

## Grid Data Access and Integration in OGSA

**Project Title:** OGSA-DAI GridServe

**Document Title:** Grid Data Access and Integration in OGSA

**Document Identifier:** EPCC-GDS-WP2-ARCH v1.2

**Document Filename:** OGSA-DAI-spec-1.2.doc

**Distribution Classification:** DRAFT: Unrestricted

**Authorship:** Malcolm Atkinson, Rob Baxter, Neil Chue Hong

**Reviewers:** Norman Paton, Dave Pearson

### Document History:

Personnel	Date	Summary	Version
MPA	26/01/2002	First draft	0.1
RMB	01/07/2002	Revised for submission to TA	0.2
RMB	02/07/2002	WP-approved for submission to TA	1.0
NCH, NP	04/07/2002	Revised for submission to DAIS-WG	1.1
RMB, DP	05/07/2002	Further revisions; resubmitted to DAIS-WG	1.2

## Contents

Contents .....	2
1 Overview of Data Access and Integration in OGSA .....	3
1.1 An Overview of the OGSA-DAI architecture.....	3
2 Database Access and Integration Functionality .....	5
2.1 Grid Data Services and Factories.....	5
2.1.1 Grid Data Service Data Elements .....	5
2.1.2 Grid Data Service Statements .....	6
2.2 Grid Data Service Registry .....	7
2.3 Grid Data Transport Services.....	8
3 References.....	9

# 1 Overview of Data Access and Integration in OGSA

This document provides an initial view of data access and integration services within the framework specified by the recently-published Open Grid Services Architecture (OGSA) [TCF+02][FKNT02]. This document offers a high-level view of an extension to the OGSA architecture to support data access and integration services (OGSA-DAI).

This document builds on earlier work by the UK e-Science Database Task Force presented at the Fourth Global Grid Forum, Toronto, Canada, in February 2002 [PAD+02]. It is a draft document and should be regarded as a work in progress.

This document should also be read in conjunction with a number of drafts describing the detailed specifications of some of the data services discussed herein. These drafts have been produced by the OGSA-DAI project team at EPCC during prototyping and architectural exploration from the perspective of XML databases. These drafts are as follows:

- Grid Data Service Specification for XML Databases [KSB02];
- Grid Data Service Factory Specification for XML Databases [KB02].

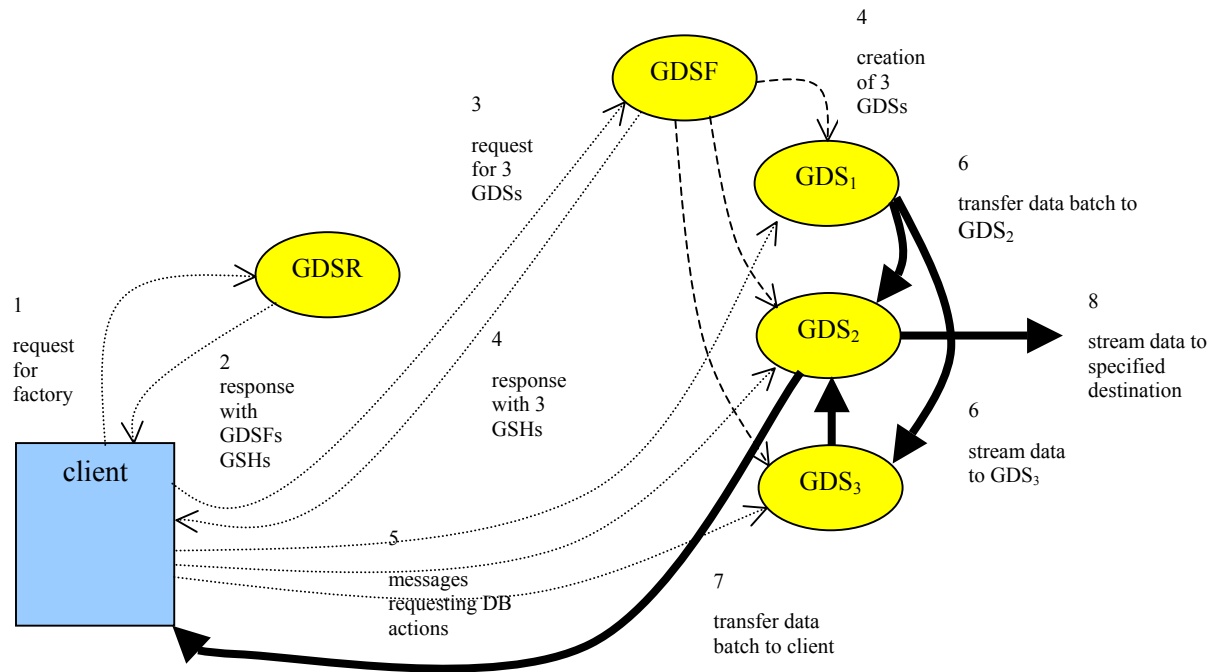
Two further drafts are in preparation:

- Grid Data Transport Service Specification [CB02];
- Grid Data Service Registry Specification [AB02].

Note that while some of these documents define prototype Grid data services in the context of XML databases (in particular, the Apache Xindice database [Xin]), others (including this document) are generic. The aim of the OGSA-DAI architecture is to define OGSA portTypes for generic data access and integration services, while requiring services deployed in the context of particular database or data store systems to distinguish themselves through their service data elements. OGSA-DAI strives to present uniform access to a wide range of data sources.

## 1.1 An Overview of the OGSA-DAI architecture

There are several classes of component that we deal with in OGSA-DAI. In an earlier note [Atk02] they were referred to as *Grid Data Services* (GDS), *Grid Data Service Factories* (GDSF) and a *Grid Data Service Registry* (GDSR). Data is moved between these by a collection of mechanisms, from the simplest use of the OGSA infrastructure to transfer XML documents to sophisticated mechanisms such as Grid-FTP [Gftp] and MQ Series [MQS]. A generic term in that earlier note for these was a *Grid Data Delivery Service*, but here the more generic term *Grid Data Transport Service* (GDTS) is suggested.



An illustrative diagram of a deployed OGSA-DAI instance is shown above.

This diagram illustrates a scenario in which a sequence of steps takes place involving a client, five OGSA-DAI components and four grid data transport services are involved (note that not all of the control messages are shown). The notation uses ellipses for OGSA-DAI components, and a rectangle for a client. It uses various forms of dotted open arrow for control messages and solid thick arrows for the applications of GDTSS. The scenario presumes that the client wishes to achieve two things using data integrated from three sources managed by GDS:

- Obtain data combined from source<sub>1</sub> and source<sub>3</sub>.
- Send data combined from source<sub>1</sub>, source<sub>2</sub> and source<sub>3</sub> as a stream to a specified third party.

In the scenario we envisage the client using OGSA-DAI as follows:

1. It knows the location of one of the GDSR and sends to that GDSR a description of the GDS it requires. Presumably the description describes the three sources that must be accessed and the operations that are required.
2. The GDSR replies with an indication that the required GDS do not exist, but provides a list of GDSFs (as their GSHs) that can generate the required GDS.
3. The client chooses one of these, perhaps after exercising some dialogues to determine more about the offered GDSF. Three calls are made to the CreateService operation on the GDSF requesting the creation of the GDSs.
4. The GDSF schedules the construction and initialisation of the three GDS, making each an OGSA-DAI compliant proxy for the three sources of interest, and returns a GSR (containing a GSH) for each GDS instance to the client.
5. After each GDS has been queried using the FindServiceData operation to determine the details of its capabilities (not shown), the client sends messages to each GDS indicating the operations (e.g. Query, Update, bulkLoad) each is required to undertake. In this example:

- GDS<sub>1</sub> should transport a batch of data (from a Query) to GDS<sub>2</sub> and send a stream of data to GDS<sub>3</sub>, to be utilised by operations on the GDSs.
- GDS<sub>2</sub> should expect a batch of data from an operation performed by GDS<sub>1</sub> and a stream of data from GDS<sub>3</sub>. It should also set up the flow of a stream of data to a specified third party.
- GDS<sub>3</sub> should perform an operation to send a stream of data (from an operation on data from GDS<sub>1</sub> and a Query on source<sub>3</sub>) to GDS<sub>2</sub>.

For each of these data transfers there would be a description of how it should be performed.

6. GDS<sub>1</sub> uses a GDTS to send the specified batch of data to GDS<sub>2</sub> and another GDTS to send a stream of data to GDS<sub>3</sub>. This may happen sequentially or concurrently if the operations carried out by GDS<sub>1</sub> are specified as parallel.
7. GDS<sub>2</sub> uses the incoming data streams, its own data and a GDTS to send a batch of data to the client and
8. to initiate the stream of data to the third party. In fact we do not know the temporal ordering of steps 7 to 9 unless it was specified.

This scenario does not illustrate all possible relationships. For example, a GDSR may use a GDS to support its own operation, and there may be a requirement for data transport between tasks within the same GDS.

## 2 Database Access and Integration Functionality

In this section we describe the fundamental services required in the OGSA-DAI architecture. Two of these – *Grid Data Services* and *Grid Data Service Factories* – we treat together (Section 2.1) as being closely related. With these two classes of service, the nature of the underlying data source must be exposed to greater or lesser extents within the service. Two others – *Grid Data Transport Services* and *Grid Data Service Registries* – are more generic.

### 2.1 Grid Data Services and Factories

We envisage a fundamental service, a Grid Data Service (GDS), which will provide the point-of-access to data sources or resources on the Grid. GDS may be persistent or transient. A persistent GDS could be envisaged as providing a permanent “Grid interface” for a single database or other data resource, while transient GDS could be created on request to manage a single user connection to a data store or temporarily replicated/cached data.

#### 2.1.1 Grid Data Service Data Elements

Both GDS and GDSF will must possess a standard set, Grid Data Service Data Elements (GDSDE), of Service Data Elements ([TCF+02] section 4.3) that denote the accessible data, access and manipulation capabilities, etc. of a GDS or of the products of a GDSF. These GDSDE will be of vital importance in enabling service requesters to discover the capabilities and contents of the data resource associated with the GDS or GDSF (cf. Section 2.2).

The GDSDE will comply with an XML schema that will determine their minimum content but permit extension for particular GDS, such as those associated with a particular data model, e.g. relational, or associated with a particular underlying DBMS, such as DB2, Oracle, or MySQL. These extensions will normally themselves comply with standardised and published XML schema.

The following is an illustrative and non-exhaustive list of the kinds of information that will be encoded by a GDSDE.

- The types of data model supported by this GDS or by the products of this GDSF, e.g. Relational, Object, XML, etc. (Static)
- The languages supported by this GDS, by this GDSF or by the products of this GDSF, e.g. SQL, Xquery, etc., and the standard to which any language conforms, e.g. SQL'92 (Static)
- The data management and manipulation operations supported by this GDS, by this GDSF or by the products of this GDSF, e.g. schema edit, bulk load, query, insert, update, delete, etc. (Static)
- Grid Data Transport Services supported. This will state which GDTs are supported by this GDS or GDSF for data input, for data return and for third party delivery. (Static)
- The data content that is accessible via this GDS or via the products of this GDSF. In early versions, this will be relatively simple technical metadata, such as collections of names, such as database names, schema names, view names and table names. In later versions we aspire to deliver more contextual metadata information about the contents that will support semantic interpretation of content-based queries. (Static)
- Data Quality indicators. The ownership, version, and currency of data from which users may establish aspects of its value and usage.
- Overall and particular quantitative information about the data that is accessible or may be stored via this GDS or via products of this GDSF. This will provide at least overall values, e.g. physical size, number of logical records, but will normally extend to deliver values pertaining to substructures, such as tables, within the accessible data. (Dynamic)
- Capacity and performance information. What volume of operations are permitted, e.g. maximum number of simultaneous transactions, maximum result set size, maximum insert, etc. (Dynamic)
- Access policies. These indicate what permissions users (or classes of users) and client software acting as their agents will be granted. (Dynamic)
- Charging policies. These indicate how a client's account would be charged. (Dynamic)

The indication *static* is used here to note that this property will be static for the lifetime of each GDS or GDSF instance and that it is therefore technical metadata and could be represented as a *structural* SDE in the terms of [TCF+02] section 4.3. The items marked *dynamic* communicate information that is likely to change during a GDS or GDSF's lifetime, and these should therefore be communicated via *instance* SDEs using that same nomenclature.

In many cases the dynamic information will need to be generated by examining the current state of the data or the data management system. There are challenges to be addressed here. For example, the production of useful content data must be automated.

### 2.1.2 Grid Data Service Statements

A Grid Data Service must be capable of interpreting statements in order that database operations, such as query and update may be performed. However, there is a requirement to support multiple data models, multiple versions of the operations provided for these models and multiple notations for describing those operations. For example, some minimum subset must be supported for all GDS presenting relational data, e.g. SQL'92 statements through JDBC version 2. On the other hand, most commercial DBMS often provide functional extensions to standards which a service provider may choose to exploit at the expense of sacrificing mobility and independence.

To achieve this we arrange that the operations of the *GridDataService* **portType** or the *GridDataServiceFactory* **portType** contain as a parameter the notation being used. Applications working with the defined minimal language will be correctly supported by any GDS with that data model.

Using ‘query’ as an example, the general form of an operation is therefore:

#### **GridDataService :: query**

##### **Input**

- The name of the query notation (which may imply the operations permitted and the data model).
- The query in that notation (which may contain escaped identifiers for parameters).
- Optionally the parameters of the query operation. These may be passed by value or by reference ( by specifying the GSH of a GDTS that will supply them).
- Optionally an expiry time until which the result may be claimed. If omitted, a default time is used.

Optionally an override to the default arrangement for returning results e.g. the GSH of a GDTS.

##### **Output**

- A status indicator
- Result of the query (if a GDTS has not been specified)

The details of the **portType** are yet to be decided.

A further issue still under discussion is how to address batching. Supplying a GDS with a batch of operations it should accomplish will often permit it to schedule them more efficiently. It is also useful to group operations together, e.g. to denote a session or transaction which operates entirely within the same binding, authentication, authorisation and accounting regime. We can therefore envisage a GDS operation that takes a script and performs all of its operations. (Note that, in this context, commercial RDBMS do support batching through SQL scripting, and through procedural SQL.)

#### **GridDataService :: perform**

Again, this could be recognised by the specified notation being that of a process scripting language, such as a web services choreography language. It is unlikely that the full richness of such a standard language could be analysed to perform the validation and optimisation that motivates a batch-processing interface. It is also inappropriate to define a new language. The compromise might well be a defined subset of a standard language with a recognised set of operations. As languages in this area are developing rapidly this is currently an open issue.

For illustration, reference [KSB02] defines a prototype specification for the *GridDataService* **portType** in the context of XML database access. Reference [KB02] defines a similar prototype specification for a Grid Data Service Factory, as a particular form of the Grid service *Factory* **portType** with related service data for XML database access.

## **2.2 Grid Data Service Registry**

A Grid Data Service Registry (GDSR), which is a specialised extension of a Registry ([TCF+02] section 9) that maintains a collection of Grid Service Handles for a set of GDS and/or GDSF that have registered with it. It provides a query service to authorised clients that will provide responses based on the GDSDE of its registered set of GDS and GDSF.

This will be a registry and will implement the *Registry* **portType**. It should support the Xpath query language and should implement the *NotificationSource* **portType** ([TCF+02] section 9).

The minimal extension it is required to support is that it will accept and process queries about the GDSDE defined above. To guarantee to do this it must verify that any grid service that registers with it supports the *GridDataService* **portType** or the *GridDataServiceFactory* **portType** and that it provides GDSDEs compliant with the specified XML schema. It will consult registered GDS and GDSF for dynamic data, but may cache static data. It may consolidate or interpret this data in order to support more sophisticated data resource discovery. Such extensions are more likely in later versions.

As an example, reference [AB02] defines a prototype specification for a Grid Data Service Registry as a particular form of the Grid service *Registry* **portType** with related service data to support Grid data services.

## 2.3 Grid Data Transport Services

Data is moved between Grid Data Services by a collection of mechanisms, from the simplest use of the OGSA infrastructure to transfer XML documents to sophisticated mechanisms such as Unix pipes, Grid-FTP [Gftp], MPI [MPI] and MQ Series [MQS]. We call these Grid Data Transport Services (GDTs). The movement may be synchronous or asynchronous, they may be to or from a third party, they may use store and forward mechanisms, they may perform a bulk transfer or provide a delivery stream and they may offer various reliability guarantees, such as reliable delivery. As they may involve significant resources they must use the soft state mechanisms to avoid indefinite or unbounded resource commitments.

Their scope is still being defined. It is clear that it must include large-volume delivery, e.g. for a bulk load, for the delivery of query results and for coupling stages in a distributed query [SGW+02]. It is not yet clear whether the transport of control and coordination signals should be part of the same scheme.

An abstract description of GDTs is required that will cover the full scope, will permit mapping to the expected target mechanisms and that will eventually support both static and dynamic optimisation decisions.

Their manifestation as a grid service will require a set of SDE to provide static data about their capabilities, capacity and performance, and dynamic data to monitor progress, performance and fault diagnosis. In this manifestation they will also require a **portType** that delivers operations such as *open*, *initialise*, *write\_all*, *write\_next\_part*, *read\_all*, *read\_next\_part*, *close*, *size*, *empty*, *full*. There may be others, such as *reset*.

Reference [CB02] illustrates some of these ideas, defining a prototype Grid Data Transport Service in terms of a new *GridDataTransportService* **portType** and related service data



### 3 References

- AB02     Antonioletti, M. and Baxter, R., *Grid Data Service Registry Specification*, in preparation.
- Atk02     Atkinson, M.P., The Open Grid Services Architecture and a Model for Database Access and Integration Services on the Grid, contribution to UK DBTF discussions, 26 January 2002.
- CB02     Chue Hong, N. and Baxter, R., *Grid Data Transport Service Specification*, in preparation.
- Fie02     Fielding, R.T., *Architectural Styles and the Design of Network-based Software Architectures*, PhD Dissertation, University of California, Irving, 2000.
- FKNT02   Foster, I., Kesselman, C., Nick, J. and Tuecke, S., *The Physiology of the Grid: An Open Grid Services Architecture for Distributed Systems Integration*, presented at GGF4, Feb. 2002.  
<http://www.globus.org/research/papers.html>
- GCG+02   Gannon, D., Chiu, K., Govindaraju, M. and Slominski, A., *An Analysis of the Open Grid Services Architecture*, Report Commission by the UK e-Science Core Programme, Department of Computer Science, Indiana University, Bloomington, IN, USA, April 2002.
- Gftp     Globus Project, *GridFTP: Universal Data Transfer for the Grid*, September 2002.  
<http://www.globus.org/datagrid/gridftp.html>
- GSB+02   Graham, S., Simeonov, S., Boubez, T., Davis, D., Daniels, G., Nakamura, Y. and Nayama, R., *Building Web Services with Java: Making Sense of XML, SOAP, WSDL and UDDI*, Sams Publishing, Indianapolis, Indiana, USA, 2002.
- Kun02     Kunnszt, P.Z., *The Open Grid Services Architecture – A Summary and Evaluation*, Report Commission by the UK e-Science Core Programme, IT Division – Database Group, CERN, 1211 Geneva, Switzerland, April 2002.
- KSB02     Krause, A., Smyllie, K. and Baxter, R., *Grid Data Service Specification for XML Databases*, in preparation.
- KB02     Krause, A. and Baxter, R., *Grid Data Service Factory Specification for XML Databases*, in preparation.
- MPI     Message Passing Interface Forum, <http://www.mpi-forum.org/>
- MQS     IBM, *WebSphere MQ family*, <http://www-3.ibm.com/software/ts/mqseries/>
- PAD+02   Paton, N.W., Atkinson, M.P., Dialani, V., Pearson, D., Storey, T. and Watson, P., *Database Access and Integration Services on the Grid*, UK DBTF working paper, January 2002 (presented at GGF4). <http://www.cs.man.ac.uk/grid-db/>
- Pea02     Pearson, D., Data Requirements for The Grid: Scoping Study Report, UK DBTF working paper, February 2002 (presented at GGF4). <http://www.cs.man.ac.uk/grid-db/>
- SGW+02   Smith, J., Gounaris, A., Watson, P., Paton, N.W., Fernandes, A.A.A., and Sakellariou, Distributed Query Processing on the Grid, June 2002.
- TCF+02   Tuecke, S., Czajkowski, K., Foster, I., Frey, J., Graham, S., Kesselman, C. and Nick, J., *Grid Service Specification*, circulated via OGSF-WG, 13 June 2002. <http://www.globus.org/research/papers.html>
- Wat02     Watson, P., *Databases and the Grid*, version 3, Technical Report CS-TR-755, Newcastle University, Department of Computer Science, January 2002. <http://www.cs.man.ac.uk/grid-db/>