The Storage Resource Manager
Web Services
Operational Interface Specification

Version 2.2

8 June 2006

Collaboration Web: http://sdm.lbl.gov/srm-wg
Document Location: http://sdm.lbl.gov/srm-wg/doc/SRM.v2.2.ws.op.pdf

Editors:

Alex Sim	Lawrence Berkeley National Laboratory

Contributors:

Timur Perelmutov Don Petravick	Fermi National Accelerator Laboratory (FNAL), USA
Junmin Gu Arie Shoshani	Lawrence Berkeley National Laboratory (LBNL), USA
Olof Barring Jean-Philippe Baud Shaun De Witt	LHC Computing Project (LCG, CERN), Switzerland
Jens Jensen Owen Synge	Rutherford Appleton Laboratory (RAL), England
Michael Haddox-Schatz Bryan Hess Andy Kowalski Chip Watson	Thomas Jefferson National Accelerator Facility (TJNAF), USA

Copyright Notice

STATE OR REFLECT THOSE OF THE UNITED STATES GOVERNMENT OR ANY AGENCY THEREOF, OR THE ENTITY BY WHICH AN AUTHOR MAY BE EMPLOYED.

This paper preparation has been partially supported by the Office of Energy Research, Office of Computational and Technology Research, Division of Mathematical, Information, and Computational Sciences, of the U.S. Department of Energy under Contract No. DE-AC03-76SF00098.

Table of Contents

Introduction

This document contains the interface specification of SRM 2.2. It incorporates the functionality of SRM 2.0 and SRM 2.1, but is much expanded to include additional functionality, especially in the area of dynamic storage space reservation and directory functionality in client-acquired storage spaces.

This document reflects the discussions and conclusions of a 2-day meeting in May 2006, as well as email correspondence and conference calls. The purpose of this activity is to further define the functionality and standardize the interface of Storage Resource Managers (SRMs) – a Grid middleware component.

The document is organized in four sections. The first, called "Defined Structures" contain all the type definitions used to define the functions (or methods). The next 5 sections contain the specification of "Space Management Functions", "Permission Functions", "Directory Functions", "Data Transfer Functions" and "Discovery Functions". All the "Discovery Functions" are newly added functions.

It is advisable to read the document SRM.v2.2.changes.doc posted at http://sdm.lbl.gov/srm-wg before reading this specification.

Each function from the functional interface specification is mapped into a separate WS portType request-response operation. The request-response operation is message exchange between the client and the endpoint, in which the request message is received by the endpoint and the endpoint sends back the response message. In the WSDL1.1 the message either an enumeration of its parts, or is of some complex type. If message is an enumeration, then each part is required and we can not express their options. If message is an instance of a complex type, then all the flexibility of the XML Schema can be applied to the definition of this type. So, express the options of some of the SRM function arguments in functional interface specification, we use the approach of defining a separate type for each request and the response message. Unfortunately, some wsdl toolkit (at least it is true in case of Apache Axis and gSoap), this leads to the generation of the stubs with one input argument of request message type and one output argument of response message type. These are not limitations of the WSDL, but of the particular wsdl stab generation toolkits.

For notes and comments for each function, please refer to the functional specification (see http://sdm.lbl.gov/srm-wg/doc/SRM.spec.v2.2.html).

Understandings and Agreements

1. By “https” we mean http protocol with GSI authentication. It may be represented as “httpg”. At this time, any implementation of http with GSI authentication could be used. It is advisable that the implementation is compatible with Globus Toolkit 3.2 or later versions.
2. We use GSI proxy from the underlying https protocol to authenticate the caller.
3. Primitive types used below are consistent with XML build-in schema types**:** i.e.
    - *long* is 64bit: **(+/-)** 9223372036854775807
    - *int* is 32 bit**: (+/-)** 2147483647
    - *short* is 16 bit**: (+/-)** 32767
    - *unsignedLong* ranges (inclusive): 0 to18446744073709551615
    - *unsignedInt* ranges (inclusive): 0 to 4294967295
    - *unsignedShort* ranges (inclusive): 0 to 65535
4. The definition of the type “anyURI” 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]".
5. In “localSURLInfo”, we mean local to the SRM that is processing the request.
6. storageSystemInfo is added in the arguments of functions srmPrepareToGet() srmPrepareToPut() and srmCopy(). This is to simplify the case when all files sent to the request share the same storageSystemInfo. If storageSystemInfo is provided at the request level and the file level, SRM will use the one provided at the file level.
7. authorizationID : from the SASL RFC 2222
   During the authentication protocol exchange, the mechanism performs authentication, transmits an authorization identity (frequently known as a userid) 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.
8. For SOAP inter-operability, we recommend Apache Axis for Java (see http://ws.apache.org/axis/) or gSOAP for C/C++ from FSU (see http://www.cs.fsu.edu/~engelen/soap.html).
9. Regarding file sharing by the SRM, it is a local implementation decision. An SRM can choose to share files by proving multiple users access to the same physical file, or by copying a file into another user’s space. Either way, if an SRM chooses to share a file (that is, to avoid reading a file over again from the source site) the SRM should check with the source site whether the user has a read/write permission. Only if permission is granted, the file can be shared.

10. The word “pinning” is limited to the “copies” or “states” of SURLs and the Transfer URLs (TURLs).
11. 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.

1. Common Types

We define the SRM WSDL namespace as following:

targetNamespace="http://srm.lbl.gov/StorageResourceManager"
xmlns:impl="http://srm.lbl.gov/StorageResourceManager"
wsdl:portType name="ISRM"
wsdl:binding name="srmSoapBinding" type="impl:ISRM"
wsdlsoap:binding style="rpc" transport="http://schemas.xmlsoap.org/soap/http"
wsdl:service name="SRMService"
wsdl:port name="srm" binding="impl:srmSoapBinding"

Notation:

- Underlined attributes are *REQUIRED and non-nillable*.
- Brackets [] are to show an arrayType.
- Min and Max are to show the number of occurrences of elements in a sequence. This can be interpreted as nillable.

1.1. Simple Types

name	type	
TFileStorageType	xsd:string	enum {Volatile, Durable, Permanent}
TFileType	xsd:string	enum {File, Directory, Link}
TRetentionPolicy	xsd:string	enum {REPLICA , OUTPUT , CUSTODIAL}
TAccessLatency	xsd:string	enum {ONLINE, NEARLINE}
TPermissionMode	xsd:string	enum {NONE, X, W, WX, R, RX, RW, RWX}
TPermissionType	xsd:string	enum {ADD, REMOVE, CHANGE}
TRequestType	xsd:string	enum {PREPARE_TO_GET, PREPARE_TO_PUT, COPY, BRING_ONLINE, RESERVE_SPACE, UPDATE_SPACE, CHANGE_SPACE_FOR_FILES. LS }
TOverwriteMode	xsd:string	enum {Never, Always, WhenFilesAreDifferent}
TFileLocality	xsd:string	enum { ONLINE, NEARLINE, ONLINE_AND_NEARLINE, LOST, NONE, UNAVAILABLE }

TRetentionPolicy

- Quality of Retention (Storage Class) is a kind of Quality of Service. It refers to the probability that the storage system lose a file. Numeric probabilities are self-assigned.
    - Replica quality has the highest probability of loss, but is appropriate for data that can be replaced because other copies can be accessed in a timely fashion.
    - Output quality is an intermediate level and refers to the data which can be replaced by lengthy or effort-full processes.
    - Custodial quality provides low probability of loss.
- The type will be used to describe retention policy assigned to the files in the storage system, at the moments when the files are written into the desired destination in the storage system. It will be used as a property of space allocated through the space reservation function. Once the retention policy is assigned to a space, the files put in the reserved space will automatically be assigned the retention policy of the space. The assigned retention policy on the file can be found thought the TMetaDataPathDetail structure returned by the srmLs function.

TAccessLatency

- Files may be Online, Nearline or Offline. These terms are used to describe how latency to access a file is improvable. Latency is improved by storage systems replicating a file such that its access latency is online.
    - The ONLINE cache of a storage system is the part of the storage system which provides file with online latencies.
    - ONLINE has the lowest latency possible. No further latency improvements are applied to online files.
    - NEARLINE file can have their latency improved to online latency automatically by staging the file to online cache.
    - For completeness, we also describe OFFLINE here.
    - OFFLINE files need a human to be involved to achieve online latency.
    - For the SRM we only keep ONLINE and NEARLINE.
- The type will be used to describe a space property that access latency can be requested at the time of space reservation. The content of the space, files may have the same or “lesser” access latency as the space.

TFileLocality

- 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.
    - 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.
    - The NEARLINE indicates that the file is located on nearline storage system, and the file may be accessed with nearline latencies.

- The ONLINE_AND_NEARLINE indicates that the file is located on online cache of a storage system as well as on nearline storage system.
- The LOST indicates when the file is lost because of the permanent hardware failure.
- The NONE value shall be used if the file is empty (zero size).
- The UNAVAILABLE indicates that the file is unavailable due to the temporary hardware failure.

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

name	type
TAccessPattern	xsd:string enum { TransferMode, ProcessingMode }
TConnectionType	xsd:string enum { WAN, LAN }
TStatusCode	xsd:string enum { SRM_SUCCESS, SRM_FAILURE, SRM_AUTHENTICATION_FAILURE, SRM_AUTHORIZATION_FAILURE, SRM_INVALID_REQUEST, SRM_INVALID_PATH, SRM_FILE_LIFETIME_EXPIRED, SRM_SPACE_LIFETIME_EXPIRED, SRM_EXCEED_ALLOCATION, SRM_NO_USER_SPACE, SRM_NO_FREE_SPACE, SRM_DUPLICATION_ERROR, SRM_NON_EMPTY_DIRECTORY, SRM_TOO_MANY_RESULTS, SRM_INTERNAL_ERROR, SRM_FATAL_INTERNAL_ERROR, SRM_NOT_SUPPORTED, SRM_REQUEST_QUEUED, SRM_REQUEST_INPROGRESS, SRM_REQUEST_SUSPENDED, SRM_ABORTED, SRM_RELEASED, SRM_FILE_PINNED, SRM_FILE_IN_CACHE, SRM_SPACE_AVAILABLE, SRM_LOWER_SPACE_GRANTED, SRM_DONE, SRM_PARTIAL_SUCCESS, SRM_REQUEST_TIMED_OUT, SRM_LAST_COPY, SRM_FILE_BUSY, SRM_FILE_LOST, SRM_FILE_UNAVAILABLE,

	SRM_CUSTOM_STATUS }

TAccessPattern

- TAccessPattern 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.

TConnectionType

- TConnectionType 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.

1.2. Complex Types

name	type		Min	Max
TRetentionPolicyInfo	TRetentionPolicy TAccessLatency	retentionPolicy accessLatency	1 0	1 1
TRequestToken	xsd:string		1	1
ArrayOfTRequestToken	TRequestToken []		1	1
TSpaceToken	xsd:string		1	1
ArrayOfTSpaceToken	TSpaceToken []		1	1
TUserID	xsd:string		1	1
TGroupID	xsd:string		1	1
TOwnerPermission	TPermissionMode		1	1
TUserPermission	TUserID TPermissionMode	userID mode	1 1	1 1
ArrayOfTUserPermission	TUserPermission []			
TGroupPermission	TGroupID TPermissionMode	groupID mode	1 1	1 1
ArrayOfTGroupPermission	TGroupPermission []			
TOtherPermission	TPermissionMode		1	1
TCheckSumType	xsd:string		1	1
TCheckSumValue	xsd:string		1	1
TSizeInBytes	xsd:unsignedLong			
ArrayOfTSizeInBytes	TSizeInBytes []			

ArrayOfString	xsd:string []		

TRequestToken

- The TRequestToken assigned by SRM is unique and immutable (non-reusable). For example, if the date:time is part of the requestToken it will be immutable.

TUserPermission

- TUserID may represent the associated client's Distinguished Name (DN) instead of unix style login name. VOMS role may be included.

TGroupPermission

- TGroupID may represent the associated client's Distinguished Name (DN) instead of unix style login name. VOMS role may be included.

TSizeInBytes

- xsd:unsignedLong — Unsigned integer of 64 bits

name	type		Min	Max
TUTCTime	xsd:dateTime		1	1
TLifeTimeInSeconds	xsd:unsignedLong			
TSURL	xsd:anyURI		1	1
TTURL	xsd:anyURI		1	1
TReturnStatus	TStatusCode	statusCode	1	1
	xsd:string	explanation	0	1
TSURLReturnStatus	TSURL	surl	1	1
	TReturnStatus	status	1	1
ArrayOfTSURLReturnStatus	TSURLReturnStatus []			
TMetaDataPathDetail	xsd:string	surl	1	1
	TReturnStatus	status	0	1
	TSizeInBytes	size	0	1
	TOwnerPermission	ownerPermission	0	1
	ArrayOfTUserPermission	arrayOfUserPermissions	0	1
	ArrayOfTGroupPermission	arrayOfGroupPermissions	0	1
	TOtherPermission	otherPermission	0	1
	TUTCTime	createdAtTime	0	1
	TUTCTime	lastModificationTime	0	1
	TUserID	owner	0	1
	TFileStorageType	fileStorageType	0	1
	TRetentionPolicyInfo	retentionPolicyInfo	0	1
	TFileLocality	fileLocality	0	1
	ArrayOfTSpaceToken	arrayOfSpaceTokens	0	1
	TFileType	type	0	1

	TLifeTimeInSeconds	lifetimeAssigned	0	1
	TLifeTimeInSeconds	lifetimeLeft	0	1
	TCheckSumType	checkSumType	0	1
	TCheckSumValue	checkSumValue	0	1
	ArrayOfTMetaDataPathDetail	arrayOfSubPaths	0	1
ArrayOfTMetaDataPathDetail	TMetaDataPathDetail []			
TMetaDataSpace	TSpaceToken	spaceToken	1	1
	TReturnStatus	status	0	1
	TRetentionPolicyInfo	retentionPolicyInfo	0	1
	TUserID	owner	0	1
	TSizeInBytes	totalSize	0	1
	TSizeInBytes	guaranteedSize	0	1
	TSizeInBytes	unusedSize	0	1
	TLifeTimeInSeconds	lifetimeAssigned	0	1
	TLifeTimeInSeconds	lifetimeLeft	0	1
ArrayOfTMetaDataSpace	TMetaDataSpace[]		1	1

TUTCTime

- Formerly TGMTTime in v2.1
- date and time in Coordinated Universal Time (UTC, formerly GMT) with no local time extention.
- Format is same as in XML dateTime type, except no local time extension is allowed. E.g. `1999-05-31T13:20:00` is ok (for 1999 May 31st, 13:20PM, UTC) but `1999-05-31T13:20:00-5:00` is not.

TLifeTimeInSeconds

- xsd:unsignedLong — Unsigned integer of 64 bits
- "0" (zero) will indicate the "infinite" lifetime.

TMetaDataPathDetail

- The *TMetaDataPathDetail* describes the properties of a file. It is used as an output parameter in *srmLs*.
- *retentionPolicyInfo* indicates the assigned retention policy.
- *fileLocality* indicates where the file is located currently in the system.
- *arrayOfSpaceTokens* as an array of *TSpaceToken* 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.

TMetaDataSpace

- *TMetaDataSpace* is used to describe properties of a space, and is used as an output parameter in *srmGetSpaceMetaData*.

- *retentionPolicyInfo* indicates the information about retention policy and access latency that the space is assigned. *retentionPolicyInfo* is requested and assigned at the time of space reservation through *srmReserveSpace* and *srmStatusOfReserveSpaceRequest*.
- *TMetaDataSpace* 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.

name	type		Min	Max
TDirOption	xsd:boolean	isSourceADirectory	1	1
	xsd:boolean	allLevelRecursive	0	1
	xsd:int	numOfLevels	0	1
TExtraInfo	xsd:string	key	1	1
	xsd:string	value	0	1
ArrayOfTExtraInfo	TExtraInfo []			
ArrayOfTSURL	TSURL []			
TSURLInfo	TSURL	SURL	1	1
	ArrayOfTExtraInfo	storageSystemInfo	0	1
ArrayOfTSURLInfo	TSURLInfo []			
TTransferParameters	TAccessPattern	accessPattern	0	1
	TConnectionType	connectionType,	0	1
	ArrayOfString	arrayOfClientNetworks	0	1
	ArrayOfString	arrayOfTransferProtocols	1	1
TGetFileRequest	TSURL	sourceSURL	1	1
	TDirOption	dirOption	0	1
ArrayOfTGetFileRequest	TGetFileRequest []			
TPutFileRequest	TSURL	targetSURL	0	1
	TSizeInBytes	expectedFileSize	0	1
ArrayOfTPutFileRequest	TPutFileRequest []			
TCopyFileRequest	TSURLInfo	sourceSURLInfo	1	1
	TSURLInfo	targetSURLInfo	1	1
	TDirOption	dirOption	0	1
ArrayOfTCopyFileRequest	TCopyFileRequest []			

TExtraInfo

- TExtraInfo is used where additional information is needed, such as for additional information for transfer protocols of TURLs in srmPing, srmGetTransferProtocols, srmStatusOfGetRequest, and srmStatusOfPutRequest. For example, when it is used for additional information for transfer protocols, the keys may specify access speed, available number of parallelism, and other transfer protocol properties.
- It is also used where additional information to the underlying storage system is needed, such as for additional information, but not limited to, for storage device, storage login ID, storage login authorization. Formerly, it was TStorageSystemInfo.

TTransferParameters

- TTransferParameters is used where arrayOfTransferProtocols was used previously in SRM v2.1.
- TGetFileRequest includes TAccessPattern which may conflict with the characteristics of the online disk of the target space associated with target space token if provided. In this case, TAccessPattern must be ignored
- File transfer protocols are specified in a preferred order on all SRM transfer functions.
- arrayOfClientNetworks is a hint of the client IPs that SRM/dCache can use for optimization of its internal storage systems based on the client's accessible IP addresses.

TPutFileRequest

- If the optional targetSURL is provided, then the reference SURL is generated by the SRM. Specific SRM implementation may require targetSURL as an input parameter.

name	type		Min	Max
TGetRequestFileStatus	TSURL	sourceSURL	1	1
	TReturnStatus	status	1	1
	TSizeInBytes	fileSize	0	1
	TLifeTimeInSeconds	estimatedWaitTime	0	1
	TLifeTimeInSeconds	remainingPinTime	0	1
	TTURL	transferURL	0	1
	ArrayOfTExtraInfo	transferProtocolInfo	0	1
ArrayOfTGetRequestFileStatus	TGetRequestFileStatus []			
TBringOnlineRequestFileStatus	TSURL	sourceSURL	1	1
	TReturnStatus	status	1	1
	TSizeInBytes	fileSize	0	1
	TLifeTimeInSeconds	estimatedWaitTime	0	1
	TLifeTimeInSeconds	remainingPinTime	0	1
ArrayOfTBringOnlineRequestFileStatus	TBringOnlineRequestFileStatus []			
TPutRequestFileStatus	TSURL	SURL	1	1
	TReturnStatus	status	1	1
	TSizeInBytes	fileSize	0	1
	TLifeTimeInSeconds	estimatedWaitTime	0	1
	TLifeTimeInSeconds	remainingPinLifetime	0	1
	TLifeTimeInSeconds	remainingFileLifetime	0	1
	TTURL	transferURL	0	1
	ArrayOfTExtraInfo	transferProtocolInfo	0	1
ArrayOfTPutRequestFileStatus	TPutRequestFileStatus []			
TCopyRequestFileStatus	TSURL	sourceSURL	1	1
	TSURL	targetSURL	1	1

	TReturnStatus	status	1	1
	TSizeInBytes	fileSize	0	1
	TLifeTimeInSeconds	estimatedWaitTime	0	1
	TLifeTimeInSeconds	remainingFileLifetime	0	1
ArrayOfTCopyRequestFileStatus	TCopyRequestFileStatus []			
TRequestStat	TStatusCode	statusCode	1	1
	int	numOfFiles	1	1
ArrayOfTRequestStat	TRequestStat []			
TRequestSummary	TRequestToken	requestToken	1	1
	TReturnStatus	status	1	1
	TRequestType	requestType	0	1
	xsd:int	totalNumFilesInRequest	0	1
	ArrayOfTRequestStat	arrayOfRequestStats	0	1
	xsd:boolean	isSuspended	0	1
ArrayOfTRequestSummary	TRequestSummary []			
TSURLPermissionReturn	TSURL	surl	1	1
	TReturnStatus	status	1	1
	TPermissionType	permission	0	1
ArrayOfTSURLPermissionReturn	TSURLPermissionReturn []			
TRequestTokenReturn	TRequestToken	requestToken	1	1
	TUTCTime	createdAtTime	0	1
ArrayOfTRequestTokenReturn	TRequestTokenReturn []			
TSupportedTransferProtocol	xsd:string	transferProtocol	1	1
	ArrayOfTExtraInfo	attributes	0	1
ArrayOfTSupportedTransferProtocol	TSupportedTransferProtocol []			

TRequestSummary

- Int totalNumFilesInRequest
  Output parameter reporting the total number of files in the request
- Boolean isSuspended
  Output parameter reporting if the request has been suspended or not

TSupportedTransferProtocol

- *transferProtocol* (required): Supported transfer protocol. For example, gsiftp, http.
- *attributes*: Informational hints for the paired transfer protocol, such how many number of parallel streams can be used, desired buffer size, etc.

2. Space Management Functions – Message Types and Operations

summary:

2.1. srmReserveSpace

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.

Input	**srmReserveSpaceRequest**
Output	**srmReserveSpaceResponse**

name	type		Min	Max
srmReserveSpaceRequest	TUserID	authorizationID	0	1
	xsd:string	userSpaceTokenDescription	0	1
	TRetentionPolicyInfo	retentionPolicyInfo	1	1
	TSizeInBytes	desiredSizeOfTotalSpace	0	1
	TSizeInBytes	desiredSizeOfGuaranteedSpace	1	1
	TLifeTimeInSeconds	desiredLifetimeOfReservedSpace	0	1
	ArrayOfTSizeInBytes	arrayOfExpectedFileSizes	0	1
	ArrayOfTExtraInfo	storageSystemInfo	0	1
	TTransferParameters	transferParametersHint	0	1
srmReserveSpaceResponse	TReturnStatus	returnStatus	1	1
	TRequestToken	requestToken	0	1
	TLifeTimeInSeconds	estimatedProcessingTime	0	1
	TRetentionPolicyInfo	retentionPolicyInfo	0	1
	TSizeInBytes	sizeOfTotalReservedSpace	0	1
	TSizeInBytes	sizeOfGuaranteedReservedSpace	0	1
	TLifeTimeInSeconds	lifetimeOfReservedSpace	0	1
	TSpaceToken,	spaceToken	0	1

2.1.1. Notes on the Behavior

a) If the input parameter *desiredLifetimeOfReservedSpace* is not provided, the lifetime of the reserved space is set to "infinite" by default.
b) If the input parameter *retentionPolicyInfo* cannot be satisfied by the SRM server, SRM_INVALID_REQUEST must be returned.
c) 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.
d) When asynchronous space reservation is necessary, the returned status code should be SRM_REQUEST_QUEUED.
e) Input parameter *arrayOfExpectedFileSize* 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.
f) Optional input parameter *storageSystemInfo* is needed in case the underlying storage system requires additional security information.
g) SRM may return its default space size and lifetime if not requested by the client. SRM may return SRM_INVALID_REQUEST if SRM does not support default space sizes.
h) If input parameter *desiredSizeOfTotalSpace* is not specified, the SRM will return its default space size.
i) Output parameter *estimateProcessingTime* is used to indicate the estimation time to complete the space reservation request, when known.
j) Output parameter *sizeOfTotalReservedSpace* is in best effort bases. For guaranteed space size, *sizeOfGuaranteedReservedSpace* should be checked. These two numbers may match, depending on the storage systems.
k) Output parameter *spaceToken* is a reference handle of the reserved space.

2.1.2. Return Status Code

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_GRANGED

- 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 *retentionPolicyInfo* 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 do not conform the SRM server. For example, client requested negative *desiredLifetimeOfReservedSpace* and SRM server cannot honor the number.

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

2.2. srmStatusOfReserveSpaceRequest

This function is used to check the status of the previous request to *srmReserveSpace*, when asynchronous space reservation was necessary with the SRM. Request token must have been provided in response to the *srmReserveSpace*.

Input	**srmStatusOfReserveSpaceRequestRequest**
Output	**srmStatusOfReserveSpaceRequestResponse**

name	type		Min	Max
srmStatusOfReserveSpaceRequestRequest	TUserID	authorizationID	0	1
	TRequestToken	requestToken	1	1
srmStatusOfReserveSpaceRequestResponse	TReturnStatus	returnStatus	1	1
	TLifeTimeInSeconds	estimatedProcessingTime	0	1
	TRetentionPolicyInfo	retentionPolicyInfo	0	1
	TSizeInBytes	sizeOfTotalReservedSpace	0	1

	TSizeInBytes sizeOfGuaranteedReservedSpace	0	1
	TLifeTimeInSeconds lifetimeOfReservedSpace	0	1
	TSpaceToken, spaceToken	0	1

2.2.1. Notes on the Behavior

a) If the space reservation is not completed yet, *estimateProcessingTime* is returned when known. The returned status code in such case should be SRM_REQUEST_QUEUED.
b) See notes for *srmReserveSpace* for descriptions for output parameters.

2.2.2. Return Status Code

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_GRANGED
- 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
- *requestToken* 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

2.3. srmReleaseSpace

srmReleaseSpace() releases an occupied space.

Input	**srmReleaseSpaceRequest**
Output	**srmReleaseSpaceResponse**

name	type		Min	Max
srmReleaseSpaceRequest	TUserID	authorizationID	0	1
	TSpaceToken	spaceToken	1	1
	ArrayOfTExtraInfo	storageSystemInfo	0	1
	xsd:boolean	forceFileRelease	0	1
srmReleaseSpaceResponse	TReturnStatus	returnStatus	1	1

2.3.1. Notes on the Behavior

a) *forceFileRelease* is false by default. This means that the space will not be released if it has files that are still pinned in the space. To release the space regardless of the files it contains and their status *forceFileRelease* must be specified to be true.
b) When space is releasable and *forceFileRelease* is true, all the files in the space are released, even in durable or permanent space.
c) *srmReleaseSpace* may not complete right away because of the lifetime of files in the space. When space is released, the files in that space are treated according to their types: If file storage types are permanent, keep them until further operation such as srmRm is issued by the client. If file storage types are durable, perform necessary actions at the end of their lifetime. If file storage types are volatile, release those files at the end of their lifetime.

2.3.2. Return Status Code

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 *spaceToken*

SRM_INVALID_REQUEST
- *spaceToken* 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

- *forceFileRelease* 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.

2.4. srmUpdateSpace

srmUpdateSpace 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.

Input	**srmUpdateSpaceRequest**
Output	**srmUpdateSpaceResponse**

name	type		Min	Max
srmUpdateSpaceRequest	TUserID	authorizationID	0	1
	TSpaceToken	spaceToken	1	1
	TSizeInBytes	newSizeOfTotalSpaceDesired	0	1
	TSizeInBytes	newSizeOfGuaranteedSpaceDesired	0	1
	TLifeTimeInSeconds	newLifeTime	0	1
	ArrayOfTExtraInfo	storageSystemInfo	0	1
srmUpdateSpaceResponse	TReturnStatus	returnStatus	1	1
	TRequestToken	requestToken	0	1
	TSizeInBytes	sizeOfTotalSpace	0	1
	TSizeInBytes	sizeOfGuaranteedSpace	0	1
	TLifeTimeInSeconds	lifetimeGranted	0	1

2.4.1. Notes on the Behavior

a) If neither size nor lifetime is provided in the input parameters, then the request will be failed.
b) newSize is the new actual size of the space.
c) *newLifetime* is the new lifetime requested regardless of the previous lifetime, and has to be positive. It might 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.

2.4.2. Return Status Code

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_GRANGED
- 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 *spaceToken*

SRM_SPACE_LIFETIME_EXPIRED
- lifetime of the space that is associated with the *spaceToken* is already expired.

SRM_INVALID_REQUEST
- *spaceToken* 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

2.5. srmStatusOfUpdateSpaceRequest

This function is used to check the status of the previous request to *srmUpdateSpace*, when asynchronous space update was necessary with the SRM. Request token must have been provided in response to the *srmUpdateSpace*.

2.5.1. Parameters

In:	TUserID	authorizationID,	
	TRequestToken	requestToken	
Out:	TReturnStatus	returnStatus,	
	TSizeInBytes	sizeOfTotalSpace,	// best effort

TSizeInBytes	sizeOfGuaranteedSpace,
TLifeTimeInSeconds	lifetimeGranted

2.5.2. Notes on the Behavior

a) Output parameters for sew sizes are the new actual sizes of the space.
b) Output parameter, *lifetimeGranted* is the new lifetime granted regardless of the previous lifetime. It might even be shorter than the previous lifetime. It is relative to the calling time.

2.5.3. Return Status Code

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_GRANGED
- 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 *spaceToken*

SRM_SPACE_LIFETIME_EXPIRED
- lifetime of the space that is associated with the *spaceToken* is already expired.

SRM_INVALID_REQUEST
- *spaceToken* 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

2.6. srmGetSpaceMetaData

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 *srmReserveSpace* or *srmStatusOfReserveSpaceRequest*.

Input	**srmGetSpaceMetaDataRequest**
Output	**srmGetSpaceMetaDataResponse**

name	type		Min	Max
srmGetSpaceMetaDataRequest	TUserID	authorizationID	0	1
	ArrayOfTSpaceToken	arrayOfSpaceTokens	1	1
srmGetSpaceMetaDataResponse	TReturnStatus	returnStatus	1	1
	ArrayOfTMetaDataSpace	arrayOfSpaceDetails	0	1

2.6.1. Return Status Code

For request level return 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

- *arrayOfSpaceToken* is 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

For space level return Status,

SRM_SUCCESS

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

SRM_AUTHORIZATION_FAILURE

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

SRM_INVALID_REQUEST

- *spaceToken* 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.

2.7. srmChangeSpaceForFiles

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 *srmReserveSpace*. 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.

Input	**srmChangeSpaceForFilesRequest**
Output	**srmChangeSpaceForFilesResponse**

name	type		Min	Max
srmChangeSpaceForFilesRequest	TUserID	authorizationID	0	1
	ArrayOfTSURLInfo	arrayOfSURLs	1	1
	TSpaceToken	targetSpaceToken	1	1
srmChangeSpaceForFilesResponse	TReturnStatus	returnStatus	1	1
	TRequestToken	requestToken	0	1
	TLifeTimeInSeconds	estimatedProcessingTime	0	1
	ArrayOfTSURLReturnStatus	arrayOfFileStatuses	0	1

2.7.1. 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_SPACE_TRANSITION 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 code should be SRM_REQUEST_QUEUED, and *arrayOfFileStatuses* may not be filled and returned.
e) All files specified in *arrayOfSURLs* will be moved to the space associated with *targetSpaceToken.*
f) When target space token is used, space allocation for a new space token must be done explicitly by the client before using this function.
g) If a directory path is provided, then the effect is recursive for all files in the directory.
h) 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 srmStatusOfChangeSpaceForFilesRequest.

2.7.2. Return Status Code

For request level return status,

SRM_SUCCESS
- All file requests are successfully completed. All *SURLs* have new *targetSpaceToken*.

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

SRM_ REQUEST_QUEUED
- request is submitted and accepted. *requestToken* must be returned.
- The status can be checked by *srmStatusOfChangeSpaceForFilesRequest*.

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.
- *targetSpaceToken* is empty.
- *targetSpaceToken* does not refer to an existing space in the SRM server.
- *targetSpaceToken* refers to a forbidden transition by the SRM implementation.

SRM_SPACE_LIFETIME_EXPIRED
- target space that is associated with *targetSpaceToken* has an expired lifetime.

SRM_EXCEED_ALLOCATION
- target space that is associated with *targetSpaceToken* is not enough to hold *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 return status,

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

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
- *targetSpaceToken* refers to a forbidden transition for the particular SURL by the SRM implementation.

SRM_EXCEED_ALLOCATION
- target space that is associated with *targetSpaceToken* 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.

2.8. srmStatusOfChangeSpaceForFilesRequest

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

Input	**srmStatusOfChangeSpaceForFilesRequestRequest**
Output	**srmStatusOfChangeSpaceForFilesRequestResponse**

name	type		Min	Max
srmStatusOfChangeSpaceForFiles RequestRequest	TUserID	authorizationID	0	1
	TRequestToken	requestToken	1	1
srmStatusOfChangeSpaceForFiles RequestResponse	TReturnStatus	returnStatus	1	1
	TLifeTimeInSeconds	estimatedProcessingTime	0	1
	ArrayOfTSURLReturnStatus	arrayOfFileStatuses	0	1

2.8.1. 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, *estimateProcessingTime* 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 code should be 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 code. Each file should have its own status code.

2.8.2. Return Status Code

For request level return status,

SRM_SUCCESS
- All file requests are successfully completed. All *SURLs* have new *targetSpaceToken*.

SRM_PARTIAL_SUCCESS
- All requests are completed. Some SURL requests have new *targetSpaceToken*, and some SURL requests are failed to have new *targetSpaceToken*. 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
- *requestToken* does not refer to an existing known request in the SRM server.

- *targetSpaceToken* refers to a forbidden transition by the SRM implementation.

SRM_SPACE_LIFETIME_EXPIRED
- target space that is associated with *targetSpaceToken* has an expired lifetime.

SRM_EXCEED_ALLOCATION
- target space that is associated with *targetSpaceToken* is not enough to hold *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 return status,

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

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
- *targetSpaceToken* refers to a forbidden transition for the particular SURL by the SRM implementation.

SRM_EXCEED_ALLOCATION
- target space that is associated with *targetSpaceToken* 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.

2.9. srmExtendFileLifeTimeInSpace

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

Input	**srmExtendFileLifeTimeInSpaceRequest**
Output	**srmExtendFileLifeTimeInSpaceResponse**

name	type		Min	Max
srmExtendFileLifeTimeInSpace Request	TUserID	authorizationID	0	1
	TSpaceToken	spaceToken	1	1
	ArrayOfTSURL	arrayOfSURLs	0	1
	TLifeTimeInSeconds	newLifeTime	0	1
srmExtendFileLifeTimeInSpace Response	TReturnStatus	returnStatus	1	1
	TLifeTimeInSeconds	newTimeExtended	0	1
	ArrayOfTSURLReturnStatus	arrayOfFileStatuses	0	1

2.9.1. Notes on the Behavior

a) When *spaceToken* is provided, the lifetime of the file copy of the SURLs in the space associated with the space token will be extended.
b) *newLifeTime* is relative to the calling time. Lifetime will be set from the calling time for the specified period.
c) The new file lifetime, *newTimeExtended* 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 *newLifeTime* is not specified, the SRM can use its default to assign the *newLifeTime*.

2.9.2. Return Status Code

For request level return 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
- *spaceToken* is empty.
- *spaceToken* 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
- 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 return 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.

2.10. srmPurgeFromSpace

This function is used when removing files from the given space is needed. Difference from *srmReleaseFiles* and *srmAbortFiles* is that *srmPurgeFromSpace* is not associated with a request. This function must not remove the SURLs, but only the "copies" or "states" of the SURLs. *srmRm* must be used to remove SURLs.

Input	**srmPurgeFromSpaceRequest**
Output	**srmPurgeFromSpaceResponse**

name	type	Min	Max

srmPurgeFromSpaceRequest	TUserID	authorizationID	0	1
	ArrayOfTSURLInfo	arrayOfSURLs	1	1
	TSpaceToken	spaceToken	1	1
srmPurgeFromSpaceResponse	TReturnStatus	returnStatus	1	1
	ArrayOfTSURLReturnStatus	arrayOfFileStatuses	0	1

2.10.1. 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. To remove the last copy of the *SURL*, *srmRm* may be used.
b) The method shall only succeed if there are no outstanding pins or requests for the specified files. Otherwise, SRM_FILE_BUSY must be returned.
c) When input parameter *spaceToken* is provided, SRM will remove only the "copies" (or "state") of the *SURLs* associated with the space token.
d) It has the effect of a release on the "copy" (or "state") of the file before being removed.

2.10.2. Return Status Code

For request level return status,

SRM_SUCCESS
- All requests are successfully completed. All *SURLs* are purged from the space specified by the *spaceToken*.

SRM_PARTIAL_SUCCESS
- All requests are completed. Some *SURLs* are successfully purged from the space specified by the *spaceToken*, and some *SURLs* are failed to be purged from the space specified by the *spaceToken*. 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 *spaceToken*

SRM_INVALID_REQUEST
- *arrayOfSURLs* is empty.
- *spaceToken* is empty.
- *spaceToken* 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
- 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 return Status,

SRM_SUCCESS

- successful request completion for the *SURL*. *SURL* is purged from the space specified by the *spaceToken*.

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 *spaceToken*

SRM_FILE_LOST
- the request 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 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. *srmRm* must be used to remove the last copy.

2.11. srmGetSpaceToken

srmGetSpaceToken() returns space tokens for currently allocated spaces.

Input	**srmGetSpaceTokenRequest**
Output	**srmGetSpaceTokenResponse**

name	type		Min	Max
srmGetSpaceTokenRequest	xsd:string	userSpaceTokenDescription	0	1
	TUserID	authorizationID	0	1
srmGetSpaceTokenResponse	TReturnStatus	returnStatus	1	1
	ArrayOfTSpaceToken	arrayOfSpaceTokens	0	1

2.11.1. Notes on the Behavior

a) If *userSpaceTokenDescription* is null, returns all space tokens this user owns.
b) If the user assigned the same name to multiple space reservations, he may get back multiple space tokens.

2.11.2. Return Status Code

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 *spaceTokens* associated with the *userSpaceDescription*

SRM_INVALID_REQUEST

- *userSpaceDescription* 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

3. Permission Functions – Message Types and Operations

summary:

3.1. srmSetPermission

srmSetPermission is to set permission on local SURL.

Input	**srmSetPermissionRequest**
Output	**srmSetPermissionResponse**

Name	type		Min	Max
srmSetPermissionRequest	TUserID	authorizationID	0	1
	TSURLInfo	SURL	1	1
	TPermissionType	permissionType	0	1
	TOwnerPermission	ownerPermission	0	1
	ArrayOfTUserPermission	arrayOfUserPermissions	0	1
	ArrayOfTGroupPermission	arrayOfGroupPermissions	0	1
	TOtherPermission	otherPermission	0	1
srmSetPermissionResponse	TReturnStatus	returnStatus	1	1

3.1.1. Notes on the Behavior

a) Applies to both dir and file.
b) Support for *srmSetPermission* is optional.
c) User permissions are provided in order to support dynamic user-level permission assignment similar to Access Control Lists (ACLs).
d) Permissions can be assigned to set of users and sets of groups, but only a single owner.
e) In this version, SRMs do not provide any group operations (setup, modify, remove, etc.)
f) Groups are assumed to be set up before *srmSetPermission* is used.
g) If *TPermissionType* is ADD or CHANGE, and *TPermissionMode* is null, then it is assumed that *TPermissionMode* is READ only.
h) If *TPermissionType* is REMOVE, then the *TPermissionMode* is ignored

3.1.2. Return Status Code

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_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

3.2. srmCheckPermission

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

Input	**srmCheckPermissionRequest**
Output	**srmCheckPermissionResponse**

Name	type		Min	Max
srmCheckPermissionRequest	ArrayOfTSURLInfo	arrayOfSURLs	1	1
	TUserID	authorizationID	0	1
	xsd:boolean	checkLocalOnly	0	1
srmCheckPermissionResponse	TReturnStatus	returnStatus	1	1
	ArrayOfTSURLPermissionReturn	arrayOfPermissions	0	1

3.2.1. Notes on the Behavior

a) When *checkLocalOnly* is true, then SRM will only check files in its local cache. Otherwise, if a file is not in its local cache, then SRM will go to the SURL to check the user permission.

b) If *checkLocalOnly* is false, SRM can choose to always check the SURL for user permission of each file. It is also ok if SRM choose to check its local cache first, if a file exists and the user has permission, return that permission. Otherwise, check the SURL and return permission.

3.2.2. Return Status Code

For request level return 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
- *arrayOfSURL* is 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
- a particular type of an input parameter (*checkLocalOnly* = false) is not supported in the SRM server

For file level return 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.

4. Directory Functions – Message Types and Operations

summary:

4.1. srmMkdir

srmMkdir creates a directory in a local SRM space.

Input	**srmMkdirRequest**
Output	**srmMkdirResponse**

Name	type		Min	Max
srmMkdirRequest	TUserID	authorizationID	0	1
	TSURLInfo	directoryPath	1	1
srmMkdirResponse	TReturnStatus	returnStatus	1	1

4.1.1. Notes on the Behavior

a) Consistent with unix, recursive creation of directories is not supported.
b) *directoryPath* can include paths, as long as all directory hierarchy exists.

4.1.2. Return Status Code

SRM_SUCCESS
- All requests are successfully completed. *directoryPath* 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 *diretoryPath*

SRM_INVALID_PATH
- *directoryPath* does not refer to a valid path
- component of *directoryPath* does not refer to an existing path

SRM_DUPLICATION_ERROR
- *directoryPath* 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

4.2. srmRmdir

srmRmdir removes an empty directory in a local SRM space.

Input	**srmRmdirRequest**
Output	**srmRmdirRespnose**

Name	type		Min	Max
srmRmdirRequest	TUserID	authorizationID	0	1
	TSURLInfo	directoryPath	1	1
	xsd:boolean	recursive	0	1
srmRmdirRespnose	TReturnStatus	returnStatus	1	1

4.2.1. Notes on the Behavior

a) It applies to directory only.
b) *recursive* is false by default.
c) To distinguish from *srmRm*(), this function is for directories only

4.2.2. Return Status Code

SRM_SUCCESS
- All requests are successfully completed. *directoryPath* 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 *directoryPath*

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

SRM_NON_EMPTY_DIRECTORY
- *directoryPath* 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 *recursive* is not supported in the SRM server

4.3. srmRm

This function will remove SURLs (the name space entries) in the storage system. Difference from *srmPurgeFromSpace* is that *srmPurgeFromSpace* removes only previously requested “copies” (or “state”) of the SURL in a particular space, and

srmPurgeFromSpace shall not remove SURLs or the name space entries. If any files are not released yet, this function will release them before removing SURLs.

Input	**srmRmRequest**
Output	**srmRmResponse**

Name	type		Min	Max
srmRmRequest	TUserID	authorizationID	0	1
	ArrayOfTSURLInfo	arrayOfSURLs	1	1
srmRmResponse	TReturnStatus	returnStatus	1	1
	ArrayOfTSURLReturnStatus	arrayOfFileStatuses	0	1

4.3.1. Notes on the Behavior

a) Applies to files only.
b) To distinguish from *srmRmdir*(), this function applies to files only

4.3.2. Return Status Code

For request level return 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
- *arrayOfSURLs* 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
- 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 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_BUSY
- client requests for files which there is an active srmPrepareToPut (no srmPutDone is not yet called) for.
- SURL is being used by other clients

SRM_FILE_UNAVAILABLE
- the request file is temporarily unavailable.

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

4.4. srmLs

srmLs() returns a list of files with a basic information. This operation may be asynchronous, and in such case, requestToken must be returned.

Input	**srmLsRequest**
Output	**srmLsResponse**

Name	type		Min	Max
srmLsRequest	TUserID	authorizationID	0	1
	ArrayOfTSURL	arrayOfSURLs	1	1
	ArrayOfTExtraInfo	storageSystemInfo	0	1
	TFileStorageType	fileStorageType	0	1
	xsd:boolean	fullDetailedList	0	1
	xsd:boolean	allLevelRecursive	0	1
	xsd:int	numOfLevels	0	1
	xsd:int	offset	0	1
	xsd:int	count	0	1
srmLsResponse	TReturnStatus	returnStatus	1	1
	TRequestToken	requestToken	0	1
	ArrayOfTMetaDataPathDetail	details	0	1

4.4.1. Notes on the Behavior

a) Applies to both directory and file
b) *fullDetailedList* is false by default.
   - For directories, only path is required to be returned.
   - For files, path and size are required to be returned.
c) If *fullDetailedList* is true, the full details are returned.
   - For directories, *path* and *userPermission* are required to be returned.
   - For *files, path, size, userPermission, lastModificationTime,* file *type,* and *lifetimeLeft* are required to be returned, similar to unix command *ls –l*.
d) If *allLevelRecursive* is true then file lists of all level below current will be provided as well.
e) If *allLevelRecursive* is "true" it dominates, i.e. ignore *numOfLevels*. If *allLevelRecursive* is "false" or missing, then do *numOfLevels*. If *numOfLevels* is

"0" (zero) or missing, assume a single level. If both *allLevelRecursive* and *numOfLevels* are missing, assume a single level.

f) If *numOfLevels* is 0, then information about directory itself is returned.
g) If *numOfLevels* is 1, then information about files in the directory is returned.
h) When listing for a particular type specified by "*fileStorageType*", only the files with that type will be in the output.
i) Empty directories will be returned.
j) We recommend width first in the listing.
k) We recommend that list of directories come before list of files in the return array (details).
l) For non-existing file or directory, SRM_INVALID_PATH must be returned.

4.4.2. Return Status Code

For request level return 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 *numOfLevels*, *offset* and *count* are provided.

SRM_NOT_SUPPORTED
- Requested *fileStorageType* is not supported in SRM
- Filtering *fileStorageType* is not supported in SRM
- Directory operation (directory SURL, allLevelRecursive and numOfLevels) is not supported in SRM

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

For file level return 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

4.5. srmStatusOfLsRequest

srmStatusOfLsRequest() returns a list of files with a basic information. This is an asynchronous operation of srmLs.

Input	**srmStatusOfLsRequestRequest**
Output	**srmStatusOfLsRequestResponse**

Name	type		Min	Max
srmStatusOfLsRequestRequest	TUserID	authorizationID	0	1
	TRequestToken	requestToken	1	1
	xsd:int	offset	0	1
	xsd:int	count	0	1
srmStatusOfLsRequestResponse	TReturnStatus	returnStatus	1	1
	ArrayOfTMetaDataPathDetail	details	0	1

4.5.1. Notes on the Behavior

a) Empty directories will be returned.
b) We recommend width first in the listing.
c) We recommend that list of directories come before list of files in the return array (details).
d) For non-existing file or directory, SRM_INVALID_PATH must be returned.

4.5.2. Return Status Code

For request level return 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 all files request is still on the queue.

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_INVALID_REQUEST
- Negative values for *offset* and *count* are provided.

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

SRM_NOT_SUPPORTED
- Requested *fileStorageType* is not supported in SRM
- Filtering *fileStorageType* is not supported in SRM
- Directory operation (directory SURL, allLevelRecursive and numOfLevels) is not supported in SRM

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

For file level return 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

4.6. srmMv

srmMv is to move a file from one local path to another local path.

Input	**srmMvRequest**
Output	**srmMvResponse**

Name	type		Min	Max
srmMvRequest	TUserID	authorizationID	0	1
	TSURLInfo	fromSURL	1	1
	TSURLInfo	toSURL	1	1
srmMvResponse	TReturnStatus	returnStatus	1	1

4.6.1. Notes on the Behavior

a) Applies to both directory and file.
b) Authorization checks need to be performed on both *fromSURL* and *toSURL*.

4.6.2. Return Status Code

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

SRM_INVALID_PATH
- *fromSURL* does not refer to an existing known path
- *toSURL* does not refer to a valid path

SRM_DUPLICATION_ERROR
- *toSURL* 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. Data Transfer Functions – Message Types and Operations

summary:

5.1. srmPrepareToGet

This function is used to bring files online upon the client's request and assign TURL so that client can access the file. 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 srmStatusOfGetRequest with the returned request token.

Input	**srmPrepareToGetRequest**
Output	**srmPrepareToGetResponse**

Name	type		Min	Max
srmPrepareToGetRequest	TUserID	authorizationID	0	1
	ArrayOfTGetFileRequest	arrayOfFileRequests	1	1
	xsd:string	userRequestDescription	0	1
	ArrayOfTExtraInfo	storageSystemInfo	0	1
	TFileStorageType	desiredFileStorageType	0	1
	TLifeTimeInSeconds	desiredTotalRequestTime	0	1

	TLifeTimeInSeconds	desiredPinLifetime	0	1
	TSpaceToken	targetSpaceToken	0	1
	TRetentionPolicyInfo	targetFileRetentionPolicyInfo	0	1
	TTransferParameters	transferParametersHint	0	1
srmPrepareToGetResponse	TReturnStatus	returnStatus	1	1
	TRequestToken	requestToken	0	1
	TLifeTimeInSeconds	remainingTotalRequestTime	0	1

5.1.1. Notes on the Behavior

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 "fileStorageType" is Volatile.
b) If input parameter *TSpaceToken* is provided, then the target space token must refer to online space. All requested files will be prepared into the target space.
c) Input parameter *targetFileRetentionPolicyInfo* of *TRetentionPolicyInfo* is to specify the desired retention policy information on the file when the file is prepared online.
d) If both input parameters *TSpaceToken* and *TRetentionPolicyInfo* are provided, then their types must match exactly. Otherwise, the request may be rejected with SRM_INVALID_REQUEST.
e) Access latency must be ONLINE always.
f) Input parameter *TAccessPattern* is provided at the request-level, and all files will have the same access pattern.
g) *TAccessPattern* may conflict with the type of the target space associated with target space token, when both provided. In this case, *TAccessPattern* in the input parameter *TTransferParameters* must be ignored.
h) The *userRequestDescription* is a user designated name for the request. It can be used in the *srmGetRequestID* function to get back the system assigned request tokens.
i) Only pull mode is supported for file transfers that client must pull the files from the TURL within the expiration time (*remainingPinTime*).
j) Input parameter *desiredPinLifetime* is for a client preferred lifetime (expiration time) on the prepared TURL.
k) If request is accepted, SRM assigns the *requestToken* for asynchronous status checking. In such case, the returned status code should be SRM_REQUEST_QUEUED.
l) *totalRequestTime* means: All the file transfer for this request must be complete within this *totalRequestTime*. Otherwise, SRM_REQUEST_TIMED_OUT must be returned as the status code with individual file status of SRM_REQUEST_SUSPENDED.
m) Client may use *srmResumeRequest*() to resume the request in the status of SRM_REQUEST_TIMED_OUT because of the *totalRequestTime*.
n) Client must use *srmAbortRequest*() to terminate the request in the status of SRM_REQUEST_TIMED_OUT because of the *totalRequestTime*.
o) SRM server may terminate the timed-out request after a certain period of time.

p) If *totalRequestTime* is 0 (zero), SRM server must continues the request until completed.
q) The invocation of *srmReleaseFile*() is expected for finished files later on.
r) The returned request token should be valid until all files in the request are released or removed.

5.1.2. Return Status Code

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

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 submit the request

SRM_INVALID_REQUEST
- *arrayOfFileRequest* is empty
- If both input parameters *TSpaceToken* and *TRetentionPolicyInfo* are provided, then their types must match exactly.
- Access latency is something other than ONLINE.
- *targetSpaceToken* does not refer to an existing known space in the SRM server.

SRM_SPACE_LIFETIME_EXPIRED
- space associated with the *targetSpaceToken* is expired.

SRM_EXCEED_ALLOCATION
- space associated with the *targetSpaceToken* 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 *targetSpaceToken*, 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 *desiredFileStorageType* that is not supported by the SRM server.
- *targetFileRetentionPolicyInfo* 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

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

5.2. srmStatusOfGetRequest

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

Input	**srmStatusOfGetRequestRequest**
Output	**srmStatusOfGetRequestResponse**

Name	type		Min	Max
srmStatusOfGetRequestRequest	TRequestToken	requestToken	1	1
	TUserID	authorizationID	0	1
	ArrayOfTSURL	arrayOfSourceSURLs	0	1
srmStatusOfGetRequestResponse	TReturnStatus	returnStatus	1	1
	ArrayOfTGetRequestFileStatus	arrayOfFileStatuses	0	1
	TLifeTimeInSeconds	remainingTotalRequestTime	0	1

5.2.1. Notes on the Behavior

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 "fileStorageType" is Volatile.
b) If *arrayOfSourceSURLs* is not provided, SRM must return status for all file requests in the request that is associated with the request token.
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.
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 srmPrepareToGet 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) *totalRequestTime* means: All the file transfer for this request must be complete within this *totalRequestTime*. Otherwise, SRM_REQUEST_TIMED_OUT must be returned as the status code with individual file status of SRM_REQUEST_SUSPENDED.
j) If *totalRequestTime* is 0 (zero), SRM server must continues the request until completed.

k) Client may use *srmResumeRequest*() to resume the request in the status of SRM_REQUEST_TIMED_OUT because of the *totalRequestTime*.
l) Client must use *srmAbortRequest*() to terminate the request in the status of SRM_REQUEST_TIMED_OUT because of the *totalRequestTime*.
m) SRM server may terminate the timed-out request after a certain period of time.

5.2.2. Return Status Code

For request level return status,

SRM_SUCCESS
- all file requests are successfully completed. All *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
- *requestToken* does not refer to an existing known request in the SRM server.

SRM_SPACE_LIFETIME_EXPIRED
- space associated with the *targetSpaceToken* is expired.

SRM_EXCEED_ALLOCATION
- space associated with the *targetSpaceToken* 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 *desiredFileStorageType* that is not supported by the SRM server.
- *targetFileRetentionPolicyInfo* 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 request is suspended.

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.

For file level return 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_REQUEST_SUSPENDED
- The requested file has been suspended because the request has timed out.

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

SRM_FILE_IN_CACHE
- pin lifetime has expired, but the file is still in the cache.

SRM_FILE_LIFETIME_EXPIRED
- *SURL* is expired
- *TUR*L is expired

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

5.3. srmBringOnline

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 *srmStatusOfBringOnlineRequest* with the returned request token.
This function is similar to srmPrepareToGet, but it does not return Transfer URL (TURL).

Input	**srmBringOnlineRequest**
Output	**srmBringOnlineResponse**

Name	type		Min	Max
srmBringOnlineRequest	TUserID	authorizationID	0	1
	ArrayOfTGetFileRequest	arrayOfFileRequests	1	1
	xsd:string	userRequestDescription	0	1
	ArrayOfTExtraInfo	storageSystemInfo	0	1
	TFileStorageType	desiredFileStorageType	0	1
	TLifeTimeInSeconds	desiredTotalRequestTime	0	1
	TLifeTimeInSeconds	desiredLifetime	0	1
	TSpaceToken	targetSpaceToken	0	1
	TRetentionPolicyInfo	targetFileRetentionPolicyInfo	0	1
	TTransferParameters	transferParametersHint	0	1
	TLifeTimeInSeconds	deferredStartTime	0	1
srmBringOnlineResponse	TReturnStatus	returnStatus	1	1
	TRequestToken	requestToken	0	1
	TLifeTimeInSeconds	remainingTotalRequestTime	0	1
	TLifeTimeInSeconds	remainingDeferredStartTime	0	1

5.3.1. Notes on the Behavior

a) Input parameter *deferredStartTime* is 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 *deferredStartTime* is reached, SRM_REQUEST_QUEUED must be returned.
b) Output parameter *remainingDeferredStartTime* indicates how long the deferredStartTime is left, if supported.
c) Input parameter *targetFileRetentionPolicyInfo* of *TRetentionPolicyInfo* is to specify the desired retention policy information on the file when the file is brought online.
d) If both input parameters *TSpaceToken* and *TRetentionPolicyInfo* are provided, then their types must match exactly. Otherwise, the request may be rejected, and SRM_INVALID_REQUEST will be returned.

e) *TAccessPattern* may conflict with the type of the target space associated with target space token, when both provided. In this case, *TAccessPattern* in the input parameter *TTransferParameters* must be ignored.
f) If the transfer protocol hints are not specified, default is assumed to be processing mode and LAN access for the site.
g) Access latency must be ONLINE always.
h) It is up to the SRM implementation to decide TConnectionType if not provided.
i) The *userRequestDescription* is a user designated name for the request. It can be used in the *srmGetRequestID* method to get back the system assigned request ID.
j) Input parameter *desiredLifetime* 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 *targetSpaceToken*.
k) This call may be an asynchronous (non-blocking) call, and SRM assigns the *requestToken* 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.
l) The returned request token should be valid until all files in the request are released, removed or aborted.
m) Input parameter *totalRequestTime* means: All the file transfer for this request must be complete within this *totalRequestTime*. Otherwise, SRM_REQUEST_TIMED_OUT must be returned as the status code with individual file status of SRM_REQUEST_SUSPENDED.
n) If *totalRequestTime* is 0 (zero), SRM server must continues the request until completed.
o) Client may use *srmResumeRequest*() to resume the request in the status of SRM_REQUEST_TIMED_OUT because of the *totalRequestTime*.
p) Client must use *srmAbortRequest*() to terminate the request in the status of SRM_REQUEST_TIMED_OUT because of the *totalRequestTime*.
q) SRM server may terminate the timed-out request after a certain period of time.
r) When *srmAbortRequest* is requested for *srmBringOnline* 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 *srmPurgeFromSpace* or *srmRm*.

5.3.2. Return Status Code

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

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 submit the request

SRM_INVALID_REQUEST

- *arrayOfFileRequest* is empty
- Access latency refers to something other than ONLINE.
- If both input parameters *TSpaceToken* and *TRetentionPolicyInfo* are provided, then their types must match exactly.
- *targetSpaceToken* does not refer to an existing known space in the SRM server.

SRM_SPACE_LIFETIME_EXPIRED
- space associated with the *targetSpaceToken* is expired.

SRM_EXCEED_ALLOCATION
- space associated with the *targetSpaceToken* 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 *desiredFileStorageType* that is not supported by the SRM server.
- *targetFileRetentionPolicyInfo* does not refer to a supported retention policy in the SRM server.
- *deferredStartTime* 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
- any other request failure. *Explanation* needs to be filled for details.

5.4. srmStatusOfBringOnlineRequest

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

Input	**srmStatusOfBringOnlineRequestRequest**
Output	**srmStatusOfBringOnlineRequestResponse**

Name	type		Min	Max
srmStatusOfBringOnlineRequestRequest	TRequestToken	requestToken	1	1
	TUserID	authorizationID	0	1
	ArrayOfTSURL	arrayOfSourceSURLs	0	1
srmStatusOfBringOnline	TReturnStatus	returnStatus	1	1

RequestResponse	ArrayOfTBringOnlineRequestFileStatus arrayOfFileStatuses	0	1
	TLifeTimeInSeconds remainingTotalRequestTime	0	1
	TLifeTimeInSeconds remainingDeferredStartTime	0	1

5.4.1. Notes on the Behavior

a) If *arrayOfSourceSURLs* is not provided, returns status for all files in this request.
b) When the file is ready online, the return status code should be SRM_FILE_IN_CACHE.
c) Output parameter *remainingDeferredStartTime* indicates how long the deferredStartTime is left, if supported.
d) When the file is ready for the client, the file is implicitly pinned in the cache and lifetime will be enforced.
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 srmBringOnline 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) Input parameter *totalRequestTime* means: All the file transfer for this request must be complete within this *totalRequestTime*. Otherwise, SRM_REQUEST_TIMED_OUT must be returned as the status code with individual file status of SRM_REQUEST_SUSPENDED.
j) If *totalRequestTime* is 0 (zero), SRM server just continues the request until completed.
k) Client may use *srmResumeRequest*() to resume the request in the status of SRM_REQUEST_TIMED_OUT because of the *totalRequestTime*.
l) Client must use *srmAbortRequest*() to terminate the request in the status of SRM_REQUEST_TIMED_OUT because of the *totalRequestTime*.
m) SRM server may terminate the timed-out request after a certain period of time.
n) If SRM decides not to bring any files until input parameter *deferredStartTime* is reached, SRM_REQUEST_QUEUED must be returned.

5.4.2. Return Status Code

For request level return status,

SRM_SUCCESS
- All requests are successfully completed. All *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
- *requestToken* 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 *desiredFileStorageType* that is not supported by the SRM server.
- *targetFileRetentionPolicyInfo* does not refer to a supported retention policy in the SRM server.
- *deferredStartTime* 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 *targetSpaceToken* is expired.

SRM_EXCEED_ALLOCATION
- space associated with the *targetSpaceToken* 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 request is suspended.

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.

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_FILE_IN_CACHE
- pin lifetime has expired, but the file is still in the cache.

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

SRM_REQUEST_SUSPENDED
- The requested file has been suspended because the request has timed out.

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

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

5.5. srmPrepareToPut

This function is used to write files into the storage. Upon the client's request, SRM prepares a TURL so that client can write data into the TURL. Lifetime (pinning expiration time) is assigned on the TURL. When a specified target space token is provided, the files will be located finally in the targeted space associated with the target space token. It is an asynchronous operation, and request token must be returned if the request is valid and accepted to the SRM. The status may be checked through srmStatusOfPutRequest with the returned request token.

Input	**srmPrepareToPutRequest**
Output	**srmPrepareToPutResponse**

Name	type		Min	Max
srmPrepareToPutRequest	TUserID	authorizationID	0	1
	ArrayOfTPutFileRequest	arrayOfFileRequests	1	1
	xsd:string	userRequestDescription	0	1
	TOverwriteMode	overwriteOption	0	1
	ArrayOfTExtraInfo	storageSystemInfo	0	1
	TLifeTimeInSeconds	desiredTotalRequestTime	0	1

	TLifeTimeInSeconds	desiredPinLifetime	0	1
	TLifeTimeInSeconds	desiredFileLifetime	0	1
	TFileStorageType	desiredFileStorageType	0	1
	TSpaceToken	targetSpaceToken	0	1
	TRetentionPolicyInfo	targetFileRetentionPolicyInfo	0	1
	TTransferParameters	transferParametersHint	0	1
srmPrepareToPutResponse	TReturnStatus	returnStatus	1	1
	TRequestToken	requestToken	0	1
	TLifeTimeInSeconds	remainingTotalRequestTime	0	1

5.5.1. Notes on the Behavior

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 "fileStorageType" is Volatile.
b) TURL returned by the srmPrepareToPut may not be used for read access with any protocol. An explicit srmPrepareToGet or srmBringOnline is required.
c) *TAccessPattern* may conflict with the type of the target space associated with target space token, when both provided. In this case, *TAccessPattern* in the input parameter *TTransferParameters* must be ignored.
d) Input parameter *TSpaceToken* 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 enough for all files.
e) Input parameter *targetFileRetentionPolicyInfo* of *TRetentionPolicyInfo* 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 *TSpaceToken* and *TRetentionPolicyInfo* 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 *targetSURL* in the *TPutFileRequest* has to be local to SRM. If *targetSURL* 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".
i) *srmPutDone*() is expected after each file is "put" into the prepared TURL.
j) Input parameter *desiredPinLifetime* is the lifetime (expiration time) on the TURL when the Transfer URL is prepared. It does not refer to the lifetime of the SURL.
k) Input parameter *desiredFileLifetime* 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.
l) The lifetime of the SURL starts as soon as SRM receives the *srmPutDone*(). If *srmPutDone*() 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 *remainingPinTime* in *TPutRequestFileStatus*.

m) If request is accepted, SRM assigns the *requestToken* for asynchronous status checking. In such case, the returned status code should be SRM_REQUEST_QUEUED.
n) Input parameter *totalRequestTime* means: All the file transfer for this request must be complete within this *totalRequestTime*. Otherwise, SRM_REQUEST_TIMED_OUT must be returned as the status code with individual file status of SRM_REQUEST_SUSPENDED.
o) If *totalRequestTime* is 0 (zero), SRM server must continues the request until completed.
p) Client may use *srmResumeRequest*() to resume the request in the status of SRM_REQUEST_TIMED_OUT because of the *totalRequestTime*.
q) Client must use *srmAbortRequest*() to terminate the request in the status of SRM_REQUEST_TIMED_OUT because of the *totalRequestTime*.
r) SRM server may terminate the timed-out request after a certain period of time.

5.5.2. Return Status Code

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

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 submit the request

SRM_INVALID_REQUEST
- If both input parameters *TSpaceToken* and *TRetentionPolicyInfo* are provided, then their types must match exactly.
- *targetSpaceToken* does not refer to an existing known space in the SRM server.

SRM_SPACE_LIFETIME_EXPIRED
- space associated with the *targetSpaceToken* is expired.

SRM_EXCEED_ALLOCATION
- space associated with the *targetSpaceToken* 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 *desiredFileStorageType* that is not supported by the SRM server.
- *targetFileRetentionPolicyInfo* 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
- any other request failure. *Explanation* needs to be filled for details.

5.6. srmStatusOfPutRequest

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

Input	**srmStatusOfPutRequestRequest**
Output	**srmStatusOfPutRequestResponse**

Name	type		Min	Max
srmStatusOfPutRequestRequest	TRequestToken	requestToken	1	1
	TUserID	authorizationID	0	1
	ArrayOfTSURL	arrayOfTargetSURLs	0	1
srmStatusOfPutRequestResponse	TReturnStatus	returnStatus	1	1
	ArrayOfTPutRequestFileStatus	arrayOfFileStatuses	0	1
	TLifeTimeInSeconds	remainingTotalRequestTime	0	1

5.6.1. Notes on the Behavior

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 "fileStorageType" is Volatile.
b) If *arrayOfTargetSURLs* is not provided, returns status for all the file requests in this request.
c) When the space is ready for client to "put" data and TURL is prepared, the return status code should be SRM_SPACE_AVAILABLE.
d) 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.
e) If a targetSURL is provided with some directory structure, the directory structure must exist, and SRM will not create the directory structure for the targetSURL. In such case, SRM_INVALID_PATH must be returned. srmMkdir may be used to create the directory structure.
f) 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.
g) SRM must fail (SRM_FAILURE) only if all files in the request failed.
h) Input parameter *totalRequestTime* means: All the file transfer for this request must be complete within this *totalRequestTime*. Otherwise,

SRM_REQUEST_TIMED_OUT must be returned as the status code with individual file status of SRM_REQUEST_SUSPENDED.

i) If *totalRequestTime* is 0 (zero), SRM server must continues the request until completed.

j) Client may use *srmResumeRequest*() to resume the request in the status of SRM_REQUEST_TIMED_OUT because of the *totalRequestTime*.

k) Client must use *srmAbortRequest*() to terminate the request in the status of SRM_REQUEST_TIMED_OUT because of the *totalRequestTime*.

l) SRM server may terminate the timed-out request after a certain period of time.

5.6.2. Return Status Code

For request level return 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
- *requestToken* does not refer to an existing known request in the SRM server.
- *targetSpaceToken* that client provided does not refer to an existing space in the SRM server.

SRM_SPACE_LIFETIME_EXPIRED
- space associated with the *targetSpaceToken* is expired.

SRM_EXCEED_ALLOCATION
- space associated with the *targetSpaceToken* 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 request is suspended.

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 *desiredFileStorageType* that is not supported by the SRM server.
- *targetFileRetentionPolicyInfo* 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
- 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_DONE
- Client's file transfer into *TURL* is completed, and *srmPutDone* on the *targetSURL* is completed. The file is now in the cache and lifetime on the *targetSURL* is started.

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

SRM_INVALID_PATH
- *targetSURL* does not refer to a valid path.

SRM_DUPLICATION_ERROR
- *targetSURL* 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_REQUEST_SUSPENDED
- The requested file has been suspended because the request has timed out.

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.

5.7. srmCopy

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 *srmCopy* 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 srmCopy request (PULL case), 2. Source SRM is where client makes a srmCopy request (PUSH case).

1. PULL case: Upon the client's *srmCopy* request, the target SRM makes a space at the target storage, and makes a request *srmPrepareToGet* to the source SRM. When TURL is ready at the source SRM, the target SR M transfers the file from the source TURL into the prepared target storage. After the file transfer completes, *srmReleaseFiles* is issued to the source SRM.
2. PUSH case: Upon the client's *srmCopy* request, the source SRM prepares a file to be transferred out to the target SRM, and makes a request *srmPrepareToPut* 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, *srmPutDone* 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 srmStatusOfCopyRequest with the returned request token.

Input	**srmCopyRequest**
Output	**srmCopyResponse**

Name	type		Min	Max
srmCopyRequest	TUserID	authorizationID	0	1
	ArrayOfTCopyFileRequest	arrayOfFileRequests	1	1
	xsd:string	userRequestDescription	0	1
	TOverwriteMode	overwriteOption	0	1
	TLifeTimeInSeconds	desiredTotalRequestTime	0	1
	TLifeTimeInSeconds	desiredTargetSURLLifetime	0	1
	TFileStorageType	targetFileStorageType	0	1
	TSpaceToken	targetSpaceToken	0	1
	TRetentionPolicyInfo	targetFileRetentionPolicyInfo	0	1
srmCopyResponse	TReturnStatus	returnStatus	1	1
	TRequestToken	requestToken	0	1
	TLifeTimeInSeconds	remainingTotalRequestTime	0	1

5.7.1. Notes on the Behavior

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 "fileType" is Volatile.
b) When aborted, target SURLs need to be provided.
c) Input parameter *TSpaceToken* 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.
d) Input parameter *targetFileRetentionPolicyInfo* of *TRetentionPolicyInfo* is to specify the desired retention policy information on the file when the file is written into the target storage system.

e) If both input parameters *TSpaceToken* and *TRetentionPolicyInfo* are provided, then their types must match exactly. Otherwise, the request may be rejected, and SRM_INVALID_REQUEST must be returned.
f) If request is accepted, SRM assigns the *requestToken* for asynchronous status checking. In such case, the returned status code should be SRM_REQUEST_QUEUED.
g) Pull mode: copy from remote location to the SRM. (e.g. from remote to MSS.)
h) Push mode: copy from the SRM to remote location.
i) Always release files through *srmReleaseFiles* from the source after copy is done, if source is an SRM and PULL mode was performed.
j) Always issue *srmPutDone* to the target after copy is done, if target is an SRM and PUSH mode was performed.
k) Note there is no protocol negotiation with the client for this request.
l) Input parameter *desiredTotalRequestTime* means: if all the file transfer for this request must be complete in this *desiredTotalRequestTime*. Otherwise, the request is returned as failed at the end of the *desiredTotalRequestTime*, and SRM_REQUEST_TIMED_OUT must be returned as the status code with individual file status of SRM_REQUEST_SUSPENDED. All completed files must not be removed, but status of the files must be returned to the client.
m) If *totalRequestTime* is 0 (zero), SRM server must continues the request until completed.
n) Client may use *srmResumeRequest*() to resume the request in the status of SRM_REQUEST_TIMED_OUT because of the *totalRequestTime*.
o) Client must use *srmAbortRequest*() to terminate the request in the status of SRM_REQUEST_TIMED_OUT because of the *totalRequestTime*.
p) SRM server may terminate the timed-out request after a certain period of time.
q) When both sourceSURL and targetSURL are local, local copy must be performed.
r) Empty directories are copied as well.
s) If a targetSURL is provided with some directory structure, the directory structure must exist, and SRM will not create the directory structure for the targetSURL. In such case, SRM_INVALID_PATH must be returned. srmMkdir may be used to create the directory structure.
t) If the sourceSURL and targetSURL 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 *TDirOption* indicates them.

5.7.2. Return Status Code

SRM_SUCCESS

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

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 submit the request
- Client is not authorized to copy files into the space that client provided with *targetSpaceToken* or *targetFileRetentionPolicyInfo*

SRM_INVALID_REQUEST
- If both input parameters *TSpaceToken* and *TRetentionPolicyInfo* are provided, then their types must match exactly.
- *targetSpaceToken* does not refer to an existing known space in the SRM server.

SRM_SPACE_LIFETIME_EXPIRED
- space associated with the *targetSpaceToken* is expired.

SRM_EXCEED_ALLOCATION
- space associated with the *targetSpaceToken* 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 *desiredFileStorageType* that is not supported by the SRM server.
- *targetFileRetentionPolicyInfo* 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
- any other request failure. *Explanation* needs to be filled for details.

5.8. srmStatusOfCopyRequest

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

Input	**srmStatusOfCopyRequestRequest**
Output	**srmStatusOfCopyRequestResponse**

Name	type		Min	Max
srmStatusOfCopyRequestRequest	TRequestToken	requestToken	1	1
	TUserID	authorizationID	0	1
	ArrayOfTSURL	arrayOfSourceSURLs	0	1
	ArrayOfTSURL	arrayOfTargetSURLs	0	1
srmStatusOfCopyRequestResponse	TReturnStatus	returnStatus	1	1
	ArrayOfTCopyRequestFileStatus	arrayOfFileStatuses	0	1
	TLifeTimeInSeconds	remainingTotalRequestTime	0	1

5.8.1. Notes on the Behavior

a) If *arrayOfSourceSURLs* and/or *arrayOfTargetSURLs* are not provided, return status for all file requests in the request.
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) *totalRequestTime* means: All the file transfer for this request must be complete within this *totalRequestTime*. Otherwise, SRM_REQUEST_TIMED_OUT must be returned as the status code with individual file status of SRM_REQUEST_SUSPENDED.
e) If *totalRequestTime* is 0 (zero), SRM server must continues the request until completed.
f) Client may use *srmResumeRequest*() to resume the request in the status of SRM_REQUEST_TIMED_OUT because of the *totalRequestTime*.
g) Client must use *srmAbortRequest*() to terminate the request in the status of SRM_REQUEST_TIMED_OUT because of the *totalRequestTime*.
h) SRM server may terminate the timed-out request after a certain period of time.

5.8.2. Return Status Code

For request level return 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

- *requestToken* does not refer to an existing known request in the SRM server.
- *targetSpaceToken* 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 *arrayOfSourceURLs and arrayOfTargetURLs* cannot fit the results to return.

SRM_REQUEST_TIMED_OUT

- Total request time is over and the request is suspended.

SRM_SPACE_LIFETIME_EXPIRED

- space associated with the *targetSpaceToken* is expired.

SRM_EXCEED_ALLOCATION

- space associated with the *targetSpaceToken* 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 spaceToken, 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 *desiredFileStorageType* that is not supported by the SRM server.
- *targetFileRetentionPolicyInfo* 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

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

For file level return status,

SRM_DONE

- successful request completion for the file. The source *SURL* is copied into the target destination *targetSURL* successfully, and lifetime on the *targetSURL* 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 (*sourceSURL*) is permanently lost.

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

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

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

SRM_INVALID_PATH
- *sourceSUR* does not exist
- *targetSURL* does not refer to a valid path.

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

SRM_AUTHORIZATION_FAILURE
- Client is not authorized to copy files from *sourceSURL*
- Client is not authorized to copy files into *targetSURL*
- Client is not authorized to copy files into the space that client provided with *targetSpaceToken* or *targetFileRetentionPolicyInfo*

SRM_REQUEST_SUSPENDED
- The requested file has been suspended because the request has timed out.

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.

5.9. srmReleaseFiles

This function is used to release pins on the previously requested “copies” (or “state”) of the SURL. This function normally follows srmPrepareToGet or srmBringOnline functions.

Input	**srmReleaseFilesRequest**
Output	**srmReleaseFilesResponse**

Name	type		Min	Max
srmReleaseFilesRequest	TRequestToken	requestToken	0	1
	TUserID	authorizationID	0	1
	ArrayOfTSURL	arrayOfSURLs	1	1
	xsd:boolean	doRemove	0	1
srmReleaseFilesResponse	TReturnStatus	returnStatus	1	1
	ArrayOfTSURLReturnStatus	arrayOfFileStatuses	0	1

5.9.1. Notes on the Behavior

a) *doRemove* by default is false. If remove is true, the pin on the file is released, the "copy" or "state" is removed and SRM may release the resource.
b) Directory is okay for SURL. In such case, it will release all files recursively in the directory.
c) If requestToken is not provided, then the SRM will release all the files specified by the SURLs owned by this user, regardless of the requestToken.
d) If requestToken is not provided, then authorizationID is needed. It may be inferred or provide in the call.

5.9.2. Return Status Code

For request level return 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
- *arrayOfSURLs* is empty.
- *requestToken* does not refer to an existing known request of *srmPrepareToGet* or *srmBringOnline* 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
- input parameter *doRemove* is not supported in the SRM. *srmRm* must be used.

For file level return 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 *remove* flag is on

SRM_FILE_LIFETIME_EXPIRED
- *SURL* is expired already.

SRM_REQUEST_SUSPENDED
- The requested file has been suspended because the request has timed out.

SRM_ABORTED
- The requested file has been aborted.

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

5.10. srmPutDone

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

Input	**srmPutDoneRequest**
Output	**srmPutDoneResponse**

Name	type		Min	Max
srmPutDoneRequest	TRequestToken	requestToken	1	1
	TUserID	authorizationID	0	1
	ArrayOfTSURL	arrayOfSURLs	1	1
srmPutDoneResponse	TReturnStatus	returnStatus	1	1
	ArrayOfTSURLReturnStatus	arrayOfFileStatuses	0	1

5.10.1. Notes on the Behavior

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

5.10.2. Return Status Code

For request level return 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 *requestToken*

SRM_INVALID_REQUEST
- *arrayOfSURLs* is empty.
- *requestToken* is empty.
- *requestToken* 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 suspended.

SRM_ABORTED
- The request has been aborted.

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

For file level return status,

SRM_DONE
- successful request completion of the “put done” for the *targetSURL*

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

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

SRM_FILE_LIFETIME_EXPIRED
- *targetSURL* has an expired *TURL*.

SRM_SPACE_LIFETIME_EXPIRED
- *targetSURL* has an expired space allocation.

SRM_REQUEST_SUSPENDED
- The requested file has been suspended because the request has timed out.

SRM_ABORTED
- The requested SURL file has been aborted.

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

5.11. srmAbortRequest

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

Input	**srmAbortRequestRequest**
Output	**srmAbortRequestResponse**

Name	type		Min	Max
srmAbortRequestRequest	TRequestToken	requestToken	1	1
	TUserID	authorizationID	0	1
srmAbortRequestResponse	TReturnStatus	returnStatus	1	1

5.11.1. Notes on the Behavior

a) Terminate all files in the request regardless of the file state. Remove files from the queue, and release cached files if a limited lifetime is associated with the file. Expired files are released.
b) 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 *srmRm* or *srmPurgeFromSpace*.
c) Abort must be allowed to all requests with *requestToken.*

5.11.2. Return Status Code

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 *requestToken*

SRM_INVALID_REQUEST
- *requestToken* 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

5.12. srmAbortFiles

srmAbortFiles() allows clients to abort selective file requests from the asynchronous requests of any type. It may include data transfer requests initiated by a call to

srmPrepareToGet(), srmBringOnline(), srmPrepareToPut(), or srmCopy(). The effect of a srmAbortFiles() depends on the type of the request.

Input	**srmAbortFilesRequest**
Output	**srmAbortFilesResponse**

Name	type		Min	Max
srmAbortFilesRequest	TRequestToken	requestToken	1	1
	ArrayOfTSURL	arrayOfSURLs	1	1
	TUserID	authorizationID	0	1
srmAbortFilesResponse	TReturnStatus	returnStatus	1	1
	ArrayOfTSURLReturnStatus	arrayOfFileStatuses	0	1

5.12.1. Notes on the Behavior

a) Abort all files in this call regardless of the state.

5.12.2. Return Status Code

For request level return 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 *requestToken*

SRM_INVALID_REQUEST
- *arrayOfSURLs* is empty.
- *requestToken* is empty.
- *requestToken* 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

For file level return 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.

5.13. srmSuspendRequest

srmSuspendedRequest is to suspend a previously submitted active request.

Input	**srmSuspendRequestRequest**
Output	**srmSuspendRequestResponse**

Name	type		Min	Max
srmSuspendRequestRequest	TRequestToken	requestToken	1	1
	TUserID	authorizationID	0	1
srmSuspendRequestResponse	TReturnStatus	returnStatus	1	1

5.13.1. Notes on the Behavior

a) Suspend all files in this request until srmResumeRequest is issued.

5.13.2. Return Status Code

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 *requestToken*

SRM_INVALID_REQUEST

- *requestToken* is empty.
- *requestToken* 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

5.14. srmResumeRequest

srmResumeRequest is to resume previously suspended requests or timed-out request**.**

Input	**srmResumeRequestRequest**

Output	**srmResumeRequestResponse**

Name	type		Min	Max
srmResumeRequestRequest	TRequestToken	requestToken	1	1
	TUserID	authorizationID	0	1
srmResumeRequestResponse	TReturnStatus	returnStatus	1	1

5.14.1. Notes on the Behavior

a) Resume the previously suspended request.
b) Resume the previously timed-out request which is in the state of SRM_REQUEST_TIMED_OUT.

5.14.2. Return Status Code

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 *requestToken*

SRM_INVALID_REQUEST
- *requestToken* is empty.
- *requestToken* 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

5.15. srmGetRequestSummary

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

Input	**srmGetRequestSummaryRequest**
Output	**srmGetRequestSummaryResponse**

Name	type		Min	Max
srmGetRequestSummary Request	ArrayOfTRequestToken	arrayOfRequestTokens	1	1
	TUserID	authorizationID	0	1
srmGetRequestSummary Response	TReturnStatus	returnStatus	1	1
	ArrayOfTRequestSummary	arrayOfRequestSummaries	0	1

5.15.1. Return Status Code

For request interface level return 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 *requestToken*

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

For request level return status,

SRM_INVALID_REQUEST
- *requestToken* 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 suspended.

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.

5.16. srmExtendFileLifeTime

srmExtendFileLifetime() 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 *srmPrepareToGet* and *srmPrepareToPut*.

Input	**srmExtendFileLifeTimeRequest**
Output	**srmExtendFileLifeTimeResponse**

Name	type		Min	Max
srmExtendFileLifeTimeRequest	TUserID	authorizationID	0	1
	TRequestToken	requestToken	0	1
	ArrayOfTSURL	arrayOfSURLs	0	1
	TLifeTimeInSeconds	newFileLifetime	0	1
	TLifeTimeInSeconds	newPinLifetime	0	1
srmExtendFileLifeTimeResponse	TReturnStatus	returnStatus	1	1
	TLifeTimeInSeconds	newExtendedFileLifetime	0	1
	TLifeTimeInSeconds	newExtendedPinLifetime	0	1
	ArrayOfTSURLReturnStatus	arrayOfFileStatuses	0	1

5.16.1. Notes on the Behavior

a) *newPinLifetime* and *newFileLifetime* are relative to the calling time. Lifetime will be set from the calling time for the specified period.
b) If only *requestToken* is provided, and none of SURLs are not provided, lifetime of all SURLs belong to the request that is associated with *requestToken* will have a new lifetime.
c) When extending pinning lifetime of *TURLs* with *newPinLifetime, requestToken* must be provided.
d) When extending lifetime of *SURLs* with *newFileLifetime*, *requestToken* is optional.
e) The number of lifetime extensions maybe 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 *newPinLifetime* or *newFileLifetime* is not specified, the SRM can use its default to assign the *newPinLifetime* or *newFileLifetime*.
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 srmExtendFileLifetimeInSpace

5.16.2. Return Status Code

For request level return 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

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

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

For file level return 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_REQUEST_SUSPENDED

- The requested file has been suspended because the request has timed out or because the request is suspended.

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.

5.17. srmGetRequestTokens

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

Input	**srmGetRequestTokensRequest**
Output	**srmGetRequestTokensResponse**

Name	type		Min	Max
srmGetRequestTokensRequest	xsd:string	userRequestDescription	0	1
	TUserID	authorizationID	0	1
srmGetRequestTokensResponse	TReturnStatus	returnStatus	1	1
	ArrayOfTRequestTokenReturn	arrayOfRequestTokens	0	1

5.17.1. Notes on the Behavior

a) If userRequestDescription is null, returns all requests this user has.
b) If the user assigned the same name to multiple requests, he may get back multiple request IDs each with the time the request was made.

5.17.2. Return Status Code

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
- *userRequestDescription* 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

6. Discovery Functions – Message Types and Operations

summary:

6.1. srmGetTransferProtocols

This function is to discover what transfer protocols are supported by the SRM.

Input	**srmGetTransferProtocolsRequest**
Output	**srmGetTransferProtocolsResponse**

Name	type		Min	Max
srmGetTransferProtocolsRequest	TUserID	authorizationID	0	1
srmGetTransferProtocolsResponse	TReturnStatus	returnStatus	1	1
	ArrayOfTSupportedTransferProtocol	protocolInfo	0	1

6.1.1. Notes on the Behavior

a) *srmGetTransferProtocols*() returns the supported file transfer protocols in the SRM with any additional information about the transfer protocol.

6.1.2. Return Status Code

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.

6.2. srmPing

This function is used to check the state of the SRM. It works as an “are you alive” type of call.

Input	**srmPingRequest**
Output	**srmPingResponse**

Name	type		Min	Max
srmPingRequest	TUserID	authorizationID	0	1
srmPingResponse	xsd:string	versionInfo	1	1
	ArrayOfTExtraInfo	otherInfo	0	1

6.2.1. Notes on the Behavior

a) *srmPing*() returns a string containing SRM v2.2 version number as a minimal "up and running" information. For this particular SRM v2.2 version, it must be "**v2.2**". Other versions may have "**v1.1**", "**v3.0**", and so on.

b) Any additional information about the SRM can be provided in the output parameter *otherInfo*.

7. Appendix

7.1. 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

7.2. 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-->
