GWD-R
Category: INFORMATIONAL
GGF DAI Working Group

Editors:
Vijay Dialani, University of Southampton
Indepal Narang, IBM

Version 1.0
19th September 2003

Grid Data Services – Transformation Service

Status of This Memo

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

Abstract

Data accessed from/by DAIS services may need to be transformed for either optimizing the system performance or to provide the data in an application acceptable format. This document proposes a transformation service. It allows integration of user-defined function types and supports multiple result sets. We propose a service interface that can either be implemented as a separate service or as a port type supported by DAIS services.

Note: The document is being submitted for discussions at GGF.

Contents

GWD-R
Category: INFORMATIONAL
GGF DAI Working Group

Editors:
Vijay Dialani, University of Southampton
Indepal Narang, IBM

Version 1.0
19th September 2003

1. Transformation Service

In many cases data accessed from one or more data resources needs to undergo transformations for it to be in application usable state [Ref 1] or for performance optimizations [Ref 3]. This may be for integrating data from heterogeneous resources, data mappings, domain specific functional processing, compressing data for delivery on the wire etc. It is envisaged that the following types of data transformations would take place and how GDS can support them.

1. Schema mappings and data transformations for federation of data resources. This type of transformation is not in scope for the current version of the DAIS spec. It is not discussed further.

2. Transformations performed as part of a DBMS request. Broadly, a DBMS supports the following types of functions – base (or builtin) functions and user defined functions (UDF's) for extensibility. There are myriad of SQL builtin functions supported by relational DBMS's. The builtin functions would be considered part of the query language support. For UDF's the GDS spec on Stored Procedures and Functions [Ref 2] provides means for introspection of signatures of the function. This would assist the application to invoke the desired function as part of a DBMS request (e.g., a SQL request).

3. Transformations of query results or stored procedure results thru Pipelining We classify these as follows:
   a) defining an activity within a Perform request. (See section "Alternative to Pipelining thru Perform Request" if Perform request goes away based on DAIS-wg f2f discussions in Aug'03). Appendix 1 gives an example of XSLT transformations of query results. Appendix 2 describes how transformations of multiple result-sets returned by a stored procedure can be handled. It also describes how XSLT transformations described in Appendix 1 can be extended for multiple result-sets.

   b) Filtering of results during the delivery activity [Ref 2]

   c) service composition of a GDS request and a transformation grid service (typically domain specific), as in a workflow. This is outside the scope of the DAIS spec.

2. Alternative to Pipelining through Perform Request

Based on DAIS-wg f2f meeting in Aug '03, Perform request may go away. In that case the transformations would be part of the workflow where the steps for transformations of query results (or multiple result-sets) could be as follows.
- query result-set is a Dataset. This Dataset becomes input for the transformation.
- transformation service is invoked as Transform (inputDataset, transformation, outputDataset).
- the ouputDataset may be accessed as final output or may become input for further transformations.

Multiple result sets can either be represented as a single dataset or as a collection of datasets. Much depends on the structure of a dataset. DAIS is yet to decide on the structure of the dataset. Assuming that each dataset can represent multiple result sets the transformation service is invoked as

Transform (inputDataset, transformation, outputDataset),

However, if the input is a collection of datasets the output will also be a collection of datasets with a one-to-one correspondence between input and output collection. In this case the modified transformation service will be of the form

Transform (inputDataset[], transformation, outputDataset[])

3. Appendix 1

(Example of XSLT Transformation of query results as an Activity within a Perform request -- provided by Neil Chue Hong, Norman Paton)

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- (c) International Business Machines Corporation, 2002, 2003. (c) University of Edinburgh
2002, 2003.-->
<!-- See OGSA-DAI-Licence.txt for licencing information.-->

<xsd:schema targetNamespace="http://ogsadai.org.uk/P2R2/schemas/gds"
xmlns:tns="http://ogsadai.org.uk/P2R2/schemas/gds"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
            xmlns:gdstypes="http://ogsadai.org.uk/P2R2/schemas/gds"
            elementFormDefault="qualified">

 <xsd:include schemaLocation="../../types/grid_data_service_types.xsd"/>

 <xsd:complexType name="XSLTransformType">
 <xsd:complexContent>
   <xsd:extension base="gdstypes:ActivityType">
   <xsd:sequence>

      <xsd:element name="inputXML" minOccurs="1" maxOccurs="1">
      <xsd:complexType mixed="true">
         <xsd:complexContent>
         <xsd:extension base="gdstypes:ActivityInputType">
            <xsd:attribute name="prefix" type="xsd:string" />
            </xsd:extension>
         </xsd:complexContent>
         </xsd:complexType>
      </xsd:element>

      <xsd:element name="inputXSLT" minOccurs="1" maxOccurs="1">
      <xsd:complexType mixed="true">
         <xsd:complexContent>
         <xsd:extension base="gdstypes:ActivityInputType"/>
         </xsd:complexContent>
         </xsd:complexType>
      </xsd:element>

      <xsd:element name="output" minOccurs="1" maxOccurs="1">
      <xsd:complexType mixed="true">
```

Editors:
Vijay Dialani, University of Southampton
Indepal Narang, IBM

```
       <xsd:complexContent>
       <xsd:extension base="gdstypes:ActivityOutputType"/>
       </xsd:complexContent>
       </xsd:complexType>
     </xsd:element>

    </xsd:sequence>
    </xsd:extension>
  </xsd:complexContent>
  </xsd:complexType>

 <xsd:element                    name="xslTransform"                    type="tns:XSLTransformType"
 substitutionGroup="gdstypes:activity"/>

</xsd:schema>

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

<!-- (c) International Business Machines Corporation, 2002, 2003. (c) University of Edinburgh
2002, 2003. -->

<!-- See OGSA-DAI-Licence.txt for licencing information. -->

<gridDataServicePerform xmlns="http://ogsadai.org.uk/P2R2/schemas/gds">

  <request name="myRequest">

    <documentation>

      This perform document demonstrates how to perform an XSL transform

      on the results of a query statement. The deliverFromURL activity

      retrieves the XSLT file. The xPathStatement activity performs the

      query. The xslTransform activity transforms the results of the query

      using the XSLT file. The deliverToResponse element delivers the

      results of the transformation in the response document.

    </documentation>

 <!-- deliverFromURL activity retrieves the XSLT file -->

    <deliverFromURL name="deliverXSLT">

      <fromURL>http://www.abc.com/transform.xsl</fromURL>

      <toLocal name="deliverXSLTOutput" />

    </deliverFromURL>
 <!-- xPathStatement activity performs the query -->

     <xPathStatement name="statement">

        <dataResource>myXMLDBDataResource</dataResource>

        <expression>/entry[@id&lt;100]</expression>

        <sequenceStream name="statementOutput"/>

     </xPathStatement>

 <!-- xslTransform activity transforms the results of the query using the XSLT file -->

 <xslTransform name="transform">

   <inputXML from="statementOutput" />

   <inputXSLT from="deliverXSLTOutput" />

   <output name="transformedOutput" />

 </xslTransform>

 <!-- deliverToResponse element delivers the results of the transformation in the response
 document -->

     <deliverToResponse name="deliverResults">

        <fromLocal                    from="transformedOutput"                    />

     </deliverToResponse>

   </request>
   <execute name="executerequestsynch" requestName="myRequest"></execute>

</gridDataServicePerform>
```

4. Appendix 2
Transformations of Multiple Result-Sets Returned By A Stored Procedure Invocation

This section describes possible ways of associating the transformation activity with the invocation of the stored procedure that may result in unknown number of result-sets. Examples of transformations of result-sets are, a) an XSLT transformation for reformatting (includes filtering) the result-set, b) an XSLT transformation to enqueue the message in notification service.

The proposed specification will allow transformation to the result-sets generated by the invocation of a stored procedure in any of the following three formats:

1 A default transformation for all the result-sets.

2. An ordinal based association between the result-set and a default transformation for the unaccounted ordinals.

3. A conditional transformation based on schema and result-set's characteristics.

In addition, there may be a common set of transformation that may be applied to the group of the result-sets rather than to the individual result-set, for example compression of the results and/or filtering of the datasets. Group transformations can be use "a group by ordinal" or "group by schema" to create a transient result-set to which a transformation can be applied.

Examples of the proposed scheme are as follows:

1. A transformation applies to all the result-sets retrieved by the stored procedure if neither the ordinal nor the schema is specified.

2. An ordinal is used to provide one-to-one mapping (association) between the result-set and the transformation. In case the number of result-sets retrieved by the stored procedure exceeds the transformation, a default transformation is applied to the remaining non-associated result-sets.
Note:
The ordinals based transformations assume that the analyst has the complete knowledge about the order in which the result-sets will be delivered. This scheme is not suitable for dynamic binding.

3. A schema-based transformation is useful for providing a dynamic binding mechanism. A schema definition uses "required" and "optional" schema fields to determine the association between transformation and a result-set. The scheme can include the ordinal parameter to further restrict the application of a transformation to a result-set.

There exists more than one way of representing schema-based transformation. Examples include, an XSLT transformation on a result set and a proprietary implementation of some business logic. However, all types of transformations require some way to represent the schema and the conjunctive and disjunctive operations for comparing the transformation schema to the result set schema. We solicit advise from the community on adoption of a standard for representing the schema and type of comparative and logical operations they envisage.

4.1. Example

(Example of XSLT Transformations of Multiple result-sets produced by a Stored Procedure Activity within a Perform request – Enhancements by Vijay Dialani)

Below modifications to Appendix 1 are shown (they are high-lighted) to accommodate requirement for associating varying transformations for multiple result sets. Again it is based on the activity within the Perform request.
The original activity signature is of type
Transform (inputResultset, transformation, outputResultset)

has been augmented to include

Transform (inputResultset, complexTransform, outputResultset),

where complexTtransform is of the form

Editors:
Vijay Dialani, University of Southampton
Indepal Narang, IBM

```
transformAssociation{
   blob schema;
   int ordinal;
}

ComplexTransform{
         boolean default;
enum String association ={“ordinal”, “schema”,”ordinal and schema”};
         transformAssociation  association;
         transformation transform; /*XSLT file, same as in original activity */
}
```

GGF8 specification describes the perform document and a list of associated activities. For complex transformation we propose inclusion of a new activity – “**complexxslTransform”**. The following example demonstrates the use of complex transformation with aid of a perform document.

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- (c) International Business Machines Corporation, 2002, 2003. (c) University of Edinburgh
2002, 2003.-->
<!-- See OGSA-DAI-Licence.txt for licencing information.-->

<xsd:schema targetNamespace="http://ogsadai.org.uk/P2R2/schemas/gds"
xmlns:tns="http://ogsadai.org.uk/P2R2/schemas/gds"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
                  xmlns:gdstypes="http://ogsadai.org.uk/P2R2/schemas/gds"
                  elementFormDefault="qualified">

  <xsd:include schemaLocation="../../types/grid_data_service_types.xsd"/>

  <xsd:complexType name="XSLTransformType">
  <xsd:complexContent>
      <xsd:extension base="gdstypes:ActivityType">
      <xsd:sequence>

          <xsd:element name="inputXML" minOccurs="1" maxOccurs="1">
          <xsd:complexType mixed="true">
              <xsd:complexContent>
              <xsd:extension base="gdstypes:ActivityInputType">
                  <xsd:attribute name="prefix" type="xsd:string" />
                  </xsd:extension>
              </xsd:complexContent>
              </xsd:complexType>
          </xsd:element>

          <xsd:element name="inputXSLT" minOccurs="1" maxOccurs="1">
          <xsd:complexType mixed="true">
              <xsd:complexContent>
              <xsd:extension base="gdstypes:ActivityInputType"/>
              </xsd:complexContent>
              </xsd:complexType>
          </xsd:element>
```

```
     <xsd:element name="output" minOccurs="1" maxOccurs="1">
     <xsd:complexType mixed="true">
       <xsd:complexContent>
       <xsd:extension base="gdstypes:ActivityOutputType"/>
       </xsd:complexContent>
       </xsd:complexType>
     </xsd:element>

    </xsd:sequence>
    </xsd:extension>
  </xsd:complexContent>
  </xsd:complexType>

 <xsd:element name="xslTransform" type="tns:XSLTransformType"
 substitutionGroup="gdstypes:activity"/>

</xsd:schema>
```

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

<!-- (c) International Business Machines Corporation, 2002, 2003. (c) University of Edinburgh
2002, 2003. -->

<!-- See OGSA-DAI-Licence.txt for licencing information. -->

<gridDataServicePerform xmlns="http://ogsadai.org.uk/P2R2/schemas/gds">

  <request name="myRequest">

    <documentation>

      This perform document demonstrates how to perform an XSL transform

      on the results of a query statement. The deliverFromURL activity

      retrieves the XSLT file. The xPathStatement activity performs the

      query. The xslTransform activity transforms the results of the query

      using the XSLT file. The deliverToResponse element delivers the

      results of the transformation in the response document.

    </documentation>

 <!-- deliverFromURL activity retrieves the XSLT file -->

    <deliverFromURL name="deliverComplexTransform">

      <fromURL>http://www.abc.com/complextransform.xsl</fromURL>
```

```
        <toLocal name="delivercomplextransformOutput" />

      </deliverFromURL>

 <!-- xPathStatement activity performs the query -->

      <xPathStatement name="statement">

        <dataResource>myXMLDBDataResource</dataResource>

        <expression>/entry[@id&lt;100]</expression>

        <sequenceStream name="statementOutput"/>

      </xPathStatement>

 <!-- complexxslTransform activity transforms the
                         results of the query using the XSLT file -->

 <complexxslTransform name="transform">

   <inputXML from="statementOutput" />

   <inputXSLT from="delivercomplexTransformOutput" />

   <output name="transformedOutput" />

 </complexxslTransform>


 <!-- deliverToResponse element delivers the results of the transformation in the response
 document -->

      <deliverToResponse name="deliverResults">

        <fromLocal                 from="transformedOutput"                 />

      </deliverToResponse>

   </request>

   <execute name="executerequestsynch" requestName="myRequest"></execute>

</gridDataServicePerform>
```

However, as discussed at August face to face meeting the perform document may give way to an alternate approach that may require strict type checking, in which case a complex transformation could be represented as follows:

```
      <xsd:element name="transform_association" minOccurs="0">
      <xsd:complexType mixed="true">
```

Editors:
Vijay Dialani, University of Southampton
Indepal Narang, IBM

```
<xsd:element name="schemadoc" type="xsd:any" use="optional">
<xsd:element name="ordinal" type="xsd:int" use="optional">
</xsd:complexType>


<xsd:element name="ComplexTransform" minOccurs="0" maxOccurs="1">
<xsd:complexType mixed="true">
  <xsd:complexContent>
    <sequence>
    <xsd:element name="default" type="xsd:boolean" use="required">
    <xsd:simpleType name="association" base="xsd:string" use="required">
        <xsd:enumeration value="ordinal">
        <xsd:enumeration value="schema">
        <xsd:enumeration value="ordinal and schema">
    </xsd:simpleType>
    <xsd:element     name="association"     type="transform_association"
      use="required">
    <xsd:element name="inputXSL" type="inputXSLT" use="required">
    </sequence>
  </xsd:complexContent>
  </xsd:complexType>
</xsd:element>
```

5. Intellectual Property Statement

6. Full Copyright Notice

Editors:
Vijay Dialani, University of Southampton
Indepal Narang, IBM
Version 1.0
19th September 2003

7. References

1. Grid Database Access and Integration – Requirements and Functionality (GGF7 Submission – Feb 17, 2003)
2. Grid Database Access and Integration – Stored Procedues and Functions – Vijay Dialani, University of Southhampton
3. DAIS-wg communication – Tom Sugden comments on Transformations 6/17/2003 <dais-wg@gridforum.org>