GWD-R
GGF DAIS Working Group

Editors:
Mario Antonioletti, University of Edinburgh
Malcolm Atkinson, NESC
Susan Malaika, IBM
Simon Laws, IBM
Norman W Paton, University of Manchester
Dave Pearson, Oracle
Greg Riccardi, Florida State University

Category: INFORMATIONAL
19 September 2003

Grid Data Service Specification

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 management systems are central to many applications across multiple domains, and play a significant role in many others. Web services provide implementation neutral facilities for describing, invoking and orchestrating collections of networked resources. The Open Grid Services Architecture (OGSA) extends Web Services with consistent interfaces for creating, managing and exchanging information among Grid Services, which are dynamic computational artefacts cast as Web Services. Both Web and Grid service communities stand to benefit from the provision of consistent, agreed service interfaces to data management systems. Such interfaces must support the description and use of data management systems using Web Service standards, taking account of the design conventions and mandatory features of Grid Services. This document presents a specification for a collection of generic data access interfaces that can be extended to support access to specific kinds of data resource, such as relational databases, XML repositories or files. The proposal is presented for discussion within the Global Grid Forum (GGF) Database Access and Integration Services (DAIS) Working Group, with a view to the document evolving to become a proposed recommendation. There are several respects in which the current proposal is incomplete, but it is hoped that the material included is sufficient to allow an informed discussion to take place concerning both its form and substance.

Contents

1. Introduction

Data access plays an important role in many Grid applications. In general, data access involves both retrieval and manipulation of data, which may be stored or accessed using a range of paradigms. This implies the need for a flexible framework for request evaluation, and close integration with functionality for managing and moving data retrieved from, or for insertion into, a data resource.

This document provides a specification for a collection of generic grid data access interfaces. These interfaces instantiate the framework provided by the *OGSA Data Services* proposal [Data Services], in that interfaces are categorized according to the support they provide for data description, data access, data service creation and data management. As such, this document should be read in conjunction with the *OGSA Data Services* proposal. The specification does not mandate how these interfaces are composed into services; the proposed interfaces may be used in isolation or in conjunction with others. Furthermore, as this document stops short of providing interfaces for specific categories of data service, such as relational or XML data services, a more complete picture is provided by reading this document in conjunction with proposals for access to relational [GDRR] and XML [GDXR] representations of data. All of these documents assume some familiarity with the Open Grid Services Infrastructure (OGSI) [OGSI].

2. Notational Conventions

The key words “MUST,” “MUST NOT,” “REQUIRED,” “SHALL,” “SHALL NOT,” “SHOULD,” “SHOULD NOT,” “RECOMMENDED,” “MAY,” and “OPTIONAL” are to be interpreted as described in RFC-2119 [RFC2119].

This specification generally adopts the terminology currently defined in the *OGSA Data Services* document [Data Services]. The OGSA Data Services document is evolving and this terminology may change in future versions of the DAIS Working Group specifications.

Term	Description
Data Service	An OGSI compliant web service that implements at least one of the four portTypes, defined in this specification, either directly or indirectly.
Data Virtualization	A abstract view of some data as defined by the operations and attributes implemented by the Data Service.
Data Resource[1]	Called Data Source in the August 14th draft of the *OGSA Data Services* document. The components with which a Data Service’s implementation interacts to implement operations on a Data Virtualization.
Data Set	An encoding of data in a syntax suitable for externalization outside of a Data Service.
Synchronous	Describes an operation that, when called, responds with the requested data.
Asynchronous	Described an operations that, when called, initiates the process of obtaining the data but does not return it in the response to the call.

This specification uses namespace prefixes throughout; these are listed in the table below. Note that the choice of any namespace prefix is arbitrary and not semantically significant.

[1] Note that in previous versions of the Grid Data Service Specifications a Data Resource was taken to mean a Grid Service representing an *external data resource*. A Data Resource now refers to what previously was referred to as an *external data resource*.

Prefix	Namespace
dais	http://www.ggf.org/namespaces/2003/10/DAIS
gsa	WS-Agreement namespace URI
gwsdl	http://www.ggf.org/namespaces/2003/03/gridWSDLExtensions
http	http://www.w3.org/2002/06/wsdl/http
ogsi	http://www.ggf.org/namespaces/2003/03/OGSI
sd	http://www.ggf.org/namespaces/2003/02/serviceData
wsdl	http://schemas.xmlsoap.org/wsdl/
wsp	http://schemas.xmlsoap.org/ws/2002/12/policy
xsd	http://www.w3.org/2001/XMLSchema
xsi	http://www.w3.org/2001/XMLSchema-instance

3. Specification Overview

3.1 Scope Of Specification

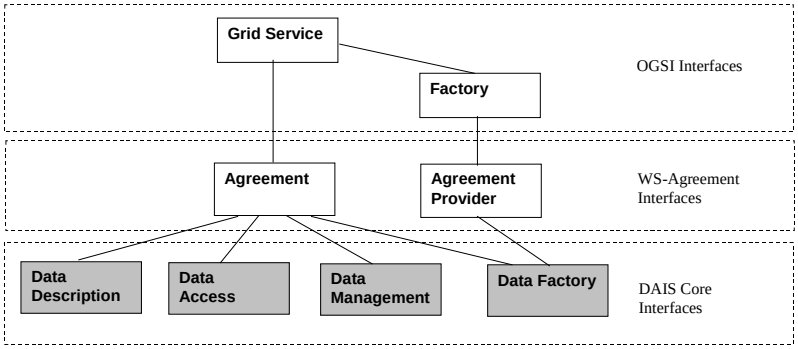
In this document the DAIS Working Group specifies the portTypes DataDescription, DataAccess and DataFactory introduced by the *OGSA Data Services* document [Data Services]. The portType DataManagement has not been considered fully but information relating to discussions to date is retained in section 9. In general the DAIS Working Group specifications are silent on Data Management related interfaces.

This document does not describe how the portTypes specified should be composed into services and implemented.

The DAIS Working Group specifications generally do not consider the interfaces required in order to manage a Data Resource. For example, no interfaces are defined for starting or stopping a relational database management system. Resource management overlaps with the work of other GGF Working Groups and is the subject of future discussion.

3.2 Port Type Summary

The *OGSA Data Services* [Data Services] document describes a Grid Service based approach to providing Grid access to Data Resources. Four base portTypes categorize a Data Service's interface. These are summarized below.



PortType	Description
Data Description	Describes the contents and structure of the Data Service.
Data Access	Provides access to data in the Data Service.
Data Factory	Creates derivations of a Data Service in the context of a valid

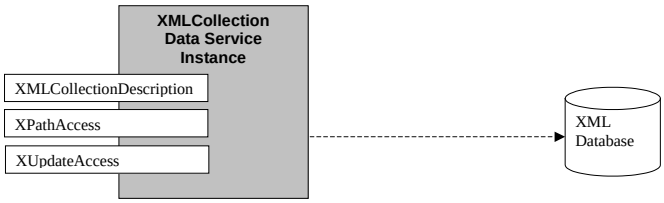
	agreement.
Data Management	Manages the Data Service and its relationship with the underlying Data Resource.

Although the base portTypes are defined in outline in the *OGSA Data Service* document they are not actually specified there. It is the purpose of this document to define a data model independent version of these interfaces. Companion documents specify further specializations of these interfaces to deal with specific data models, i.e. relational and XML.

3.3 Port Type Composition

The *OGSA Data Services* document defines a Data Services compliant Grid Service as being a service that implements at least one of the four base portTypes. A Data Service may thus be constructed by implementing, or extending, one or more of the portTypes that the DAIS Working Group defines.

The specification of separate categories of portTypes does not imply that they may be combined arbitrarily. PortTypes should be combined in a way that is relevant to the semantics of the data access that a Data Service represents. For example:

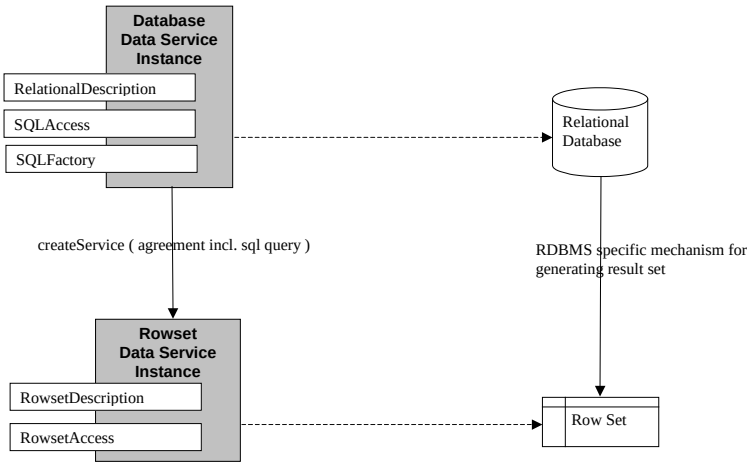


This example shows a Data Service providing XPathAccess and XUpdateAccess portTypes for access to an XMLCollection Data Servicethat, in this case, is associated with an XML Database. The XMLCollectionDescription portType provides information about the XML Collection Data Service.

This specification relies on WS-Agreement [WS-Agreement] as the mechanism for agreeing the interfaces and properties of Data Services. WS-Agreement documents that are advertised as being supported by Data Factory portTypes should describe valid combinations of portTypes.

3.4 Data Services Derivations

The ability to derive one Data Service from another to provide different views of the same Data Resources leads to a collection of notionally related Data Service instances. For example:



The *Database Data Service* in this example presents an SQLAccess portType for synchronous interactions with the Relational Data Resources. The createService() operation in the SQLFactory portType is used to construct the derived *Rowset Data Service*. This service provides access to the RowSet resulting from an SQL query against a Relational Database. The RowSet is a subset or restriction of the data in the database and is presented in tabular form. The RowSet could be stored as a table in a relational database or decoupled from the database, but the important distinction here is that the data is virtualized as a collection of rows that does not implement the SQLAccess portType. Instead the *Rowset Data Service* presents the RowsetAccess portType that allows rows to be retrieved but does not provide facilities for submitting SQL queries.

4. Interface Design Principles

In this specification, a position is taken on a number of interface design principles. This position is described here and presented in the portTypes that form the DAIS Working Group specifications. They are presented here in this section for convenience but may be moved in the future to a more appropriate document.

4.1 Data Service Artifact Relationships

The DAIS Working Group assumes the following relationships between artifacts identified in the OGSA Data Services document.

- One or more portTypes are exposed by one Data Service instance.
- One Data Service instance represents one Data Virtualization, as defined in the *OGSA Data Services* document.
- One Data Service interacts with one or more Data Resources.

DAIS compliant services MUST implement at least one of the DAIS specified portTypes.

4.2 Data Service Creation

4.2.1 Overview

Each Data Service instance, through its GSH, is uniquely namedwith an appropriate interface for its intended use. It may be desirable to create a new Data Service instance for a number of reasons, for example:

- To create a Data Service from an existing Data Service in order to represent a derived view of the data currently represented.

  - For example, a Data Service representing an RDBMS could be used to create a new Data Service which represents just one of the databases in the RDBMS.

- To create a session in which to perform asynchronous operations against a DataService.
  - For example, a long running query.

- To create an unpopulated Data Service to which new data can subsequently be added.
  - For example, an empty file handle to which content can be added.

The Data Factory portType, through the portTypes that extend it, is expected to be the primary mechanism by which one Data Service is explicitly created from another. Data Service creation may also be achieved implicitly by operations defined in the Data Access portTypes

4.2.2 Explicit Factories

Where Data Factory interfaces are defined, they will support Data Service creation in the context of a WS-Agreement document explicitly using a createService type operation. For instance, the example below specifies a number of terms describing the properties of a Data Service that is to be created. Note though that WS-Agreement is still embryonic and liable to change so this example is only illustrative.

```
<dais:daisBaseAgreement>

       <wsp:OneOrMore wsp:Usage=″wsp:Required″>
              <dais:readable gsa:Negotiability=″gsa:Fixed″.../>
              <dais:updateable gsa:Negotiability=″gsa:Fixed″ .../>
       <wsp:OneOrMore/>

       <wsp:OneOrMore wsp:Usage=″wsp:Required″>
         <dais:supportsInterface dais:qname=″DataDescription″
                                 sa:Negotiability=″gsa:Fixed″.../>
         <dais:supportsInterface dais:qname=″DataAccess″
                                 sa:Negotiability=″gsa:Fixed″.../>
         <dais:supportsInterface dais:qname=″DataFactory″
                                 sa:Negotiability=″gsa:Fixed″.../>
         <dais:supportsInterface dais:qname=″DataManagement″
                                 sa:Negotiability=″gsa:Fixed″.../>
       <wsp:OneOrMore/>
</dais:daisBaseAgreement>
```

A WS-Agreement document such as this is supplied by the client through the OGSI Factory::createService() operation of the Data Service from which a new Data Service is to be derived.

The creation of Data Services is sufficiently complex to make the precise naming of Factory portTypes difficult. For example, factory portTypes could include in their naming scheme an indication of what terms need to be supplied to characterize the new service instance and/or what type of service they can create. The DAIS Working Group has adopted the position that Factory portTypes should be named in accordance with the type of Data Access expressions that may be used to define the scope of the resulting Data Service. For instance an SQLFactory can receive an SQL expression that can be used to populate the Data Service that the Factory interface will create.

The creation of a Grid Data Service by a Data Factory can be considered a two-stage process,

1. Creation of a service providing the required operations/port types.
2. Materializing the data behind the Data Service. (i.e. For SQLFactory the results of the SQL Query).

The handle for this new service is returned at the end of step 1. Step 2 occurs asynchronously with respect to the Factory. The willingness of a Data Service to accept Data Access operations is indicated by the status SDE of the Data Access portType.

Characteristics of the newly created Data Service, as defined by the WS-Agreement document for the service, may dictate variations on this service creation theme. For example,

- A Data Service may not materialize the data behind the Data Service until it is requested via a Data Access operation.
- A Data Serivce may never materialize the data behind the Data Service, instead always retrieving it from a Data Resource on request.

The Factory::createService() operation returns a handle to one Grid Service. If the result of a creation request is more than one Data Service then these must be grouped together and the handle to the group returned from the createService operation.

4.2.3 Implicit Factories

When Data Services are created as the result of the execution of Data Access operations - that is without the use of an explicit Factory interface - an agreement must still be provided to the newly created Data Service. The agreement document will either be:

- Presented as an optional formal parameter to the operation.
- Defined by a service default.
- Derived from the agreement associated with the Data Service performing the operation.

The use of Data Access operations for Data Service creation presents an opportunity for the Data Service to decide, possibly based on data volume, whether a new Data Service should be created or whether any resulting data should be returned directly to the client.

4.3 Agreements

WS-Agreement [WS-Agreement] sets out a mechanism by which WS-Agreement XML documents are constructed from application specific terms and by which WS-Agreement terms may be negotiated and, if necessary, renegotiated.

Agreements to use Data Services could include such terms as:

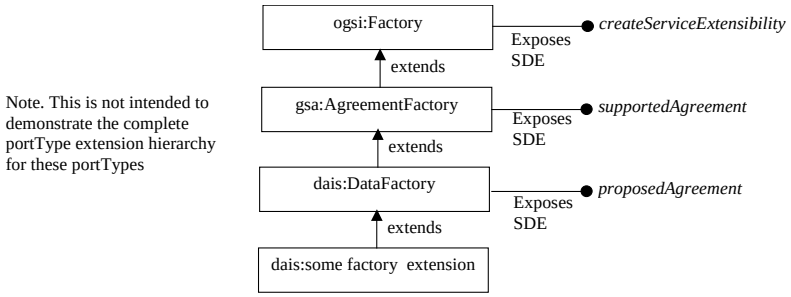
- Whether one or more users can access a Data Service concurrently.
- Whether a Data Service's data is cached in the Data Service or refreshed from the Data Resource on every access.
- Whether a Data Virtualization is readable, updateable or both.
- Whether a Data Service persists beyond the lifetime of the creating service.
- Whether updates pass through Data Service to the Data Resource(s).
- Whether a Data Service is updated when the Data Resource(s) is updated.
- What the transaction and concurrency model the Data Service should observe is.
- What data formats should be supported by the access interfaces.
- The maximum data set size that will be returned directly to a caller.
- The portTypes that a service will support.
- Security criteria for accessing a service.
- Query expression types that may be presented.

The WS Agreement document schema is intended to be extensible to add capabilities. This is achieved by:

1. Defining term types by extending gsa:TermType to describe terms.
2. Defining agreement types by extending gsa:AgreementType to describe documents that compose the terms appropriate to the Data Services being

All agreement types defined by DAIS will extend DAISBaseAgreementType (see 8.2) which in turn extends gsa:AgreementType. This currently adds no extra elements but forms a convenient base type from which general DAIS related agreement structures can be described if necessary.

Details of the WS-Agreement languages and documents that are understood by a service are exposed through a number of SDEs at various levels in the portType extension hierarchy.



The Factory:createServiceExtensibility SDE defines the elements that may validly appear as the optional creationParameters argument to the Factory::createService() operation. For example,

```
<ogsi:createServiceExtensibility inputElement="dais:daisBaseAgreement">
  <createsInterface>dais:DataDescription</createsInterface>
  <createsInterface>dais:DataAccess</createsInterface>
  <createsInterface>dais:DataFactory</createsInterface>
  <createsInterface>dais:DataManagement</createsInterface>
</ogsi:createServiceExtensibility>
```

The AgreementFactory:supportedAgreement SDE shows the agreement languages that may be used when creating a new agreement. The language is described by the type of the agreement structure and a list of agreement terms that may appear. For example,

```
<gsa:supportedAgreement qname="dais:DAISBaseAgreementType">
    <gsa:term qname="dais:ReadableType"/>
    <gsa:term qname="dais:UpdateableType"/>
    <gsa:term qname="dais:DataFormatType"/>
    <gsa:term qname="dais:MaxDataSetReturnSizeType"/>
    <gsa:term qname="dais:SupportsInterfaceType"/>
<gsa:supportedAgreement>
```

The DataFactory:proposedAgreement SDE presents documents that may validly be used to form the basis of WS-Agreements used during the creation of a new Data Service. It combines the terms with the agreement type using WS-Policy [WS-Policy] operators. For example,

```
<dais:proposedAgreement>
  <dais:daisBaseAgreement>

      <wsp:OneOrMore wsp:Usage="wsp:Required">
            <dais:readable gsa:Negotiability="gsa:Fixed".../>
            <dais:updateable gsa:Negotiability="gsa:Fixed" .../>
      <wsp:OneOrMore/>

      <wsp:OneOrMore wsp:Usage="wsp:Required">
        <dais:supportsInterface dais:qname="DataDescription"
                                sa:Negotiability="gsa:Fixed".../>
        <dais:supportsInterface dais:qname="DataAccess"
                                sa:Negotiability="gsa:Fixed".../>
        <dais:supportsInterface dais:qname="DataFactory"
                                sa:Negotiability="gsa:Fixed".../>
```

```
        <dais:supportsInterface dais:qname=”DataManagement”
                                    sa:Negotiability=”gsa:Fixed”.../>
      <wsp:OneOrMore/>

      ...
  </dais:daisBaseAgreement>
<dais:proposedAgreement>
```

The WS-Agreement approach is attractive as it provides a home for all of the terms that describe the characteristics of a Data Service and describes how these terms may be advertised, requested and negotiated.

This specifications use of WS-Agreement does mean however that most of the complex aspects of characterizing data access are dependent on efficient WS-Agreement manipulation and management mechanisms.

4.4 Query Expression Types

In a Data Access service that virtualizes query results using Data Services, part of an agreement may be a query expression describing the data that is to be virtualized by a Data Service. For example, when creating a Data Service to represent a result set the agreement should contain the expression to be executed across the parent Data Virtualization.

The inclusion of query expressions in agreement documents suggests that such expressions will appear in Data Access and Data Factory interfaces defined by the DAIS Working Group.

The DAIS Working Group will reuse the same structures and types to represent expressions wherever they appear. For example the XPathQuery operation in the XPathAccess portType might define an input message as follows:

```
<xsd:complexType name="XPathExpressionType">
    <xsd:element name="expression" type="xsd:string"/>
    <xsd:element name="collection" type="xsd:string" minOccurs="0"/>
    <xsd:element name="resourceId" type="xsd:string" minOccurs="0"/>
</xsd:complexType>

<message name="xPathStatementInputMessage">
      <part name="expression" type="daisxml:XPathExpressionType"/>
</message>
```

The type XPathExpressionType can be used as part of an agreement. For example,

```
<xsd:complexType name="XPathFactoryAgreementType">
  <xsd:complexContent>
    <xsd:extension base="dais:DAISBaseAgreementType">
      <xsd:sequence>
      <xsd:element name=”xpathExpression” type=”XPathExpressionType”>
      </xsd:sequence>
    </xsd:extension>
  </xsd:complexContent>
</xsd:complexType>
```

Here the expression is mandated as a element within the agreement as opposed to being represented as a separate term.

Using this approach common types can be used in Data Access operations and in createService() alike.

4.5 Context

Grid Services are Web Services that conform to the interfaces and behaviour defined by the OGSI specification. The notion of state in a Grid Service can be used to provide context to Data Access operations performed on a Data Service.

Consider a file system where a client navigates through the directory structure using the change directory (cd) command to locate a file.

1) In a Data Service representing a file system the notion of a “current working directory” could be represented as an SDE on the Data Service. This works well when one client is accessing the Data Service and there is no contention for what “current working directory” should be set to.

The definition of a stateful Grid Service does not preclude the use of context information passed in as a parameter to an operation in order to characterize the execution of a service operation.

Consider the file system example again. This time two clients must be allowed to navigate the directory structure independently.

2) Each client derives their own Data Service representing the file system and proceeds as in 1) setting current directory appropriately.
3) Access operations against the Data Service representing the file systemtake the directory path as context in order to locate the required file. The clients must maintain their own notion of a “current working directory”.

External context is crucial to other concerns such as Transactions and Security, and context may span many Data Services.

4.6 Access Control

The possibility of many client processes accessing a Data Service interface, possibly concurrently, is assumed to be the default situation.

Conflicting concurrent access can be controlled through the use of context information, for example, a Data Service representing a rowset may expose Data Access operations which expect to be provided with an iterator location so that a client can access a specified row without regard to which row other clients are accessing

Access to a Data Service may be denied, where appropriate, using suitable access mechanisms either at the service or the underlying Data Resource. Each call to an operation comes with a security context and the call is accepted as long as the context is allowed. For example, a system representing a relational database may wish to restrict access to the Data Services representing rowsets to the client that created those Data Services.

4.7 Operation Validity

The DAIS Working Group specification groups operations and SDEs in accordance with the type of Data Resource to be accessed. The appearance of an operation in a portType does not guarantee that it may be validly called in any particular situation. Faults are provided to notify the caller that an operation could not be completed successfully.

4.8 Data Formats

Data Access portTypes define operations for retrieving data from and adding data to Data Resources. By default such operations strongly type the parameters and return types that hold this data. As an alternative, the DAIS Working Group specifications also support a mechanism of dynamically typing data passing into and out of Data Access operations. Using this approach can

reduce the combinatorial effect of variable expression, input and return types on operation and portType naming.

A DataSet type is defined which wraps data along with a type description. This type is used as the input or output type for operations where variable types are required. DataSets are currently being scoped by members of the DAIS Working Group and may form future input to this Working Group.

An SDE, dataFormats, defined on the base Data Access portType, lists the data formats that the service is able to process and produce.

4.9 Operation Styles

The DAIS Working Group specifications assume that all operations defined on Data Access portTypes operate synchronously, i.e. the requested data is returned to the client on completion of the call. Hence data is “pulled” from Data Services.

Asynchronous operations are modeled by creating a new Data Service. The new Data Service is free to continue processing outside of the scope of the call to Factory::createService() operation that created it.

Data Access operations on the newly created service will of course be synchronous operations.

4.10 Sessions

A single client's session can be modeled as a Data Service, which is accessible by only that client. Alternatively, session semantics can be obtained by defining context to be passed to Data Access operations.

4.11 Data Service Management

The addition and removal of artifacts from a Data Service is considered to be a Data Management portType activity. For example, the addition of two files to a file based Data Service that represents them as a contiguous set of data.

The addition and removal of artifacts from some Data Resource that is represented by a Data Service are considered to be a Data Access activity. For example the addition and removal of files from a file system represented by a file system Data Service.

Update operations, for example, createResource() and removeResource() from a Data Access portType dealing with XML collections, will be supported in Data Access interfaces in as much that they update the current Data Service.

Any management interfaces described by the DAIS Working Group are aimed at management of the Data Service. Management of the underlying Data Resource is not considered. Data Management portTypes are not yet considered in detail in the DAIS Working Group specifications. However information is collected regarding the types of SDEs and operations that would be expected in this category.

5. Related Standards

5.1 WS-Agreement

WS-Agreement is under development by the GGF GRAAP Working Group. This specification is extended by the core DAIS Working Group Data Services portTypes to represent the interfaces and properties that are to be observed by Data Service. WS-Agreement is used as the basis of negotiation and agreement of terms when creating and interacting with a Data Service instance.

While WS-Agreement is still evolving the DAIS Working Group specifications do not describe details of the structure of the agreement terms required. The options that agreements terms will make available are collected together and presented in the specialized Data Factory portTypes.

5.2 Metadata

The DAIS Working Group specifications include references to metadata expressed within port types that incorporate Service Data Elements (SDEs), which in turn use XML schema definitions. The specification emphasizes general-purpose metadata for describing and accessing data and metadata in flexible ways. Grid systems do require application specific metadata, even at the data instance level, e.g., to retrieve detailed data instance provenance information. Application-specific metadata will be described in separate specifications.

Some of the general-purpose standards required by the specification appear in other standards documents. For example:

- XQueryRequest as defined in W3C XQueryX [XQueryX].
- WebRowSet format returned to a client, defined in JCP JSR 114 [WebRowSet].

5.3 Notification

Grid applications will generally rely on notifications to support the loosely coupled soft state nature of their construction. This is also true of data Grids. Applications using interfaces defined by the DAIS Working Group specifications will rely on notification of a number of items, including:

- Data creation, update and deletion.
- Schema changes.
- Request progress and completion.
- Service creation and deletion.

Generally implementations MAY be able to act as notification sources for any or all SDEs they implement. Other than that, the DAIS Working Group specifications currently provide no support for other forms of Data Resource notification.

5.4 Transactions

Grid applications will have access to language or requests, such as those described in WS-Transaction [WS-Transaction], to specify: transaction start, transaction end (commit, rollback) and intermediate checkpoints (points to rollback to). Resource managers and Coordinators will have access to language or requests, such as those described in WS-Transaction and WS-Coordination [WS-Coordination] to indicate participation in a transaction, disposition of their part of the transaction. e.g., committed, rolled back, in doubt, etc. In a transactional environment, when a grid application accesses multiple recoverable resources that support the transaction concept, all resources updated by the grid application within a single transaction are committed or rolled back. Coordination context tokens flow between the calling application, the resource managers being coordinated and the coordinating system, in accordance with WS-Coordination.

Within the context of this specification, all recoverable Data Services that can be coordinated in transactions are managed by recoverable Data Resources such as database managers. This specification does not introduce additional recoverable resources outside the domain of the Data Resources it describes how to access through grid services.

The coordination of transactions across grid and non-grid resources is not considered.

There are a variety of possible transaction options for grid data access. For example:

- Each Data Access request executes in a separate transaction.
- Multiple Data Access requests to a single Data Service execute in a single transaction, but requests across multiple Data Services are not coordinated. Here the Data Service

must expose transaction control requests, but a separate coordinator system is not required.
- Multiple requests to multiple Data Services are coordinated. To support this scenario, a single transaction context must be transmitted through multiple Data Services.

While further investigation is required to define the full range of transaction support and its impact on the DAIS Working Group specifications the following holds.

By default the interfaces assume no transaction support.

If there is a transactional schema, as indicated in the WS-Agreement document for the Data Service, then the transactional context is assumed to be passed to the relevant operations (whether the transaction is local or distributed). Each operation is assumed to act in accordance with the transactional scheme.

5.5 Security

There are many aspects to security for data access in a grid, e.g., database access security, service access security. In general, the DAIS Working Group specifications will adopt the mechanisms specified by the GGF OGSA Security Working Group. There are areas of security that DAIS or a related data specification will consider in the future, e.g., for linking OGSA security with underlying database security and for mapping database users to grid users.

5.6 Data Distribution

The provision of services that provide flexible, high level distribution of data, using both push and pull models, is beginning to be addressed within the GGF [Data Distribution]. It is anticipated that portTypes defined in the Data Distribution specification will be used by other Data Services to support flexible and efficient data movement.

5.7 GGF Data Area: OREP-WG

Intends to use the DAIS Working Group Data Services portTypes to provide access to/from sources and sinks at the ends of the replication.

5.8 GGF Data Area: GridFTP

Intends to use the DAIS Working Group Data Services portTypes to provide data access to/from data sources and sinks at the ends of the transfer.

5.9 OASIS: WSDDM

Potentially provides a home for the Data Resource management issues and interfaces that the DAIS Working Group will not address.

5.10 GGF Information Systems and Performance Area: CMM-WG

Potentially provides a home for the Data Resource management issues and interfaces that the DAIS Working Group will not address.

5.11 GGF Information Systems and Performance Area: CGS-WG

Current defining CIM based models for grid resources. Will likely have an impact on the way that portTypes are extended from the base Grid Data Service portTypes.

6. DataDescription portType

The DataDescription portType extends the WS-Agreement Agreement portType. DataDescription defines SDEs that describe the data artifacts accessed by a particular Data Service.

As this is the base of all specific Data Description portTypes the SDEs and operations defined here are general across all extensions.

6.1 Service Data Declarations

- name: a name associated with the data represented by the Data Service

```
<sd:serviceData name=”name”
                type=”NameType?”
                minOccurs=”1”
                maxOccurs=”1”
                mutability=”mutable”
                modifiable=”false”
                nillable=”true”/>
```

- structure: a description of the structure of the Data represented by a Data Service. The mechanism by which the structure is described must be extensible as it is specific to the data model.

```
<sd:serviceData name=”structure”
                type=”StructureType?”
                minOccurs=”1”
                maxOccurs=”1”
                mutability=”mutable”
                modifiable=”false”
                nillable=”true”/>
```

- size: the size, in bytes, of the Data represented by a Data Service.

```
<sd:serviceData name=”size”
                type=”xsd:int”
                minOccurs=”0”
                maxOccurs=”1”
                mutability=”mutable”
                modifiable=”false”
                nillable=”false”/>
```

6.2 Operations

None.

7. DataAccess portType

The DataAccess portType extends the WS-Agreement Agreement portType. DataAccess provides operations to access and modify the contents of a Data Service.

As this is the base of all specific Data Access portTypes the SDEs and operations defined here are general across all extensions.

7.1 Service Data Declarations

- status: status of the Data Service with respect to data access. An enumeration with the values:

Ready	– The Data Service is ready to be accessed.
Initializing	– The Data Service is not ready to be accessed.
Error	– An error has occurred which has left the Data Service in an error state.

```
<sd:serviceData name="status"
                type="StatusType"
                minOccurs="1"
                maxOccurs="1"
                mutability="mutable"
                modifiable="false"
                nillable="false"/>
```

- dataFormat: valid formats of data passed out of the get() operation and into the put() operation.

```
<sd:serviceData name="dataFormat"
                type="xsd:QName"
                minOccurs="0"
                maxOccurs="unbounded"
                mutability="mutable"
                modifiable="false"
                nillable="false"/>
```

7.2 Operations

7.2.1 get

Retrieve all of the data in a Data Service.

Input

- *ResultFormat:* as selected from the set of dataFormat SDE values

Output

- *Result*: The data in the selected format

Fault(s)

- *GetNotSupportedFault:* Indicates that the service cannot return all of the data in one go.

7.2.2 put

Populate all of the data in a Data Service.

Input

- *Data:* The data in a format included in the set of dataFormat SDE values

Output

- *None*

Fault(s)

- *PutNotSupportedFault:* Indicates that the service cannot accept all of the data in one go.

8. DataFactory portType

The DataFactory portType extends the WS-Agreement Agreement and AgreementFactory portTypes. DataFactory supports a request to create a new data service whose data is derived from the data of the parent data service (the one that implements the DataFactory).

As this is the base of all specific Data Factory portTypes, the SDEs and operations defined here are general across all extensions.

8.1 Inherited Service Data Declarations

The WS-Agreement AgreementFactory portType defines an SDE, supportedAgreement, that describes and agreement language that is supported by the factory. The set of values of this SDE describe to the client what agreement languages may be used for deriving a new Data Serivce from the current Data Service.

The OGSI Factory portType defines an SDE, createServiceExtensibility, which describes the extensibility types that the Factory::createService() operation is able to accept. The SDE will define valid elements matching the WS-Agreement languages defined in the supportedAgreement SDE.

8.2 Service Data Declarations

- proposedAgreement: WS-Agreement documents that may be used as the basis for constructing specific WS-Agreement documents during the process of requesting the creation of a new Data Service.

```
<sd:serviceData name=”proposedAgreement”
                type=”gsa:DAIBaseAgreementType”
                minOccurs=”1”
                maxOccurs=”unbounded”
                mutability=”mutable”
                modifiable=”false”
                nillable=”false”/>
```

The DataFactory portType defines the following initial set of service data value elements.

```
<sd:staticServiceDataValues>
  <ogsi:createServiceExtensibility inputElement=”dais:daisBaseAgreement”>
    <createsInterface>dais:DataDescription</createsInterface>
    <createsInterface>dais:DataAccess</createsInterface>
    <createsInterface>dais:DataFactory</createsInterface>
    <createsInterface>dais:DataManagement</createsInterface>
  </ogsi:createServiceExtensibility>

  <gsa:supportedAgreement qname=”dais:DAISBaseAgreementType”>
    <gsa:term qname=”dais:ReadableType”/>
    <gsa:term qname=”dais:UpdateableType”/>
    <gsa:term qname=”dais:DataFormatType”/>
    <gsa:term qname=”dais:MaxDataSetReturnSizeType”/>
    <gsa:term qname=”dais:SupportsInterfaceType”/>
  <gsa:supportedAgreement>
</sd:staticServiceDataValues>
```

The following types are defined.

```
<xsd:complexType name=”DAISBaseAgreementType”>
  <xsd:complexContent>
    <xsd:extension base=”gsa:AgreementType”>
    </xsd:extension>
```

```
  </xsd:complexContent>
</xsd:complexType>

<xsd:complexType name="ReadableType">
  <xsd:complexContent>
    <xsd:extension base="gsa:TermType">
    </xsd:extension>
  </xsd:complexContent>
</xsd:complexType>

<xsd:complexType name="UpdateableType">
  <xsd:complexContent>
    <xsd:extension base="gsa:TermType">
    </xsd:extension>
  </xsd:complexContent>
</xsd:complexType>

<xsd:complexType name="DataFormatType">
  <xsd:complexContent>
    <xsd:extension base="gsa:TermType">
      <xsd:attribute name="qname" type="xsd"Qname"/>
    </xsd:extension>
  </xsd:complexContent>
</xsd:complexType>

<xsd:complexType name="MaxDataSetReturnSizeType">
  <xsd:complexContent>
    <xsd:extension base="gsa:TermType">
      <xsd:element name="bytes" type="xsd:int"/>
    </xsd:extension>
  </xsd:complexContent>
</xsd:complexType>

<xsd:complexType name="SupportsInterfaceType">
  <xsd:complexContent>
    <xsd:extension base="gsa:TermType">
      <xsd:attribute name="qname" type="xsd:QName"/>
    </xsd:extension>
  </xsd:complexContent>
</xsd:complexType>
```

Further agreement types will be defined in later versions of this document, as discussed in Section 4.3.

8.3 Inherited Operations

The Factory:createService() operation is fundamental to WS-Agreement and is used here to create one Data Service from another. The extensibility parameter to the createService() operation is the WS-Agreement document which characterizes the Data Service to be created.

8.4 Operations

None.

9. DataManagement portType

The DataManagement portType extends the WS-Agreement Agreement portType. DataManagement forms the basis of a set of portTypes used to manage Data within Data Services. In doing this it controls the interface between the real and virtual worlds. The interface generally contains monitor and control pairs and deals with issues orthogonal to data access.

9.1 Service Data Declarations

- version: a structure containing version information available from the underlying Data Resources that is exposed by the Data Service,

```
<sd:serviceData name="version"
                type="VersionType"
                minOccurs="0"
                maxOccurs="1"
                mutability="mutable"
                modifiable="false"
                nillable="true"/>
```

- lastModified: a timestamp indicating the point at which the data was changed through this Data Service.

```
<sd:serviceData name="lastModified"
                type="xsd:dateTime"
                minOccurs="1"
                maxOccurs="1"
                mutability="mutable"
                modifiable="false"
                nillable="true"/>
```

- dataResource: information about a Data Resource that this Data Service references.

```
<sd:serviceData name="dataResource"
                type="DataResource?"
                minOccurs="1"
                maxOccurs="unbounded"
                mutability="mutable"
                modifiable="false"
                nillable="false"/>
```

- roleMapping: a mapping from a Grid role to a database role.

```
<sd:serviceData name="roleMapping"
                type="RoleMapping"
                minOccurs="1"
                maxOccurs="unbounded"
                mutability="mutable"
                modifiable="false"
                nillable="false"/>
```

Further management Service Data Declarations will be defined in later versions of this document.

9.2 Operations

None.

10. Security Considerations

The Relational Realisation of a Grid Data Service will use standard Grid Security mechanisms as specified by OGSA Security working group combined with standard ways of relating Grid credentials and authorities to resource access rights. The assumption is that these standards will also indicate how to make information related to authentication, authorization security etc available.

11. Conclusions

This document has described a proposal for a collection of Data Services, which are extended in companion documents to provide support for multiple data storage paradigms. The services proposed are Grid services, in that they conform to and make use of the Open Grid Services Infrastructure [OGSI]. XML Schema and WSDL definitions for this specification will be made available and incrementally updated on the DAIS Working Group Web Site. This is a work in progress, and feedback is welcomed on this document.

Editor Information

Mario Antonioletti,
EPCC,
University of Edinburgh,
James Clerk Maxwell Building,
Mayfield Road,
Edinburgh EH9 3JZ,
United Kingdom.

Malcolm Atkinson,
e-Science Institute,
15 South College Street,
Edinburgh EH8 9AA,
UK.

Simon Laws,
IBM United Kingdom Limited,
Hursley Park,
Winchester,
Hampshire, SO21 2JN,
United Kingdom.

Susan Malaika,
IBM Corporation,
Silicon Valley Laboratory,
555 Bailey Avenue,
San Jose, CA 95141,
USA.

Norman W. Paton,
Department of Computer Science,
University of Manchester,
Oxford Road,
Manchester M13 9PL,
United Kingdom.

Dave Pearson,
Oracle Corporation Ltd,
Thames Valley Park,
Reading,
Berkshire RG6 1RA,
United Kingdom.

Greg Riccardi,
Department of Computer Science,
Florida State University,

Tallahassee, FL 32306-4530,
USA.

Contributors

Vijay Dialani, University of Southampton.
Amy Krause, EPCC, University of Edinburgh.
Allen Luniewski, IBM.
Inderpal Narang, IBM.
Steve Tuecke, Globus/ANL.
Jay Unger, IBM.

Acknowledgements

The DAIS Working Group of the Global Grid Forum is active, and many people have contributed to discussions within the group in recent months, including but not limited to: Jay Unger, Bill Allcock, Dieter Gawlick, Shannon Hastings, Stephen Langella, Sastry Malladi, Paul Watson and Martin Westhead.

Intellectual Property Statement

Full Copyright Notice

WILL NOT INFRINGE ANY RIGHTS OR ANY IMPLIED WARRANTIES OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE."

References

[Data Distribution]
A. Downing, D. Gawlick, V. Gogate, C. Madsen, S. Mishra, I. Narang, M. Subramanian, Data Distribution in the Grid Environment, DAIS-WG Informational Draft, 9th Global Grid Forum, 19th September 2003.

[Data Services]
I.Foster, S.Tuecke, J.Unger, *OGSA Data Services*, DAIS-WG Informational Draft, 9th Global Grid Forum, 14th August 2003.

[GDRR]
M. Antonioletti, A. Krause, S. Hastings, S. Langella, S. Malaika, J. Magowan, S. Laws, N. W. Paton. *Grid Data Service Specification: The Relational Realisation.* DAIS-WG Informational Draft, 9th Global Grid Forum, 19th September 2003.

[GDXR]
M. Antonioletti, A. Krause, S. Hastings, S. Langella, S. Malaika, S. Laws, N. W. Paton. *Grid Data Service Specification: The XML Realisation* DAIS-WG Informational Draft, 9th Global Grid Forum, 19th September 2003.

[OGSA]
I. Foster, D. Gannon, The Open Grid Services Architecture Platform, Global Grid Forum Working Draft, 16th February 2003.

[OGSI]
S. Tuecke, K. Czajkowski, I. Foster, J. Frey, S. Graham, C. Kesselman, D. Snelling, P. Vanderpilt, Open Grid Services Infrastructure, Version 1.0, http://www.gridforum.org/ogsi-wg, March 13, 2003.

[RFC2119]
S. Bradner, *Key words for use in RFCs to Indicate Requirement Levels*, Internet Engineering Task Force, RFC 2119, http://www.ietf.org/rfc/rfc2119.txt, March 1997.

[WebRowSet]
J. Bruce, *Java Specification Request 114 JDBC Rowset Implementations* , Public Review 2, http://jcp.org/en/jsr/detail?id=114

[WS-Agreement]
K.Czajkowski, A.Dan, J.Rofrano, S.Tuecke, M.Xu, *Agreement-based Grid Service Management*, Version 0, June 12, 2003

[WS-Coordination]
F. Cabrera, G. Copeland, T. Freund, J. Klein, D. Langworthy, D. Orchard, J. Schwchuk and T. Storey, *Web Services Coordination (WS-Coordination)*, http://www-106.ibm.com/developerworks/library/ws-coor/, 2002-a.

[WS-Policy]
D. Box, F. Curbera, M. Hondo, C. Kale, D. Langworthy, A. Nadalin, N. Nagaratnam, M. Nottignham, C. Riegen, J. Shewchuk, Web Services Policy Framework, http://www-106.ibm.com/developerworks/library/ws-polfram/

[WS-Transaction]

F. Cabrera, G. Copeland, B. Fox, T. Freund, J. Klein, T. Storey and S. Thatte, *Web Services Transaction (WS-Transaction)*, http://www.ibm.com/developerworks/library/ws-transpec/, 2002-b.

[XQueryX]
A. Malhotra, J. Robie and M. Rys. *XML Syntax for XQuery 1.0 (XQueryX).* W3C Working, See: http://www.w3.org/TR/xqueryx.