Addendum to the SRM v2.2 WLCG Usage Agreement

To: CCRC'08 Storage Solutions Working Group

From: Flavia Donno

Date: 4/21/2008

Version: 0.5

Comments: O. Barring, G. Behrmann, A. Frohner, 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

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
- GFAL
- lcg-utils
- 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 (May 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 feature 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.

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

1. WLCG requirements

In this section we describe the experiment requirements that took 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 clean solution still is not available.

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, interfering with production operations; CASTOR honors the token if passed to select the pool where 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 keeping it into account must be given.

D. ***<u>Correct implementation of srmGetSpaceMetaData</u>***
srmGetSpaceMetaData should return information about the space used and available (as defined in [4]) for a given token description and possibly for the so-called "default" area. A default area is not defined in the SRM spec and it is used by some implementations whenever the clients do not pass space tokens in their write operations.

E. ***<u>Providing the necessary information so that data could be efficiently stored on tapes.</u>***
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 ***<u>online</u>*** physical space 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 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.

When a file is removed or purged from a space the space occupied by the file is not accounted against the space reservation and the space occupied by the file can be re-used. There must not be an expectation that the space occupied by that file is immediately released.

NOTE: For this reason, it has been noted that the srmRm operation should be made asynchronous.

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 a "DEFAULT" space nor implies that concept. An implementation of a DEFAULT space/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.

2.2 SRM files and copies

Definition: A file is a set of data with the properties defined on pag. 19 of [1], paragraph 2.25:

- SURL (*)
- Path
- Size
- Creation Time
- Modification Time
- Storage Type (*)
- Retention Policy Info (Retention Policy, Access Latency)
- File Locality (*)
- Array of Space Tokens (*)
- File Type (*)
- Assigned Lifetime
- Left Lifetime
- Permissions (*)
- Checksum Type (*)
- Checksum Value (*)
- Array of Sub Paths (if the file is a directory) (*)

The Retention Policy Info attribute 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 copies in several spaces.

Definition: A ***copy*** of a file is a physical instance of the file in a given space. It is characterized by the properties listed above for a file with the exception of those marked with a (*).

A copy of a file might have an associated transport URL (TURL). Such TURL has an associated pin-lifetime. Multiple TURLs may refer to the same copy of the same file or to different physical copies of the same file. A TURL is a user –guaranteed handle to the file. It is valid till its lifetime or associated pin is not expired. If one of the TURLs associated to a copy of a file is released, the other TURLs whose lifetime have not yet expired will continue to stay valid. A TURL is generated by the system after a srmPrepareToGet or srmPrepareToPut request and might be associated to the request that has generated it.

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 space occupied by the file is not accounted against the space reservation and the space occupied by the file can be re-used. There must not be an expectation that the space occupied by that file is immediately released.

For this reason, it has been noted that the srmRm operation should be made asynchronous.

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

3.2 srmLs

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

3.3 srmPurgeFromSpace

In WLCG no tape transitions are allowed.

The srmPurgeFromSpace method only removes disk copies of a file.

If a file is in a Custodial-Nearline space and at a given moment the file has a copy on disk and one on tape, a srmPurgeFromSpace operation removes the copy on disk and leaves the copy on tape.

If a file has 2 copies 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.

3.4 srmBringOnline

The current SRM v2.2 specifications are wrong in paragraph 5.3.2, point g). In fact the Access Latency can be either ONLINE or NEARLINE in WLCG.

3.5 srmRelease

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 copies of PERMANENT files in there.

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

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.

3.1 VOMS Groups, Roles, and Access Control Lists

Every user is assigned 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:  root
    # 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

An SRM space is further characterized by the following properties:

- Owner/Group Permission (Release-Space, Update-Space, Read-from-Space, Write-to-Space, Stage-to-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.
- ***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.
- ***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 primary FQAN should be considered when matching ACLs. (Experiments must comment on this assumption).

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:

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

where

aceType can be ALLOWED_ACE or DENIED_ACE;

subject can be DN or FQAN. Empty DN/FQAN is interpreted as ANY (a reserved name can be used, e.g. EVERYONE);

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
UPDATE_SPACE
READ_FROM_SPACE
WRITE_TO_SPACE
STAGE_TO_SPACE
PURGE_FROM_SPACE
QUERY_SPACE
MODIFY_SPACE_ACL
```

TODO:

Each SpaceManager operation have to be associated with corresponding access mask. Some 'read' operation can be in reality 'write' (like srmPrepareToGet).

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:DURWSPQM: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
'**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 :

fqan:EVERYONE:DURWSPQM:DENY
dn:EVERYONE:DURWSPQM:DENY

To overcome this situation the following strategy can be used: if there is an ACL associated with Space Reservation, than the ACL is respected, otherwise access is allowed.

5.3 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.

5.3.1 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 permission on spaces.

Additional Data Types

```
enum        TSpaceRequestType {
                                RELEASE-SPACE,
                                 UPDATE-SPACE,
                                 READ-FROM-SPACE,
                                 WRITE-TO-SPACE,
                                 STAGE-TO-SPACE,
                                 REMOVE-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,	
string	Owner (VOMS DN),	
string	Group (VOMS FQAN),	
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	

5.3.2 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 dynamic space reservation in the future with the requested permission 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
```

5.3.3 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.

5.4 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 storage 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 online copies 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.

5.5 SRM directory methods

No modifications are required for srmLs with the exception that if there are multiple copies of a file in several spaces, the fileLocality parameter must reflect the status of all copies. Therefore if there is a copy on tape only in a CUSTODIAL-NEARLINE space and at the same time there is a copy on disk in a REPLICA-ONLINE space, the resulting fileLocality must be NEARLINE_AND_ONLINE.

5.6 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 provide through this structure by the clients. Therefore, no extra functionality needs to be implemented.

REFERENCES

[1] The Storage Resource Manager Interface Specification, Version 2.2, OGF – Grid Storage Resource Management Working Group, 15 February 2008, http://datagrid.lbl.gov/tmp2/OGF-GSM-SRM-v2.2-080121.doc
[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/