# Web Services Data Grid Architecture

*Chip Watson, <>*

*Version 2: 6-Mar-2002*

## Summary

Web services are an appropriate mechanism for a wide range of grid services, particularly those focused upon information services (queries) and upon control. When coupled with efficient data transfer services, they provide an powerful mechanism for building a flexible, open, extensible data grid for science applications.

This document presents an architectural description of a data grid implemented as web services. It attempts to identify the set of web services needed, and what functionality each would encompass (system decomposition).

The HRM document (http://sdm.lbl.gov/srm/documents/joint.docs/hrm.v1.0.doc) developed jointly by LBNL, FNAL, and JLab was used as backgroud to define part of the necessary functionality (a more recent document is in development. To a great extent this document also follows the PPDG Data Grid Architecture document, doc00012.

## Architectural Elements

The standard PPDG diagram for the grid is shown in the following figure:

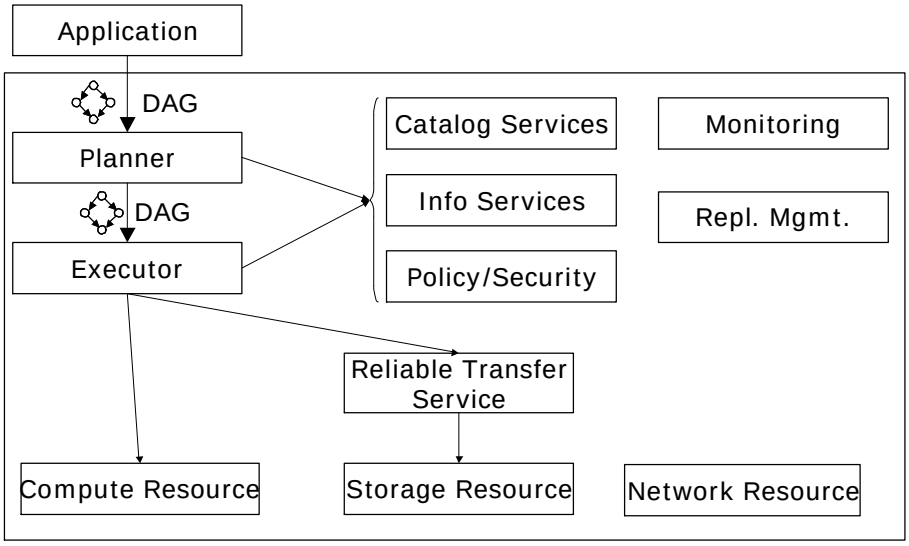



Figure 1: PPDG Architecture

This document will focus only upon the subset of this figure which is related to the data grid, and elaborate it further in terms of web services. In the following discussion, data sets will be described as “files”, but in principle this can be extended (in future versions and implementations) into any digital data, virtual entities, slices of files, etc.

In the following view, the file client could be the science application, or the planner, or the executor from the previous diagram (as each at some point is a client of file services). Additional interactions are elaborated, including those involving replica management functionality.

In the following discussion, the following terms from the HRM (Hierarchical Resource Manager) document are used:

GFN == global file name; logical file name

SRM == storage resource manager, a storage element

SURL == site URL; contains a reference to a data grid node (site) plus site file name

TURL == transfer URL; includes the protocol name and name of the file server machine

For context, the data grid is assumed to logically have a single meta data catalog (which may be replicated for fault tolerance and/or performance), and similarly a single replica catalog. It will have multiple data grid sites where data resides. All of the components below implicitly operate within a single virtual organization, or equivalently presume a particular namespace (the virtual organization) for all operations.

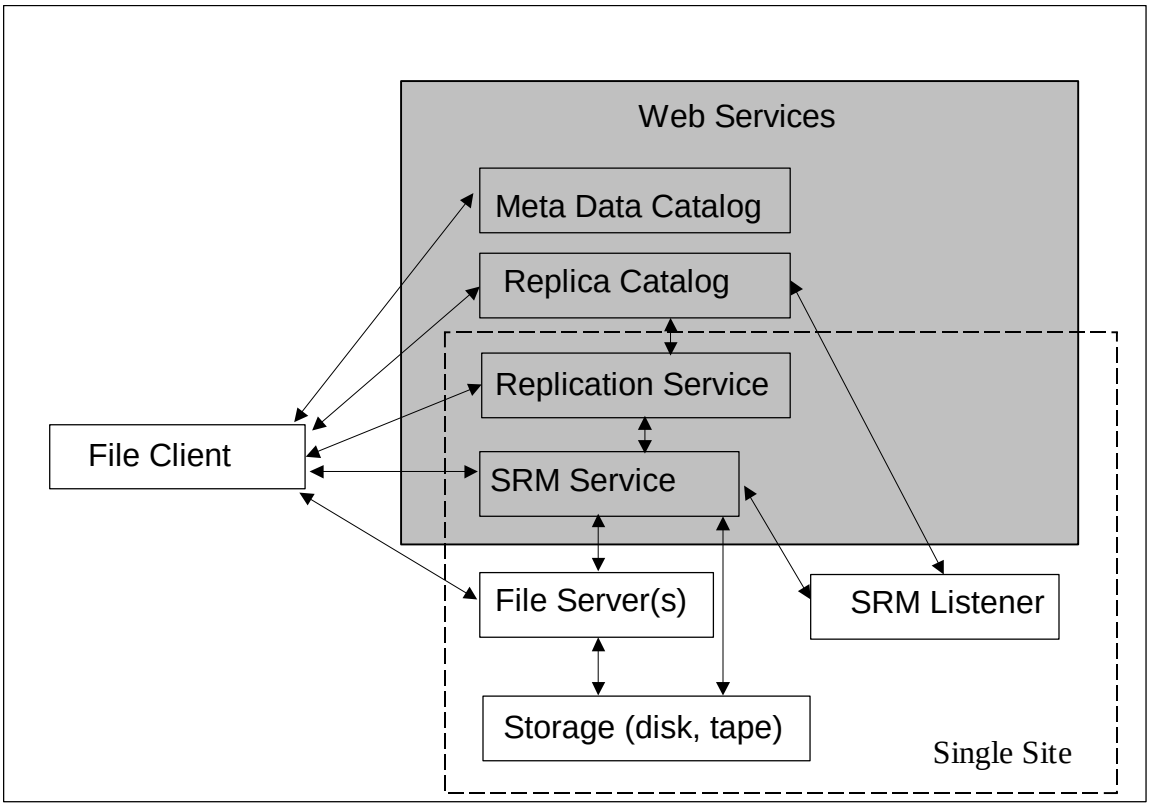



Figure 2: Data Grid Components

***Meta Data Catalog***

This entity manages the mapping between GFNs and data set meta data, most of which will probably be in the form of {tagname,value}, where value is a string or number, but this could be extended to any type which extends a virtual interface *comparable*. This catalog could also be application domain specific and then have a more general structure (such as a complex object instead of a set of tag,value pairs).

***Replica Catalog***

This manages the mapping between GFNs and SURLs (one to many), plus keeps a small amount of application independent meta-data necessary to manage the replicas, including file size, file checksum and checksum type, owner, etc, and some basic file status information (on disk / on tape). This meta-data is of the form {tag,value}. Web services include:

1. Directory Listing: optionally terse or verbose, optionally more than 1 level deep, optionally matching a pattern (regexp?)
2. Get replicas: GFN -> SURLs (get best replica?)
3. Create replica, specifying GFN, SURL, <meta data> (SURL must be unique)
4. Remove replica, specifying SURL
5. Mkdir; rmdir
6. Link GFN2->GFN1 (makes two names equivalent, usually in different directories); link directories or files; remove links
7. (Optional) Subscribe to changes in the replica catalog, or changes within a path scope.

Many of the above functions will operate on arrays of files for higher performance (to be determined).

***SRM Listener***

This component serves as the link between the replica-unaware SRM and the replica system. The SRM generates 2 possible types of events:

1. Advice request: proposed deletion of file X. Listener responds with advice as a number in the range of 0.0 (please don't) to 1.0 (OK). The listener could base this advice upon interaction with the replica catalog to discover if this is the last disk resident copy, for example.
2. State change notification: File X is added, or deleted, or cache state is changed. In this case the listener updates the replica catalog.

***Replication Service***

This manages, in a distributed way, the replication of files, and the updating of the replica catalog to be consistent with the local grid node. It is the place where replication policies are implemented and managed, and the service to which a request to have a file replicated is made.

Client web services include:

1. Copy a replica of GFN / SURL to site X.
2. Get status of replication operation.
3. Add / edit / remove a local replication policy (push, maybe pull)

To implement a replication policy, it may register as a listener with the HRM.

***SRM Service***

This manages all of the storage resources at a site. It is essentially the front end to the storage, which may have a distributed implementation at that site (many servers). Note that this document assumes capabilities beyond those in LBL's SRM document.

SRM Web Services include:

1. Get file status (cached, pinned, permanent, size, owner, etc. – details to be determined)
2. Request file status changes (e.g. stage a file, pin a file, make permanent)
3. Map from SURL to TURL for file get, including protocol negotiation
4. Make space allocations for put, including protocol negotiation to yield TURL

Extended functions (beyond LBL's spec) include:

1. Directory listings, search (like replica catalog)

The following extended function could instead be presented as a separate Reliable File Transfer service:

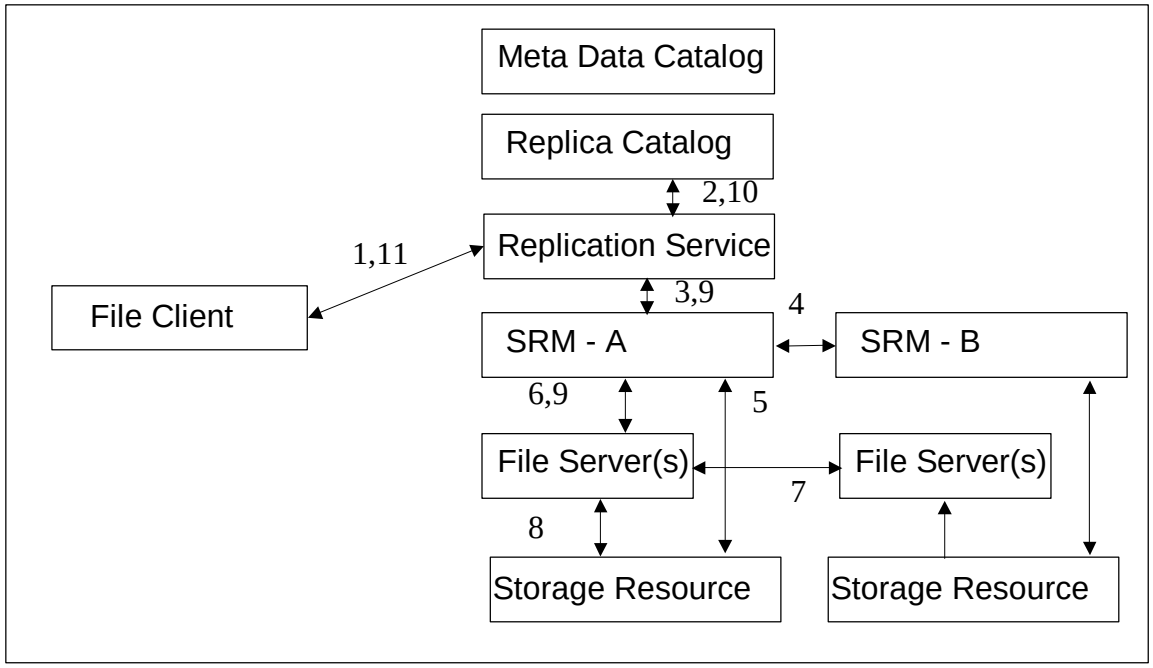
2. Reliable (as much as possible) third party file transfers to/from another Data Grid Site (reliable), or to/from a site with a supported protocol (e.g. ftp site)

***File Server***

This server supports data get/put, and is not usually a web service. It can also serve as the client or server in a third party file transfer. Each instance generally supports a single protocol, such as ftp or gridftp. A standard web server can be used as a file server supporting the HTTP protocol for file get, and perhaps HTTPS for file put.

***Use Case 1: Make Replica***

The following shows the connections involved when an application requests that a replica of a file be placed at a specified site (presumably where that application or a future job will use it).




Here are the steps involved in this operation:

1. File client requests of replication service at site X for GFN to be moved to site A.

2. Replication service looks up GFN in replica catalog, and finds file is at site B (SURL). (Optional: Pending new instance is noted in replica catalog).
3. Replica service requests SRM (or alternatively a file transfer service if multi-site operations are not defined as within the SRM scope – tbd) at site A to get the file from B (transfer request)
4. Site A asks site B for B's TURL for this SURL.
5. Site A allocates space (may force other files to be deleted).
6. Site A invokes appropriate local file client/server to fetch from B's TURL.
7. File is transferred.
8. File is written to disk.
9. When file transfer completes, the local storage resource is now informed file is under his control (can be written to tertiary, deleted, etc. as appropriate in the future)
10. Replication service is informed / discovers replication is done.
11. Catalog is updated to indicate availability of new replica.
12. Client discovers replication is done (can use file).

This use case could be further elaborated to show the interaction (indirect) between the storage resource (as it proposes to delete files to make room for the new replica) and the replication service and catalog. Also, site B may need to move the file from tape to disk, and then would tell A to check back later, providing a time estimate.