

Sample Mappings from DAIS Conceptual Model to Service Instances

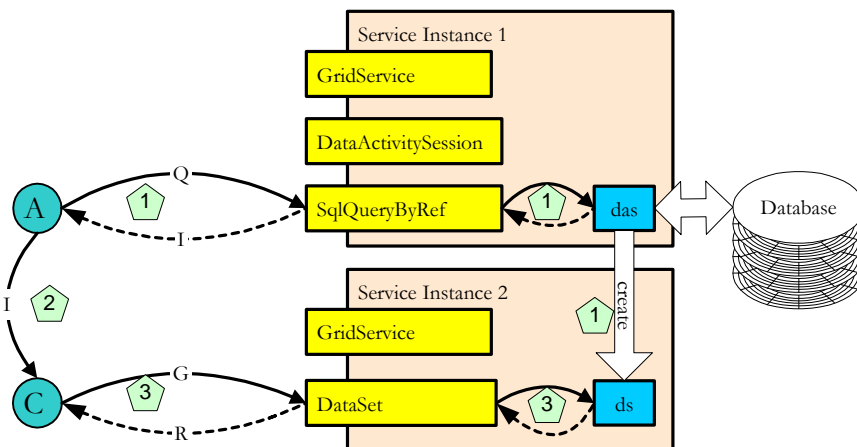
Greg Riccardi

riccardi@cs.fsu.edu

The DAIS specification for GGF 8 (<http://www.cs.man.ac.uk/grid-db/papers/draft-ggf-dais-gdss-ggf8.pdf>) presents a model for database access that defines the major conceptual objects and their operations. The specification defines port types but does not mandate how service instances should embody those port types. In contrast, the DAIS/GGF 7 specification (http://www.cs.man.ac.uk/grid-db/papers/DAIS_GGF7StatementSpec.pdf) and the OGSA-DAI implementation (<http://www.ogsa-dai.org.uk>) give service-oriented descriptions.

This document illustrates ways in which the port types of the GGF8 specification may be represented by service instances that support particular port types. Several scenarios for client-service interaction are described in terms of port types, operations and service instances.

1. Representing Each Conceptual Object as a Separate Service Instance



Key to Symbols			
Data Flows			
	Call		Response
			Create dataset
Actors			
	Non OGSA process		DAIS conceptual object
	Grid service port		Grid service instance
A	Analyst	das	Grid activity session, bound to relational edr with an SqlQueryByRef port type
C	Consumer	ds	Grid dataset, bound to relational eds
Operations			
Q	sqlQueryByRefAsync	G	get
Data			

I	Dataset identifier (URI)	R	Result of query execution
Comments			
1	Indication of order of execution		

Figure 1. Representing conceptual objects as service instances

Figure 1 illustrates the interaction between two clients and two Grid services. The interaction follows these steps:

Step 0 (not shown): Client A, the analyst, gains access to the Grid Service Handle (GSH) of data activity session `das` which provides interactions with the appropriate external data resource. This GSH maps to the GSR of a service instance that implements the `SqlQueryByRef` port type and provides access to `das`.

The service instance may have been created specifically for client A as a result of a `create` request to a data resource. Alternatively, the service instance may support access to many data activity sessions and have a lifetime beyond the lifetime of `das`.

Step 1: Client A, the analyst, sends a request to `das` by calling the `sqlQueryByRefAsynch` operation of the `SqlQueryByRef` port of service instance 1.

`<input>select * from table</input>`

In response, the service instance applies the operation to `das`, which causes service instance 2 to be created to contain dataset `ds`. The query is sent to the database and the results used to populate `ds`. The response to client A contains the grid service handle for the new dataset service instance.

Step 2: Client A sends the grid service handle of `ds` to client C, the consumer. This interaction is not necessarily conducted as a Grid service activity.

Step 3: Client C sends a request to the `get` operation of the `DataSet` port of service instance 2. Because the query is executing asynchronously, `ds` may block this request until the dataset is fully populated. The response to this request contains the complete result of the query execution represented as an XML document.

2. Representing Multiple Conceptual Objects as a Single Service Instance

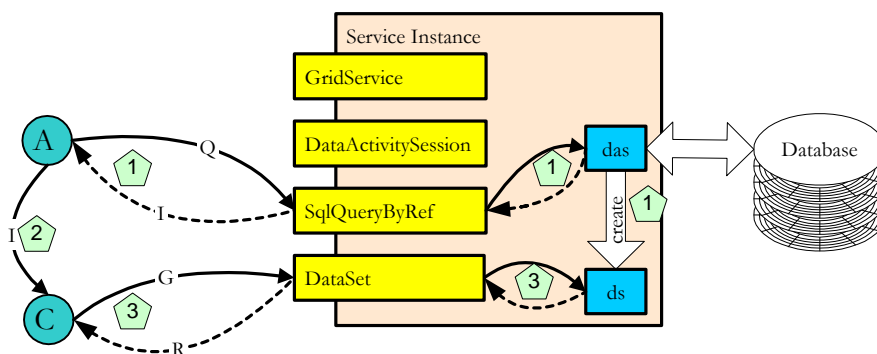


Figure 2. Representing multiple conceptual objects as a single service instance

In this case, the new dataset `ds` is managed by the same service instance. Thus, the grid service handle for `ds` will refer to the same grid service as the grid service handle for `das`.

3. Moving datasets

In this example, a dataset is created by a client and moved to a new location. As with all Grid services, a location service is employed to determine the current location of a dataset. The location service maps a Grid service handle to a Grid service resource. OSGI specifies that mappings from grid service handles (GSH) to grid service resources (GSR) are managed by such a locator service. Figure 3 shows an example of how service instances might interact with an external, Grid-enabled movement service.

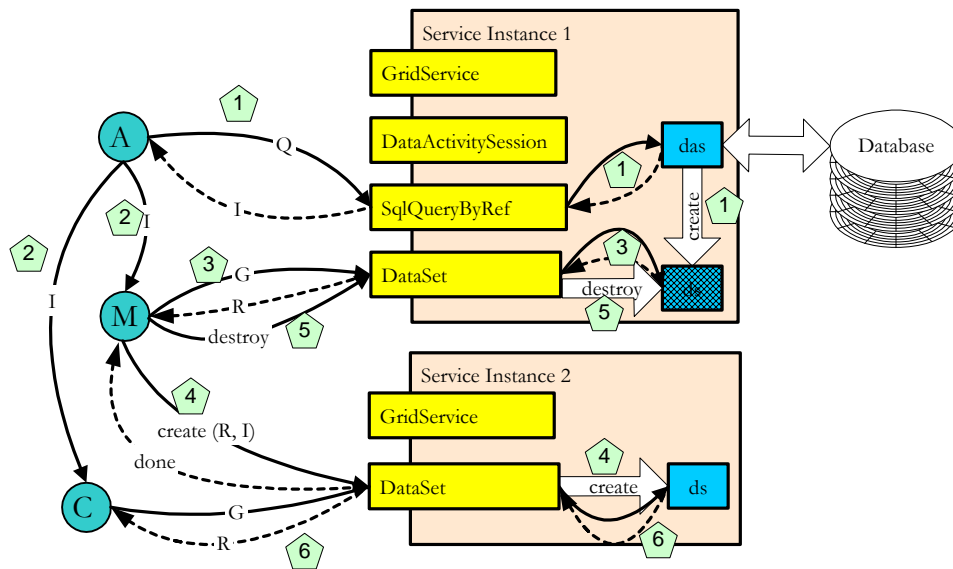


Figure 3. Movement of dataset by movement service M. Order of operations denoted by numbered pentagons

The interaction follows these steps:

- Step 1: Client A, the analyst, sends a request to the `sqlQueryByRefASynch` operation of the `SqlQueryByRef` port of the service instance 1, just as in Figure 1. Dataset `ds` is created by `das` in response to this request and the GSH of `ds` is returned to A.
- Step 2: Client A sends the GSH of `ds` to consumer C and movement service M.
- Step 3: Movement service M sends a get operation on `ds` to the `DataSet` port of service instance 1. The value of `ds` is returned to M.
- Step 4: M sends the value of `ds` and its GSH to the `DataSet` port of service instance 2 as a request to create a copy of `ds`. Service instance 2 creates a copy of `ds` and informs the locator service that an instance of `ds` has been created.
- Step 5: M sends a destroy request to the copy of `ds` in service instance 1. That copy is destroyed and the locator service is informed that service instance 1 no longer has a copy of `ds`. The cross hatching on `ds` in service instance 1 indicates that the object has been destroyed.
- Step 6: C uses the GSH of the `ds` to issue a get request. C is directed to service instance 2 by the locator service.

4. Service Containers and Conceptual Objects

In this example, a single database is used by multiple clients with a separate data activity session for each client. In addition, the two service instances support different collections of port types. One service instance implements the data activity session port types and another service instance implements the DataSet port type. Thus the creation of a dataset by a data activity session requires an interaction between these two service instances. This example does not imply that the two service instances are on different machines, or even different processes.

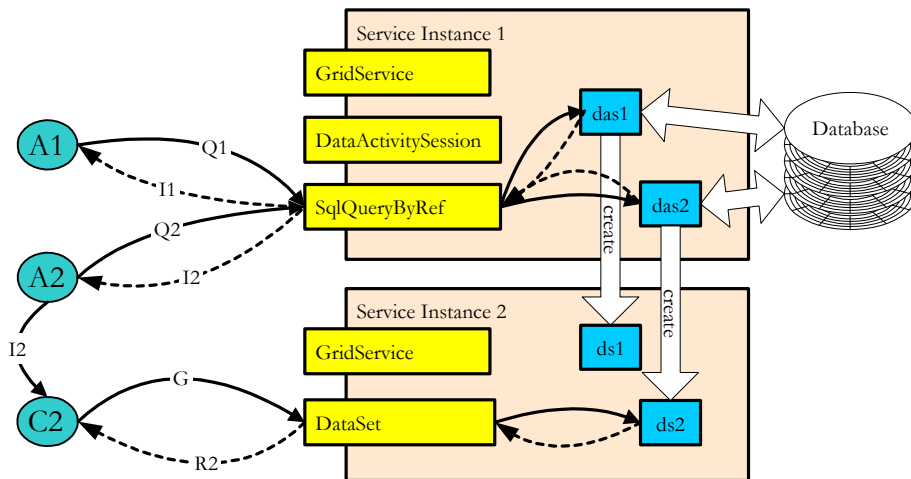


Figure 4. Scenario with separate service instances for data activity sessions and datasets

The example of Figure 2 shows 2 analyst clients interacting with separate data activity sessions that are located in a single service instance. Both of the clients submit queries and 2 datasets are created. The datasets are located in a separate service instance which has been configured to support interaction with datasets.

The purpose of this example is to show how a single service may support many conceptual objects and that different service instances may implement different port types. Service instance 1 has been configured to support data activity sessions, but not datasets. Similarly service instance 2 supports many datasets.