Table of Contents

SRM Implementations

Introduction

This document is a companion to the SRM interface specification and its purpose is to describe how each SRM server (CASTOR, dCache, DPM, StoRM and Bestman) implements the SRM functionalities relevant to WLCG.

Ideally, this document should become the place where to look for answers to questions on the peculiarities of each implementation, on the features actually supported, on the real meaning of status codes, etc. It is intended for SRM developers who like to know how the other implementations behave, to users who need to take into account the difference in the behaviour of different servers and to sites that want to know exactly how SRM interacts with the storage backend. Last but not least, it could serve as a solid basis to start a discussion on the future of SRM in WLCG.

It is unlikely that this document will ever be considered complete, as the topic itself is huge and could be treated with an arbitrarily high level of detail. Therefore, the authors will start concentrating on the most important parts of the SRM functionality set and on the most controversial topics (that is, those which keep being discussed over and over, often just due to a lack of information). Similarly, at least initially, the level of detail will inevitably vary from one implementation to the other, although the final goal is to provide the same level of detail across all of them.

Guidelines

This document does not supersede or change in any way the SRM interface specification. It contains concepts and definitions taken or derived from the specification, as needed to make the text understandable, but without any pretension of completeness. The reader is therefore invited to refer to the specification to find more details on the interface.

Space Management

Concepts and definitions

Interface specification

- **Storage Element**. A Grid element providing storage services; a storage system. It has one or more storage areas.
- **Storage Area**. A set of hardware components (disks, RAID arrays, tape systems, etc.) providing some storage space, having certain properties, like the access latency and the retention policy. This is not an SRM concept.
- **Online cache**. A part of a storage system intended to contain files with an online access latency.
- **Space allocation**. It is a portion of space in a storage area returned by the SRM space reservation mechanism. Space reservation is said to be static, if space allocations can be created and modified only by an administrator, and dynamic if the user is allowed to do that. Often people use (improperly) the term *space token* to refer to a space allocation: the space allocation token is just the unique identifier string of a space allocation.
- **Storage class**. A combination of space properties (access latency + retention policy). Examples relevant to WLCG are: T1D0 (nearline-custodial), T0D1 (online-replica), T1D1 (online-custodial).
- **Retention policy (RP)**. It expresses the probability of losing a file. The possible values (in WLCG) are REPLICA and CUSTODIAL. It is used:
    - as a property of space allocated via the space reservation mechanism;

    - as a property of the files put in a reserved space, automatically set to be equal to the retention policy of the space. **In WLCG, RP as file attribute does not exist.**
- **Access latency (AL)**. It expresses how latency to access a file is improvable (not to be confused with the file locality!). Possible values in WLCG are: ONLINE (no further improvements are possible) and NEARLINE (a file may need to be staged to an online cache). It is a property of:
    - a space allocated via the space reservation mechanism;
    - a file in a reserved space. The access latency of a file is lesser or equal to the access latency of the space. **In WLCG, AL as file attribute does not exist.**
- **File locality**. It is a file property that indicates where copies of the file exist. The possible values are:
    - ONLINE: there is a copy of the file on an online cache which can be accessed with an ONLINE latency;
    - NEARLINE: the file needs to be staged to an online cache before being accessed;
    - ONLINE_AND_NEARLINE: the file has a copy on an online cache and another copy on a nearline storage (tape);
    - LOST: the file is permanently lost;
    - UNAVAILABLE: the file is temporarily unavailable;
    - NONE: the file is empty

The following table illustrates the possible combinations of the above properties in the typical case of a storage element consisting of a tape system and an online cache or of just a disk-based space.

Retention policy	Space access latency	File access latency	File locality	Storage class	Description
CUSTODIAL	NEARLINE	NEARLINE	NEARLINE	T1D0	Space: files need to be staged; the file needs to be staged
CUSTODIAL	NEARLINE	NEARLINE	ONLINE_AND_NEARLINE	T1D0	Space: files need to be staged; the file is staged
CUSTODIAL	NEARLINE	ONLINE	ONLINE_AND_NEARLINE	T1D0	Space: files need to be staged; the file is permanently pinned?
CUSTODIAL	NEARLINE	ONLINE	ONLINE	T1D0	Space: files need to be staged; the file has not yet been copied to tape
CUSTODIAL	ONLINE	ONLINE	ONLINE_AND_NEARLINE	T1D1	Space: files are on tape and on disk; the file is on tape and on disk
CUSTODIAL	ONLINE	ONLINE	ONLINE	T1D1	Space: files are on tape and on disk; the file has not yet been copied to tape
REPLICA	ONLINE	ONLINE	ONLINE	T0D1	Space: files are only on disk; the file is on disk

The specification does not make clear if a file with CUSTODIAL retention policy can be copied to a space with REPLICA retention policy and, in general, if the retention policy and the access latencies of a file are immutable or not, and, when a file has more copies on different spaces with different retention policy and access latency (if at all possible), how the space properties relate to the file properties.

Similarly, it is not clear if the file access latency is a general property of the file, or if it depends (and how) on the file copies. A possible interpretation is that a file's access latency is ONLINE if it is guaranteed that a read operation will not trigger a file stage, and NEARLINE otherwise, but this does not seem consistent with the possibility to specify a space where to copy the file before reading it.

These inconsistencies are resolved by different implementations in different ways, as described below.

Questions to the implementations:

- Is a file allowed to have more than one copy in different spaces?
- Is dynamic space reservation supported?
- What is the relationship between a reserved space and the underlying storage (e.g. disk pools)?
- What kind of access control is applied to space reservation?
- Is the access latency implemented as a space property, a file property or both?
- Can the access latency of a file be different from the access latency of the space containing it?
- If a file has different copies on different spaces, what is the meaning of its retention policy and its access latency?
- What are the possible values for the file locality, and how is it constrained by the file access latency?
- What are the access latency and the file locality of a file before and after a successful BringOnline request?
- Can a file be stored outside of an SRM-managed space?

dCache

In dCache, both static and dynamic space reservation are supported. To describe how they work, it is necessary to briefly describe some dCache-specific concepts.

A *pool* is the elementary file container. Pools can be aggregated in *pool groups*. A transfer request for a file can be of type *read*, *write*, *cache* or *p2p*. A *link* is a rule to determine which pools can be used by a transfer request, depending on its type, the directory containing the file in the namespace or on the client IP. This allows, for example, to dedicate different pools for reading and writing, or for different parts of the namespace. Finally, a *link group* is a set of links, which also translates to a set of pools. A link group has some attributes which relate to the retention policy and the access latency for the files it contains, namely `custodialAllowed`, `outputAllowed`, `replicaAllowed`, `onlineAllowed` and `nearlineAllowed`.

How does all this map to SRM? Let us assume that a dCache installation has a certain number of pools, all of which belonging to one and only one link, all of which belonging to one and only one link group. Each link group then is a logical partition of the storage, configured to contain files with a certain retention policy and access latency.

When the user or the administrator creates a space allocation, she will create it in a link group. In the case of an administrator, she can explicitly choose on which link group to create the space allocation using an administrative interface; this is what is normally done in WLCG sites. In the case of a user, she can use *srmReserveSpace* to allocate some space, and an appropriate link group will be automatically selected. Permissions to allow or disallow space reservation are VOMS-based.

In dCache, the retention policy and the access latency of a file are inherited from those of the space where the file is written the first time. In practice, they can be set in two ways:

1. specifying them as input arguments to *srmPrepareToPut* or *srmCopy*; in this case an appropriate space allocation will be automatically selected if the "implicit space reservation" functionality is enabled in dCache;
2. specifying a space allocation token as input argument to *srmPrepareToPut* or *srmCopy*.

If neither a space allocation token and the RP and the AL are given, dCache will look for a RP and an AL in some `pnfs` tags for the file directory, and if missing, will use some globally configured defaults. This allows to couple the namespace to the space allocations. If the "implicit space reservation" is turned off and no space allocation is explicitly specified, the file will be written in a pool not belonging to any link group.

Something to be kept in mind is that dCache allows to specify a space allocation only for write operations. This means that, when accessing or staging a file, it is not put on any space allocation, but on some pool not belonging to any link group and reserved for read or cache transfer requests

It is important to note how dCache deals with the concept of space and file copies. From the SRM point of view, a file can have only one copy, residing in the space allocation where the file was initially written, and it is there until it is physically on disk: if it is garbage-collected after being migrated to tape, for example, the space it occupies will be freed, and if the file is staged back, it will go to some dedicated non-SRM disk pool. Only if the space where the file is put has access latency ONLINE, the file will permanently stay on the same pool. dCache can still replicate a file on many pools to improve performance, of course, but this is totally transparent from the SRM perspective.

CASTOR

To understand how SRM concepts are mapped to CASTOR concepts, it is necessary to give a few definitions.

- **Disk pool**. It is a collection of disk servers whose space is aggregated and can be seen as a single "physical" partition. It corresponds to the concept of Storage Area.
- **Service class**. It ties together a disk pool, tape resources and the policies for how those resources should be used. More service classes can share the same disk pool. In that case, CASTOR reports the same used and free space for all the service classes in a pool, being equal to those of the pool. A service class has its own garbage collector, which means that if several service classes share the same pool, several garbage collectors act on the same pool. It corresponds to the concept of SRM space allocation (a static one). A file is not permanently linked to a service class, and may exist on different service classes at the same time. A service class can be configured to migrate files to tape by specifying an associated tape pool and a migration policy.
- **Stager**. It is a software system which manages files on a pool of disk servers, and transfers (stages) those files to and from the tape system.
- **File class**. It is an attribute of a path in the CASTOR namespace. Its main purpose is to determine if a file, or files in a directory should have a copy on tape. For migration to happen, though, the file must exist in a service class set up for migration.

Let's see how CASTOR interprets some of the fundamental SRM concepts.

- **File access latency**. If specified in an *srmPut* request, it indicates if the user wants to use a space with garbage collection. For *srmLs* requests it indicates that a copy of the file is ONLINE somewhere, or (ONLINEAND)NEARLINE if not.
- **Space access latency**. It is NEARLINE if the garbage collector of the service class corresponding to the space is enabled; it is ONLINE if it is disabled.
- **File retention policy**. If it is CUSTODIAL, it means that the file is on tape; it is REPLICA otherwise. The fact that a file should be migrated to tape cannot be set via SRM, but it is inherited from the space (i.e. service class) where the file is written; it can also be derived from the file class of the file or of the directory containing it in the namespace. The retention policy is a property of the file itself, not of a particular copy of it on a space, because it relates to its existence on tape.
- **Space retention policy**. It is CUSTODIAL if the service class corresponding to the space is configured to migrate to tape all files in it.
- **File locality**. It is ONLINE if the file exists only on disk, ONLINE_AND_NEARLINE if the file exists on disk and on tape and NEARLINE if the file exists only on tape.

In CASTOR it is perfectly possible to have a file in more than one space allocation (service class). This does not create any inconsistency, as the file retention policy depends only on the existence of a copy of the file on tape, and the file access latency is simply not defined.

A fundamental difference with dCache is that in CASTOR a user can choose the space allocation where to bring online a file, and a file is brought online on a space allocation.

DPM

a pool is a set of file systems

a space can be any subset of a pool, cannot span pools

a space can be either allocated by the admin, or by a normal user

it is possible to have multiple copies of a file in the same space or in different spaces, but are not automatically generated. a replication is done using dom-replicate, but files cannot have several instances in the same file system. it is impossible to create more copies with srm. for next version one will be able to say in the directory that files in that directory should have a given number of copies.

Dynamic space reservation is supported according to the specs. A user can only do a reservation for himself or the group he is member of. An admin can do a reservation for any user or group. The permission to create a space reservation in a pool depends on the primary FQAN. JP will check who is getting permission to write into a space reservation. dpm-update-space can be used to change permissions on spaces (decide who can write to a space, etc.).

AL is a property of the space. In the current version of DPM, when a file is in DICOM one can register a file in DPM with an AL of NEARLINE, but in any other case the AL is ONLINE.

DPM can create files with a limited lifetime. There is a lifetime of a file, and the lifetime of a copy of a file in a space (which cannot be longer than the former).

Retention policy: one can have different copies of a file with different RP. It is a property of the space.

There is the concept of the primary copy and secondary copies of the file. ChangeSpaceForFiles was supposed to change the primary copy, but it does not work for DPM (the code is there but not in production).

In the file is in DICOM but not in the disk cache, the locality is NEARLINE, if it is also on DPM it is NEARLINE_AND_ONLINE, if it is only on disk the locality is ONLINE.

Methods

srmReserveSpace

Questions to the implementations:

- Is the method supported?
- Is it synchronous or asynchronous?
- How does it work?
- Are *desiredSizeOfTotalSpace* and *desiredSizeOfGuaranteedSpace* taken into account, and how?
- Is *desiredLifetimeOfReservedSpace* taken into account?

DPM

The method is supported and is synchronous. Only *desiredSizeOfGuaranteedSpace* is taken into account. Spaces with limited lifetime can be allocated.

CASTOR

Not supported.

srm_ReleaseSpace

Questions to the implementations:

- Is the method supported?
- How does it work if there are files inside the space?

CASTOR

Not supported.

DPM

It is supported. Files whose lifetime has not yet expired, will stay.

srmUpdateSpace

Questions to the implementations:

- Is the method supported?
- Which properties can be changed: total space, guaranteed space, lifetime?

CASTOR

Not supported.

DPM

Supported according to specs.

srmGetSpaceMetaData

Questions to the implementations:

- Is the method supported?

DPM

Works according to specs. Can get info only on spaces for which the user is authorized (?).

srmChangeSpaceForFiles

This method allows to move a file from a space to another without changing its SURL. In WLCG, however, this method does not need to be implemented, as it can be replaced by invoking a *srmBringOnline* or a *srmPrepareToGet* with a target space allocation and a *srmPurgeFromSpace*.

Questions to the implementations:

- Is the method supported?
- How does it work?

CASTOR

Supported

DPM

Not in production.

srmExtendFileLifeTimeInSpace

Questions to the implementations:

- Is the method supported?

DPM

Works.

CASTOR

Not supported.

srmPurgeFromSpace

This method removes a list of files from a given space. By definition, it cannot remove the last copy of a file (for that, *srmRm* must be used). Questions to the implementations:

- Is the method supported?
- How does it work?
- Does it return SRM_LAST_COPY if the copy to be deleted is the last one?

CASTOR

DPM

Works.

srmGetSpaceTokens

Questions to the implementations:

- Is the method supported?

Works according to specs. Can get info only on spaces for which the user is authorized (?).

File staging

Concepts and definitions

One of the most important functionalities of SRM is the ability to "stage" files to disk and make sure that they will stay online for a certain period of time. It is performed by submitting a request via A_srmBringOnline_; the status of the request is queried via *srmStatusOfBringOnline*.

Methods

srmBringOnline

srmBringOnline has the following inputs (some are omitted because not so relevant):

- a list of SURLs
- the desired total request time (the time after which, if the request has not completed, it will time out)
- the desired lifetime of the copy created
- a target space allocation, or
- the desired retention policy for the file copy created (the desired access latency must be ONLINE)
- an access pattern (TRANSFER_MODE or PROCESSING_MODE (default)) and a connection type (WAN or LAN (default))

It will produce the following outputs:

- a return status
- a request token
- for each file in the request:
  - a return status
  - the file size
  - the estimated wait time (irrelevant according to WLCG MoU)
  - the remaining pin time
- the remaining request time (if the value is zero, the request has been timed out)

The specification allows to specify the desired RP and AL for the file when it is brought online and says that, if a target space allocation is specified, its RP and AL must match the ones desired for the file. It also says that if the request is successful, the file access latency must be ONLINE. This is rather contradictory, as it is quite possible to stage a file on a NEARLINE space, which would imply the file AL to be NEARLINE, not ONLINE; it looks like a confusion with the file locality. This is why WLCG has decided to disregard the AL as a file attribute and to consider the input AL and RP in a *srmBringOnline* request as space properties, used to select an appropriate target space allocation.

CASTOR

In CASTOR, this request will cause a stage-in request to be sent to the CASTOR stager, very much like the `stager_get` command. The request status is updated by the SRM server by regularly querying the stager every 30 s; if no changes are seen, the polling time increases exponentially until it reaches one hour, at which point the request finishes (files not stages are marked as "failed"), if >0 the request will have SRM_PARTIAL_SUCCESS (instead of SRM_REQUEST_TIMED_OUT as it should).

The desired total request time is ignored, and the remaining request time is set to NULL.

The desired lifetime is used to set the expiration time for the online copy of the file. The remaining pin time is then defined as the time left to the expiration time, and may or may not have an effect on the probability of the file to be deleted from disk, depending on the configuration of the garbage collector for the space allocation where the file resides.

The estimated wait time is initially set to 5. For file requests pending or in progress, it is set to the time between the last and the next status check of the file status by the SRM server. This does not provide the client with an actual estimation of the time for the file to be staged, but at least allows the client to avoid polling for the file status more often than needed. The estimated wait time is the same for all files in the request.

In CASTOR it is possible to specify a target space allocation; if it is not and the file is not already ONLINE, a default pool will be chosen. The input retention policy and access pattern are ignored (to check).

- does it use the desired total request time?

- when does a request time out (SRM_REQUEST_TIMED_OUT)? Not returned by CASTOR, PARTIAL SUCCESS
- does it use the desired lifetime?
- what if a target space token is not specified? If the file is already on disk, nothing is done, otherwise a default pool is used
- what is a retention policy is specified?
- does it use the access pattern? Yes, to select the default pool if the target space token is not specified
- does it return an estimated wait time? Yes
- does it return the remaining pin time? Yes
- does it return the remaining request time? No

-- AndreaSciaba - 21 Jul 2009

This topic: LCG > SrmImplementations
Topic revision: r6 - 2009-12-18 - AndreaSciaba