

Web Services Data Access and Integration (WS-DAI)

Status of This Memo

This memo provides information regarding the specification of service based interfaces to data resources. The specification is presently a draft for discussion. It does not define any standards or technical recommendations. Distribution is unlimited.

Copyright Notice

Copyright © Global Grid Forum (2005). All Rights Reserved.

Abstract

Data resources play a significant role in many applications across multiple domains. Web services provide implementation neutral facilities for describing, invoking and orchestrating collections of networked resources. The GGF (Global Grid Forum) Open Grid Services Architecture (OGSA), and its associated specifications, defines consistent interfaces through web services to components of a grid infrastructure. Both the web and grid communities stand to benefit from the provision of consistent and agreed web service interfaces for data resources and the systems that manage them.

This document, *Web Services Data Access and Integration (WS-DAI)*, presents a specification for a collection of generic data interfaces that can be extended to support specific kinds of data resources, such as relational databases, XML repositories, object databases, or files. Related DAIS specifications define how specific data resources and systems can be described and manipulated through such extensions. The DAIS specifications form part of a broader activity within the GGF to develop OGSA. The DAIS specifications can be applied in regular web services environments or as part of a grid fabric.

Contents

Abstract.....	1
1. Introduction	3
1.1 Specification Scope.....	3
1.2 Specification Organization	3
2. Notational Conventions.....	3
3. Terminology	4
3.1 Data Resource	4
3.2 Data Conduit	5
3.3 Data Catalog	5
3.4 Data Service.....	5
3.5 Consumer.....	5
3.6 Data Set	5
3.7 Data Resource Abstract Name	5
3.8 Data Resource Address.....	5
4. Concepts.....	5
4.1 Data Service Model.....	5
4.2 Interface	7
4.3 Interface Composition	7
4.4 Naming.....	8
4.5 Properties.....	8
4.6 Direct Data Access.....	8
4.7 Indirect Data Access	9
4.8 Subscription Based Data Access.....	9
4.9 Lifetime.....	9
4.10 Sessions	9
4.11 Access Control	10
4.12 Operation Validity	10
4.13 Faults	10
5. Core	10
5.1 Static Data Description.....	10
5.2 Configurable Data Description	12
5.3 Data Access	14
5.4 Data Factory.....	16
5.5 Data Catalog	18
6. Data Conduit.....	19
6.1 Data Description.....	19
6.2 Data Access	20
6.3 Data Factory.....	20
7. Mapping The Abstract Model To WDSL	20
7.1 Related Specifications.....	20
8. Security Considerations.....	21
9. Conclusions	22
Editor Information	22
Contributors	23
Acknowledgements	23
Intellectual Property Statement	23
Full Copyright Notice	23
References	24
Appendix A – Core XML Schema.....	27
Appendix B – Core WSDL	31
Appendix C – Conduit XML	37
Appendix D – Conduit WSDL	38

1. Introduction

Data access plays a central role for many types of grid applications. Data access generally involves the retrieval, manipulation and insertion of data, which may be stored using a range of different formats and infrastructures. Data access in grids requires a flexible framework for handling data requests to a data resource that is to be integrated within a grid fabric as defined by the Open Grid Services Architecture (OGSA) [OGSA] of the Global Grid Forum (GGF).

This document specifies a collection of generic data access interfaces that are made available as web services. The interfaces described here are grouped into the following functional categories:

- Data description: the provision of metadata about the pertinent characteristics of a data resource that a service may wish to expose as well as associated properties that affect the interaction between a service and the data resource.
- Data access: the provision of access to data through a service interface.
- Data factory: the provision of indirect access to data resources through new service interfaces.

This specification provides a pattern for data service interfaces and the properties that describe or modify the behavior of these interfaces. The pattern described here can be extended to define interfaces to access particular types of data, as is done in the proposals to access relational [WS-DAIR] and XML [WS-DAIX] representations of data. Future specifications for accessing other specialized forms of data, for example, files or object databases, should also extend the base set of interfaces defined in this document.

1.1 Specification Scope

This document specifies a data service in terms of the base data access and data factory interfaces and base data description properties that a data service may implement.

This specification does not define new query languages or data models. Data access interfaces are therefore described in terms of existing language interfaces supported by the underlying data resource to which the service is providing access.

In this document data management refers to the management of the relationship between the web service interface and the data resource that stores the data. Data management is considered outside the scope of this version of this specification.

1.2 Specification Organization

This specification separates the abstract model of a data service from its operational representation. The abstract model is described using the terminology defined in Section 3, and employs the concepts elaborated upon in Section 4. Sections 5 and 6 present the core and conduit aspects of the abstract model respectively.

A mapping of the abstract model to the web services resource framework (WSRF) is described in Section 7.

Section 8 discusses security. Section 9 draws conclusions from this specification exercise.

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].

When describing concrete XML schemas and XML instance fragments, this specification uses the notational convention of [WS-Security]. Specifically, each member of an element's children or attributes property is described using an XPath-like notation (e.g., `/x:MyHeader/x:SomeProperty/@value1` indicates that namespace *x* is being used, the root element *MyHeader* and a child element *SomeProperty* with an attribute *value1*). The use of `{any}` indicates the presence of an element wildcard (`<xsd:any/>`). The use of `@{any}` indicates the presence of an attribute wildcard (`<xsd:anyAttribute/>`).

Italicized element names are used when an element is intended to be specified by the DAIS specification realizations.

In the body of the specification, when patterns of messages are described, the layout of the XML of each message is presented, as opposed to the XML schema; the XML Schema is provided in Appendix A. The following notation is used to indicate cardinality of XML elements in the XML fragments:

- * zero or more
- + one or more
- ? zero or one

Where no notation is added to an element, one instance of the element is expected.

This specification generally adopts the terminology defined in the Open Grid Services Architecture Glossary of Terms [OGSA Glossary]. The terms data service, data resource and data set are used as described in Section 3.

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.

Prefix	Namespace
xsd	http://www.w3.org/2001/XMLSchema
wsdai	http://www.ggf.org/namespaces/2005/06/WS-DAI

3. Terminology

3.1 Data Resource

A data resource represents any system that can act as a source or sink of data. Examples of data resources include relational or XML databases, file systems, sensor networks, etc.

The expectation is that data in a grid will still generally be managed using existing systems such as relational databases or file systems. An existing system will already provide consumers with mechanisms for accessing stored data. It is the responsibility of the existing system to manage the lifetime of the data within it. Such data resources are called *externally managed data resources*.

There will be situations where data is created in the context of DAIS data service alone and where the data is not managed by an existing system. In this case the data resource, as implemented as part of the DAIS service, is responsible for managing the lifetime of the data. Such data resources are called *service managed data resources*.

The DAIS specifications provide a common service oriented treatment for data resources that exposes these mechanisms in a manner that supports integration into an OGSA based grid.

3.2 Data Conduit

A data conduit provides messages to manage the lifetime of a data resource and to support querying over the properties of a data resource.

While the system represented by a data resource can exist outside the context of DAIS data services, a data conduit cannot. The data conduit is an optional component of a DAIS service.

3.3 Data Catalog

A data catalog defines messages that provide access to the abstract names and addresses of all of the resources available via a data service. The data catalog is optional.

3.4 Data Service

In this specification a data service is taken to mean a web service that implements one or more of the DAIS specified interfaces to provide access to data resources. As has already been stated, it is not the intention of DAIS to define new query languages or data models. The specifications of the DAIS working group provide a web service based data access framework, exposing existing data access techniques already available in a data resource or through the use of other relevant specifications as required.

3.5 Consumer

A consumer is an application that exploits the interface provided by a data service in order to access a data resource. A data service may act as a consumer to another data service.

3.6 Data Set

A data set is an encoding of data suitable for externalization outside a data service, for example, as an XML document or as a binary stream. The concept of a data set is introduced to describe data as it appears in the messages passing to and from data services, i.e. between the consumer and the data service.

3.7 Data Resource Abstract Name

For this purposes of the DAIS specification a data resource abstract name is defined as a unique and persistent name for a data resource suitable for machine processing that does not necessarily contain location information. An abstract name takes the form of a URI.

3.8 Data Resource Address

A concrete name that specifies the location of a data resource as accessed via a data service. The address takes the form of a web service end point and enough information to distinguish the data resource at that end point.

4. Concepts

4.1 Data Service Model

The focus of this specification is on defining base data service interfaces. DAIS considers interfaces for two main kinds of data resource:

Externally managed data resource: A data resource that contains and provides access to data conforming to a model associated with a WS-DAI realization. An externally managed data resource:

1. Normally has an existence outside the DAIS service.
2. Has its lifetime managed in ways that are not specified in the WS-DAI specifications.

An example of an externally managed data resource is a relational database.

Service managed data resource: A data resource that contains and provides access to data conforming to a model associated with a WS-DAI realization. A service managed resource:

1. Does not normally have an existence outside the service-oriented middleware.
2. Has its lifetime managed in ways that are specified in the WS-DAI specification.

An example of a service managed resource is a query result rowset preserved by a WS-DAI implementation for subsequent access.

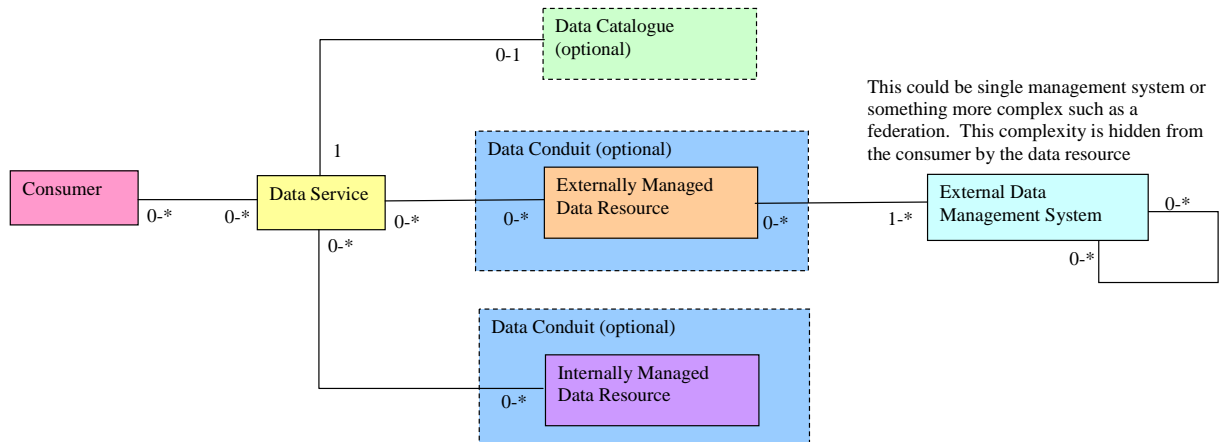


Figure 1: The relationships between DAIS constructs

Two main categories of interface information, that apply to both kinds of data resource, are described in this specification:

1. Core properties of the data resources being accessed.
2. Core messages for accessing data resources.

These messages contain the names of the resources being accessed

The DAIS realization specifications specialize the core by defining:

1. Specific properties of the kind of data resource being accessed.
2. Specific messages for accessing data resources.

The following optional resource type, called a data conduit, defines a generic WS-Resource whose purpose is to:

1. Control the lifetime of an individual data resource.
2. Provide messages for querying the properties of an individual data resource.

When present, the data conduit provides a unifying facade over both the externally managed and service managed data resources. Where a data conduit exists, it has a one-to-one correspondence with an externally managed or service managed data resource.

The following optional service level items for a given service are described in this specification:

A data catalog: a list of resources

A message for retrieving the catalog

A message for translating from a resource name to a resource address (can be the EPR of a WS-Resource)

A data service ultimately presents a consumer with an interface to a data resource. A data resource can have arbitrary complexity, for example, a federation of relational databases. A consumer is not typically exposed to this complexity, and operates within the bounds and semantics of the interface provided by the data service.

A data service implementation:

MAY include a catalog of resources.

MAY include at least one externally managed resource (with or without an associated conduit).

MAY include service managed resources (with or without associated conduits).

If a data service implementation supports a data conduit for any one resource, then it **MUST** support:

Conduits for all individual resources.

A message for translating from a resource name to resource address; the data catalog.

A data service implementation that does not include a data conduit is not dependent on WS-RF [WS-Resource].

The DAIS specifications include operations for creating service managed resources through a Data Factory.

4.2 Interface

The word interface refers to the collections of messages and XML structures that describe the ways in which a consumer can validly, through DAIS, interact with a data service. It is not intended to refer specifically to the proposed use of the word interface found in the current working draft of the WSDL 2.0 specification although this may be an appropriate mapping in the future.

4.3 Interface Composition

This specification does not mandate how interfaces are composed into data services; the proposed interfaces may be used in isolation or in conjunction with others. Viable compositions of interfaces will, initially, follow established patterns for data access. For example:

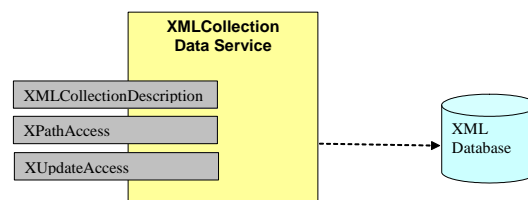


Figure 2: Interface composition

Here, a data service provides XPathAccess and XUpdateAccess interfaces for an XMLCollection data service that, in this case, is associated with an XML Database.

4.4 Naming

The OGSA naming scheme is a three-tiered system that allows for arbitrary web and grid resources to be identified in a global context. These resources are identified through hierarchically arranged human-oriented names, globally unique abstract names or low-level addresses.

For the purposes of the DAIS working group specifications, the data resource abstract name is a URI. The data resource address is a web service end point combined with the information required to discriminate the data resource at that endpoint. An example of a web service end point is the tag “soap:address” used in WSDL “port” descriptions when a SOAP binding is in use. An example of the information required to discriminate a data resource is the data resource abstract name when no data conduit is used. When a data conduit is used the conduit itself identified the data resource and the name is optional in messages.

The structure and use of human-oriented names is not considered by the DAIS working group.

A property, through which the data resource abstract name MAY be obtained, is provided as part of the base data description.

4.5 Properties

Properties describe the characteristics of a data resource as well as the data service's relationship with that data resource. Properties are represented by XML elements. They SHOULD be made available through the data service associated with a data resource.

Properties MAY be readable only through the data service interface.

Properties MAY be settable through the data service interface.

Properties MAY be required to be set on construction of a data service / data resource relationship.

The structure and valid states of any particular property are dependent on the data service and the data resource that the property describes.

Properties are associated with each collection of operations (portType) in the DAIS specifications. They are collected in the data description section for each collection of messages.

A subset of properties controls the behavior of each collection of operations. These properties appear in each data description section under the configuration properties heading.

Configuration properties MAY be set on construction of the data service / data resource relationship and are passed in as part of the message exchanges that implement the factory pattern.

4.6 Direct Data Access

In this specification direct data access means that a consumer can expect a direct response, containing the requested data, following a request made to a data service. For example, passing an XPathQuery message to a data service will result in a response message containing a set of XML fragments – this is considered to be direct data access.

4.7 Indirect Data Access

In this specification indirect data access means that a consumer does not expect the results to be contained in the response to a request made to a data service. The request to access data will be processed by the data service and data resource, with the results being made available to the consumer indirectly as a new data resource, often through a different data service that may support a different set of interfaces. The type and behavior of the new data resource are determined by the configuration parameters passed in with the original request.

For example, passing an SQL query message to a data service in this mode can result in a reference to another data service, and data resource, being produced that allows access to the result of the original query via a rowset interface.

Holding intermediate results at the service side can also minimize unnecessary data movement.

The indirect data access model can be used when data is being added. For example, with a data service representing a directory in a file system, indirect data access could be used to represent a new file into which new data can be inserted. However, it is not mandatory to apply it in all situations. For example, in the case of a relational database, indirect data access does not naturally model an empty table into which data is to be added.

4.8 Subscription Based Data Access

The DAIS specifications do not consider subscription based data access, where the consumer supplies a profile describing the data of interest and the conditions under which it will be delivered. Work is ongoing in the GGF Information Dissemination working group to specify this model¹.

4.9 Lifetime

When a data resource is removed, via its corresponding data conduit, the underlying data resource ceases to be accessible to the consumer. This does not necessarily mean that the data resources are themselves removed.

For externally managed data resources the lifetime of the DAIS data service has no effect on the lifetime of the data management system or the data it contains. As the data is managed by an external management system any data will remain until the external management system decides to remove it. For example, a relational data management system's databases will not be removed when the data resources that represent them are removed from a data service.

For service managed data resources the lifetime of the data resource is related directly to the lifetime of the data itself. When the data resource, or its corresponding data conduit, is removed so is the data. For example, the rowset resulting from a SQL query should be removed when the consumer has finished with it. This can be achieved by removing the data conduit that provides access to the rowset.

4.10 Sessions

The DAIS specifications do not describe how multiple requests to a data service are correlated either for single or multiple consumers, or for single or multiple requests. This is left to other proposed web service specifications; for example, WS Coordination [WS-Coordination] or WS Context [WS-Context].

¹ For more information see the Infod WG page at <http://forge.gridforum.org/projects/infod-wg>.

4.11 Access Control

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

Access to a data service and the data it represents may be granted or denied, where appropriate, using suitable access mechanisms and interfaces either at the service or the underlying data resource. In particular, a data service implementation is responsible for mapping grid level credentials to credentials that are acceptable to the underlying data resource. DAIS does not specify these mechanisms and interfaces.

The requirement to pass and control security related information is common with many other specifications. The DAIS working group expects this requirement to be satisfied using other specifications such as WS Security [WS-Security].

4.12 Operation Validity

The DAIS working group specifications describe messages and properties in accordance with the type of interface being presented. The appearance of an operation in a portType does not guarantee that it may be called validly in any particular situation. Faults are provided to notify the caller that an operation could not be completed successfully.

4.13 Faults

This base specification defines general fault patterns. The message patterns and properties included in this document imply that any message in a DAIS realization **MUST** implement a minimum set of faults:

InvalidRequestDocumentFault	The message is not supported at all or it is not supported at this time.
InvalidResourceNameFault	The data resource that is the target of the message is not available. This could because the data resource has been identified incorrectly or it has stopped operating.
ServiceBusyFault	The service is already processing a message and concurrent operations are not supported.

5. Core

The core interface describes those properties and message required to access data resources. Where no data conduit is present the entire properties document **MUST** be retrieved using the realization specific messages defined for this purpose. Where a data conduit is present the messages associated with the data conduit **MAY** also be used.

5.1 Static Data Description

The data description contains XML structures that describe the properties of a data resource.

Properties described here are not required to be set as part of a factory pattern. The properties defined here **MUST** appear in all data services that implement data description. DAIS realizations based on this specification can extend the list of properties as required.

The mapping section describes how these elements are made available in WSDL.

5.1.1 DataResourceAbstractName

```
<xsd:element name="DataResourceAbstractName" type="xsd:anyURI"/>
```

```
/wsdai:DataResourceAbstractName
```

The abstract name associated with the data resource(s) as represented by the data service.

5.1.2 ParentDataResource

```
<xsd:element name="Parent" type="xsd:string"/>
```

/wsdai:Parent

If this data resource is a service managed data resource this property holds the data resource abstract name of the data resource from which it was generated. If the data resource is an externally managed data resource this property will be empty.

5.1.3 MessageDatasetMap

```
<xsd:element name="MessageDatasetMap">
  <xsd:complexType name="MessageDatasetType" >
    <xsd:sequence>
      <xsd:element name="MessageType" type="xsd:QName"/>
      <xsd:element name="DatasetType" type="xsd:QName"/>
    </xsd:sequence>
  </xsd:complexType>
</xsd:element>
```

/wsdai:MessageDatasetMap

A mapping between the QName of a message and the QName of a dataset type. For direct access operations the dataset type refers to the schema type of the returned dataset. Multiple instances of this property provide the map for a data resource.

/wsdai:MessageType

The QName of a message.

/wsdai:DatasetType

The QName of a schema describing a dataset type (for direct access operations).

5.1.4 MessageConfigurationMap

```
<xsd:element name="MessageConfigurationMap">
  <xsd:complexType name="MessageConfigurationType" >
    <xsd:sequence>
      <xsd:element name="MessageType" type="xsd:QName"/>
      <xsd:element name="ConfigurationType" type="xsd:QName"/>
    </xsd:sequence>
  </xsd:complexType>
</xsd:element>
```

/wsdai:MessageResponseMap

A mapping between the QName of a message and the QName of a configuration type. For indirect operations the resource type refers to the schema of the configuration document that describes the required data resource. Multiple instances of this property provide the map for a data resource.

/wsdai:MessageType

The QName of a message.

/wsdai:ConfigurationType

The QName of a schema describing a configuration document (for indirect operations).

5.1.5 GenericQueryLanguage

```
<xsd:element name="GenericQueryLanguage" type="xsd:anyURI" />
```

/wsdai:GenericQueryLanguage

The identifier of a query language that may be passed to the GenericQuery operation. An example of an identifier that may appear here is one of the DAIS namespace URIs. Multiple instances of this property describe the full set of valid languages for a data resource.

5.2 Configurable Data Description

Properties defined here MUST appear in all data services that implement data description. They SHOULD appear in configuration documents passed to factory operations as they describe data service and data resource behavior.

5.2.1 DataResourceDescription

```
<xsd:element name="DataResourceDescription" >
  <xsd:complexType mixed="true">
    <xsd:sequence>
      <xsd:any minOccurs="0" maxOccurs="unbounded" processContent="lax" />
    </xsd:sequence>
  </xsd:complexType>
</xsd:element>
```

/wsdai:DataResourceDescription

A free format description of the data resource(s) represented by a data service.

5.2.2 Readable

```
<xsd:element name="Readable" type="xsd:boolean" />
```

/wsdai:Readable

Has the value true if a data service is able to return data in response to query operations otherwise has the value false. The value of this property MAY depend on the credentials of the consumer.

5.2.3 Writeable

```
<xsd:element name="Writeable" type="xsd:boolean" />
```

/wsdai:Writeable

Has the value true if a data service is able to update data represented by the data service in response to update or insert operations. Otherwise has the value false. The value of this property MAY depend on the credentials of the consumer.

5.2.4 ConcurrentAccess

```
<xsd:element name="ConcurrentAccess" type="xsd:boolean" />
```

/wsdai:ConcurrentAccess

Has the value true if a data service is able to process more than one message concurrently otherwise it has the value false. If a service is unable to support concurrent operations the ServiceBusy fault will result from a message submitted while one is already being processed.

5.2.5 TransactionInitiation

```
<xsd:element name="TransactionInitiation">
```

```

<xsd:simpleType>
  <xsd:restriction base="xsd:token">
    <xsd:enumeration value="NotSupported"/>
    <xsd:enumeration value="Automatic"/>
    <xsd:enumeration value="Manual"/>
  </xsd:restriction>
</xsd:simpleType>
</xsd:element>

```

/wsdai:TransactionInitiation

Describes under what circumstances a transaction is initiated in response to messages. Takes the values:

NotSupported	Does not support Transactions.
Automatic	Transaction initiated for each message.
Manual	Transaction context under control of the consumer. DAIS does not define interfaces for controlling transactions manually. It is expected that other specifications such as WS-Transaction [WS-Transaction] will be used alongside the DAIS specifications.

5.2.6 TransactionIsolation

```

<xsd:element name="TransactionIsolation">
  <xsd:simpleType>
    <xsd:restriction base="xsd:token">
      <xsd:enumeration value="NotSupported"/>
      <xsd:enumeration value="ReadUncommitted"/>
      <xsd:enumeration value="ReadCommitted"/>
      <xsd:enumeration value="RepeatableRead"/>
      <xsd:enumeration value="Serialisable"/>
    </xsd:restriction>
  </xsd:simpleType>
</xsd:element>

```

/wsdai:TransactionIsolation

Describes how transactions behave with respect to other ongoing transactions. Takes the values:

NotSupported	Does not support transactions.
ReadUncommitted	Accesses uncommitted changes made by other transactions.
ReadCommitted	Accesses only committed changes made by other transactions.
RepeatableRead	Accesses only committed changes made by other transactions and ensures that no records read during the transaction are changed by other transactions.
Serialisable	Accesses only committed changes made by other transactions, ensures that no records read during the transaction are changed by other transactions and ensures that result sets read during the transaction are not extended by other transactions.

5.2.7 Sensitivity

```

<xsd:element name="Sensitivity">
  <xsd:simpleType>
    <xsd:restriction base="xsd:token">
      <xsd:enumeration value="NoSensitivity"/>
      <xsd:enumeration value="Insensitive"/>
    </xsd:restriction>
  </xsd:simpleType>
</xsd:element>

```

```

        <xsd:enumeration value="Sensitive"/>
    </xsd:restriction>
</xsd:simpleType>
</xsd:element>

```

/wsdai:Sensitivity

Given a data resource A and a data resource B from which A was created. This property describes the sensitivity to change of data resource A with respect to changes in data resource B.

Takes the values:

NoSensitivity	There is no data resource to which this data resource can be considered to be sensitive.
Insensitive	Changes to the parent data resource do not affect the data presented by this data service/data resource.
Sensitive	Changes to the parent data resource are reflected in this data service/data resource. The property ParentDataResource gives the name of the parent data resource

For example, when reading forwards and backwards in a RowSet, "Insensitive" means that you will read the same results when you read forwards and then backwards regardless of any changes in the data resource that created the RowSet. "Sensitive" means that any changes in the data resource that created the RowSet will be observed.

5.3 Data Access

Data access collects together messages that directly access and modify the data represented by a data service along with the properties that describe the behavior of these access messages. For example,

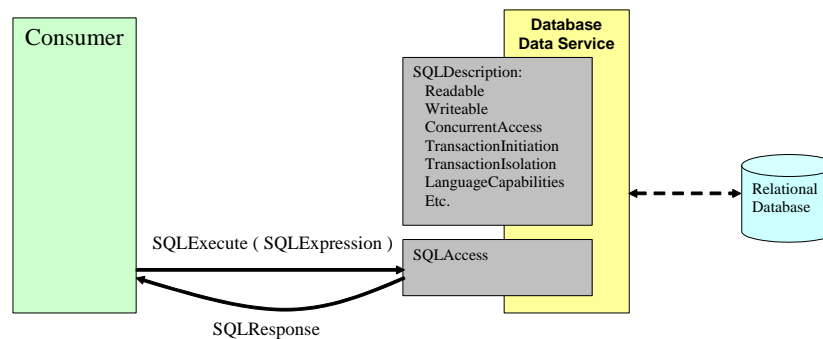


Figure 3: Data access example

The database data service, in the diagram above, implements the SQLAccess messages and exposes the SQLDescription properties; more details about these properties can be found in [WS-DAIR]. In this example, a consumer uses the SQLExecute message to submit an SQL expression. The associated response message will contain the results of the SQL execute request. When the SQL expression used is a SELECT statement, the SQL response will contain a RowSet.

The behavior of the database data service during data access is controlled, in part, by its configurable data description properties. These properties include the properties defined in this specification. For example, ConcurrentAccess appears in SQLDescription in this example and will

be set to true if the database data service is able to process messages from more than one consumer at the same time.

All configuration data description properties defined in this specification **MUST** appear in all data services. DAIS realizations based on this specification **MAY** extend the list of properties and messages as required and **MUST** define one or more access messages.

5.3.1 Message Patterns

DAIS specified data access interfaces support messages that allow data sets to be passed into or retrieved from a data service and describe the messages that provide direct data access from a data service.

The structure of a direct data access request message XML instance is:

```
<RequestMessage>
  <wsdai:DataResourceAbstractName/>?
  <RequestDocument/>
  <wsdai:ResponseType/>?
</RequestMessage>
```

/RequestMessage

This is the root element for a request message. The type of this element is specific to each message.

/RequestMessage/wsdai:DataResourceAbstractName

The abstract name of the data resource to which the message is directed. This element is optional.

/RequestMessage/RequestDocument

This element contains the request expression. The structure of this document is specific to the expression being used. The name of this element **SHOULD** indicate the language used by the expression.

/RequestMessage/wsdai:ResponseType

An optional element that can be used to define the type of the response message. This element **MUST** contain a QName from the set that appears in the *RequestMessageResponseTypeList* informational property element. When only one QName is advertised this element **MAY** be omitted in which case the format of the response message will follow that of the type reference by the advertised QName.

The structure of a direct access response message is:

```
<ResponseMessage>
  Data goes here formatted according to the response format parameter of
  the request message.
</ResponseMessage>
```

The structure of the response message is determined by the *<wsdai:ResponseType/>* element in the request message. This element contains the QName of a response message supported by the data service. Valid response message QNames are exposed by the data service using the *MessageResourceMap* property.

Realizations MAY choose to define interfaces containing statically typed response messages. In this case, the property `MessageResourceMap` SHOULD be omitted.

5.3.2 `GetDataResourcePropertyDocument`

Return the core property document associated with the service implementing this message.

Input

- *GetDataResourcePropertyDocumentRequest*
 - *ResourceName* – optional name of the resource the messages should be targeted at.

Output

- *GetDataResourcePropertyDocumentResponse*
 - *PropertyDocument* – the properties described in the data description section.

Faults

- *InvalidResourceName* – the supplied resource name is not known to the service.

5.3.3 `GenericQuery`

A generalised message for passing query documents to a data resource. The URIs of valid `GenericExpression` languages is provided by the `GenericQueryLanguage` property described in the data description section.

Input

- *GenericQueryRequest*
 - *ResourceName?* – optional name of the resource the messages should be targeted at.
 - *GenericExpression* – the query expression document.
 - *ReturnType?* – the required return type.

Output

- *GenericQueryResponse* – The response document formatted according to *ReturnType*

Faults

- *InvalidResourceName* – the supplied resource name is not known to the service.
- *InvalidExpression* – the supplied resource name is not known to the service.
- *InvalidResponseType* – the supplied resource name is not known to the service.

5.4 Data Factory

Messages defined in this section implement the factory pattern. Such messages create a new relationship between a data resource and a data service. In this way, a data resource may be used to represent the results of a query or act as a place holder for data to be inserted. A data factory describes properties that dictate how a data service must behave on receiving factory messages.

The factory pattern MAY involve the creation of a new data resource. The factory pattern MAY involve the deployment of a web service.

The factory pattern allows a new relationship between a data service and a data resource to be established as the consequence of a message exchange with a data service. This ability to derive one data resource from another, or to provide alternative views of the same data resources, leads to a collection of notionally related data resources. For example:

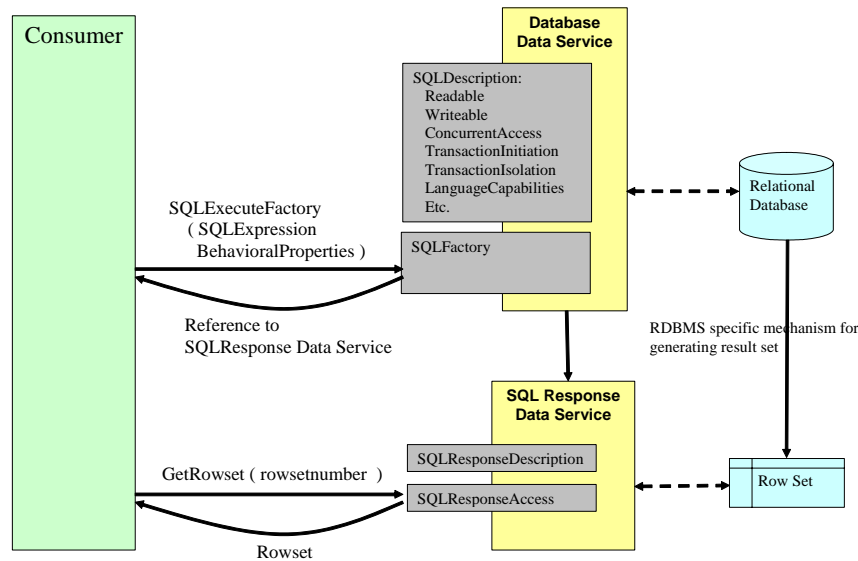


Figure 4: Data factory example

The database data service in this example presents a SQLFactory interface. The SQLExecuteFactory operation is used to construct the SQL response data service. This service provides access to the RowSet resulting from a SQL expression against the Relational Database, assuming that the expression contains a SELECT statement. The RowSet could be stored as a table in a relational database or decoupled from the database. The RowSet is represented as a collection of rows via a data service that does not implement the SQLAccess portType. Instead the SQL response data service presents the SQLResponseAccess collection of operations that allows the RowSet to be retrieved but does not provide facilities for submitting SQL expressions.

5.4.1 Message Patterns

The structure of a message implementing the factory pattern is:

```
<RequestMessage>
  <wsdai:DataResourceAbstractName/>?
  <RequestDocument/>
  <wsdai:Configuration/>
</RequestMessage>
```

/RequestMessage

This is the root element for a request message. The type of this element is specific to each message.

/RequestMessage/DataResourceAbstractName

The abstract name of the resource to which the message is directed. This element is optional.

/RequestMessage/RequestDocument

This is the request part of the message. The structure of this document is specific to the request message. This name of this element implies the language used by the request document.

/RequestMessage/wsdai:Configuration

The initial values for the properties of the data resource that is to be constructed as a result of this message. These are provided by the consumer. The type of this element is specific to the request message and, in particular, the type of data resource that is expected to result from the processing of this message.

This specification does not adopt any particular mechanism for advertising the set of initial values of a service's behavioral proprieties. Nor has it adopted a mechanism for negotiating these initial values. It is expected that these will be defined in other specifications, for example, WS Agreement [WS-Agreement]. It is the responsibility of the consumer to provide initial values for all required behavioral properties given the constraints of the configuration document schema.

Configuration properties are not universally applicable and will make sense only in the context of a data service implementing a particular interface. For example, it does not make sense to discuss the forward or backward nature of an iterator for a service that only accepts the SQLExecute message. The document containing configuration properties **MUST** also specify the type of interface that is required to result from the factory message.

A valid set of behavioral property document schemas **MUST** be advertised using the MessageResponseMap property (see the Data Access section).

The structure of the factory pattern response message is:

```
<wsdai:DataResourceAddress>
  <wsa:EndPointReference>+
  <wsdai:DataResourceAbstractName?>
</wsdai:DataResourceAddress>
```

wsdai:DataResourceAddress

This is the root element for the response message

/wsdai:DataResourceAddress /wsa:EndPointReference

The end point (s) of a service that provides access to the newly created data resource

/wsdai:DataResourceAddress /wsdai:DataResourceAbstractName

The abstract name of the newly created resource. This is optional.

5.5 Data Catalog

The data catalog is an optional part of data access. It defines a message for retrieving a list of names and addresses of data resources accessible via the data service implementing the message. It also provides a message which translates between a data resource name and the address of a data service that is able to provide access to that named data resource

5.5.1 GetDataResourceCatalog

Return a list of names and addresses of data resource that are accessible via the data service that implements the message.

Input

- *GetDataResourceCatalogRequest*

Output

- *GetDataResourceCatalogResponse*

- *DataResourceAddress** – a structure containing the data resource abstract name and the data resource address.

Faults

5.5.2 Resolve

Translate a data resource abstract name into a list of data resource addresses

Input

- *ResolveRequest*
 - *DataResourceAbstractName* – The abstract name of the data resource

Output

- *ResolveResponse*
 - *DataResourceAddress** – a structure containing the data resource abstract name and the data resource address.

Faults

- *InvalidResourceName* – the supplied resource name is not known to the service.

6. Data Conduit

The data conduit describes those properties and messages required to access the properties of and control the lifetime of data resources.

6.1 Data Description

6.1.1 DataResourceProperties

```
<xsd:element name="DataResourceProperties">
  <xsd:complexType >
    <xsd:sequence>
      <xsd:element ref="wsdai:DataResourceAbstractName"/>
      <xsd:element ref="wsdai:ParentDataResource"/>
      <xsd:element ref="wsdai:MessageDatasetMap" maxOccurs="unbounded"/>
      <xsd:element ref="wsdai:MessageConfigurationMap" maxOccurs="unbounded"/>
      <xsd:element ref="wsdai:GenericQueryLanguage" maxOccurs="unbounded"/>
      <xsd:element ref="wsdai:DataResourceDescription" />
      <xsd:element ref="wsdai:Readable"/>
      <xsd:element ref="wsdai:Writeable"/>
      <xsd:element ref="wsdai:ConcurrentAccess"/>
      <xsd:element ref="wsdai:TransactionInitiation"/>
      <xsd:element ref="wsdai:TransactionIsolation"/>
      <xsd:element ref="wsdai:Sensitivity"/>
      <xsd:any namespace="##other" minOccurs="0" maxOccurs="unbounded"/>
    </xsd:sequence>
  </xsd:complexType>
</xsd:element>
```

/wsdai:DataResource

A structure that describes the data resource that the data conduit refers to

/wsdai:the various elements that appear

The core set of properties for the data resource as described in previous sections.

/wsdai:any

Any further information that describes the data resource. For example this could be the properties document defined by a realization.

6.2 Data Access

6.2.1 GetResourceProperty

Return a named resource property

Input

- *GetPropertiesRequest*
 - *Name* – the name of the property to retrieve

Output

- *GetPropertiesResponse*
 - *Property* - the property identified by the name parameter

Faults

6.2.2 Destroy

Destroy the data resource referenced by the data conduit

Input

- *DestroyRequest*

Output

- *DestroyResponse*

Faults

6.3 Data Factory

No data factory messages are defined in the data conduit.

7. Mapping The Abstract Model To WDSL

7.1 Related Specifications

The DAIS WSDL makes use of the following specifications:

- The specifications referenced by WS-I Basic Profile 1.0 [WS-I]
 - SOAP 1.1 [SOAP]
 - WSDL 1.1 [WSDL]
- WS-Addressing 10th August 2004 [WS-Addressing]
- WS-ResourceProperties 1.2 [WS-ResourceProperties]
- WS-ResourceLifetime 1.2 [WS-ResourceLifetime]

The use of WSDL 1.1 forces the manual aggregation of messages and properties from the DAIS base specification and from DAIS realizations into the port type for each data service.

The Web Services Resource Framework (WSRF) is a set of web services specifications that describe a WS-Resource construct as a means of expressing the relationship between stateful resources and web services. The reader is referred to the WSRF documentation for more information, for example “The WS-Resource Framework” [WS-Resource]. These specifications are being standardized with the OASIS WSRF technical committee.

It is expected that WS-Addressing, WS-ResourceProperties and WS-ResourceLifetime specifications will change before standardization is agreed; all such changes will be reflected in the DAIS specifications before they complete the standardization process within the GGF.

7.1.1 Mapping the Model

The DAIS model is mapped to a set of WSDL defined messages in the usual way. WSRF is employed in mapping the data conduit to WSDL thus providing access to data resource properties and allowing the lifetime of a data resource to be controlled.

For data services that do not implement the optional data conduit the messages **MUST** include the name of the data resource being accessed.

For data services that implement the optional data conduit each message **MAY** include the name of the data resource being accessed.

7.1.2 Data Conduit

WS-Addressing End Point References (EPR) implement the implied resource pattern [WS-Resource] identifying a data conduit within a data service.

The properties of each data conduit are accessed using the approach described by the WS Resource Properties specification [WS-ResourceProperties].

The lifetime of the data conduit is controlled using the messages and properties described in the WS Resource Lifetime specification [WS-ResourceLifetime].

Data resource properties may be accessed using the WS Resource Properties messages as well as via messages described by each specialization.

7.1.3 Data Resource

When a data conduit is used Data resources are identified using the address of the data conduit. The use of data resource names is optional in messages in this case.

When no data conduit is used names must be provided with messages in order to identify data resources.

Each DAIS specialization describes the set of messages required to access the data resource in question. This includes messages required to retrieve the properties document for each data resource.

The lifetime of the data resource is controlled using messages described by the data conduit.

7.1.4 Port Type Aggregation

The base specification defines a single generic access message. Data access messages defined in the realizations will appear in WSDL 1.1 definitions as operations on the port type of the data service. Due to the restrictions of WSDL 1.1, messages must be cut and pasted from the WSDL in the DAIS specification into the WSDL for the specific data service by the service developer.

Properties from realization specifications are grouped together into an XML schema type which is particular to the data service being developed. An element of this type is returned by specific message for the interface in question.

Further grouping of properties into configuration documents allows them to be passed easily with any factory messages.

8. Security Considerations

The realizations of a grid data service will use standard grid security mechanisms as specified by the OGSA AUTHZ 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.

9. Conclusions

This document has described a proposal for a collection of top level interfaces for access to data resources as services, which are extended in companion documents to provide support for multiple data storage paradigms. The interfaces proposed are intended to be compatible with the architecture to be proposed by the GGF Open Grid Services Architecture working group.

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.

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

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,
School 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, IBM.
Allen Luniewski, IBM.
Inderpal Narang, IBM.
Savas Parastatidis, University of Newcastle upon Tyne.
Steve Tuecke, Univa.
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, including but not limited to: Bill Allcock, Dieter Gawlick, Shannon Hastings, Stephen Langella, Sastry Malladi, Paul Watson and Martin Westhead.

Intellectual Property Statement

The GGF takes no position regarding the validity or scope of any intellectual property or other rights that might be claimed to pertain to the implementation or use of the technology described in this document or the extent to which any license under such rights might or might not be available; neither does it represent that it has made any effort to identify any such rights. Copies of claims of rights made available for publication and any assurances of licenses to be made available, or the result of an attempt made to obtain a general license or permission for the use of such proprietary rights by implementers or users of this specification can be obtained from the GGF Secretariat.

The GGF invites any interested party to bring to its attention any copyrights, patents or patent applications, or other proprietary rights which may cover technology that may be required to practice this recommendation. Please address the information to the GGF Executive Director.

Full Copyright Notice

Copyright (C) Global Grid Forum (2005). All Rights Reserved.

This document and translations of it may be copied and furnished to others, and derivative works that comment on or otherwise explain it or assist in its implementation may be prepared, copied, published and distributed, in whole or in part, without restriction of any kind, provided that the above copyright notice and this paragraph are included on all such copies and derivative works. However, this document itself may not be modified in any way, such as by removing the copyright notice or references to the GGF or other organizations, except as needed for the purpose of developing grid Recommendations in which case the procedures for copyrights defined in the

GGF Document process must be followed, or as required to translate it into languages other than English.

The limited permissions granted above are perpetual and will not be revoked by the GGF or its successors or assigns.

This document and the information contained herein is provided on an "AS IS" basis and THE GLOBAL GRID FORUM DISCLAIMS ALL WARRANTIES, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO ANY WARRANTY THAT THE USE OF THE INFORMATION HEREIN WILL NOT INFRINGE ANY RIGHTS OR ANY IMPLIED WARRANTIES OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE."

References

[OGSA]

I. Foster (Ed), H. Kishimoto (Ed), A. Savva (Ed), D. Berry, A. Djaoui, A. Grimshaw, B. Horn, F. Maciel, R. Subramaniam, J. Treadwell, J. Von Reich. *The Open Grid Services Architecture, Version 1.0*. Global Grid Forum. GFD-I.030. 29 January 2005.
<http://forge.gridforum.org/projects/ggf-editor/document/GFD.30/en/1>.

[OGSA Glossary]

J. Treadwell, *Open Grid Services Architecture Glossary of Terms*, GFD-I.044, January 25th 2005. <http://forge.gridforum.org/projects/ggf-editor/document/GFD.44/en/1>.

[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.

[SOAP]

D. Box, D. Ehnebuske, G. Kakivaya, A. Layman, N. Mendelsohn, H. F. Nielsen, S. Thatte, D. Winer. *Simple Object Access Protocol (SOAP) 1.1*. W3C Note 08 May 2000. Available from: <http://www.w3.org/TR/2000/NOTE-SOAP-20000508/>.

[WebRowSet]

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

[WS-Addressing]

A. Bosworth, D. Box (Ed), E. Christensen, F. Curbera (Ed), D. Ferguson, J. Frey, C. Kaler, D. Langworthy, F. Leymann, S. Lucco, S. Millet, N. Mukhi, M. Nottingham, D. Orchard, J. Shewchuk, T. Storey, S. Weerawarana. *Web Services Addressing (WS-Addressing)*. 10 August 2004. <http://www.w3.org/Submission/ws-addressing/>.

[WS-Agreement]

A. Andrieux, K. Czajkowski, A. Dan, K. Keahey, H. Ludwig, J. Pruyne, J. Rofrano, S. Tuecke and M. Xu. *Web Services Agreement Specification (WS-Agreement)*.

[WS-Context]

D. Bunting, M. Chapman, O. Hurley, M. Little, J. Mischinsky, E. Newcomer, J. Webber, K. Swenson. *Web Services Context (WS-Context)*, <http://www.oasis-open.org/committees/download.php/4344/WSCTX.pdf>

[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-DAIR]

M. Antonioletti, B. Collins, A. Krause, S. Malaika, J. Magowan, S. Laws, N. W. Paton. *Web Services Relational Data Access and Integration*. DAIS-WG Informational Draft, 14th Global Grid Forum. 17th June 2005.

[WS-DAIX]

M. Antonioletti, A. Krause, S. Hastings, S. Langella, S. Malaika, S. Laws, N. W. Paton. *Web Service XML Data Access and Integration*. DAIS-WG Informational Draft, 14th Global Grid Forum. 17th June, 2005.

[WSDL]

E. Christensen, F. Curbera, G. Meredith and S. Weerawarana. *Web Services Description Language 1.1*. World Wide Web Consortium. W3C Note 15 March 2001. <http://www.w3.org/TR/wsdl>.

[WS-I]

K. Ballinger, D. Ehnebuske, M. Gudgin, M. Nottingham, P. Yendluri. *Basic Profile 1.0*. 16 April 2004. Available from: <http://www.ws-i.org/Profiles/BasicProfile-1.0.html>.

[WS-Resource]

S. Graham (Ed), A. Karmarkar (Ed), J. Mischinsky (Ed), I. Robinson (Ed), I. Sedukhin (Ed). *Web Services Resource 1.2 (WS-Resource)*, Version 1.2, Committee Draft 01, 19 May 2005. http://docs.oasis-open.org/wsrf/wsrf-ws_resource-1.2-spec-cd-01.pdf

[WS-ResourceProperties]

S. Graham (Ed), J. Treadwell (Ed). *Web Services Resource Properties 1.2 (WS-ResourceProperties)*, Version 1.2, Committee Draft 01, 19 May 2005. http://docs.oasis-open.org/wsrf/wsrf-ws_resource_properties-1.2-spec-cd-01.pdf

[WS-ResourceLifetime]

L. Srinivasan (Ed), T. Banks (Ed). *Web Services Resource Lifetime 1.2 (WS-ResourceLifetime)*, Version 1.2, Committee Draft 01, 19 May 2005. http://docs.oasis-open.org/wsrf/wsrf-ws_resource_lifetime-1.2-spec-cd-01.pdf

[WS-Security]

OASIS Web Services Security 1.0 (WS-Security 2004) standard as of April 6th 2004, http://www.oasis-open.org/committees/tc_home.php?wg_abbrev=wss.

[WS-Transaction]

F. Cabrera, G. Copeland, B. Cox, T. Freund, J. Klein, T. Storey and S. Thatte, *Web Services Transaction (WS-Transaction)*, August 2002, <http://www-106.ibm.com/developerworks/webservices/library/ws-transpec/>.

[XQueryX]

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

[JDBC]

Ellis, J. Ho, L. Fisher, M. *JDBC 3.0 Specification*. Sun Microsystems, Inc.

[XPath]

J. Clark and S. DeRose. *XML Path Language (XPath)*, Version 1.0 W3C Recommendation 16 November 1999. See: <http://www.w3.org/TR/xpath>.

Appendix A – Core XML Schema

```
<?xml version="1.0" encoding="UTF-8"?>
<xsd:schema targetNamespace="http://www.ggf.org/namespaces/2005/06/WS-DAI"
  xmlns:xsd="http://www.w3.org/2001/XMLSchema"
  xmlns:wsdai="http://www.ggf.org/namespaces/2005/06/WS-DAI"
  xmlns:wsa="http://wsanamespace">

  <xsd:import namespace="http://wsanamespace"
    schemaLocation="./wsaddressing-220405.xsd" />

  <!-- general types -->
  <!-- A type that holds the abstract name (a URI) of a data resource -->
  <xsd:simpleType name="AbstractNameType">
    <xsd:restriction base="xsd:anyURI">
      </xsd:restriction>
    </xsd:simpleType>
  <xsd:element name="DataResourceAbstractName" type="wsdai:AbstractNameType"/>

  <!-- a complex type that describes the address of a data resource -->
  <xsd:complexType name="AddressType">
    <xsd:sequence>
      <xsd:element name="EPR" type="wsa:EndpointReferenceType" minOccurs="1" maxOccurs="unbounded"/>
      <xsd:element ref="wsdai:DataResourceAbstractName" minOccurs="0" maxOccurs="1"/>
    </xsd:sequence>
  </xsd:complexType>
  <xsd:element name="DataResourceAddress" type="wsdai:AddressType"/>

  <!-- the base type wrapper for input/output datasets -->
  <!-- when a service is built it is anticipated that -->
  <!-- wrappers will be defined to describe the required -->
  <!-- input and output data formats -->
  <xsd:complexType name="DatasetType" mixed="true">
    <xsd:sequence>
      <xsd:any namespace="##any"/>
    </xsd:sequence>
  </xsd:complexType>
  <xsd:element name="Dataset" type="wsdai:DatasetType"/>

  <!-- the base type for query expressions -->
  <xsd:complexType name="ExpressionType"/>
```

```
<xsd:element name="Expression" type="wsdai:ExpressionType"/>

<!-- the base type for requests that include an abstract resource name parameter -->
<xsd:complexType name="RequestType">
  <xsd:sequence>
    <xsd:element ref="wsdai:DataResourceAbstractName" minOccurs="0" maxOccurs="1" />
  </xsd:sequence>
</xsd:complexType>
<xsd:element name="Request" type="wsdai:RequestType"/>

<!-- Properties -->

<!-- An open description of the data resource -->
<xsd:element name="DataResourceDescription" >
  <xsd:complexType mixed="true">
    <xsd:sequence>
      <xsd:any minOccurs="0" maxOccurs="unbounded" processContents="lax"/>
    </xsd:sequence>
  </xsd:complexType>
</xsd:element>

<xsd:element name="Readable" type="xsd:boolean" default="true" />

<xsd:element name="Writable" type="xsd:boolean" />

<xsd:element name="ConcurrentAccess" type="xsd:boolean" />

<xsd:element name="TransactionInitiation">
  <xsd:simpleType>
    <xsd:restriction base="xsd:token">
      <xsd:enumeration value="NotSupported"/>
      <xsd:enumeration value="Automatic"/>
      <xsd:enumeration value="Manual"/>
    </xsd:restriction>
  </xsd:simpleType>
</xsd:element>

<xsd:element name="TransactionIsolation">
  <xsd:simpleType>
    <xsd:restriction base="xsd:token">
      <xsd:enumeration value="NotSupported"/>
      <xsd:enumeration value="ReadUncommitted"/>
    </xsd:restriction>
  </xsd:simpleType>
</xsd:element>
```

```
<xsd:enumeration value="ReadCommitted"/>
<xsd:enumeration value="RepeatableRead"/>
<xsd:enumeration value="Serialisable"/>
</xsd:restriction>
</xsd:simpleType>
</xsd:element>

<xsd:element name="Sensitivity">
  <xsd:simpleType>
    <xsd:restriction base="xsd:token">
      <xsd:enumeration value="NoSensitivity"/>
      <xsd:enumeration value="Insensitive"/>
      <xsd:enumeration value="Sensitive"/>
    </xsd:restriction>
  </xsd:simpleType>
</xsd:element>

<xsd:element name="ParentDataResource" type="xsd:string"/>

<!-- The mapping between message type and resource type. -->
<!-- For direct access, where results are returned directly -->
<!-- the DatasetType refers to the supported dataset types. -->
<xsd:element name="MessageType" type="xsd:QName"/>
<xsd:element name="DatasetType" type="xsd:QName"/>

<xsd:complexType name="MessageDatasetType" >
  <xsd:sequence>
    <xsd:element ref="wsdai:MessageType" />
    <xsd:element ref="wsdai:DatasetType" />
  </xsd:sequence>
</xsd:complexType>
<xsd:element name="MessageDatasetMap" type="wsdai:MessageDatasetType"/>

<!-- For indirect access where a new resource results the -->
<!-- ConfigurationType refers to the QName of the configuration -->
<!-- document that identifies the required port type and -->
<!-- provides initial property values -->
<xsd:element name="ConfigurationType" type="xsd:QName"/>

<xsd:complexType name="MessageConfigurationType" >
  <xsd:sequence>
    <xsd:element ref="wsdai:MessageType" />
```

```

    <xsd:element ref="wsdai:ConfigurationType" />
  </xsd:sequence>
</xsd:complexType>
<xsd:element name="MessageConfigurationMap" type="wsdai:MessageConfigurationType"/>

<!-- The URI of a valid query language that -->
<!-- may be used in the GenericQuery message -->
<xsd:element name="GenericQueryLanguage" type="xsd:anyURI"/>

<!-- property and configuration documents -->

<xsd:complexType name="ConfigurationDocumentType">
  <xsd:sequence>
    <xsd:element ref="wsdai:DataResourceDescription" minOccurs="0" maxOccurs="1"/>
    <xsd:element ref="wsdai:Readable" minOccurs="0" maxOccurs="1"/>
    <xsd:element ref="wsdai:Writeable" minOccurs="0" maxOccurs="1"/>
    <xsd:element ref="wsdai:ConcurrentAccess" minOccurs="0" maxOccurs="1"/>
    <xsd:element ref="wsdai:TransactionInitiation" minOccurs="0" maxOccurs="1"/>
    <xsd:element ref="wsdai:TransactionIsolation" minOccurs="0" maxOccurs="1"/>
    <xsd:element ref="wsdai:Sensitivity" minOccurs="0" maxOccurs="1"/>
  </xsd:sequence>
</xsd:complexType>
<xsd:element name="ConfigurationDocument" type="wsdai:ConfigurationDocumentType"/>

<xsd:complexType name="PropertyDocumentType">
  <xsd:complexContent>
    <xsd:extension base="wsdai:ConfigurationDocumentType">
      <xsd:sequence>
        <xsd:element ref="wsdai:DataResourceAbstractName" minOccurs="1" maxOccurs="1" />
        <xsd:element ref="wsdai:ParentDataResource" minOccurs="1" maxOccurs="1" />
        <xsd:element ref="wsdai:MessageDatasetMap" minOccurs="1" maxOccurs="unbounded"/>
        <xsd:element ref="wsdai:MessageConfigurationMap" minOccurs="1" maxOccurs="unbounded"/>
        <xsd:element ref="wsdai:GenericQueryLanguage" minOccurs="1" maxOccurs="unbounded"/>
      </xsd:sequence>
    </xsd:extension>
  </xsd:complexContent>
</xsd:complexType>
<xsd:element name="PropertyDocument" type="wsdai:PropertyDocumentType"/>

<!-- The head of the substitution group of configuration documents -->
<xsd:complexType name="ConfigurationType" abstract="true"/>
<xsd:element name="Configuration" type="wsdai:ConfigurationType" abstract="true"/>

```

```

<!-- The configuration document for the Core port type -->
<xsd:complexType name="CorePTConfigurationType">
  <xsd:complexContent>
    <xsd:extension base="wsdai:ConfigurationType">
      <xsd:sequence>
        <xsd:element name="PortType">
          <xsd:simpleType>
            <xsd:restriction base="xsd:QName">
              <xsd:enumeration value="wsdai:CorePT"/>
            </xsd:restriction>
          </xsd:simpleType>
        </xsd:element>
        <xsd:element ref="wsdai:ConfigurationDocument"/>
      </xsd:sequence>
    </xsd:extension>
  </xsd:complexContent>
  <xsd:element name="CorePTConfiguration" type="wsdai:CorePTConfigurationType"
    substitutionGroup="wsdai:Configuration"/>
</xsd:schema>

```

Appendix B – Core WSDL

```

<?xml version="1.0" encoding="UTF-8"?>
<wsdl:definitions name="wsdai"
  targetNamespace="http://www.ggf.org/namespaces/2005/06/WS-DAI"
  xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/"
  xmlns:xsd="http://www.w3.org/2001/XMLSchema"
  xmlns:wsdai="http://www.ggf.org/namespaces/2005/06/WS-DAI">

  <!-- WSDL IMPORTS ##### -->

  <!-- WSDL TYPES ##### -->
  <wsdl:types>

    <xsd:schema targetNamespace="http://www.ggf.org/namespaces/2005/06/WS-DAI"
      elementFormDefault="qualified">
      <xsd:include schemaLocation="./wsdai_core_types_0.7.xsd" />
    </xsd:schema>
  </wsdl:types>

```

```

<!-- ##### -->
<!-- ### General Types ### -->
<!-- ##### -->
<xsd:element name="InvalidRequestDocumentFault">
  <xsd:complexType/>
</xsd:element>

<xsd:element name="InvalidResourceNameFault">
  <xsd:complexType/>
</xsd:element>

<xsd:element name="ServiceBusyFault">
  <xsd:complexType/>
</xsd:element>

<!-- ##### -->
<!-- ### GetDataResourceCatalog ### -->
<!-- ##### -->
<xsd:element name="GetDataResourceCatalogRequest">
  <xsd:complexType/>
</xsd:element>

<xsd:element name="GetDataResourceCatalogResponse" >
  <xsd:complexType>
    <xsd:sequence>
      <xsd:element ref="wsdai:DataResourceAddress" minOccurs="0" maxOccurs="unbounded"/>
    </xsd:sequence>
  </xsd:complexType>
</xsd:element>

<!-- ##### -->
<!-- ### Resolve ### -->
<!-- ##### -->
<xsd:element name="ResolveRequest">
  <xsd:complexType>
    <xsd:sequence>
      <xsd:element ref="wsdai:DataResourceAbstractName"/>
    </xsd:sequence>
  </xsd:complexType>
</xsd:element>

<xsd:element name="ResolveResponse" >

```



```

    <xsd:complexType>
      <xsd:sequence>
        <xsd:element ref="wsdai:DataResourceAddress" maxOccurs="unbounded" />
      </xsd:sequence>
    </xsd:complexType>
  </xsd:element>

  <!-- ##### -->
  <!-- ### GetDatResourcePropertyDocument ### -->
  <!-- ##### -->
  <xsd:element name="GetDataResourcePropertyDocumentRequest">
    <xsd:complexType>
      <xsd:complexContent>
        <xsd:extension base="wsdai:RequestType" />
      </xsd:complexContent>
    </xsd:complexType>
  </xsd:element>

  <xsd:element name="GetDataResourcePropertyDocumentResponse" >
    <xsd:complexType>
      <xsd:sequence>
        <xsd:element ref="wsdai:PropertyDocument" />
      </xsd:sequence>
    </xsd:complexType>
  </xsd:element>

  <!-- ##### -->
  <!-- ### GenericQuery Messages ### -->
  <!-- ##### -->
  <xsd:element name="GenericExpression">
    <xsd:complexType>
      <xsd:sequence>
        <xsd:any namespace="##any" />
      </xsd:sequence>
      <xsd:attribute name="Language" type="xsd:anyURI" />
    </xsd:complexType>
  </xsd:element>

  <xsd:element name="GenericQueryRequest" >
    <xsd:complexType>
      <xsd:complexContent>
        <xsd:extension base="wsdai:RequestType">

```

```

        <xsd:sequence>
          <xsd:element ref="wsdai:GenericExpression"/>
          <xsd:element ref="wsdai:DatasetType" minOccurs="0" maxOccurs="1" />
        </xsd:sequence>
      </xsd:extension>
    </xsd:complexContent>
  </xsd:complexType>
</xsd:element>

  <xsd:element name="GenericQueryResponse" >
    <xsd:complexType>
      <xsd:sequence>
        <xsd:any minOccurs="0" maxOccurs="unbounded" />
      </xsd:sequence>
    </xsd:complexType>
  </xsd:element>

  <xsd:element name="InvalidExpressionFault" >
    <xsd:complexType/>
  </xsd:element>

  <xsd:element name="InvalidResponseTypeFault" >
    <xsd:complexType/>
  </xsd:element>

</xsd:schema>
</wsdl:types>

<!-- WSDL MESSAGES ##### -->
  <wsdl:message name="InvalidRequestDocumentFault">
    <wsdl:part name="InvalidRequestDocumentFault"
      element="wsdai:InvalidRequestDocumentFault" />
  </wsdl:message>

  <wsdl:message name="InvalidResourceNameFault">
    <wsdl:part name="InvalidResourceNameFault"
      element="wsdai:InvalidResourceNameFault" />
  </wsdl:message>

  <wsdl:message name="ServiceBusyFault">
    <wsdl:part name="ServiceBusyFault"
      element="wsdai:ServiceBusyFault" />

```

```
</wsdl:message>

<wsdl:message name="GetDataResourceCatalogRequest">
  <wsdl:part name="GetDataResourceCatalogRequest"
    element="wsdai:GetDataResourceCatalogRequest" />
</wsdl:message>

<wsdl:message name="GetDataResourceCatalogResponse">
  <wsdl:part name="GetDataResourceCatalogResponse"
    element="wsdai:GetDataResourceCatalogResponse" />
</wsdl:message>

<wsdl:message name="ResolveRequest">
  <wsdl:part name="ResolveRequest" element="wsdai:ResolveRequest" />
</wsdl:message>

<wsdl:message name="ResolveResponse">
  <wsdl:part name="ResolveResponse" element="wsdai:ResolveResponse" />
</wsdl:message>

<wsdl:message name="GetDataResourcePropertyDocumentRequest">
  <wsdl:part name="GetDataResourcePropertyDocumentRequest"
    element="wsdai:GetDataResourcePropertyDocumentRequest" />
</wsdl:message>

<wsdl:message name="GetDataResourcePropertyDocumentResponse">
  <wsdl:part name="GetDataResourcePropertyDocumentResponse"
    element="wsdai:GetDataResourcePropertyDocumentResponse" />
</wsdl:message>

<wsdl:message name="GenericQueryRequest">
  <wsdl:part name="GenericQueryRequest"
    element="wsdai:GenericQueryRequest" />
</wsdl:message>

<wsdl:message name="GenericQueryResponse">
  <wsdl:part name="GenericQueryResponse"
    element="wsdai:GenericQueryResponse" />
</wsdl:message>

<wsdl:message name="InvalidExpressionFault">
  <wsdl:part name="InvalidExpressionFault"
```

```
        element="wsdai:InvalidExpressionFault" />
</wsdl:message>

<wsdl:message name="InvalidResponseTypeFault">
  <wsdl:part name="InvalidResponseTypeFault"
    element="wsdai:InvalidResponseTypeFault" />
</wsdl:message>

<!-- WSDL PORTTYPES ##### -->
<wsdl:portType name="CoreCatalogPT">

  <wsdl:operation name="GetDataResourceCatalog">
    <wsdl:input name="GetDataResourceCatalogRequest"
      message="wsdai:GetDataResourceCatalogRequest" />
    <wsdl:output name="GetDataResourceCatalogResponse"
      message="wsdai:GetDataResourceCatalogResponse" />
    <wsdl:fault name="InvalidResourceNameFault"
      message="wsdai:InvalidResourceNameFault" />
  </wsdl:operation>

  <wsdl:operation name="Resolve">
    <wsdl:input message="wsdai:ResolveRequest" />
    <wsdl:output message="wsdai:ResolveResponse" />
    <wsdl:fault name="InvalidResourceNameFault"
      message="wsdai:InvalidResourceNameFault" />
  </wsdl:operation>
</wsdl:portType>

<wsdl:portType name="CorePT">

  <wsdl:operation name="GetDataResourcePropertyDocument">
    <wsdl:input name="GetDataResourcePropertyDocumentRequest"
      message="wsdai:GetDataResourcePropertyDocumentRequest" />
    <wsdl:output name="GetDataResourcePropertyDocumentResponse"
      message="wsdai:GetDataResourcePropertyDocumentResponse" />
    <wsdl:fault name="InvalidResourceNameFault"
      message="wsdai:InvalidResourceNameFault" />
  </wsdl:operation>

  <wsdl:operation name="GenericQuery">
```

```

    <wsdl:input message="wsdai:GenericQueryRequest" />
    <wsdl:output message="wsdai:GenericQueryResponse" />
    <wsdl:fault name="InvalidResourceNameFault"
        message="wsdai:InvalidResourceNameFault" />
    <wsdl:fault name="InvalidExpressionFault"
        message="wsdai:InvalidExpressionFault" />
    <wsdl:fault name="InvalidResponseTypeFault"
        message="wsdai:InvalidResponseTypeFault" />
  </wsdl:operation>

</wsdl:portType>

</wsdl:definitions>

```

Appendix C – Conduit XML

```

<?xml version="1.0" encoding="UTF-8"?>
<xsd:schema targetNamespace="http://www.ggf.org/namespaces/2005/06/WS-DAI"
    xmlns:xsd="http://www.w3.org/2001/XMLSchema"
    xmlns:wsdai="http://www.ggf.org/namespaces/2005/06/WS-DAI">

  <xsd:include schemaLocation="./wsdai_core_types_0.7.xsd" />

  <!-- data description -->
  <xsd:element name="DataResourceProperties">
    <xsd:complexType >
      <xsd:sequence>
        <xsd:element ref="wsdai:DataResourceAbstractName"/>
        <xsd:element ref="wsdai:ParentDataResource"/>
        <xsd:element ref="wsdai:MessageDatasetMap" maxOccurs="unbounded"/>
        <xsd:element ref="wsdai:MessageConfigurationMap" maxOccurs="unbounded"/>
        <xsd:element ref="wsdai:GenericQueryLanguage" maxOccurs="unbounded"/>
        <xsd:element ref="wsdai:DataResourceDescription"/>
        <xsd:element ref="wsdai:Readable"/>
        <xsd:element ref="wsdai:Writeable"/>
        <xsd:element ref="wsdai:ConcurrentAccess"/>
        <xsd:element ref="wsdai:TransactionInitiation"/>
        <xsd:element ref="wsdai:TransactionIsolation"/>
        <xsd:element ref="wsdai:Sensitivity"/>
        <xsd:any namespace="##other" minOccurs="0" maxOccurs="unbounded"/>
      </xsd:sequence>
    </xsd:complexType>

```

```

</xsd:element>

</xsd:schema>

```

Appendix D – Conduit WSDL

```

<?xml version="1.0" encoding="UTF-8"?>
<wsdl:definitions name="wsdai"
  targetNamespace="http://www.ggf.org/namespaces/2005/06/WS-DAI"
  xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/"
  xmlns:xsd="http://www.w3.org/2001/XMLSchema"
  xmlns:wsdai="http://www.ggf.org/namespaces/2005/06/WS-DAI">

  <!-- WSDL IMPORTS ##### -->

  <!-- WSDL TYPES ##### -->
  <wsdl:types>

    <xsd:schema targetNamespace="http://www.ggf.org/namespaces/2005/06/WS-DAI"
      elementFormDefault="qualified">
      <xsd:include schemaLocation="./wsdai_core_types_0.7.xsd" />

      <!-- ##### -->
      <!-- ### General Types ### -->
      <!-- ##### -->
      <xsd:element name="InvalidRequestDocumentFault">
        <xsd:complexType/>
      </xsd:element>

      <xsd:element name="InvalidResourceNameFault">
        <xsd:complexType/>
      </xsd:element>

      <xsd:element name="ServiceBusyFault">
        <xsd:complexType/>
      </xsd:element>

      <!-- ##### -->
      <!-- ### GetDataResourceCatalog ### -->
      <!-- ##### -->
      <xsd:element name="GetDataResourceCatalogRequest">

```

```

    <xsd:complexType/>
  </xsd:element>

  <xsd:element name="GetDataResourceCatalogResponse" >
    <xsd:complexType>
      <xsd:sequence>
        <xsd:element ref="wsdai:DataResourceAddress" minOccurs="0" maxOccurs="unbounded"/>
      </xsd:sequence>
    </xsd:complexType>
  </xsd:element>

  <!-- ##### -->
  <!-- ### Resolve ### -->
  <!-- ##### -->
  <xsd:element name="ResolveRequest">
    <xsd:complexType>
      <xsd:sequence>
        <xsd:element ref="wsdai:DataResourceAbstractName"/>
      </xsd:sequence>
    </xsd:complexType>
  </xsd:element>

  <xsd:element name="ResolveResponse" >
    <xsd:complexType>
      <xsd:sequence>
        <xsd:element ref="wsdai:DataResourceAddress" maxOccurs="unbounded"/>
      </xsd:sequence>
    </xsd:complexType>
  </xsd:element>

  <!-- ##### -->
  <!-- ### GetDatResourcePropertyDocument ### -->
  <!-- ##### -->
  <xsd:element name="GetDataResourcePropertyDocumentRequest">
    <xsd:complexType>
      <xsd:complexContent>
        <xsd:extension base="wsdai:RequestType"/>
      </xsd:complexContent>
    </xsd:complexType>
  </xsd:element>

  <xsd:element name="GetDataResourcePropertyDocumentResponse" >

```

```
<xsd:complexType>
  <xsd:sequence>
    <xsd:element ref="wsdai:PropertyDocument" />
  </xsd:sequence>
</xsd:complexType>
</xsd:element>

<!-- ##### -->
<!-- ### GenericQuery Messages ### -->
<!-- ##### -->
<xsd:element name="GenericExpression">
  <xsd:complexType>
    <xsd:sequence>
      <xsd:any namespace="##any"/>
    </xsd:sequence>
    <xsd:attribute name="Language" type="xsd:anyURI"/>
  </xsd:complexType>
</xsd:element>

<xsd:element name="GenericQueryRequest" >
  <xsd:complexType>
    <xsd:complexContent>
      <xsd:extension base="wsdai:RequestType">
        <xsd:sequence>
          <xsd:element ref="wsdai:GenericExpression"/>
          <xsd:element ref="wsdai:DatasetType" minOccurs="0" maxOccurs="1" />
        </xsd:sequence>
      </xsd:extension>
    </xsd:complexContent>
  </xsd:complexType>
</xsd:element>

<xsd:element name="GenericQueryResponse" >
  <xsd:complexType>
    <xsd:sequence>
      <xsd:any minOccurs="0" maxOccurs="unbounded" />
    </xsd:sequence>
  </xsd:complexType>
</xsd:element>

<xsd:element name="InvalidExpressionFault" >
  <xsd:complexType/>
```



```
        </xsd:element>

        <xsd:element name="InvalidResponseTypeFault" >
          <xsd:complexType/>
        </xsd:element>

      </xsd:schema>
    </wsdl:types>

<!-- WSDL MESSAGES ##### -->
  <wsdl:message name="InvalidRequestDocumentFault">
    <wsdl:part name="InvalidRequestDocumentFault"
      element="wsdai:InvalidRequestDocumentFault" />
  </wsdl:message>

  <wsdl:message name="InvalidResourceNameFault">
    <wsdl:part name="InvalidResourceNameFault"
      element="wsdai:InvalidResourceNameFault" />
  </wsdl:message>

  <wsdl:message name="ServiceBusyFault">
    <wsdl:part name="ServiceBusyFault"
      element="wsdai:ServiceBusyFault" />
  </wsdl:message>

  <wsdl:message name="GetDataResourceCatalogRequest">
    <wsdl:part name="GetDataResourceCatalogRequest"
      element="wsdai:GetDataResourceCatalogRequest" />
  </wsdl:message>

  <wsdl:message name="GetDataResourceCatalogResponse">
    <wsdl:part name="GetDataResourceCatalogResponse"
      element="wsdai:GetDataResourceCatalogResponse" />
  </wsdl:message>

  <wsdl:message name="ResolveRequest">
    <wsdl:part name="ResolveRequest" element="wsdai:ResolveRequest" />
  </wsdl:message>

  <wsdl:message name="ResolveResponse">
    <wsdl:part name="ResolveResponse" element="wsdai:ResolveResponse" />
  </wsdl:message>
```

```
<wsdl:message name="GetDataResourcePropertyDocumentRequest">
  <wsdl:part name="GetDataResourcePropertyDocumentRequest"
    element="wsdai:GetDataResourcePropertyDocumentRequest" />
</wsdl:message>

<wsdl:message name="GetDataResourcePropertyDocumentResponse">
  <wsdl:part name="GetDataResourcePropertyDocumentResponse"
    element="wsdai:GetDataResourcePropertyDocumentResponse" />
</wsdl:message>

<wsdl:message name="GenericQueryRequest">
  <wsdl:part name="GenericQueryRequest"
    element="wsdai:GenericQueryRequest" />
</wsdl:message>

<wsdl:message name="GenericQueryResponse">
  <wsdl:part name="GenericQueryResponse"
    element="wsdai:GenericQueryResponse" />
</wsdl:message>

<wsdl:message name="InvalidExpressionFault">
  <wsdl:part name="InvalidExpressionFault"
    element="wsdai:InvalidExpressionFault" />
</wsdl:message>

<wsdl:message name="InvalidResponseTypeFault">
  <wsdl:part name="InvalidResponseTypeFault"
    element="wsdai:InvalidResponseTypeFault" />
</wsdl:message>

<!-- WSDL PORTTYPES ##### -->
<wsdl:portType name="CoreCatalogPT">

  <wsdl:operation name="GetDataResourceCatalog">
    <wsdl:input name="GetDataResourceCatalogRequest"
      message="wsdai:GetDataResourceCatalogRequest" />
    <wsdl:output name="GetDataResourceCatalogResponse"
      message="wsdai:GetDataResourceCatalogResponse" />
    <wsdl:fault name="InvalidResourceNameFault"
      message="wsdai:InvalidResourceNameFault" />
  </wsdl:operation>
</wsdl:portType>
```

```
</wsdl:operation>

<wsdl:operation name="Resolve">
  <wsdl:input message="wsdai:ResolveRequest" />
  <wsdl:output message="wsdai:ResolveResponse" />
  <wsdl:fault name="InvalidResourceNameFault"
    message="wsdai:InvalidResourceNameFault" />
</wsdl:operation>

</wsdl:portType>

<wsdl:portType name="CorePT">

  <wsdl:operation name="GetDataResourcePropertyDocument">
    <wsdl:input name="GetDataResourcePropertyDocumentRequest"
      message="wsdai:GetDataResourcePropertyDocumentRequest" />
    <wsdl:output name="GetDataResourcePropertyDocumentResponse"
      message="wsdai:GetDataResourcePropertyDocumentResponse" />
    <wsdl:fault name="InvalidResourceNameFault"
      message="wsdai:InvalidResourceNameFault" />
  </wsdl:operation>

  <wsdl:operation name="GenericQuery">
    <wsdl:input message="wsdai:GenericQueryRequest" />
    <wsdl:output message="wsdai:GenericQueryResponse" />
    <wsdl:fault name="InvalidResourceNameFault"
      message="wsdai:InvalidResourceNameFault" />
    <wsdl:fault name="InvalidExpressionFault"
      message="wsdai:InvalidExpressionFault" />
    <wsdl:fault name="InvalidResponseTypeFault"
      message="wsdai:InvalidResponseTypeFault" />
  </wsdl:operation>

</wsdl:portType>

</wsdl:definitions>
```