

DAIS Data Service Interactions and the SkyQuery Portal

Greg Riccardi

Department of Computer Science, Florida State University

riccardi@cs.fsu.edu, <http://www.nesc.ac.uk/~greg>

- 1. Using access recipes for service interactions2
 - 1.1 Getting data from dr22
 - 1.2 Specifying processing in dr13
- 2. Using retained state for service interactions5
 - 2.1 Requesting that das2 create and retain dataset ds26
 - 2.2 Requesting that das1 fetch ds2, perform its query, and retain dataset ds17

This document demonstrates how the DAIS conceptual model can support interacting data services, third party data delivery, and retained state in data activity sessions. In addition, the document uses an XML document schema that expresses the capabilities of the conceptual model.

The capabilities of DAIS services are explained in the context of the SkyQuery federated query processing application [<http://www.skyquery.org>]. The SkyQuery portal allows astronomers to federate information stored in several databases that contain observations of astronomical objects. The portal accepts a query from a user and coordinates the execution of queries on one or more databases. In essence, the user requests the selection of data from a particular area of the sky from the databases and the linking of the individual objects from the various databases according to their astronomical locations.

Figure 1 illustrates the SkyQuery processing for 2 databases. The portal creates a pipeline of Web services (called SkyNodes), each with access to one database and issues a specific request to the last service in the pipeline. The service on the left requests data from the next service, creates a table from the data, performs a join operation with one of its own tables, and returns the result table to its caller. The service at the start of the data pipeline (shown on the right) executes a select query and returns the result to its caller.

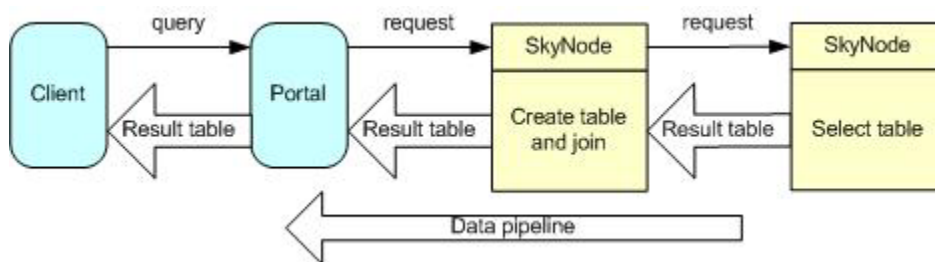


Figure 1. Interaction between SkyNode services

This document describes two strategies for implementing the SkyQuery processing using Grid Data Services. The first strategy gives one service a recipe for requesting data from another service. The second strategy creates dataset objects that are retained as state by some service.

1. Using access recipes for service interactions

The SkyQuery service shown in Figure 1 has a straightforward representation as the interactions of DAIS services, as shown in Figure 2. Each SkyNode service of Figure 1 is represented as a data resource in the DAIS implementation. The client creates a query and sends it to the data service manager—the service that coordinates the query processing. The data service manager creates a request document and sends it to the performImmediate method of dr1, which in turn creates and sends a request to the performImmediate method of dr2. Service dr2 executes the appropriate query and sends the result dataset to dr1. Service dr1 uses the returned data to create a table and then performs its join. The result data from dr1 is sent to an ftp server provided by the client.

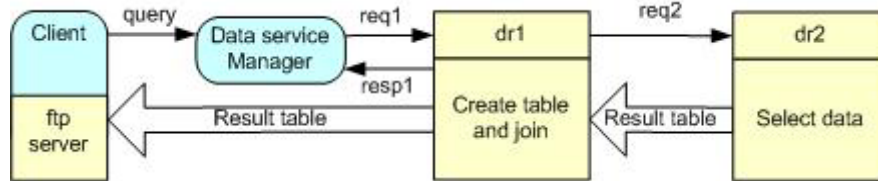


Figure 2 DAIS service interactions for SkyQuery

The basic structure of the requests in this example come from the requests show in “Model Examples: Simple Client-Service Interactions” (<http://www.nesc.ac.uk/~greg/modelexample.pdf>). That document includes details on how to specify the select and create table queries required herein.

This example features the nesting of requests. Thus the request req1 contains req2. Service dr1 must extract req2 and call the performImmediate method of dr2. The result of that call will be used to populate the new table.

1.1 Getting data from dr2

The document req2 that must be sent to dr2 is shown in Figure 3. It includes the spatial query of line 7. The dataset element can be considered a recipe for extracting the required information from dr2. This recipe will be embedded in req1.

```

1    <DataSet>
2      <element>
3        <name>req1</element>
4        <type> Request.Query.SqlQuery </type>
5        <value>
6          <SqlQuery>
7            select * from photoObject o where area(1.85,-.5,4.5)
8          </SqlQuery>
9        </value>
10     </element>
11  </DataSet>
  
```

Figure 3 Request document req2

The response document, shown in Figure 4, includes the element named req1.resp1 that contains the data selected by the query. This data will be extracted from the result and used to create and populate a table in dr1.

Data Service Interaction: The Sky Query Processing Example

```
1 <DataSet>
2   <element>
3     <name>req1</name>
4     <type>Response.Query.SqlQuery</type>
5     <value>
6       <request><SqlQuery>
7         select * from photoObject o where area(1.85,-.5,4.5)
8       </SqlQuery></request>
9       <Result name="req1.resp1"/>
10      <status>complete</status>
11    </value>
12  </element>
13  <element>
14    <name>req1.resp1</name><!-- contains the result table -->
15    <type>
16      <xs:schema>--table schema here--</xs:schema>
17    </type>
18    <value><Table>--table rows here--</Table></value>
19  </element>
20 </DataSet>
```

Figure 4 Response document for req2

1.2 Specifying processing in dr1

Request req1 must instruct dr1 to create a temporary table using req1 as its source, perform the SkyNode spatial join query, deliver its result, and drop the temporary table. As shown in Figure 5, the request document has 3 request elements beginning at lines 2, 30, and 56. It also contains 2 data delivery recipe elements, at lines 13 and 43.

```
1 <DataSet>
2   <element>
3     <name>CreateTableRequest</element>
4     <type> Request.Update.SqlCreateTable </type>
5     <value>
6       <CreateTable name="temp1">
7         <Input>
8           <DataAccessRecipe element="recipe1"/>
9         </Input>
10      </CreateTable>
11    </value>
12  </element>
13  <element>
14    <name>recipe1</name>
15    <type>DataAccessRecipe</type>
16    <value>
17      <DataAccessRecipe>
18        <GridService>
19          <gsh>dr2</gsh>
20          <type>DAIS.dataresource</type>
21          <method>performImmediate</method>
```

Data Service Interaction: The Sky Query Processing Example

```

22         </GridService>
23         <datasetElement name="req1.resp1"/>
24         <dataset>
25             <!-- the dataset of Figure 4 goes here -->
26         </dataset>
27     </DataAccessRecipe>
28 </value>
29 </element>
30 <element>
31     <name>QueryRequest</element>
32     <type>Request.Query.SqlQuery</type>
33     <value>
34         <SqlQuery>
35             select * from temp1 t, photoObject o
36                 where area(1.85,-.5,4.5) and xmatch(t,o)
37         <output>
38             <DataDeliveryRecipe element="recipe2"/>
39         </output>
40         </SqlQuery>
41     </value>
42 </element>
43 <element>
44     <name>recipe2</name>
45     <type>DataDeliveryRecipe</type>
46     <value>
47         <DataDeliveryRecipe>
48             <ftp>
49                 <server>ftp.nesc.ac.uk</server>
50                 <file>/pub/skyquery1.xml</file>
51                 <usr>ftp</usr>
52             </ftp>
53         </DataDeliveryRecipe>
54     </value>
55 </element>
56 <element>
57     <name>dropTableRequest</name>
58     <type>Request.update.SqlDropTable</type>
59     <value>
60         <DropTable name="temp1"/>
61     </value>
62 </element>
63 </DataSet>

```

Figure 5 Request req1 for dr1.performImmediate

The create table tag in lines 6-10 specifies that its input data comes from the DataAccessRecipe tag contained in the dataset element named “recipe1” which can be found in lines 13-29. The data access recipe specifies the service to be called (lines 18-22), the name of the dataset element of the call's result that provides the data (line 23), and the dataset that must be sent to the service (lines 24-26 are placeholders for the dataset).

The processing of the create table request requires that dr1 decode the data access recipe, perform the service call, extract the data values from the result dataset, and create and populate the new table.

Once the create table request has been executed, dr1 can process the select query request of lines 30-42. The SqlQuery tag of lines 34-40 includes an output tag that specifies a data delivery recipe. The results of the query will not be returned to the data service manager in the response document.

The data delivery recipe of lines 43-55 specifies that an ftp protocol is to be used. The ftp tag specifies the name of the server (line 49), the target file name (line 50), and the user id (line 51). The data to be transferred comes from the request that uses the recipe. In this case, the reference to the tag is in line 38 and the data to be transferred is the result of the execution of the query in lines 35 and 36.

Once the query execution and data delivery is complete, the drop table request (lines 56-62) is executed to drop the temporary table. {I note that not all data resources will support controlling execution according to dependencies between requests.}

Thus, the DAIS service model easily supports the configuration and execution of pipelined data processing. The pipelines could be hierarchical as well as linear. The example shows only 2 interacting services. However, additional services could be added by embedding req1 into itself with suitable modifications.

2. Using retained state for service interactions

A DAIS data activity session can create and retain state in the form of named datasets. A request element can specify that its results should be retained beyond the execution of the request dataset. Access to retained datasets may be through the data activity session or through some other service.

The SkyQuery federated query processing can be implemented by creating an interaction between data activity sessions, as illustrated in Figure 6. The Sky Query data service manager is a client of the services, In response to a query request from a client, the data service manager interacts with the data activity sessions to create the SkyQuery processing. It first requests das2 to execute a spatial query and create a dataset with the results. The response to this asynchronous request includes a reference to the new dataset (ds2). The reference will consist of the GSH of the service, the name of the dataset object within the service and the name of the element of the dataset that contains the query result.

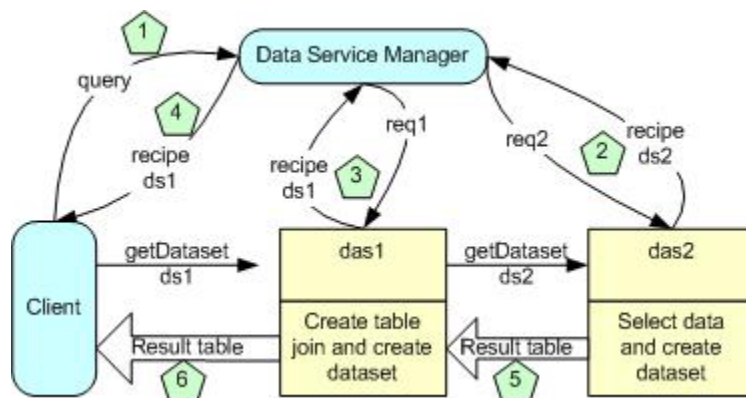


Figure 6 SkyQuery as the interaction between data activity sessions

The data service manager extracts the dataset reference from the response and sends a request to `das1` with 3 request elements. First, `das1` must create a temporary table and populate it with the values from the dataset (`ds2`) produced by `das2`. Then `das1` will perform a crossmatch join query with the temporary table and one of its tables. The results of the join become a new dataset (`ds1`). The reference to the new dataset is returned in the response document. Finally, the data service manager returns the recipe for accessing dataset `ds1` to the client.

The client is now free to call `das1` and ask for dataset `ds1`. The client's request for the contents of `ds1` will wait for the data to be produced.

2.1 Requesting that `das2` create and retain dataset `ds2`

The first request document produced by the data service manager is shown in Figure 7. The single request element specifies that the result of the select statement of line 7 is to be saved in a dataset.

```

1    <DataSet>
2      <element>
3        <name>req2</element>
4        <type> Request.Query.SqlQuery </type>
5        <value>
6          <SqlQuery>
7            select * from photoObject o where area(1.85,-.5,4.5)
8            <toDataSet>
9              <CreateDataSet element="ds2" readOnce/>
10           </toDataSet>
11          </SqlQuery>
12        </value>
13      </element>
14    </DataSet>
```

Figure 7 Request to `das2.performAsynchronous`

Lines 8-10 specify that the result of the request is to be retained as an element named `resp1` in a new dataset. The tag `toDataSet` (line 8) specifies that the result is to be retained as state. The tag `CreateDataSet` (line 9) specifies that a new dataset is to be created to hold the elements. (An alternative to `CreateDataSet` is `AddElement` which specifies that the result is to be added to an existing dataset.) It is the responsibility of `das2` to name the dataset and arrange for the dataset to be held by a dataset service.

```

1    <DataSet>
2      <element>
3        <name>QueryRequest</element>
4        <type>Response.Query.SqlQuery</type>
5        <value><!-- status and other response to query -->
6          <request>
7            <SqlQuery>
8              select * from photoObject o ...
9            <toDataSet>
10             <CreateDataSet element="ds2" readOnce>
11            </toDataSet>
```

```

12         </SqlQuery>
13     </request>
14     <Result name="QueryRequest.resp1">
15         <DataAccessRecipe>
16             <GridService>
17                 <gsh>das2</gsh>
18                 <type>DAIS.dataactivitysession</type>
19                 <method>performImmediate</method>
20             </GridService>
21             <dataSetElement name="getdataset.resp1"/>
22             <dataset>
23                 <element>
24                     <name>getdataset</name>
25                     <type>Request.DataSet.getDataSetElement</type>
26                     <value>
27                         <getDataSetElement name="ds2" element="resp1">
28                             </value>
29                     </element>
30             </dataset>
31         </DataAccessRecipe>
32     </Result>
33     <Status>in progress</Status>
34 </value>
35 </element>
36 </DataSet>

```

Figure 8 Response from das2

Lines 15-31 contain a data access recipe for the dataset that was created in response to the request. According to that recipe, das2 is managing the dataset and will deliver it in response to the `getDataSetElement` request (lines 23–29) of the request dataset of lines 21–30.

One alternative to das2 managing dataset ds2 is to have das2 create a new dataset service and return the GSH of that service as the `service` tag in the response document. A change in GSH for the dataset ds2 will have no impact on the data service manager or other services. The data service manager simply copies the dataset recipe from the response document into the request document for das1. Service das1 will fetch the dataset from whichever service is referenced in the recipe.

A dataset service is a service that has a `performImmediate` method that accepts request datasets and implements the `getDataSetElement` activity.

2.2 Requesting that das1 fetch ds2, perform its query, and retain dataset ds1

Figure 9 shows the request dataset that is sent to das1. Lines 2–12 are the create table request and line 8 is a placeholder for the data activity recipe created by das2 (lines 15–31 of Figure 8). As in Figure 7, the request includes a `toDataSet` tag (lines 20–22) to specify that the result of the query is to be retained for later delivery. The remainder of the request document is the same as in Figure 5.

```

1     <DataSet>
2         <element>

```

Data Service Interaction: The Sky Query Processing Example

```
3      <name>CreateTableRequest</element>
4      <type> Request.Update.SqlCreateTable </type>
5      <value>
6          <CreateTable name="temp1">
7              <Input>
8                  <!-- recipe of lines 15-31 of Figure 8 -->
9              </Input>
10         </CreateTable>
11     </value>
12 </element>
13 <element>
14     <name>req3</element>
15     <type>Request.Query.SqlQuery</type>
16     <value>
17         <SqlQuery>
18             select * from temp1 t, photoObject o
19                 where area(1.85,-.5,4.5) and crossmatch(t,o)
20         <toDataSet>
21             <CreateDataSet element="ds1" readOnce>
22         </toDataSet>
23     </SqlQuery>
24 </value>
25 </element>
26 <element>
27     <type>Request.update.SqlDropTable</type>
28     <value>
29         <DropTable name="temp1"/>
30     </value>
31 </element>
32 </DataSet>
```

Figure 9 Request to das1.performAsynchronous

The response, shown in Fig 10, includes the data access recipe (lines 24–40) that can be used by the Data Service Manager to fetch the results of the federated query.

```
1      <DataSet>
2      <element>
3          <name>CreateTable</element>
4          <type>Response.Update.SqlCreateTable </type>
5          <value>
6              <request>
7                  <CreateTable>
8                      --repeat of create table request
9                  </CreateTable>
10             </request>
11             <RowCount>15</RowCount>
12         </value>
13 </element>
14 <element>
15     <name>Query</element>
16     <type>Response.Query.SqlQuery</type>
```

Data Service Interaction: The Sky Query Processing Example

```
17     <value>
18         <request>
19             <SqlQuery>
20                 --repeat of query request
21             </SqlQuery>
22         </request>
23         <Result name="req1.resp1">
24             <DataAccessRecipe>
25                 <GridService>
26                     <gsh>das1</gsh>
27                     <type>DAIS.dataactivitysession</type>
28                     <method>performImmediate</method>
29                 </GridService>
30                 <dataSetElement name="getdataset.resp1"/>
31                 <dataset>
32                     <element>
33                         <name>getdataset</name>
34                         <type>Request.DataSet.getDataSetElement</type>
35                         <value>
36                             <getDataSetElement name="ds1" element="resp1">
37                                 </value>
38                             </element>
39                         </dataset>
40                     </DataAccessRecipe>
41                 </Result>
42                 <Status>in progress</Status>
43             </value>
44         </element>
45         <element>
46             <name>DropTable</name>
47             <type>Response.update.SqlDropTable</type>
48             <value>
49                 <DropTable name="temp1"/>
50                 <Result>deferred</Result>
51             </value>
52         </element>
53     </DataSet>
```

Figure 10 Response from das1 to the request of Fig. 9