Storage Element Model for SRM 2.2 and GLUE schema description

Flavia Donno[1], Paolo Badino[1], Jean-Philippe Baud[1],
Stephen Burke[2], Ezio Corso[4], Shaun De Witt[2],
Patrick Fuhrmann[3], Maarten Litmaath[1], Riccardo Zappi[5]

(1) CERN, European Organization for Nuclear Research, Switzerland
(2) CCLRC, Rutherford Appleton Laboratory, UK
(3) DESY, Deutches Elektron-Synchroton, Germany
(4) ICTP, The Abdus Salam, international Centre for Theoretical Physics, Italy
(5) CNAF, Istituto Nazionale di Fisica Nucleare, Italy

27 October 2006
v3.5

1. Introduction

In this document we describe the model for a Grid Storage Element to clarify the behavior of SRM v2.2 interface implementations and to allow for a better understanding and justification of the proposed GLUE schema description for a Storage Element.

This document is based on the discussions that took place during the pre-GDB workshop on Storage Classes at CERN held on the 3rd October 2006, the e-mail exchanges on the srm-devel mailing list, the document on the proposed Storage Element GLUE schema v0.5 as a contribution to GLUE v1.3 and the WLCG SRM v2.2 MoU (https://srm.fnal.gov/twiki/bin/view/WorkshopsAndConferences/GridStorageInterfacesWSAgenda).

The GLUE schema description that follows is not WLCG specific. However, the use-cases provided apply mostly to LHC experiments.

2. The Storage Classes

In WLCG the Storage Class Working Group has been established to understand the requirements of the LHC experiments in terms of storage and the implementations of such requirements for the various storage solutions available.

A *Storage Class* *determines the properties that a storage system needs to provide in order to store data.*

The LHC experiments have asked for the availability of combinations of the following storage devices: Tapes (or reliable storage system always referred to as tape in what follows) and Disks. If a file resides on Tape then we say that the file is in Storage Class Tape1. If a file resides on an experiment-

managed disk, we say that the file is in Storage Class Disk1. Tape0 means that the file does not have a copy stored on a reliable storage system. Disk0 means that the disk where the copy of the file resides is managed by the system: if such a copy is not pinned or it is not being used, the system can delete it.

Following what as been decided in various WLCG Storage Class Working Group meetings and discussions only the following combinations (or Storage Classes) are supported for the next implementations of SRM v2.2:

- Custodial-Nearline: this is the so-called Tape1Disk0 class.
- Custodial-Online: this is the so-called Tape1Disk1 class
- Replica-Online: this is the so-called Tape0Disk1 class

- Tape0Disk0 is not implemented. It is pure scratch space that could be emulated using one of the available classes and removing the data explicitly once done. However, it could be handy for LHC VOs to have such a type of space actually implemented.

In the *Custodial-Nearline* storage class data is stored on some reliable secondary storage system (such as a robotic tape or dvd library). Access to data may imply certain latency. In WLCG this means that a copy of the file is on tape (Tape1). When a user accesses a file, the file is recalled in a cache that is managed by the system (Disk0). The file can be "*pinned*" for the time the application needs the file. However, the treatment of a pinned file on a system-managed disk is implementation dependent, some implementations choosing to honor pins and preventing additional requests, others removing unused on-line copies of files to make space for new requests.

In the *Custodial-Online* storage class data is always available on disk. A copy of the data resides permanently on tape, dvd or on a high-quality RAID system as well. The space owner (the virtual organization) manages the space available on disk. If no space is available in the disk area for a new file, the file creation operation fails.
This storage class guarantees that a file is never removed by the system.

The *Replica-Online* storage class is implemented through the use of disk-based solutions not necessarily of high quality. The data resides on disk space managed by the virtual organization.

3. Storage Class Transitions

Through the SRM call ChangeSpaceForFiles it is possible to schedule Storage Class Transitions for a list of files.
Only the following transitions are allowed in WLCG:

- Tape1Disk1 -> Tape1Disk0. On some systems this can be implemented as a metadata operation only, while other systems may require more operations to guarantee such a transition.
- Tape1Disk0 -> Tape1Disk1. During the Storage Class pre-GDB of October 3rd, 2006 it was decided that this transition would be implemented with some restrictions: the srmChangeSpaceForFiles call will complete successfully but the files will remain on tape. The files will be actually recalled from tape to disk only after a BringOnline operation is executed. This is done in order to avoid that a big set of files is unnecessarily scheduled for staging and therefore to smoothen operations in particular for those Mass Storage Systems that do not have a scheduler (namely TSM).
- Tape0<->Tape1 transitions are not supported at the start of LHC (if ever). For physics validation operations, since the amount of data to transfer to tape after the validation is not big (only 1-2% of total data) a change class operation from Tape0Disk1 to Tape1DiskN can be approximated by copying the files to another part of the name space, specifying Tape1DiskN as the new storage class, and then removing the original entries.

4. The Storage Element

A *Storage Element* *is a Grid service that allows Grid users to store and manage files together with the space assigned to them.*

Examples of a storage element are:

- The CASTOR system at CERN with its SRM interface, the physical storage backend being the set of robotic tape libraries and the pool of disk servers in front of them offering online/cache storage;
- The DPM system with its SRM interface and its set of disk servers, each of them managing given filesystems;
- The dCache system with its SRM interface able to manage disk space and to interface to tape systems such as TSM, HPSS, Enstore and OSM;
- The StoRM system, offering an SRM interface to parallel filesystems such as GPFS.

A Storage Element has properties such as:

- A *Globally Unique Identifier* that univocally identifies the SE.
- The *Implementation*, the software system used to manage the storage devices and servers. Examples of this can be: CASTOR, dCache, DPM, StoRM, etc.
- The *ImplementationVersion*. Through the version, specific features of the SE can be exposed.
- The *OnlineSizeTotal* and the *NearlineSizeTotal*. This is the sum up of the Total Size of Online and Nearline nominal space. For instance, the NearlineSizeTotal reports the total nominal space on tape not considering compression. These sizes are reported in GB (10^9bytes).

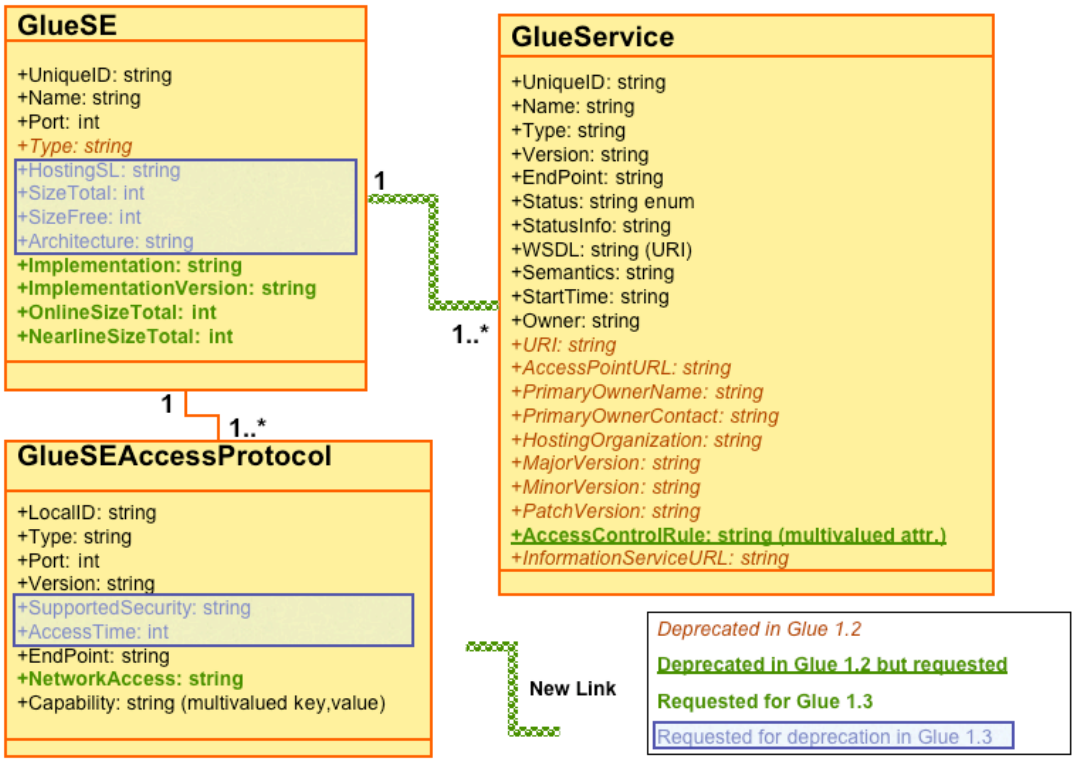

Figure 1: Storage Element description in SE GLUE Schema v1.3

Figure 1 shows the Glue proposed schema for GLUE v1.3 for a Storage Element. The classes shown are existent. The (blue) boxes highlight the attributes that we propose to deprecate in GLUE v1.3, the (red) italic attributes are already deprecated in GLUE v1.2. The bold (green) attributes are the ones we propose to introduce, while the (green) bold underlined are the ones that are deprecated in GLUE v1.2 and we propose to keep in GLUE v1.3. In the appendix we give use-cases that express the need for the proposed changes.

Note:

1. It was felt that a GlueSE should not publish a size since it is not easy to come up with a meaningful and coherent definition of a size in case of very complex systems such as those managing robotic and online devices. It is better to move the size property to lower level descriptions of the storage systems as detailed later in this document. However, it seems feasible to expose Online and Nearline Total Sizes that can be used when making reports and obtained with simple queries that do not require looping through the GLUE schema.
2. The Storage Element Control protocol is described in the GlueService class (as defined by the GLUE schema v1.2) that defines the service. For instance, the SRM is one control protocol for the Storage Element. Storage Elements proprietary or internal protocols can also

be exposed to the information system. In this case a GlueService will describe the service associated to them.
However, we do not propose to deprecate the GLUE 1.2 Control Protocol class for the moment, since we need it for backward compatibility.

3. The Service Access Control Rule needs to be kept in order to describe which VOs have access to the Storage Element service. This attribute was deprecated in GLUE v1.2.

Access Protocol

The *Access Protocol* *describes how files can be accessed on this Storage Element*.
An Access Protocol is identified by a Type and a Version (gsiftp 1, gsiftp 2, dcap 1.7.6, etc.)
The list of Access Protocol Types is defined already in GLUE schema v1.2. Please check:
http://glueschema.forge.cnaf.infn.it/V12/SEAccessProtocolType.

The *Capability* entry is used to define further Access protocol properties for which there is no explicit entry in the GLUE schema.

The *NetworkAccess* attribute in the Access Protocol can be used to specify if the protocol is valid for a LAN or a WAN

(For a more detailed explanation please refer to the document “Proposal for GLUE v1.3 for Storage” by Jens Jensen et al..)

Figure 1 shows the proposed description for the GlueSEAccessProtocol as we would like to see it in GLUE v1.3. This Class has links to the GlueSE class as well as the GlueStorageArea class introduced later.

Note:

4. The protocol “file” does not appear in the list of SEAccessProtocolType. We feel that this protocol should be added in order to avoid describing all those proprietary protocols that allow for native POSIX I/O.
5. The protocol “xrot” is also missing in the list of SEAccessProtocolType.
6. Should the protocols appearing in SEAccessProtocolType be approved by IANA? This would allow for the standardization of names used.
7. A list of access protocols should appear as well in the CESEBind GLUE class in order to specify which SE protocols can be used from the particular CE bound to that SE and which properties they expose.
8. CESEBind might describe CE and SE that are not in the same domain. An example of this is the NIKHEF/SARA connection.

9. We assume that WNs are always in the same domain as a CE and that the CE describes also the characteristics of WNs in its own domain (there are no examples to the contrary).
10. Nothing can be said for protocols not explicitly mentioned in a CE-SE binding.
11. The Capability field can be used to specify other restrictions such as WAN read-only/LAN read-write.
12. If an SE can be accessed directly using one of the supported access protocols, then the SE should be described as a GLUE service using that access protocol as control protocol.

Use Case: A possible use case is a user who wants to find all CEs that can access the input data available on an SE and/or can write output data on an SE that supports the same protocols that the application uses (this is a typical scenario for a user using JDL requirements with the gLite WMS or the LCG Resource Broker).

Use Case: It should be possible to distinguish between protocols that are allowed for read-only operations on WAN like for dcap in certain cases, or protocols that can only be used by a subset of the supported VOs. This can be expressed through the Capability attribute in the GlueSEAccessProtocol object.

Use Case: The gsiftp protocol used by a Classic SE should be considered a control protocol as well. Therefore there is a GLUE service associated to Classic SE reporting the gsiftp as control protocol. Operations such as mkdir, rmdir are normally done by the control protocol. On the Classic SE such operations are performed through gsiftp. Therefore this protocol should be published as both control and access protocol for a Classic SE.

5. The Storage Component

A Storage Component identifies a specific storage with certain properties. They are the following:

1. Retention Policies: CUSTODIAL or REPLICA or OUTPUT
2. Access Latency: NEARLINE or ONLINE (or OFFLINE)
3. Access Protocols (examples: rfio, dcap, file, etc.)

The concept of a Storage Component was introduced to describe the type of storage that is used to offer a certain quality of storage. For instance a Storage Component is a tape set or a pool of filesystems.

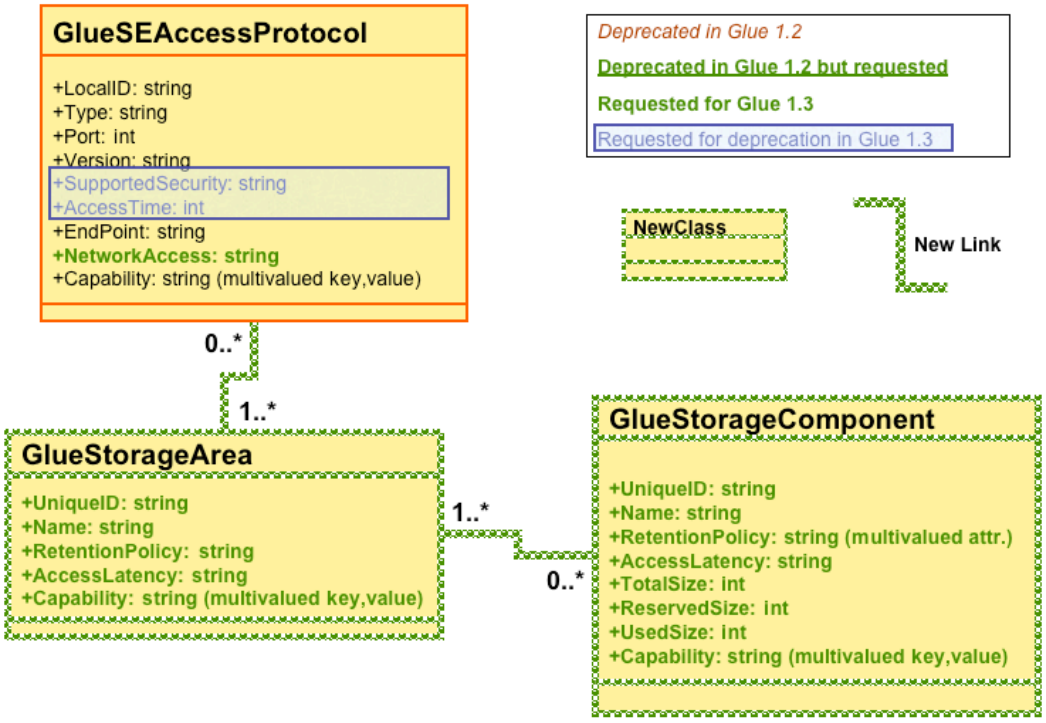

Figure 2: The Storage Component in the GLUE schema

The Storage Component implements a multivalued *Retention Policy* (the ones supported) and an *Access Latency*. The Storage Component publishes Total, Reserved and Used Size. They are defined as follows:

Total Size: is the total space size that this Storage Component can provide expressed in Gigabytes. It is the nominal capacity of the Storage Component subsystem (tape, dvd, disk, etc.)

Reserved Size: is the size of space reserved but not yet used, expressed in Gigabytes.

Used Size: is the size occupied by files that are not candidates for garbage collection.

For a discussion on storage sizes, please refer to the paragraph “Free and available space” in this document.
A Storage Component can be optionally identified by a *Name*. This can refer for instance to the name of the DPM pool of filesystems composing the Storage Component.

Note:

13. The WLCG MoU for the implementation of SRM v2.2 states that the only retention policies supported in SRM v2.2 are CUSTODIAL and REPLICA.
14. Two Storage Components cannot overlap in order not to double count space.

15. In case of tapes the Total Size reports the nominal capacity of the tape (i.e. without considering the nominal compression factor). LHC experiments data are normally compressed and the compression algorithm used by most tape drives does not provide any benefit. Normally LHC experiments compress and uncompress data on the fly. Other VOs might take advantage of the compression algorithm used by tape drives. It is up to them to take this factor into account when counting up the space published by Storage Components.
16. A tape-only storage component might still have some disks in front for performance reasons. Such disks should be totally hidden and not published in the information system.
17. We felt that quotas should be specified at the level of a Storage Component. However, at this time we would like to defer the matter of thinking through how to express and/or implement quotas in the GLUE schema, as a tentative exploration showed that there are non-trivial issues to be dealt with.

Use Case: The Storage Component represents the storage space the LHC experiments are willing to pay for. Any extra storage needed for optimizing system performance (for instance a specialized cache space layer between a tape system and an online disk system) should be hidden and not exposed via the information system.

Use Case: The Storage Component allows for a description of each single component type in a Storage Element. Therefore, it is possible to find out the total amount of space on tape and/or disk summing up the published total size for all Storage Components part of a Storage Element.

6. The Storage Area

A Storage Element can have multiple Storage Areas.

A *Storage Area defines a portion of the total available space that can span different kinds of storage devices within a Storage Element and in case of WLCG it implements a Storage Class instance*.

In principle any combination of Storage Components in a Storage Area is possible. For instance, a Storage Area can include a certain set of tapes and the space served by several disk servers. However, two Storage Areas can share the same Storage Components, but implement different policies.

Several Virtual Organizations can share the same Storage Area and can reserve space in it.
This is for instance the case for the default Storage Area in WLCG.

For WLCG only the space in a Storage Area reserved by Virtual Organization Managers and identified per VO by a User Space Token Description is optionally published in the Information System.

In the same SE, there could be many Storage Areas that implement the same Storage Class characteristics.

A VO might have multiple Storage Areas reserved to it. Through the ReserveSpace SRM call, an application passes the User Space Token Description and gets back a *Space Token*. For put operations, the middleware (GFAL, lcg-utils, FTS, etc.) takes the token description as input, converts it to a space token, which is passed to the underlying SRM method. A User Space Token Description might have multiple Space Tokens associated to it in a single SRM. In fact, the same User Space Token Description can be used in separate requests to reserve space (statically and dynamically). The SRM has a call (srmGetSpaceTokens) that lists all the space tokens associated with a particular Token Description. The client can loop over them and pick a token that has the desired properties, e.g. the space associated is not marked as full by a metadata query result.

Figure 3 represents a Storage Element, a Storage Area shared among a few Virtual Organizations, and User Space Token Descriptions for space reserved by those Virtual Organizations.

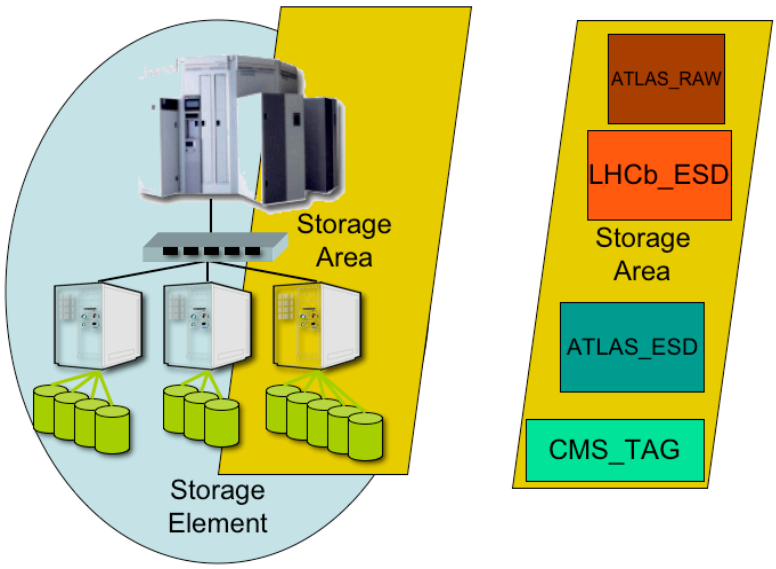

Figure 3: Storage Element, Storage Area shared among ATLAS,LHCb and CMS, and User Space Token Descriptions for ATLAS RAW, LHCb ESD, ATLAS ESD and CMS TAG. The Storage Area is shared by three VOs.

It has to be noted that in WLCG, dynamic user space reservation is optional. Furthermore, at least initially clients should not use dynamic space reservation. Therefore, for shared Storage Areas it is impossible to guarantee a given amount of space to a VO. The situation is depicted in Figure 4. Therefore, **in WLCG Storage Areas are normally dedicated to a single VO.**

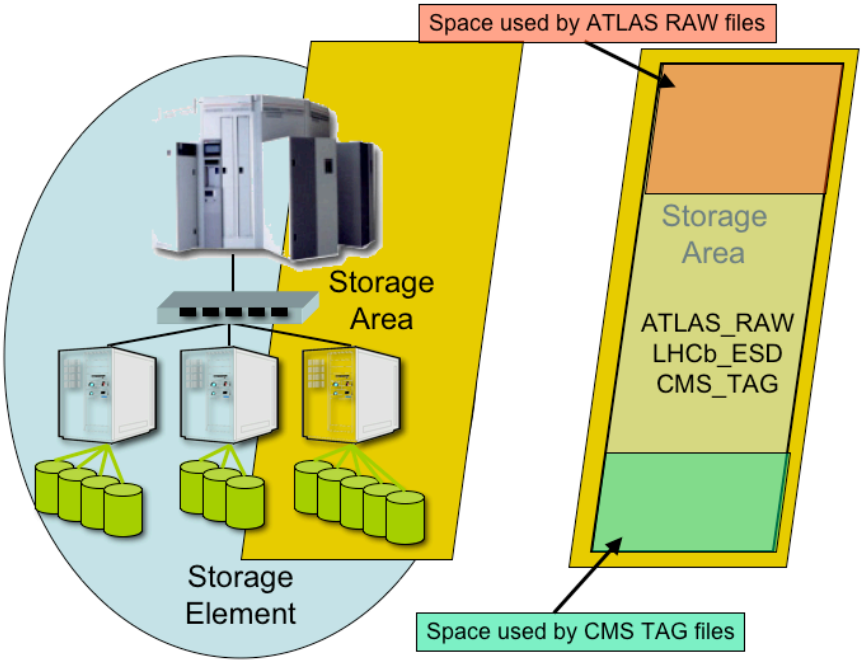

Figure 4: Storage Area shared between VOs in a system with no dynamic space reservation functionalities. The space for LHCb data gets smaller because of the space occupied by ATLAS and CMS data.

Figure 5 represents a Storage Area allocated to one VO only, its Storage Components, the User Space Token Description and the Space Token returned by a dynamic SRM Reserve Space operation (possibly generally available with SRM v2.3).

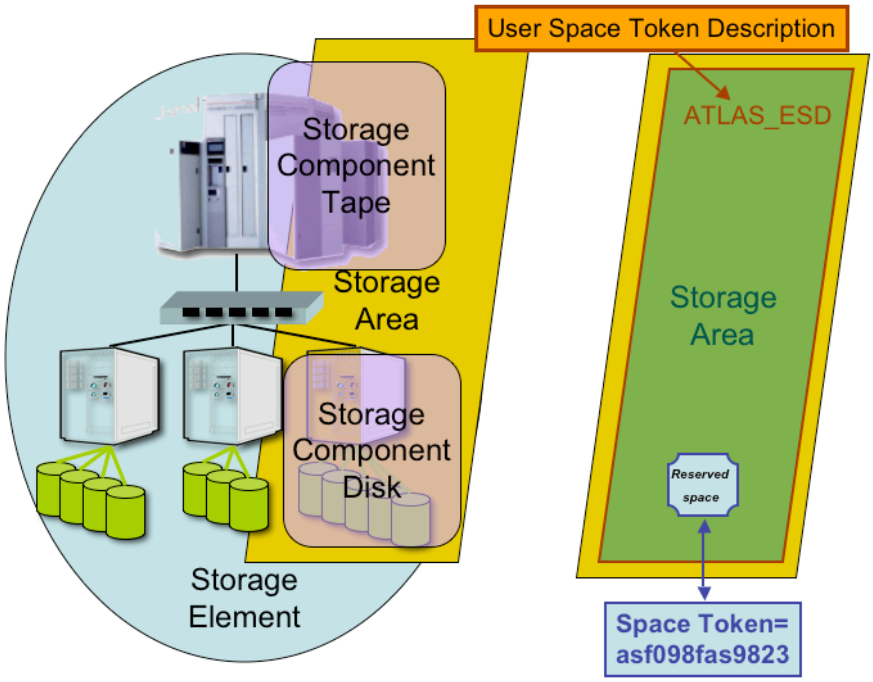

Figure 5: Storage Components and Space Tokens

In case a Space Token is not specified for an SRM Put operation, the system will use the default Storage Area that is generally shared between multiple VOs.

In the Glue Schema, a *Name* optionally identifies the Storage Area. This can be an internal identifier the site administrator uses to refer to that Storage Area. The Name can be published in the Information System.

Beside its name, a Storage Area has associated multivalued *Capability* that can vary from implementation to implementation.

Retention Policy and *Access Latency* are also properties of a Storage Area and contribute to define the Storage Class associated to this Storage Area. The SRM v2.2 specification defines the possible values for Retention Policy and Access Latency. The SRM v2.2 WLCG MoU specifies the possible values for WLCG (the ones that define the Storage Classes specified in the first chapter of this document).

Figure 6 describes the GlueStorageArea and GlueVoStorageAreaAssociation classes. These classes do not exist in GLUE v1.2.

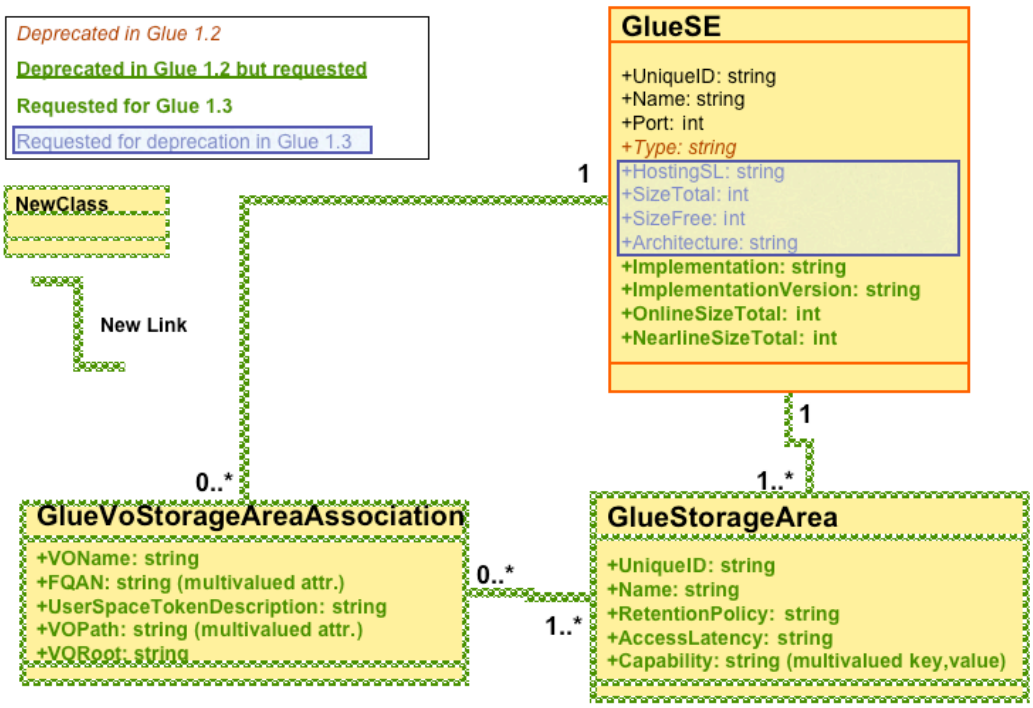

Figure 6: The Storage Area in the GLUE schema

For statically allocated space and for space reserved by administrators, the VO *User Space Token Description* is published in the Information System as

part of the *VoStorageAreaAssociation* GLUE class, in order to allow a VO to find out per SE the space and the space characteristics reserved to that VO.

The *VoStorageAreaAssociation* GLUE class allows for the association between User Space Token Description and a Storage Areas. The *VO Name* describes which VO has access to this Storage Area. A *VOMS FQAN* can be published as well to describe which users (with specific privileges) within the VO actually have access to that Storage Area.

Optionally, VO specific Storage Area associated multivalued *VOPath* can be published. The paths can be used to build VO SURLs for files, and we assume that they are protocol independent. VOPath is mandatory when referring to classic Storage Elements to find out VO specific gsiftp directories on the server. For Classic Storage Elements, the *VORoot* subdirectory published for the "file" protocol per VO, indicates the VO specific path that needs to be appended to the CEAccessPoint published in the CESEBind class to have the full VO Path for direct file access on the WN.

Note:

18. Another alternative to arrange for the "file" protocol case is indeed to continue using the current GlueSA class (this could be a motivation to reuse and redefine the GlueSA class instead of creating the new GlueStorageArea class) where both a Root and a Path are published. We leave it up to the Glue Working Group to decide what the best way to proceed is.
19. The VOName attribute in GlueVoStorageAreaAssociation cannot be avoided since we cannot assume that the VO name can be derived from the FQAN.

Storage Areas might have zero or more VoStorageAreaAssociation objects linked to them. This is to accommodate the need to publish available storage not yet assigned.

Note:

20. The WLCG Memorandum of Understanding for SRM 2.2 states that dynamic space reservation is only optionally supported. Therefore space in a Storage Area can be allocated statically and can be reserved to one specific VO. Furthermore clients shall not use dynamic space reservation initially.
21. ReserveSpace SRM calls also return a request token that can be used in subsequent calls to check on the status of the request, or to abort it. It is not to be confused with the space token.
22. Several Space Tokens can refer to a given Storage Area at a given time. In case multiple VOs can reserve space in the same storage element, the Space Token Description does not need to be unique between VOs. It is up to the specific Storage Service implementation to identify all User Space Token Descriptions a VO FQAN can access.
23. In WLCG a VO manager can explicitly control which User Space Token Descriptions are published, to make them publicly available.

The tool to do that could be similar to the one that VO managers now use to publish experiment specific software tags. This gives the ability to VO managers to hide Storage Areas from public use, if necessary.

24. The GlueStorageArea class is new and not a redefinition of GlueSA. Infact, GlueSA stands for Glue **S**torage sp**A**ce. It is up to the Glue Working Group to recommend a redefinition of the Glue v1.2 class instead.
25. If a VO would like to count up all the space allocated to it, it can sum up the space of the storage components uniquely belonging to Storage Areas allocated to that VO. The same cannot be done for Storage Areas or Components shared among VOs, unless some kind of quota system is imposed and published.

Use Case: Storage Areas can be used to match all storage services that can provide a given class of Storage to a VO FQAN. In this case, the User Space Token Description does not need to be known or published. This could be the case for non LHC VOs.

7. The Storage Paths

Note:

26. It has been decided that for WLCG a file is always ***permanent***. This means that the file remains in the system namespace unless it is explicitly removed by an SRM remove operation. Copies of the files on Disk0 can be removed by the system, if needed. Users can migrate copies of the files to/from Disk1 via an SRM ChangeSpaceForFiles operation. The file has therefore only the SURL that does not change when the file is migrated from tape to disk. The TURL of the file may change per request. TURLs have a lifetime. When the lifetime expires the system can remove the file copy pointed to by the TURL (the next time it could be served by another disk or another machine).

In SRM v2.2 the name space is orthogonal to the Storage Class, which can be derived from the Space Token supplied when the file is stored. In practice, however, there may be reasons to couple the name space with certain aspects of the quality of storage. dCache and CASTOR allow users to specify in which tape set a file should reside. dCache implements such a feature through the so-called Storage Groups. Storage Groups are associated to directory paths. Therefore in dCache a path may not only identify a Storage Class but also many other properties such as the tape set the file should reside on.
StoRM as well implements Storage Classes through paths: at the moment a Space Token is not enough to describe the Storage Class.

Open question: Can the experiments structure the directory namespace to map it appropriately to Storage Classes, tape sets etc.? In this case, all sites

would have to agree to do the same mappings. The user space token description needs to be passed anyway as input to the SRM Put/Copy calls. Can the path also determine the tape set associated to the files?

Note:

27. The name space structure ought not to be necessary, as dCache can infer the Storage Group from the space token as well in the near future, and also the StoRM developers are trying to implement a characterization of the space to be used via the space token only. However, just to be on the safe side, we would advise the VOs to apply a structure nonetheless.

8. Free and available space

It has been discussed if space is an attribute that needs to be published in the GLUE schema in other storage related classes beside the Storage Component. It was felt that only at the Storage Component level one could differentiate among the different kinds of storage available and avoid double counting of storage capacity assigned.

As far as the used space, it has been noted that such information might be inaccurate and not useful.

In WLCG tape space is considered to be infinite. As far as disk cache space, *Used space* can be of three types:

1. Space used by files
2. Space allocated by space reservation methods. Part of this space is potentially available to put files.
3. Space which is currently used by files being migrated to tape but will be available as soon as the migration is over.

We felt that it was better to define used space as the size occupied by valid files (not candidates for garbage collection) and to differentiate between used size and reserved size, as the space reserved but not yet used.
The free space can be inferred from the two values published, but in general it cannot be taken as available to store more files (there can be ACLs, quotas, … that could make such attempts fail).
As a further consideration we have to notice that the Information System can only give a snapshot of the Grid status at a given time. Therefore it could well be that a request for space reservation might fail even if the information system shows space available at a given time.

During the "WLCG Data Management Coordination Group" phone conference on October 20th, 2006 the WLCG experiments have shown no interest in having these numbers published, especially if there is no agreement on the definition. They prefer to query directly through the SRM the specific SE service. Furthermore, also DPM, d-Cache and CASTOR developers were quite reluctant in publishing such information. It was therefore decided that

"free","available","used" and "reserved" spaces should not be published. The publication of the entire "Storage Component" object is optional.

9. The Storage Component GLUE Class

Storage Components describe low-level details of the storage devices part of a Storage Area. In principle, the storage unit exposed to applications is the Storage Area. Therefore the entire Storage Component class is optional and can remain unpublished for very complex systems. Also, it is not clear what the use-cases are that justify the need to expose the Storage Components in the information system. Figure 7 represents this situation.

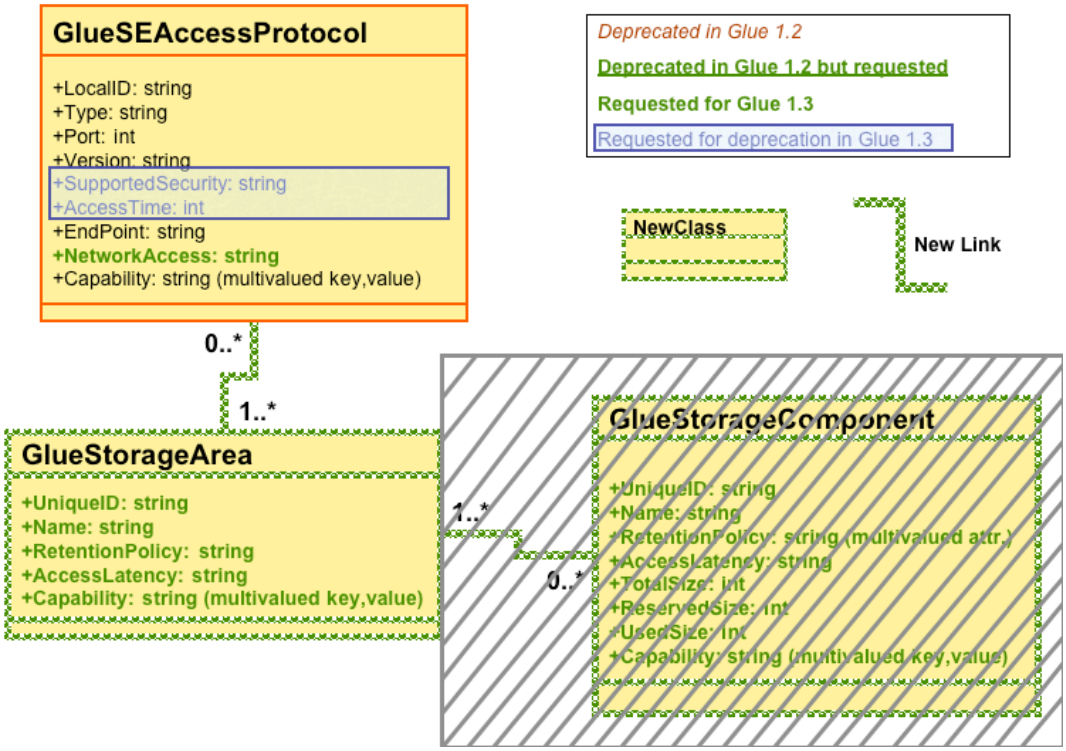

Figure 7: The Storage Component Class is optional

APPENDIX

Here we show the complete UML diagram that defines the Storage Element Service in GLUE v1.3 (cf. Figure 8). We also list all known use-cases and provide the queries that need to be performed with the new proposed schema. We show that the proposed schema can accommodate old Storage Services as well, such as the Classic SE.

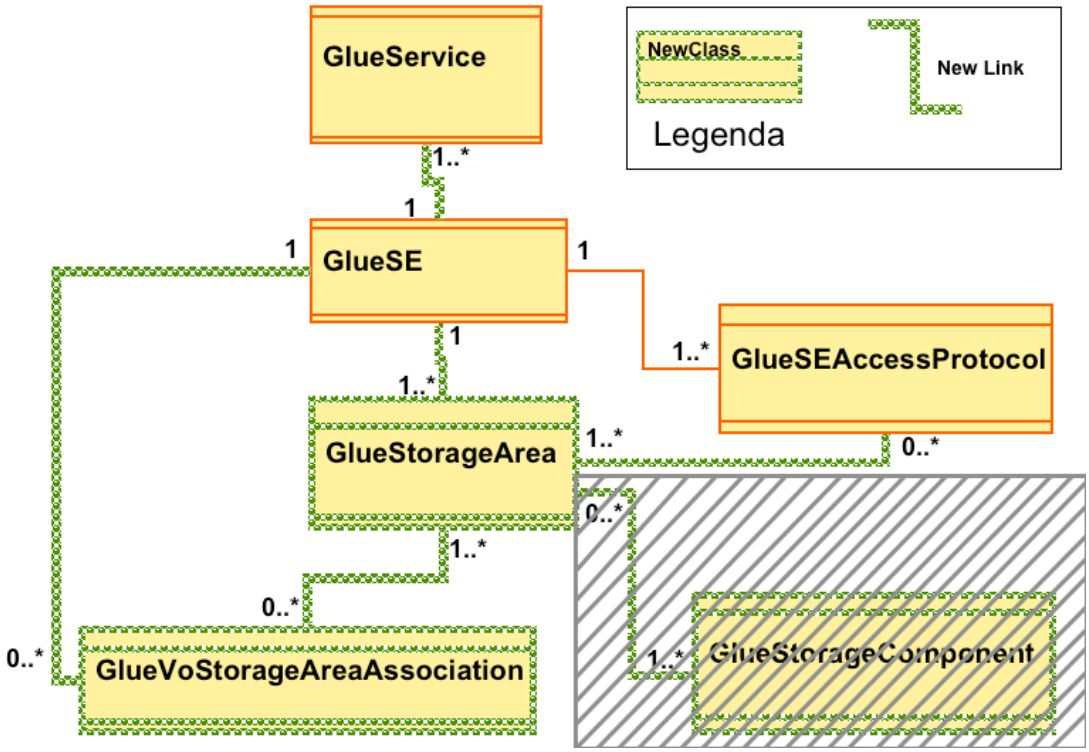

Figure 8: Storage Service Description in proposed GLUE v1.3

GFAL Queries:

a) *Find the SE type of host.*

This query is used to find out if the SE is classic or is SRM.
The query performed:
ldapsearch -x -h prod-bdii:2170 -b "o=grid"
'(GlueSEUniqueID=lxb2003.cern.ch)' GlueSEName
[…]
lxb2003.cern.ch, CERN-PROD, grid
dn: GlueSEUniqueID=lxb2003.cern.ch,mds-vo-name=CERN-PROD,o=grid
GlueSEName: CERN-PROD-LHCB:disk
[…]

Now the GlueService object publishes the SE Type. The new query is therefore:
Ldapsearch –x –h prod-bdii:2170 –b “o=grid”
‘(&(GlueServiceName=StorageElement)(GlueChunkKey=GlueSEUniqueID=lxb2003.cern.ch))’ GlueServiceType

b) Discover version and endpoint for SE Service Type srm for given host. Discover also the VO FQAN that has access to the service.
This is a call that GFAL needs to make in case of SRM v2.2:
‘(&(GlueServiceName=StorageElement)(GlueChunkKey=GlueSEUniqueID=lxb2003.cern.ch)(GlueServiceType=srm))’ GlueServiceVersion GlueServiceEndpoint GlueServiceAccessControlRule
Among the list of services returned by the query GFAL can choose and can even decide to use SRM v1 or SRM v2.
Through the AccessControlRule it is possible to discover which VO FQAN has access to the service.

c) Find the SE endpoint for host
The current query:
‘(&(GlueServiceURI=*castorgridsc.cern.ch*)(GlueServiceType=srm_v1))’ GlueServiceURI
[…]
httpg://castorgridsc.cern.ch:8443/srm/managerv1, CERN-PROD, grid
dn:
GlueServiceURI=httpg://castorgridsc.cern.ch:8443/srm/managerv1,mds-vo-name
=CERN-PROD,o=grid
GlueServiceURI:
httpg://castorgridsc.cern.ch:8443/srm/managerv1
[…]

Now the GlueService object publishes the SE Endpoint. The new query is therefore:
‘(&(GlueServiceName=StorageElement)(GlueChunkKey=GlueSEUniqueID=castorgridsc.cern.ch)(GlueServiceType=srm)(GlueServiceVersion=1))’ GlueServiceEndpoint

d) Find SE port for host (for classic SEs)
‘(GlueSEUniqueID=lxb2003.cern.ch)’ GlueSEPort
[…]
lxb2003.cern.ch, CERN-PROD, grid
dn: GlueSEUniqueID=lxb2003.cern.ch,mds-vo-name=CERN-PROD,o=grid
GlueSEPort: 8443
[…]
This query does not change in GLUE v1.3

e) Find SA Root for VO on host

The old query is:
'(&(GlueSARoot=LHCB:*)(GlueChunkKey=GlueSEUniqueID=lxb2003.cern.ch))' GlueSARoot
[…]
lhcb:lhcb, lxb2003.cern.ch, CERN-PROD, grid
dn:
GlueSALocalID=lhcb:lhcb,GlueSEUniqueID=lxb2003.cern.ch,mds-vo-name=CERN-PR
OD,o=grid
GlueSARoot: lhcb:lhcb
[…]

This call was needed for classic SEs to find out the path for direct file access on a WN for a specific VO. The path following the <VO>: tag was appended to the mount point published in CESEBindCEAccessPoint in order to obtain the full directory path to access files for that VO on a WN.
The VORoot is now published in the VoStorageAreaAssociation object.
The query in this case is the following:

'(&(GlueVoStorageAreaAssociationVOname=ATLAS)(GlueChunkKey=GlueSEUniqueID=castorgridsc.cern.ch))'
GlueVoStorageAreaAssociationVORoot

Alternatively, the Root "absolute" path can still be published in the old GlueSA object as done up to now.

f) Find SA Path for VO on host

The old query is:
'(&(GlueSALocalID=ATLAS)(GlueChunkKey=GlueSEUniqueID=lxn1183.cern.ch))' GlueSARoot GlueSAPath
[…]
atlas, lxn1183.cern.ch, CERN-PROD, grid
dn: GlueSALocalID=atlas,GlueSEUniqueID=lxn1183.cern.ch,mds-vo-name=CERN-PROD,o
=grid
GlueSARoot: atlas:atlas
GlueSAPath: /storage/atlas
[…]

The GlueSAPath was used to find out the GridFTP Path in case of a classic SE. Now such path is published per VO in the GlueVoStorageAreaAssociation object. Here are examples of the new queries:

'(&(GlueVoStorageAreaAssociationVOname=ATLAS)(GlueChunkKey=GlueSEUniqueID=castorgridsc.cern.ch))'
GlueVoStorageAreaAssociationVoPaths

or
‘(&(GlueVoStorageAreaAssociationVOname=ATLAS)(GlueChunkKey
=GlueSEUniqueID=castorgridsc.cern.ch)(GlueVoStorageAreaAssocia
tionStorageSpaceDescriptionToken=ATLAS_RAW))’
GlueVoStorageAreaAssociationVoPaths
or
‘(&(GlueStorageAreaRetentionPolicy=custodial)(GlueChunkKey=Gl
ueSEUniqueID=castorgridsc.cern.ch)(GlueChunkKey=GlueVoStorage
AssociationVoName=ATLAS))’ GlueSAUniqueID
and
‘(GlueChunkKey=GlueStorageAreaUniqueID=<from previous call>)’
GlueVoStorageAreaAssociationVoPaths

Please, note that in one of the examples above we assume that the GlueVoStorageAssocuationVoName can be used as a ChunkKey.

g) Find SE access protocol for host
‘(&(ObjectClass=GlueSEAccessProtocol)(GlueChunkKey=GlueSEUni
queID=srm.cern.ch))’ GlueSEAccessProtocolPort,
GlueSEAccessProtocolType
[…]
rfio, srm.cern.ch, CERN-PROD, grid
dn:
GlueSEAccessProtocolLocalID=rfio,GlueSEUniqueID=srm.cern.ch,m
ds-vo-name=CE
RN-PROD,o=grid
GlueSEAccessProtocolType: rfio
GlueSEAccessProtocolPort: 5001

root, srm.cern.ch, CERN-PROD, grid
dn:
GlueSEAccessProtocolLocalID=root,GlueSEUniqueID=srm.cern.ch,m
ds-vo-name=CE
RN-PROD,o=grid
GlueSEAccessProtocolType: root
GlueSEAccessProtocolPort: 1094

gsiftp, srm.cern.ch, CERN-PROD, grid
dn:
GlueSEAccessProtocolLocalID=gsiftp,GlueSEUniqueID=srm.cern.ch,
mds-vo-name=
CERN-PROD,o=grid
GlueSEAccessProtocolType: gsiftp
GlueSEAccessProtocolPort: 2811
[…]
The query is the same also with the proposed v1.3 GLUE schema for the SE.

h) Find CE access point for host (for “file” protocol or for Classic SE)
‘(GlueCESEBindSEUniqueID=srm.cern.ch)’
GlueCESEBindCEAccessPoint

[…]
castorgridsc.cern.ch, ce107.cern.ch:2119/jobmanager-lcglsf-grid_2nh_dteam,
CERN-PROD, grid
dn:
GlueCESEBindSEUniqueID=castorgridsc.cern.ch,GlueCESEBindGroupCEUniqueID=ce
107.cern.ch:2119/jobmanager-lcglsf-grid_2nh_dteam,mds-vo-name=CERN-PROD,o=gri
d
GlueCESEBindCEAccesspoint: /castor/cern.ch/grid/

castorgridsc.cern.ch, ce107.cern.ch:2119/jobmanager-lcglsf-grid_dteam, CERN
-PROD, grid
dn:
GlueCESEBindSEUniqueID=castorgridsc.cern.ch,GlueCESEBindGroupCEUniqueID=ce
107.cern.ch:2119/jobmanager-lcglsf-grid_dteam,mds-vo-name=CERN-PROD,o=grid
GlueCESEBindCEAccesspoint: /castor/cern.ch/grid/
[…]
The query is the same also with the proposed v1.3 GLUE schema for the SE.

FTS Queries:

The FTS executes the same queries as in GFAL. In particular, the Service Type (if SRM or gridftp) and its endpoint are discovered. FTS also queries for the name of the site where the service is running and the hostname. Some VO catalog FTS plugins need to discover the VOPaths.

RB Possible Queries:

i) Find a CE that has a close SE that supports ATLAS_RAW

‘(GlueVoStorageAreaAssociationStorageSpaceTokenDescription=ATLAS_RAW)’ GlueVoStorageAreaAssociationVoName

‘(&(ObjectClass=GlueSE)(GlueChunkKey=GlueVoStorageAreaAssociationVoName=<GlueVoStorageAreaAssociationVoName>))’
GlueSEUniqueID
<SEUniqueID(i)>
foreach i ; do
‘(GlueCESEBindSEUniqueID=<SEUniqueID(i)>)’
GlueCESEBindCEName
done

The query could be simplified if in the GlueVoStorageAreaAssociation the GlueSEUniqueID is stored (as a chunkkey?).

j) Find a CE that has a close SE with an SA accessible by a given VO FQAN that supports "replica" as a retention policy

'(&(GlueStorageAreaRetentionPolicy=replica)(GlueChunkKey=GlueVoStorageAssociationVoName=<VO>))' GlueStorageAreaUniqueID GlueChunkKey
SAUniqueIDs(i), SEUNiqueIDs(i)

'(GlueVoStorageAreaAssociationFQAN=<FQAN>)' GlueChunkKey
SAUniqueIDs(j)

```
foreach j; do
foreach i; do
   if SAUniqueIDs(i) = SAUniqueIDs(j); then
      CE(n)='(GlueCESEBindSEUniqueID=SEUniqueIDs(i))'
GlueCESEBindCEName
   fi
done
done
```

k) Find an SE close to a specific CE which supports ATLAS_ESD and the protocol gsidcap on WAN

'(&(ObjectClass=GlueSEAccessProtocol)(GlueSEAccessProtocolNetworkAccess=WAN)(GlueSEAccessProtocolType=gsidcap))'
GlueChunkKey
SEUniqueIDs(i)

'(GlueVoStorageAreaAssociationSpaceTokenDescription=ATLAS_ESD) GlueChunkKey
SEUniqueIDs(j)

```
foreach j; do
foreach i; do
   if SEUniqueIDs(i) = SEUniqueIDs(j); then
        CE(n)='(GlueCESEBindSEUniqueID=SEUniqueIDs(i)'
              GlueCESEBindCEName
   fi
done
done
```

"Others" Queries:

The SFT system publishes the SE implementation and its version. This is useful in order to find out if sites have performed an upgrade, for instance.

Ldapsearch –x –h prod-bdii:2170 –b “o=grid” ‘(GlueSEImplementation=DPM)’ GlueSEImplementationVersion

The SAM system stores internally the Total Size of an SE, which is now directly published in the GlueSE class as the sum of OnlineSizeTotal and NearlineSizeTotal.

l) Find the TotalSize of all StorageAreas with token description ATLAS_RAW

```
‘(GlueVoStorageAreaAssociationSpaceTokenDescription=ATLAS_RA
W) GlueChunkKey
SAUniqueIDs(i)

foreach i; do
TotalComponentSize(j)=‘(GlueObject=GlueStorageComponent)(Glu
eChunkKey=SAUniqueIDs(i) GlueStorageComponentTotalSize
foreach k; do
   ATLAS_RAWTotalSize=ATLAS_RAWTotalSize+
                              TotalComponentSize(j)
done
done
echo ATLAS_RAWTotalSize
```