

Web Services Data Access and Integration – The XML Realization (WS-DAIX)

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 the 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: The XML Realization (WS-DAIX), presents a specification for a collection of data access interfaces for XML data resources, which extends interfaces defined in the Web Services Data Access and Integration document [WS-DAI].

Related DAIS specifications define how other 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

Abstract.....	1
1 Introduction.....	3
1.1 Specification Scope.....	3
1.2 Specification Organization	3
1.3 Interface Composition	3
2 Notational Conventions	3
3 Terminology.....	4
3.1 XML Data Resource.....	4
4 Concepts	4
4.1 Port Types	4
4.2 Relationships with other specifications	5
5 XMLCollection	6
5.1 XMLCollectionDescription	6
5.2 XMLDocumentDescription	7
5.3 XMLCollectionAccess	7
5.4 XMLCollectionFactory	10
6 XPath.....	10
6.1 XPathDescription	10
6.2 XPathAccess.....	11
6.3 XPathFactory	11
7 XUpdate.....	12
7.1 XUpdateDescription	12
7.2 XUpdateAccess.....	12
8 XQuery	13
8.1 XQueryDescription	13
8.2 XQueryAccess	14
8.3 XQueryFactory	15
9 XMLSequence	16
9.1 XMLSequenceDescription.....	16
9.2 XMLSequenceAccess	17
10 Mapping to WSRF	18
11 Security Considerations.....	18
12 Conclusion	18
13 Editor Information	18
14 Contributors	19
15 Acknowledgements.....	19
16 Intellectual Property Statement.....	19
17 Full Copyright Notice	19
18 References.....	20
A Appendix.....	21
A.1 XQueryAccess: WSDL Port Types	21
A.2 XQueryAccess: WSDL Messages and Types	21
A.3 XQueryAccess: XML Schema	22

1 Introduction

XML data access can play a central role for many types of Grid applications. By data access we mean the ability to retrieve, manipulate or insert data into an XML data resource.

This document presents a specification for a collection of data access interfaces for XML data resources. An XML data resource is a data source/sink, together with any associated management infrastructure, that can be queried or updated using XPath, XQuery, XUpdate or any other suitable XML query/update language. The interfaces are thus categorized according to the support they provide for:

- Data Description
- Data Access
- Data Factories

As such, this document should be read in conjunction with the generic *Web Services Data Access and Integration* document [WS-DAI], which defines the base interfaces that are extended in this document.

1.1 Specification Scope

The data access and integration set of specifications are being developed to expose data resources through web services, and form part of a broader activity within the Global Grid Forum to develop the Open Grid Services Architecture (OGSA) [OGSA]. It builds on the framework established by the WS-DAI document.

1.2 Specification Organization

This specification separates the functional model for a data service from its mapping to a particular web services infrastructure. As such, Section 4 explains the concepts of the model in the context of XML repositories. Sections 5, 6, 7, 8 and 9 present XMLCollection, XPath, XUpdate, XQuery and XMLSequence interfaces. A particular mapping of the functional model presented is made to the Web Services Resource Framework (WSRF) [WSRF] in Section 10. Section 11 discusses security. Section 12 draws conclusions.

1.3 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.

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, 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/>).

When patterns of messages are described the layout of the XML of each message is presented, as opposed to the XML schema. 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 only one instance of the element is expected.

This specification adopts the terminology defined in the WS-DAI document [WS-DAI]. In particular the terms data service, data resource and data set are used.

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
wsdai	http://www.ggf.org/namespaces/2004/09/WS-DAI
wsdaix	http://www.ggf.org/namespaces/2004/09/WS-DAIX
xqx	http://www.w3.org/2003/12/XQueryX
xsd	http://www.w3.org/2001/XMLSchema
xsi	http://www.w3.org/2001/XMLSchema-instance

3 Terminology

The model independent terminology, i.e., Data Resource, Data Service, Consumer and Data Set, is given in the *Web Services Data Access and Integration* document [WS-DAI].

3.1 XML Data Resource

An XML Data Resource is taken to mean any system that can act as a source or sink for XML data, together with its associated management infrastructure, that exhibits capabilities that are characteristic of XML repositories, e.g., can be queried using XQuery [XQuery] or updated using XUpdate [XUpdate] or any another suitable XML query/update language.

We assume that data in an XML repository is structured into collections and documents. Collections may be nested and contain other collections or documents. Collections are identified uniquely by a URI. Documents may be held within a collection and are identified by a name and – if applicable – by the collection URI.

4 Concepts

4.1 Port Types

DAIS classes its Port Types into four broad categories, which are defined in [WS-DAI], three of which are extended in this document to target XML data resources.

4.1.1 Data Description

Data Description contains XML structures that describe the *properties* of a data resource. The model independent properties presented in the WS-DAI document MUST be supported by any implementation of an XML Data Description. The property set is extended here to provide support for XML based data resources. There are two main extension points for XML data resources:

- *XMLCollectionDescription*: provides additional properties about an XML collection that a data service may represent.
- *XMLDocumentDescription*: provides additional properties about a particular instance of a document that a data service may represent. The description makes available properties about the structure representing an XML instance document as well as any other relevant data.

These are described in Sections **Error! Reference source not found.** to 9.

4.1.2 Data Access

Data Access operations allow XML data resources to be modified through insertion or updates, or queried through an appropriate language.

- *XMLCollectionAccess*: provides access to subcollections and documents in a collection.
- *XQueryAccess*: allows the evaluation of XQuery requests across a collection of XML documents.
- *XUpdateAccess*: allows XML documents to be updated using XUpdate.
- *XPathAccess*: allows the evaluation of XPath requests across a collection of XML documents.
- *XMLSequenceAccess*: provides access to a sequence of items, which are usually the results of an XQuery or XPath query.

These are covered in more detail in Sections 5 to 9.

4.1.3 Data Factory

The *DataFactory* operations allow data derived from XML data resources, usually the results of a query, to be represented by a data service. The specializations in this instance thus deal with the type of expression that can be passed to a *DataFactory* to expose the results in a meaningful fashion. *DataFactory* specializations are:

- *XMLCollectionFactory*: provides access to subcollections and documents in a collection.
- *XPathFactory*: provides access to the results of an XPath query.
- *XQueryFactory*: provides access to the results of an XQuery request.

These are covered in more detail in Sections 5, 6 and 8.

4.2 Relationships with other specifications

DAIS does not propose to provide its own query/update languages for XML based data resources. Instead, it acts as a conduit for existing XML based query and update languages to be conveyed to the appropriate data resources, in this instance XML based data resources or, for example, a relational data resource that supports XML type queries. As such, DAIS relies on existing XML based query and update languages. In this document, interface support is explicitly provided for languages based on the following standards:

- *XPath*: Version 1.0 is a W3C recommendation defining a language for addressing parts of an XML document [XPath]. There is work in progress to define a second version of XPath that is closely aligned with XQuery.
- *XUpdate*: is a language for updating XML documents [XUpdate]. XUpdate is a *de facto* standard not standardized by any of the main standardization bodies. It is still a working draft. Nevertheless it is supported by several of the XML DBMS products hence this specification defines interfaces for XUpdate.

- *XQuery*: proposes to provide a query (and update) language for XML data resources [XQuery]. XQuery is expected to be a standard soon.
- *XQueryX*: currently a W3C working draft [XQueryX] proposes an XML representation for the XQuery language.

The DAIS framework could be extended to encompass any new or emerging XML query/update standards by employing the patterns established in this document.

5 XMLCollection

5.1 XMLCollectionDescription

The metadata described in this section are associated with the current state of the XML data resource.

5.1.1 Collections

Describes the hierarchy of collections in an XML repository.

```
<xsd:element name="Collections" type="wsdaix:CollectionType"/>

<xsd:complexType name="CollectionType">
  <xsd:sequence>
    <xsd:element name="Collection" type="wsdaix:CollectionType"
      minOccurs="0" maxOccurs="unbounded"/>
  </xsd:sequence>
  <xsd:attribute name="name" type="xsd:anyURI"/>
</xsd:complexType>
```

5.1.2 Schemas

The locations and namespaces of the XML Schemas associated with the collections. Each document in the collection must conform to one or more of the XML Schemas contained in this informational property, if there are any schemas present.

```
<xsd:element name="Schemas" type="wsdaix:SchemaCollectionType"/>

<xsd:complexType name="SchemaCollectionType">
  <xsd:sequence>
    <xsd:element name="Collection" type="wsdaix:SchemaCollectionType"
      minOccurs="0" maxOccurs="unbounded"/>
    <xsd:element name="Schema" type="wsdaix:SchemaType"
      minOccurs="0" maxOccurs="unbounded"/>
  </xsd:sequence>
</xsd:complexType>

<xsd:complexType name="SchemaType">
  <xsd:attribute name="name" type="xsd:string" use="optional"/>
  <xsd:attribute name="schemaLocation" type="xsd:anyURI" use="required"/>
  <xsd:attribute name="namespace" type="xsd:anyURI" use="required"/>
</xsd:complexType>
```

5.1.3 DocumentNames

The set of names that uniquely identify each XML document belonging to the collection being described.

```
<xsd:element name="DocumentNames" type="wsdaix:DocumentCollectionType"/>

<xsd:complexType name="DocumentCollectionType">
```

```

<xsd:sequence>
  <xsd:element name="Collection" type="wsdaix:DocumentCollectionType"
    minOccurs="0" maxOccurs="unbounded"/>
  <xsd:element name="Document" type="xsd:string"
    minOccurs="0" maxOccurs="unbounded"/>
</xsd:sequence>
</xsd:complexType>

```

5.1.4 NumberOfDocuments

The number of documents in the data resource.

```
<xsd:element name="NumberOfDocuments" type="xsd:long"/>
```

5.2 XMLDocumentDescription

Data description, at the document level, works at a finer granularity. It provides information about a single document represented by a data service.

5.2.1 Schema

XML Schema that this document conforms to if the schema name is not null.

```

<xsd:element name="Schema" type="wsdaix:SchemaType"/>

<xsd:complexType name="SchemaType">
  <xsd:attribute name="name" type="xsd:string" use="optional"/>
  <xsd:attribute name="schemaLocation" type="xsd:anyURI" use="required"/>
  <xsd:attribute name="namespace" type="xsd:anyURI" use="required"/>
</xsd:complexType>

```

5.3 XMLCollectionAccess

The *XMLCollectionAccess* interface provides access to a collection of XML documents, with operations for adding, updating or removing documents.

5.3.1 Operations

5.3.1.1 XMLCollectionAccess::AddDocuments

Create new XML documents in the specified collections. An attempt should be made to add all the documents even if some additions fail. If there are schemas associated with a collection, added documents must validate with one of the schemas.

Input

- *RequestDocument*:
 - *Document*: For each document:
 - *Name*: the name of the new document.
 - *Collection*: the URI of the collection to which the document should be added.
 - *Data*: the content of the document.

Output

- *ResponseDocument*:
 - *Status*: A boolean for each document indicating whether it was successfully added.
 - *Fault (*)*: Any faults that occurred.
 - *Warning (*)*: Any warnings that occurred.

Fault(s)

- *Fault*: The operation failed.
 - *Message*: A description of the fault.

5.3.1.2 XMLCollectionAccess::RemoveDocuments

Remove a set of documents from the specified collection. An attempt should be made to remove all documents even if some removals fail.

Input

- *RequestDocument*:
 - *Names*: names of the documents to be removed.
 - *Collection*: the URI of the collection holding the documents.

Output

- *ResponseDocument*:
 - *Status*: A boolean for each document indicating whether it was successfully removed.
 - *Faults*: For each document, any faults that have occurred during the operation (e.g. the error message when a document could not be removed).
 - *Warnings*: For each document, any warnings that have occurred.

Fault(s)

- *Fault*: The operation failed.

5.3.1.3 XMLCollectionAccess::CreateSubcollection

Create a new subcollection of the current collection. This creates the named collection or throws a fault if an error occurred and the collection could not be created. It does not, however, return a reference to a service that is attached to the newly created resource.

Input

- *RequestDocument*:
 - *Name*: The URI of the new subcollection.

Output

- *None*.

Fault(s)

- *CollectionAlreadyExists*: a collection with the given URI already exists.
- *CreateSubcollectionFault*: The creation of a subcollection failed.
 - *Message*: A description of the fault.

5.3.1.4 XMLCollectionAccess::RemoveSubcollection

Remove a subcollection of the collection that is being represented. If the subcollection cannot be removed the operation must throw a fault.

Input

- *RequestDocument*:
 - *Name*: the URI of the collection to be removed.

Output

- *None*.

Fault(s)

- *NoSuchCollection*: the collection could not be found.
- *RemoveSubcollectionFault*: The requested operation failed.
 - *Message*: A description of the fault.

5.3.1.5 XMLCollectionAccess::AddSchema

Associate an XML schema with a collection. If a schema cannot be associated with the collection because the documents in the collection do not validate against the schema, the operation must throw a fault.

Input

- *Collection*: The URI of the collection this schema will be added to.
- *Name*: A name that uniquely identifies this schema within the collection.
- *Schema*: an XML Schema document.

Output

- *None*.

Faults(s)

- *SchemaAlreadyExists*.
- *SchemaInvalid*.
- *Fault*: Any other fault.
 - *Message*: The description of the fault.

5.3.1.6 XMLCollectionAccess::RemoveSchema

Remove an XML Schema from this collection.

Input

- *Collection*: The URI of the collection the schema will be removed from.
- *Name*: A name that uniquely identifies the schema within the collection.

Output

- *None*.

Faults(s)

- *SchemaDoesNotExist*.
- *Fault*: Any other fault.

5.3.1.7 XMLCollectionAccess::BulkLoad

Load structured data into a collection (including subcollections, documents and schemas).

Input

- *RequestDocument*: data including collection structure and resources.
 - *Collections*: the URIs of the collections to be created.
 - *Documents*:
 - *Name*: the name of the document.
 - *Collection*: the collection URI where this document should be inserted.
 - *Data*: the contents of the XML document.

Output

- *Report*: Any results of the bulk load operation.

Fault(s)

- *InvalidRequestDocument*: The format of the input data is not valid.
- *Fault*: The requested bulk load failed.
 - *Message*: A description of the fault.

5.4 XMLCollectionFactory

5.4.1 Operations

5.4.1.1 XMLCollectionFactory::XMLCollectionSelectionFactory

Returns the endpoint reference of a data service that represents a subcollection of the current collection. This service implements access or factory interfaces such as *XPathAccess/Factory*, *XQueryAccess/Factory*, *XUpdate* or *XMLCollectionAccess/Factory*.

Input

- *RequestDocument*:
 - *CollectionName*: the URI of the collection the new service should represent.
- *PropertiesDocument*: Initial values of the properties of the resulting Data Service.

Output

- *Reference*: endpoint reference of the service representing the subcollection.

Fault(s)

- *CollectionDoesNotExist*: A specified collection does not exist.
- *Fault*: any other fault.
 - *Message*: a description of the fault.

5.4.1.2 XMLCollectionFactory::XMLDocumentSelectionFactory

Returns an endpoint reference of a data service that represents an existing XML document in a collection. This service implements an Access interface such as *XPathAccess*, *XQueryAccess* or *XUpdateAccess*.

Input

- *RequestDocument* containing:
 - *Name*: the document identifier (within *Collection*).
 - *Collection*: the URI of the collection that holds the document.
- *PropertiesDocument*: Initial values of the properties of the resulting Data Service.

Output

- *Reference*: endpoint reference of the service.

Fault(s)

- *DocumentDoesNotExist*: The specified document could not be found.
- *Fault*: any other fault.
 - *Message*: a description of the fault

6 XPath

This section describes the properties and operations associated with XPath access to XML data resources.

6.1 XPathDescription

6.1.1 XPathVersion

The XPath versions that are supported.

```
<xsd:element name="XPathVersion">
  <xsd:simpleType>
    <xsd:restriction base="xsd:string">
      <xsd:enumeration value="1.0"/>
      <xsd:enumeration value="2.0"/>
    </xsd:restriction>
  </xsd:simpleType>
</xsd:element>
```

6.1.2 XPathQueryRequestFormat

The XML schema for the *Expression* element within the *RequestDocument*.

```
<xsd:element name=" XPathQueryRequestFormat">
  <xsd:simpleType>
    <xsd:restriction base="xsd:string">
      <xsd:enumeration value="XQueryX" />
    </xsd:restriction>
  </xsd:simpleType>
</xsd:complexType>
```

6.2 XPathAccess

This interface facilitates the evaluation of XPath queries across an XML resource or a collection of resources. The response document will contain the results of the query.

6.2.1 Overview

An XML Data Service may implement the XPathAccess operations and expose the *XPathDescription* properties. In this example a consumer uses the *XPathQuery* message to submit a *RequestDocument* formatted in XQueryX. The associated *XPathQueryResponse* message will contain a sequence of items.

6.2.2 Operations

6.2.2.1 XPathAccess::XPathQuery

Query an XML data resource or a collection of resources and return the result immediately.

Input

- *RequestDocument*: An XML document containing the XQuery request and parameters. The request must conform to the XQueryX schema.
- *ResponseFormat*: Format of the result data.
 - Output method: one of xml, html, xhtml, text.
 - Serialisation parameters (see [XQuerySerialization]).

Output

- *ResponseDocument*:
 - *Items (*)*: The results of the XPath query, serialized as specified by [XQuerySerialization].

Fault(s)

- *InvalidRequestDocument*: The request document is not a valid format.
- *XPathFault*: The requested XPath query failed during execution.
 - *Message*: Details of the fault.

6.3 XPathFactory

This interface facilitates the evaluation of XPath queries across an XML resource or a collection of resources. The response is made available via a data service.

6.3.1 Operations

6.3.1.1 XPathFactory::XPathQueryFactory

Returns the endpoint reference of a Data Service which represents the results of an XPath query. A document holding an XPath request is passed to this operation. The resulting reference provides access to the results of the query. This service implements the *XMLSequenceAccess* interface.

Input

- *RequestDocument*:
 - *Expression*: The XPath expression.
 - *Documents (*)*.
 - *Name*.
 - *Collection*: the URI of the collection that contains the document.
 - *Collection (?)*: the URI of the collection to be queried.
- *PropertiesDocument*: Initial values of the properties of the resulting Data Service, including the interface that is required.
 - *ResolveLinks*: true or false.
 - *XMLSerializationMethod*: xml, html, xhtml, text.
 - *XMLSerializationParameters*.

Output

- *Reference*: reference of a service implementing *XMLSequenceAccess*.

Fault(s)

- *InvalidRequestDocument*: The request document does not have a valid format.
- *XPathFault*: The requested XPath query failed.
 - *Message*: Details of the fault.

7 XUpdate

7.1 XUpdateDescription

7.1.1 XUpdateRequestFormat

The XML Schema for the XUpdate request.

7.1.2 XUpdateVersion

The version of XUpdate that is supported (current version based on the working draft document is 1.0).

```
<xsd:element name="XUpdateVersion">
  <xsd:simpleType>
    <xsd:restriction base="xsd:string">
      <xsd:enumeration value="1.0"/>
    </xsd:restriction>
  </xsd:simpleType>
</xsd:element>
```

7.2 XUpdateAccess

A service implementing XUpdateAccess will typically be associated with one or more XML resources (or a collection of XML resources) allowing the resources to be updated using XUpdate.

7.2.1 Overview

An XML Data Service may implement the XUpdateAccess operations and expose the *XUpdateDescription* properties. A consumer uses the *XUpdate* message to submit a *RequestDocument*. The associated *XUpdateResponse* message will contain an update count.

7.2.2 Operations

7.2.2.1 XUpdateAccess::XUpdate

Update an XML resource using XUpdate.

Input

- *RequestDocument*: An XUpdate request.

Output

- *ResponseDocument*:
 - *Count*: The number of modified nodes.

Fault(s)

- *InvalidRequestDocument*: The request document is not well-formed or invalid.
- *XUpdateFault*: The requested XUpdate operation failed during execution. Details of the fault are passed within the error description.
- *InvalidResponseFormat*: The requested responseFormat is invalid.

8 XQuery

8.1 XQueryDescription

8.1.1 XQueryExecuteRequestFormat

The XML schema for the request document (e.g. XQueryX schema).

```
<xsd:element name=" XQueryExecuteRequestFormat">
  <xsd:simpleType>
    <xsd:restriction base="xsd:string">
      <xsd:enumeration value="XQueryX" />
    </xsd:restriction>
  </xsd:simpleType>
</xsd:complexType>
```

8.1.2 XQueryVersion

The XQuery versions that are supported.

```
<xsd:element name="XQueryVersion">
  <xsd:simpleType>
    <xsd:restriction base="xsd:string">
      <xsd:enumeration value="1.0" />
    </xsd:restriction>
  </xsd:simpleType>
</xsd:element>
```

8.1.3 XMLSerializationMethod

A list of QNames of supported serialization methods (as defined in [XQuerySerialization]), i.e. *xml*, *html*, *xhtml*, *text*, or the QName of an implementation defined output method.

```
<xsd:element name="XMLSerializationMethod">
  <xsd:simpleType>
    <xsd:restriction base="xsd:QName">
      <xsd:enumeration value="xml" />
      <xsd:enumeration value="html" />
      <xsd:enumeration value="xhtml" />
      <xsd:enumeration value="text" />
      <xsd:enumeration value="wsdaix:ImplementationDefined" />
    </xsd:restriction>
  </xsd:simpleType>
</xsd:element>
```

/wsdaix:XMLSerializationParameters

Values of the serialization parameters as defined in [XQuerySerialization]:

- encoding
- cdata-section-elements

- doctype-system
- doctype-public
- escape-uri-attributes
- include-content-type
- indent
- media-type
- normalize-unicode
- omit-xml-declaration
- standalone
- undeclare-namespaces
- use-character-maps
- version

8.2 XQueryAccess

This interface supports XQuery requests across an XML data resource.

8.2.1 Overview

An XML Data Service may implement the XQueryAccess operations and expose the *XQueryDescription* properties. In this example a consumer uses the *XQueryExecute* message to submit a *RequestDocument* formatted in XQueryX. The associated *XQueryExecuteResponse* message will contain a sequence of items.

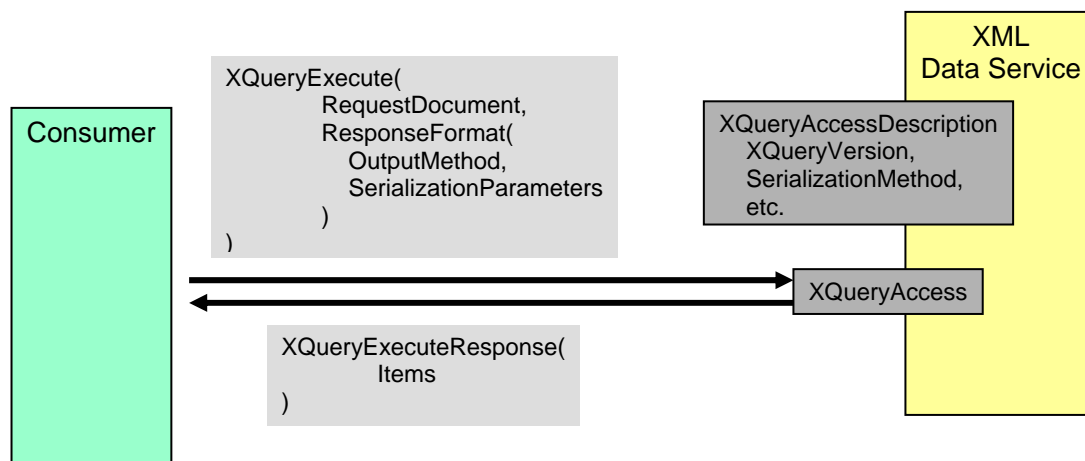


Figure 1: Overview – XQueryAccess

8.2.2 Operations

8.2.2.1 XQueryAccess::XQueryExecute

Input

- *RequestDocument*: An XML document containing the XQuery request and parameters. The request must conform to the XQueryX schema.
- *ResponseFormat*: Format of the result data.
 - Output method: one of xml, html, xhtml, text.
 - Serialisation parameters (see [XQuerySerialization]).

Output

- *ResponseDocument*:
 - *Items* (*): The results of the request, a sequence of items. The sequence is serialized as specified in *ResponseFormat*.

Fault(s)

- *InvalidRequestDocument*: The request document is not well-formed or invalid.
- *XQueryFault*: The requested XQuery operation failed during execution.
 - *Message*: Details of the fault.
- *InvalidResponseFormat*: The requested *ResponseFormat* is invalid.
- *Fault*: Any other fault.
 - *Message*: a description of the fault.

8.3 XQueryFactory

8.3.1 Overview

The example in **Error! Reference source not found.** presents a *XQueryFactory* interface. The *XQueryExecuteFactory* operation is used to construct the derived XQueryResponse Data Service. This service provides access to the *XMLSequence* resulting from a *XQuery* against the data resource. The *XMLSequence* is a subset or restriction of the data in the database and is presented in the form of a sequence of items. The *XMLSequence* could be stored in a database or decoupled from the database, but the important distinction here is that the data is represented as a sequence of items that does not implement the *XQueryAccess* portType. Instead, the XQuery Response Data Service presents the *XMLSequenceAccess* collection of operations that allows the *XMLSequence* to be retrieved but does not provide facilities for submitting XQuery expressions.

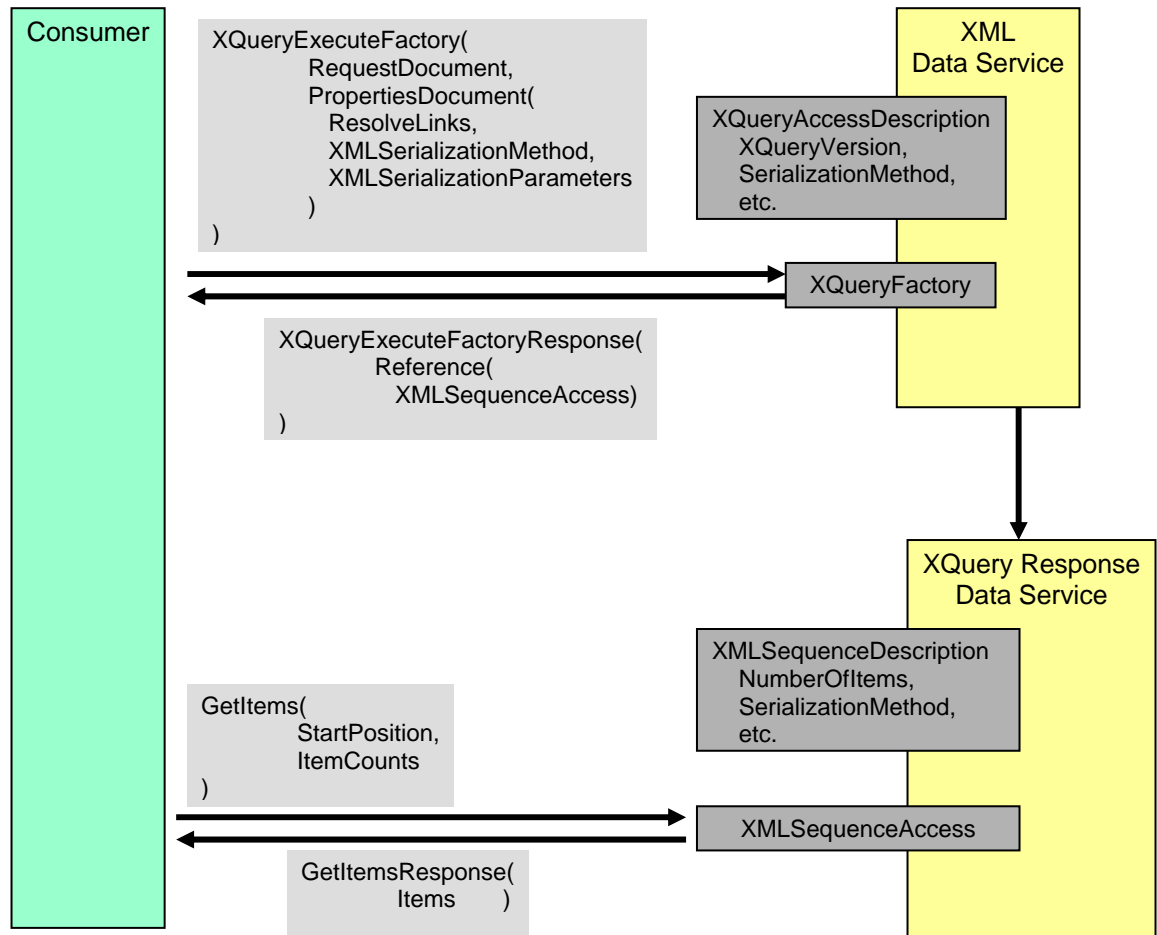


Figure 2: Overview – XQueryFactory

8.3.2 Operations

8.3.2.1 XQueryFactory::XQueryExecuteFactory.

Provide access to the results of an XQuery request. This is usually a service that implements the *XMLSequenceAccess* interface.

Input

- *RequestDocument*: An XML document containing the XQuery request and parameters. The request must conform to the XQueryX schema.
- *PropertiesDocument*: Initial values of the properties of the resulting Data Service.
 - *ResolveLinks*: true or false.
 - *XMLSerializationMethod*.
 - *XMLSerializationParameters*: Values of the serialization parameters as defined in [XQuerySerialization]:
 - encoding
 - cdata-section-elements
 - doctype-system
 - doctype-public
 - escape-uri-attributes
 - include-content-type
 - indent
 - media-type
 - normalize-unicode
 - omit-xml-declaration
 - standalone
 - undeclare-namespaces
 - use-character-maps
 - version

Output

- *Reference*: reference of a Data Service.

Fault(s)

- *InvalidRequestDocument*: The request document is not well-formed or invalid.
- *XQueryFault*: The requested XQuery operation failed during execution.
 - *Message*: Details of the fault.
- *Fault*: Any other fault.

9 XMLSequence

9.1 XMLSequenceDescription

9.1.1 NumberOfItems

The total number of items in the sequence.

```
<xsd:element name="NumberOfItems" type="xsd:long"/>
```

9.1.2 ResolveLinks

Indicates whether links should be resolved when serializing a document.

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


9.1.3 XMLSerializationMethod

A list of QNames of supported serialization methods (as defined in [XQuerySerialization]), i.e. *xml*, *html*, *xhtml*, or *text*.

```
<xsd:element name="XMLSerializationMethod">
  <xsd:simpleType>
    <xsd:restriction base="xsd:QName">
      <xsd:enumeration value="xml" />
      <xsd:enumeration value="html" />
      <xsd:enumeration value="xhtml" />
      <xsd:enumeration value="text" />
    </xsd:restriction>
  </xsd:simpleType>
</xsd:element>
```

9.1.4 XMLSerializationParameters

Values of the serialization parameters as defined in [XQuerySerialization].

9.2 XMLSequenceAccess

This interface provides access to a result sequence of an XQuery request. A sequence is an ordered collection of zero or more items. An item may be a node or an atomic value (see [XQueryDataModel]).

9.2.1 Overview

An XML Data Service may implement the XMLSequenceAccess operations and expose the *XMLSequenceDescription* properties. A consumer uses the *GetItems* message to retrieve a number of items from the sequence. It submits a *RequestDocument* containing the *StartPosition* and *ItemCount* parameters. The associated *GetItemsResponse* message will contain the requested items.

9.2.2 Operations

9.2.2.1 XMLSequenceAccess::GetItems

Returns a specified number of items.

Input

- *RequestDocument*:
 - *StartPosition*: The position of the first item to be returned. (Sequence starts with position 1).
 - *ItemCount*: The number of items to be returned.

Output

- *ResponseDocument*:
 - *Items*: The requested items, serialized as specified by the *SerializationMethod* and *SerializationParameters* property.

Fault(s)

- *InvalidStartPosition*: The start position is not valid.
- *InvalidCount*: Cannot return this number of items.
- *Fault*: A fault occurred while trying to retrieve the requested items.
 - *Message*: a description of the fault.

10 Mapping to WSRF

For a representative mapping to the Web Services Resource Framework (WSRF) proposal see the following sections:

- XQueryAccess and XQueryFactory:
 - WSDL Port Types – Appendix A.1.
 - WSDL Messages and Types – Appendix A.2.
 - XML Schema – Appendix A.3.

11 Security Considerations

The XML Realization of a data service will use standard Grid security mechanisms as specified by the 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.

12 Conclusion

This document has presented a specialization of the interfaces defined in the *WS Data Access and Integration* [WS-DAI] document providing the additional capabilities required to address XML based data resources. This is a work in progress and feedback is welcomed on this document.

13 Editor Information

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

Shannon Hastings,
Ohio State University,
333 W. Tenth Ave.,
Columbus OH, 43210,
USA.

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

Stephen Langella,
Ohio State University,
333 W. Tenth Ave.,
Columbus OH, 43210,
USA.

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.

14 Contributors

Malcolm Atkinson, NESC.
Dave Pearson, Oracle.
Greg Riccardi, Florida State University.

15 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, Vijay Dialani, Dieter Gawlick, Allen Luniewski, Sastry Malladi, Inderpal Narang, Steve Tuecke, Jay Unger, Paul Watson and Martin Westhead.

16 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.

17 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."

18 References

[OGSA]

I. Foster (Editor), D. Berry, A. Djaoui, A. Grimshaw, B. Horn, H. Kishimoto (Editor), F. Maciel, A. Savva (Editor), F. Siebenlist, R. Subramaniam, J. Treadwell, J. Von Reich. *The Open Grid Services Architecture, Version 1.0.*

[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.

[WS-DAI]

M. Antonioletti, M. Atkinson, S. Malaika, S. Laws, N. W. Paton D. Pearson and G. Riccardi. *Web Services Data Access and Integration (WS-DAI)*. DAIS-WG Informational Draft, 13th Global Grid Forum, 18th February 2005..

[WSRF]

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>.

[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>.

[XQuery]

S. Boag, D. Chamberlin, M. F. Fernández, D. Florescu, J. Robie and J. Siméon. *XQuery 1.0: An XML Query Language*, W3C Working. See: <http://www.w3.org/TR/xquery/>.

[XQueryDataModel]

M. Fernández, A. Malhotra, J. Marsh, N. Walsh. *XQuery 1.0 and XPath 2.0 Data Model*. W3C Working Draft 12 November 2003. See: <http://www.w3.org/TR/xpath-datamodel/>.

[XQuerySerialization]

M. Kay, N. Walsh, H. Zongaro. *XSLT 2.0 and XQuery 1.0 Serialization*. W3C Working Draft 12 November 2003. See <http://www.w3.org/TR/xslt-xquery-serialization/>.

[XQueryX]

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

[XUpdate]

A. Laux and L. Martin. *XUpdate Working Draft*, last release September 14, 2000. See: <http://xmldb-org.sourceforge.net/xupdate/xupdate-wd.html>.

A Appendix

This section presents a mapping of the XQueryAccess and XQueryFactory interfaces to the WSRF proposal. The remaining WSDL and XML Schema documents will be available soon.

A.1 XQueryAccess: WSDL Port Types

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

  <wsdl:import namespace="http://www.ggf.org/namespaces/2004/05/WS-DAIX"
    location="./wsdaix-types.wsdl"/>

  <wsdl:portType name="XQueryDataService">
    <wsdl:operation name="XQueryExecute">
      <wsdl:input message="wsdaix:XQueryExecuteRequest"/>
      <wsdl:output message="wsdaix:XQueryExecuteResponse"/>
    </wsdl:operation>
    <wsdl:operation name="XQueryExecuteFactory">
      <wsdl:input message="wsdaix:XQueryExecuteFactoryRequest"/>
      <wsdl:output message="wsdaix:XQueryExecuteFactoryResponse"/>
    </wsdl:operation>
  </wsdl:portType>

</wsdl:definitions>
```

A.2 XQueryAccess: WSDL Messages and Types

```
<?xml version="1.0" encoding="UTF-8"?>
<wsdl:definitions name="wsdaix"
  targetNamespace="http://www.ggf.org/namespaces/2004/09/WS-DAIX"
  xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/"
  xmlns:xsd="http://www.w3.org/2001/XMLSchema"
  xmlns:xqx="http://www.w3.org/2003/12/XQueryX"
  xmlns:wsdai="http://www.ggf.org/namespaces/2004/09/WS-DAI"
  xmlns:wsdaix="http://www.ggf.org/namespaces/2004/09/WS-DAIX">

  <xsd:schema targetNamespace="http://www.ggf.org/namespaces/2004/05/WS-DAI"
    elementFormDefault="qualified">
    <xsd:include schemaLocation="./wsdai-types-0.3.xsd"/>
  </xsd:schema>
  <xsd:schema targetNamespace="http://www.ggf.org/namespaces/2004/05/WS-DAIX"
    elementFormDefault="qualified">
    <xsd:include schemaLocation="./wsdaix-types.xsd"/>
    <xsd:element name="XQueryExecuteRequest">
      <xsd:complexType>
        <xsd:sequence>
          <xsd:any namespace="http://www.w3.org/2003/12/XQueryX"/>
        </xsd:sequence>
      </xsd:complexType>
    </xsd:element>
    <xsd:element name="XQueryExecuteResponse">
      <xsd:complexType>
        <xsd:sequence>
          <xsd:element ref="wsdai:Dataset" minOccurs="1" maxOccurs="1"/>
        </xsd:sequence>
      </xsd:complexType>
    </xsd:element>

  </xsd:schema>
```

```

<xsd:element name="XQueryExecuteFactoryRequest">
  <xsd:complexType>
    <xsd:sequence>
      <xsd:any namespace="http://www.w3.org/2003/12/XQueryX"
        minOccurs="1" maxOccurs="1"/>
      <xsd:element ref="wsdai:TermDocument"
        minOccurs="0" maxOccurs="1"/>
    </xsd:sequence>
  </xsd:complexType>
</xsd:element>
<!-- assumes that these messages result in a service/resource
  that contains an XML sequence -->
<xsd:element name="XQueryExecuteFactoryResponse">
  <xsd:complexType>
    <xsd:sequence>
      <xsd:element ref="wsa:EndPointReference"
        minOccurs="0" maxOccurs="1"/>
    </xsd:sequence>
  </xsd:complexType>
</xsd:element>
</xsd:schema>

<message name="XQueryExecuteRequest">
  <part name="XQueryExecuteRequest" element="wsdaix:XQueryExecuteRequest"/>
</message>
<message name="XQueryExecuteResponse">
  <part name="XQueryExecuteResponse"
    element="wsdaix:XQueryExecuteResponse"/>
</message>

<message name="XQueryExecuteFactoryRequest">
  <part name="XQueryExecuteFactoryRequest"
    element="wsdaix:XQueryExecuteFactoryRequest"/>
</message>
<message name="XQueryExecuteFactoryResponse">
  <part name="XQueryExecuteFactoryResponse"
    element="wsdaix:XQueryExecuteFactoryResponse"/>
</message>

```

A.3 XQueryAccess: XML Schema

```

<?xml version="1.0" encoding="UTF-8"?>
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema"
  xmlns:wsdaix="http://www.ggf.org/namespaces/2004/09/WS-DAIX"
  targetNamespace="http://www.ggf.org/namespaces/2004/09/WS-DAIX"
  elementFormDefault="qualified" attributeFormDefault="unqualified">

  <!-- Properties -->
  <xsd:element name="XQueryVersion">
    <xsd:simpleType>
      <xsd:restriction base="xsd:string">
        <xsd:enumeration value="1.0"/>
      </xsd:restriction>
    </xsd:simpleType>
  </xsd:element>

  <xsd:element name="XQueryExecuteRequestFormat"
    type="wsdaix:RequestFormatType"/>
  <xsd:complexType name="RequestFormatType">
    <xsd:sequence>
      <xsd:element name="schema" ref="xsd:schema"/>
    </xsd:sequence>
  </xsd:complexType>

```

```

<!-- Terms for the resulting Data Service -->
  <xsd:element name="CursorDirection">
    <xsd:simpleType>
      <xsd:restriction base="xsd:string">
        <xsd:enumeration value="ForwardOnly"/>
        <xsd:enumeration value="ForwardAndReverse"/>
      </xsd:restriction>
    </xsd:simpleType>
  </xsd:element>
  <xsd:element name="ResolveLinks" type="xsd:boolean" />
  <xsd:element name="XMLSerializationMethod">
    <xsd:simpleType>
      <xsd:restriction base="xsd:QName">
        <xsd:enumeration value="xml"/>
        <xsd:enumeration value="html"/>
        <xsd:enumeration value="xhtml"/>
        <xsd:enumeration value="text"/>
        <xsd:enumeration value="wsdaix:ImplementationDefined"/>
      </xsd:restriction>
    </xsd:simpleType>
  </xsd:element>
  <xsd:element name="XMLSerializationParameters">
    <xsd:complexType>
      <xsd:sequence>
        <xsd:element name="encoding" type="xsd:string"/>
        <xsd:element name="cdata-section-elements" type="xsd:string"/>
        <xsd:element name="doctype-system" type="xsd:string"/>
        <xsd:element name="doctype-public" type="xsd:string"/>
        <xsd:element name="escape-uri-attributes" type="xsd:string"/>
        <xsd:element name="include-content-type" type="xsd:string"/>
        <xsd:element name="indent" type="xsd:string"/>
        <xsd:element name="media-type" type="xsd:string"/>
        <xsd:element name="normalize-unicode" type="xsd:string"/>
        <xsd:element name="omit-xml-declaration" type="xsd:string"/>
        <xsd:element name="standalone" type="xsd:string"/>
        <xsd:element name="undeclare-namespaces" type="xsd:string"/>
        <xsd:element name="use-character-maps" type="xsd:string"/>
        <xsd:element name="version" type="xsd:string"/>
      </xsd:sequence>
    </xsd:complexType>
  </xsd:element>

<!-- ### XQueryAccess Term Documents ##### -->

  <xsd:complexType name="XQueryAccessTermDocumentType">
    <xsd:complexContent>
      <xsd:restriction base="wsdai:TermDocumentType">
        <xsd:sequence>
          <xsd:element name="PortType">
            <xsd:simpleType>
              <xsd:restriction base="xsd:QName">
                <xsd:enumeration value="wsdaix:XMLSequenceDataService"/>
              </xsd:restriction>
            </xsd:simpleType>
          </xsd:element>
          <xsd:element name="Terms" type="wsdaix:XQueryAccessTermsType"/>
        </xsd:sequence>
      </xsd:restriction>
    </xsd:complexContent>
  </xsd:complexType>
  <xsd:element name="XQueryAccessTermDocument"

```

```
        type="wsdaix:XQueryAccessTermDocumentType"
        substitutionGroup="wsdai:TermDocument" />

<!-- the terms -->
  <xsd:complexType name="XQueryAccessTermsType">
    <xsd:complexContent>
      <xsd:extension base="wsdai:DataAccessTermsType">
        <xsd:sequence>
          <xsd:element name="ResolveLinks" ref="wsdaix:ResolveLinks"/>
          <xsd:element name="CursorDirection"
            ref="wsdaix:CursorDirection"/>
          <xsd:element name="XMLSerializationMethod"
            ref="wsdaix:XMLSerializationMethod"/>
          <xsd:element name="XMLSerializationParameters"
            ref="wsdaix:XMLSerializationParameters"/>
        </xsd:sequence>
      </xsd:extension>
    </xsd:complexContent>
  </xsd:complexType>
</xsd:schema>
```