GWD-R
GGF DAIS Working Group

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

Category: INFORMATIONAL

18 February 2005

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

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 web services. 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

1. Introduction

Data access plays a central role for many types of grid applications. By data access we mean the retrieval, manipulation and insertion of data, which may be stored using a range of different formats and infrastructures. Grid data access 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 provides a specification for a collection of generic grid data access interfaces that are made available as web services. The interfaces described here are categorized according to the support they provide for:

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

This specification aims to provide a framework for data service interfaces and properties. It stops short of describing specialised interfaces. The framework 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 specialised forms of data, for example, files or object databases, may 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 the specification; Section 8 is therefore a work in progress.

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 introduced in Section 4. Sections 5, 6 and 7 present the Data Description, Data Access and Data Factory aspects of the abstract model respectively.

A mapping of the abstract model to the web services resource framework (WSRF) is described in Section 8.

Section 9 discusses security. Section 10 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/>).

Italicised element names are used when an element is intended to be specified by DAIS specification realisations.

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 the appendix. The following notation is used to indicate cardinality of XML elements in these cases:

*	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]. In particular the terms data service, data resource and data set are used (see 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
http	http://www.w3.org/2002/06/wsdl/http
wsdl	http://schemas.xmlsoap.org/wsdl/
xsd	http://www.w3.org/2001/XMLSchema
xsi	http://www.w3.org/2001/XMLSchema-instance
wsdai	http://www.ggf.org/namespaces/2004/09/WS-DAI

3. Terminology

3.1 Data Resource

A data resource is 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. We expect that data in the grid will still generally be managed using existing technologies such as relational databases or file systems.

An existing data resource will already provide consumers with mechanisms for accessing stored data. 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 Service

A data service is a web service that implements one or more of the DAIS specified interfaces to provide access to data resources. 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 and using other relevant specifications as required.

3.3 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.

3.4 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.

3.5 Abstract Name

A unique and persistent name for a data resource suitable for machine processing that does not necessarily contain location information. Abstract names are bound to addresses. This specification does not define the format of abstract names

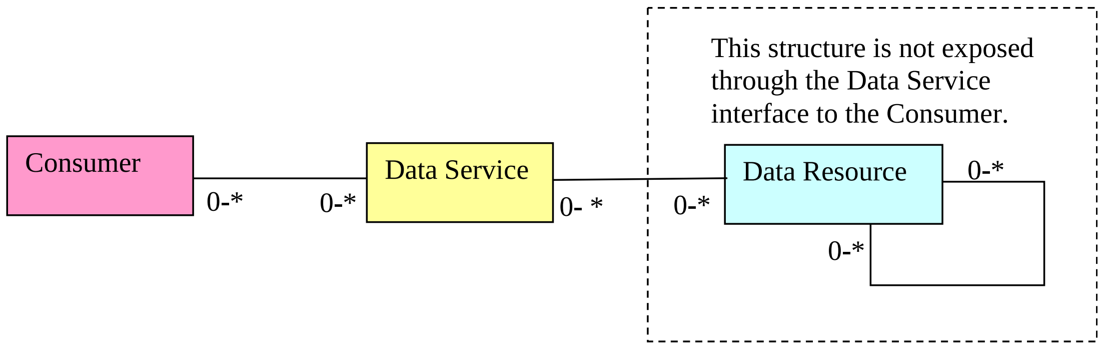
3.6 Address

A concrete name that specifies the location of a data resource as accessed via a data service.

4. Concepts

4.1 Data Service Model

The focus of this specification is on defining base data service interfaces. The specifications assume that there can be a many to many relationship between consumers and data services, and between data services and data resources.


This structure is not exposed through the Data Service interface to the Consumer.
Consumer
Data Service
Data Resource
0-*
0-*
0- *
0-*
0-*
0-*

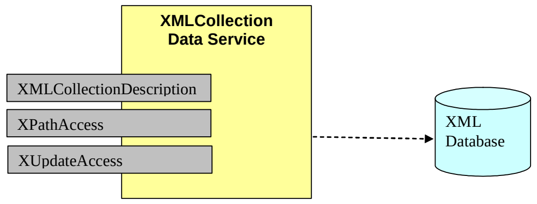
A data service presents a consumer with an interface to a data resource. A data resource can have arbitrary complexity, for example, a federation of relational databases. A consumer is not typically exposed to this complexity and operates within the bounds and semantics of the interface provided by the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

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 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 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:



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 Open Grid Services Architecture [OGSA] currently describes three levels of naming; human readable names, abstract names and addresses. The Open Grid Services Architecture does not specify the format of names and addresses or how translation from one level to another is achieved.

The DAIS working group specifications are concerned with access to data resources via data services. The mapping section in each of the specifications describes the mechanism used to address a data resource in order to access it via a data services.

This specification does not define a data resource naming or addressing scheme and gives no guidance on the information that might be used by a consumer to locate a suitable data service or data resource.

An informational property through which the abstract name of a data resource MAY be obtained is provided as part of the base Data Description but this specification does not describe the type of the abstract name or the scope in which it is considered valid.

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 read 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 the property describes.

Some properties describe the behavior of the data service given its data access and data factory messages. In the DAIS specifications these properties are described alongside the Data Access and Data Factory operations, the behavior of which they describe.

Other more general informational properties are collected in the data description sections.

Those properties that are set on construction of the data service / data resource relationship are collected together into XML structures called property documents that form part of the message exchanges that implement the factory pattern. The properties document combines the properties themselves with the name of the portType to which they relate.

4.6 Direct Data Access

In this specification direct data access means that a consumer can expect a direct response, containing requested data, to the requests 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 as 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, usually through a different data service and interface.

For example, passing an SQL query message to a data service in this mode will result in a reference to another data service (and possibly another data resource) being produced that allows access to the result of the original query. This allows results to be held at the service side for further processing, thus minimizing unnecessary data movement. The consumer may then use this new reference to retrieve the results or allow for further processing to take place at the server side when the results become available. A data service reference could be either a name or an address and is specific to the mapping being used.

This model can also 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 data can be added.

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 model [INFOD].

4.9 Lifetime

The DAIS specifications propose that the relationship between the lifetime of a data service and the associated data resource is either formed by operations that implement the factory pattern or by out of band mechanisms.

The relationship ends when one of the following occurs:

- The consumer sends a message requesting that the relationship ends (DAIS does not specify such a message. It is expected that others will specify how to control the lifetime of the service/resource relationship).
- The lifetime of the relationship elapses (DAIS does not specify how lifetime is determined. It is expected that others will specify how to control the lifetime of the service/resource relationship).
- The data service is removed or terminates.
- The data resource is removed or terminates.

The last two cases are management scenarios and are not considered. The first two cases are of interest to this specification. Depending on the mapping and the type of data resource in question the data resource MAY be removed when the relationship between data service and data resource ends.

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].

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 no messages and hence faults are discussed generally. The message patterns and properties included in this document imply that any message in a DAIS realisation MUST implement a minimum set of faults:

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

5. Data Description

Data Description contains XML structures that describe the properties of a data resource. The 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 will extend the list of properties as required.

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

5.1 AbstractName

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

/wsdai:AbstractName

The abstract name associated with the data resource(s) represented by the data service. The scope and structure of this name is not defined.

5.2 Description

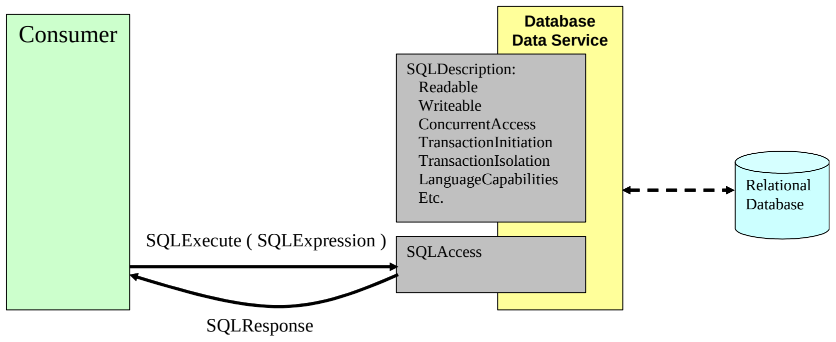
```
<xsd:element name="Description" >
    <xsd:complexType mixed="true">
      <xsd:sequence>
        <xsd:any minOccurs="0" maxOccurs="unbounded" processContent=”lax”/>
      </xsd:sequence>
    </xsd:complexType>
</xsd:element>
```

/wsdai:Description

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

6. 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,



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 behavioral properties. The behavioral properties will 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 data access behavioral 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.

6.1 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:

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

/wsdai:*RequestMessage*

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

/wsdai:*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.

/wsdai:*RequestMessage*/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 *RequestMessage*ResponseTypeList 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:

```
<wsdai:ResponseMessage>
       Data goes here formatted according to the response format parameter of
       request message
</wsdai: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 messages are exposed by the data service using an informational property whose name takes the form:

```
<wsdai:RequestMessageResponseTypeList />
```

Where *RequestMessage* MUST be the name of one of the supported request message types contained in this list.

Realizations MAY choose to define interfaces containing statically typed response messages. In this case the informational property *RequestMessage*ResponseTypeList SHOULD be omitted. When realizations define data access mechanisms that allow the consumer to define the response type the realization MUST provide an informational property of the form *RequestMessage*ResponseTypeList.

6.2 Properties

The behavioral properties shown here describe the behavior of a data service's data access messages. They are exposed by a data service, and optionally appear as part of message exchanges that implement the factory pattern.

6.2.1 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.

6.2.2 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.

6.2.3 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.

6.2.4 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].

6.2.5 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	Access uncommitted changes made by other transactions.
ReadCommitted	Access only committed changes made by other transactions.
RepeatableRead	Access only committed changes made by other transactions and ensure that no records read during the transaction are changed by other transactions.
Serialisable	Access only committed changes made by other transactions, ensure that no records read during the transaction are changed by other transactions and ensure that result sets read during the transaction are not extended by other transactions.

6.2.6 Sensitivity

```
<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>
```

/wsdai:Sensitivity

Describes the sensitivity to change of data being accessed indirectly through a data service/data resource relationship when compared with the data service that created it. 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.

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.

6.2.7 DataAccessPropertyType

The properties defined previously are collected together in an XML schema type that can be used as the base type for extension by the realizations. This type SHOULD be used as part of the properties document of a message that implements the factory pattern in order set the properties of a new data service / data resource relationship.

```
<xsd:complexType name="DataAccessPropertyType">
    <xsd:sequence>
      <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>
```

The properties in the type may all be validly omitted. Thus, by restriction, realisations may remove a property altogether. Alternatively, a realisation may allow a property to appear in the interface and be marked as not supported by an individual implementation by selecting the appropriate enumeration.

6.3 Messages

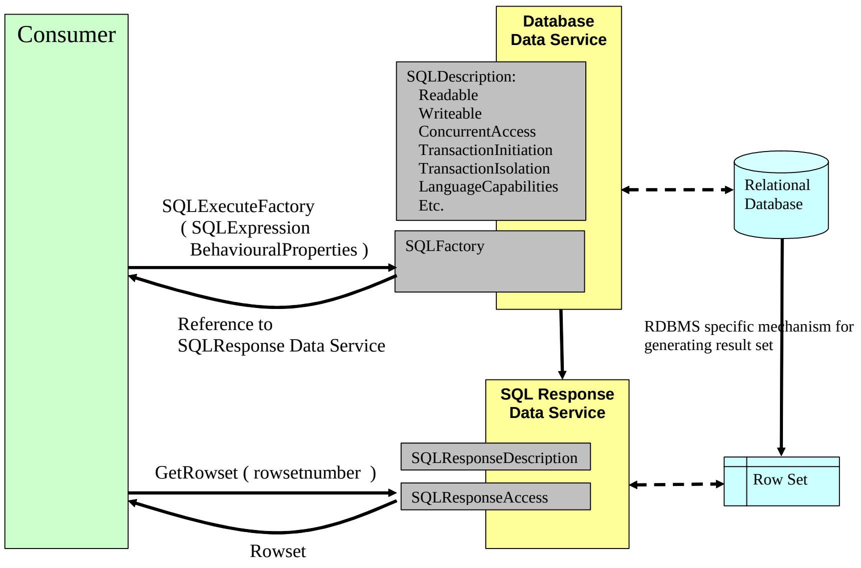
No data access messages are defined in the base specification. The specifications that extend this specification MUST define data access messages.

7. 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 service may be used to represent the results of a query or act as a place holder for data to be inserted into a data resource. A Data Factory describes behavioral 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 details of how the factory pattern is applied are mapping dependent.

The factory pattern allows a new relationship between a data service and a data resource to result as the consequence of a message exchange with a data service. This ability to derive one data service/data resource relationship from another or to provide alternative views of the same data resources leads to a collection of notionally related data service/data resource relationships. For 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 an 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, but the important distinction here is that the data 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.

7.1 Message Patterns

The structure of a message implementing the factory pattern is:

```
<wsdai:RequestMessage>
       <wsdai:RequestDocument/>
       <wsdai:Properties/>
</wsdai:RequestMessage>
```

/wsdai:*RequestMessage*

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

/wsdai:*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.

/wsdai:*RequestMessage*/*Properties*

The initial values of the properties of the data service 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 service that is expected to result from the processing of this message.

This specification does not adopt any particular mechanism for advertising the 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 properties document schema.

Behavioral 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 behavioral properties SHOULD also specify the type of interface that is required to result from the factory message. This is achieved by specifying the QName of the portType that is required in order to access the data resource.

A valid set of behavioral property document schemas SHOULD be advertised using the *RequestMessage*PropertyTypeList informational property. If this is present the provided service property document MUST match one of the XML types included in this property.

The structure of the factory pattern response message is:

```
<wsdai:ResponseMessage>
      reference*
</wsdai:ResponseMessage>
```

/wsdai :*ResponseMessage*

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

/wsdai:*ResponseMessage* /reference

> The response to a message employing the factory pattern is zero or more references to the new data service/data resource relationships. The type of this reference is mapping specific.

7.2 Properties

No behavioral properties are defined in the base specification.

7.3 Messages

No data factory messages are defined in the base specification. The specifications that extend this specification MAY define data factory messages.

8. Mapping to WSRF

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 the specifications will change, possibly significantly, before standardization is agreed.

In mapping the DAIS properties and messages to WSRF, the DAIS working group assumes that a number of specifications are available for use:

- 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 interfaces described by the DAIS specifications and the realizations are not necessarily coherent on their own. They simply present sets of messages and properties that can be combined with other specifications from elsewhere into meaningful port types.

8.1 Data Services

In the context of this mapping a data service is a web service that employs WSRF to allow access to data resources using the DAIS specified interfaces

In this mapping, WS-Addressing End Point References (EPR) implements the implied resource pattern [WS-Resource] identifying the data service-data resource relationship. For example, a relational database may be accessed via a data service using a set of SQL access messages by referring to EPR1. Alternatively the same relational database may be accessed via a data service using a set of XML access messages by referring to EPR2.

Consumers may be provided with EPRs or may discover EPRs through registries. EPRs registered in registries will be associated with properties particular to a data service. These may include the name and other descriptive information.

As in the previous example, many different EPRs can provide access to the same data resource through different interfaces. Many different EPRs can provide access to different data resources through the same data service.

To obtain a new EPR from an existing EPR, in order to access an existing data resource through a new interface, the consumer must return to the registry or call a factory operation.

8.2 Data Resources

No assumption is made about the form that a data resource takes. The data service implements a set of messages and informational properties. The data service is able to interact with a data resource in order to process these messages and provide values for the informational properties.

8.3 Data Description

Properties are represented as resource properties [WS-ResourceProperties] and hence the structure of the properties for a given data service is statically defined in the service’s WSDL document.

It is expected that resource properties will also be used to describe valid ranges of input values for certain message elements, in particular:

The Data Factory *RequestMessage*PropertyTypeList

> A list of QNames that identifies the valid behavioral properties that a factory operation can accept. Each of these structures combines a portType name with the property values that must be provided. This is used to tell the factory what type of service should be created and how it should be configured

The Data Access *RequestMessage*ResponseTypeList

> A list of QNames of valid response structures that a Data Access messages can generate.

In accordance with the approach outlined in the WS-ResourceProperties specification [WS-ResourceProperties], all property elements appropriate to a data service will be aggregated into a single resource property type which is in turn associated with the port type for the data service.

8.4 Data Access

The base specification defines no Data Access messages. 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 the base specification and from any 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 exposed as the resource properties structure for the data service.

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

8.5 Data Factory

The base specification defines no Data Factory messages. As with Data Access, Data Factory 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 data service by the service developer.

The base WSDL defines a structure that provides a template, as described in Section 7.1, through restriction, for combining the properties for a data service with the name of a port type to which they are relevant. This structure is used in the <wsdai:*Properties*/> element of messages implementing the factory pattern.

8.6 Data Management

Data Management messages have not yet been defined.

8.7 Faults

No faults are defined in the base specification.

8.8 Sessions

The creation of sessions across a data resource relies on further context information being provided in the messages, e.g., authentication information or other context information. Similarly, sessions across data resources also rely on further context information being provided.

8.9 Lifetime

The relationship between a data service and a data resource is constructed by messages implementing the factory pattern or by out of band means. The lifetime of the relationship is controlled using the properties and messages defined in the WS-ResourceLifetime specification.

9. Security Considerations

The Realizations 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.

10. 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. 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.

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, University of Southampton.
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: Bill Allcock, Dieter Gawlick, Shannon Hastings, Stephen Langella, Sastry Malladi, Paul Watson and Martin Westhead.

Intellectual Property Statement

Full Copyright Notice

References

[DAISCIM]
S. Malaika and N.W. Paton, *CIM Database Model for Data Access and Integration Services: Scenarios*, CGS-WG Informational Draft, 10th Global Grid Forum, 24th February 2004.

[INFOD]
D. Gawlick, V. Gogate, C. Kantariiew, C. Madsen, S. Mishra, I. Narang, M. Subramanian, Information Dissemination in the Grid Environment, DAIS-WG Informational Draft, 10th Global Grid Forum, 19th September 2003.

[OGSA]
I. Foster (Ed), H. Kishimoto (Ed). *The Open Grid Services Architecture, Version 1.0.* Global Grid Forum.

[OGSA Glossary]
J. Treadwell (Ed). *Open Grid Services Architecture Glossary of Terms*, July 2004.

[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. Mischkinsky, 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, 13th Global Grid Forum. 18th February 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, 13th Global Grid Forum. 18th February, 2005.

[WSDM]
J. DeCarlo, I. Sedukhin (editors), *Web Services Distributed Managegemt: Management Oof Web Services,* Draft 0.5, 2 April 2004, http://www.oasis-open.org/committees/download.php/6255/cd-wsdm-mows-0.5-20040402.pdf.

[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]
K. Czajkowski, D. Ferguson, I. Foster, J. Frey, S. Graham, I. Sedukhin, D. Snelling, S. Tuecke, W. Vambenepe, *The WS-Resource Framework*, Version 1.0, May 3rd 2004, http://www.oasis-open.org/committees/download.php/6796/ws-wsrf.pdf.

[WS-ResourceProperties]
S. Graham (Ed), J. Treadwell (Ed), *Web Services Resource Properties (WS-ResourceProperties).* Version 1.2 . 10 June 2004 http://docs.oasis-open.org/wsrf/2004/06/wsrf-WS-ResourceProperties-1.2-draft-04.pdf.

[WS-ResourceLifetime]
L. Srinivasan (Ed), T. Banks (Ed). *Web Services Resource Lifetime (WS-ResourceLifetime).* Version 1.2 10 June 2004. http://docs.oasis-open.org/wsrf/2004/06/wsrf-WS-ResourceLifetime-1.2-draft-03.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.

Appendix A – XML Schema

```
<?xml version="1.0" encoding="UTF-8"?>
<xsd:schema targetNamespace="http://www.ggf.org/namespaces/2004/09/WS-DAI"
            xmlns:xsd="http://www.w3.org/2001/XMLSchema"
            xmlns:wsdai="http://www.ggf.org/namespaces/2004/09/WS-DAI">

<!-- general types -->
  <!-- the base type for input/output datasets -->
  <xsd:complexType name="DatasetType"/>
  <xsd:element name="Dataset" type="wsdai:DatasetType"/>

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

<!-- data description -->
  <xsd:element name="Name" >
    <xsd:complexType mixed="true">
      <xsd:sequence>
        <xsd:any minOccurs="0" maxOccurs="unbounded" processContents="lax"/>
      </xsd:sequence>
    </xsd:complexType>
  </xsd:element>

  <xsd:element name="Description" >
    <xsd:complexType mixed="true">
      <xsd:sequence>
        <xsd:any minOccurs="0" maxOccurs="unbounded" processContents="lax"/>
      </xsd:sequence>
    </xsd:complexType>
  </xsd:element>

<!-- data access -->
  <xsd:element name="Readable" type="xsd:boolean" />

  <xsd:element name="Writeable" 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: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>

  <!-- presents a list of response types. The consumer selects from this list -->
  <!-- when constructing the response format element of a message for a direct access message -->
  <xsd:complexType name="ResponseTypeListType">
       <xsd:sequence>
             <xsd:element name="ResponseTypeQName" type="xsd:QName" minOccurs="1" maxOccurs="unbounded"/>
       </xsd:sequence>
  </xsd:complexType>

  <!-- the element used in direct access messages to pass the QName -->
  <!-- of the require response type -->
  <xsd:element name="ResponseType" type="xsd:QName"/>
```

```
<!-- data factory -->
  <!-- the base type for property documents that are passed in as part of the property -->
  <!-- document type -->
  <xsd:complexType name="DataAccessPropertyType">
    <xsd:sequence>
      <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>

  <!-- the base types for property documents passed into operations following the factory pattern-->
  <xsd:complexType name="PropertyDocumentType">
    <xsd:sequence>
      <xsd:element name="PortType" type="xsd:QName" minOccurs="0" maxOccurs="1"/>
      <xsd:element name="Properties" type="wsdai:DataAccessPropertyType" minOccurs="1" maxOccurs="1"/>
    </xsd:sequence>
  </xsd:complexType>

  <!-- the base of the substitution group used in factory messages -->
  <xsd:element name="PropertyDocument" type="wsdai:PropertyDocumentType"/>

  <!-- A list of property document names that can validly be passed into an  -->
  <!-- operation following the factory pattern -->
  <xsd:complexType name="PropertyDocumentTypeListType">
    <xsd:sequence>
      <xsd:element name="PropertyDocumentTypeQName" type="xsd:QName" minOccurs="1" maxOccurs="unbounded"/>
    </xsd:sequence>
  </xsd:complexType>

<!-- data management -->
    <!-- TBD -->

</xsd:schema>
```

Appendix B – WSDL

```
<?xml version="1.0" encoding="UTF-8"?>
<wsdl:definitions name="wsdai"
                  targetNamespace="http://www.ggf.org/namespaces/2004/09/WS-DAI"
                  xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/"
                  xmlns:xsd="http://www.w3.org/2001/XMLSchema"
                          xmlns:wsdai="http://www.ggf.org/namespaces/2004/09/WS-DAI">

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



<!-- WSDL TYPES #################################################### -->
       <wsdl:types>
              <!-- ######################## -->
              <!-- ### DAIS Base Schema ### -->
              <!-- ######################## -->
              <xsd:schema targetNamespace="http://www.ggf.org/namespaces/2004/09/WS-DAI"
                                 elementFormDefault="qualified">
                     <xsd:include schemaLocation="./wsdai_types_0.5.xsd" />


              <!-- ########################### -->
              <!-- ### DataService Schemas ### -->
              <!-- ########################### -->
                <xsd:element name="DataServiceDescription">
                  <xsd:complexType>
                    <xsd:sequence>
                        <!-- from wsdai - data description - properties of the data resource -->
                         <xsd:element ref="wsdai:Name" minOccurs="0" maxOccurs="1" />
                         <xsd:element ref="wsdai:Description" minOccurs="0" maxOccurs="1"/>

                         <!-- from wsdai - data access - properties controlling access behaviour -->
                         <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>

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

<!-- WSDL MESSAGES ############################################# -->

    <!-- none defined -->


</wsdl:definitions>
```