

Category: INFORMATIONAL

## GGF10 DAIS Usage Scenarios

### *Status of This Memo*

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

Copyright © Global Grid Forum (2004). All Rights Reserved.

### Table of Contents

Table of Contents .....	1
1. Information Dissemination with DAIS services.....	2
2. DAIS Usage Scenarios .....	3
2.1 Synchronous query with result in return value .....	3
2.2 Asynchronous query with delivery to third party.....	4
2.3 Asynchronous query with independent pull delivery .....	5
2.3.1 DS Support for Scenario 3.....	5
2.3.2 ID support for scenario 3 Pull model .....	6
2.3.3 ID support for scenario 3 Push model .....	6
2.3.4 ID support for scenario 3a (pull model).....	7
2.4 Synchronous request for bulk update .....	7
2.5 Asynchronous request for bulk update with pull of update data from non-Grid service.....	8
2.6 Asynchronous update with independent delivery .....	8
2.7 Pipelining OGSA DS services. Query result data is pulled from producer DS by consumer DS for update .....	9
2.8 Pipelining OGSA DS services. Query result data is pushed from producer DS to consumer DS for update.....	10

This document illustrates some of the ways the porttypes of the GGF10 draft specification for Database Access and Integration Services (DAIS) and the proposal for information dissemination services may be used to create client-service interactions.

The DAIS working group has maintained a collection of scenarios that represent a variety of interaction styles that are important to users of Grid databases. This document presents 8 of these scenarios and explains how they can (or cannot) be realized through the services supported by the DAIS specifications.

Since GGF9, the DAIS working group, led by researchers at IBM and Oracle, among others, has proposed the Information Dissemination (ID) model to support a wide range of data transfer

activities. The ID model is described in detail in the “Information Dissemination in the Grid Environment” document that is available as one of the GGF 10 DAIS documents.

Table 1 lists 8 scenarios that have been presented at previous GGF meetings. Of these scenarios, 3 are supported directly by the DAIS Data Services porttypes, 5 are supported by the ID porttypes, and 1 is not currently supported in the draft specifications. The unsupported scenario 5 requires that a DAIS service be capable of pulling data from non-DAIS services such as ftp. In addition, the full realizations of scenarios 4, 5, and 6 require bulk update capabilities that are not included in the DAIS specification.

**Table 1. Summary of support for DAIS scenarios**

Scenario	Description	Support in DS	Support in ID
1	Synchronous query with result in return value	Yes	No
2	Asynchronous query with delivery to third party	No	Yes
3	Asynchronous query with independent pull delivery	Yes	Yes
4	Synchronous request for bulk update	Supported in database-specific way	No
5	Asynchronous request for bulk update with pull of update data from non-Grid service	No	No
6	Asynchronous update with independent delivery	No	Supported in database-specific way
7	Pipelining OGSA DS services. Query result data is pulled from producer DS by consumer DS for update	No	Yes
8	Pipelining OGSA DS services. Query result data is pushed from producer DS to consumer DS for update	No	Yes

This document describes the 8 scenarios and illustrates how they are supported by capabilities of the draft specification and the ID proposal.

## 1. Information Dissemination with DAIS services

This scenarios document illustrates how the Information Dissemination model, integrated with data services, will enhance the capabilities of those data services.

The Information Dissemination model allows users to share information between publishers and subscribers in a timely fashion. Publishers are considered to be the sources of information, while subscribers are considered to be the consumers of information. Subscribers and Consumers could be separate entities too. Information can be pushed or pulled from the publisher to the subscriber/consumer. The publisher can also alert subscribers/consumers of the existence of information.

In this model, publishers and subscribers can control how information is published, distributed, and consumed. The model supports efficient asynchronous distribution of information, so

publishers don't necessarily need to know of the target recipients and vice-versa through the use of staging areas.

There are three ways in which data movement could happen using the OGSA Data Service framework. They are Data Access, Notification and Information Dissemination. Data Access provides the synchronous data movement (RPC like request-response characteristics). It could also be extended to support the asynchronous data access (for example, by returning a handle to the result data as response and the result data could later be retrieved through the Data Access using the handle).

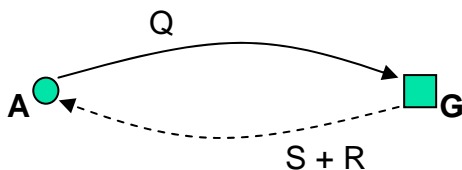
Information dissemination (ID) portType would be defined on an OGSA Data Service as a separate portType (at the same level as the DataAccess and DataManagement portTypes). The ID portType would have two sub portTypes IDProducer and IDConsumer.

For a detailed explanation of the ID portTypes and operations, please refer to the informational document on "Data Distribution in the Grid Environment" that was submitted / presented at GGF9. <http://www.cs.man.ac.uk/grid-db/papers/draft-ggf-dais-dist-ggf9-2.pdf>. A revised version of this ID informational document is being generated for GGF10.

## 2. DAIS Usage Scenarios

### 2.1 Synchronous query with result in return value

Scenario 1 illustrates the execution of a synchronous query. The client calls the data service and passes the query specification. The value returned by the service includes the data produced by the query.



Key to Symbols			
Data Flows			
	Call		Response
Actors			
	Non OGSA actor		DAIS service instance
A	Analyst	G	Grid data service
Data			
C	Create request	Q	Query execution request
GSH	Grid service handle	R	Result of query execution
S	Status information		

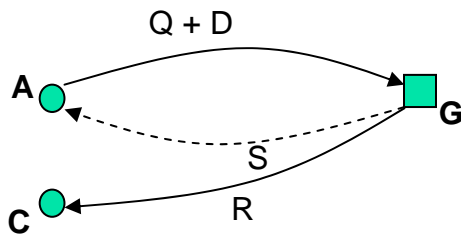
#### Scenario 1. Execution of database query with immediate return of results

Scenario 1 is supported directly by data services. Each realization of the data service porttypes includes a query execution porttype.

ID has no role to play here.

## 2.2. Asynchronous query with delivery to third party

Scenario 2 illustrates a request for asynchronous SQL query execution with delivery to a third-party 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. This scenario is supported by the ID model.



Key to Additional Symbols	
<b>Actors</b>	
C	Consumer client
<b>Data</b>	
D	Delivery specification

### Scenario 2. Asynchronous query and delivery to third party

Using ID, scenario 2, could be achieved as follows:

- The analyst locates the Data service representing the data of interest, by looking up a global registry that lists such Data services. The analyst receives in return the GSH of this Data Service (DSGSH).
- The analyst defines a subscription at DSGSH, expressing interest in the required data by specifying a SQL Query, the time at which this query has to be executed (3PM) and that it would result in a implicit publication called QueryPublication:

```

IDProducer::createSubscription([implicitname=QueryPublication, SQL Query,
                               scheduleat = 3PM], Analyst)
  returns SubsID.
  
```

The subscription generates the publication implicitly.

At 3PM, this data service would execute the query and maintain the result data in a queue/temporary table. This result data would be required for delivery purposes as mentioned through the propagation rules.

- The analyst specifies that the results of the subscription SubsID (the result of the query) be delivered to a 3<sup>rd</sup> party consumer. This is done by specifying the propagation rules. Through propagation, the analyst specifies the consumer location (URI), the time when the results have

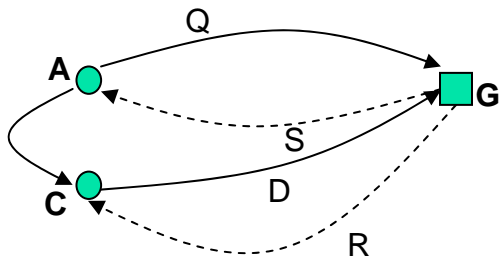
to be delivered (9PM), protocol to be used (SMTP), format in which the data needs to be delivered (WebRowSet), referring to the subscription by the subscription id (SubsID).

Note: 1. No need for consumption rules, in this case.

```
IDProducer::createPropagation(ConsumerURI, [subscription=SubsID, scheduleat = 9PM,
    protocol=SMTP, deliveryFormat=WebRowSet])
returns propagationId2.
```

At 9PM, the data service at DSGSH, would use the protocol (SMTP) mentioned for propagationId2 to send the result data to the consumer at consumerURI. The data service has information about where the result data for subscription SubsID is maintained.

### 2.3. Asynchronous query with independent pull delivery



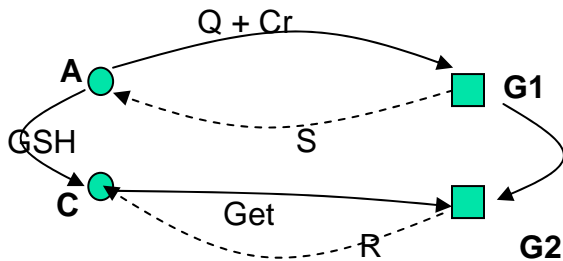
### Scenario 3. Asynchronous query with independent delivery

In scenario 3, the analyst A delivers a query to the data service and requests that the results be held for a subsequent delivery request. The data service responds with information that can be used to fetch the data. Analyst A then delivers that information to the consumer C which issues its own request for the data.

The realization of scenario 3 in the data services specification is shown in scenario 3a, in Section 2.3.1 below. The initial query is sent as part of a request to create a new service. Data service G1 creates a new dataset service G2 and passes the GSH of G2 back to A, which gives the GSH to C.

The ID proposal can be used to realize scenario 3 directly, as described below in Sections 2.3.2 and 2.3.3., and to realize scenario 3a, as described in Section 2.3.4.

#### 2.3.1 DS Support for Scenario 3



Key to Additional Symbols
Data

Cr	Request to create dataset
Get	Get request

### Scenario 3a. Asynchronous query with independent pull delivery and dataset service

Scenario 3a illustrates a request for asynchronous SQL query execution with delivery to a third-party consumer. The interaction between clients and Grid services can be as follows:

- a) (not shown): Client A, the analyst, gains access to data service G1. The data service may have been created specifically for client A through a `create` request to a data resource.
- b) A sends a request to the G1 factory porttype that includes the query
- c) G1 creates a new dataset service G2. G1 returns the GSH of G2 to A as part of the return value
- d) A sends the GSH of G2 to consumer C.
- e) C sends a `Get` request to G2 and receives the query result in the return value.

#### 2.3.2 ID support for scenario 3 Pull model

- a) The analyst locates the Data service representing the data of interest, by looking up a global registry that lists such Data services. The analyst receives in return the GSH of this Data Service (DSGSH).

- b) The analyst defines a subscription at DSGSH, expressing interest in the required data by specifying a SQL Query, the time at which this query has to be executed (3PM) and that it would result in a implicit publication called QueryPublication:

```
IDProducer::createSubscription([implicitname=QueryPublication, SQL Query,
    scheduleat = 3PM], Analyst)
returns SubsID.
```

The subscription generates the publication implicitly.

At 3PM, this data service would execute the query and maintain the result data in a queue/temporary table. This result data would be required for delivery purposes.

- c) The analyst would ask a 3<sup>rd</sup> party consumer to get the result data from the data service, by sending to consumer, the GSH of the data service (DSGSH) and the subscription information (SubsID).

- d) The 3<sup>rd</sup> party consumer will retrieve the results of the subscription SubsID (the result of the query), whenever it wants to. The consumer specifies the consumption rules (in what format the data would be consumed) and uses `getData()` method to retrieve the result data.

Note: 1. No need for propagation rules, in this case.

```
IDConsumer::createConsumption([subscription=SubsID,
    dataConsumptionFormat=WebRowSet], Consumer)
```

Returns consumptionId.

```
IDProducer::getData(consumptionId)
```

#### 2.3.3 ID support for scenario 3 Push model

Steps a, b and c are the same as in Section 2.3.2, above.

- d) The 3<sup>rd</sup> party consumer would specify a schedule to the data service (DSGSH) as to when the delivery of the results of the subscription SubsID (the result of the query) has to happen. Through propagation, the analyst specifies the consumer location (URI), the time when the

results have to be delivered (11PM), protocol to be used (FTP), format in which the data needs to be delivered (WebRowSet), referring to the subscription by the subscription id (SubsID).

Note: 1. No need for consumption rules, in this case.

```
IDProducer::createPropagation(ConsumerURI, [subscription=SubsID,
    scheduleat = 11PM, protocol=FTP, deliveryFormat=WebRowSet])
    returns propagationId.
```

At 11PM, the data service at DSGSH, would use the protocol mentioned for propagationId to send the result data to the consumer at consumerURI. The data service has information about where the result data for subscription SubsID is maintained.

### 2.3.4 ID support for scenario 3a (pull model)

Steps a, b, and d are the same as those of Section 2.3.2.

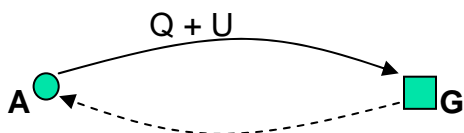
In step b, at 3PM, the data service (G1) would execute the query and create a new data service (G2) and populate this new data service using the query result.

Step c would change as follows:

- c) The analyst would enquire with G1 data service to get the GSH of the new data service G2. (The other alternative is that G1 would notify the analyst (along with the GSH of G2) after the new data service G2 is created and populated). The analyst would then ask a 3<sup>rd</sup> party consumer to get the result data from the data service G2, by sending to the consumer, the GSH of the data service G2 (G2GSH), the subscription information (SubsID).

Step d of Section 2.3.2 is then performed on the GSH of G2 to retrieve the result data from G2.

## 2.4. Synchronous request for bulk update

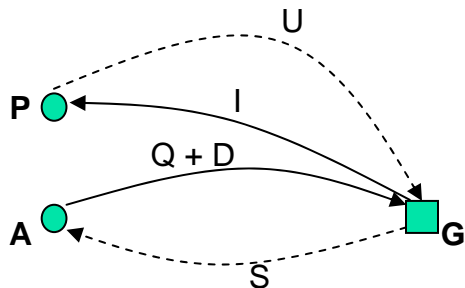


### Scenario 4. Synchronous request for bulk update

The intention of scenario 4 is that A sends a specification of an update, together with update values, to G. The support for update in the DAIS draft specification is for update through database-specific mechanisms. A relational database service can perform updates by executing SQL update, insert or delete statements. Similarly, updates to XML database services rely on the syntax and semantics of Xupdate.

ID does not play a role in this synchronous operation.

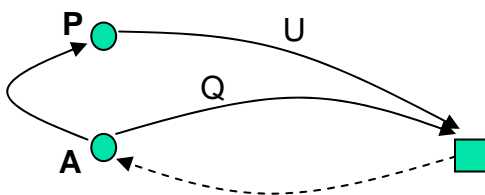
## 2.5. Asynchronous request for bulk update with pull of update data from non-Grid service



### Scenario 5. Asynchronous request for bulk update with third party delivery of update data

Scenario 5 is not currently supported by the draft DAIS specifications. ID does not play a role in this scenario, where data is pulled from a non-Grid service.

## 2.6 Asynchronous update with independent delivery



### Scenario 6. Asynchronous update operation. Update data would be pushed by the non-service producer to the data service

The ID proposal supports database-specific updates, using SQL statements and Xupdate. The ID proposal will include support for bulk updates when they appear in the DAIS specification.

The ID proposal includes support for the database-specific update operations as follows.

- a) The analyst locates the Data service where the update operation has to be performed, by looking up a global registry that lists such Data services. The analyst receives in return the GSH of this Data Service (DSGSH).
- b) The analyst defines a subscription at this data service DSGSH, expressing interest in executing an update statement, the data for which would come from an external source (from a non-service producer, not mentioned in the subscription).  
 IDProducer::createSubscription([Update Query, <ExecuteOnDataArrival>], Analyst)  
 returns SubsID.
- c) The analyst defines consumption rules at this data service (DSGSH), for consuming the update data for the subscription SubsID, specifying the format in which the data would be consumed (WebRowSet).

```
IDConsumer::createConsumption([subscription=SubsID,
                               dataConsumptionFormat=WebRowSet], Analyst)
Returns consumptionId.
```

The analyst would convey this consumptionId information to the producer.

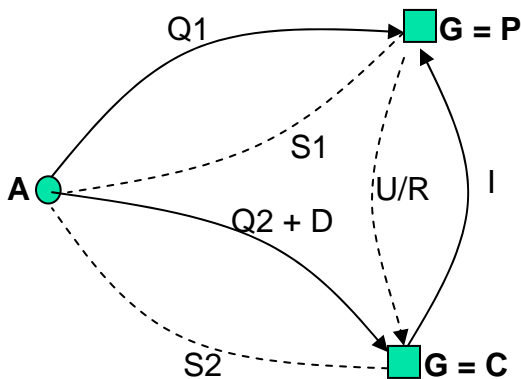
- d) The non-service producer would push the update data to the data service using the deliverData() operation.

```
IDConsumer::deliverData(consumptionId, null, <Data>)
```

Upon receipt of this update data, the data service would perform the update operation.

The result of the update operation is returned to the analyst, through notification (or) could be queried (using the status SDE) by the analyst at the consumer data service.

## 2.7 Pipelining OGSA DS services. Query result data is pulled from producer DS by consumer DS for update



### Scenario 7: Pipelining OGSA DS services. Query result data is pulled from producer DS by consumer DS for update

The ID proposal supports scenario 7 as follows.

- The analyst looks up a global registry to locate the producer and consumer data services. The producer data service contains the data of interest for the update operation that needs to be performed at the consumer data service. The analyst receives in return the GSH of these Data Services (PDSGSH, CDSGSH).
- The analyst defines a subscription at PDSGSH, expressing interest in the required data by specifying a SQL Query, the time at which this query has to be executed (3PM) and that it would result in a implicit publication called QueryPublication:

```
IDProducer::createSubscription([implicitname=QueryPublication, SQL Query,
                                scheduleat = 3PM], Analyst)
returns PSubsID.
```

The subscription generates the publication implicitly.

At 3PM, this data service would execute the query and maintain the result data in a queue/temporary table. This result data would be required for delivery purposes.

- c) The analyst defines consumption rules at the producer data service PDSGSH, for the result of subscription PSubsID, specifying the format in which the data would be consumed (WebRowSet).

```
IDConsumer::createConsumption([subscription=PSubsID,
                                dataConsumptionFormat=WebRowSet], Analyst)
Returns consumptionId.
```

- d) The analyst defines a subscription at the consumer data service CDSGSH, expressing interest in executing an update statement, using data that would need to be retrieved from the producer data service PDSGSH.

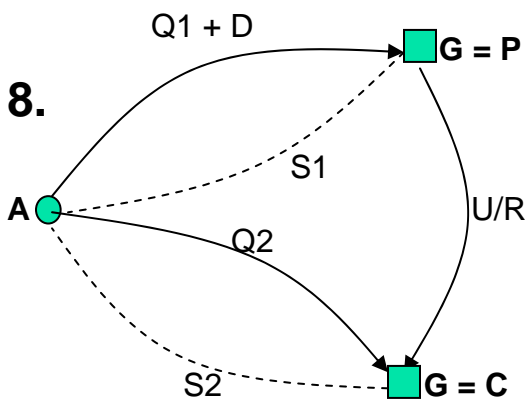
```
IDProducer::createSubscription([Update Query, dataFromDS=PDSGSH,
                                consumptionIdentifier=consumptionId, scheduledat=5PM], Analyst)
returns CSubsID.
```

At 5PM, this consumer data service would execute the update query by retrieving the data for the update from the producer data service (PDSGSH), using the `getData()` operation performed on the producer data service.

```
IDProducer::getData(consumptionId).
```

The result of the update operation is returned to the analyst, through notification (or) could be queried (using the status SDE) by the analyst at the consumer data service.

## 2.8 Pipelining OGSA DS services. Query result data is pushed from producer DS to consumer DS for update



### Scenario 8: Pipelining OGSA DS services. Query result data is pushed from producer DS to consumer DS for update

The ID proposal supports scenario 8 as follows.

- a) The analyst looks up a global registry to locate the producer and consumer data services. The producer data service contains the data of interest for the update operation that needs to be performed at the consumer data service. The analyst receives in return the GSH of these Data Services (PDSGSH, CDSGSH).
- b) The analyst defines a subscription at PDSGSH, expressing interest in the required data by specifying a SQL Query, the time at which this query has to be executed (3PM) and that it would result in a implicit publication called QueryPublication:

```
IDProducer::createSubscription([implicitname=QueryPublication, SQL Query,
    scheduleat = 3PM], Analyst)
    returns PSubsID.
```

The subscription generates the publication implicitly.

At 3PM, this data service would execute the query and maintain the result data in a queue/temporary table. This result data would be required for delivery as mentioned through the propagation rules.

- c) The analyst specifies that the results of the subscription PSubsID (the result of the query) be delivered to the consumer data service CDSGSH. This is done by specifying the propagation rules. Through propagation, the analyst specifies the consumer data service GSH (CDSGSH), the time when the results have to be delivered (4PM), protocol to be used (deliverData), format in which the data needs to be delivered (WebRowSet), referring to the subscription by the subscription id (PSubsID).

```
IDProducer::createPropagation(CDSGSH, [subscription=PSubsID,
    scheduleat = 4PM, protocol=deliverData, deliveryFormat=WebRowSet])
    returns propagationId.
```

At 4PM, the producer data service PDSGSH, would use the deliverData() method on the consumer data service, to send the result data to the consumer data service CDSGSH. The producer data service has information about where the result data for subscription PSubsID is maintained.

```
IDConsumer::deliverData(propagationId, null, <Data>)
```

- d) The analyst specifies the consumption rules at the consumer data service (CDSGSH), for the result of the subscription PSubsId that is propagated through propagationId, specifying the format in which the data would be consumed (WebRowSet), the location for the result data (tempTable).

```
IDConsumer::createConsumption([propagation=propagationId,
    dataConsumptionFormat=WebRowSet, location=tempTable], Analyst)
    Returns consumptionId.
```

- e) The analyst defines a subscription at the consumer data service CDSGSH, expressing interest in executing an update statement, using data that comes from the producer data service PDSGSH for the propagation propagationId (as defined by the consumption rules for this propagation).

```
IDProducer::createSubscription([Update Query, dataFromDS=PDSGSH,
```

consumptionIdentifier=consumptionId, scheduledat=5PM], Analyst)  
returns CSubsID.

At 5PM, this consumer data service would execute the update query using the data that came from the producer data service (PDSGSH).

The result of the update operation is returned to the analyst, through notification (or) could be queried (using the status SDE) by the analyst at the consumer data service.