

Grid Database Service Specification Primer

Status of This Memo

This memo provides information to the Grid community regarding the specification of Grid Database Services. This memo illustrates the possible use of Grid Database Services and is not a specification. It does not define any standards or technical recommendations. Distribution is unlimited.

Copyright Notice

Copyright © University of Edinburgh, IBM, University of Manchester, Florida State University (2003). All Rights Reserved.

Abstract

This primer illustrates some of the proposed capabilities of Grid Data Services with descriptions and examples. The document includes a brief introduction to the GDS specification draft, a presentation of some of the possible scenarios for interaction between clients and GDSs, and a detailed description of several examples of Grid Data Service requests and responses.

Contents

1. Vision and Goals of DAIS-WG	2
2. Basic Functionality of a Grid Data Service.....	2
3. Capabilities and Scenarios for Data Access Through the Grid Data Service Port Type	3
4. Sample GDS Input and Output Documents	4
4.1 A Simple GDS Request.....	5
4.2. Asynchronous Execution and Delivery by ftp	5
4.3. Asynchronous Query with Independent Request for Delivery.....	7
4.4. Asynchronous Update Query with Independent Delivery of Data.....	8
5. Interaction with Grid Data Transport Services.....	9

1. Vision and Goals of DAIS-WG

The DAIS working group strives to provide a standard specification for access to data resources on the Grid in order to create consistent and scalable data access services. By focusing on services, the intention is to ease application development through the provision of composable components, and to track developments in the Open Grid Services Architecture (OGSA). The group does not seek to develop new data storage systems, but rather to make such systems more readily usable within a Grid framework.

The separation of the request for data and the delivery of the data is of primary importance to the goal of achieving scalable services. Grid services are expected to handle requests that involve massive amounts of data and computation. The remote procedure call model of most database clients, where data is returned directly to the client, is unsuitable to be used directly in this environment. Creating effective database access requires the configuration of middleware services. DAIS-WG intends to provide a consistent framework for these services.

A basic principle of DAIS-WG is that a client of a Grid Data Service (GDS) must be able to specify several operations in a single request. The GDS draft specification defines the `GridDataServicePerform` XML document schema to be used by GDS clients to describe a collection of operations and the `GridDataServiceResponse` XML document schema to be used by GDSs to return results to clients.

2. Basic Functionality of a Grid Data Service

A Grid Data Service (GDS) is based on the OGSA model. Figure 1 illustrates the basic architecture of the GDS environment. A client who wishes to interact with a particular database through a GDS will contact a suitable Grid Service Registry to find a Grid Data Service Factory that is able to create a GDS for the database. The request to the registry may specify which database is required, or may specify the characteristics of the data. The registry request may also specify required characteristics of the GDS. The client will interact with the registry to determine which factory can create a suitable GDS.

The client continues by requesting that the factory provide a GDS. The factory may create a new GDS object or allocate an existing GDS, depending on the needs of the client and the capabilities of the GDS. The factory returns a Grid Service Handle to the client. The request to the factory may include authorization information for the database connection and other details of the required service.

The draft GDS specification includes the definition of the `GridDataService` porttype with its `GridDataService::perform` method. The client of a GDS can describe a sequence of operations by creating a document according to the `GridDataServicePerform` XML schema. The client may then call the `perform` method of the GDS with the document as its parameter. The GDS will perform the required actions and return a `GridDataServiceResponse` document to the client.

The underlying OGSA infrastructure will provide authentication, lifetime management, transaction management and other services to the GDS. An OGSA Grid Data Transport porttype is also being proposed. This will support the basic requirements of GDS to GDS transport. The specification of more sophisticated and general transport services is considered to be outside of the scope of the DAIS working group.

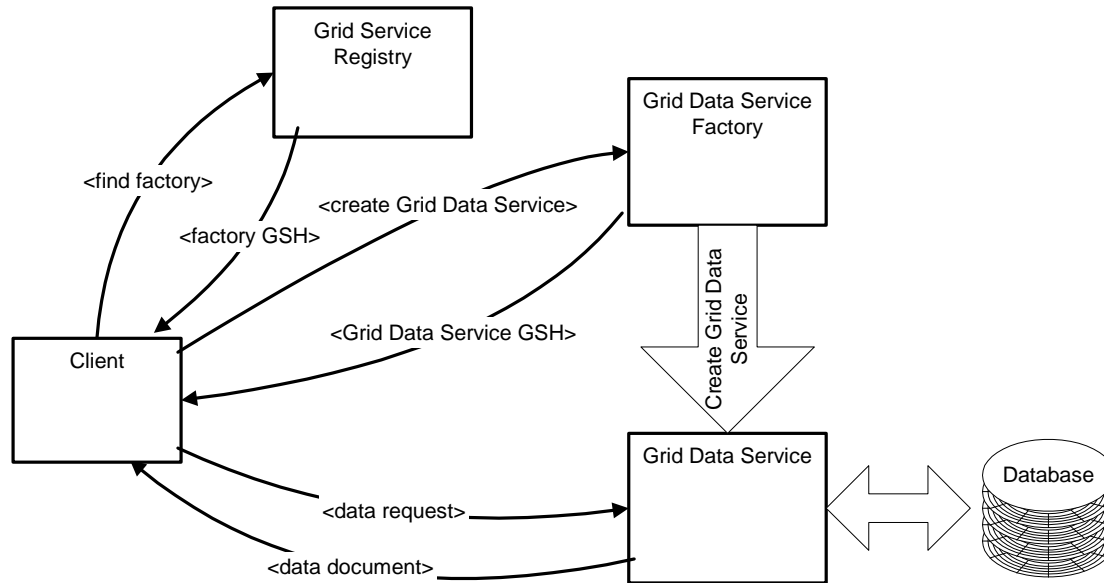


Figure 1. Basic GDS Architecture

3. Capabilities and Scenarios for Data Access Through the Grid Data Service Port Type

3.1 Grid Data Service Realization

what does a GDS instance mean?

3.2 Grid Data Service Interaction

Each Grid Data Service (GDS) supports a variety of interaction strategies. Figure 2 illustrates some of the possible scenarios in which an analyst (A), a producer of data (P), a consumer of data (C) and one or more GDSs (G) may interact. Each scenario in Figure 2 represents a single query, which may be a request for data or an update. Interactions between clients and GDSs may contain any number of queries and thus can be much more complex than these scenarios.

In scenario 1, the analyst sends a query (Q) to the GDS and receives the result data in response. The analyst communicates with the GDS by creating a `GridDataServicePerform` document and calling the `perform` method of the GDS. The GDS responds to this call with a `GridDataServiceResponse` document that contains a response to each request in the `perform` document. An example of this simplest of all client-service interactions is described in Section 4.1.

In scenario 2, the analyst sends a query to the GDS along with a delivery specification (D) that directs the GDS to send the results of the query to the consumer. In this case, the analyst receives a status response from the GDS once the query has been validated and is ready to be executed. The processing of the query and the delivery of the data to the consumer may proceed asynchronously with the original request. The analyst may make subsequent requests to the GDS to inquire about the status of the execution and the delivery. Section 4.2 illustrates this scenario with a detailed example.

In scenario 3, the analyst again sends a query for asynchronous execution and receives a status response, but this time, the analyst does not specify what to do with the results of the query. The analyst then sends information to the consumer and the consumer makes its own request for the data, which is returned in its response document. An obvious extension of this scenario involves the consumer requesting an asynchronous delivery of the data queried by the analyst. In this case the results of the query would not be returned directly to the consumer. Section 4.3 illustrates this scenario with a detailed example.

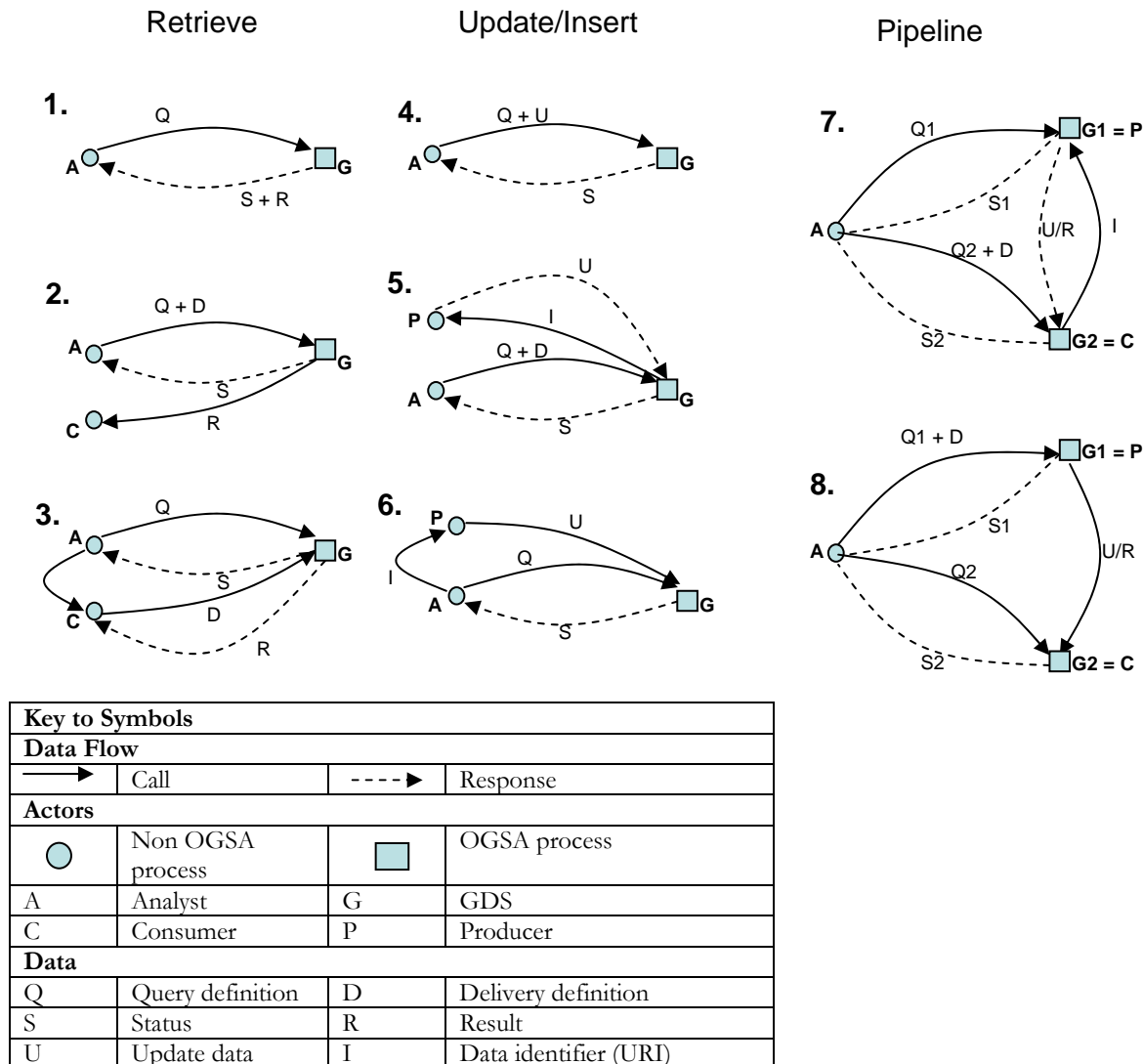


Figure 2. Scenarios for Interactions Between GDSs and Clients.

Scenarios 4, 5, and 6 are database update operations. Scenario 4 is a synchronous update in which the analyst sends the update query (Q) and the update data (U) to the GDS. The status response is typically a report on how many data objects were modified by the update. Scenario 5 is an asynchronous update in response to a query plus delivery document from the analyst. In scenario 6, the analyst specifies the GDS update without specifying where the data will come from. The producer provides the data in a subsequent interaction. Section 4.4 illustrates scenario 5 with a detailed example.

Scenarios 7 and 8 illustrate the possibility of pipelining GDS services. In each scenario, data from producing GDS G1 is used to update the data source of the consuming GDS G2. Scenario 7 shows the analyst directing GDS (G1) to execute query Q1 and hold the results for later delivery. Then the analyst sends an update query (Q2) and a delivery specification to G2. In response, G2 acts as a GDS client and calls G1 with its request for the data. The update takes place once the data begins flowing from G1 to G2. Scenario 8 shows the analyst sending a delivery specification to G1 that directs it to send the result data to G2.

4. Sample GDS Input and Output Documents

4.1 A Simple GDS Request

Example 1 shows the request and response documents for the processing of a simple SQL query by a GDS, as illustrated in scenario 1 of Figure 2. Lines 1–10 is a `GridDataServicePerform` document that is used as the argument to a call to `GridDataService::perform`. The perform document contains a `GridDataServiceRequest` (lines 3–8) named “request1” that declares an SQL query (lines 4–7). The result format (line 6) will conform to documents produced by a Java `WebRowSet`.

The perform document includes an execute tag (line 9) that directs the GDS to execute the `GridDataServiceRequest` named “request1.” The SQL statement is executed by the database server.

The `GridDataServiceResponse` document includes two tags: the response to the `GridDataServiceRequest` in lines 13-15 and the response to the Execute in lines 16–21. The data is not shown in the example.

The response to the `GridDataServiceRequest` is simply an acknowledgement that the request is acceptable. A request would be unacceptable if it included any activities that were not supported by the GDS. For example, an `SQLStatement`, as in lines 4–7 may not be supported by an XML data service.

The response to the execute tag contains a `RowSet` document which is the data returned by the query of line 5.

Example 1: SQL Query with data returned in response document

Document submitted to GDS:Perform

```

1    <?xml version = "1.0" encoding = "UTF-8"?>
2    <GridDataServicePerform>
3      <Request name = "request1">
4        <SQLQueryStatement name = "statement1">
5          <Expression> select * from mypictures </Expression>
6          <Result name="result1"/>
7        </SQLQueryStatement>
8      </Request>
9      <Execute name="execute1" request="request1"/>
10   </GridDataServicePerform>

```

Document returned by GDS:Perform

```

11   <?xml version="1.0" encoding="UTF-8"?>

```

```

12  <GridDataServiceResponse>
13    <Response name = "request1">
14      OK
15    </Response>
16    <Response name = "execute1">
17      <Result name = "execute1.statement1.result1">
18        <RowSet> table data inline here
19        </RowSet>
20      </Result>
21    </Response>
22  </GridDataServiceResponse>

```

4.2. Asynchronous Execution and Delivery by ftp

The request of lines 7–21 includes a BasicDelivery (lines 9–14) to specify that the result of the execution of the SQLStatement is to be sent by ftp to another computer. The execution of the GridDataServiceRequest is not required to wait for the delivery to be completed. As soon as statement2 begins its execution, the response document of lines 17-24 can be created and returned to the caller. The response document includes sufficient information to allow the client to request information about the status of the execution of statement2 and the delivery of the result.

Thus the query and delivery are executed asynchronously and the GDS client does not wait for them to be completed.

Example 2: Asynchronous execution of Parameterized SQL query with data returned via smtp

Document submitted to GDS:Perform

```

1  <?xml version = "1.0" encoding = "UTF-8"?>
2  <GridDataServicePerform>
3    <Request name = "request2">
4      <SQLQueryStatement name = "statement2">
5        <Expression> select * from myimages </Expression>
6        <Result name="result2"/>
7      </SQLQueryStatement>
8      <BasicDelivery name = "d2">
9        <FromLocal from = "result2"/>
10       <ToURL url = "smtp://ogsadai@nesc.ac.uk"/>
11     </BasicDelivery>
12   </Request>
13   <Execute name = "execute2" request = "request2"/>
14 </GridDataServicePerform>

```

Document returned by GDS:Perform

```

15 <?xml version = "1.0" encoding = "UTF-8"?>
16 <GridDataServiceResponse>
17   <Response name = "execute2">
18     <Result name = "execute2">
19       GSH of service that I must contact to interact with request

```

```

20         </Result>
21     </Response>
22 </GridDataServiceResponse>

```

4.3. Asynchronous Query with Independent Request for Delivery

Example 3 illustrates the way that data response to a query may be held in a GDS awaiting specification of the required delivery, as shown in scenario 3 of Figure 2. The initial perform request and response are in lines 1–14 and 15–24, respectively. The second perform request to the GDS (lines 25–34) results in the delivery of the data in the response document of lines 35–43.

The BasicDelivery in lines 8–11 of the first perform document specifies that the result of statement3 is to be held (line 10) in the GDS awaiting further delivery instructions. When the GridDataServiceRequest is executed (line 13), the query execution may begin. The response of lines 15–24 will be returned to the client as soon as the query has begun execution. The result includes a DataId (line 21) that identifies the result data for subsequent requests. The execution thus proceeds asynchronously with the client's perform request.

The perform document of lines 25–34 may be delivered to the GDS at any time after the previous response document is returned. The BasicDelivery of lines 28–31 specifies that the data held as "execute3.d3" is to be returned to the caller in the response document. This GridDataServiceRequest is executed synchronously with the query execution. The response to this second perform document cannot be returned until the query has completed.

Example 3: Asynchronous Query with result held in GDS and returned by separate request

Document submitted to GDS:Perform

```

1     <?xml version = "1.0" encoding = "UTF-8"?>
2     <GridDataServicePerform>
3         <GridRequest name = "request3">
4             <SQLQueryStatement name = "statement3">
5                 <Expression> select * from myimages </Expression>
6                 <Result name="result3"/>
7             </SQLQueryStatement>
8             <BasicDelivery name = "d3">
9                 <FromActivity from = "result3"/>
10                <ToWait/>
11            </BasicDelivery>
12        </Request>
13        <Execute name = "execute3" request = "request3"/>
14    </GridDataServicePerform>

```

Document returned by GDS:Perform

```

15    <GridDataServiceResponse>
16        <Response name = "execute3">
17            <Result name = "execute3">
18                GSH of service that I must contact to interact with request
19            </Result>

```

```

20         <Result name = "d3">
21             <DataId>execute3.d3</DataId>
22         </Result>
23     </Response>
24 </GridDataServiceResponse>

```

Document submitted to GDS:Perform to retrieve data

```

25 <?xml version = "1.0" encoding = "UTF-8"?>
26 <GridDataServicePerform>
27     <Request name = "request3a">
28         <BasicDelivery name = "d3a">
29             <FromWait dataId = "execute3.d3"/>
30             <ToResult/>
31         </BasicDelivery>
32     </Request>
33     <Execute name = "execute3a"/>
34 </GridDataServicePerform>

```

Document returned by GDS:Perform with data attached

```

35 <?xml version="1.0" encoding="UTF-8"?>
36 <GridDataServiceResponse>
37     <Response name = "execute3a">
38         <Result name = "execute3a">
39             <RowSet> table data inline here
40         </RowSet>
41     </Result>
42 </Response>
43 </GridDataServiceResponse>

```

4.4. Asynchronous Update Query with Independent Delivery of Data

The final example of this primer demonstrates scenario 5 of Figure 2. In this example, a client sends a insertion request to the GDS that identifies the data to be inserted through an ftp request to another server. Fetching the data and inserting it into the data source is performed asynchronously from the request.

Lines 4–7 specify that data item d4 of the GDS is to be created with data available from the ftp service of line 6. The execution (line 13) of the update statement of lines 8–11 initiates the data fetch and insertion but does not wait for the insertion to be completed. The response in lines 18–23 to the execution contains the information necessary for the client to find out the status of the transfer and insertion.

A final example of client–GDS interaction is illustrated by the termination in line 14 and its response in lines 24–26. The termination operation refers to the asynchronous execution in Example 2 of a select statement. This execution was defined to deliver its result data by ftp. The response in line 25 indicates that execute2 was terminated properly. This use of terminate highlights the way that GDSs maintain state through named objects that can be referenced by clients. The terminate also emphasizes the need for security control in GDSs: in this case to make sure that a request to terminate an operation is authorized to do so.

Example 4: SQL insert with data coming from ftp**Document submitted to GDS:Perform**

```

1    <?xml version = "1.0" encoding = "UTF-8"?>
2    <GridDataServicePerform>
3      <Request name = "request4">
4        <BasicDelivery name = "d4">
5          <FromURL url =
6            "ftp://ogsadai.org.uk/resultsets/newresultset.txt"/>
7          <ToLocal name="input"/>
8        </BasicDelivery>
9        <SQLUpdateStatement name = "statement1">
10         <SQLParameter position = "1" from = "input"/>
11         <Expression>insert into myimages values ? </Expression>
12       </SQLUpdateStatement>
13     </Request>
14     <Execute name = "execute4" request = "request4"/>
15     <Terminate name = "terminate1" runningRequest = "execute2"/>
16   </GridDataServicePerform>

```

Document returned by GDS:Perform

```

17   <?xml version = "1.0" encoding = "UTF-8"?>
18   <GridDataServiceResponse>
19     <Response name = "execute4">
20       <Result name = "execute4">
21         GSH of service that I must contact to interact with request
22       </Result>
23       <Result name = "statement1">started</Result>
24     </Response>
25     <Response name = "terminate1">
26       <Result name = "execute2">terminated correctly</Result>
27     </Response>
28   </GridDataServiceResponse>

```

5. Interaction with Grid Data Transport Services

Grid data transport services (GDTS) will include capabilities for streaming and block transfer, as well as data distribution and merging. The details of how GDT services will be defined and implemented is beyond the scope of the draft GDS specification. However, the delivery specifications of the draft GDS specification provide a strategy for connecting data and transport services. Each GDTS will be able to interact with another GDTS to negotiate for optimal transportation of data objects.

We anticipate that the GridDataServicePerform XML schema will be extended to define the notion of a Grid Data Handle (GDH). The GDH samples in this document have taken the form of the identification of a running request within a Grid Data Service. The Grid Data Service itself being identified by a GSH. A GDH may use the GDTS protocol to specify that a pair of GDTSs will collaborate to move the data from one to the other.

It is likely that GDS implementations will include GDTS porttypes or will be configured with closely coupled GDTS to allow for optimal delivery of data from one data service to another.

Currently the GDS specification defines a simple GridDataTransport portType to enable simple GDS to GDS communications.

SL: Other stuff comes at the end of the spec document...

Author Information

Trademarks

Intellectual Property Statement

Full Copyright Notice

References