

# 1           **Replica Management Architecture and Integration**

2  
3                           Craig E. Tull  
4                           NERSC, LBNL, PPDG

## 5 6 7                           **ABSTRACT**

8           This report summarizes the results of a study of Replica Management products  
9           connected with the Particle Physics Data Grid (PPDG) project. Nine different  
10          products are compared and contrasted and strategies for integration and  
11          interoperability of these products are explored. A set of recommendations for  
12          future PPDG work are set forth to promote replica management integration and  
13          interoperability.

14  
15  
16  
17  
18                           Version: \$Id: report.txt,v 1.45 2003/09/25 04:20:06 cetull Exp \$  
19  
20

**INTRODUCTION & OVERVIEW ..... 3**

    PARTICIPANTS ..... 3

*EDG-WP2 RLS* ..... 4

*Globus-EDG RLS*..... 4

*Magda*..... 4

*Phenix*..... 4

*SAM*..... 5

*SRB* ..... 5

*SRM*..... 5

*STAR*..... 6

*Jlab* ..... 6

**TECHNICAL COMPARISON & CONTRAST ..... 6**

    EDG-WP2 AND GLOBUS RLSS ..... 6

    EXPERIMENT SOLUTIONS (MAGDA, PHENIX, STAR) ..... 8

    STANDALONE SYSTEMS (SAM, SRB) ..... 9

    SRM..... 10

    TECHNOLOGY CHOICES ..... 10

*Conventional Interfaces* ..... 11

*Protocol Interfaces*..... 12

*Database Interfaces* ..... 12

**INTERFACE ISSUES..... 13**

**CONCLUSION & WAY FORWARD..... 16**

**REFERENCES ..... 20**

**GLOSSARY ..... 21**

## Introduction & Overview

This report summarizes the results of a study of Replica Management products connected with the Particle Physics Data Grid (PPDG) project. This study was conducted over the course of 4 months at 50% time by the author.

The intent of the study is to examine different products and/or projects connected with PPDG that are related to Replica Management for Distributed computing in a Grid-like environment. There are a variety of projects connected with PPDG, either through experiments or through computing science groups, that perform some aspects of data file replica management.

Some of these projects are explicitly Grid components (eg. even expressing their interfaces in WSDL), while others are only incidentally Grid aware and some are even completely independent of Grid services and/or tools.

We intended with this study to centrally collect information about each of the involved participants, initiate communication between projects, and to promote integration of products which have not interoperated before.

I would like to thank all the PPDG collaborators and non-PPDG developers who participated in the course of this study. The amount of material provided to me was vast and impressive. As always, any mistakes and/or misrepresentations are my own and the documentation available from the various projects should be taken as definitive. Finally, the rate of change of software (especially open source software) is often quite high in general. This is doubly true in the Grid middleware arena and in HENP Grid projects. Hence, this document should be considered a snapshot in time and a reference milestone for the project involved. I expect that much of this document will become dated with age.

### Participants

The list of participants in the study is shown below with appropriate URLs for information. The range of functionality of the various projects is quite large. Although the initial charge for this study was to investigate Replica Managers, few of the participating projects fall completely within the replica management domain.

These projects all have some responsibility for data location and management of multiple copies of data.

- EDG-WP2 (<http://edg-wp2.web.cern.ch/edg-wp2/>)
- Magda (<http://www.atlasgrid.bnl.gov/magda/info>)
- Phenix (<http://www.phenix.bnl.gov/replicator/fileCatalog.html>)
- RLS (<http://www.isi.edu/~annc/RLS.html>)
- SAM (<http://d0db.fnal.gov/sam/>)
- SRB(<http://www.npaci.edu/DICE/SRB/index.html>)
- SRM (<http://sdm.lbl.gov/srm>)
- STAR (<http://www.star.bnl.gov/>)
- JLab (<http://www.jlab.org/hpc/datagrid/>)

## **EDG-WP2 RLS**

WP2 is the "Data Management" Work Package of the European Data Grid (EDG). The goal of this work package is to specify, develop, integrate and test tools and middleware infrastructure to coherently manage and share petabyte-scale information volumes in high-throughput production quality grid environments. The work package will develop a general purpose information sharing solution with unprecedented automation, ease of use, scalability, uniformity, transparency and heterogeneity

It will allow to securely access massive amounts of data in a universal global name space, to move and replicate data at high speed from one geographical site to another, and to manage synchronization of remote replicas. Novel software for automated wide-area data caching and distribution will act according to dynamic usage patterns. Generic interfacing to heterogeneous mass storage management systems will enable seamless and efficient integration of distributed resources.

## **Globus-EDG RLS**

The Replica Location Service being maintained as part of the Globus project by Information Sciences Institute (ISI) was originally designed, implemented, and tested jointly by ISI, Chicago, and the EDG-WP2.

The replica location service (RLS) maintains and provides access to mapping information from logical names for data items to target names. These target names may represent physical locations of data items, or an entry in the RLS may map to another level of logical naming for the data item.

The RLS is intended to be one of a set of services for providing data replication management in grids. By itself, it does not guarantee consistency among replicated data or guarantee the uniqueness of filenames registered in the directory. The RLS is intended to be used by higher-level grid services that provide these functionalities.

## **Magda**

Magda is a distributed data manager prototype for grid-resident data. It makes use of the MySQL open source relational database, Perl, Java, and C++ to provide file cataloging, retrieval and replication services. For data movement, gridftp, bbftp and scp can be chosen depending on available protocols. Third party transfers are supported. Magda currently catalogs 330K files with total size of 26 TB. It was successfully used in the Data Challenge 1 production on the US grid test beds for the Atlas experiment. It has been used to replicate more than 4 TB of data between BNL and CERN mass stores. We will present the architecture of Magda, and how it is used as a file catalog and replication tool.

## **Phenix**

The Phenix experiment at RHIC has developed Argo, a distributed file catalog based upon PostgreSQL. Argo is intended to be a largely off-the-shelf database solution to the problem of managing data files and replicas. Argo makes use of Postgres Replicator and PostgreSQL triggers to enable peer-to-peer synchronization of file catalogs and to do other actions (such as DB scrubbing). Argo has a very simple C++ interface (FROG - File Replica On Grid) which provides translation between LFNs and PFNs suitable for direct I/O operations.

## **SAM**

Sam is a file based data management and access layer between the Storage Management System and the data processing layers. The goal of this SAM is to optimize the use of data storage and delivery resources, such as tape mounts, drive usage, and network bandwidth. In order to facilitate this goal, the primary objectives are:

- Clustering the data onto tertiary storage in a manner corresponding to access patterns,
- Caching frequently accessed data on disk or tape,
- Organizing data requests to minimize tape mounts, and
- Estimating the resources required for file requests before they are submitted and, with this information, making administrative decisions concerning data delivery priority.

In addition, it is desired to unload the burden of individual file tracking from the analysis physicists, and place it onto the data management system. This is an added bonus integrated into the SAM system.

## **SRB**

The SDSC Storage Resource Broker (SRB) is a client-server middleware that provides a uniform interface for connecting to heterogeneous data resources over a network and accessing replicated data sets. SRB, in conjunction with the Metadata Catalog (MCAT), provides a way to access data sets and resources based on their attributes rather than their names or physical locations.

## **SRM**

The purpose of this project is to address the problem of managing the access to large amounts of data distributed over the sites of the network. It is necessary to have Storage Resource Managers (SRMs) associated with each storage resource on the grid in order to achieve the coordinated and optimized usage of these resources. The term "storage resource" refers to any storage system that can be shared by multiple clients. The goal is to use shared resources on the grid to minimize data staging from tape systems, as well as to minimize the amount of data transferred over the network. The main advantages of developing SRMs as part of the grid architecture are:

- Support for local policy. Each storage resource can be managed independently of other storage resources. Thus, each site can have its own policy on which files to keep in its resource and for how long.
- Temporary locking. Files residing in one storage system can be temporarily locked before being transferred to another system that needs them. This provides the flexibility to read frequently accessed files from disk caches on the grid, rather than reading files repeatedly from the archival tape system.
- Advance reservations. SRMs are the components that manage the storage content dynamically. Therefore, they can be used to plan the storage system usage by making advanced reservations.

- Dynamic space management. It is essential to have SRMs in order to provide the dynamic management of replicas according to the locations they are needed the most (based on access patterns).
- Estimates for planning. SRMs are essential for planning the execution of a request. They can provide estimates on space availability and the time till a file will be accessed. These estimates are essential for planning, and for providing dynamic status information on the progress of multi-file requests.

## **STAR**

The STAR experiment has an older, flat-file-based file catalog with ~2.2 M distinct files that has been evidencing performance problems. A new, temporary STAR File Catalog has been developed which uses MySQL for metadata storage, SRM for data migration and cache management, and spiders for updating the MySQL database.

## **Jlab**

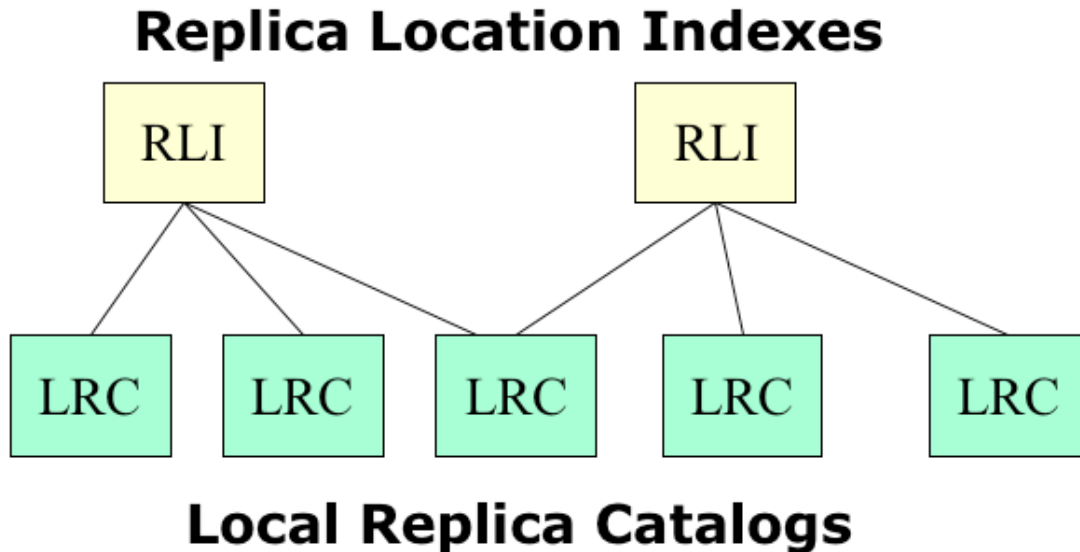
Jefferson Lab has a Replica Catalog Web Service which presents both a programmatic API and a Web Browser interface on top of a Web Service (implemented as a Java SOAP servlet). This Replica Catalog was not an explicit part of this study, but is mentioned for completeness. Details can be found at the Jefferson Lab web site.

## **Technical Comparison & Contrast**

### **EDG-WP2 and Globus RLSs**

Two products are routinely referred to as Replica Location Service (RLS). One, RLS from Globus-EDG WP2 (<http://www.isi.edu/~annc/RLS.html>) was designed, implemented, and tested jointly by ISI, Chicago, and WP2. This RLS is part of the Globus Toolkit and is maintained by ISI. The second RLS is based upon the same architecture and has been developed and is being maintained by EDG-WP2 (<http://edg-wp2.web.cern.ch/edg-wp2/>). I will make the distinction by referring to RLS (or the Globus RLS) and the EDG RLS.

Both RLSs are based on a peer-to-peer architecture consisting of Local Replica Catalogs (LRC) containing information about data file replicas within some specific domain, and Replica Location Indices (RLI) which contain Bloom Filter hashes allowing LRCs to be located that have replicas of for a particular LFN.



**FIGURE 1 – RLS peer-to-peer architecture**

LRCs are typically, but not necessarily, associated with a single storage resource on the Grid (eg. The advertised storage resource associated with a Tier 1 regional center.).

The RLIs are updated asynchronously, leading to a soft state nature typical of p2p systems. EDG expects that *all* RLIs will be updated with info from *all* LRCs and further draws a distinction between RLIs associated with a particular LRC (in particular a Tier 0/1/2 LRCs) which are updated immediately, and RLIs that are associated only (or primarily) with CEs. These second type of RLI will likely be updated less frequently.

The Globus RLS allows for this "all RLIs know about all LRCs" model. But their advertised plan is to allow hierarchic aggregation of RLI info (similar to the information aggregation in an LDAP hierarchy).

One aspect of the use of Bloom Filters for populating the RLIs bears noting as it is not obvious to those unfamiliar with the approach. The RLIs store a bit-field per LRC which is filled with Bloom Filter hashes of the LFNs (or GUIDs) contained within that LRC. This has two normal consequences. 1> Queries against an RLI can give false positive answers. This is dealt with by subsequently querying the LRC which has the complete LFN info. 2> Queries against an RLI can only be queries for a specific LFN (or GUID). No regular expression matching or range queries can be answered by the RLI.

As of the writing of this document, the API, communication protocol, and the payload (ie. the hash function used for the Bloom Filter) are different. This implies that any common interface or adaptation layer giving access to both flavors of RLS would require some non-trivial functionality. However, since the architectures are virtually identical, such an adaptation layer would be well localized (ie. would not require any refactorization of the functionality).

The EDG RLS has evolved more rapidly in the last year and has kept more closely aligned with the expressed needs of the LCG projects, notably POOL. There has been presented several functional extensions of the architectural model by the EDG WP2 beyond that of the Globus RLS

model. The explicit inclusion of GUIDs as the unique identifier of logical files rather than LFNs and the resultant ability to use multiple LFNs to refer to a single logical file was perhaps the first.

### **Experiment Solutions (Magda, Phenix, STAR)**

Each of the three experiments involved in the study have implemented their own versions of replica management systems. The two RHIC experiments (Phenix and STAR) to handle their real data, and ATLAS, to handle simulated data challenge data.

All three experiments have chosen to implement their systems using readily available open source relational database packages. Magda (ATLAS) and the STAR File Catalog use the MySQL database, while the Phenix File Catalog (Argo) uses PostgreSQL. Advantages of this choice are clear: A well-understood, high-quality, and widely used and supported software product which meets most, if not all of the requirements of the experiments.

As well, each of the experiments use a semi-automated mechanism for updating the database. Magda and STAR use spiders while Phenix uses cron jobs and triggers.

Of the three experiments only the STAR File Catalog also uses a recognizable middleware package in addition to the OSS RDB. STAR uses the SRM from LBNL for data movement and cache management.

All three experiments have significant amounts of data in their respective catalogs had have varying degrees of willingness to consider evolution of their systems based upon developments in the HENP community in particular and the Grid middleware community in general.

ATLAS has been using Magda for it's Data Challenges and has been generally satisfied with it. Magda, however, has always been intended to be a temporary stop-gap solution while waiting the development of the official LHC solution. Now that RLS from the EDG has reached a state of maturity which makes feasible testing and eventual adoption, Magda development is being frozen. It seems there are two possible scenarios for Magda: Either Magda will be replaced with an LHC-wide solution or the Magda interface will be layered on top of one or more RLS components.

STAR has explicitly stated that their current Replica Catalog solution is a temporary solution and one that they are willing to either evolve or even abandon when a suitable replacement is available from the HENP or Grid community which meets their requirements. The MySQL-based STAR Replica Catalog was developed to address both the scalability issues seen with the old, flat-file catalog, and to initiate a user shift away from simplistic Unix shell-based approaches of locating available data files.

It is clear that the STAR experience with LBNL's SRM has been quite positive and they would likely not abandon SRM without tremendous motivation. They have expressed interest in the Replica Registration Service work being done within PPDG as a complement to SRM. As well, the STAR experience of developing an in-house file catalog has gelled their own view of what functionality such a service should provide.

Phenix has adopted a strategy which (of the three experiments) seems most reliant upon external open source software products and most independent of future HENP/Grid solutions. At a recent PPDG meeting, the Phenix Computing Coordinator expressed his belief that the Phenix File Catalog met their requirements and that they had little interest in change for change's sake.

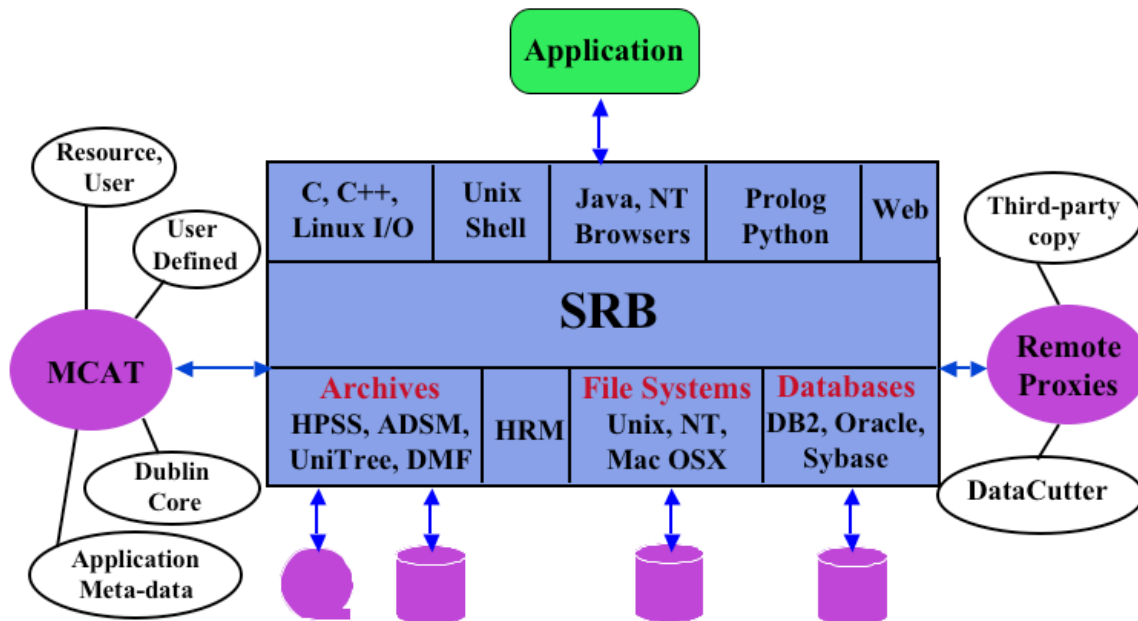
Indeed, some of the capabilities, and/or characteristics of the PostgreSQL database upon which Phenix made their initial choice of technology are not reproducible with other products.

Each of the three experiments expressed awareness of other replica management solutions extant in the HENP community, but each concluded that for their specific requirements and/or their particular stage in data taking and handling these other solutions were not viable. It is the author's opinion that none of the experiments are sufficiently different in final requirements to demand an experiment-specific replica management solution. Rather, the development of three disjoint solutions for three different experiments indicates the perception of a lack of consensus on replica management in Grid middleware projects.

**Standalone Systems (SAM, SRB)**

SAM and SRB are standalone systems conceived and implemented as full solutions for data handling, and in the case of SAM, job scheduling.

SRB makes clear distinction between management of data and of storage. It uses a peer-to-peer catalog federation to keep track of data, and has a storage repository abstraction layer to interface to multiple storage systems. The management of files has explicit write-lock and synchronization semantics to ensure data consistency and to prevent corruption of data and supports GSI 1 & 2.2 (GSI 3 support is expected in the near future), and supports Grid FTP for file movement.



**FIGURE 2 – SRB Architecture**

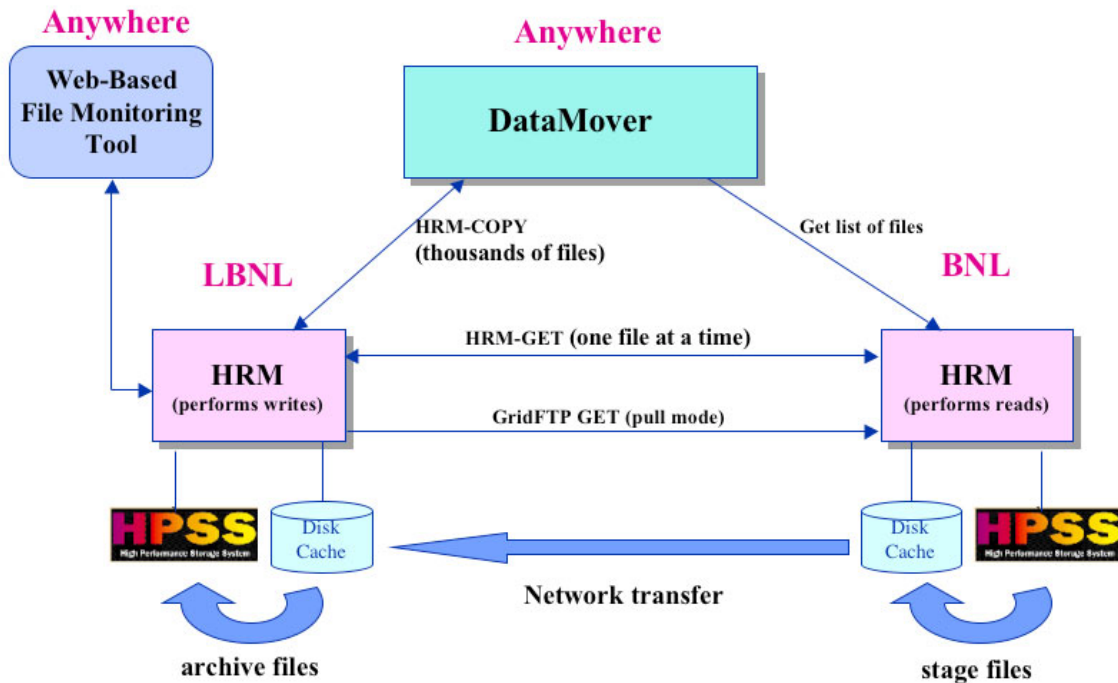
SRB also supports several interesting and unique capabilities such as semantic replicas (Replicas with respect to data content, but in a different data file format.), and slicing (Delivery of partial files equivalent to remote I/O.).

SRB also has it's own metadata catalog, MCAT, for storing and managing metadata about data files and file collections, and which provides a mechanism for accessing data files based upon queries against that metadata.

**SRM**

There are in fact several SRMs extant. SRM is explicitly a definition of functionality and an interface presenting that functionality. Berkeley Lab, Fermi Lab, and Jefferson Lab all have implementations of SRM in use and maintained. This has clearly been a very successful effort at generating consensus and encouraging healthy competition within established boundaries and with shared interface definitions.

SRM has carved out a particular functional niche in which other products have not specialized. SRM provides no real replica management per se, but it does provide cache space management. This cache management complements file migration and file replication both for single files and for large groups of files.



**FIGURE 3 - SRM Multi-file transfer between STAR HPSS systems.**

SRM from Berkeley Lab is in routine use by the STAR experiment for replication of files from BNL to LBNL for analysis on the LBNL PDSF cluster. It forms an integral part of the current STAR replica management solution as mentioned above.

**Technology choices**

The technology choices made by the 9 projects cover a wide spectrum. The technology choices important to understanding integration issues are:

- Interfaces & APIs  
The explicit interfaces and APIs defined by a service or product are the intended points with which external clients are intended to interact with the product.
- Communication Protocols  
Since Grid is by definition a distributed environment, the choice of communication protocol is an important consideration. Talking the same communication protocol is one

way in which two products can potentially interoperate. Use of a widely accepted standard such as SOAP increases the likelihood of interoperability.

- **Database Technologies**  
Most of the products utilize commercial and/or open-source databases for storage of metadata and internal bookkeeping. Being able to access the database and digest the schema of different products is one way to exchange information.

	<b>DataBases</b>	<b>Interfaces</b>	<b>Protocols</b>
<b>EDG</b>	Oracle 9iAS mySQL Tomcat	Java <sup>‡</sup> , C <sup>‡</sup> , C++ <sup>‡</sup> WSDL	SOAP-RPC
<b>JLab</b>	JDBC mySQL	Web Browser	SOAP
<b>Magda</b>	mySQL	Unix CLI Web Browser Perl, C++ <sup>‡</sup> , Java <sup>‡</sup> CGI	
<b>Phenix</b>	PostgreSQL PG Replicator	Web Browser C++ <sup>¶</sup>	[PostgreSQL]
<b>RLS</b>	PostgreSQL mySQL <sup>§</sup> Oracle <sup>†</sup>	Unix CLI C, Java WSDL <sup>†</sup>	TCP/IP
<b>SAM</b>		Unix CLI Python IDL	CORBA
<b>SRB</b>	mySQL BerkeleyDB	Unix CLI Web Browser C, C++, Java, Perl, Python WSDL <sup>†</sup>	
<b>SRM</b>		C++ <sup>‡</sup> IDL	CORBA
<b>STAR</b>	mySQL	Web Browser Perl, C++ <sup>¶</sup> XML	[SRM]

<sup>‡</sup> -- Partially or fully autogenerated.

<sup>§</sup> -- Deprecated.

<sup>†</sup> -- Not yet available.

<sup>¶</sup> -- Partial or incomplete.

**TABLE 1 – Technology choices by PPDG projects.**

## Conventional Interfaces

The lack of a coherent interface definition mechanism for the programmatic APIs (WSDL is still too nascent to be considered coherent at this point), makes the direct use of one system's API by another a wholly manual process. The use of strongly-typed languages such as C++ and Java in several of the APIs means explicitly tying the development cycle of a client system to the development cycle of that API. This is the typical problem faced when trying to integrate or interface two large, complex and distinct systems in any domain.

Clearly the ascendancy of WSDL, if adopted across the board can help tremendously. However, without common interfaces against which to program this is at most a tool applicable to the problem rather than a solution in itself.

The Web Browser interfaces available for JLab, Phenix, Magda, and STAR are entirely intended for human interaction. Though it seems theoretically possible to access another system through HTTP and parse the resultant web page, this is not an efficient nor maintainable approach.

Several of the projects provide a Unix Command Line Interface (CLI) that presents the systems functionality for use at the shell prompt. Other systems can invoke said Unix CLI through the normal `exec()`-like mechanisms (This goes for the scripting interfaces of Python and Perl as well.). This still requires manual programming to invoke the Unix CLI, but does not require recompiling & linking unless the footprint of the CLI changes. Even then the rebuild does not require a build against a changed external code base. However, this does imply a fully functional client system installed on all machines where the CLI invocation will occur. A requirement which may significantly increase system administrators' support burden.

## Protocol Interfaces

Again, the sheer variety of protocols foils blanket recommendations on strategies for interoperability. The Globus-RLS use of a proprietary wire protocol makes interoperability at this communication level quite difficult. The use of CORBA for two of the projects (SAM and SRM) opens a possible mode of communication. CORBA interfaces can be queried with the BOA reflection interface. My experience is that although possible, this is seldom done in practice. CORBA communication between two such products implies at the least exchange of IDL for the generation of the appropriate object stubs.

The most currently compatible communication protocol appears again to be OSGI-based. The use of SOAP communication in two products (EDG & JLab) would enable those two systems to exchange information payloads wrapped by the conventional SOAP protocol. However the meaning of the content of the payload (i.e. content of the SOAP body) still needs to be understood by the client system.

## Database Interfaces

Overwhelmingly, these eight projects have chosen to use one or several relational database backends as the method for storing and accessing information about data replicas and associated metadata. This is a natural, even inevitable, choice driven by the advantages of using an RDBMS.

All the databases in use in these projects support concurrency at some level. This concurrency is normally viewed as multiple users or jobs accessing the database through the same replica management system. However, it is possible to imagine a system where two different replica management systems access the same RDB. This would imply not only a shared schema and RDB technology, but also that each of the two systems respect the execution model of the other or that one of the two systems have supremacy (i.e. have read-write access to the RDB while the other has read-only access).

None of these projects have tried to initiate such a shared RDB interface. My expectation is that the likelihood of a peer interface is quite low. It would require a very close interaction between the projects to define a mutually acceptable schema and transaction model. A more likely possibility is that of one replica management system accessing the data of another through a read-

only interface written specifically to couple to that other system. At this point no projects are pursuing such a coupling.

## Interface Issues

From the list of participants in this study, it is clear that the topic of data replication and data management has attracted a significant number of players, both on the Computing Science side of the fence and on the experiment's. These various programs are motivated by a shared conviction of the importance of replica and data management, and by perceptions that other projects/solutions in the field are not easily adopted, or do not meet requirements, or that there exists an opportunity to apply expertise in this area to this important domain of Grid middleware.

The Grid is still in a formative stage of development where competition and new ideas are fruitful, even necessary. This competition permits exploration and development of new approaches and ideas while still driving real implementations to satisfy clients and/or demonstrate these new concepts.

On the other side of the fence, the clients of Grid middleware (e.g. the Experiments in our case) would like to test competing and complementary products and their associated ideas without pre-determining the eventual winners and without incurring prohibitive migration costs if a change in middleware is required.

In an ideal world, these two laudable goals could be satisfied by competition within established standards or adhering to common interfaces. Such standards or common interfaces allow clients to test Grid middleware and be confident that, if a migration is necessary, it can be accomplished with a minimum of pain and effort.

Of course integration, interfacing, and standards covers a wide range of possibilities. Interfacing can involve actual use of an API between two products as peers, the use of one as a component or part of another, data and/or information exchange using compatible or transformable schema, or use of common or self-describing communication protocols.

As of the writing of this document, there exists no widely accepted standard for replica and data management, though several efforts are underway to jump-start such standards. Notably efforts to establish such standards include:

- The Global Grid Forum OGSA Data Replication Services Working Group is ramping up with just this mandate.
- Several middleware providers are in discussions on a common Replica Registration Service (RRS) interface along the same lines as the SRM interface.
- Both the EDG-WP2 and Globus-EDG RLS teams are engaged in discussion to reconverge their products at the interface level.

The move towards SOAP-based communication and WSDL-described service interfaces increases the apparent likelihood of inter-product compatibility. There are three obvious advantages to using these Web Service technologies:

1. They are external standards which each of the products could easily adopt without expending significant internal manpower. (i.e. No internal manpower would be required to design and implement the actual standards themselves, just the payload.)
2. They are explicitly designed to allow loosely coupled, relatively autonomous services to communicate. (i.e. There would be no tight coupling between products.)
3. WSDL allows interface inspection which typically increases robustness to change, though at the cost of performance. (i.e. The different development cycles of the projects would not necessarily need to be synchronized as closely.)

As well, the individual projects involved in this study are in good communication through PPDG, GGF, LCG, and other channels.

	STAR	SRM	SRB	SAM	RLS	Phenix	Magda
EDG	Y			Y	Y		Y
Magda					Y		
Phenix							
RLS	Y	Y					
SAM		Y					
SRB		Y					
SRM	Y						

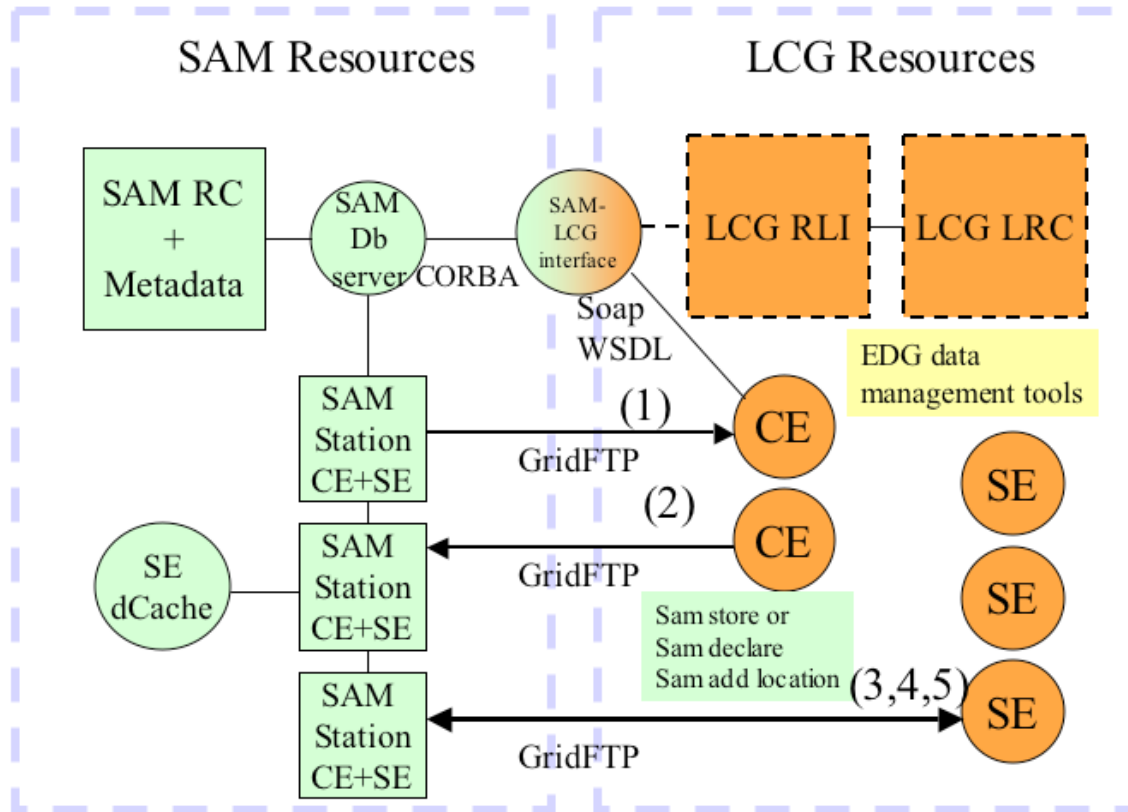
**TABLE 2: Inter-project collaborations underway or under investigation at the time of the writing of this report.**

Current efforts to integrate or interface projects include:

- SRM - SRM is in fact an interface standard with at least three different implementations at LBNL, FNAL, and JLab.
- RRS - There is an on-going discussion on establishing an RRS interface standard similar to the establishment of the SRM interface. A workshop at LBNL is planned for the middle of September 2003.
- RLS - As stated above, the two RLS implementations are based upon common architectural design principles. Though their implementations have diverged to the point of incompatibility, both teams seem committed to re-converging at the interface level. Discussions prompted by this PPDG work are on-going and it is hoped that a common interface to the LRCs and RLIs can be defined. Both projects expect this to happen even before a GGF-based WSDL standard is established.
- SAM & EDG - During the course of this study, the SAM and EDG-WP2 teams have begun discussion on how to exchange information between their systems. N.B. This is really an information exchange as opposed to an integration as the processing and data movement models in both domains are unchanged (see Figure 3). Rather, this is an effort to import knowledge about EDG resources into SAM and vice-versa.

There is a potential problem with this approach in that scheduling, cache management, and most types of through-put optimization implicitly require a level of global knowledge

and/or control over resources. Simple information exchange is probably not sufficient. Rather, resource locking and/or reservation will also be needed.



**FIGURE 3 - SAM-EDG peer-to-peer communication**

- SAM & SRM** - The SAM team is currently investigating the concept of using SRM as a file transfer protocol. This would have the advantage of adding space management to the SAM file transfer mechanism and insulate the system from the concrete mass storage system being accessed. Figure 3 shows a high-level illustration of the interaction model being discussed.
- SRB & SRM** - An integration study is underway by the SRB team which is expected to result in a concrete plan to interface SRB and SRM. Interestingly, preliminary thoughts by SRM developers indicate that a potential SRM-SRB integration might be accomplished by one of two inverted scenarios: SRM could incorporate SRB as a storage repository, or SRM could be incorporated as a storage repository within SRB. SRM is expecting to provide a Web Service interface soon. This could indeed form the most convenient interface for integration with SRM and other systems if SRB were to be treated as the storage repository interface.
- Magda** - Magda development within ATLAS is explicitly frozen except to address specific ATLAS requests. Discussions within ATLAS on how to evolve their data management system include the potential for replacing Magda's MySQL-based backend with an interface to the EDG-WP2 or Globus RLS. Magda's explicitly global view of the VO's replica space is at odds with the peer-to-peer architecture of the RLS. Consequently, it is not yet clear whether that interface would need to aggregate information from many

LRCs, or whether the Magda browsing interface and query interface would need to be abandoned or altered.

- STAR & SRM - STAR and SRM are treated as two separate projects in this report. However, the LBNL SRM is an integral part of the STAR system. STAR has expressed strong interest in and support for the emerging RRS standardization work.
- GGF OREP - The Global Grid Forum (GGF) has recently established a new OGSA Data Replication Services Working Group (OREP). The GGF OREP is intended to create, review and refine grid service specifications for data replication services. These specifications will conform to the Grid Services Specification being developed by the OGSF Grid Service Infrastructure Working Group.

## Conclusion & Way Forward

Discussions with all 9 teams were quite fruitful and produced a great deal of information. However, the sheer volume of information accrued as well as the disparities in level of detail of documentation, approaches to documentation, and coverage of functionality by the different projects make direct comparison or global statements quite difficult.

It had been hoped that this study might result in some more concrete interface implementation plans. That it did not is largely a reflection of the scale of the job given the large number of projects and perspectives involved. The author believes that a similar effort constrained to 2-3 projects and with more dedicated FTE-power (say 1 full FTE over ~4 months) could come up with concrete interface definitions for some aspects of the problem that could be implemented first by those problems and later adopted by others.

One shortcoming of the current effort was the inability due to time constraints to get hands on experience with the software being considered. The largest portion of effort was dedicated to reading and comparing documentation and having discussions with individual projects and groups of projects. Given the large number of projects and the relative complexity of the interfaces for many, it was too ambitious to read and understand all the user and developer interfaces.

It would be more efficient and productive to compare user interface functional categories between this large a number of projects rather than raw user interfaces. As an example, review and categorization of the Unix CLI for SAM reveals:

SAM Unix CLI Categories:

- Administration & System Configuration (28 commands)
- User Configuration (4 commands)
- Job Submission & Control (21 commands)
- File Handling & Replication (9 commands)
- Monitoring & Informational (16 commands)
- Internal (8 commands)
- Miscellaneous or Unknown (0 commands)

As a contrast, from the Globus RLS documentation: The Globus Replica Location Service (RLS) C API provides functions to view and update data in a RLS catalog. There are 2 types of RLS servers, Local Replica Catalog (LRC) servers, which maintain Logical to Physical File Name

mappings (LFN to PFN), and Replica Location Index (RLI) servers, which maintain LFN to LRC mappings. Note an RLS server can act as both an LRC and RLI server.

Functions are divided into the following groups:

- Activation
- Connection Management
- Operations on an LRC server
- Operations on an RLI server
- Miscellaneous Types and Functions
- Query Results
- Status Codes

Applications using this API should include `globus_rls_client.h`, and be linked with the library `globus_rls_client_FLAVOR`.

This documentation is clearly oriented towards an RLS-services view of the world rather than towards user actions. A systematic attempt by each of the projects to categorize their own interfaces using a common set of user-oriented functional categories would accomplish a huge step towards a more complete comparison of these projects.

There are a myriad of common concepts shared by most or all of the projects involved. All of the projects deal with replication of files or file collections rather than some other level of data collection. This is natural as the atomic unit of data transfer and management for Unixes is still at the file level. Almost all projects abstract file identity to either a LFN or a GUID-like concept.

Metadata is another common concept shared among many of the projects. However, there is an incredible diversity to what the term metadata refers. Types of metadata include:

- Physics MetaData
  - Physics metadata are used to select data by physics quantities at the event-by-event or at some coarser granularity. This is sometimes called Tag-DB data.
- File Replica Location MetaData
  - This is the heart of replica catalogs. Information about physical location, physicality, accessibility, etc of all physical replicas of a logical file.
- Provenance-like MetaData
  - This includes information about the production history, ancestors, and descendents of datasets. This is a very important type of metadata for reproducibility of analysis and is getting a great deal of attention in Grid circles these days.
- Physical File Attributes
  - File system-like information such as file size, ACLs, I/O technologies supported.

There are some areas that still have weak coverage by the studied projects. For example, with some notable exceptions, the support for physics and provenance metadata is much less than support for file related metadata.

There have been discussions in several forums (eg. individual LHC experiments, PPDG-CS11, LCG-GAG/HEPCAL, GEA workshop) about non-file-level datasets. IE. Datasets which do not correspond to a single file, but rather span multiple logical files or are logically a sub-logical file unit, perhaps within more than one single logical file. The extension of data handling from a

single file to a collection of files is already a good step, but is only supported explicitly by a subset of projects. The only project which explicitly supports sub-file data management is the SRB's remote file access concept.

There are several of the current integration efforts which I think deserve PPDG's attention and support in the near future. Either because they are particularly timely or can benefit from continued outside interest and pressure.

- GGF OREP - GGF's primary objective is to promote and support the development, deployment, and implementation of Grid technologies and applications via the creation and documentation of "best practices" - technical specifications, user experiences, and implementation guidelines. Given this objective, the GGF is an ideal forum for development of Replica Management standards. Successful (ie. wide) acceptance of standards presupposes one, or few forums for the establishment of those standards. GGF, in the author's opinion, is the appropriate forum for the formal establishment of Grid standards. It is the first and largest of the official forums for this kind of effort and PPDG should strongly support that role for GGF.
- RRS - The SRM effort is an excellent example of a grass-roots (ie. developer-level) establishment of an interface with multiple, distinct implementations. The success of the multi-implementation SRM is attempting to be reproduced by discussions on the Replica Registration Service. A group of PPDG developers will be meeting at Berkeley Laboratory to discuss the role and scope of the RRS. These kind of low-level, implementation-oriented discussions are a fine compliment to, and input for, the more formal process of the GGF. I believe that PPDG developers should continue these kind of discussions and bring the results to the table in the GGF.
- RLS - The Globus and WP2 RLSs are central players in the replica management domain. There appears to be a real willingness to address some of the factors which resulted in their divergence. As they are modeled upon the same architecture, their reconvergence could be relatively painless for both projects. IE. Changes of one or both RLSs to a common interface and/or communication protocol should require only relatively localized changes rather than a full refactorization of the overall system.

The Globus RLS/EDG RLS story also reflects an important consideration for PPDG in my opinion. There is an on-going concern about synchronization of, and coherency between, the US and European Grid efforts. There is a troubling tendency for similar and/or overlapping projects in the US and EU to diverge unless there is a conscious effort to sustain communication and ensure, through on-going testing, compatibility between these projects. This is a natural, though unfortunate consequence of the differing timeframes and customer bases of middleware developers in the EU and the US.

As a US-based Grid projects with significant, close ties to the LHC experiments (ALICE, ATLAS, CMS), PPDG is ideally situated to promote, establish, and sustain the inter-project communication necessary to prevent EU/US divergence. PPDG should insist that EU or US middleware projects or products justify any incompatibilities with their US or EU counterparts and/or related projects. The most concrete instrument for ensuring compatibility is real tests of compatible services. PPDG would make a tremendous contribution to EU-US coherency if PPDG projects, whenever feasible, would include both EU and US components in their tests.

During the course of this study, the author was struck by the openness of the various middleware developers to our plea for interoperability. Indeed, the mere existence of this study helped motivate several inter-project dialogs on the subject. It is clear that PPDG is in an ideal position to help continue these dialogs and to apply pressure as a significant client community towards convergence and standardization.

## REFERENCES

- Axis  
<http://ws.apache.org/axis/>
- EDG-WP2  
<http://edg-wp2.web.cern.ch/edg-wp2/>
- GGF OGSA Data Replication Services Working Group (OREP)  
<http://www.zib.de/ggf/copy/data/orep/>
- Global Grid Forum (GGF)  
<http://www.ggf.org/>
- gSOAP  
<http://gsoap2.sourceforge.net/>
- JLab Replica Catalog Web Service  
<http://www.jlab.org/hpc/datagrid/>
- Magda  
<http://www.atlasgrid.bnl.gov/magda/info>
- mySQL  
<http://www.mysql.com/>
- Oracle 9iAS  
<http://www.oracle.com/ip/deploy/ias/>
- Phenix File Catalog  
<http://www.phenix.bnl.gov/replicator/fileCatalog.html>
- PostgreSQL  
<http://www.postgresql.org/>
- Postgres Replicator  
<http://pgreplicator.sourceforge.net/>
- RLS  
<http://www.isi.edu/~annc/RLS.html>
- SAM  
<http://d0db.fnal.gov/sam/>
- SOAP  
<http://www.w3.org/TR/SOAP/>
- SRB  
<http://www.npaci.edu/DICE/SRB/index.html>
- SRM  
<http://sdm.lbl.gov/srm/>
- STAR  
<http://www.star.bnl.gov/>
- STAR File Catalog  
<http://www.star.bnl.gov/STAR/comp/sofi/FileCatalog/>
- Tomcat  
<http://jakarta.apache.org/tomcat/>

## GLOSSARY

- API == Application Programming Interface
- ATLAS == A Large Toriodal LHC ApparatuS
- BNL == Brookhaven National Laboratory
- BOA == Basic Object Adapter
- C == Procedural programming language.
- CMS == Compact Muon Spectrometer
- CORBA == Common Object Request Broker Architecture
- EDG == European Data Grid
- FTP == File Transfer Protocol
- GAG == Grid Application Group
- GGF == Global Grid Forum
- GSI == Grid Security Infrastructure
- GUID == Globally Unique IDs
- Globus == Grid toolkit & project.
- HENP == High Energy & Nuclear Physics
- HPSS == High Performance Storage System
- IDL == Interface Definition Language
- JLab == Jefferson Laboratory
- LBNL == Lawrence Berkeley National Laboratory
- LCG == LHC Computing Grid Project
- LDAP == Lightweight Directory Access Protocol
- LFN == Logical File Name
- LHC == Large Hadron Collider
- LRC == Local Replica Catalogs
- MCAT == Metadata CATalog
- OGSA == Open Grid Services Architecture
- OGSi == Open Grid Services Infrastructure
- PFN == Physical File Name
- PPDG == Particle Physics Data Grid
- RDB == Relational DataBase
- RDBMS == Relational DataBase Management System
- RLI == Replica Location Indices
- RLS == Replica Location Service
- SAM == Sequential data Access via Meta-data
- SOAP == Simple Object Access Protocol
- SRB == SDSC Storage Resource Broker
- VO == Virtual Organization
- WSDL == Web Services Definition Lanugage