GWD-I Martin Westhead, EPCC, University of Edinburgh
Category: INFORMATIONAL
GGF Data Access and Integration Working Group

19TH Sept 2003

Data Access and Integration Services – File Access

Status of This Memo

This memo provides information to the Grid community regarding the implementation work done to inform the discussion of file access within the Data Access and Integration Services Working Group. This is an informational draft. It does not define any standards or technical recommendations. Distribution is unlimited.

Abstract

This describes implementation work to allow access to files from within OGSA-DAI, a partial reference implementation of the DAIS recommendation. It goes on to indicate areas where standards may be needed and discusses the relevance to the existing work of DAIS.

Contents

1. Introduction

This document reports on work carried out in the University of Edinburgh on introducing file access to OGSA-DAI.

OGSA-DAI is a partial reference implementation of the DAIS recommendation. It provides an OGSI interface to relational and XML databases as well as supporting a number of different delivery and transformation mechanisms. It is currently based around the GGF7 version of the DAIS drafts.

The reason for incorporating access to files, file systems and file contents into the OGSA-DAI framework was to establish how effectively file access could be incorporated into the DAIS framework. The aim of this short prototyping project was to implement access to files, file systems and file contents. The implementation raised a number of issues and suggested areas where standards definitions might be useful.

This document begins by describing what was implemented

2. OGSA-DAI File Access

The broad aim of the work was to provide

- Access to files – By access to files we mean ftp get and put i.e. the ability to pull down a whole file identified by a path name or URL of some sort, and likewise the ability to push a file onto a specified location on a file server.

- Access to file systems –being able to query and modify the directory structure, including information about file ownership, access permissions etc.

- Access to file contents – for files of an appropriate type, such as XML or files with an appropriate description (e.g. DFDL, BinX, BFD, ESML) to access pieces of the file though the use of a query string (e.g. XPath).

Futher the design aimed to provide a common access framework so that:

- Files are accessed using the same paradigm and mechanisms as databases within OGSA-DAI
- Filesystems, files and file contents are accessed using a common query language.

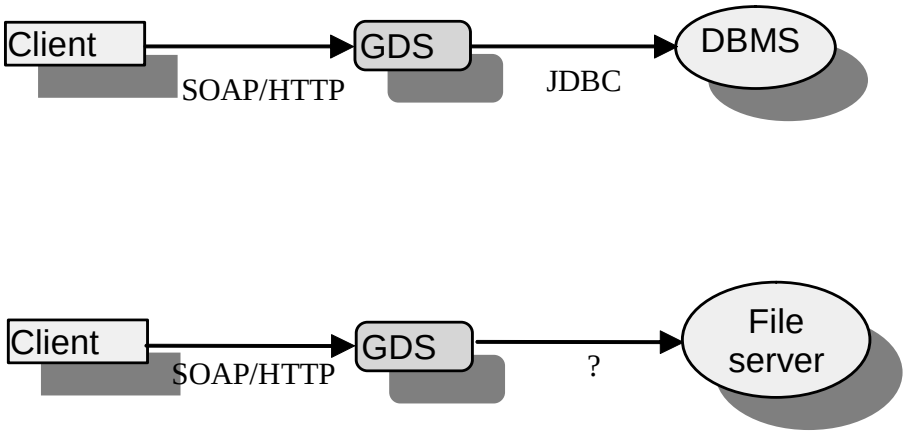

Figure 1 Showing the protocols used by Client, GDS and servers for interacting.

As it stands OGSA-DAI provides a series of components to allow OGSI-compatible access to Databases. That is it provides a component (called the Grid Data Service GDS) that attaches to a database using some sort of database connectivity library (such as JDBC) and acts as a proxy between the grid client and the database. The client is able to carry out actions on the database by submitting queries to it. The GDS provides additional functionality enabling the configuration of translations and delivery mechanisms. The interactions between Client, GDS and DBMS are depicted in Figure 1.

The second diagram in Figure 1 shows the same interactions depicted for file access and really highlights where most of the issues in this project arose from. DAIS and OGSA-DAI are technologies about using an OGSI wrapper to expose the functionality of a database server in an appropriate way. And in the relational case there is a well defined standard in the form of JDBC/ODBC to provide that connection. In the case of files it is less clear what we should use. It would be possible to have a grid service serve local files directly, or you could stay closer to the wrapper model and access files through an existing file server such as FTP of HTTP. In order to take advantage of the Grid security we decided to build our prototype accessing the Globus GSI-enhanced wu-ftp server. However we believe that the issues that are exposed are more general in nature.

The implementation itself added functionality to the ftp server so that it could access pieces of files specified using XPath. BinX [1] was used as a technology for describing file structure and the BinX library was used to implement the XPath access. The implementation also added a new activity to the OGSA-DAI implementation and extensions to the XML input documents in order to access the modified ftp server. The implementation is still in progress at time of writing.

The next section looks at some of the issues that were encountered in the design, and describes the solutions chosen. It is worth noting that all of the issues that we encountered were specific to the problems of accessing files. The access model defined by DAIS was entirely appropriate for this purpose and needed no adjustment.

3. Design issues encountered

The design encountered the following issues:

- Description of file/result contents - DFDL, MIME
- Query language - URL, XPath, FTP CLI…
- Associating description with data
- XML representation of directory listing
- Working directory definition

In this section we describe the problem faced and present the solution that was chosen for this proof of concept implementation.

3.1 Description of file/result contents

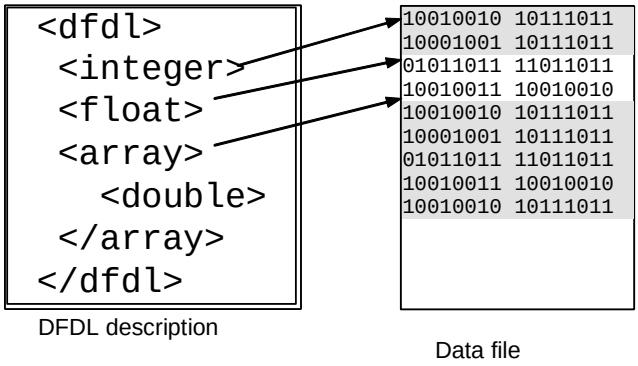

Figure 2 Showing the basic mechanism for a DFDL description of a binary file

A key feature of what we were trying to do was to access pieces of files. DFDL [2] is a GGF WG that is defining an XML language for describing and labelling the structure of data files. Once this structure has been defined for data file an XML representation of that data is implied. This XML representation can then be queried using XPath to return an individual parameter a tree of values or even, say, an array slice. This query can be executed without, in principle, realising the XML representation of the data. The basic mechanism is illustrated in Figure 2.

Unfortunately DFDL is still in its early stages of being defined, so the implementation was built on a technology called BinX that predates DFDL but operates on the same basic mechanism.

A related issue that we encountered was defining the return type. If you extract a portion of a binary document, using something like XPath and DFDL, the result may be a binary object of varying size. We need a way to describe its contents if we are to be able to use it. The natural way to do this is to use BinX again. A BinX description can provide the ability to define the structure of the return type. This leaves us with the problem of returning the description and the data in a sensible way. Our solution was to use MIME encapsulation to combine the two logical files into one. This is illustrated in Figure 3.

```
Content-Type: multipart/related; boundary=--xxxxxxxxxx;
--xxxxxxxxxx
Content-Type: text/xml
Content-ID: Contents

<?xml version="1.0" ?>
<dfdl>
  <!-- data description in here -->
</dfdl>
--xxxxxxxxxx
Content-Type: application/binary
Content-Transfer-Encoding: Little-Endian
Content-ID: Pixels
Content-Length: 524288
....binary data here...
--xxxxxxxxxx
```

Figure 3 Illustrating the use of MIME encapsulation to include the DFDL description with binary data in the same physical file.

3.2 Query language and data-description association

The DAIS specification that we worked with on this implementation had a very minimal set of operations on its porttype. The rich functionality of the database was accessed through the use of a powerful query language (SQL). By analogy we chose to use a query language based on FTP command language.

In the context of an OGSI standard for accessing files it is probably the case that you would take the same view as the more recent DAIS specifications that have had much richer port type definitions with multiple, specific operations. Nevertheless there is a relevant issue even in this context to do with specifying access to file contents.

Since XPath describes a hierarchical structure, it would be convenient to consider this hierarchy as a simple extension of the file hierarchy. This is indeed what we did by simply appending the XPath expression to the filename. (This is illustrated in Figure 4).

```
get /some/path/myfile.xml/xpath/expr
get /some/path/myfile.bnx/xpath/expr
get /some/path/myfile.csv/xpath/expr
get /some/path/myfile.bin/xpath/expr
```

Figure 4 - Showing the XPath expression appended to files of different types

An related issue is that of associating file types with the correct descriptions. Figure 4 illustrates some of the different cases that we handle.

In the first case an XPath expression is applied directly to an XML file this needs no extra description it just applies the XPath to the XML and returns.

The second case the *bnx* extension is for a BinX file that contains a pointer to an associated binary file. In this case the XPath is applied over the XML representation of the data implied by

the description and the file it points to. The return result is a MIME encapsulated combination of the resulting data and a BinX description.

The last two cases are both examples where the file is actually the data file itself. In these cases the implementation spots the attempt to apply XPath to the file and looks for an *association* file in the same directory. This is a file in a given XML format that specifies the relationship between files and descriptions, either by individual files, or by extension. We would expect there to be a global file of this type as well as (potentially) overriding instances in each directory.

An important note on the use of XPath: Technically XPath returns a collection of subtrees as its result. This presents a problem for both the BinX case and the XML case in that it complicates the result type. However it is the case that all the subtrees have a common root (since they came from the same document). So all we do is to include the ancestors of the subtrees as well allowing us to combine the subtrees in their original hierarchy. There may be a good reason why this is a bad thing to do but it gives us the result we need in the BinX case at least.

3.3 XML representation of directory listing

We were unable to find a standard XML format for returning a directory listing in (although it has since been suggested that WebDAV may provide one). We therefore defined a simple directory representation shown in Figure 5. Clearly this is something that should be standardised (if it has not already been).

```
<directory name="myDir">
  <file name="myFile.ext" modifiedDate="08/18/03"
        modifiedTime="16:36"
        owner="martin" .../>
   <file  .../>
   <file  .../>
   <file  .../>
<directory/>
```

Figure 5 XML representation of a directory listing

3.4 Working directory definition

The definition of a current working directory (or not) raised an interesting issue. If a service like this changes directory should it return a new service with a different root directory, or should it just maintain state of a new directory.

In our implementation we avoided the issue and just insisted that all accesses were absolute.

However the issue is a recurring one within the OGSI world and analogous questions have been repeatedly raised within DAIS which is trying to understand the ins and outs of each approach. One approach is to make both possibilities available and create a new service when the intention is to pass that handle on to a third party and the subdirectory is all you want to expose.

4. Discussion

It is clear from the work that has been carried out that accessing files has a lot of strong analogies and raises many similar issues to the problems of accessing databases.

For example, one of the valuable features of adding file access into OGSA-DAI was that all the infrastructure that had been constructed to allow for delivery of different modes and transformation was immediately available. The same should be true of the standards work in the sense that all the standards that DAIS needs to develop for interfacing with data movement or transformation technology should be immediately inherited for files.

This implementation built on an earlier version of DAIS and that the specification has moved on considerably since then. We believe however that the recent discussions which involve more operation rich porttypes are even more relevant to file access. Questions as to the right model for representing cursor access to a dataset are directly analogous to the questions of the right model for representing file pointer access to a file.

It is certainly the case that there are file access issues that are clearly not relevant to the DAIS work. These include representations of distributed file hierarchies, or perhaps issues associated with distributed file locking semantics. Although even here there have been relevant discussions within DAIS about database transactions.

It is clear that there is a lot of work involved in defining these file access mechanisms. There are a number of relevant standards including FTP, HTTO, GridFTP, WebDAV, POSIX etc. As a first pass there may be port types to define in the following sorts of areas

- FileSystemAccess – A fairly operation rich port type but one that could be used as a wrapper onto legacy file access sytems such as FTP and HTTP servers. Some of the more advanced features would be allowed to return "not implemented" exceptions and SDEs could be used to export actual service capabilities. The sorts of features that have been discussed include:
    - the ability to instantiate services that represent only a subset of the files in a file system such as all files matching *.jpg (or possibly the results of a 'find' command).
    - the ability to extend the file path to access file contents. For example, appending an XPath query and have the results of running that query over the file returned.

- FileAccess – The FileAccess port type might present traditional pointer based read and write access to a particular file, and might be extended to support detailed query of files of specific type, e.g. XPath access to XML or DFDL described files, or operations to query the header values of JPEG files.
- FileSysemRepresentation – operations for “mounting” distributed files systems into a logical hierarchy

5. Conclusion

In conclusion file access appears to fit into the same framework that DAIS is defining for database access. Many of the issues are directly analogous and appropriate consistency of solution is extremely desirable. However it is also the case that there is a lot of work to be done to get this right and furthermore some of the material, at least, is more appropriately in scope for other groups, such as GridFTP or the newly forming GFS group.

6. Security Considerations

This is an informational document that does not explore the security issues associated with file access.

Author Information

Martin Westhead, M.Westhead@epcc.ed.ac.uk, EPCC, University of Edinburgh. James Clerk Maxwell Building, Mayfield Road, Edinburgh EH9 3JZ, UK.

Glossary

DFDL – Data Format Description Language

Intellectual Property Statement

The GGF takes no position regarding the validity or scope of any intellectual property or other rights that might be claimed to pertain to the implementation or use of the technology described in this document or the extent to which any license under such rights might or might not be available; neither does it represent that it has made any effort to identify any such rights. Copies of claims of rights made available for publication and any assurances of licenses to be made available, or the result of an attempt made to obtain a general license or permission for the use of such proprietary rights by implementers or users of this specification can be obtained from the GGF Secretariat.

The GGF invites any interested party to bring to its attention any copyrights, patents or patent applications, or other proprietary rights which may cover technology that may be required to practice this recommendation. Please address the information to the GGF Executive Director.

Full Copyright Notice

References

[1] BinX project webpage http://www.edikt.org/binx

[2] DFDL WG webpage https://forge.gridforum.org/projects/dfdl-wg