an input parameter is not supported in the SRM server

6.3.7. SEE ALSO

GetPermission, CheckPermission

7. Advanced feature set 5 : Request Administration Functions

summary:

ResumeRequest
SuspendRequest

details:

7.1. ResumeRequest

7.1.1. NAME

ResumeRequest - is to resume previously suspended requests.

7.1.2. Functional Description

ResumeRequest is to resume previously suspended requests.

7.1.3. Parameter Description

In	Out
User ID	Request Status {
Authorization ID	Status information
Request Token	}

Input parameters

- *User ID* (required)
  User authentication identifier.
- *Authorization ID*
  User authorization information. The *Authorization ID* may be NULL.
- *Request Token* (required)
  A request handle associated with the previously submitted request to resume its activities. The *Request Token* was returned by the function initiating the request (e.g. *PrepareToGet*).

Output parameters

- *Request Status* (required)
  Output reporting the success or failure of the request. Textual description of explanation for what happened to the request, specially in case of any failures, would help client diagnose the case. *Status information* is required and it is recommended to return as an enumerated codes separated as status code and error code, so that when client is a non-human program, response to the status information can be pre-programmed.

7.1.4. NOTES on the Behavior

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

7.1.5. Status information returned upon request execution

On successful request, the *status information* returns SRM_SUCCESS.
On failure, the *status information* returns SRM_FAILURE and the detailed failure information along with the textual explanation in a human readable form would help the client diagnose the failed request.
When detailed failure can be set to one of the followings:

SRM_SUCCESS
- successful request completion. Request is resumed successfully.

SRM_AUTHENTICATION_FAILURE
- SRM server failed to authenticate the client

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

SRM_INVALID_REQUEST
- *Request Token* is empty.
- *Request Token* does not refer to an existing known request in the SRM server.

SRM_INTERNAL_ERROR
- SRM has an internal transient error, and client may try again.

SRM_FAILURE
- any other request failure. *Explanation* needs to be filled for details.

SRM_NOT_SUPPORTED
- *function* is not supported in the SRM server

7.1.6. SEE ALSO

GetRequestTokens, SuspendRequest

7.2. SuspendRequest

7.2.1. NAME

SuspendedRequest - is to suspend a previously submitted active request.

7.2.2. Functional Description

SuspendedRequest - is to suspend a previously submitted active request.

7.2.3. Parameter Description

In	Out
User ID	Request Status {
Authorization ID	Status information
Request Token	}

Input parameters

- *User ID* (required)
  User authentication identifier.
- *Authorization ID*
  User authorization information. The *Authorization ID* may be NULL.
- *Request Token* (required)
  A request handle associated with the previously submitted request to suspend its activities. The *RquestToken* was returned by the function initiating the request (e.g. *PrepareToGet*).

Output parameters

- *Request Status* (required)
  Output reporting the success or failure of the request. Textual description of explanation for what happened to the request, specially in case of any failures, would help client diagnose the case. *Status information* is required and it is recommended to return as an enumerated codes separated as status code and error code, so that when client is a non-human program, response to the status information can be pre-programmed.

7.2.4. NOTES on the Behavior

a) Suspend all files in the request until *ResumeRequest* 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 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) *AbortRequest* may be performed to end the suspended request.

7.2.5. Status information returned upon request execution

On successful request, the *status information* returns SRM_SUCCESS.
On failure, the *status information* returns SRM_FAILURE and the detailed failure information along with the textual explanation in a human readable form would help the client diagnose the failed request.
When detailed failure can be set to one of the followings:

SRM_SUCCESS
- successful request completion. Request is suspended successfully.

SRM_AUTHENTICATION_FAILURE
- SRM server failed to authenticate the client

SRM_AUTHORIZATION_FAILURE

- client is not authorized to suspend the request specified by the *Request Token*

SRM_INVALID_REQUEST

- *Request Token* is empty.
- *Request Token* does not refer to an existing known request in the SRM server.

SRM_INTERNAL_ERROR

- SRM has an internal transient error, and client may try again.

SRM_FAILURE

- any other request failure. *Explanation* needs to be filled for details.

SRM_NOT_SUPPORTED

- *function* is not supported in the SRM server

7.2.6. SEE ALSO

GetRequestTokens, ResumeRequest

8. Appendix

8.1. Appendix A : Status Code Specification

Note:

- Status codes represent errors, warnings and status.
- For each function, status codes are defined with basic meanings for the function. Only those status codes are valid for the function. Specific cases are not stated for each status code.
- If other status codes need to be defined for a specific function, send an email to the collaboration to discuss the usage

8.2. SRM WSDL discovery method

May 1, 2003

A) SURL format:
srm://host[:port]**/**[soap_end_point_path**?SFN=**]site_file_name

where […] means optional, and letters in bold are fixed.

We note if the SURL contains the soap_end_point_path, then it is not possible to change the soap endpoint without changing all the previously published SURLs.

Example SURLs:

Without soap_end_point_path:
srm://dm.lbl.gov:4001/ABC/file_x

with soap_end_point_path:
srm://dm.lbl.gov:4001/srm_servlet?SFN=ABC/file_x

B) Given that soap-end-point-path clause is provided, then the soap endpoint is:
https://host[:port]/soap_end_point_path

C) If port is missing, the default port assumed is 8443, which is the port for https with GSI.

The discussion below assumes no endpoint in the SURL, and shows how the soap endpoints and wsdl can be found given an SURL

Issues:

1. We wish to have a way of finding the SRM WSDL for multiple versions from the SURL.

2. We wish to support clients that know what SRM version they want to use. For example, a client that uses version 1.1, should be able to got the WSDL and/or the SOAP endpoint for it directly.

3. We wish to have a default where an SRM version number is not mentioned. The version returned in this case is whatever the SRM currently supports, or if multiple versions are supported, the SRM chooses what to return.

4. We wish to allow a file accessed by a previous SRM version to be accessed by a future SRM version without having to change the SURL. Furthermore, if the file can be accessed by either version simultaneously (that depend on the SRM implementation) that should be possible too.

5. We wish to have a way for a client to find out which version the SRM supports and the endpoint without having to read the WSDL. This is necessary in a changing world, where new version can be introduced.

6. We wish to have a client that can use multiple SRM versions to find out which SRM version is supported by the SRM. This is probably the most realistic scenario, since we cannot expect all SRMs to support the same version at any one time.

7. We wish to have a client find out which SRM versions are supported for accessing a particular file, in case that files can be accessed by multiple SRM versions simultaneously. This is related to point 3 above.

This is a long wish list, but the proposed solution is simple. We assume that the WSDL will contain the version number. First, we propose that every SRM WSDL starts with:
SRM version number--> (e.g. <!--SRM version 2.1.3-->)

Now, the solution is as follows:

Give an SURL: srm://host[:port]/path/file (e.g. srm://dm.lbl.gov:4001/ABC/file_x)
The following can be derived:

Case 1)

For clients that know what SRM versions they want to use:
https://host:port/srm/srm.version.wsdl
https://host:port/srm/srm.version.endpoint

For example, given the SURL above, and the client uses version 1.1, you derive:
https://dm.lbl.gov:4001/srm/srm.1.1.wsdl
https://dm.lbl.gov:4001/srm/srm.1.1.endpoint

Note: the endpoint returned can be any URI, e.g.: https://gizmo.lbl.gov:10001/srm/v1.0 or: https://dm.lbl.gov:12345/servlet/srm.1.1)

Case 2)

For clients that don't know the version, and want to use the default:
https://host:port/srm/srm.wsdl
https://host:port/srm/srm.endpoint

For the example above:
https://dm.lbl.gov:4001/srm/srm.wsdl
https://dm.lbl.gov:4001/srm/srm.endpoint

Case 3)

For clients that want to find out the SRM version and endpoint without getting the entire WSDL:
https://host:port/srm/srm.info

The srm.info file will contain:
<!--SRM version number-- --srmEndpoint-->
For example:
<!--SRM version 2.1.3-- -- https://gizmo.lbl.gov:10001/srm-->

Case 4)

For servers that support multiple srm version accessing the SAME file:
The same format as above repeating for each srm version.
For example:
<!--SRM version 1.1-- -- https://sdm.lbl.gov:5005/srm-->
<!--SRM version 2.1.3-- -- https://gizmo.lbl.gov:10001/srm-->

To summarize, the following is what should be supported for WSDL and endpoint discovery:

Given an SURL:
srm://host[:port]**/**site_file_name

The following can be derived:

a) https://host[:port]/srm/srm[.version].wsdl
b) https://host[:port]/srm/srm[.version].endpoint
c) https://host[:port]/srm/srm.info
Where the content have the format repeated as many time as there are supported versions:
<!--SRM version number-- --srmEndpoint-->

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