Addendum to the SRM v2.2 WLCG Usage Agreement

To: CCRC'08 Storage Solutions Working Group

From: Flavia Donno

Date: 6/10/2008

Version: 1.3

Contributors/Authors:

Storage Providers: O. Barring, G. Behrmann, S. De Witt, P. Fuhrmann, I. Kozlova, D. Litvintsev, G. Lo Presti, L. Magnoni, P. Millar, T. Mkrtchyan, G. Oleynik, D. Smith, T. Perelmutov, M. Radicke, O. Synge, R. Zappi

Data Management Client Providers: A. Frohner, R. Mollon, P. Tedesco

Experiment representatives:

ATLAS:

CMS: B. Bockelman

LHCb: N. Brook, P. Charpentier, A. Smith

Purpose of this document

This note summarizes the agreed client usage and server behavior for the SRM v2.2 implementations used by WLCG applications. The agreement encompasses the clients:

- FTS (WLCG official client)
- GFAL (WLCG official client)
- lcg-utils (WLCG official client)
- dCache srm clients
- StoRM srm clients

and storage providers:

- CASTOR
- dCache
- DPM
- StoRM

The agreement shall be focused on meeting the LHC experiments' requirements for Grid storage interfaces. The note specifies how the various existing methods and data objects, as they are specified in the agreed v2.2 specification [1], should be interpreted and/or extended in order to meet the LHC requirements. The outlined WLCG interpretation of the interfaces may suggest sometimes a more restrictive or extended use of the specification than other implementations of SRM client and servers although care is taken to preserve interoperability with the latter.

The document also specifies what is required in the medium-term (end of 2008) and what is required in the long-term.

NOTE: This document has been agreed from a technical point of view by the developers of the storage services. Such an agreement does not imply a commitment to implement the following specification. The implementation of the needed features depends on availability of funds and human resources for development and support. It is up to the WLCG Management Board to address the issues connected to man power.

NOTE: The short term solution will allow this group to better understand the experiments requirements and the feasibility of the long term solution proposal. In case the short term solution demonstrates to already address the requirements and use cases of the experiments, then this group might decide to either not implement or partially implement or revise the full proposal.

This document is open for discussions/corrections/comments/addition.

1. WLCG requirements

In this section we describe the experiments' requirements that lead to the definition of this Addendum to the SRM v2.2 WLCG Usage Agreement [5]. The requirements are listed in order of the priority given by the LHC experiments, from the most important and more urgent to implement to the less urgent.

A. ***Protecting spaces from (mis-)usage by generic users***
Some implementations do not allow at the moment to efficiently protect the spaces from mis-usage by generic users. As an example, it is possible that a generic VO user releases the space allocated to a VO. A request to read a file can trigger either a stage operation from tape in pools dedicated to production activities or internal copies between pools with no possibility to control such actions. Some attempt has been made to protect the storage resources but a consistent approach to space protection would be beneficial.

B. ***Full VOMS-awareness of the SRM implementations***

Some SRM implementations are still not VOMS-FQAN aware at the authorization level. Even though authentication takes place at the SRM server level using GSI and recognizing VOMS groups and roles, VOMS-FQANs are not used for authorization. This creates problems in managing the resources allocated to a VO and forces the use of specific proxies to execute certain tasks.

Definition: ***Fully VOMS-awareness*** is an implemented access control based on VOMS FQANs, including groups and roles.

C. ***Selecting spaces for read operations in srmPrepareToGet, srmBringOnline, and srmCopy requests.***

The SRM v2.2 WLCG Usage Agreement specifies that clients must not specify a token for read operations. However, the document assumes silently that the file will be retrieved somehow in the "right" place. The SRM servers at the moment implements different behaviors. dCache serves the file from the place it was put if an online copy exists and is accessible by the process that requested it. Otherwise either a pool-to-pool automatic copy operation is triggered to make a copy of the file available in the pool where the process requesting it has access or a stage from tape operation can be triggered. This behavior can interfere with production operations. CASTOR honors the token if passed to select the pool where the copy of the file should be read from. This complicates the task to manage the space and to control the access. The possibility to specify a token for read operations and have the server properly keep it into account must be given.

D. ***Correct implementation of srmGetSpaceMetaData***

srmGetSpaceMetaData should return information about the space used and available (as defined in [4]) for a given space token. The concept of a default area is not defined in the SRM spec [1]. Some implementations might provide an implementation specific definition of a default that can serve different purposes.

E. ***Providing the necessary information so that data could be efficiently stored on tapes.***

It is absolutely important to efficiently store data on tape sets so that data can be retrieved efficiently, minimizing the number of tape mounts to be executed. The SRM interface and WLCG clients should allow applications to pass all necessary information to perform the operations efficiently. Tokens should be passed to the tape backend as well as directory paths and any other useful info.

2. WLCG SRM data model interpretation agreement

In what follows, we detail the interpretation of the SRM v2.2 concepts that were found to be controversial during the experience acquired operating an SRM v2.2 based Grid infrastructure.

2.1 The SRM space

Definition: An ***SRM SPACE*** is a logical view of an ***online*** physical storage allocation that is reserved for read/write operations on files.

An *SRM SPACE* is characterized by several properties:

- Retention Policy Information (Retention Policy and Access Latency)
- Owner
- Connection Type (WAN , LAN)
- Supported File Access/Transfer Protocols
- Space Token
- Space Token Description (optional)
- Status
- Total Size
- Guaranteed Size
- Unused Size
- Assigned Lifetime
- Left Lifetime
- Client Networks

In what follows we refer to an SRM SPACE as “SPACE”.

In WLCG the concept of the “Owner” of a SPACE is not used.

In WLCG SPACEs are statically reserved, although support for truly dynamic SPACE reservation can be provided by some implementations.

The same SPACE token description can be assigned to multiple SPACE tokens.

When the copy of a file is removed or purged from a SPACE the physical blocks occupied by the physical copy of the file are not accounted against the SPACE reservation and can be therefore re-used. There must not be an expectation that the blocks occupied by that copy are immediately released.

NOTE: The behavior of the srmRm operation must **NOT** be re-discussed. The operation must remain synchronous.

SPACEs can have other implementation specific properties, such as the “supported operations” in dCache. In order to benefit of possible system optimizations, the client can specify further properties using the TExtraInfo SRM structure in srmReserveSpace, or the service administrator can manually configure the SPACE in the “optimal way”.

NOTE: The SRM v2.2 spec does not define the “DEFAULT” space (even though the concept is somehow implied). An implementation of a DEFAULT area could be provided by some storage services. However, it must be clear that this is implementation specific and the resulting functionality might not be the same across implementations. It is up to the service administrator to appropriately configure such a feature in agreement with the experiments.

In WLCG only three types of SPACEs are supported:

- T1D1 where Retention Policy = CUSTODIAL, Access Latency = ONLINE
  This implies that when a copy of the file is put in this SPACE, automatically a copy is created on the connected tape system as well.
- T1D0 where Retention Policy = CUSTODIAL, Access Latency = NEARLINE

This implies that when a copy of the file is put in this SPACE, a copy is automatically created on the tape system. The copy on this SPACE can be garbage collected if not pinned (see later)

- T0D1 where Retention Policy = REPLICA, Access Latency = ONLINE
  This implies that when a copy of the file is put in this SPACE, it stays there until a privileged process requires its removal through either an srmPurgeFromSpace operation or through an srmRm.

2.2 SRM files, copies and TURLs

Definition: A ***FILE*** is a set of data with the properties defined on pages 17-18 of [1], paragraph 2.25:

- SURL
- Path
- Size
- Creation Time
- Modification Time
- Storage Type (PERMANENT is the only allowed value in WLCG)
- Retention Policy Info (Retention Policy, Access Latency)
- File Locality (ONLINE, NEARLINE, ONLINE_AND_NEARLINE, LOST, NONE[=0 size], UNAVAILABLE [temporary hardware failure])
- Array of Space Tokens
- File Type (File or Directory)
- Assigned Lifetime
- Left Lifetime
- Permissions
- Checksum Type
- Checksum Value
- Array of Sub Paths (if the file is a directory)

The Retention Policy Info attribute for files is not used in WLCG.

The concept of a *file primary copy* implied in the SRM v2.2 spec is not used in WLCG.

A FILE can have several COPYs in several spaces.

Definition: A ***COPY*** of a file is a logical view of a physical instance of the file in a given SPACE. It is characterized by the following properties:

- User's DN (the DN of the user who has instantiated this copy)
- Request ID (of the request generating the copy: srmPrepareToPut, srmPrepareToGet, srmBringOnline)
- SPACE Token (of the space where the copy resides – this is optional)
- PinLifetime (the time the copy is guaranteed to be in the online space where the copy resides – it is defined in the srmBringOnline). The PinLifetime of a copy suggests to the system that the copy is needed by the application and therefore such a copy should not be garbage-collected while its PinLifetime is still valid. At the moment srmPrepareToPut and srmCopy do not allow to specify the PinLifetime for the resulting created copy.

In what follows we refer to an SRM COPY simply as COPY

A COPY can be associated to many physical instances of the FILE. Multiple COPYs can be associated to the same or to several physical instances of the FILE.

Definition: A ***TURL*** is the transport URL associated to a COPY of a FILE in a SPACE. It is characterized by the following properties:

- TURL (transport URL associated to a file access/transfer protocol)
- User's DN (the DN of the user who has instantiated this TURL)
- Request ID (of the request generating the copy: srmPrepareToPut, srmPrepareToGet)
- SPACE Token (of the space where the copy resides – this is optional)
- PinLifetime (the time for which the TURL must remain valid)

A TURL is generated by the system after a srmPrepareToGet or srmPrepareToPut request and is associated to the request that has generated it. A TURL refers to one COPY of the same file and it might be associated to one or more different physical copies of the same file.

In Figure 1 we show in the UML diagram the relation between FILE, COPY, and TURL. Figure 2 shows the relationship between physical copies, COPYs and TURLs.

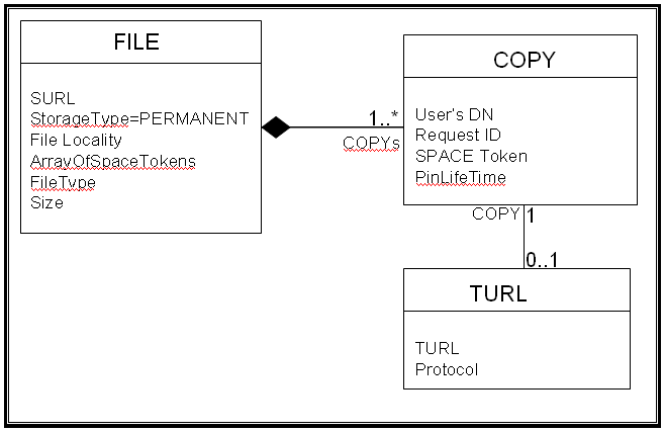

Figure 1: The relationship between FILE, COPY, and TURL

A TURL has an associated pin-lifetime. A TURL is valid while the associated pin is valid. If one of the TURLs associated to a copy of a file is released, the other TURLs whose PinLifetime has not yet expired will continue to stay valid.

Pins can be treated as advisory by the storage systems or they can be enforced. If a pin is enforced, the storage system will refuse to remove a pinned copy (a copy/TURL with a pin still valid) and therefore abort further requests if the space where the copy is in is full. If a pin is advisory, the storage system might consider as a factor the pin lifetime of a copy when running the garbage collector. However, if new requests for space arrive, the requests are honored removing files with the lowest weight.

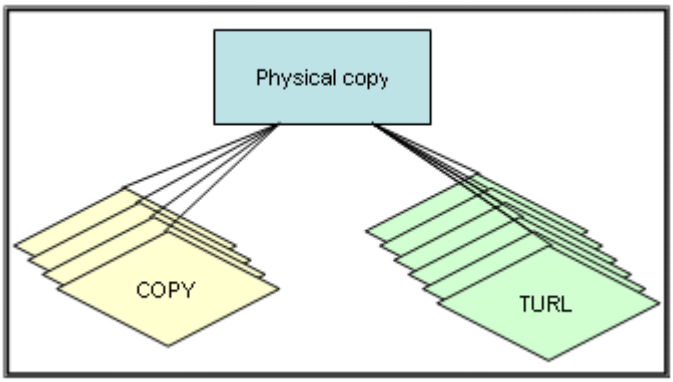

Figure 2: Relationship between physical copies, COPYs and TURLs.

It has been noted that at the moment it is not possible to retrieve the request ID and the request type, given a TURL. This must be discussed when the next release of SRM will be proposed.

3. WLCG SRM v2.2 functionality interpretation agreement

In what follows, we detail the interpretation of the SRM v2.2 functionality that were found to be controversial during the experience acquired operating an SRM v2.2 based Grid infrastructure.

3.1 srmRm

When a FILE is removed, the bytes occupied by the physical copies of that FILE are not accounted against the SPACE reservation where the corresponding COPYs/TURLs are and those bytes can be re-used. There must not be an expectation that the bytes occupied by those physical copies are immediately released.

However, it has been stressed that no requests for released space should fail. A srmPrepareToPut/srmStatusOfPrepareToPut cycle should return successfully only when the space has been made available.

However, it has been stressed that the behavior of the srmRm operation must not be re-discussed later and the method must stay synchronous.

In order to perform a srmRm operation, the client must have privileges to perform the operation in the SRM namespace, regardless of any restrictions on the SPACEs in which COPYs/TURLs of the FILE reside.

3.2 srmLs

In order to perform a srmLs operation, the client must have privileges to perform the operation in the SRM namespace only.

In WLCG, the return attribute *arrayOfSpaceTokens* is mandatory and is only supported on a single FILE. It is not to be returned for directories.

The srmLs method returns an empty list of SPACE tokens for FILEs for which the COPYs are in areas that are non-SRM managed. No SPACE tokens are reported for COPYs that are offline.

Comment from LHCb: It is highly desirable to have an SRM method that given a SPACE token returns the associated SPACE token description. This should be provided by later versions of the SRM interface.

3.2.1 srmPurgeFromSpace

Please note that this method was not part of the original WLCG SRM v2.2 Usage Agreement [5].

In WLCG no tape transitions are allowed. This means that tape transitions cannot be performed using srmPrepareToGet with a SPACE token and performing a srmPurgeFromSpace on the original SPACE.

The srmPurgeFromSpace method only removes disk copies of a FILE from a specified SPACE token.

If a COPY of a FILE is in a Custodial-Nearline SPACE and at a given moment the FILE has a physical copy on disk and one on tape, a srmPurgeFromSpace operation removes only the copy on disk and leaves the copy on tape.

If a FILE has 2 COPYs in 2 different Replica-Online SPACEs, a srmPurgeFromSpace on the first SPACE succeeds while a srmPurgeFromSpace on the second COPY fails with the error message SRM_LAST_COPY at the file level. An explicit srmRm is needed to remove the last COPY.

In point b) of the SRM v2.2 specification describing srmPurgeFromSpace there is written:

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

This must be interpreted as the client having "Purge-From-Space" privilege on the SPACE where the COPY to be purged is. See later the chapter on SPACE ACLs.

3.2.2 srmBringOnline

The current SRM v2.2 specifications are unclear in paragraph 5.3.2, point g). The SPACE in which the FILE is brought online can have the attribute NEARLINE (as well as ONLINE). In WLCG we have decided to disregard the Access Latency attribute of a FILE (which, after a successful srmBringOnline operation would be ONLINE).

3.2.3 srmPrepareToPut/srmPutDone

The current SRM v2.2 specifications do not specify as input parameters of the srmPrepareToPut method the PinLifetime of the COPY of the FILE created after the correspondent successful srmPutDone operation. Such a PinLifetime is set at the moment to some system specific default.

In order not to change the SRM v2.2 WSDL (which would require updates both on client and server side, together with changes in the published endpoints), in WLCG we decided to use the TExtraInfo structure. In order to pass such a parameter, the key name to be used is:

- CopyPinLifetime

The current PinLifeTime parameter in srmPrepareToPut call has to be interpreted as the PinLifeTime of the TURL generated after a successful srmPrepareToPut operation. The srmPutDone call for that TURL has to come within the TURL PinLifetime. This is explicitly defined in point 5.5.2k of the SRM v2.2 spec [1].

3.2.4 srmCopy

The current SRM v2.2 specifications do not specify as input parameter of the srmCopy method the source SPACE token and the PinLifeTime of the resulting COPY. Please, note that the PinLifetime of the resulting COPY cannot be specified since srmPrepareToPut does not allow client to specify it.

In order not to change the SRM v2.2 WSDL (which would require updates both on client and server side, together with changes in the published endpoints), in WLCG we decided to use the TExtraInfo structure. In order to pass such parameters, the key names to be used are:

- SourceSpaceToken
- CopyPinLifetime

It is not possible to release COPYs created by srmCopy based on request ID.
It is also not possible to release COPYs created by srmPrepareToPut/srmPutDone cycle based on request ID.

3.2.5 srmReleaseSpace

In WLCG all FILES are of StorageType PERMANENT.

From the current SRM v2.2 specs, it is not clear what action must be performed when a SPACE is released and there are last COPYs of PERMANENT FILEs in there.

Even though, this is a minor case in WLCG, the behavior must be agreed and clarified.

3.2.6 srmReleaseFiles

The srmReleaseFiles allows users to release pinned COPYs of FILEs. Only the COPYs created by the same DN issuing the srmReleaseFiles request are released. The corresponding physical copies can then be garbage collected. Figure 3 shows the behavior of srmReleaseFiles.

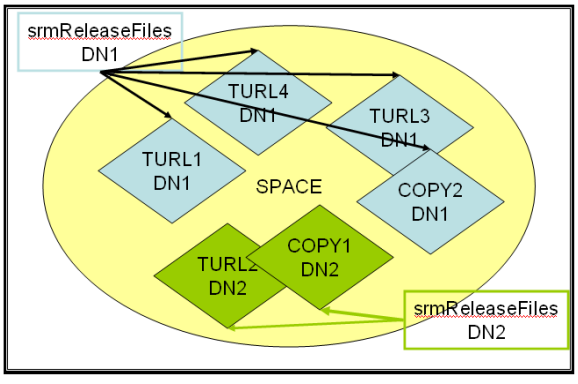

Figure 3: srmReleaseFiles behavior

For the reprocessing task, LHC experiments requires the possibility to specify only SURLs (and not the associated active request IDs) when releasing copies or TURLs of files. Therefore, in WLCG this feature must be implemented by all storage services.

NOTE: Experiments are asked to comment if such functionality is required providing valid use cases for it. Experiment name and contact person requesting such a feature will be noted. If such a functionality is required, how long can the experiments live without it?

LHCb: N. Brook, P. Charpentier, A. Smith – Releasing copies without specifying the request ID is useful when jobs fail. In this case, the request ID is lost with the job. However, it is important to be able to release all copies/TURLs associated to a given DN. Another point is that it is difficult to build persistency for the request ID in the LHCb production and analysis frameworks.

4. Some background on security

In this section we give some background information on VOMS groups and roles, VOMS FQANs and VOMS FQAN based Access Control Lists. We would also like to point out to the document in [3] where recommendations for changes in gLite authorization are given and suggestions on how to implement security in Storage and Data Management.

4.1 VOMS Groups, Roles, and Access Control Lists

Every user is presenting a VOMS proxy when using the WLCG Grid. In the context of this document a simple grid proxy is equivalent to a VOMS proxy with the (single) VO being the only extra attribute (determined from a grid-mapfile). A proxy is first characterized by the Subject Distinguished Name (DN), and can have extensions that define the privileges of the user holding that proxy at a given moment. Please check [2] for details. Example DN:

(1) /DC=ch/DC=cern/OU=Organic Units/OU=Users/CN=flavia/CN=388195/CN=Flavia Donno

To define the privileges of a user at a given moment, groups, subgroups, and roles can be defined. In particular, a user can belong to multiple groups and sub-groups and have a number of roles at a given time. Example of groups and roles:

(2) /dteam/Role=lcgadmin
(3) /dteam
(4) /dteam/cern

An Access Control List (ACL) is a list of entries defining the authorization on a given resource. ACLs can be positive, i.e. defining who is authorized to perform a given set of operations or access a given resource, or negative, negating permission to the service. An example of a DPM positive ACL on a file follows:

(5) # file: /grid/dteam
owner: /DC=ch/DC=cern/OU=Organic Units/OU=Users/CN=flavia/CN=388195/CN=Flavia Donno
group: dteam

```
user::rwx
group:: rwx
group:dteam/Role=lcgadmin:rwx
group:dteam/Role=production:rwx
mask::rwx
other::r-x
default:user::rwx
default:group:rwx
default:group:dteam/Role=lcgadmin:rwx
default:group:dteam/Role=production:rwx
default:mask::rwx
default:other::r-x
```

5. WLCG proposed extensions

We specify here the properties of the SRM data objects that are being considered for addition to the SRM protocol.

5.1 Extensions of space properties

The following operations are allowed on a SRM SPACE:

- Release-Space, Update-Space, Read-from-Space, Write-to-Space, Stage-to-Space, Replicate-from-Space, Purge-from-Space, Modify-Space-ACL, Query-Space

A service administrator is the person who can change the system configuration for a storage service. The service administrator has all rights on all created spaces.

The initial requestor of a space has automatically all rights on the space.

Permissions on the spaces are expressed in terms of DNs and/or VOMS FQANs [2].

ACLs define the set of operations that a user whose FQAN matches the space ACLs can perform on the SPACE. If no match is found for a user proxy in the SPACE ACLs, then the default action is to deny access.

ACLs apply to space tokens and **not** to SPACE token descriptions.

The set of possible operations are:

- ***Release-Space*** : specifies which users/groups can release the space. The Purge-Space right is not needed to release SPACEs.
- ***Update-Space*** : specifies which users/groups can update the SPACE, such as modifying its size, lifetime, etc.
- ***Read-from-Space*** : specifies which users/groups can perform srmPrepareToGet, srmCopy, and srmBringOnline operations on this space. The operation succeeds if it does not imply/trigger a staging request from tape or an internal disk-to-disk copy. In fact, for this last case, the user/group must also have the Replicate-from-Space permission on the space

where the original COPY of the FILE resides together with the Write-to-Space permission on the SPACE where he/she would like to read or bring online the file from.

- ***Write-to-Space*** : specifies which users/groups can perform srmPrepareToPut and srmCopy operations or srmPrepareToGet and srmBringOnline with a token operations to this SPACE.
- ***Stage-to-Space*** : specifies which users/groups can perform srmPrepareToGet or srmBringOnline operations that imply a tape recall to this SPACE.
- ***Replicate-from-Space*** : specifies which users/groups can replicate a FILE from the SPACE where the ACE applies to another SPACE, where the user has Write-To-Space permissions. This operation is not associated to any of the SRM requests but enable internal disk-to-disk copies.
- ***Purge-from-Space*** : specifies which users/groups can perform srmPurgeFromSpace operations from this SPACE.
- ***Modify-Space-ACL*** : specifies which users/groups can modify the ACLs on the SPACE. The syntax and semantic of the modify operations must be specified.
- ***Query-Space*** : specifies which users/groups can perform srmGetSpaceMetadata operations on the SPACE.

Only the following operations are mandatory in WLCG:

Read-from-Space, Write-to-Space, Stage-to-Space, Replicate-from-Space, Purge-from-Space

The following operations are optional:

Release-Space, Update-Space, Modify-Space-ACL, Query-Space

Only the primary FQAN should be considered when matching ACLs. Experiments must comment on this assumption.

LHCb: This is OK. FQANs are only Role based for LHCb.

Wildcards in FQANs can be optionally supported by a storage system. Recommendations in [3] must be followed.

ACLs can be positive or negative. Positive ACLs are meant to grant access while negative ACLs specify who should be negated the authorization to the resources the ACLs apply to.

The proposed syntax and semantic to follow is based on NFSv4 [6].

Management tools must be available to the resource administrator to manipulate ACLs directly and change the internal resolution of proxies. The set of allowed management operations must be agreed on.

5.2 SRM ACLs on SPACEs

The syntax and semantic proposed for SRM ACLs for SPACEs are based on NFSv4 minor version 1 draft 21.

An Access Control Entry (ACE) is a record associated with SPACE reservation. Each ACE is defined as following:

```
subjectType,
subject,
accessMask,
aceType
```

where

aceType can be ALLOWED_ACE or DENIED_ACE;

subject can be DN or FQAN. The reserved name EVERYONE is interpreted as ANY;

subjectType specifies if the subject is a DN or FQAN;

accessMask is a list set of actions associated with this ACE. The valid actions

are:

```
RELEASE_SPACE        (D)
UPDATE_SPACE         (U)
READ_FROM_SPACE      (R)
WRITE_TO_SPACE       (W)
STAGE_TO_SPACE       (S)
REPLICATE_FROM_SPACE(C)
PURGE_FROM_SPACE     (P)
QUERY_SPACE          (Q)
MODIFY_SPACE_ACL     (M)
```

Access control list (ACL) is an ***ordered*** array of ACEs. Only ACEs which have a subject that matches the requester are considered. Each ACE is processed until all of the bits of the requester's access have been ALLOWED. Once a bit has been ALLOWED by an ALLOWED_ACE, it is no longer considered in the processing of later ACEs. If a DENIED_ACE is encountered where the requester's access still has ALLOWED bits in common with the accessMask of the ACE, the request is denied (in other words: to ALLOW all bits have to be checked, while for DENY one matching is sufficient). When the ACL is fully processed, if there are bits in the requester's mask that have not been ALLOWED or DENIED, access is denied.

The following syntax can be used to set/display ACLs:

subjectType:subject:accessMask:aceType

Example:

production write, admin all, regular user read, others – deny

```
fqan:dteam/Role=production:RSWQP:ALLOW
fqan:dteam/Role=lcgamin:DURWSPQM:ALLOW
fqan:dteam/Role=NULL:RSQ:ALLOW
fqan:EVERYONE:DURWSPQMC:DENY
```

read, no stage:

```
fqan:EVERYONE:R:ALLOW
fqan:EVERYONE:S:DENY
```

power user:

```
dn:/O=GermanGrid/OU=DESY/CN=Tigran Mkrtchyan:S:ALLOW
fqan:EVERYONE:RQ:ALLOW
fqan:EVERYONE:S:DENY
```

where accessMask is:
'**D**' for RELEASE_SPACE
'**U**' for UPDATE_SPACE
'**R**' for READ_FROM_SPACE
'**W**' for WRITE_TO_SPACE
'**S**' for STAGE_TO_SPACE
'**C**' for REPLICATE_TO_SPACE
'**P**' for PURGE_FROM_SPACE
'**Q**' for QUERY_SPACE
'**M**' for MODIFY_SPACE_ACL

While we do not have ONWER and DEFAULT ACL, all newly created reservation will have *de facto* default:

dn:<Creator DN>: DURWSCPQM:ALLOW
fqan:EVERYONE:DURWSCPQM:DENY
dn:EVERYONE:DURWSCPQM:DENY

Service administrators have an ***implicit ACE*** that allows them to perform all operations on a SPACE.

5.2.1 SRM SPACEs management methods

In what follows we list the new SRM SPACE management functions that would offer an interface to the requested new functionality.

5.2.1.1 srmReserveSpace

This function is used to reserve a SPACE in advance for the upcoming requests to get some guarantee on the FILE management. Asynchronous SPACE reservation may be necessary for some SRMs to serve many concurrent requests.

Please note: The proposed change implies modification of the current SRM v2.2 WSDL. We do not propose to change the WSDL at the moment. However we list the needed modification for supporting dynamic SPACE reservation in the future with the requested permissions on SPACEs.

Additional Data Types

```
enum        TSpaceRequestType {
                RELEASE-SPACE,
                UPDATE-SPACE,
```

```
                                READ-FROM-SPACE,
                                WRITE-TO-SPACE,
                                STAGE-TO-SPACE,
                                REPLICATE-FROM-SPACE,
                                PURGE-FROM-SPACE,
                                MODIFY-SPACE-ACL,
                                QUERY-SPACE}
enum        AceType {
                        ALLOWED_ACE
                        DENIED_ACE}

enum        SubjectType {
                        DN
                        FQAN}

typedef     struct {
                AceType             Type,
                String              Subject,
                SubjectType         SubjectType
                TSpaceRequestType[] AccessMask
            } TAccessControlEntry
```

Parameters

```
In:
string                      authorizationID,
string                      userSpaceTokenDescription,
TAccessControEntry[]        ACLs,
TRetentionPolicyInfo        retentionPolicyInfo,
unsigned long               desiredSizeOfTotalSpace,
unsigned long               desiredSizeOfGuaranteedSpace,
int                         desiredLifetimeOfReservedSpace,
unsigned long []            arrayOfExpectedFileSizes,
TExtraInfo[]                storageSystemInfo,
TTransferParameters         transferParameters

Out:
TReturnStatus               returnStatus,
string                      requestToken,
int                         estimatedProcessingTime,
TRetentionPolicyInfo        retentionPolicyInfo,
unsigned long               sizeOfTotalReservedSpace,       // best effort
unsigned long               sizeOfGuaranteedReservedSpace,
int                         lifetimeOfReservedSpace,
string                      spaceToken
```

5.2.1.2 srmGetSpaceMetadata

This function is used to get information of a SPACE. SPACE token must be provided, and SPACE tokens are returned upon a completion of a SPACE reservation through *srmReserveSpace* or *srmStatusOfReserveSpaceRequest*.

Please note: The proposed change implies modification of the current SRM v2.2 WSDL. We do not propose to change the WSDL at the moment. However we list the needed modification for supporting in the future the requested permissions on SPACEs.

Additional Data Types

```
typedef     struct {
                string                  spaceToken,
                TReturnStatus           status,
                TRetentionPolicyInfo    retentionPolicyInfo,
                TAccessControlEntry[]   ACLs,
                string                  owner,
                unsigned long           totalSize,          // best effort
                unsigned long           guaranteedSize,
                unsigned long           unusedSize,
                int                     lifetimeAssigned,
                int                     lifetimeLeft
} TMetaDataSpace
```

Parameters

```
In:
string          authorizationID,
string[]        arrayOfSpaceTokens

Out:
TReturnStatus       returnStatus,
TMetaDataSpace[]    arrayOfSpaceDetails
```

NOTE: Instead of changing the definition of srmReserveSpace and srmGetSpaceMetadata to introduce ACLs, it has been proposed to introduce new SRM methods for managing ACLs. This is a good solution since it allows for the new WSDL based services to be fully backward compatible.

5.2.1.3 srmPurgeFromSpace and srmChangeSpaceForFiles

It has been agreed that

- since SPACE tokens can be specified on SRM Get operations
- since the concept of the file primary copy is not used in WLCG
- since no tape transitions are allowed in WLCG

the method srmChangeSpaceForFiles does not need to be provided. The same functionality can be reached invoking srmBringOnline or srmPrepareToGet with a token (specifying the new space) and srmPurgeFromSpace.

5.2.2 SRM get methods

The definition of the srmPrepareToGet/srmStatusOfGetRequest, srmBringOnline/srmStatusOfBringOnline, and srmCopy/srmStatusOfCopy request remains unchanged with respect to what has been specified in the SRM v2.2 specification [1]. The only difference is that now clients can pass a SPACE token that has to be honored in the calls. A COPY of the FILE associated to the list of SURLs specified must be retrieved in the SPACE specified if the user making the request has sufficient privileges on the specified SPACE token and namespace entry.

NOTE: When serving a srmPrepareToGet request without a token on a FILE that has multiple COPYs in several SPACEs, the COPY served to the user must be one for which the user has read permission on the correspondent SPACE. In case many of these exist, the system can choose one of them.

5.2.3 SRM directory methods

No modifications are required for srmLs with the exception that if there are multiple COPYs of a FILE in several SPACEs, the fileLocality parameter must reflect the status of all COPYs. Therefore if there is a physical copy on tape only in a CUSTODIAL-NEARLINE SPACE and at the same time there is a COPY in a REPLICA-ONLINE SPACE, the resulting fileLocality must be NEARLINE_AND_ONLINE.

5.2.4 Tape usage optimization

It has been noted that all implementations provide at the moment SPACE token [or SPACE token description] and directory information to the tape callback mechanisms so that appropriate tape selection and migration policies can be defined in the system.

All relative SRM calls have the extra parameter TExtraInfo[] defined as follows:

```
typedef  struct {
                  string      key,
                  string      value
                } TExtraInfo
```

TExtraInfo is used wherever additional information is needed. Some system might honor extra information provided through this structure by the clients. Therefore, no extra functionality needs to be implemented.

6. Time estimation to implement the proposed features

6.1. CASTOR

Editor: Shaun De Witt, Giuseppe Lo Presti

6.1.1 Space Protection

CASTOR already implements a sophisticated protection mechanism which meets the requirements outlined in this document. The 'black and white' lists provide both negative and positive access rights to spaces. However, they are based on uids/gids rather than DNs or FQANs.

Administrative tools to help managing black and white lists have been recently released in CASTOR and are going to be deployed in the coming weeks.

6.1.2 VOMS Awareness

Since access control is delegated to the CASTOR instance it is not proposed that the CASTOR SRM will be made VOMS aware as defined within the scope of this document before the CASTOR backend itself is made VOMS aware. To attempt to do so would lead to operations teams having to maintain both CASTOR and SRM protections and ensuring they are consistent at all times. It may be possible to implement restrictions on specific SRM APIs to VOMS roles, but this is outside the scope of this document.

The current plan is to implement VOMS primary role based authentication in CASTOR for Q1 2009, and a fully VOMS awareness of CASTOR and the SRM interface is envisaged for Q3 2009.

6.1.3 Tokens on Get operations

This is already implmented in the CASTOR SRM. If no space token is passed the SRM has a configurable mechanism to define whether a copy in an existing space is to be used, or whether to trigger a copy into a new space.

6.1.4 srmGetSpaceMetaData

This is already implemented for CASTOR.

6.1.5 srmPurgeFromSpace

It will be possible to implement srmPurgeFromSpace within Q3 2008.

6.1.6 Other points

It should be possible to implement returning space tokens in srmLs within Q4 2008.

With respect to the additional pin lifetimes as defined in section 3.2.3, the current implementation is that the pinLifeTime parameter on Put operations is negotiated to be always equal to a system defined default. No change is envisaged in the future for this behavior.

6.2. dCache

Editor: Gene Oleynik

Our current estimate, based on V1.1, was 75 working days, 13 of which are for the short-term solution. However, we have higher priority work on our Fermilab installations, in particular work to significant stabilize dCache/SRM for US-CMS-T1.

Considering vacations, holidays and our other work load, with current staffing of the SRM work of 1.5 FTE, we estimated it would take one year to complete our program of work. With the addition of another staff, we estimate we could finish the program by the end of the calendar year. These estimates may not hold for any changes since V1.1.

Our number one priority now is upgrading our public dCache system and stabilizing US-CMS-T1 dCache ad resolving their transfer problems. In addition, we do not yet have buy-in from upper management to do this extension work.

6.3. DPM

Editor: David Smith

Time estimates are for development, deployment is not explicitly considered

Medium Term:

Support a list of VOMS FQANs for the space write permission check, rather than just the current single FQAN.
Time estimate: September 2008

Long Term:

Return array of space tokens when using srmLs on a single file.
Time estimate: October 2008

TExtraInfo parameters on SRMCopy: SRMCopy itself is currently not available from the DPM. There is no time estimate on the delivery of SRMCopy at the moment.

Allow srmReleaseFiles on surls without giving the associated request ID.
Time estimate: January 2009

Honor space tokens on srmPrepareToGet and srmBringOnlinerequests for disk only files. (Space tokens are already supported on DICOM recall, but this is outside the scope of WLCG).
Time estimate: June 2009

ACLs on spaces: For WLCG, of the attributes listed in the MoU those relevant to the DPM are (numbering as in addendum document):

(1) Read-from-space
(2) Write-to-space

(4) Replicate-from-space
(5) Purge-from-space

It is believed that a number of issues will arise and need to be solved during the development and testing of the space ACLs; for instance the handling of undesirable conditions arising for various combinations of ACLs, such as the ability of a given user to replicate to a space but being disallowed to remove the newly created copy. For this reason it is believed important to allow for time to identify and resolve those issues.

Estimate: June 2010

6.4. StoRM

Editor: Luca Magnoni

This is the implementation plan for StoRM. Please note that all the time estimation could change depending on man power availability.

Short term plan as published in this document

Available by the end of October 2008.

Full implementation

8 men/months of work.

The implementation phase will start after the short term release.

For more details, please visit the StoRM web site:
http://storm.forge.cnaf.infn.it/documentation/storm_release_plan

Short-Term Implementation Specific Solutions

7. CASTOR

Editor: Shaun De Witt

7.1 Space protection

CASTOR already implements a sophisticated protection mechanism which meets the requirements outlined in this document. The 'black and white' lists provide both negative and positive access rights to spaces. However, it is based on uid/gid rather than DN or FQAN.

In the next three months an administrative interface will be made available for CASTOR administrators to easily set white and black lists.

7.2 VOMS Awareness

Since access control is delegated to the CASTOR instance it is not proposed that the CASTOR SRM will me made fully VOMS aware as defined within the scope of this document. To attempt to do so would lead to operations teams having to maintain both castor and srm protections and ensuring they are consistent at all times. It may be possible to implement restrictions on specific SRM APIs to VOMS roles, but this is outside the scope of this document.

In the next three months an administrative interface will be made available for CASTOR administrators to easily set white and black lists.

7.3 Tokens on Get operations

This is already implemented in the CASTOR SRM. If no space token is passed the SRM has a configurable mechanism to define whether a copy in an existing space is to be used, or whether to trigger a copy into a new space.

7.4 srmGetSpaceMetaData

This is already implmented for CASTOR.

7.5 Other points

It will be possible to implement purgeFromSpace within the time frame envisaged. In addition it may be possible to implement returning space tokens in srmLs.

It is unlikely that we will be able to implement the additional pin lifetimes on Put and Copy specified in sections 3.1.5 and 3.1.6.

8. dCache

Editor: Patrick Fuhrmann

These are the issues which according to Flavia's and my understanding would solve the most pressing issues for the experiments in order to survive the first 12 months of LHC data taking e.t.c.

8.1 Space Tokens for operations other than 'write'

This modification would need large parts of dCache to be refurbished and therefore we would prefer to find alternative solutions.

Use Case

The purpose of tokens in 'read' and 'bring online' primarily is to steer data streams or better to steer the location of files. The space reservation aspect of tokens is of minor interest. An example is that the same dataset may be needed by the reprocessing system as well as for FTS export or user analysis. It would be envisioned that this file is served to the various competing processes by different locations in the system mainly not to interfere or slowdown expensive reprocessing.

Proposed alternative Solution

A solution already available in dCache would be to select an appropriate pool by the IP number of the client, the requested transfer protocol or the path of the file. This would require that either IP number or protocol differ between the different processes requesting the same file or that files are requested, which are located in different directory subtrees. To our current understanding this is not possible in all cases. Therefore a particular key-value pair within the TExtraInfo of the bring-online and prepare-to-get would allow the dCache decision engine to find the appropriate area to stage the file or to serve the file from. Other implementations would just ignore this extra info so that it can be provided to all implementations (resp sites). Other systems may use different key value pairs to achieve the same purposes. It seems that the LCG tools already provide the possibility to forward this information to the SRM calls, though FTS doesn't do this yet. The difference between using the TExtraInfo instead of space tokens is that space tokens require proper space calculation and reservation which is actually not really required assuming that there is enough space to store the reprocessing data.

Remarks

Not having T1D1 and not honoring Space Tokens for prepare-to-get and bring-online, implies that Space Tokens are only used to guaranty available space for incoming data. Further management of spaces (e.g. srmPurgeFromSpace) is obsolete, because in the T1D0 case, the file is removed from that space token as soon as it has been successfully migrated to tape. In the T0D1 case, files can not be purged from space because the space contains the only “Space Token” managed copy. Though the file can have different copies in dCache, only one of them can be in a Token.

The ability to pin files on disk is not touched by the new approach. Even files brought to disk areas, not managed by Space Tokens, will honor the specified pin time of the corresponding bring-online or prepare-to-get and this pin(time) can be modified by the appropriate SRM command(s).

Sometimes the term “unmanaged space” is used in conjunction with certain storage partitions in dCache. This term is slightly misleading. It actually only indicates that “Space Tokens” can't be created within those areas. But files can be pinned and data can be implicitly (Ip number, protocol, directory tree) or explicitly (TExtraInfo) directed to them. Moreover, the dCache provides the ability of fine grained data location steering by means of IP number, data direction, transfer protocol, directory subtree and as of this proposal by TextraInfo.

8.2 Protecting "Write Space Tokens"

Space Tokens in dCache can be created dynamically by the appropriate SRM function calls. Though, while the creation of a token is protected by some very basic mechanism, the usage and the release of the token are not. It is clear to us that this is a security issue which we would like to address.

Proposed Solution

We would propose to protect two operations on write tokens only: the 'write into a token' and the 'release token'. It is not yet clear how detailed the ACL system needs to be in order to satisfy the experiment requirements. Our current assumption is that a simple one will do.

8.3 Protecting expensive operations (Restore from tape robotics or pool to pool copies)

There is the concern that uncoordinated access to expensive resources, e.g. the Tape System may be misused by inexperienced or malicious users. A plan often discussed is to disallow triggering tape operations by regular VO users. Therefore, protecting those resources would be envisioned. In dCache, it would be possible to allow or disallow the access to the tape system for particular DN's or FQAN's. Here as well the details have not been discussed yet. We would try to keep the ACL mechanism simple and only the 'tape restore' and 'pool to pool transfers' would be protected.

8.4 Replacing T1D1 by T1D0 using pins.

For dCache, T1D1 can be perfectly replaced by T1D0 plus a pin on the file. “Hard pins” can already be set to files in the T1D0 storage class through the prepare-to-get or the bring-online SRM operations. One essential requirement for this approach would be the possibility to 'release' the pin without knowing the SRM request ID of the initial prepare-to-get or bring-online operation. dCache would try to do the 'pin release' based on the DN or FQAN of the requester. Here as well, details still need to be discusses.

8.5 Estimate on the time is takes to implement the changes in dCache.

To be done.

9. DPM

Editor: David Smith

For the DPM the alternate proposal is that we are covering the issues highlighted by the experiment requirements already.

9.1 Space protection

Spaces in DPM belong to a group (or possibly a user, but usually a group) and to be able to write into a space a user has to belong to the space's group. In that respect the space is protected. The requirement's example of generic users triggering staging is not a concern for DPM for the WLCG.
Sites would like to be able to have a list of FQANs which are allowed to write to a space and not just one FQAN which is currently allowed. It is expected a change to allow for the list can be made within

three months. However the timeframe is still subject to agreement and this will be moved to section 6 in a later revision of this document. Section 6 will also list time estimates on the other proposed features for the long term changes.

9.2 VOMS Awareness

DPM has full voms-awareness. There are access controls on pools, spaces and the namespace. Of course only the namespace is POSIX, the others are somewhat simpler lists, but they do take VOMS FQANs into account.

9.3 Tokens on Get operations

It is true that for disk based copies (i.e. what concerns the WLCG) DPM will not use the space token provided for get/bringonline. However the requirement seems to be concerned with staging a file back from tape into the right area. Do we need to add the implied disk replication to satisfy the experiments' needs?

9.4 Space statistics

We have the srmGetSpaceMetaData call available.

9.5 Tape optimization

For WLCG, DPM is disk based only, so there is no issue with us providing information to the tape system.

10. StoRM

Editor: Luca Magnoni, Riccardo Zappi

This is the short term plan for StoRM in the next three months. The time estimation could change depending on man power availability. The release plan for StoRM can be found in [9].

10.1 Space protection

Spaces in StoRM will be protected via ACLs. ACLs are based at the moment on user DN and FQANs. Wildcards on DN or FQAN are also supported. ACLs apply at the moment to space token descriptions only (and not to space tokens). However, in StoRM there is a one-to-one mapping between space tokens and space token descriptions. Therefore, setting ACLs on space token descriptions is equivalent to setting ACLs on space tokens. This is an advantage for site administrators that do not have to deal with the internal representation of the space tokens.

It is in the short term plans to make the ACLs compliant with the ones defined in this document.
Service administrators can manage space protection via StoRM configuration files.

10.2 VOMS Awareness

StoRM has full voms-awareness. An improvement on the permissions management will be provided.

There will be the possibility to define ACLs on a per-directory base. These ACLs will also be based on DNs and FQANs and will allow for wildcards.

10.3 Tokens on Get operations

In the current short term plan (three months) we will not support tokens on Get operations. We need to know if experiments need to be able to use this functionality to change spaces for online copies.

10.4 Space statistics

srmGetSpaceMetaData already returns up to date information on the space usage for the different storage areas and dynamic space reservations. Space accounting is based on the assumption that different logical copies of a file in the same space are accounted only once, and different logical copies of the same file in different spaces are accounted multiple times.

10.5 Tape usage optimization

For WLCG, StoRM provides only 2 types of spaces: REPLICA-ONLINE and CUSTODIAL-ONLINE. Path and space tokens can be passed to the tape back-end for tape usage optimization, if needed.

11. Client tools (lcg-utils/GFAL, FTS)

The client tools will avoid using switches internally, since it has been agreed that the interface to the Grid storage services is SRM.

11.1 Tokens on Get operations

In order to make the differences between implementations transparent to users, the client tools would need to pass on Get operations both the SPACE token and the TExtraInfo structure. Passing the TExtraInfo structure requires a change of the database schema in FTS and the creation of tables that complicate a lot the logic of the code. Furthermore, such table may need to be made obsolete once the proper solution is implemented. Therefore, the developers of the client tools agree to such modification only if the number of the new key is restricted to a minimum and well define set of values.

SPACE types

The same type of SPACE might be implemented as T1D1 for CASTOR, while T1D0 + pinning is used in dCache. This implies different actions for the client tools to release a COPY. It has been decided that to release a COPY the client tools will perform a srmReleaseFiles together with a srmPurgeFromSpace. While dCache will honor the srmReleaseFiles request releasing the corresponding COPYs and ignore the srmPurgeFromSpace returning SRM_NOT_SUPPORTED, CASTOR will honor the srmPurgeFromSpace call ignoring the srmReleaseFiles. The same is true when pinning a COPY.

The needed changes can only be coded starting in September 2008.

REFERENCES

[1] The Storage Resource Manager Interface Specification, Version 2.2, OGF – Grid Storage Resource Management Working Group, 15 February 2008, http://www.ogf.org/documents/GFD.129.pdf
[2] A VOMS Attribute Certificate Profile for Authorization, V. Ciaschini, 15 April 2004, http://grid-auth.infn.it/docs/AC-RFC.pdf
[3] Recommendations for changes in gLite Authorization, C. Witzig, 6 February 2008, https://edms.cern.ch/document/887174/1
[4] Storage Element Model for SRM 2.2 and GLUE schema description, F. Donno et al., v. 3.5, 27 October 2006, https://forge.gridforum.org/sf/docman/do/downloadDocument/projects.glue-wg/docman.root.background.specifications/doc14619;jsessionid=58E33DC10A69FABED90ACD4C8EFE6E1F
[5] SRM v2.2 WLCG usage agreement, May 2005, http://cd-docdb.fnal.gov/0015/001583/001/SRMLCG-MoU-day2%5B1%5D.pdf
[6] NFSv4 minor version 1 draft 21, http://www.nfsv4-editor.org/
[7] POSIX Access Control Lists on Linux, http://www.suse.de/~agruen/acl/linux-acls/online/
[8] CASTOR Roadmap, https://twiki.cern.ch/twiki/bin/view/FIOgroup/RoadMap
[9] StoRM Release plan, http://storm.forge.cnaf.infn.it/documentation/storm_release_plan