The Storage Resource Manager
Web Services
Operational Interface Specification

Version 2.1.1

18 May 2004

Contributors to this document are:

EDG-WP2	Peter Kunszt, Heinz Stockinger, Kurt Stockinger, Erwin Laure
EDG-WP5	Jean-Philippe Baud, Stefano Occhetti, Jens Jensen, Emil Knezo, Owen Synge, Olof Barring
JLAB	Bryan Hess, Andy Kowalski, Chip Watson
FNAL	Don Petravick, Timur Perelmutov, Rich Wellner
LBNL	Junmin Gu , Arie Shoshani, Alex Sim

This version prepared by:

Alex Sim at Lawrence Berkeley National Laboratory
Timur Perelmutov at Fermi National Accelerator Laboratory

http://sdm.lbl.gov/srm-wg/doc/SRM-WSDL.v2.1.1.doc

Table of Contents

Introduction

This document contains the Web Service (WS) operational interface specification of SRM 2.1.1. It translates the functional interface specification of SRM 2.1.1 (see http://sdm.lbl.gov/srm-wg/doc/SRM.spec.v2.1.1.html).

The purpose of this document is to standardize the Web Service interface of Storage Resource Managers (SRMs) – a Grid middleware component. This document is a follow up to the basic SRM design consideration document (see http://sdm.lbl.gov/srm-wg/doc/SRM.Functionality.And.Interface.Design.Version2.1.pdf).

It is advisable to read the document SRM.v2.1.1 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 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 optionality. 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 optionality 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.v2.1.1.html).

Change Log

2.1.1 – 14/5/2004

- Web Service operational interface document first written.

Understandings and Agreements

1. We use "https" where we mean http: protocol with GSI authentication. 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 as of May 2004.
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. TStorageSystemInfo 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).

We define the SRM WSDL namespace as following:

```
targetNamespace="urn://StorageResourceManagerV2.1.1"
xmlns:impl="urn://StorageResourceManagerV2.1.1"
portType name="ISRM"
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.**

Simple Types

name	type
TSpaceType	xsd:string enum {Volatile, Durable, Permanent}
TFileStorageType	xsd:string enum {Volatile, Durable, Permanent}
TFileType	xsd:string enum {File, Directory, Link}
TPermissionMode	xsd:string enum {NONE, X, W, WX, R, RX, RW, RWX}
TPermissionType	xsd:string enum {ADD, REMOVE, CHANGE}
TRequestType	xsd:string enum {PrepareToGet, PrepareToPut, Copy}
TOverwriteMode	xsd:string enum {Never, Always, WhenFilesAreDifferent}
TStatusCode	xsd:string enum { SRM_SUCCESS, SRM_FAILURE, SRM_AUTHENTICATION_FAILURE, SRM_UNAUTHORIZED_ACCESS, 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_CUSTOM_STATUS }

Complex Types

name	type		Min	Max
TRequestToken	xsd:string		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:long			
TGMTTime	xsd:dataTime		1	1
TLifeTimeInSeconds	xsd:long			
TSURL	xsd:anyURI		1	1
TTURL	xsd:anyURI		1	1
TMetaDataPathDetail	xsd:string TReturnStatus TSizeInBytes TOwnerPermission ArrayOfTUserPermission ArrayOfTGroupPermission	path status size ownerPermission userPermission groupPermission	1 0 0 0 0 0	1 1 1 1 1 1

	TOtherPermission	otherPermission	0	1
	TGMTTime	createdAtTime	0	1
	TGMTTime	lastModificationTime	0	1
	TUserID	owner	0	1
	TFileStorageType	fileStorageType	0	1
	TFileType	type	0	1
	TLifeTimeInSeconds	lifetimeAssigned	0	1
	TLifeTimeInSeconds	lifetimeLeft	0	1
	TCheckSumType	checkSumType	0	1
	TCheckSumValue	checkSumValue	0	1
	TSURL	originalSURL	0	1
	ArrayOfTMetaDataPathDetail	subPath	0	1
ArrayOfTMetaDataPathDetail	TMetaDataPathDetail []			
TMetaDataSpace	TSpaceType	type	0	1
	TSpaceToken	spaceToken	1	1
	xsd:boolean	isValid	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
TStorageSystemInfo	xsd:string		1	1
TDirOption	xsd:boolean	isSourceADirectory	0	1
	xsd:boolean	allLevelRecursive	1	1
	xsd:int	numOfLevels		
TSURLInfo	TSURL	SURLOrStFN	1	1
	TStorageSystemInfo	storageSystemInfo	0	1
ArrayOfTSURLInfo	TSURLInfo []			
TGetFileRequest	TSURLInfo	fromSURLInfo	1	1
	TLifeTimeInSeconds	lifetime	0	1
	TFileStorageType	fileStorageType	0	1
	TSpaceToken	spaceToken	0	1
	TDirOption	dirOption	0	1
ArrayOfTGetFileRequest	TGetFileRequest []			
TPutFileRequest	TSURLInfo	toSURLInfo	1	1
	TLifeTimeInSeconds	lifetime	0	1
	TFileStorageType	fileStorageType	0	1
	TSpaceToken	spaceToken	0	1
	TSizeInBytes	knownSizeOfThisFile	0	1
ArrayOfTPutFileRequest	TPutFileRequest []			
TCopyFileRequest	TSURLInfo	fromSURLInfo	1	1
	TSURLInfo	toSURLInfo	1	1
	TLifeTimeInSeconds	lifetime	0	1
	TFileStorageType	fileStorageType	0	1

	TSpaceToken	spaceToken	0	1
	TOverwriteMode	overwriteMode	0	1
	TDirOption	dirOption	0	1
ArrayOfTCopyFileRequest	TCopyFileRequest []			
TReturnStatus	TStatusCode	statusCode	1	1
	xds: string	explanation	0	1
TSURLReturnStatus	TSURL	surl	1	1
	TReturnStatus	status	1	1
ArrayOfTSURLReturnStatus	TSURLReturnStatus []			
TGetRequestFileStatus	TSURL	fromSURLInfo	1	1
	TSizeInBytes	fileSize	0	1
	TReturnStatus	status	1	1
	TLifeTimeInSeconds	estimatedWaitTimeOnQueue	0	1
	TLifeTimeInSeconds	estimatedProcessingTime	0	1
	TTURL	transferURL	0	1
	TLifeTimeInSeconds	remainingPinTime	0	1
ArrayOfTGetRequestFileStatus	TGetRequestFileStatus []			
TPutRequestFileStatus	TSizeInBytes	fileSize	0	1
	TReturnStatus	status	1	1
	TLifeTimeInSeconds	estimatedWaitTimeOnQueue	0	1
	TLifeTimeInSeconds	estimatedProcessingTime	0	1
	TTURL	transferURL	0	1
	TSURL	siteURL	0	1
	TLifeTimeInSeconds	remainingPinTime	0	1
ArrayOfTPutRequestFileStatus	TPutRequestFileStatus []			
TCopyRequestFileStatus	TSURL	fromSURL	1	1
	TSURL	toSURL	1	1
	TSizeInBytes	fileSize	0	1
	TReturnStatus	status	1	1
	TLifeTimeInSeconds	estimatedWaitTimeOnQueue	0	1
	TLifeTimeInSeconds	estimatedProcessingTime	0	1
	TLifeTimeInSeconds	remainingPinTime	0	1
ArrayOfTCopyRequestFileStatus	TCopyRequestFileStatus []			
TRequestSummary	TRequestToken	requestToken	0	1
	TRequestType	requestType	0	1
	xsd:int	totalFilesInThisRequest		
	xsd:int	numOfQueuedRequests		
	xsd:int	numOfFinishedRequests		
	xsd:int	numOfProgressingRequests		
	xsd:boolean	isSuspended		
ArrayOfTRequestSummary	TRequestSummary []			
TSURLPermissionReturn	TSURL	surl	0	1
	TReturnStatus	status	0	1
	TPermissionType	userPermission	0	1
ArrayOfTSURLPermissionReturn	TSURLPermissionReturn []			
TRequestTokenReturn	TRequestToken	requestToken	0	1

	TGMTTime createdAtTime	0	1
ArrayOfTRequestTokenReturn	TRequestTokenReturn[]		
ArrayOf_xsd_string	xsd:string []		
ArrayOfTSURL	TSURL []		

- **TGMTTime** 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.
- **TstorageSystemInfo** can contain but is not limited to the following: storage device, storage login ID, storage login authorization. StorageSystemInfo is a string that contains the login and password required by the storage system. For example, it might have the form of login:passwd@hostname, where ":" is a reserved separator between login and passwd. If hostname is not provided, it is defaulted to what's in the accompanying site URL or the host of SRM.
- In TGetFileRequest, TPutFileRequest, TCopyFileRequest, 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.
- TMetaDataSpace can refer to a single space of each type (i.e. volatile, durable, permanent). It does not include the extra space needed to hold the directory structures.
- 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, 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.
- The type definition SURL above is used for both site URL and the "Storage File Name" (stFN). This was done in order to simplify the notation. Recall that stFN is the file path/name of the intended storage location when a file is put (or copied) into an SRM controlled space. Thus, a stFN can be thought of a special case of an SURL, where the protocol is assumed to be "srm" and the machine:port is assumed to be local to the SRM. For example, when the request srmCopy is made, the source file is specified by a site URL, and the target location can be optionally specified as a stFN. By considering the stFN a special case of an SURL, an srmCopy takes SURLs as both the source and target parameters.
- The requestToken assigned by SRM is unique and immutable (non-reusable). For example, if the date:time is part of the requestToken it will be immutable.

Message types and Operations

Space Management Functions

summary:

srmReserveSpace

srmReleaseSpace
srmUpdateSpace
srmCompactSpace

srmGetSpaceMetaData
srmChangeFileStorageType
srmGetSpaceToken

details:

srmReserveSpace

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

name	type		Min	Max
srmReserveSpaceRequest	TUserID	authorizationID	0	1
	TSpaceType	typeOfSpace	1	1
	String	userSpaceTokenDescription	0	1
	TSizeInBytes	sizeOfTotalSpaceDesired	0	1
	TSizeInBytes	sizeOfGuaranteedSpaceDesired	0	1
	TLifeTimeInSeconds	lifetimeOfSpaceToReserve	0	1
	TStorageSystemInfo	storageSystemInfo	0	1
srmReserveSpaceResponse	TSpaceType	typeOfReservedSpace	0	1
	TSizeInBytes	sizeOfTotalReservedSpace	0	1
	TSizeInBytes	sizeOfGuaranteedReservedSpace	0	1
	TLifeTimeInSeconds	lifetimeOfReservedSpace	0	1
	TSpaceToken	referenceHandleOfReservedSpace	0	1
	TReturnStatus	returnStatus	1	1

notes:

- *lifetimeOfSpaceToReserve is not needed if requesting permanent space.*
- *SRM can provide default size and lifetime if not supplied.*
- *storageSystemInfo is optional in case storage system requires additional security check.*
- *If sizeOfTotalSpaceDesired is not specified, the SRM will return its default quota.*

srmReleaseSpace

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

name	type		Min	Max
srmReleaseSpaceRequest	TUserID	authorizationID	0	1

	TSpaceToken	spaceToken	1	1
	TStorageSystemInfo	storageSystemInfo	0	1
	Xsd:boolean	forceFileRelease	0	1
srmReleaseSpaceResponse	TReturnStatus	returnStatus	1	1

notes:

- *forceFileRelease=false is 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=true must be specified.*
- *To be safe, a request to release a reserved space that has an on-going file transfer will return false, even forceFileRelease= true.*
- *When space is releasable and forceFileRelease=true, all the files in the space are released, even in durable or permanent space.*
- *When space is released, the files in that space are treated according to their types: If permanent, keep it. If durable, perform action at the end of lifetime. If Volatile, release it at the end of lifetime.*

srmUpdateSpace

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	newLifeTimeFromCallingTime	0	1
	TStorageSystemInfo	storageSystemInfo	0	1
srmUpdateSpaceResponse	TSizeInBytes	sizeOfTotalSpace	0	1
	TSizeInBytes	sizeOfGuaranteedSpace	0	1
	TLifeTimeInSeconds	lifetimeGranted	0	1
	TReturnStatus	returnStatus	1	1

notes:

- *Includes size and time*
- *If neither size nor lifetime are supplied in the input, then return will be null.*
- *newSize is the new actual size of the space, so has to be positive.*
- *newLifetimeFromCallingTime 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.*

srmCompactSpace

Input	**srmCompactSpaceRequest**
Output	**srmCompactSpaceResponse**

name	type		Min	Max
srmCompactSpaceRequest	TUserID	authorizationID	0	1
	TSpaceToken	spaceToken	1	1
	TStorageSystemInfo	storageSystemInfo	0	1
	xsd:boolean	doDynamicCompactFromNowOn	0	1
srmCompactSpaceResponse	TSizeInBytes	newSizeOfThisSpace	0	1
	TReturnStatus	returnStatus	1	1

notes:

- *This function is called to reclaim the space for all released files and update space size in Durable and Permanent spaces. Files not released are not going to be removed (even if lifetime expired.)*
- *doDynamicCompactFromNowOn=false by default, which implies that only a one time compactSpace will take place.*
- *If doDynamicCompactFromNowOn=true, then the space of released files will be automatically compacted until the value of doDynamicCompactFromNowOn is set to false.*
- *When space is compacted, the files in that space do not have to be removed by the SRM. For example, the SRM can choose to move them to volatile space. The client will only perceive that the compacted space is now available to them.*
- *To physically force a removal of a file, the client should use srmRm.*

srmGetSpaceMetaData

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

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

srmChangeFileStorageType

Input	**srmChangeFileStorageTypeRequest**
Output	**srmChangeFileStorageTypeResponse**

name	type	Min	Max

srmChangeFileStorageTypeRequest	TUserID ArrayOfTSURLInfo TFileStorageType TSpaceToken	authorizationID arrayOfPath desiredStorageType spaceToken	0 1 1 0	1 1 1 1
srmChangeFileStorageTypeResponse	TReturnStatus ArrayOfTSURLReturnStatus	returnStatus, arrayOfFileStatus	1 0	1 1

notes:

- *Applies to both dir andfile*
- *Either path must be supplied.*
- *If a path is pointing to a directory, then the effect is recursive for all the files in this directory.*
- *Space allocation and de-allocation maybe involved.*

srmGetSpaceToken

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

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

notes:

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

Permission Functions

summary:

srmSetPermission: (applies to both *dir* and *file*)
srmReassignToUser:
srmCheckPermission:

details:

srmSetPermission

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

Name	type		Min	Max
srmSetPermissionRequest	TUserID	authorizationID	0	1
	TSURLInfo	path	1	1
	TPermissionType	permissionType	0	1
	TOwnerPermission	ownerPermission	0	1
	ArrayOfTUserPermission	userPermission	0	1
	ArrayOfTGroupPermission	groupPermission	0	1
	TOtherPermission	otherPermission	0	1
srmSetPermissionResponse	TReturnStatus	returnStatus	1	1

Notes:

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

srmReassignToUser

Input	**srmReassignToUserRequest**
Output	**srmReassignToUserResponse**

Name	type		Min	Max
srmReassignToUserRequest	TUserID	authorizationID	0	1
	TUserID	assignedUser	1	1
	TLifeTimeInSeconds	lifeTimeOfThisAssignment	1	1
	TSURLInfo	path	0	1
srmReassignToUserResponse	TReturnStatus	returnStatus	1	1

notes:

- *After lifeTimeOfThisAssignment time period, or when assignedUser obtained a copy of files through srmCopy(), the files involved are released and space is compacted automatically, which ever is first.*
- *This function implies actual lifetime of file/space involved is extended up to the lifeTimeOfThisAssignment.*

- *The caller must be the owner of the files to be reassigned.*
- *permission is omitted because it has to be READ permission.*
- *lifeTimeOfThisAssignment is relative to the calling time. So it must be positive.*
- *If the path here is a directory, then all the files under it are included recursively.*
- *If there are any files involved that are released before this function call, then these files will not be involved in reassignment, even if they are still in the space.*
- *If a compact() is called before this function is complete, then this function has priority over compact(). Compact will be done automatically as soon as files are copies to the assignedUser. Whether to dynamically compact or not is an implementation choice.*

srmCheckPermission

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

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

notes:

- *When checkInLocalCacheOnly=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 siteURL to check the user permission.*
- *If checkInLocalCacheOnly = false, SRM can choose to always check the siteURL 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 siteURL and return permission.*

Directory Functions

summary:

srmMkdir:
srmRmdir: (applies to *dir*)
srmRm: (applies to *file*)
srmLs: (applies to both *dir* and *file*)
srmMv: (applies to both *dir* and *file*)

details:

srmMkdir

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

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

notes:

- *Consistent with unix, recursive creation of directories is not supported.*
- *newDiretoryPath can include paths, as long as all sub directories exist.*

srmRmdir

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

notes:

- applies to dir
- doRecursiveRemove is false by default.
- To distinguish from srmRm(), this function is for directories only.

srmRm

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

Name	type		Min	Max
srmRmRequest	TUserID	authorizationID	0	1
	ArrayOfTSURLInfo	arrayOfFilePaths	1	1
srmRmResponse	TReturnStatus	returnStatus	1	1
	ArrayOfTSURLReturnStatus	arrayOfFileStatus	0	1

notes:

- *Applies to files*
- *To distinguish from srmRmDir(), this function applies to files only.*

srmLs

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

Name	type		Min	Max
srmLsRequest	TUserID	authorizationID	0	1
	ArrayOfTSURLInfo	path	1	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	ArrayOfTMetaDataPathDetail	details	0	1
	TReturnStatus	returnStatus	1	1

notes:

- *Applies to both dir and file*
- *fullDetailedList=false by default.*
  - *For directories, only path is required to be returned.*
  - *For files, path and size are required to be returned.*
- *If fullDetailedList=true, the full details are returned.*
  - *For directories, path and userPermission are required to be returned.*
  - *For files, path, size, userPermission, lastModificationTime, typeOfThisFile, and lifetimeLeft are required to be returned, similar to unix command ls –l.*
- *If allLevelRecursive=true then file lists of all level below current will be provided as well.*
- *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.*
- *When listing for a particular type specified by "fileStorageType", only the files with that type will be in the output.*
- *Empty directories will be returned.*
- *We recommend width first in the listing.*
- *We recommend that list of directories come before list of files in the return array (details).*

srmMv

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

Name	type		Min	Max
srmMvRequest	TUserID	authorizationID	0	1
	TSURLInfo	fromPath	1	1
	TSURLInfo	toPath	1	1
srmMvResponse	TReturnStatus	returnStatus	1	1

notes:

- *Applies to both dir and file*
- *Authorization checks need to be performed on both fromPath and toPath.*

Data Transfer Functions

---

summary:

srmPrepareToGet:
srmPrepareToPut:
srmCopy:

srmReleaseFiles:
srmRemoveFiles:
srmPutDone:

srmAbortRequest:
srmAbortFiles:
srmSuspendRequest:
srmResumeRequest:

srmStatusOfGetRequest:
srmStatusOfPutRequest:
srmStatusOfCopyRequest:

srmGetRequestSummary:

srmExtendFileLifeTime:
srmGetRequestID:

details:

srmPrepareToGet

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

Name	type		Min	Max
srmPrepareToGetRequest	TUserID	authorizationID	0	1
	ArrayOfTGetFileRequest	arrayOfFileRequest	1	1
	ArrayOf_xsd_string	arrayOfTransferProtocols	0	1
	xsd:string	userRequestDescription	0	1
	TStorageSystemInfo	storageSystemInfo	0	1
	TLifeTimeInSeconds	totalRetryTime	0	1
srmPrepareToGetResponse	TRequestToken	requestToken	0	1
	TReturnStatus	returnStatus	1	1
	ArrayOfTGetRequestFileStatus	arrayOfFileStatus	0	1

notes:

- *The userRequestDescription is a user designated name for the request. It can be used in the getRequestID method to get back the system assigned request ID.*
- *Only pull mode is supported.*
- *SRM assigns the requestToken at this time.*
- *Normally this call will be followed by srmRelease().*
- *"retryTime" means: if all the file transfer for this request are complete, then try previously failed transfers for a total time period of "retryTime".*
- *In case that the retries fail, the return should include an explanation of why the retries failed.*
- *This call is an asynchronous (non-blocking) call. To get subsequent status and results, separate calls should be made.*
- *When the file is ready for the user, the file is implicitly pinned in the cache and lifetime will be enforced.*
- *The invocation of srmReleaseFile() is expected for finished files later on.*

srmPrepareToPut

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

Name	type		Min	Max
srmPrepareToPutRequest	TUserID	authorizationID	0	1
	ArrayOfTPutFileRequest	arrayOfFileRequest	1	1
	ArrayOf_xsd_string	arrayOfTransferProtocols	0	1
	xsd:string	userRequestDescription	0	1
	TOverwriteMode	overwriteOption	0	1
	TStorageSystemInfo	storageSystemInfo	0	1
	TLifeTimeInSeconds	totalRetryTime	0	1
srmPrepareToPutResponse	TRequestToken	requestToken	0	1
	TReturnStatus	returnStatus	1	1
	ArrayOfTPutRequestFileStatus	arrayOfFileStatus	0	1

notes:

- *Only push mode is supported for srmPrepareToPut.*
- *StFN ("toSURLInfo" in the TPutFileRequest) has to be local. If stFN is not specified, SRM will name it automatically and put it in the specified user space. This will be returned as part of the "transfer URL".*
- *srmPutDone() is expected after each file is "put" into the allocated space.*
- *The lifetime of the file starts as soon as SRM get the srmPutDone(). If srmPutDone() is not provided then the files in that space are subject to removal when the space lifetime expires.*
- *"retryTime" is meaningful here only when the file destination is not a local disk, such as tape or MSS.*
- *In case that the retries fail, the return should include an explanation of why the retires failed.*

srmCopy

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

Name	type		Min	Max
srmCopyRequest	TUserID	authorizationID	0	1
	ArrayOfTCopyFileRequest	arrayOfFileRequest	1	1
	xsd:string	userRequestDescription	0	1
	TOverwriteMode	overwriteOption	0	1
	xsd:boolean	removeSourceFiles	0	1
	TStorageSystemInfo	storageSystemInfo	0	1
	TLifeTimeInSeconds	totalRetryTime	0	1
srmCopyResponse	TRequestToken	requestToken	0	1
	TReturnStatus	returnStatus	1	1
	ArrayOfTCopyRequestFileStatus	arrayOfFileStatus	0	1

notes:

- *Pull mode: copy from remote location to SRM. (e.g. from remote to MSS.)*
- *Push mode: copy from SRM to remote location.*
- *Always release files from source after copy is done.*
- *When removeSourceFiles=true, then SRM will remove the source files on behalf of the caller after copy is done.*
- *In pull mode, send srmRelease() to remote location when transfer is done.*
- *If in push mode, then after transfer is done, notify the caller. User can then release the file. If user releases a file being copied to another location before it is done, then refuse to release.*
- *Note there is no protocol negotiation with the client for this request.*
- *"retryTime" means: if all the file transfer for this request are complete, then try previously failed transfers for a total time period of "retryTime".*

- *In case that the retries fail, the return should include an explanation of why the retires failed.*
- *When both fromSURL and toSURL are local, perform local copy*
- *Empty directories are copied as well.*

srmRemoveFiles

Input	**srmRemoveFilesRequest**
Output	**srmRemoveFilesResponse**

Name	type		Min	Max
srmRemoveFilesRequest	TRequestToken	requestToken	0	1
	TUserID	authorizationID	0	1
	ArrayOfTSURL	siteURLs	1	1
srmRemoveFilesResponse	TReturnStatus	returnStatus,	1	1
	ArrayOfTSURLReturnStatus	arrayOfFileStatus	0	1

notes:

- *If requestToken is not provided, then the SRM will do nothing.*
- *It has the effect of a release before the file is removed.*
- *If file is not in cache, do nothing*

srmReleaseFiles

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

Name	type		Min	Max
srmReleaseFilesRequest	TRequestToken	requestToken	0	1
	TUserID	authorizationID	0	1
	ArrayOfTSURL	siteURLs	1	1
	xsd:boolean	keepSpace	0	1
srmReleaseFilesResponse	TReturnStatus	returnStatus	1	1
	ArrayOfTSURLReturnStatus	arrayOfFileStatus	0	1

notes:

- *dir is ok. Will release recursively for dirs.*
- *If requestToken is not provided, then the SRM will release all the files specified by the siteURLs owned by this user, regardless of the requestToken.*
- *If requestToken is not provided, then authorizationID is needed. It may be inferred or provide in the call.*

- *Releasing files will be followed by compacting space, if doDynamicCompactFromNowOn was set to true in a previous srmCompactSpace call.*

srmPutDone

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

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

notes:

- *Called by user after srmPut()*

srmAbortRequest

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

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

notes:

- *Abort all files in this request regardless of the state. Expired files are released.*

srmAbortFiles

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

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

notes:

- *Abort all files in this call regardless of the state*

srmSuspendRequest

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

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

notes:

- *Suspend all files in this request until srmResumeRequest is issued*

srmResumeRequest

Input	**srmResumeRequestRequest**
Output	**srmResumeRequestResponse**

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

notes:

- *Resume suspended files in this request*

srmStatusOfGetRequest

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

Name	type		Min	Max
srmStatusOfGetRequestRequest	TRequestToken	requestToken	1	1
	TUserID	authorizationID	0	1
	ArrayOfTSURL	arrayOfFromSURLs	0	1
srmStatusOfGetRequestResponse	TReturnStatus	returnStatus	1	1
	ArrayOfTGetRequestFileStatus	arrayOfFileStatus	0	1

notes:

- *If* arrayOfFromSURLs *is not provided, returns status for all the file requests in this request.*

srmStatusOfPutRequest

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

Name	type		Min	Max
srmStatusOfPutRequestRequest	TRequestToken	requestToken	1	1
	TUserID	authorizationID	0	1
	ArrayOfTSURL	arrayOfToSURLs	0	1
srmStatusOfPutRequestResponse	TReturnStatus	returnStatus	1	1
	ArrayOfTPutRequestFileStatus	arrayOfFileStatus	0	1

notes:

- *If* arrayOfFromSURLs *is not provided, returns status for all the file requests in this request.*

srmStatusOfCopyRequest

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

Name	type		Min	Max
srmStatusOfCopyRequestRequest	TRequestToken	requestToken	1	1
	TUserID	authorizationID	0	1
	ArrayOfTSURL	arrayOfFromSURLs	0	1
	ArrayOfTSURL	arrayOfToSURLs	0	1
srmStatusOfCopyRequestResponse	TReturnStatus	returnStatus	1	1
	ArrayOfTCopyRequestFileStatus	arrayOfFileStatus	0	1

notes:

- *If arrayOfFromSURLs and/or arrayOfToSURLs are not provided, return status for all the file requests in this request.*

srmGetRequestSummary

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

Name	type		Min	Max
srmGetRequestSummaryRequest	ArrayOfTRequestToken	arrayOfRequestToken	1	1
	TUserID	authorizationID	0	1

srmGetRequestSummaryResponse	ArrayOfTRequestSummary	arrayOfRequestSummary	0	1
	TReturnStatus	returnStatus	1	1

srmExtendFileLifeTime

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

Name	type		Min	Max
srmExtendFileLifeTimeRequest	TRequestToken	requestToken	1	1
	TSURL	siteURL	1	1
	TUserID	authorizationID	0	1
	TLifeTimeInSeconds	newLifeTime	0	1
srmExtendFileLifeTimeResponse	TReturnStatus	returnStatus	1	1
	TLifeTimeInSeconds	newTimeExtended	0	1

notes:

- *newLifeTime is relative to the calling time. Lifetime will be set from the calling time for the specified period.*
- *The number of lifetime extensions maybe limited by SRM according to its policies.*
- *IsExtended = false if SRM refuse to do it. (set newTimeExtended = 0 in this case.)*
- *If original lifetime is longer than the requested one, then the requested one will be assigned.*
- *If newLifeTime is not specified, the SRM can use its default to assign the newLifeTime.*

srmGetRequestID

Input	**srmGetRequestIDRequest**
Output	**srmGetRequestIDResponse**

Name	type		Min	Max
srmGetRequestIDRequest	xsd:string	userRequestDescription	0	1
	TUserID	authorizationID	0	1
srmGetRequestIDResponse	ArrayOfTRequestTokenReturn	arrayOfRequestToken	0	1
	TReturnStatus	returnStatus	1	1

notes:

- *If userRequestDescription is null, returns all requests this user has.*
- *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.*

StatusCode specification

Note:

- Status codes represent errors, warnings and status.

Status code Explanation

SRM_SUCCESS:

- SRM request was successful

Errors:

SRM_FAILURE:

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

SRM_AUTHENTICATION_FAILURE:

- Requester has an invalid authentication information.

SRM_UNAUTHORIZED_ACCESS:

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

SRM_INVALID_REQUEST:

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

SRM_INVALID_PATH:

- The requested file/directory path is invalid.

SRM_FILE_LIFETIME_EXPIRED:

- The life time on the pinned file has expired

SRM_SPACE_LIFETIME_EXPIRED:

- The life time on the reserved space has expired

SRM_EXCEED_ALLOCATION:

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

SRM_NO_USER_SPACE:

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

SRM_NO_FREE_SPACE:

- SRM has not more space.

SRM_DUPLICATION_ERROR :

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

SRM_NON_EMPTY_DIRECTORY:

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

SRM_TOO_MANY_RESULTS:

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

SRM_INTERNAL_ERROR:

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

SRM_FATAL_INTERNAL_ERROR:

- SRM has a severe internal error that cannot be recovered.

SRM_NOT_SUPPORTED:

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

Status:

SRM_REQUEST_QUEUED
SRM_REQUEST_INPROGRESS
SRM_REQUEST_SUSPENDEND
SRM _ABORTED
SRM _RELEASED

SRM_FILE_PINNED

- The requested file is pinned

SRM_FILE_IN_CACHE

- The file is in cache, but not pinned

SRM_SPACE_AVAILABLE

- The requested space is reserved and ready to be used

SRM_LOWER_SPACE_GRANTED

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

SRM _DONE

SRM_CUSTOM_STATUS:

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

Appendix

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-->
