GSM-WG

Storage Resource Managers (SRM)
Design Document for version 3.0

Version 1: 27 December, 2006

Editors: Arie Shoshani and Alex Sim,
Lawrence Berkeley National Laboratory

Contributors:

Contributors:

Timur Perelmutov Don Petravick	Fermi National Accelerator Laboratory (FNAL), USA
Ezio Corso Luca Magnoni	Istituto Nazionale di Fisica Nucleare (INFN), Italy International Centre for Theoretical Physics (ICTO), Italy
Arie Shoshani Alex Sim Junmin Gu	Lawrence Berkeley National Laboratory (LBNL), USA
Olof Barring Jean-Philippe Baud Flavia Donno Maarten Litmaath Peter Kunszt (Now at CSCS)	LHC Computing Project (LCG, CERN), Switzerland
Shaun De Witt 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

Contents

Preface

This design document reflects several years of evolution of a standard specification for Storage Resource Managers. Storage Resource Managers (SRMs) are middleware components whose function is to provide dynamic space allocation and file management of shared storage components on the Grid. This document describes concepts that have emerged as a result of several implementations of SRMs in the context of High Energy Physics (HEP) projects in the USA and Europe, as well as their application to other projects, such as LIGO and Earth System Grid (ESG). Previous implementations are referred to as versions 1.x, and version 2.x. The latest version being implemented at the time of this writing is version 2.2, which has been adopted by the World-wide Large Hadron Collider (LHC) Computing Grid (WLCG) that supports ATLAS and CMS High Energy Physics experiments. The purpose of this design document is to define and explain the new concepts as extensions of functionality of previous versions. This will provide the basis for the specification of the next version v3.0.

Storage Resource Managers complement Compute Resource Managers and Network Resource Managers in providing storage reservation and dynamic information on storage

availability for the planning and execution of Grid jobs. SRMs manage two types of resources: storage spaces and files. When managing space, SRMs negotiate storage space allocation with the requesting client, and/or assign default space quotas. When managing files, SRMs assign files to storage spaces, invoke file transfer services to move files into the space if file are not available locally, pin files for a certain lifetime, release files upon the client's request, and use file replacement policies to optimize the use of the shared space. SRMs can be designed to provide effective sharing of files, by monitoring the activity of shared files, and making dynamic decisions on which files to replace when space is needed. In addition, SRMs perform automatic garbage collection of unused files by removing files that were released by the clients, or whose lifetime has expired when space is needed.

This document describes new concepts of storage space definition and ownership, clarifies lifetime definitions, and summarizes functions to get files into spaces, as well as releasing or removing files from storage spaces. It is organized according to the table of content below. It is intended as the "functional design document" for SRM version 3.0.

1. Storage Areas and Storage Spaces

A Storage Element can have multiple Storage Areas. Each storage area can be specified by the storage area administrator as consisting of one or more storage components. A storage component is specified by its properties which include:
a) Access Latency: online, nearline (see definition in appendix)
b) Retention Policy: custodial, output, replica (see definition in appendix)
c) Size (in bytes)

Any combination of storage components in a storage area is permissible. Examples of storage components are: online-replica (e.g. a common disk space allocated for online access), online-custodial (e.g. a highly protected online disk that may keep multiple replicas), and nearline-custodial (e.g. a high-quality robotic tape system with backup capability). Thus, a storage area can consist of a single storage component (such as online-replica) or multiple storage components (such as online-replica AND nearline-custodial). Storage areas that consist of multiple storage components are referred to as "composite storage areas". Storage areas can share one or more storage components. This allows storage components to be partitioned for use by different user-groups or Virtual Organizations (VOs).

The SRM interfaces expose *only* the storage area, not its components. However, a space reservation to a composite storage area can be made requesting Access Latency-Retention Policy combinations that may effect which parts of the storage components are assigned. Specifically, a space reservation to a composite storage area can request the following combinations to target the online or nearline storage components:
a) online-replica to target the online storage components;
b) nearline-custodial to target the nearline storage components (assuming it support custodial retention policy);

c) online-custodial to target both the online and nearline storage components.

The space associated with the storage area is referred to as the "storage space". The storage area administrator needs to have a space_token in order to have users refer to the storage area, and request space reservation in it. The space in the storage area is initially declared in the SRM using the administrative function srmDeclareStorageArea. The SRM will assign and return a space_token for the space in the declared storage area. Optionally, a storage_space_name can be assigned by the administrator in order to discover the space_token if it is lost, or provide users with a human readable way of referring to the space. The above declaration capability can make it possible for multi-component storage areas to be declared as a single space. This permits the SRM to assume full control of where files end up and even how many replicas to maintain. In contrast, a single-component storage area does not provide such flexibility.

If the storage element manager wishes to make each of the storage components visible to users, each needs to be declared separately to the SRM. For example, a storage element may wish to have three storage areas visible. The first is a global space that includes online-replica and nearline-custodial. The second is a fast online-replica space to be used by a particular VO that paid for it, and the third a slow online-replica space for other VOs. These three spaces can be declared and given any desired name such as Disk-Tape, FastDisk, and OnlineDisk, respectively.

Once the storage area is declared and assigned a space_token, clients can reserve space in it if they are given the privilege to do so. This is discussed in the next section.

2. Space ownership and space lifetime

All the storage areas are initially owned by the storage element administrator, and can be assigned to one or more clients or VOs. For example, a single storage area can be assigned to two VOs, where each is assigned half the space. Each has to be reserved by the VO manager with the srmReserveSpace function where the space_token of the storage area is provided as a parameter, as well as the desired size and lifetime of the reservation. After this operation is performed, the reserved space is assigned a space_token (for the VO to use) and each VO is considered the owner of that space. For example, the space in a storage area that was given the space_name of Disk1Tape1, can be partitioned into two spaces: Disk1Tape1-CMS, Disk1Tape1-ATLAS, using the srmReserveSpace function.

Once a VO acquires a space, it can permit other users to reserved space in it. This is performed by a user with the same srmReserveSpace function, where the VO space_token is provided as a parameter, as well as the size and desired lifetime. If successful, a space_token will be assigned to the user to use. Of course, the requested size and lifetime cannot exceed the size and lifetime of the parent space owned by the VO. Once the space reservation has been performed, the reserved space is owned by the user, but the parent owner has the right to reclaim that space at any time.

Space reservation is hierarchical, in that any user of a space can grant permission to others to reserve space in the space they own. The SRM needs to verify that such reservations do not generate a cycle of ownership, and deny the request in such a case. In addition, it is possible for the owner of a space to reserve space in their own space. This permits a user to partition a space into sub-spaces for control over the use of the space if desired. For example, an experiment manager may wish to partition their space for use by two groups, and even replicate files in the two spaces. The number of levels of space hierarchy supported is a choice of the SRM implementation (or the storage element manager). Similarly, the minimum space size supported is an implementation choice.

An owner of a space can reclaim any of the subspaces at any time by using the srmPurgeFilesFromSpace function by providing the space token of the sub-space. This will be discussed in a later section. The space owner is also allowed to discover the space_tokens of all its sub-spaces with the srmGetSpaceMetadata function.

The size and lifetime that a user can request is an authorization policy that VOMS need to enforce. For the purpose of this document, we assume that VOMS will enforce permissions, size and lifetime. However, if a VOMS is not available, such limitation could be set by the SRM, and the SRM enforce that. Currently, there are no functions for setting space usage permissions and policies. We envisioned that in a future version, ACL support for file ownership and granting permission for the use of the space will be supported.

3. File ownership and file lifetime

3.1 File ownership

A file can have multiple replicas in an SRM. When a file is first put into an SRM, it is assigned an SURL (site URL), and it is placed into some space. A specific SURL can be requested by the owner of the file in a directory structure if the SRM support directory functions; otherwise the SRM assigns it an SURL. That first replica is referred to as the "primary replica". The owner of that primary replica is referred to as the SURL-owner. Thus, by definition, the SURL-owner is the owner of the primary replica. There are three functions that can create new SURLs: srmPerpareToPut, srmRemoteCopy, and srmCp. We will describe each later. We note that the concept of "primary replica" is used in order to prevent accidental removal of the "last replica" of a file. This is enforced by allowing a removal of a primary replica by the SURL owner only with the srmRm function. That function removes the SURL, the primary replica, and all of the secondary replicas.

SRM v3.0 supports ACLs for files. Therefore, an SURL-owner can assign other users as SURL-owners. That original owner can then be removed as an SURL-owner, allowing a simple way to change (add or remove) ownership (for example, when one user wishes to assign files to another user). The SRM has to enforce that an SURL has a least one owner. As pointed above, all SURL-owners are also the owners of the primary replica for that SURL.

Replicas can be made using several functions for different purposes, including srmPrepareToGet, srmBringOnline, and srmAddToSpace. When replicas are created, they are assigned a "replica-owner", who is the client that performed the function that created the replica.

3.2 File lifetime

When a file is first put into some space, it can be assigned a lifetime by specifying the desired "fileLifetime"; otherwise a default lifetime is assigned. That file lifetime is the lifetime of the "primary replica", and is considered by definition the SURLLifetime as well. A file_lifetime is assigned to a file in a storage space when it is first put into that space by a user with the srmPrepareToPut, srmRemoteCopy, and srmCp functions.

After a file is brought into an SRM space, it can be replicated into the same or other spaces with the srmAddToSpace function. When the file is replicated it is assigned a lifetime, either as requested, or by default. The lifetime of a replicated file must always be shorter than the SURLLifetime (i.e. the lifetime of the primary replica). Thus, the SURLLifetime is the longest time of all the replicas, and therefore is considered a property of the SURL in the SRM namespace.

A fileLifetime is also referred to as a "pinLifetime", since it implies that the file replica will be pinned in a space for the duration of the lifetime. FileLifetimes are intended prevent files from clogging storage spaces. When the lifetime expires, files can be removed by the SRM. The only exception is primary replicas that have to be explicitly removed by the owner. Typically, files that are intended to be in the SRM for a long time should have an "indefinite" lifetime (a specific reserved). Of course, files can be explicitly removed before the lifetime expires.

Another way to make a file eligible for removal is to "release the file", or "release the pin" on the file replica. In this case, the file is marked for a possible removal, but is only removed when space is needed by the SRM. Releasing of pins is also a way of managing one own's space. For example, a request for large number of files can be made, but the assigned space is not large enough to hold all the files. In such cases, a fraction of the files can be put in the space, and when they are consumed, the client can "release" the files, so additional files can be brought by the SRM into the "released space".

Only the owner of the SURL can change the SURLLifetime, by changing the lifetime of the primary replica. This can be done with srmAssignNewFileLifetime, including a shorter lifetime than the current lifetime. (Note: this function was previously referred to as srmExtendFileLifetime, but was renamed to allow shorter lifetime assignment). If the SURLLifetime is made longer than the current lifetime, the primary replica's lifetime has to be extended. However, if a shorter lifetime is assigned, the lifetime of all replicas have to be shortened accordingly.

The lifetime of a replica cannot exceed the lifetime of the space it is in. Therefore, when extending a file lifetime, it may be necessary to extend first the space lifetime.

When placing a file in a composite space, only a single lifetime can be provided (i.e. one cannot specify a lifetime for each component), since the composite space is a considered a single entity. The SRM decides where to place the file in order to accommodate the requested lifetime.

4. File placement functions

There a several functions that can place files in storage spaces. srmPrepareToPut is used for placing new files into a space. srmRemoteCopy also places new files into a space by copying it from a remote site. In both cases a new SURL is assigned, as well as its SURLLifetime (i.e. the fileLifetime of the primary replica). srmCp is a function that copies a file that exists in the SRM, but gives it a new SURL, and therefore a new SURLLifetime is assigned as well.

4.1 srmPrepareToPut

When the srmPrepareToPut request is made for multiple files, the SRM may return transfer URLs (TURLs) for some of then. These TURLs are space holders for files to be put into the SRM by the client. Since there is no way for the SRM to know when the file was placed in successfully, it is expecting to get an srmPutDone call.

There is another lifetime concept (not related to the file lifetime) when using srmPrepareToPut. It is the lifetime allocated by the SRM for placement of the file into a space. This lifetime is NOT the lifetime that will be assigned to the file after srmPutDone is issued. In order to avoid confusion, this lifetime is referred to as the "putLifetime". Thus, the srmPrepareToPut function has two lifetime parameters: the putLifetime - the length of time that the SRM will keep the space for the file to be put in, and the fileLifetime - the length of time assigned to the primary replica after the file was put into the SRM).

Once a file has be put into an SRM, it cannot be put in again, except if the "overwriteOption" parameter is set. In this case the file will be overwritten by the new version, as well as all replicas removed. However, existing replicas that are in use remain unchanged if not released or their lifetime expired.

In addition, there are three functions that place replicas in spaces, and therefore keep the same SURL. Since they are replicas, only a shorter lifetime than the SURLLifetime can be assigned to them. These are: srmPrepareToGet, srmAddFilesToSpace, and srmBringOnline. We describe the behavior of each of these three functions next.

4.2 srmPrepareToGet

When files are replicated by an srmPrepareToGet they are said to be pinned in the space. A “pin” of a replica for a lifetime is associated with the request and with the user that issued the request. The SRM may want a file replica to be shared by multiple users. The users have the illusion of having there own replica. Furthermore, a single user can issue multiple requests for the same file (for example, from multiple cluster nodes that access the same space). For this reason, it is important for the SRM to keep the association of a pin on a replica with both the userID and the requestID. This is necessary in order to keep the illusion of multiple replicas, and allow various undo operations, such as releasing files and purging files. Thus, if the same file is requested by two (or more) requests, the file will have two (or more) PinLifetimes. This permits the SRM to share the file by the requests. Even if the requests were from the same user, there will still be multiple PinLifetimes assigned to the replica, because the requests are different. All requests are identified by requestIDs that are assigned by the SRM. A user can provide a requestIDDescription to the SRM in order to recover the requestID in case it is lost.

srmPrepareToGet is a function intended for incremental use of files. It brings files into an online space if the files are not there already and returns Transfer URLs (TURLs) for each file that is already in the space when performing a srmStatusOfGetRequest. A lifetime is assigned to each file as soon as it is placed in the space, and the client can access that file as soon as the TURL is provided. Thus, the end of lifetime for the files can be staggered over time. A space_token can be provided to designate the online space to be used for the request. If the space is not large enough for all files in the request, this function brings in as many files as the space permits. When files are “released” (i.e. the“pin” associated with that request is released) by the user after using them, the released space can be used to bring in additional files.

4.3 srmBringOnline

srmBringOnline is a function that was introduced in order to ask for files that are in a composite space to be brought online for subsequent use. It is similar conceptually to srmPrepareToGet, but unlike srmPrepareToGet no TURLs are returned. Also, it cannot succeed by bringing online some of the files. After all the files are brought online, they are all assigned the requested lifetime, so that all file lifetimes expire at the same time (i.e. the end-of-lifetime is the same for all the files). Consequently, the end-of-lifetime for all the files is determined by the time that the last file in brought in plus the requested lifetime. This is unlike srmPrepareToGet where file lifetime can be staggered.

srmBringOnline can be applied only to a single space that has nearline space as well as online space. When performing this function the SRM is in full control as to where files end up and this information is not visible to the client. For example, the SRM may have multiple online spaces, and it can choose which will be used for each file of the request. Similarly, the SRM can choose to keep multiple online replicas of the same file for transfer efficiency purposes. Once srmBringOnline is performed, subsequent srmPrepareToGet can be issued by clients, and TURLs returned, where each TURL indicates where the corresponding file can be accessed, and the protocol to be used.

Similar to srmPrepareToGet, multiple srmBringOnline can be issued for the same files. Here again, the SRM can provide the illusion of keeping multiple replicas, by keeping track of pins according to userID and requestID.

A target space_token cannot be provided with this function. If one desires files to be brought into another space srmAddFilesToSpace should be used.

4.4 srmAddFilesToSpace

srmAddFilesToSpace is a function intended for getting secondary replicas into a new space. The source space cannot be specified, and it is up to the SRM to select where to get the replica (it could choose to copy the primary replica or to copy another replica if it exists in a more convenient space). A target space_token has to be provided (i.e. it is a required parameter). Conceptually, srmAddFilesToSpace is similar to srmPrepareToPut in that once the file is in a space, a subsequent srmAddFilesToSpace for the same file is not allowed. However, unlike srmPrepareToPut an "overwriteOption" parameter is not available, since only primary replicas can be overwritten.

The execution of this function is considered complete only after all files are brought into the space. If the space is not large enough for all the files, the execution of this function is considered "partial success". Similar to srmBringOnline, this function is intended for making files available for subsequent operations (such as srmPrepareToGet) the end-of-lifetime for all the files must be the same.

5. Releasing, removing, and purging files

There are four functions provided for releasing, removing, and purging files.

5.1 Releasing files

Releasing a file is used to let the SRM know that the client has no need for the replica at the time of the release. Releasing a file is associated with a request only, and therefore we can used the term "releasing the pin" of a file. The SRM may choose to keep the file, especially if the file is shared by another client. Only when all pins are released, can the file be removed if the SRM needs the space.

The most usual case, a client would want to release files associated with a particular request. This can be performed with the srmReleaseRequestedFiles function. If no files are specified, then all files for that request are released, and therefore this function can be used to release all the files in a previous request. This function can be used to release file brought into a space with the srmRequestToGet and srmBringOnline functions.

However, it is also useful to release all files (or selected files) in a particular space regardless of the request the client issued. This is useful for a quick way of releasing all files requested previously by a client from a particular space. This can be performed with

the srmReleaseFilesFromSpace function. In this case, only files that were brought in by the client are released; that is, files for which the client is the pin-owner.

5.2 Removing files

Release files based on requests is not sufficient. It should be possible for SURL owners to remove files from a space, regardless of who brought them in. This can be performed with the srmReleaseFilesFromSpace function. This function will forcefully remove files regardless of whether they are currently pinned by other users. However, if the file to be removed is a primary replica, the function will fails. Instead, an advisory about primary replica files is returned, so that such files can be explicitly removed with the srmRm function. The srmRm function is a namespace function, and as such the SURL will be removed and all the replicas of the file from all spaces they are in. Of course, only the owner of a file can execute this function.

5.3 Purging files

The owner of a space has the power to remove all files from the space regardless of how they were brought into the space and who owns the file replicas. This is referred to as purging files from a space. For this purpose the function srmPurgeFilesFromSpace can be used. However, this function can be dangerous in that a primary replica of the file may be removed by a space owner who is not the SURL owner of the file. Therefore, this function will not remove the primary replica. Here again, advisory about primary replica files is returned, and only the owner will be able to remove the primary replica with the function srmRm.

To summarize, 5 functions for releasing, removing, and purging files are available: srmReleaseRequestedFiles (release requested files for a particular request), srmReleaseFilesFromSpace (release all files by replica-owner regardless of request), srmRemoveFilesFromSpace (remove all replicas of the files by SURLowner), srmPurgeFilesFromSpace (all replicas regardless of file ownership), and srmRm (remove SURL from namespace and all replicas from all spaces).

6. Directory management

The functions srmMkdir, srmRmdir, srmCp, srmMv, and srmRm are intended to have the same behavior of a file system. In SRMs they are considered namespace functions. Thus, Mkdir has only the effect of adding a directory SURL, and srmRmdir removes the directory SURL provided the directory has no files in it. srmCp into a space creates a new incarnation of the file and is given a new (different) SURL, and it is owned by the client who issued the function. Therefore, a new lifetime can be assigned to the file, including a longer lifetime than the original file. srmMv leaves the physical file and all its replicas in the space they reside, but assigns a new SURL to the file. The ownership of the file and its SURLLIfetime do not change. Note that in order to physically move a

file from one space to another and have a new target SURL assigned, an srmCp has to be performed, followed by an srmRm for the source SURL.

7. Semantic rules

- A reserved space cannot exceed the lifetime and size of its parent space.
- The lifetime of a replicated file must always be shorter than the SURLLifetime.
- A lifetime of a file put into a space cannot exceed the lifetime of the space.
- Since srmAddFilesToSpace and srmBringOnline are considered completed only after all files are brought into the corresponding spaces, the end-of-lifetime of all files is set to the end-of-lifetime of the last file brought in. This is not the case for srmPrepareToGet or srmPrepareToPut.

8. New functions and terms since SRM v2.2

Function name changes

- srmAssignNewFileLifetime instead of srmExtendFileLifetime
- srmPurgeFilesFromSpace instead of srmPurgeFromSpace
- srmAddFilesToSpace instead of srmChangeSpaceForFiles
- srmReleaseRequestedFiles instead of srmReleaseFiles (to avoid confusion with srmReleaseFilesFromSpace)

Term changes

- putLifetime instead of desiredPinLifetime, // on TURL
- TFileExpirationMode instead of TFileStorageType (refers to volatile, durable, permanent). Also, new terms for volatile, durable, permanent are introduced: “releaseWhenExpired”, warnWhenExpired, and neverExpire.

New functions

- srmDeclareStorageArea
- srmReleaseFilesFromSpace

Appendix: definition of terms

1) Retention Policy: REPLICA, OUTPUT, CUSTODIAL

- Quality of Retention (Storage Class) is a kind of Quality of Service. It refers to the probability that the storage system loses a file.
- 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.
- Description of Retention Policy Types
    - 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.

2) Access Latency: ONLINE, NEARLINE

- Files may be Online or Nearline. 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. We do not include here “offline” access latency, since a human has to be involved to achieve online latency. For SRMs, one can only specify ONLINE and NEARLINE.
- The type will be used to describe an access latency property that 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.
- The ONLINE cache of a storage system is the part of the storage system which provides file with online latencies.
- Description of Access Latency types
    - 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.

Intellectual Property Statement

The OGF takes no position regarding the validity or scope of any intellectual property or other rights that might be claimed to pertain to the implementation or use of the technology described in this document or the extent to which any license under such rights might or might not be available; neither does it represent that it has made any effort to identify any such rights. Copies of claims of rights made available for publication and any assurances of licenses to be made available, or the result of an attempt made to obtain a general license or permission for the use of such proprietary rights by implementers or users of this specification can be obtained from the OGF Secretariat.

The OGF invites any interested party to bring to its attention any copyrights, patents or patent applications, or other proprietary rights which may cover technology that may be required to practice this recommendation. Please address the information to the OGF Executive Director.

Disclaimer

This document and the information contained herein is provided on an "As Is" basis and the OGF disclaims all warranties, express or implied, including but not limited to any warranty that the use of the information herein will not infringe any rights or any implied warranties of merchantability or fitness for a particular purpose.

Full Copyright Notice