J Grid Computing (2009) 7:309–317
DOI 10.1007/s10723-009-9127-2

Practical Grid Storage Interoperation

Interoperation of SRM and SRB Now

Jens Jensen · Roger Downing · Derek Ross · Matt Hodges · Alex Sim

Received: 31 October 2008 / Accepted: 12 August 2009 / Published online: 26 August 2009

Abstract In global Grids, interoperation is important. It enables communities to work together, helps prevent vendor lock-in, and in principle enables "cloud-like" resource provision by permitting different resources to meet needs from other communities. In this paper, we discuss a practical example of achieving interoperation between storage resources, and the lessons learned. The aim is to meet current use cases for interoperation with no additional software development. Apart from the practical results, experiences from this work will be relevant to other interoperation activities.

Keywords Grid storage · SRM · SRB · gLite · Interoperation

J. Jensen (✉) · D. Ross · M. Hodges
Rutherford Appleton Laboratory, Harwell Science and Innovation Campus, Oxon, OX11 0QX, UK
e-mail: jens.jensen@stfc.ac.uk

R. Downing
Daresbury Laboratory, Warrington, Cheshire WA4 4AD, UK

A. Sim
Lawrence Berkeley National Laboratory, Berkeley, CA 94720, USA

1 Introduction

With large Grid resources being shared by scientists across the world, standardisation and interoperation become increasingly important:

- It enables data sharing between different scientific communities who were previously using distinct interfaces to resources.
- It enables sharing resources and moving data and computation across Grids, thus enabling users to use more service providers, and helps prevent users from being "locked in" to a single service provider.
- Interoperating interfaces make it possible for service providers to use different implementations to provide services. Service providers can choose implementations that fit their infrastructure best, and for users, the interoperating interfaces make the Grid appear more "cloud-like" because their services can, at least in principle, be seamlessly migrated.
- As a side benefit it promotes stability of software, as interfaces are used outside their usual envelope (but hopefully within their specification).
- It is useful in promoting more "advanced" middleware stacks, since users have the assurance they can move back to the "less advanced" middleware if the more advanced

stack is not to their liking. Data can be migrated between the Grids when required.

This paper looks at storage resources. Together, storage elements with the Storage Resource Manager (SRM) interface and the Storage Resource Broker (SRB) manage many petabytes of data storage for Grids across the world. Interoperation between the two has been discussed for years, and the work presented in this paper was the first to actually achieve it: the goal was to transfer files between the two using the gLite middleware stack, and without doing any software development.

In the UK, we have two national Grid infrastructures, the National Grid Service, currently based on Globus and using SRB, and the UK Grid for particle physics (GridPP) using gLite and SRM. This is not uncommon: an analogous situation can be found in the US, where OpenScienceGrid uses SRM and TeraGrid uses SRB.

Our primary motivation was to get such Grids to interoperate. They already interoperate on many levels, with authentication and authorisation, and information systems, being the most obvious successes. Being able to transfer data between storage systems on these Grids extends the opportunities for interoperation.

Thus, data interoperation between these Grids is relevant and is likely to become increasingly important as Grids within the European Union move to a National Grid Infrastructure model—these NGIs will not in general run a single middleware stack and users will need interoperation to make the most of the resources.

This work will hopefully eventually be replaced by more advanced work (see "related work" in Section 3.3), but the lessons learned (Section 3.4) will remain relevant for other interoperation activities.

2 Background

SRM is the Storage Resource Manager, a protocol for interfacing the Grids to mass storage systems [10]. The protocol is an open standard, standardised under the OGF [16]. (In practice, we take the view that a "Grid" is based on either gLite or Globus, such as the Grids EGEE and OSG.) It forms the control interface to a *Storage Element* (SE), which itself forms part of a Grid with higher level services such as Computing Elements, Resource Brokers, data movers, metadata and replica catalogues, etc. The SE itself also provides data transfer interfaces, one of which is required by the standard to be GridFTP [2]. Finally, it provides information interfaces which are based on or derived from Globus MDS ("Metacomputing Directory Service," [11]). By abuse of notation, an SE with an SRM interface is often called "an SRM."

It is necessary to give a simplified overview of SRM file management (for details, see, e.g., [12].) In Grids using SRM, a file is referenced with a Site (or Storage) URL, or SURL.[1] When a client opens a file, it passes the SURL and the desired transfer protocol, and the SRM readies the file on a disk pool supporting the desired protocol.[2] It returns a Transfer URL (TURL) which the client must use to transfer the file. SURLs are permanent, and TURLs are normally ephemeral, allocated for individual transfers.

SRB, the Storage Resource Broker from SDSC [14], implements what is referred to as a "data Grid," aiming to provide a complete interface between the user (or the user's application) to the storage system. SRB itself provides its own catalogue, MCAT, which also has limited ability to manage user metadata (key/value pairs). Its strength is that it can manage multiple storage resources utilising diverse technologies ranging from POSIX file systems to large-scale tape arrays and present all of these as a single unified filesystem.

An application accessing files held in SRB does not need to know how or even where the files are physically stored. The MCAT stores a mapping between a logical filename and the physical location of the data, meaning that a file may be physically relocated or replicated within SRB and yet the logical positioning of it remains the same. This

[1]E.g., srm://host.example.ac.uk/mydir/myfile

[2]E.g., gsiftp://disk001.example.ac.uk/filesystem/tmp0001 (GridFTP URLs use the scheme *gsiftp*.)

gives benefits when managing large-scale digital repositories since it simplifies the transfer of data to long-term storage more suited to preservation activities but still provides applications with direct access to the contents of the archive.

There are currently six different SRM implementations in the world. Interoperability between them has been achieved by extensive and carefully coordinated testing [17]. In contrast, we know of only two implementations of SRB, the standard SDSC one used in this work and the Nirvana implementation forked from it from version 1.x. We have not tested whether these two implementations interoperate or whether this work can be done on the Nirvana implementation.

The background to this work is the GIN ("Grid Interoperability Now") activity in the Open Grid Forum [13], where groups formed to look into interoperability of data (GIN-DATA), jobs, information (GIN-INFO), and authentication and authorisation (GIN-AUTH). The work described in this paper fits into the GIN-DATA activity, and depends on the interoperability already achieved in GIN-INFO. GIN distinguishes between *interoperability* and *interoperation* [15]: *interoperability* refers to native communication between the resources using standard protocols, and *interoperation* refers to what we can do today, building glue and short term fixes as needed, to move (in this case) data between the resources.

We have previously demonstrated or presented earlier results in this area, using the SRM, not FTS (see next section), as the data mover at the SuperComputing 2007 conference, and with gLite as the data mover at the EGEE User Forum in Clermont-Ferrand in Feb 2008. The work in this paper provides more details, but also extends upon the previous results, bringing it up to date with recent middleware developments.

3 Main Result—Interoperation

The main result of this paper is to transfer files between an SRM and an SRB instance using data management facilities from gLite [12]. Ideally, the data is to be transferred with replica catalogues tracking the copies of the data. GridFTP is used as the core data mover protocol, using the fact that GridFTP is the only common data transfer protocol between SRM implementations, and that a GridFTP interface was contributed to SRB by ANL [5].

The core principle is to pretend that the SRB is a so-called "Classic SE." The Classic SE is essentially a GridFTP server with an information system bolted on to it, previously used in the absence of mature SRM implementations [7]. In SRM terminology, this means that all SURLs are GridFTP URLs, and the TURL is always the same as the SURL.

This works as long as the data movers support Classic SEs. Although the gLite-based Grids have few Classic SEs left and there has been talk about dropping support for them, FTS (the gLite File Transfer Service [4]) still supports them. (Should support be fully discontinued, moving data can still be done using "manual methods" described below.) FTS is primarily used in this context to improve ease of use, error handling, and transfer scheduling. If support for GridFTP SURLs is discontinued in the replica catalogues, we will have to take the "safer" approach to tracking replicas, see Section 3.1.

Figure 1 shows the two main modes of operation, showing transfer from the SRM to the SRB. The left hand side of the diagram shows the "normal" mode of operation where FTS transfers the data: FTS connects to the SRM and gets a Transfer URL (TURL) which is then copied to the SRB using normal GridFTP 3rd party copying.[3] FTS does not depend on the information system. Moreover, transfers from SRB to SRM work precisely the same way when using FTS: FTS "creates" the file in the SRM (i.e., calls srmPrepareToPut [16] to get a TURL), and once the TURL is ready, calls the SRB GridFTP server to transfer the file.

The right hand side of Fig. 1 shows the "manual" mode of operation. Using gLite command line tools, we get a TURL from the SRM and use `globus-url-copy` to transfer the file—note that the GridFTP call is made to the destina-

[3]The client moves data between two remote servers so data need not be moved to the client first.

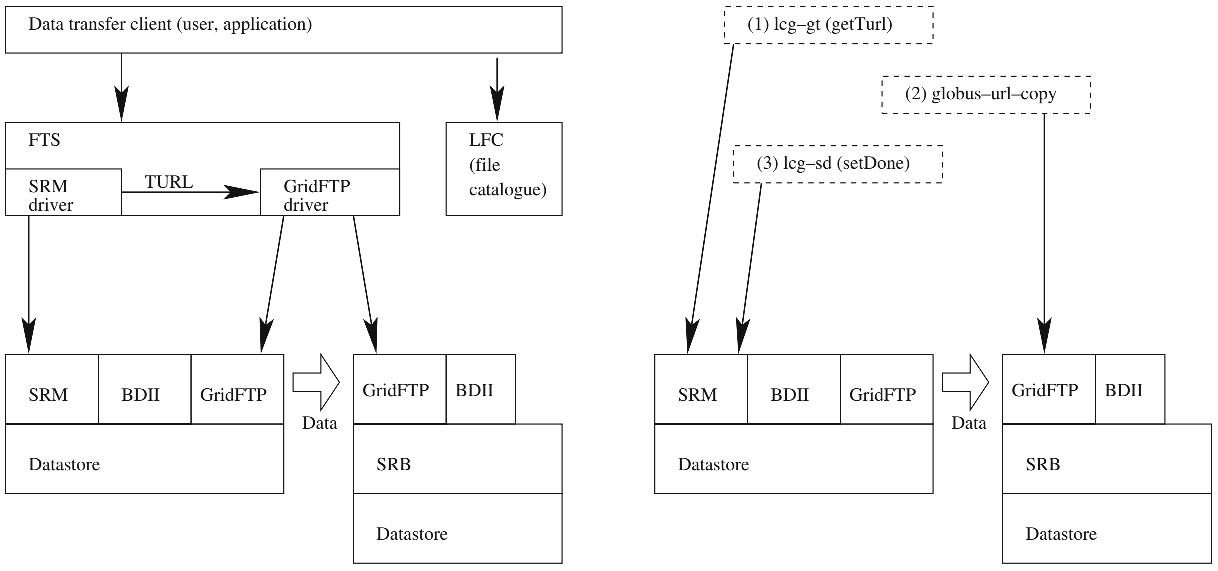

Fig. 1 Main SRM-SRB interoperation mode

tion host which then opens a connection to the source host and transfers the data. In this diagram, the file is not registered in the catalogue. In principle, this could still be done with the `lcg-rf` command provided it supported GridFTP SURLs (see Section 3.1). This process can of course be scripted.

Unlike FTS, the “manual” method is not symmetric: copying from SRB to SRM with `lcg-gt` does not work *unless* the target file (in the SRM) already exists (in other words, the `lcg-gt` does not create the file.) The solution is to use `lcg-cr` (copy and register) which copies to a destination SRM, and is able to copy from a remote GridFTP SURL.

Figure 2 show another two modes of achieving interoperation. On the left hand side, the SRM is asked to do an SRM copy: this will normally trigger a put or get cycle (as appropriate) to a remote SRM, but in this case we ask it to do a copy

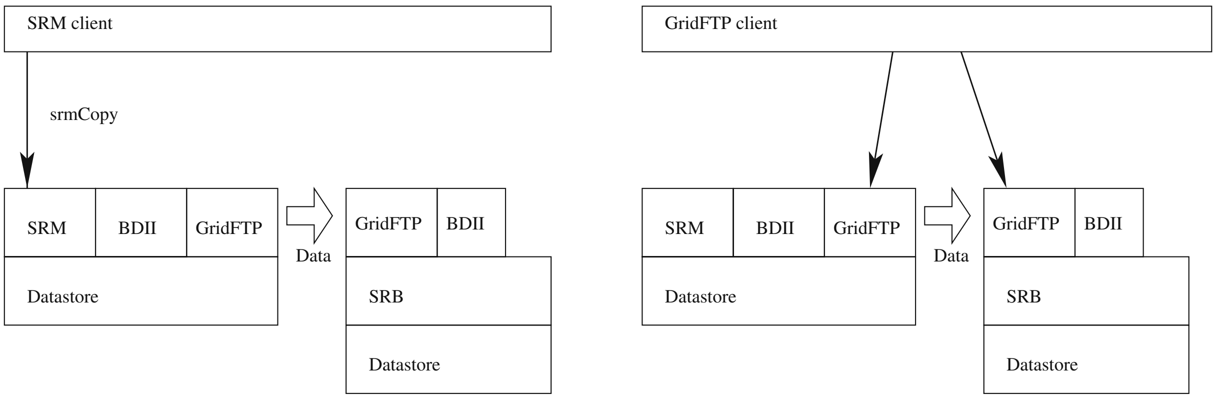

Fig. 2 Secondary SRM-SRB interoperation mode

with a GridFTP SURL. This requires that the SRM supports srmCopy (not all do), and that they support it for GridFTP SURLs. We demonstrated this at Supercomputing 2007 with the dCache implementation from DESY/Fermilab [8], and with the BeStMan implementation from LBNL [6].

Finally, the right hand side shows a GridFTP copy from one of the SRM's disk servers to SRB. This also works in both directions, we have demonstrated this on dCache, but it has the disadvantage that one needs to know the hostname and path (i.e., GridFTP TURL) of the file on the SRM side. This, however, will not work with all SRM implementations.

3.1 Catalogues

In gLite Grids, SEs are relatively low-level storage services; higher level services such as FTS move data, and track replicas of files in replica catalogues.

There are two types of catalogues in gLite; the LFC (file catalogue) which keeps track of replicas of files, and the metadata catalogues. Although we have later investigated bringing metadata catalogues into this interoperation activity, it falls outside the scope of this paper. The file replica catalogue, however, is quite relevant: the copy in SRB can be tracked like any other replica of a file.

One other distinction between FTS and the "manual" method is that FTS does not, by default, update the replica catalogue. This approach was chosen because it is often quicker to bulk register files in the catalogues once a batch of FTS transfers have been done.

There are two approaches to tracking our SRB-replicas of files in LFC: the "safe" way and the "less safe" way.

3.1.1 Safe Replica Tracking

The "safe" approach to tracking replicas is to copy *all* data from SRB to an SRM. The SRM SURL is then registered in the LFC as before. This approach has the advantage of bringing all SRB data into the gLite middleware framework, thus ensuring that any further processing of the data will work as it does for any other data in gLite. The disadvantage, of course, is that data is duplicated: it is held in both SRM and SRB. This data movement can conveniently be done with `lcg-cr` which copies the file from the GridFTP SURL in SRB and registers the *SRM* SURL, the *target* SURL, in the LFC.

3.1.2 "Less Safe" Replica Tracking

The second approach is to register GridFTP SURLs pointing to SRB in the LFC. It is potentially less safe because gLite does not guarantee supporting GridFTP SURLs. In our tests, it worked fine with everything described above, in particular it worked with the `lcg-cr` (copy and register) command, as well as replication and deletion. The only problem we found was that it was not possible to independently register an existing file (in SRB) with `lcg-rf`; if it is created with another command (e.g., `lcg-rep`, replicate) then it worked.

3.2 SRB Configuration

Integration of SRB into an SRM was achieved through the use of an SRB Data Storage Interface (DSI, [9]) for GridFTP. It can be downloaded from San Diego Supercomputing Centre's SRB web pages [5].

Table 1 SRB/Globus environment variables

Variable	Description
GLOBUS_LOCATION	Location of the Globus installation
GLOBUS_SRB_HOSTNAME	DNS name or IP address plus TCP port number of an SRB server
GLOBUS_SRB_DN	x.509 DN string of the certificate held by the SRB server
GLOBUS_SRB_DEFAULT_RESOURCE	Name of an SRB resource to be used when sending files into the SRB system
GRIDMAP	Full path of a modified Grid mapfile used by the GridFTP server.

The DSI effectively treats an SRB Grid as a backend storage system for a GridFTP server. Through a modified Grid mapfile one can use x.509 [3] authentication into the SRB for access control. Environment variables are specified which are used by the DSI and translated into parameters used when interacting with SRB. These parameters are listed in Table 1.

3.2.1 Limitations in Using SRB

Since SRB presents storage as a set of discrete resources one must specify one in some fashion when copying data into the SRB Grid; no such specification is required when copying data out of SRB however. The "GLOBUS_SRB_DEFAULT_RESOURCE" setting described above is used to specify which resource should be used in copying data into the SRB Grid. This is fixed for each invocation of the GridFTP server using the SRB DSI, and no method is apparent for altering this for each processed request. For larger or more complex SRB Grids it would be necessary to provide one GridFTP server per resource available for writing. If the GridFTP server with SRB DSI is treated as a read-only resource then no problem exists, any data held in the SRB Grid can (assuming complete network connectivity) be accessed via GridFTP.

Another potential limitation is the necessity of maintaining a separate gridmap file for use by the SRB-enabled GridFTP server. Any integration with VOMS [1] would have to take this into account.

It should also be noted that we were unable to make the transfers work unless the GridFTP server was running on the standard GridFTP port, 2811. The `lcg` command line utilities described above apparently did not pick up a non-standard port number, even if it was published in the information system.

Finally, the definition of a fixed SRB server hostname for use by the DSI could pose problems for load-balancing and/or failover, though this is not a problem specific to the DSI and exists for any SRB client.

3.2.2 FTS Configuration

To enable FTS to transfer data between the systems, we had to configure it to recognise the Classic SE (as mentioned above, it does not query the information system directly).

In the file `/opt/glite/etc/services.xml` on the FTS server, the following XML snippet is used:

```
<service name="gsiftp://kisumu.esc.
 rl.ac.uk:2811">
  <parameters>
    <endpoint>gsiftp://kisumu.esc.
     rl.ac.uk:2811
    </endpoint>
    <type>GridFTP</type>
    <version>1</version>
    <site>RAL-LCG2</site>
    <wsdl>unset</wsdl>
    <volist>
      <vo>dteam</vo>
    </volist>
  </parameters>
</service>
```

Once the channels are configured correctly, transfers can then be submitted with `glite-transfer-submit` (the FTS client which schedules transfers) using SURLs with `srm` schema (i.e., `srm://`) to/from SRM, and `gsiftp` (GridFTP) to/from SRB.

3.2.3 Information System

The Grids cannot "see" the SE if it is not published in the information system. To publish our SRB as an SE, we created a minimal entry for service discovery (ignoring the other main use case, usage accounting), and published it in a site EGEE BDII information provider (sometimes, the resource BDII is run on the same host, because the information then disappears if the host crashes, but this is not a strict requirement). The basic LDIF (for Globus MDS or the EGEE BDII) is replicated in Appendix A.

Once the information was published, we were able to successfully use the lcg-utils command line tools to copy a file into the SRB and register the newly created file into the LFC for the Virtual Organisation (VO) that we had used to perform the test, `dteam`.

3.3 Related and Future Work

Other service providers with both SRM and SRB services have looked at using GridFTP to transfer data to SRB. We were to our knowledge the first to use this implementation, rather than attempting to write an adapter interface.

As a longer term solution, Academia Sinica Grid Computing (ASGC) are building an SRM interface to SRB [18] (and conversely, they are of course aware of our work, we even presented together at the EGEE User Forum in Feb. 2008). Such work has been proposed before but is only now coming to fruition. By building such an interface we should obtain long term interoperability, without depending on maintaining legacy interfaces.

iRODS, the proposed replacement for SRB, has a rule engine which will remove one of the limitations of the current system: the target resource can be decided by a rule rather than having to be set manually. Of course, iRODS will need a GridFTP interface—there is no other data mover protocol shared by the SRMs.

3.4 Lessons Learned

Starting with a basic idea, to bring SRB into gLite as a "classic SE", we brought together experts in each of the relevant areas—SRM, SRB, data movers, and information systems. As in projects generally, it was important to have a leader coordinating the work, ensuring that the work is driven by use cases. The experts generally had no knowledge of each other's fields of expertise, so it was essential for the team leader to have at least a basic level of knowledge of *all* the components. Moreover, as we had constrained ourselves to not do software development, it was easier to persuade the experts and their managers that this was doable.

Like many interoperation (as opposed to long-term interoperability, see Section 2) activities, this work is more a proof-of-concept prototype than a production infrastructure. In bringing this to production, the main issues are:

– Performance: since we need to treat the components as black boxes, we cannot alter the performance of any component. Limits of performance (total file volume, total number of files) seem to be defined by the components rather than by bringing them together. These limits (beyond the scope of this paper) should be understood to ensure we do not promise interoperation where this cannot practicably be achieved.
– Since the work brings together components external to this project, bringing the whole to production requires support for these.
– The workarounds chosen to make things work. For example, permissions in SRB are currently relaxed: we have assumed that all members in a given virtual organisation have write access to all the virtual organisation's files.
– Managing expectations. It is easy to summarise the work and say "SRM and SRB now interoperate" but they in fact only do so within the framework we have built. Specifically, we built the framework to make SRB accessible in a gLite Grid.
– Higher level services: more recent work has looked at bringing interoperation together for metadata management and file catalogues and are outside the scope of this paper. It should be made clear that we are transferring *files* rather than *data* (which will most likely have associated metadata.)

4 Conclusion

This paper presents results achieved in interoperation between SRM and SRB, two Grid "storage elements" that hitherto had not been able to interoperate. Interoperation is achieved by enabling movement of files between the two using gLite data movers—in other words, we bring SRB into gLite as a simple storage element (largely neglecting the underlying features of SRB.) It should be noted that interoperation between the storage resources was achieved only within the framework described in the present paper; thus it would be inaccurate to say that "SRM and SRB now interoperate."

Various ways of interoperating were discussed, e.g., with our without tracking replicas in catalogues, and using either FTS or "lower level" services for moving data. We found this work most useful in promoting resources to users: primarily to enable users using SRB to "try out" SRM on gLite Grids with the confidence that they will not lose their files.

Acknowledgements The authors would like to thank the reviewers for useful comments. This work was funded by the Science and Technology Facilities Council (www.scitech.ac.uk) and the GridPP project (www.gridpp.ac.uk). A. Sim was funded by LBNL. The document was typeset with LaTeX.

Appendix A

```
dn: GlueServiceUniqueID=gsiftp://kisumu.esc.rl.ac.
 uk:2500,mds-vo-name=local,o=grid
objectClass: GlueTop
objectClass: GlueService
objectClass: GlueKey
objectClass: GlueSchemaVersion
GlueServiceUniqueID: gsiftp://kisumu.esc.rl.ac.
 uk:2500/
GlueServiceName: RAL-PPS-classicSE
GlueServiceType: classic
GlueServiceVersion: 1.1.0
GlueServiceEndpoint: gsiftp://kisumu.esc.rl.ac.
 uk:2500
GlueServiceURI: gsiftp://kisumu.esc.rl.ac.uk:2500
GlueServiceAccessPointURL: gsiftp://kisumu.esc.rl.
 ac.uk:2500
GlueServiceStatus: OK
GlueServiceStatus: OK
GlueServiceStatusInfo: No Problems
GlueServiceWSDL: unset
GlueServiceSemantics: unset
GlueServiceStartTime: 1970-01-01T00:00:00Z
GlueServiceOwner: LCG
GlueServiceAccessControlRule: dteam
GlueForeignKey: GlueSiteUniqueID=RAL-PPS
GlueSchemaVersionMajor: 1
GlueSchemaVersionMinor: 2

# kisumu.esc.rl.ac.uk, RAL-SRB, local, grid
dn: GlueSEUniqueID=kisumu.esc.rl.ac.uk,
 mds-vo-name=local,o=grid
objectClass: GlueSETop
objectClass: GlueSE
objectClass: GlueInformationService
objectClass: GlueKey
objectClass: GlueSchemaVersion
GlueSEUniqueID: kisumu.esc.rl.ac.uk
GlueSEName: RAL-SRB:disk
GlueSEPort: 2500
GlueSESizeTotal: 0
GlueSESizeFree: 0
GlueSEArchitecture: multidisk
GlueInformationServiceURL: ldap://kisumu.esc.rl.ac.
 uk:2170/mds-vo-name=resource,o=grid
GlueForeignKey: GlueSiteUniqueID=RAL-PPS
GlueSchemaVersionMajor: 1
GlueSchemaVersionMinor: 2

# gsiftp, kisumu.esc.rl.ac.uk, RAL-SRB, local, grid
dn: GlueSEAccessProtocolLocalID=gsiftp,
 GlueSEUniqueID=kisumu.esc.rl.ac.uk,
 mds-vo-name=local,o=grid
objectClass: GlueSETop
objectClass: GlueSEAccessProtocol
objectClass: GlueKey
objectClass: GlueSchemaVersion
GlueSEAccessProtocolLocalID: gsiftp
GlueSEAccessProtocolType: gsiftp
GlueSEAccessProtocolEndpoint: gsiftp://kisumu.esc.
 rl.ac.uk
GlueSEAccessProtocolCapability: file transfer
GlueSEAccessProtocolVersion: 1.0.0
GlueSEAccessProtocolPort: 2500
GlueSEAccessProtocolSupportedSecurity: GSI
GlueChunkKey: GlueSEUniqueID=kisumu.esc.rl.ac.uk
GlueSchemaVersionMajor: 1
GlueSchemaVersionMinor: 2

# classic, kisumu.esc.rl.ac.uk, RAL-SRB, local,
 grid
dn: GlueSEControlProtocolLocalID=classic,
 GlueSEUniqueID=kisumu.esc.rl.ac.uk,
 mds-vo-name=local,o=grid
objectClass: GlueSETop
objectClass: GlueSEControlProtocol
objectClass: GlueKey
objectClass: GlueSchemaVersion
GlueSEControlProtocolLocalID: classic
GlueSEControlProtocolType: classic
GlueSEControlProtocolEndpoint: classic
GlueSEControlProtocolCapability: control
GlueSEControlProtocolVersion: 1.0.0
GlueChunkKey: GlueSEUniqueID=kisumu.esc.rl.ac.uk
GlueSchemaVersionMajor: 1
GlueSchemaVersionMinor: 2

# dteam, kisumu.esc.rl.ac.uk, RAL-SRB, local, grid
dn: GlueSALocalID=dteam,GlueSEUniqueID=kisumu.
 esc.rl.ac.uk,mds-vo-name=local, o=grid
objectClass: GlueSATop
objectClass: GlueSA
objectClass: GlueSAPolicy
objectClass: GlueSAState
objectClass: GlueSAAccessControlBase
objectClass: GlueKey
objectClass: GlueSchemaVersion
GlueSARoot: dteam:/tahdszone/home/testahds2.tahds
GlueSAPath: /
GlueSAType: permanent
GlueSALocalID: dteam
GlueSAPolicyMaxFileSize: 10000
GlueSAPolicyMinFileSize: 1
GlueSAPolicyMaxData: 100
GlueSAPolicyMaxNumFiles: 10
GlueSAPolicyMaxPinDuration: 10
```

```
GlueSAPolicyQuota: 0
GlueSAPolicyFileLifeTime: permanent
GlueSAStateAvailableSpace: 155353952
GlueSAStateUsedSpace: 496716
GlueSAAccessControlBaseRule: dteam
GlueChunkKey: GlueSEUniqueID=kisumu.esc.rl.ac.uk
GlueSchemaVersionMajor: 1
GlueSchemaVersionMinor: 2
```

References

1. Alfieri, R., et al.: From gridmap-file to VOMS: managing authorization in a Grid environment. Future Gener. Comput. Syst. **21**, 549–558 (2005)
2. Allcock, W. (ed.): GridFTP: protocol extensions to FTP for the Grid, OGF GFD.20. http://www.ogf.org/documents/GFD.20.pdf
3. Astalos, J., et al.: International Grid CA interworking, peer review and policy management through the European DataGrid certification authority coordination group. In: Advances in Grid Computing—EGC 2005, pp. 285–295 (2005)
4. Badino, P.: The gLite file transfer service, EGEE User Forum 2006. CERN. http://egee2.eu-egee.org/egee_events/userforum (2006)
5. Bresnahan, J.: Contributed software. http://www.sdsc.edu/srb/index.php/Contributed_Software (2007)
6. BeStMan: Berkeley Storage Manager (BeStMan). http://datagrid.lbl.gov/bestman/ (2009)
7. Burke, S., et al.: GridPP: running a production Grid. In: Proc. UK e-Science All-Hands 2006, pp. 598–605 (2006)
8. dCache: Scope of the project. http://www.dcache.org/ (2009)
9. The Globus Project: SRB-DSI. http://www.globus.org/toolkit/docs/4.0/data/gridftp/GridFTP_SRB.html (2009)
10. Jensen, J., Synge, O.: Experiences with implementing the SRM protocol and configuring and packaging modular systems. In: Proc. IEEE Mass Storage Systems and Technologies Symposium on Local to Global Data Interoperability, pp. 93–96 (2006)
11. Fitzgerald, S., et al.: A directory service for configuring high perfomance distributed computations. In: Proc. 6th IEEE Symp. on High Performance Dist. Comp., pp. 365–375 (1997)
12. Laure, E., et al.: Programming the Grid with gLite. In: Computational Methods in Science and Technology. Poznań (2006)
13. Laure, E., Riedel, M., et al.: Open Grid forum Grid interoperability now activity. http://forge.gridforum.org/sf/projects/gin (2009)
14. Rajasekar, A.K., Wan, M.: SRB & SRBRack—components of a virtual data Grid architecture. In: Advanced Simulation Technologies Conference (ASTC02), San Diego, 15–17 April 2002
15. Riedel, M., et al.: Interoperation of world-wide production e-Science infrastructures. J. Grid Comp. (2009, in press)
16. Shoshani, A., Sim, A. (eds.): The storage resource manager interface specification, version 2.2. Open Grid forum document GFD.129. http://www.ogf.org/documents/GFD.129.pdf (2008)
17. Sim, A., et al.: SRM interoperation test reports (GGF 17 and 18). http://datagrid.lbl.gov/doc.html (2007)
18. Tsai, F: The development of SRM interface for SRB. In: EGEE 3rd User Forum. Clermont-Ferrand. http://egee-uf3.healthgrid.org/ (2008)