



Distributed File Management

Data- and Replica-management in AstroGrid-D¹

Deliverable	D3.2
Authors	Working Group Distributed File Management
Editors	Mikael Höggqvist
Date	November 14, 2007
Document Version	1.0.0
Current Version	1.0.0
Previous Versions	

A: Status of this Document

Deliverable 2 of working group 3.

B: Reference to project plan

Second deliverable of working group *Distributed File Management*.

C: Abstract

There are three main aspects of data management in AstroGrid-D. Staging of input and output data to the resources involved in job execution, replica registration and lookup and annotation of datasets and files. The focus of this deliverable is to describe the installation and how to use the

¹This work is part of the AstroGrid-D project and D-Grid. The project is funded by the German Federal Ministry of Education and Research (BMBF).

necessary software packages for data-staging and replica management. In addition, a core schema for describing files is presented.

D: Change History

Version	Date	Name	Brief summary
1.0.0	14.11.2007	Mikael Höggqvist	Public release with comments incorporated.
0.9.0	27.07.2007	Mikael Höggqvist	Second draft. Added scenario section and the file vocabulary.
0.0.1	27.04.2007	Mikael Höggqvist	First draft.

E:

Contents

Abstract	1
Change History	3
1 Introduction	5
2 Data Staging	5
3 Replica Management	6
4 Usage Scenarios	6
4.1 Input staging	7
4.2 Output staging	7
4.3 Input and output staging	7
4.4 File removal	8
4.5 Staging of directories	8
4.6 Input and output staging with intermediary storage	8
4.7 Replica Registration and Lookup	9
4.8 Replica lookup and output registration combined with staging	9
5 File-related metadata	10
6 Conclusion	11
References	12

1 Introduction

Staging of input and output data and replica management are the main tasks for the data management services. Globus Toolkit 4 (GT4), used as resource middleware in AstroGrid-D, includes two software packages fulfilling our basic requirements; GridFTP for data transfer and the Replica Location Service for replica management. This deliverable describes how these tools are used for data staging and replica management in addition to how they interact with GridWay, the grid job meta-scheduler used in AstroGrid-D. In addition, we define a core schema in RDF which is recommended to be used when annotating files and datasets. RDF is the data model used by Stellaris, the AstroGrid-D metadata management service.

The terms used in this document related to job submission and data management include

Frontend A Globus Toolkit 4 enabled host, able to receive and execute job requests via WS-GRAM.

Job Submission Host The host from where the job submission was initiated, for example the user's laptop.

File Storage Stores files and enable transfer of files both from and to the storage. All files stored in a *File Storage* are assumed to be accessible via a URL. Transfer mechanisms are typically FTP (via GridFTP) or HTTP.

Job Management Provides meta-scheduling and brokering of jobs submitted by the Job Submission Host. In AstroGrid-D this is implemented by the GridWay and GridGateWay software. GridGateWay makes it possible to submit normal Globus jobs to the GridWay broker and scheduling mechanisms.

Replica Management Maintains an index with mappings between Logical File Names (LFNs), also known as Object IDentifiers (OIDs)², and list of URLs representing the locations of a replica.

2 Data Staging

The constraints for data staging is that 1) a job must have access to its input data at the starting time of and during job execution, 2) the job output data is accessible to a user after the job execution has finished. We focus on two issues, how the data is transferred to and from the *Frontend* and the location of the data before and after job execution. For the first issue, we assume that the data location is represented with a URL. Since a URL also defines the access protocol, for example GSI-FTP or HTTP, we also know how to transfer the data. Data transfer via URLs is supported by the Globus tool `globus-url-copy`.

Second, the location of input data is either at the *Job Submission Host* or a *File Storage*. We assume that data can always be transferred to and from a *File Storage*. However, the data can be on a *Job Submission Host* without *File Storage* capabilities or the possibility to allow incoming connections from remote hosts. For example a laptop without a full Globus installation. In this case the *Job Submission Host* will also not be able to receive the output data. In order to ensure

²This was used in Deliverable 3.1, in this deliverable LFN is used.

that the data is available at job submission time and that it can be staged out after job finish, we introduce an intermediary *File Storage*. When submitting jobs directly from a host with a globus installation including a *File Storage* such as a grid-ftp service, this should not be necessary.

Job requests in AstroGrid-D are passed via the *Job Management* consisting of the GridWay scheduler/broker and a frontend called GridGateWay (GGW) based on GT4. Integration with GGW when using an intermediary *File Storage* is straight-forward since it supports the staging directives used in a normal Globus job description file. However, we must consider the different ways GGW uses for file staging.

GGW support two different modes for staging; two-hop mode and one-hop mode [3]. In the two-hop mode, all input and output files are staged via the GGW host to the *Frontend*. In the one-hop mode, the data is staged from the *Frontend* according to the directives in the RSL job description without the GGW host as intermediary. The usecases from AstroGrid-D has a large variation in both input and output data and would quickly saturate a GridGateWay instance under the estimated workload [1]. Therefore, we only recommend to use the one-hop mode.

3 Replica Management

The main task of the replica management system is to provide an indirection layer for the location of one or more files. The indirection layer is composed of an information system, called Replica Location Service (RLS), that performs a lookup operation from a Logical File Name (LFN) to one or more URLs representing file locations. RLS also support reverse lookups from a URL to a LFN. Details on how to use the RLS is presented in section 4.

Although LFNs can be constructed using arbitrary strings, we recommend that the use case developers uses a namespace with a URI-prefix. For example, LFNs within AstroGrid-D could be prefix with `http://astrogrid.net/files/`. This is useful for two reasons. First, as covered in section 5 it may be useful to add additional metadata using Stellaris and RDF. By using a valid URI as identifier for a LFN, it is easier to integrate with RDF. Second, in the HTTP protocol it is possible to redirect requests to other locations. This could be used in an extension of the system to provide transparent access to replicated files and datasets. However, this extension is not implemented since the current tools provided by GT4 does not support redirection between URIs with different URL schemes.

4 Usage Scenarios

Since applications has different needs we are not advocating a single solution. Instead, we have different services specialized in different tasks. For example the RLS for replica location and Stellaris for metadata annotations. However, to simplify the steps the grid application developer has to go through, WG-3 is providing a set of test-cases. These test-cases provide example code on how to wrap the job submission to support different staging scenarios in combination with and without GridWay. Additionally, they identify scenarios which are currently not working together with GridWay or Globus.

The full code of the test-cases can be found in the AstroGrid-D SVN repository³. Note that to support the one-hop solution (Section 2) transparently, the WS-GRAM service at the GGW host must be patched as described in the GGW documentation [3].

4.1 Input staging

This scenario does input staging when it is assumed that the *Frontend* can access the data directly using a protocol support by the Globus-tools (e.g. GridFTP, HTTP, FTP, etc.). To perform input-staging of data, the following should be added to an RSL job-description. With this syntax, data is transferred from the `sourceUrl` to the `destinationUrl`. The variable `${GLOBUS_USER_HOME}` is translated to the actual user home directory on the *Frontend* assigned to execute the job.

```
<fileStageIn>
  <transfer>
    <sourceUrl>gsiftp://<frontend hostname>/test_file</sourceUrl>
    <destinationUrl>file:///${GLOBUS_USER_HOME}/test_file</destinationUrl>
  </transfer>
</fileStageIn>
```

4.2 Output staging

Output staging is described in a similar way compared to input staging. The main difference is the direction of the transfer. The `sourceUrl` is now local to the *Frontend* and the `destinationUrl` is a remote location.

```
<fileStageOut>
  <transfer>
    <sourceUrl>file:///${GLOBUS_USER_HOME}/test_file</sourceUrl>
    <destinationUrl>gsiftp://<frontend hostname>/test_file</destinationUrl>
  </transfer>
</fileStageOut>
```

4.3 Input and output staging

This scenario combines input and output staging in a single request.

```
<fileStageIn>
  <transfer>
    <sourceUrl>gsiftp://<frontend hostname>/input.dat</sourceUrl>
    <destinationUrl>file:///${GLOBUS_USER_HOME}/input.dat</destinationUrl>
  </transfer>
</fileStageIn>
<fileStageOut>
  <transfer>
    <sourceUrl>file:///${GLOBUS_USER_HOME}/results.dat</sourceUrl>
    <destinationUrl>gsiftp://<frontend hostname>/results.dat</destinationUrl>
  </transfer>
</fileStageOut>
```

³[svn://svn.gac-grid.org/software/stagetests/trunk](http://svn.gac-grid.org/software/stagetests/trunk)

4.4 File removal

After the job has finished it is recommended to remove any files stored on the used compute resource. This is done by using the `fileCleanUp`-element. The following example removes a file named `test.file` located in the users home directory.

```
<fileCleanUp>
  <deletion>
    <file>file:///${GLOBUS_USER_HOME}/my_echo</file>
  </deletion>
</fileCleanUp>
```

4.5 Staging of directories

Directory staging works exactly like normal input/output staging except that a directory is given in the `sourceUrl` or `destinationUrl`. However, this is not working with `GridGateWay` at the time of this writing.

4.6 Input and output staging with intermediary storage

As described in section 2, an intermediary storage is necessary when the client does not have a data hosting server available (e.g. `GridFTP` or an `HTTP` server). In this case, the input data must be transferred to a *File Storage* which can serve the data to the *Frontend* before submitting the job. Similarly, the output data must be staged out to the *File Storage* after the job has finished. There it can optionally be retrieved back to the client or be made available at the *File Storage* for sub-sequent jobs. The RSL syntax is the same as for the basic input/output staging, with the exception that the `sourceUrl`-element for input staging and `destinationUrl`-element for output staging should contain a URL on the *File Storage*. Figure 1 shows a time-sequence diagram of the described scenario.

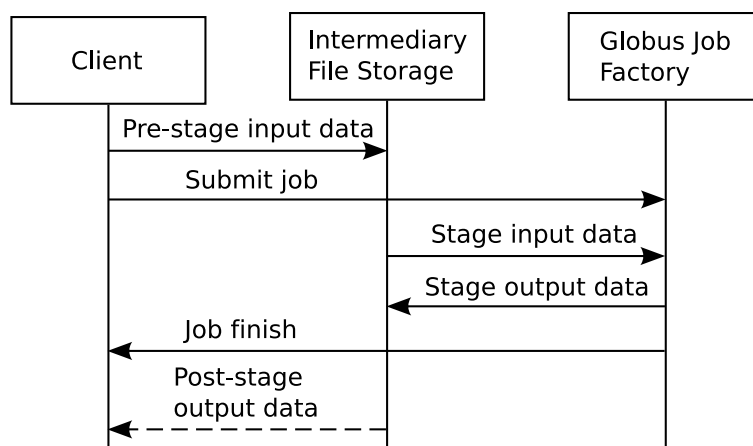


Figure 1: Input and output staging with an intermediary file storage.

4.7 Replica Registration and Lookup

Replicas are managed with the Replica Location Service (RLS). The RLS is basically a two-way lookup-table, meaning that it is indexing both the key and value. Therefore, it is possible to make lookups of all values associated with a key as well as all keys associated with a value. Note that the RLS is able to store any key/value combinations, why it is the users responsibility to ensure that the registered replicas are accessible.

The RLS service is used through the Globus-rls-cli program. The following listings contain the basic commands for creating a key-value mapping, adding a replica location to an existing key and how to query a key or LFN to find a set of replica URLs. Further documentation can be found in the Globus documentation ⁴ or the NorduGrid cheat-sheet ⁵.

```
globus-rls-cli create /dynamo/results \
gsiftp://astrodata01.aip.de/dynamo/output.dat rls://hydra.ari.uni-heidelberg.de

globus-rls-cli add /dynamo/results \
gsiftp://buran.aei.mpg.de/dynamo/output.dat rls://hydra.ari.uni-heidelberg.de

globus-rls-cli query lrc lfn /dynamo/results rls://hydra.ari.uni-heidelberg.de
```

4.8 Replica lookup and output registration combined with staging

In this scenario, the input file has been registered with the RLS and the output file is registered with the RLS after the job finish. Before the job is submitted to a Globus *Frontend*, the LFN must be resolved and the result inserted in the RSL job description. Similarly, to let RLS know about the newly created file, it must be registered after the job has finished. In this scenario (figure 2) the job description is defined to stage the file back to the Job Submission Host when the job execution is done. An immediate issue with replica lookup is that Globus does not support this transparently. As a result, the user must insert the file locations derived from RLS in the job description. In order to work around this problem, further developments are envisioned to be described in Deliverable 3.4.

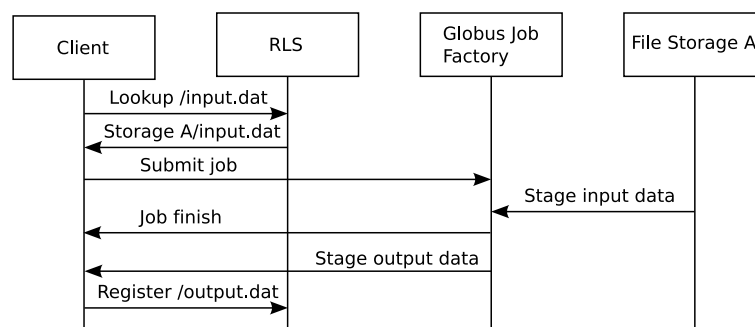


Figure 2: Scenario with RLS lookup of input file and registration of the output file.

⁴<http://www.globus.org/toolkit/docs/4.0/data/rls/rn01re02.html>

⁵<http://www.nordugrid.org/documents/rls-cheatsheet.html>

5 File-related metadata

Files and dataset distributed within a grid-environment is preferably registered and located using the RLS. In addition to replica location, RLS also supports the addition of arbitrary (attribute, value)-based metadata to LFNs. While this works well for simple metadata structure and queries, AstroGrid-D provides the Stellaris metadata management system [4]. In this section, a core set of terms is described which are recommended for description of files and datasets stored within AstroGrid-D. It is also recommended that the use case owner add terms describing the datasets produced by their use case with higher granularity. For example, parameters used to produce the data or a "link" to the job description.

Vocabulary

The vocabulary itself is a combination of different vocabularies. For example, general terms for describing the creator and creation timestamp are taken from the Dublin Core Metadata[2]. The listing below shows an example of how an RDF instance of the schema could look like followed by an explanation of the terms. The different namespaces are defined within the listing.

```
@prefix agd:<http://astrogrid.net/schema/#> .
@prefix dc:<http://purl.org/dc/elements/1.1/> .
@prefix xsd:<http://www.w3.org/2001/XMLSchema#> .

<http://astrogrid.net/files/geo600/test/output.dat> rdf:type agd:File ;
dc:creator <http://astrogrid.net/people/robert> ;
dc:created "2007-07-24T14:23:33+02:00"^^xsd:dateTime ;
agd:tag "test" ;
agd:tag "important" ;
agd:URL <gsiftp://buran.aei.mpg.de/geo600/job1/output.dat> ;
agd:URL <gsiftp://mardschana.zib.de/geo600/job1/output.dat> ;
agd:belongsToSet <http://astrogrid.net/files/geo600/test/> .

<http://astrogrid.net/files/geo600/test/> rdf:type agd:FileSet ;
agd:hasFile <http://astrogrid.net/files/geo600/test/output.dat> ;
agd:hasChildSet <http://astrogrid.net/files/geo600/test/tmp/> .

<http://astrogrid.net/files/geo600/test/tmp/> rdf:type agd:FileSet ;
agd:hasParentSet <http://astrogrid.net/files/geo600/test/> .
```

First, we have a subject that is of type `agd:File`. The subjects that has this type should either be an LFN or a URL pointing to a file. From the Dublin Core vocabulary we use two different terms: `dc:creator` and `dc:created`. These are used to identify the user who created the file and when it was created. The objects related to these terms should be an AstroGrid-D user, here represented with a URI, and an XML Schema [5] `DateTime` type.

The term `agd:tag` is used to allow users to "tag" a file with one or more arbitrary strings. Tagging can be seen as a more light-weight way of classification. This should be seen as a complement to the more rigorous description framework that RDF/RDFS provides. The term, `agd:URL`, refers to a URL from where the file can be retrieved. One or more URLs can be used to identify replicas.

Part of the vocabulary is also a way to describe sets and hierarchies between sets. For that there is class `agd:FileSet`. `agd:belongsToSet` indicates that a file is part of a certain set. Hierarchy is described with the two terms `agd:hasChildSet` and `agd:hasParentSet`.

It is also useful to add statements using terms from the general AstroGrid-D vocabulary currently being defined. This will include different classes for Projects, Jobs, Users and resources. An example could be that a certain file set is created from a job execution. This could be indicated as in the listing below.

```
@prefix agd:<http://astrogrid.net/schema/#> .  
  
<http://astrogrid.net/files/geo600/test/output.dat> agd:belongsToJob <http://  
  astrogrid.net/jobs/geo600/job-123abc> .
```

6 Conclusion

This document describes different usage scenarios for GridFTP, RLS and the Globus Job submission in combination with GridGateWay. These scenarios should be used as a guide by the use case implementer on how to use the file management tools available in AstroGrid-D. In addition, we have presented a core RDF vocabulary that is recommended to use when describing files and datasets. The related metadata is stored and queried using the Stellaris metadata management service.

F: References / Bibliography**References**

- [1] AstroGrid-D Use Cases. <http://www.gac-grid.de/project-documents/UseCases.html>, 2007.
- [2] DCMI Usage Board. DCMI Metadata Terms. <http://dublincore.org/documents/dcmi-terms/>, December 2006.
- [3] GridGateWay documentation on staging. http://www.gridway.org/relatedcomponents/files/1.0.3/doc/g4u_installconfiguide/c255.htm#FILESTAGING, 2007.
- [4] M. Höggvist. Stellaris: The AstroGrid-D Information Service. Technical Report D2.2, AstroGrid-D project, 2007.
- [5] XML Schema. <http://www.w3.org/XML/Schema>, 2007.