GFD-R-P.075
DAIS Working Group

Mario Antonioletti, University of Edinburgh
Shannon Hastings, Ohio State University
Amy Krause, University of Edinburgh
Stephen Langella, Ohio State University
Steven Lynden, University of Manchester
Simon Laws, IBM
Susan Malaika, IBM
Norman W Paton, University of Manchester
2 August 2006

Web Services Data Access and Integration – The XML Realization (WS-DAIX) Specification, Version 1.0

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 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]. The specification can be applied in regular web services environments or as part of a grid fabric.

Contents

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.

This document should be read in conjunction with the generic *Web Services Data Access and Integration* specification [WS-DAI], which defines base interfaces that are extended in this document to cater for XML data resources. These specifications have been developed for representing data resources as web services, and form part of a broader activity within the Global Grid Forum to develop the Open Grid Services Architecture (OGSA) [OGSA].

1.1 Specification Scope

The base interfaces and properties for data access services are described in the *Web Services Data Access and Integration* specification [WS-DAI]. This specification extends these interfaces to allow access to and provide descriptions of XML databases. XML data resources are assumed to consist of collections of XML documents that are accessed and modified using XPath, XQuery and/or XUpdate. Interfaces are provided for these types of languages in this specification. At some future point the specification may be extended to incorporate any other suitable XML-based query/update languages.

1.2 Specification Organization

The specification is described using the notational conventions and terminology defined in Sections 2 and 3. Section 4 explains the terminology and concepts of the model in the context of XML repositories. Sections 5 to 9 present XMLCollection, XQuery, XPath, XUpdate and XMLSequence interfaces respectively. A mapping of the functionality presented is made to WSDL in Section 10. Section 11 discusses security. Section 12 draws conclusions. The WSDL and XML Schema included in the document appendices should be taken as the normative interface and property descriptions of this DAIS specification.

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 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/>).

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

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

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

This specification generally adopts the terminology defined in the Open Grid Services Architecture Glossary of Terms [OGSA Glossary]. This terminology is extended in Section 3.

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

Prefix	Namespace
wsa	http://www.w3.org/2005/08/addressing
wsdai	http://www.ggf.org/namespaces/2005/12/WS-DAI
wsdaix	http://www.ggf.org/namespaces/2005/12/WS-DAIX
xsd	http://www.w3.org/2001/XMLSchema

3 Terminology

Model-independent terminology, i.e., data resource, data access service, consumer and data set, is given in the WS-DAI specification [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 XPath [XPath] or 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 documents and, optionally, collections. Documents are identified by a document name and – if held within a collection – by the collection URI. Collections are identified uniquely by a URI. Collections MAY be nested and contain other collections or documents as defined in [XQuery].

4 Concepts

4.1 Interfaces

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

This specification extends the base interfaces and corresponding properties defined in the WS-DAI specification to provide access to XML data resources. These data resources are considered to consist of collections of documents. To cater for this representation, the following interfaces are defined:

- XMLCollectionDescription: provides properties of an XML collection that a data access service may represent.

- XMLCollectionAccess: provides access to subcollections and documents in a collection.

Direct access to a data access service allows the results of a request to be delivered to the consumer directly in the response message. To cater for this mode of operation the following interfaces are defined for accessing an XML data resource using established query and update languages:

- 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.

Indirect access is supported through the use of the factory pattern. This allows data, usually the result of a query, to be accessed by way of a new service-managed data resource, and thus the data is not returned directly to the consumer. To cater for this mode of operation the following interfaces are defined:

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

To support access to the data resources resulting from the use of the factory pattern, additional interfaces and properties are defined, in particular:

- XMLSequenceDescription: provides properties of a sequence of XML data items.
- XMLSequenceAccess: provides access to a sequence of items, which are usually the result of an XQuery or XPath query.

These interfaces, and their corresponding properties, are specified in Sections 5 to 9.

4.2 Relationships with other specifications

WS-DAIX does not specify its own query/update languages for XML based data resources. Instead, it acts as a channel for existing XML query and update languages to be conveyed to the appropriate data resources, in this instance XML data resources or a data resource that supports XML type queries. In this document, interface support is 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 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 currently a W3C candidate recommendation.

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

5 XMLCollection

An XML collection collects together XML documents. Collections MAY be nested into hierarchies. Collections MAY use XML Schema documents to describe the structure of the documents they hold. An XML collection is typed if it has associated with it one or more XML Schema documents.

Each document in a typed XML collection must validate against one or more of the schemas associated with the collection holding the document. The XML Schema documents associated with a collection MUST be held by the data resource. Documents MUST NOT be validated against schemas held remotely. The XML Schema used to validate a given document SHOULD be determined by the QName or xsi:type attribute of the root element of the document. The schemas associated with a collection are identified by their target namespaces, which MUST be unique within any one collection. The following table describes the constraints applied to all documents in a typed XML collection.

QName / xsi:type attribute of the root element of the document	Constraint
None	The document cannot exist in a typed collection.
Matches the target namespace of an XML schema document associated with the collection	The document is validated against the schema with the corresponding target namespace. Only valid documents may exist in the collection.
Does not match the target namespace of any of the XML schemas documents associated with the collection.	The document cannot exist in a typed collection.

Documents contained within any subcollections MUST NOT be validated against schemas in a parent collection.

5.1 Static XMLCollection Description

The elements described in this section extend those defined in the WS-DAI specification. They are contained within an XMLCollectionPropertyDocument element that extends the PropertyDocument type defined in the WS-DAI specification.

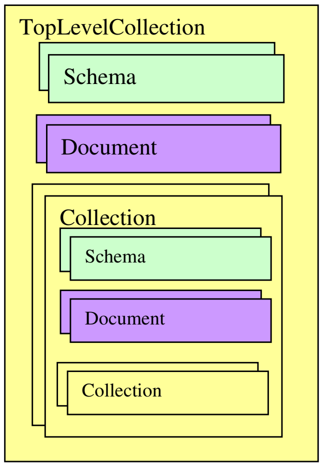

Figure 1: Collection, Schema, Document Hierarchies

Using the properties described below an XML data resource can represent a number of things, for example,

- A hierarchy of collections of documents.
- A single document.
- A single collection that is the top of a hierarchy of collections.

5.1.1 TopLevelCollection

```
<xsd:complexType name="SchemaDescriptionType">
  <xsd:sequence>
    <xsd:element name="targetNamespace" type="xsd:anyURI"/>
  </xsd:sequence>
</xsd:complexType>

<xsd:complexType name="DocumentDescriptionType">
  <xsd:attribute name="name" type="xsd:string"/>
</xsd:complexType>

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

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

/wsdaix:XMLPropertyDocument/wsdaix:TopLevelCollection

Describes the hierarchy of collections, schemas and documents in an XML data resource. This is the default collection that wraps top level documents and schema. When collections are not supported by the data resource this top level collection is the home for all documents.

/wsdaix:XMLPropertyDocument/wsdaix:TopLevelCollection/wsdaix:Collection

Collections in the XML data resource. When nested collections are supported then these collections in turn may contain collections.

/wsdaix:XMLPropertyDocument/wsdaix:TopLevelCollection/wsdaix:Schema

The XML schemas that documents in the containing collection are validated against.

/wsdaix:XMLPropertyDocument/wsdaix:TopLevelCollection/wsdaix:Document

The string name that uniquely identifies an XML document within the collection. The name is not necessarily unique if taking into account any subcollections that exist within the collection.

5.1.2 NumberOfDocuments

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

/wsdaix:XMLPropertyDocument/wsdaix:NumberOfDocuments

The total number of documents contained under a TopLevelCollection including any subcollections, and descendants, if these are present.

5.1.3 SupportsCollections

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

/wsdaix:XMLPropertyDocument/wsdaix:SupportsCollections
Indicates whether the data resource supports collections or not.

5.1.4 SupportsCollectionNesting

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

/wsdaix:XMLPropertyDocument/wsdaix:SupportsCollectionNesting
Indicates whether the nesting of collections is supported. If SupportsCollections is set to false then this property MUST also be false.

5.1.5 SupportsSchemas

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

/wsdaix:XMLPropertyDocument/wsdaix:SupportsSchemas
Indicates whether the data resource supports XML Schema based document validation.

5.2 Configurable XMLCollection Description

None.

5.3 Example XMLCollectionPropertyDocument

A non-normative example of an XMLCollection PropertyDocument follows:

```
<wsdaix:XMLCollectionPropertyDocument xmlns:...>
  <wsdai:DataResourceAbstractName>
     urn:dais:ds1
  </wsdai:DataResourceAbstractName>
  <wsdai:DataResourceManagement>ExternallyManaged</wsdai:DataResourceManagement>
  <wsdai:DatasetMap>
    <wsdai:MessageQName>wsdaix:AddDocuments</wsdai:MessageQName>
    <wsdai:DatasetFormatURI>
      http://www.ggf.org/schemas/somedocumentschema
    </wsdai:DatasetFormatURI>
  </wsdai:DatasetMap>
  <wsdai:ConfigurationMap>
    <wsdai:MessageQName>wsdaix:CollectionSelectionFactory</wsdai:MessageQName>
    <wsdai:PortTypeQName>wsdaix:XMLCollectionAccess</wsdai:PortTypeQName>
    <wsdai:ConfigurationDocumentQName>
         wsdaix:XMLCollectionConfigurationDocument
    </wsdai:ConfigurationDocumentQName>
    <DefaultConfigurationDocument>
      <wsdai:ConfigurationDocument>
        <wsdai:DataResourceDescription/>
        <wsdai:Readable>true</wsdai:Readable>
        <wsdai:Writeable>true</wsdai:Writeable>
        <wsdai:TransactionInitiation>NotSupported</wsdai:TransactionInitiation>
        <wsdai:TransactionIsolation>NotSupported</wsdai:TransactionIsolation>
        <wsdai:ChildSensitiveToParent>Sensitive</wsdai:ChildSensitiveToParent>
        <wsdai:ParentSensitiveToChild>Sensitive</wsdai:ParentSensitiveToChild>
      </wsdai:ConfigurationDocument>
    </DefaultConfigurationDocument>
  </wsdai:ConfigurationMap>
  <wsdai:DataResourceDescription/>
  <wsdai:Readable>true</wsdai:Readable>
  <wsdai:Writeable>true</wsdai:Writeable>
  <wsdai:ConcurrentAccess>true</wsdai:ConcurrentAccess>
  <wsdai:TransactionInitiation>NotSupported</wsdai:TransactionInitiation>
  <wsdai:TransactionIsolation>NotSupported</wsdai:TransactionIsolation>
  <wsdai:ChildSensitiveToParent>Sensitive</wsdai:ChildSensitiveToParent>
  <wsdai:ParentSensitiveToChild>Sensitive</wsdai:ParentSensitiveToChild>

  <wsdaix:TopLevelCollection name="urn:dais:collection1">

    <!-- This is a subcollection in collection1 -->
    <Collection name="urn:dais:collection2">
      <Document name="document1"/>
    </Collection>

     <!-- XML Schemata used to validate the contained documents->
    <Schema>
      <targetNamespace>
        http://www.ggf.org/schemas/somedocumentschema
      </targetNamespace>
    </Schema>

    <!-- Document belonging to collection1 -->
    <Document name="document1"/>
  </wsdaix:TopLevelCollection>

  <wsdaix:NumberOfDocuments>2</wsdaix:NumberOfDocuments>
  <wsdaix:SupportsCollections>true</wsdaix:SupportsCollections>
  <wsdaix:SupportsCollectionNesting>true</wsdaix:SupportsCollectionNesting>
  <wsdaix:SupportsSchemas>true</wsdaix:SupportsSchemas>

</wsdaix:XMLCollectionPropertyDocument>
```

5.4 XMLCollectionAccess

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

5.4.1 XMLCollectionAccess::GetCollectionPropertyDocument

Allows a copy of the CollectionPropertyDocument document to be retrieved.

Input

- GetCollectionPropertyDocumentRequest
  - DataResourceAbstractName – the abstract name of the resource from which the properties are to be obtained.

Output

- GetCollectionPropertyDocumentResponse
  - PropertyDocument – the properties described in the data description section.

Faults

- InvalidResourceNameFault – the supplied data resource abstract name is not known to the service.
- DataResourceUnavailableFault – the specified data resource is unavailable.
- NotAuthorizedFault – the consumer is not authorized to perform this operation at this time.
- ServiceBusyFault – the service is already processing a request and ConcurrentAccess is false.

5.4.2 XMLCollectionAccess::AddDocuments

Create new XML documents in the specified collection. 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

- AddDocumentsRequest
  - DataResourceAbstractName –the abstract name of the data resource to which the message is directed.
  - CollectionName? – an OPTIONAL parameter that contains the URI of the collection to which the documents will be added. If no collection name is provided the top level collection is assumed.
  - AddDocumentRequestWrapper+ – for each document:
    - DocumentName – the name of the new document.
    - Data – the content of the document.

Output

- AddDocumentsResponse
  - AddDocumentResponseWrapper+ – for each document:
    - DocumentName – name of document.
    - Response – result of the add operation. Possible values for this are:
      - Success.
      - Document not added as it does not validate against collection schema.
      - Document not added as an XML Schema document with the corresponding target namespace does not exist within the (typed) collection.
      - Document not added as the consumer is not authorized.
      - Existing document of same name is overwritten.
    - Detail – Details on the add operation including warnings and error messages such as validation constraint failures[1] and well formedness errors[2].

Faults

- InvalidResourceNameFault – the supplied resource name is not known to the service.
- DataResourceUnavailableFault – the specified data resource is unavailable.
- InvalidCollectionNameFault – the supplied collection name is not known to the XML resource.
  - ServiceBusyFault – The service is already processing a request and ConcurrentAccess is false.

5.4.3 XMLCollectionAccess::GetDocuments

Retrieve XML documents from the specified collection. An attempt should be made to retrieve all the documents even if some documents do not exist.

Input

- GetDocumentsRequest
  - DataResourceAbstractName –the abstract name of the data resource to which the message is directed.
  - CollectionName? – an OPTIONAL parameter that contains the URI of the collection from which the documents will be retrieved. This is used to specify a

[1] See http://www.w3.org/TR/xmlschema-1/#validation_failures.
[2] See http://www.w3.org/TR/xmlschema-1/#concepts-schemaConstraints.

subcollection from which the documents should be retrieved. If no collection name is provided the top level collection is assumed.
    - GetDocumentRequestWrapper+ – for each document:
        - DocumentName – the name of the document to be retrieved.

Output

- GetDocumentsResponse
    - GetDocumentResponseWrapper+ – for each document:
        - DocumentName – name of document.
        - Response – result of the retrieval operation. Possible values for this are:
            - Success.
            - Document not retrieved as it does not exist.
            - Document not retrieved as the consumer is not authorized
        - Data – the content of the document retrieved.

Faults

- InvalidResourceNameFault – the supplied resource name is not known to the service.
- DataResourceUnavailableFault – the specified data resource is unavailable.
- InvalidCollectionNameFault – the supplied collection name is not known to the XML resource.
- ServiceBusyFault – The service is already processing a request and ConcurrentAccess is false.

5.4.4 XMLCollectionAccess::RemoveDocuments

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

Input

- RemoveDocumentsRequest
    - DataResourceAbstractName – the abstract name of the data resource to which the message is directed.
    - CollectionName? – an OPTIONAL parameter that contains the URI of the collection from which the documents will be removed. If no collection name is provided the top level collection is assumed.
    - RemoveDocumentRequestWrapper
        - DocumentName – the name of each document to be removed.

Output

- RemoveDocumentsResponse
    - RemoveDocumentResponseWrapper+ – for each document the input document name and the result of the remove operation is recorded.
        - DocumentName – name of the document response applies to
        - Response – possible results are:
            - Success.
            - Document not removed as the consumer is not authorized.
            - Document not removed as the document name specified does not exist.
        - Detail – Details on the add operation such as warnings and error messages.

Faults

- InvalidResourceNameFault – the supplied resource name is not known to the service.
- DataResourceUnavailableFault – the specified data resource is unavailable.
- InvalidCollectionNameFault – the supplied collection name is not known to the XML resource.

- ServiceBusyFault – the service is already processing a request and ConcurrentAccess is false.

5.4.5 XMLCollectionAccess::CreateSubcollection

Create a new subcollection within a specified collection. This creates the named collection or throws a fault if an error occurrs and the collection cannot be created.

Input

- CreateSubcollectionRequest
    - DataResourceAbstractName – the abstract name of the data resource to which the message is directed.
    - CollectionName? – an OPTIONAL parameter that contains the URI of the collection to which the subcollection will be added. If no collection name is provided the top level collection is assumed.
    - SubcollectionName – the URI of the new subcollection.

Output

- CreateSubcollectionResponse
  This response will always indicate success. In error conditions a fault will be returned

Faults

- InvalidResourceNameFault – the supplied resource name is not known to the service.
- DataResourceUnavailableFault – the specified data resource is unavailable.
- InvalidCollectionNameFault – the supplied collection name is not known to the XML resource.
- NotAuthorizedFault – the consumer is not authorized to perform this operation at this time.
- ServiceBusyFault – the service is already processing a request and ConcurrentAccess is false.
- CollectionAlreadyExistsFault – The subcollection name specified already exists in the collection specified.

5.4.6 XMLCollectionAccess::RemoveSubcollection

Remove a collection given its name. This operation also removes all documents contained in the collection and – if applicable – subcollections and any associated schemas. If the collection cannot be removed the operation must throw a fault.

Input

- RemoveSubcollectionRequest
    - DataResourceAbstractName – the abstract name of the data resource to which the message is directed.
    - CollectionName? – an OPTIONAL parameter that contains the URI of the collection from which the subcollection will be removed. If no collection name is provided the top level collection is assumed.
    - SubcollectionName: the URI of the collection to be removed.

Output

- CreateSubcollectionResponse
  This response will always indicate success. In error conditions a fault will be returned.

Faults

- InvalidResourceNameFault – the supplied resource name is not known to the service.
- DataResourceUnavailableFault – the specified data resource is unavailable.
- InvalidCollectionNameFault – the supplied collection name is not known to the XML resource.
- NotAuthorizedFault – the consumer is not authorized to perform this operation at this time.

- ServiceBusyFault – the service is already processing a request and ConcurrentAccess is false.

5.4.7 XMLCollectionAccess::AddSchema

Associate an XML schema with a collection. If an attempt is made to add a schema with the same target namespace as a schema that already exists in the collection, a fault MUST be returned and the specified schema MUST NOT overwrite the schema that is already present. All documents in the collection MUST validate against a schema associated with the collection.

Input

- AddSchemaRequest
  - DataResourceAbstractName – the abstract name of the data resource to which the message is directed.
  - CollectionName? – an OPTIONAL parameter that contains the URI of the collection to which the schema will be added. If no collection name is provided the top level collection is assumed.
  - Schema – an XML Schema document.

Output

- AddSchemaResponse
  - This response will always indicate success. In the case of errors occuring a fault will be returned.

Faults

- InvalidResourceNameFault – the supplied resource name is not known to the service.
- DataResourceUnavailableFault – the specified data resource is unavailable.
- InvalidCollectionNameFault – the supplied collection name is not known to the XML resource.
- NotAuthorizedFault – the consumer is not authorized to perform this operation at this time.
- ServiceBusyFault – the service is already processing a request and ConcurrentAccess is false.
- SchemaAlreadyExistsFault – a schema with the same target namespace already exists and hence it has not been added.
- SchemaAdditionMakesDocumentsInvalidFault – adding the schema would make existing documents invalid and hence it has not been added.
- SchemaInvalidFault – the imported schema is invalid.

5.4.8 XMLCollectionAccess::RemoveSchema

Remove an XML Schema from a specified typed collection. When attempting to remove the schema the service MUST ensure that the documents in the collection validate against the remaining schemas. If this is not the case the operation MUST throw a fault and the schema MUST NOT be removed.

Input

- RemoveSchemaRequest
  - DataResourceAbstractName – the abstract name of the data resource to which the message is directed.
  - CollectionName – the collection from which a schema is to be removed. If no collection name is provided the top level collection is assumed.
  - SchemaNamespace – the target namespace of the schema that is to be removed.

Output

- RemoveSchemaResponse
  - This response will always indicate success. If an error occurs a fault is returned.

Faults

- InvalidResourceNameFault – the supplied resource name is not known to the service.
- DataResourceUnavailableFault – the specified data resource is unavailable.
- InvalidCollectionNameFault – the supplied collection name is not known to the XML resource.
- NotAuthorizedFault – the consumer is not authorized to perform this operation at this time.
- ServiceBusyFault – the service is already processing a request and ConcurrentAccess is false.
- SchemaDoesNotExistFault – the named schema could not be found.
- SchemaRemovalMakesDocumentsInvalidFault – removing the schema would make existing documents invalid and hence it has not been removed.
- SchemaRemovalMakesSchemaInvalidFault – removing the schema would make one or more existing schemas invalid, as the schema to have been removed is imported by other schemas.

5.4.9 XMLCollectionAccess::GetSchema

Retrieve an XML Schema from the specified collection. The schema is identified by its target namespace.

Input

- GetSchemaRequest
  - DataResourceAbstractName – the abstract name of the data resource to which the message is directed.
  - CollectionName – the collection from which a schema is to be removed. If no collection name is provided the top level collection is assumed.
  - SchemaNamespace – the target namespace of the schema that is to be retrieved.

Output

- GetSchemaResponse
  - The response contains the requested XML Schema document.

Faults

- InvalidResourceNameFault – the supplied resource name is not known to the service.
- DataResourceUnavailableFault – the specified data resource is unavailable.
- InvalidCollectionNameFault – the supplied collection name is not known to the XML resource.
- NotAuthorizedFault – the consumer is not authorized to perform this operation at this time.
- ServiceBusyFault – the service is already processing a request and ConcurrentAccess is false.
- SchemaDoesNotExistFault – a schema with the specified target namespace does not exist in the collection.

5.5 XMLCollectionFactory

5.5.1 XMLCollectionFactory::CollectionSelectionFactory

Returns the endpoint reference of a data resource that represents a specified collection. This factory operation can be used if access to a collection through a different port type, such as XPathAccess or XQueryAccess, if required.

Input

- CollectionSelectionFactoryRequest

  - DataResourceAbstractName – the abstract name of the resource the messages should be directed to.
  - PortTypeQName? – the QName of the portType through which the resulting data should be accessed. The QName value here MUST correspond to one that is specified in the ConfigurationMap property.
  - ConfigurationDocument? – a document that specifies the properties of the data resource that is to be used to access the data.
  - PreferredTargetService? – the EPR of the preferred service that is to act as the host for the new data resource.
  - CollectionName? – An OPTIONAL parameter that contains the URI of the collection from which the subcollection will be obtained. If no collection name is provided the top level collection is assumed.

Output

- CollectionSelectionFactoryResponse
  - DataResourceAddress – a data resource address.

Faults

- InvalidResourceNameFault – the supplied resource name is not known to the service.
- DataResourceUnavailableFault – the specified data resource is unavailable.
- InvalidCollectionNameFault – the supplied collection name is not known to the XML resource.
- NotAuthorizedFault – the consumer is not authorized to perform this operation at this time.
- ServiceBusyFault – the service is already processing a request and ConcurrentAccess is false.
- InvalidPortTypeQNameFault – the PortTypeQName specified is not in the collection defined by ConfigurationMap property.
- InvalidConfigurationDocumentFault – the ConfigurationDocument specified is not valid according to the ConfigurationDocumentQName when the ConfigurationMap is indexed by the specified PortTypeQName.

5.5.2 XMLCollectionFactory::DocumentSelectionFactory

Returns an endpoint reference to a data resource that represents an existing XML document in a collection. This factory operation could be used if access to an individual document through a different port type, such as XPathAccess or XQueryAccess, is required.

Input

- DocumentSelectionFactory
  - DataResourceAbstractName – the abstract name of the data resource to which the message is directed.
  - PortTypeQName? – the QName of the portType through which the resulting data should be accessed. The QName value here MUST correspond to one that is specified in the ConfigurationMap property.
  - ConfigurationDocument? – a document that specifies the properties of the data resource that is to be used to access the data.
  - PreferredTargetService? – the EPR of the preferred service that is to act as the host for the new data resource.
  - CollectionName? – an OPTIONAL parameter that contains the URI of the collection from which the subcollection will be obtained. If no collection name is provided the top level collection is assumed.
  - DocumentName – the document that is to be exposed through a service.

Output

- DocumentSelectionFactoryResponse
  - DataResourceAddress – a data resource address.

Faults

- InvalidResourceNameFault – the supplied resource name is not known to the service.
- DataResourceUnavailableFault – the specified data resource is unavailable.
- InvalidCollectionNameFault – the supplied collection name is not known to the XML resource.
- NotAuthorizedFault – the consumer is not authorized to perform this operation at this time.
- ServiceBusyFault – the service is already processing a request and ConcurrentAccess is false.
- DocumentDoesNotExistFault – the specified document could not be found.
- InvalidPortTypeQNameFault – the PortTypeQName specified is not in the collection defined by ConfigurationMap property.
- InvalidConfigurationDocumentFault – the ConfigurationDocument specified is not valid according to the ConfigurationDocumentQName when the ConfigurationMap is indexed by the specified PortTypeQName.

6 XQuery

6.1 Static XQuery Description

No extra static properties are defined for XQuery.

6.2 Configurable XQuery Description

No extra configurable properties are defined for XQuery.

6.3 Example PropertyDocument

No XQuery specific properties are defined so the structure of this document corresponds to the core properties document defined in the WS-DAI specification which may be retrieved using the interfaces defined in the same document.

```
<wsdai:PropertyDocument xmlns:...>
  <wsdai:DataResourceAbstractName>urn:dais:ds2
  </wsdai:DataResourceAbstractName>
  <wsdai:DataResourceManagement>
     ExternallyManaged
  </wsdai:DataResourceManagement>
  <wsdai:ParentDataResource>
    <wsa:Address>http://www.ggf.org/services/daisservice</wsa:Address>
    <wsa:ReferenceParameters>
      <DataResourceAbstractName>urn:dais:ds1</DataResourceAbstractName>
    </wsa:ReferenceParameters>
    <wsa:Metadata/>
  </wsdai:ParentDataResource>
  <wsdai:DatasetMap>
    <wsdai:MessageQName>wsdaix:XQueryExecute</wsdai:MessageQName>
    <wsdai:DatasetFormatURI>
       http://www.w3.org/TR/xml11/
    </wsdai:DatasetFormatURI>
  </wsdai:DatasetMap>
  <wsdai:DatasetMap>
    <wsdai:MessageQName>wsdaix:XQueryExecute</wsdai:MessageQName>
    <wsdai:DatasetFormatURI>
       http://www.w3.org/TR/xhtml11/
    </wsdai:DatasetFormatURI>
  </wsdai:DatasetMap>
  <wsdai:DatasetMap>
    <wsdai:MessageQName>wsdaix:XQueryExecute</wsdai:MessageQName>
    <wsdai:DatasetFormatURI>
```

```
      http://www.w3.org/TR/html401/
    </wsdai:DatasetFormatURI>
  </wsdai:DatasetMap>
  <wsdai:DatasetMap>
    <wsdai:MessageQName>wsdaix:XQueryExecute</wsdai:MessageQName>
    <wsdai:DatasetFormatURI>urn:myService:text</wsdai:DatasetFormatURI>
  </wsdai:DatasetMap>
  <wsdai:ConfigurationMap>
    <wsdai:MessageQName>wsdaix:XQueryExecuteFactory</wsdai:MessageQName>
    <wsdai:PortTypeQName>wsdaix:XMLSequenceAccesPT</wsdai:PortTypeQName>
    <wsdai:ConfigurationDocumentQName>
       wsdaix:XMLSequenceConfigurationDocument
    </wsdai:ConfigurationDocumentQName>
    <DefaultConfigurationDocument>
      <wsdai:ConfigurationDocument>
        <wsdai:DataResourceDescription/>
        <wsdai:Readable>true</wsdai:Readable>
        <wsdai:Writeable>true</wsdai:Writeable>
        <wsdai:TransactionInitiation>NotSupported</wsdai:TransactionInitiation>
        <wsdai:TransactionIsolation>NotSupported</wsdai:TransactionIsolation>
        <wsdai:ChildSensitiveToParent>Sensitive</wsdai:ChildSensitiveToParent>
        <wsdai:ParentSensitiveToChild>Sensitive</wsdai:ParentSensitiveToChild>
      </wsdai:ConfigurationDocument>
    </DefaultConfigurationDocument>
  </wsdai:ConfigurationMap>
  <wsdai:LanguageMap>
    <wsdai:MessageQName>wsdaix:XQueryExecute</wsdai:MessageQName>
    <wsdai:LanguageURI>http://www.w3.org/2005/XQueryX</wsdai:LanguageURI>
  </wsdai:LanguageMap>
  <wsdai:DataResourceDescription/>
  <wsdai:Readable>true</wsdai:Readable>
  <wsdai:Writeable>true</wsdai:Writeable>
  <wsdai:ConcurrentAccess>true</wsdai:ConcurrentAccess>
  <wsdai:TransactionInitiation>NotSupported</wsdai:TransactionInitiation>
  <wsdai:TransactionIsolation>NotSupported</wsdai:TransactionIsolation>
  <wsdai:ChildSensitiveToParent>Sensitive</wsdai:ChildSensitiveToParent>
  <wsdai:ParentSensitiveToChild>Sensitive</wsdai:ParentSensitiveToChild>
</wsdai:PropertyDocument>
```

6.4 XQueryAccess

This interface supports XQuery requests to be made to an XML data resource. An XML data access 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.

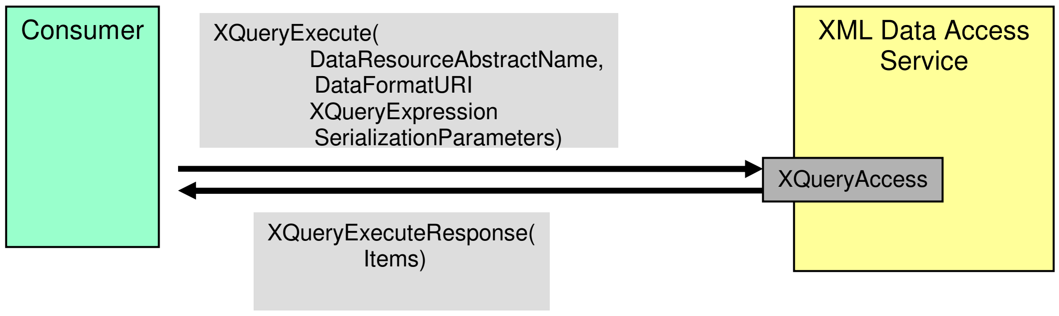

Figure 2: Overview – XQueryAccess

6.4.1 XQueryAccess::XQueryExecute

Input

- XQueryExecuteRequest
    - DataResourceAbstractName – the abstract name of the data resource to which the message is directed.
    - DatasetFormatURI? – the format used to represent the resulting XML in the corresponding XQueryExecuteResponse. If present, the value specified here MUST correspond to one found in the DatasetMap propery, or properties, at the data access service.
    - XQueryExpression – the request must conform to the XQueryX schema; this allows for the possibility that the query is expressed as a string.
    - SerializationParameters – define how resulting XML is serialized. See [XQuerySerialization].

Output

- XQueryExecuteResponse
    - Dataset
        - DatasetFormatURI – specifies how the sequence is serialized. The value MUST correspond to one specified in DatasetMap at the data access service (see the WS-DAI specification for details).
        - DatasetData – the results from the request, a sequence of items.

Faults

- InvalidResourceNameFault – the supplied resource name is not known to the service.
- DataResourceUnavailableFault – the specified data resource is unavailable.
- InvalidDatasetFormatFault – the supplied dataset format is not known to the service.
- InvalidExpressionFault – the supplied expression is not of a form known to the service.
- NotAuthorizedFault – the consumer is not authorized to perform this operation at this time.
- ServiceBusyFault – the service is already processing a request and ConcurrentAccess is false.
- XQueryFault – the requested XQuery operation failed during execution. This fault contains an OPTIONAL element which MAY be used to specify details regarding the failure of the XQuery query.

6.5 XQueryFactory

The example in Figure 3 presents an XQueryFactory interface. The XQueryExecuteFactory operation is used make the results of a query available through, potentially, a separate data access service; for example, a data access service which implements the XMLSequence port type. In this example the XMLSequence could be stored in a database or decoupled from the database, but the important distinction is that the data is represented as a sequence of items that does not implement the XQueryAccess portType and hence does not provide facilities for submitting XQuery expressions to this data.

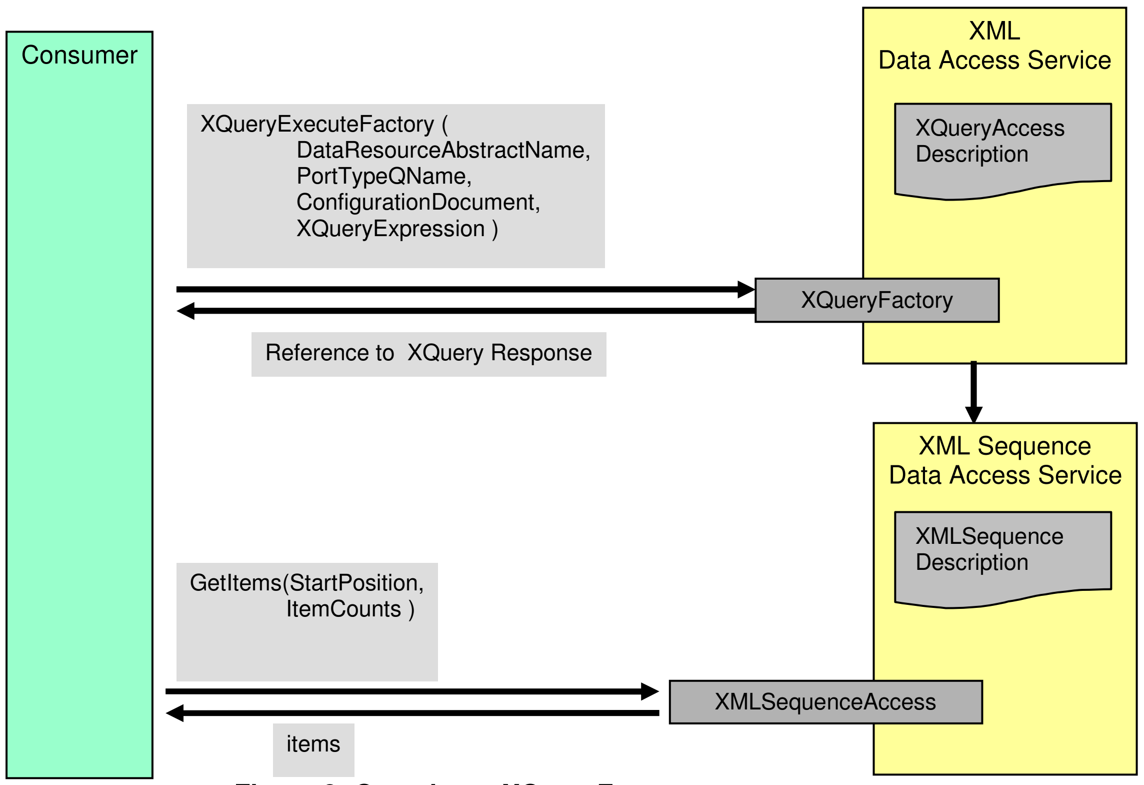
Consumer
XQueryExecuteFactory (DataResourceAbstractName, PortTypeQName, ConfigurationDocument, XQueryExpression)
Reference to XQuery Response
XML Data Access Service
XQueryAccess Description
XQueryFactory
XML Sequence Data Access Service
XMLSequence Description
GetItems(StartPosition, ItemCounts)
XMLSequenceAccess
items

Figure 3: Overview – XQueryFactory

6.5.1 XQueryFactory::XQueryExecuteFactory.

Provides access to the results of an XQuery request in a separate data access service. The XMLSequenceAccess interface SHOULD be supported as one of the interfaces that can be specified by the consumer using the PortTypeQName parameter.

Input

- XQueryExecuteFactoryRequest
    - DataResourceAbstractName – the abstract name of the data resource to which the message is directed.
    - PortTypeQName? – the QName of the portType through which the resulting data should be accessed. The QName value here MUST correspond to one that is specified in the ConfigurationMap property.
    - ConfigurationDocument? – a document that specifies the properties of the data resource that is to be used to access the data.
    - PreferredTargetService? – the EPR of the preferred service that is to act as the host for the new data resource.
    - XQueryExpression – an XQuery expression that must conform to the XQueryX schema.

Output

- XQueryExecuteFactoryResponse
    - DataResourceAddress – a data resource address.

Faults

- InvalidResourceNameFault – the supplied resource name is not known to the service.
- DataResourceUnavailableFault – the specified data resource is unavailable.

- InvalidPortTypeQNameFault – the PortTypeQName specified is not in the collection defined by ConfigurationMap property.
- InvalidConfigurationDocumentFault – the ConfigurationDocument specified is not valid according to the ConfigurationDocumentQName when the ConfigurationMap is indexed by the specified PortTypeQName.
- InvalidExpressionFault – the supplied expression is not of a form known to the service.
- NotAuthorizedFault – the consumer is not authorized to perform this operation at this time.
- ServiceBusyFault – the service is already processing a request and ConcurrentAccess is false.
- XQueryFault – the requested XQuery operation failed during execution. This fault contains an OPTIONAL element which MAY be used to specify details regarding the failure of the XQuery query.

7 XPath

This section describes the properties and operations associated with XPath access to XML data resources. They are contained within an XPathPropertyDocument element that extends the PropertyDocument type defined in the WS-DAI specification.

7.1 Static XPath Description

No extra static properties are defined for XPath.

7.2 Configurable XPath Description

No extra configurable properties are defined for XPath.

7.3 Example XPathPropertyDocument

No XPath specific properties are defined and so the structure of this document is defined by the core property document that may be retrieved using the appropriate interfaces defined in the WS-DAI specification.

7.4 XPathAccess

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

An XML data access service may implement the XPathAccess operations and expose the XPath property document. For example, a consumer may use the XPathExecute message to submit a request document containing an XPath expression formatted according to XQueryX [XQueryX]. The associated XPathExecuteResponse message will contain a sequence of items.

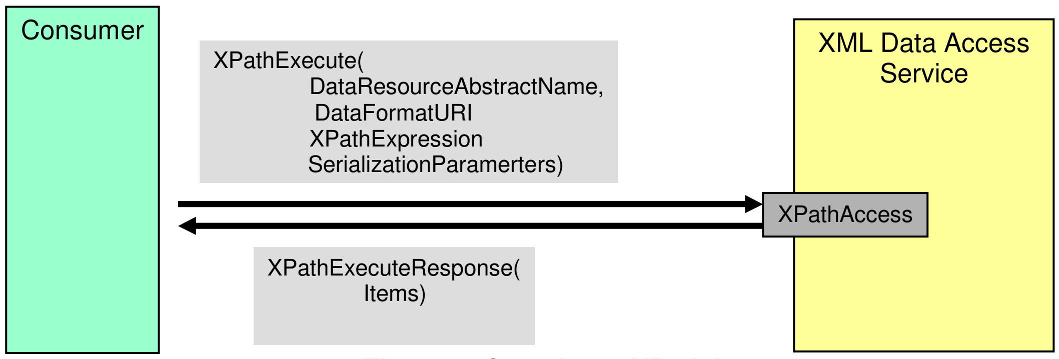

Figure 4: Overview – XPathAccess

7.4.1 XPathAccess::XPathExecute

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

Input

- XPathExecuteRequest
  - DataResourceAbstractName – the abstract name of the data resource to which the message is directed.
  - DatasetFormatURI? – the format used to represent the resulting XML in the corresponding XPathExecuteResponse. The value specified here MUST correspond to one found in the DatasetMap propery, or properties, at the data access service.
  - XPathExpression – the request must conform to the XPath portion of the XQueryX schema.
  - XMLSerializationParameters – define how resulting XML is serialized.

Output

- XPathExecuteResponse
  - Dataset
    - DatasetFormatURI – the format of the data being used to return the results.
    - Dataset – results of the XPath query, serialized as specified by [XQuerySerialization].

Faults

- InvalidResourceNameFault – the supplied resource name is not known to the service.
- DataResourceUnavailableFault – the specified data resource is unavailable.
- InvalidDatasetFormatFault – the supplied dataset format is not known to the service.
- InvalidExpressionFault – the supplied expression is not of a form known to the service.
- NotAuthorizedFault – the consumer is not authorized to perform this operation at this time.
- ServiceBusyFault – the service is already processing a request and ConcurrentAccess is false.
- XPathFault – the requested XPath query failed during execution. This fault contains an OPTIONAL element which MAY be used to specify details regarding the failure of the XPath query.

7.5 XPathFactory

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

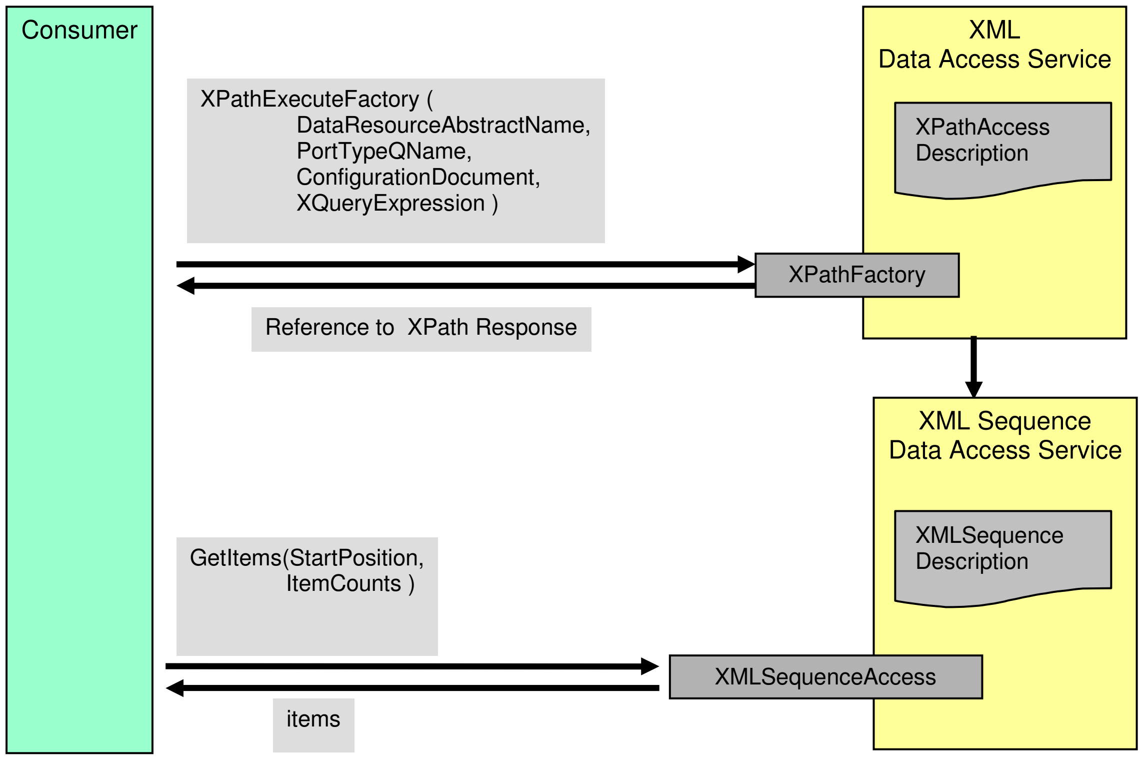
Consumer
XML Data Access Service
XPathAccess Description
XPathExecuteFactory (DataResourceAbstractName, PortTypeQName, ConfigurationDocument, XQueryExpression)
XPathFactory
Reference to XPath Response
XML Sequence Data Access Service
XMLSequence Description
GetItems(StartPosition, ItemCounts)
XMLSequenceAccess
items

Figure 5: Overview – XPathFactory

7.5.1 XPathFactory::XPathExecuteFactory

Returns the endpoint reference of a data access service that 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 separate service SHOULD implement the XMLSequenceAccess interface.

Input

- XPathExecuteFactoryRequest
  - DataResourceAbstractName – the abstract name of the data resource to which the message is directed.
  - PortTypeQName? – the QName of the portType through which the resulting data should be accessed. The value specified here MUST correspond to one found in the DatasetMap propery, or properties, at the data access service.
  - ConfigurationDocument? – a document that specifies the properties of the data resource that is to be used to access the data.
  - PreferredTargetService? – the EPR of the preferred service that is to act as the host for the new data resource
  - XPathExpression – the XPath expression.

Output

- XPathExecuteFactoryResponse
  - DataResourceAddress – a data resource address.

Faults

- InvalidResourceNameFault – the supplied resource name is not known to the service.
- DataResourceUnavailableFault – the specified data resource is unavailable.

- InvalidPortTypeQNameFault – The PortTypeQName specified is not in the collection defined by ConfigurationMap property.
- InvalidConfigurationDocumentFault - The ConfigurationDocument specified is not valid according to the ConfigurationDocumentQName when the ConfigurationMap is indexed by the specified PortTypeQName.
- InvalidExpressionFault – the supplied expression is not of a form known to the service.
- NotAuthorizedFault – the consumer is not authorized to perform this operation at this time.
- ServiceBusyFault – the service is already processing a request and ConcurrentAccess is false.
- XPathFault – the requested XPath operation failed during execution. This fault contains an OPTIONAL element which MAY be used to specify details regarding the failure of the XPath query.

8 XUpdate

8.1 Static XUpdate Description

No extra static properties are defined by XUpdate.

8.2 Configurable XUpdate Description

No extra configurable properties are defined by XUpdate.

8.3 Example XUpdatePropertyDocument

No XUpdate specific properties are defined and so the structure of this document is defined by the core property document, which may be retrieved using the appropriate interfaces defined in the WS-DAI specification.

8.4 XUpdateAccess

A service implementing XUpdateAccess allows the XML data resource to be updated using XUpdate structured expressions.

An XML data access 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.

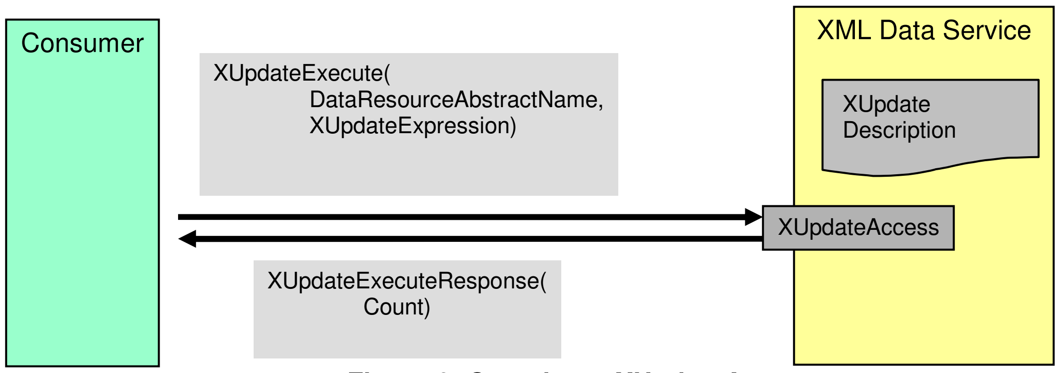
Consumer
XUpdateExecute(DataResourceAbstractName, XUpdateExpression)
XML Data Service
XUpdate Description
XUpdateAccess
XUpdateExecuteResponse(Count)

Figure 6: Overview – XUpdateAccess

8.4.1 XUpdateAccess::XUpdateExecute

Update an XML resource using XUpdate expressions. If the expression updates one or more documents in a typed collection, the service must ensure that all documents validate with the schemas associated with the collection following the execution of the XUpdate expression. If the update results in any validation failures, the XUpdate expression MUST NOT be applied and the service MUST respond with a fault.

Input

- XUpdateExecuteRequest
  - DataResourceAbstractName – the abstract name of the data resource to which the message is directed.
  - DatasetFormatURI? – the format used to represent the result of the XUpdate. This will generally be a count so this parameter SHOULD not be used for XUpdate expressions.
  - XUpdateExpression – the request must conform to XUpdate.

Output

- XUpdateExecuteResponse
  - Count - the number of modified nodes.

Faults

- InvalidResourceNameFault – the supplied resource name is not known to the service.
- DataResourceUnavailableFault – the specified data resource is unavailable.
- InvalidExpressionFault – the supplied expression is not of a form known to the service.
- NotAuthorizedFault – the consumer is not authorized to perform this operation at this time.
- ServiceBusyFault – the service is already processing a request and ConcurrentAccess is false.
- XUpdateFault – the requested XUpdate operation failed during execution. This fault contains an OPTIONAL element which MAY be used to specify details regarding the failure of the XUpdate statement.
- InvalidDatasetFormatFault – the supplied dataset format is not known to the service.
- XUpdateSchemaInvalidationFault – the requested XUpdate expression could not be executed as it makes one or more documents in a typed collection invalid.

9 XMLSequence

An XMLSequence represents the result sequence of an XQuery or XPath request. A sequence is an ordered collection of zero or more items. An item may be a node or an atomic value (see [XQueryDataModel]). An item is not necessarily an XML document or fragment and is identified by its position in the sequence.

9.1 Static XMLSequence Description

9.1.1 NumberOfItems

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

/wsdaix:XMLSequencePropertyDocument/wsdaix:NumberOfItems
The total number of items in the sequence.

9.2 Configurable XMLSequence Description

No extra configurable properties are defined for XMLSequence.

9.3 Example XMLSequencePropertyDocument

```
<wsdaix:XMLSequencePropertyDocument xmlns:...>
    <wsdai:DataResourceAbstractName>urn:dais:ds2</wsdai:DataResourceAbstractName>
    <wsdai:DataResourceManagement>ExternallyManaged</wsdai:DataResourceManagement>
    <wsdai:ParentDataResource>
        <wsa:Address>http://www.ggf.org/services/daisservice</wsa:Address>
        <wsa:ReferenceParameters>
            <DataResourceAbstractName>urn:dais:ds1</DataResourceAbstractName>
        </wsa:ReferenceParameters>
        <wsa:Metadata />
    </wsdai:ParentDataResource>
    <wsdai:DataResourceDescription />
    <wsdai:Readable>true</wsdai:Readable>
    <wsdai:Writeable>true</wsdai:Writeable>
    <wsdai:ConcurrentAccess>true</wsdai:ConcurrentAccess>
    <wsdai:TransactionInitiation>NotSupported</wsdai:TransactionInitiation>
    <wsdai:TransactionIsolation>NotSupported</wsdai:TransactionIsolation>
    <wsdai:ChildSensitiveToParent>Sensitive</wsdai:ChildSensitiveToParent>
    <wsdai:ParentSensitiveToChild>Sensitive</wsdai:ParentSensitiveToChild>
    <wsdaix:NumberOfItems>10</wsdaix:NumberOfItems>
</wsdaix:XMLSequencePropertyDocument>
```

9.4 XMLSequenceAccess

This interface provides access to an XMLSequence. An XML data access 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 in a serialized form.

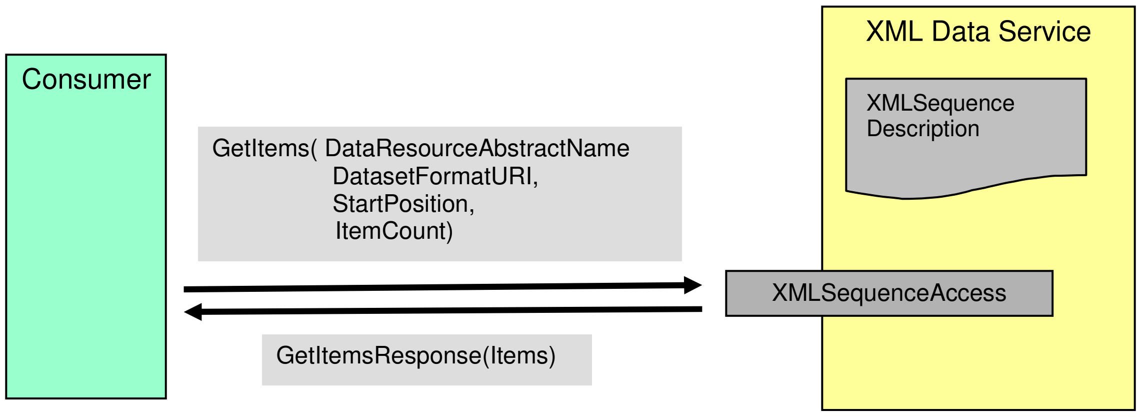

Figure 7: XMLSequence Access

9.4.1 XMLSequenceAccess::GetXMLSequencePropertyDocument

Allows a copy of the XMLSequencePropertyDocument document to be retrieved.

Input

- GetXMLSequencePropertyDocumentRequest
    - DataResourceAbstractName – the abstract name of the resource from whose are required.

Output

- GetXMLSequencePropertyDocumentResponse
    - PropertyDocument – the properties described in the data description section.

Faults

- InvalidResourceNameFault – the supplied data resource abstract name is not known to the service.
- ServiceBusyFault – the service is already processing a request and ConcurrentAccess is false.
- DataResourceUnavailableFault – the specified data resource is unavailable.
- NotAuthorizedFault – the consumer is not authorized to perform this operation at this time.

9.4.2 XMLSequenceAccess::GetItems

Returns a specified number of items.

Input

- GetItemsRequest
  - DataResourceAbstractName – the abstract name of the data resource to which the message is directed.
  - DatasetFormatURI? – the format used to represent the resulting XML in the corresponding XPathExecuteResponse. The value specified here MUST correspond to one found in the DatasetMap propery, or properties, at the data access service
  - StartPosition – the position of the first item to be returned. (Sequence starts with position 0).
  - ItemCount – the number of items to be returned.
  - XMLSerializationParameters – define how resulting XML is serialized.

Output

- GetItemsResponse
  - Dataset
    - DatasetFormatURI – the format of the data being used to return the results.
    - Dataset – the requested items, serialized as specified by the DocumentFormatURI and SerializationParameters parameters.

Faults

- InvalidResourceNameFault – the supplied resource name is not known to the service.
- DataResourceUnavailableFault – the specified data resource is unavailable.
- InvalidDatasetFormatFault – the supplied dataset format is not known to the service.
- NotAuthorizedFault – the consumer is not authorized to perform this operation at this time.
- ServiceBusyFault – the service is already processing a request and ConcurrentAccess is false.
- InvalidStartPositionFault – The start position is not valid.
- InvalidCountFault – Cannot return this number of items.

10 Mapping to WSDL

For a mapping to the WSDL proposal see the following sections:

- XMLCollection
    - XML Schema – Appendix A.1.
    - WSDL – Appendix A.2.
- XQuery
    - XML Schema – Appendix B.1.
    - WSDL – Appendix B.2.
- XPath
    - XML Schema – Appendix C.1.
    - WSDL – Appendix C.2.

- XUpdate
    - XML Schema – Appendix D.1.
    - WSDL – Appendix D.2.
- XMLSequence
    - XML Schema – Appendix E.1.
    - WSDL – Appendix E.2.

11 Security Considerations

The realizations of a grid data access service will use standard web service security mechanisms as specified by other standards bodies. The assumption is that these standards will also indicate how to make information related to authentication, authorization security, etc., available.

12 Conclusion

This specification has presented a specialization of the interfaces defined in the *WS Data Access and Integration* [WS-DAI] specification providing the additional capabilities required to address XML based data resources.

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.

Steven Lynden
School of Computer Science,
University of Manchester,
Oxford Road,
Manchester M13 9PL,
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.
Scott Oster, Ohio State University.
Dave Pearson, Oracle.
Greg Riccardi, Florida State University.

15 Acknowledgements

The Database Access and Integration Services (DAIS) Working Group of the Global Grid Forum has been active over several years, and many people have contributed to discussions within the group, 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

17 Full Copyright Notice

18 References

[OGSA]
Foster (Ed), H. Kishimoto (Ed), A. Savva (Ed), D. Berry, A. Djaoui, A. Grimshaw, B. Horn, F. Maciel, R. Subramaniam, J. Treadwell, J. Von Reich. The Open Grid Services Architecture, Version 1.0. Global Grid Forum. GFD-I.030. 29 January 2005. http://www.ggf.org/documents/GFD.30.pdf.

[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, A. Krause, S. Laws, S. Malaika, N. W. Paton D. Pearson and G. Riccardi. *Web Services Data Access and Integration – The Core (WS-DAI) Specification, Version 1.0.* Global Grid Forum. 2006.

[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 Candidate Recommendation 8 June 2006.. See: http://www.w3.org/TR/xquery/.

[XQueryDataModel]

M. Fernández, A. Malhotra, J. Marsh, M. Nagy and N. Walsh. *XQuery 1.0 and XPath 2.0 Data Model (XDM).* W3C Candidate Recommendation 8 June 2006. See: http://www.w3.org/TR/xpath-datamodel/.

[XQuerySerialization]
S. Boag, M. Kay, J. Tong, N. Walsh and H. Zongaro. *XSLT 2.0 and XQuery 1.0 Serialization.* W3C Candidate Recommendation 8 June 2006. See http://www.w3.org/TR/xslt-xquery-serialization/.

[XQueryX]
J. Melton and S. Muralidhar. *XML Syntax for XQuery 1.0 (XQueryX).* W3C Candidate Recommendation 8 June 2006, 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.

[OGSA Glossary]
J. Treadwell, *Open Grid Services Architecture Glossary of Terms*, GFD-I.044, January 25th 2005. http://www.ggf.org/documents/GFD.44.pdf.

Appendix A.1 – XMLCollection XML Schema

```
<?xml version="1.0" encoding="UTF-8" ?>
<xsd:schema targetNamespace="http://www.ggf.org/namespaces/2005/12/WS-DAIX" xmlns:xsd="http://www.w3.org/2001/XMLSchema"
    xmlns:wrs="http://java.sun.com/xml/ns/jdbc" xmlns:wsdai="http://www.ggf.org/namespaces/2005/12/WS-DAI"
    xmlns:wsdaix="http://www.ggf.org/namespaces/2005/12/WS-DAIX">
    <xsd:import namespace="http://www.ggf.org/namespaces/2005/12/WS-DAI" schemaLocation="./wsdai_core_types.xsd" />
    <!-- type for wrapping mixed content XML -->
    <xsd:complexType name="XMLWrapperType" mixed="true">
        <xsd:sequence>
            <xsd:any namespace="##any" />
        </xsd:sequence>
    </xsd:complexType>
    <!-- properties -->
    <!-- the descriptive properties for an XML schema document  -->
    <xsd:complexType name="SchemaDescriptionType">
        <xsd:sequence>
            <xsd:element name="targetNamespace" type="xsd:anyURI" />
        </xsd:sequence>
    </xsd:complexType>
    <!-- the descriptive properties for an XML document  -->
    <xsd:complexType name="DocumentDescriptionType">
        <xsd:attribute name="name" type="xsd:string" />
    </xsd:complexType>
    <!-- a structure respresenting the hierarchy of collections -->
    <!-- schemas and document names  -->
    <xsd:complexType name="CollectionType">
        <xsd:sequence>
            <xsd:element name="Collection" type="wsdaix:CollectionType" minOccurs="0" maxOccurs="unbounded" />
            <xsd:element name="Schema" type="wsdaix:SchemaDescriptionType" minOccurs="0" maxOccurs="unbounded" />
            <xsd:element name="Document" type="wsdaix:DocumentDescriptionType" minOccurs="0" maxOccurs="unbounded" />
        </xsd:sequence>
        <xsd:attribute name="name" type="xsd:anyURI" />
    </xsd:complexType>
    <xsd:element name="TopLevelCollection" type="wsdaix:CollectionType" />
    <!-- the number of documents in the data resource -->
    <xsd:element name="NumberOfDocuments" type="xsd:long" />
    <!-- indicates whether the XML resource supports collections -->
    <xsd:element name="SupportsCollections" type="xsd:boolean" />
    <!-- indicates whether the XML resource supports XML Schemas -->
    <xsd:element name="SupportsSchemas" type="xsd:boolean" />
    <!-- indicates whether the XML resource supports nested collections -->
    <xsd:element name="SupportsCollectionNesting" type="xsd:boolean" />
    <!-- property and configuration documents -->
    <xsd:complexType name="XMLCollectionPropertyDocumentType">
        <xsd:complexContent>
            <xsd:extension base="wsdai:PropertyDocumentType">
```

```
                <xsd:sequence>
                    <xsd:element ref="wsdaix:TopLevelCollection" />
                    <xsd:element ref="wsdaix:NumberOfDocuments" />
                    <xsd:element ref="wsdaix:SupportsCollections" />
                    <xsd:element ref="wsdaix:SupportsCollectionNesting" />
                    <xsd:element ref="wsdaix:SupportsSchemas" />
                </xsd:sequence>
            </xsd:extension>
        </xsd:complexContent>
    </xsd:complexType>
    <xsd:element name="XMLCollectionPropertyDocument" type="wsdaix:XMLCollectionPropertyDocumentType" />
    <!-- the properties used to configure an xml collection data resource -->
    <xsd:complexType name="XMLCollectionConfigurationDocumentType">
        <xsd:complexContent>
            <xsd:extension base="wsdai:ConfigurationDocumentType">
                <xsd:sequence>
                    <xsd:element ref="wsdaix:SupportsCollections" />
                    <xsd:element ref="wsdaix:SupportsCollectionNesting" />
                </xsd:sequence>
            </xsd:extension>
        </xsd:complexContent>
    </xsd:complexType>
    <xsd:element name="XMLCollectionConfigurationDocument" type="wsdaix:XMLCollectionConfigurationDocumentType"
        substitutionGroup="wsdai:ConfigurationDocument" />
</xsd:schema>
```

Appendix A.2 – XMLCollection WSDL

```
<?xml version="1.0" encoding="UTF-8"?>
<wsdl:definitions name="wsdaix"
                  targetNamespace="http://www.ggf.org/namespaces/2005/12/WS-DAIX"
                  xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/"
                  xmlns:xsd="http://www.w3.org/2001/XMLSchema"
                  xmlns:wsdai="http://www.ggf.org/namespaces/2005/12/WS-DAI"
                  xmlns:wsdaix="http://www.ggf.org/namespaces/2005/12/WS-DAIX">

<!-- WSDL IMPORTS #################################################### -->
    <wsdl:import location="./wsdai_core_porttypes.wsdl"
                 namespace="http://www.ggf.org/namespaces/2005/12/WS-DAI"/>

<!-- WSDL TYPES ###################################################### -->
    <wsdl:types>
      <xsd:schema targetNamespace="http://www.ggf.org/namespaces/2005/12/WS-DAIX"
                  elementFormDefault="qualified">
      <xsd:import namespace="http://www.ggf.org/namespaces/2005/12/WS-DAI"
                   schemaLocation="./wsdai_core_types.xsd" />
      <xsd:include schemaLocation="./wsdaix_xmlcollection_types.xsd" />
```

```
<!-- ############################ -->
<!-- ### Common Message Types ### -->
<!-- ############################ -->

  <xsd:complexType name="InvalidCollectionNameFaultType"/>
  <xsd:element name="InvalidCollectionNameFault" type="wsdaix:InvalidCollectionNameFaultType"/>

  <xsd:complexType name="SchemaDoesNotExistFaultType"/>
  <xsd:element name="SchemaDoesNotExistFault" type="wsdaix:SchemaDoesNotExistFaultType"/>

<!-- ################################## -->
<!-- ### GetDocuments Message Types ### -->
<!-- ################################## -->
    <xsd:element name="GetDocumentRequestWrapper">
      <xsd:complexType>
        <xsd:sequence>
          <xsd:element name="DocumentName" type="xsd:string"/>
        </xsd:sequence>
      </xsd:complexType>
    </xsd:element>

    <xsd:element name="GetDocumentResponseWrapper">
      <xsd:complexType>
        <xsd:sequence>
          <xsd:element name="DocumentName" type="xsd:string"/>
          <xsd:element name="Response">
              <xsd:simpleType>
                 <xsd:restriction base="xsd:token">
                   <xsd:enumeration value="Success"/>
                   <xsd:enumeration value="DocumentNotRetrieved-DocumentDoesNotExist"/>
                   <xsd:enumeration value="DocumentNotRetrieved-NotAuthorized"/>
                 </xsd:restriction>
              </xsd:simpleType>
           </xsd:element>
           <xsd:element name="Data" type="wsdaix:XMLWrapperType" minOccurs="0"/>
        </xsd:sequence>
      </xsd:complexType>
    </xsd:element>

    <xsd:element name="GetDocumentsRequest">
      <xsd:complexType>
        <xsd:complexContent>
          <xsd:extension base="wsdai:BaseRequestType">
            <xsd:sequence>
              <xsd:element name="CollectionName" type="xsd:anyURI" minOccurs="0" maxOccurs="1"/>
              <xsd:element ref="wsdaix:GetDocumentRequestWrapper"  minOccurs="1" maxOccurs="unbounded"/>
              </xsd:sequence>
            </xsd:extension>
          </xsd:complexContent>
        </xsd:complexType>
      </xsd:element>

      <xsd:element name="GetDocumentsResponse">
          <xsd:complexType>
              <xsd:sequence>
                  <xsd:element ref="wsdaix:GetDocumentResponseWrapper"  minOccurs="1" maxOccurs="unbounded"/>
              </xsd:sequence>
          </xsd:complexType>
      </xsd:element>


  <!-- ################################## -->
  <!-- ### AddDocuments Message Types ### -->
  <!-- ################################## -->
       <xsd:element name="AddDocumentRequestWrapper">
        <xsd:complexType>
          <xsd:sequence>
            <xsd:element name="DocumentName" type="xsd:string"/>
            <xsd:element name="Data" type="wsdaix:XMLWrapperType" minOccurs="0"/>
          </xsd:sequence>
        </xsd:complexType>
      </xsd:element>

      <xsd:element name="AddDocumentResponseWrapper">
        <xsd:complexType>
          <xsd:sequence>
            <xsd:element name="DocumentName" type="xsd:string"/>
            <xsd:element name="Response">
                <xsd:simpleType>
                   <xsd:restriction base="xsd:token">
                     <xsd:enumeration value="Success"/>
                     <xsd:enumeration value="DocumentNotAdded-DocumentDoesNotValidate"/>
                     <xsd:enumeration value="DocumentNotAdded-SchemaDoesNotExist"/>
                     <xsd:enumeration value="DocumentNotAdded-NotAuthorized"/>
                     <xsd:enumeration value="DocumentOfSameNameOverwritten"/>
                   </xsd:restriction>
                </xsd:simpleType>
             </xsd:element>
             <xsd:element name="Detail" type="wsdaix:XMLWrapperType" minOccurs="0"/>
          </xsd:sequence>
        </xsd:complexType>
      </xsd:element>
```

```
    <xsd:element name="AddDocumentsRequest">
      <xsd:complexType>
        <xsd:complexContent>
          <xsd:extension base="wsdai:BaseRequestType">
            <xsd:sequence>
              <xsd:element name="CollectionName" type="xsd:anyURI" minOccurs="0" maxOccurs="1"/>
              <xsd:element ref="wsdaix:AddDocumentRequestWrapper"  minOccurs="1" maxOccurs="unbounded"/>
            </xsd:sequence>
          </xsd:extension>
        </xsd:complexContent>
      </xsd:complexType>
    </xsd:element>

    <xsd:element name="AddDocumentsResponse">
        <xsd:complexType>
            <xsd:sequence>
                <xsd:element ref="wsdaix:AddDocumentResponseWrapper"  minOccurs="1" maxOccurs="unbounded"/>
            </xsd:sequence>
        </xsd:complexType>
    </xsd:element>

<!-- ##################################### -->
<!-- ### RemoveDocuments Message Types ### -->
<!-- #####################################0u45 ?->

     <xsd:element name="RemoveDocumentRequestWrapper">
      <xsd:complexType>
        <xsd:sequence>
          <xsd:element name="DocumentName" type="xsd:string"/>
        </xsd:sequence>
      </xsd:complexType>
    </xsd:element>

    <xsd:element name="RemoveDocumentResponseWrapper">
      <xsd:complexType>
        <xsd:sequence>
          <xsd:element name="DocumentName" type="xsd:string"/>
          <xsd:element name="Response">
              <xsd:simpleType>
                 <xsd:restriction base="xsd:token">
                   <xsd:enumeration value="Success"/>
                   <xsd:enumeration value="DocumentNotRemoved-NotAuthorized"/>
                   <xsd:enumeration value="DocumentNotRemoved-DocumentDoesNotExist"/>
                 </xsd:restriction>
              </xsd:simpleType>
           </xsd:element>
           <xsd:element name="Detail" type="wsdaix:XMLWrapperType" minOccurs="0"/>
          </xsd:sequence>
        </xsd:complexType>
      </xsd:element>

      <xsd:element name="RemoveDocumentsRequest">
        <xsd:complexType>
          <xsd:complexContent>
            <xsd:extension base="wsdai:BaseRequestType">
              <xsd:sequence>
                <xsd:element name="CollectionName" type="xsd:anyURI" minOccurs="0" maxOccurs="1"/>
                <xsd:element ref="wsdaix:RemoveDocumentRequestWrapper"  minOccurs="1" maxOccurs="unbounded"/>
              </xsd:sequence>
            </xsd:extension>
          </xsd:complexContent>
        </xsd:complexType>
      </xsd:element>

      <xsd:element name="RemoveDocumentsResponse">
          <xsd:complexType>
              <xsd:sequence>
                <xsd:element ref="wsdaix:RemoveDocumentResponseWrapper"  minOccurs="1" maxOccurs="unbounded"/>
              </xsd:sequence>
          </xsd:complexType>
      </xsd:element>

  <!-- ######################################### -->
  <!-- ### CreateSubCollection Message Types ### -->
  <!-- ######################################### -->

      <xsd:element name="CreateSubcollectionRequest">
        <xsd:complexType>
          <xsd:complexContent>
            <xsd:extension base="wsdai:BaseRequestType">
              <xsd:sequence>
                <xsd:element name="CollectionName" type="xsd:anyURI" minOccurs="0" maxOccurs="1"/>
                <xsd:element name="SubcollectionName" type="xsd:anyURI"/>
              </xsd:sequence>
            </xsd:extension>
          </xsd:complexContent>
        </xsd:complexType>
      </xsd:element>

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

      <xsd:complexType name="CollectionAlreadyExistsFaultType"/>
```

```
        <xsd:element name="CollectionAlreadyExistsFault" type="wsdaix:CollectionAlreadyExistsFaultType"/>

    <!-- ######################################### -->
    <!-- ### RemoveSubCollection Message Types ### -->
    <!-- ######################################### -->

        <xsd:element name="RemoveSubcollectionRequest">
          <xsd:complexType>
            <xsd:complexContent>
              <xsd:extension base="wsdai:BaseRequestType">
                <xsd:sequence>
                  <xsd:element name="CollectionName" type="xsd:anyURI" minOccurs="0" maxOccurs="1"/>
                  <xsd:element name="SubcollectionName" type="xsd:anyURI"/>
                </xsd:sequence>
              </xsd:extension>
            </xsd:complexContent>
          </xsd:complexType>
        </xsd:element>

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

        <xsd:complexType name="CollectionDoesNotExistFaultType"/>
        <xsd:element name="CollectionDoesNotExistFault" type="wsdaix:CollectionDoesNotExistFaultType"/>

    <!-- ############################### -->
    <!-- ### GetSchema Message Types ### -->
    <!-- ############################### -->
        <xsd:element name="GetSchemaRequest">
          <xsd:complexType>
            <xsd:complexContent>
              <xsd:extension base="wsdai:BaseRequestType">
                <xsd:sequence>
                  <xsd:element name="CollectionName" type="xsd:anyURI" minOccurs="0" maxOccurs="1"/>
                </xsd:sequence>
              </xsd:extension>
            </xsd:complexContent>
          </xsd:complexType>
        </xsd:element>

        <xsd:element name="GetSchemaResponse">
          <xsd:complexType>
            <xsd:sequence>
              <xsd:any namespace="http://www.w3.org/2001/XMLSchema"/>
            </xsd:sequence>
          </xsd:complexType>
        </xsd:element>

    <!-- ############################### -->
    <!-- ### AddSchema Message Types ### -->
    <!-- ############################### -->
        <xsd:element name="AddSchemaRequest">
          <xsd:complexType>
            <xsd:complexContent>
              <xsd:extension base="wsdai:BaseRequestType">
                <xsd:sequence>
                  <xsd:element name="CollectionName" type="xsd:anyURI" minOccurs="0" maxOccurs="1"/>
                  <xsd:any namespace="http://www.w3.org/2001/XMLSchema"/>
                </xsd:sequence>
              </xsd:extension>
            </xsd:complexContent>
          </xsd:complexType>
        </xsd:element>

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

        <xsd:complexType name="SchemaInvalidFaultType"/>
        <xsd:element name="SchemaInvalidFault" type="wsdaix:SchemaInvalidFaultType"/>

        <xsd:complexType name="SchemaAlreadyExistsFaultType"/>
        <xsd:element name="SchemaAlreadyExistsFault" type="wsdaix:SchemaAlreadyExistsFaultType"/>

        <xsd:complexType name="SchemaAdditionMakesDocumentsInvalidFaultType"/>
        <xsd:element name="SchemaAdditionMakesDocumentsInvalidFault" type="wsdaix:SchemaAdditionMakesDocumentsInvalidFaultType"/>

    <!-- ################################## -->
    <!-- ### RemoveSchema Message Types ### -->
    <!-- ################################## -->
        <xsd:element name="RemoveSchemaRequest">
          <xsd:complexType>
            <xsd:complexContent>
              <xsd:extension base="wsdai:BaseRequestType">
                <xsd:sequence>
                  <xsd:element name="CollectionName" type="xsd:anyURI" minOccurs="0" maxOccurs="1"/>
                  <xsd:element name="SchemaNamespace" type="xsd:anyURI"/>
                </xsd:sequence>
              </xsd:extension>
            </xsd:complexContent>
          </xsd:complexType>
        </xsd:element>
        <xsd:element name="RemoveSchemaResponse">
            <xsd:complexType/>
        </xsd:element>

        <xsd:complexType name="SchemaRemovalMakesDocumentsInvalidFaultType">
          <xsd:sequence>
            <xsd:element name="DocumentName" type="xsd:string"/>
          </xsd:sequence>
        </xsd:complexType>
        <xsd:element name="SchemaRemovalMakesDocumentsInvalidFault" type="wsdaix:SchemaRemovalMakesDocumentsInvalidFaultType"/>

        <xsd:complexType name="SchemaRemovalMakesSchemaInvalidFaultType">
          <xsd:sequence>
            <xsd:element name="SchemaNamespace" type="xsd:anyURI"/>
          </xsd:sequence>
        </xsd:complexType>
        <xsd:element name="SchemaRemovalMakesSchemaInvalidFault" type="wsdaix:SchemaRemovalMakesSchemaInvalidFaultType"/>

    <!-- ######################################### -->
    <!-- ###  CollectionSelectionFactory Types ### -->
    <!-- ######################################### -->
        <xsd:element name="CollectionSelectionFactoryRequest">
           <xsd:complexType>
            <xsd:complexContent>
              <xsd:extension base="wsdai:FactoryRequestType">
                <xsd:sequence>
                  <xsd:element name="CollectionName" type="xsd:anyURI" minOccurs="0" maxOccurs="1"/>
                </xsd:sequence>
              </xsd:extension>
            </xsd:complexContent>
          </xsd:complexType>
        </xsd:element>

        <xsd:element name="CollectionSelectionFactoryResponse" type="wsdai:DataResourceAddressType" />

    <!-- ####################################### -->
    <!-- ###  DocumentSelectionFactory Types ### -->
    <!-- ####################################### -->
        <xsd:element name="DocumentSelectionFactoryRequest">
          <xsd:complexType>
            <xsd:complexContent>
              <xsd:extension base="wsdai:FactoryRequestType">
                <xsd:sequence>
                  <xsd:element name="CollectionName" type="xsd:anyURI" minOccurs="0" maxOccurs="1"/>
                  <xsd:element name="DocumentName" type="xsd:string"/>
                </xsd:sequence>
              </xsd:extension>
            </xsd:complexContent>
          </xsd:complexType>
        </xsd:element>

        <xsd:element name="DocumentSelectionFactoryResponse" type="wsdai:DataResourceAddressType" />

        <xsd:complexType name="DocumentDoesNotExistFaultType"/>
        <xsd:element name="DocumentDoesNotExistFault" type="wsdaix:DocumentDoesNotExistFaultType"/>

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

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

    <wsdl:message name="InvalidCollectionNameFault">
      <wsdl:part name="InvalidCollectionNameFault"
                 element="wsdaix:InvalidCollectionNameFault"/>
    </wsdl:message>

    <wsdl:message name="SchemaDoesNotExistFault">
      <wsdl:part name="SchemaDoesNotExistFault"
                 element="wsdaix:SchemaDoesNotExistFault"/>
    </wsdl:message>

    <!-- ############################################## -->
    <!-- ### GetCollectionPropertyDocument Messages ### -->
    <!-- ############################################## -->
    <wsdl:message name="GetCollectionPropertyDocumentRequest">
      <wsdl:part name="GetCollectionPropertyDocumentRequest"
                 element="wsdai:GetDataResourcePropertyDocumentRequest" />
    </wsdl:message>

    <wsdl:message name="GetCollectionPropertyDocumentResponse">
      <wsdl:part name="GetCollectionPropertyDocumentResponse"
                 element="wsdaix:XMLCollectionPropertyDocument" />
    </wsdl:message>

    <!-- ############################# -->
    <!-- ### GetDocuments Messages ### -->
    <!-- ############################# -->
    <wsdl:message name="GetDocumentsRequest">
      <wsdl:part name="GetDocumentsRequest" element="wsdaix:GetDocumentsRequest"/>
    </wsdl:message>

    <wsdl:message name="GetDocumentsResponse">
      <wsdl:part name="GetDocumentsResponse" element="wsdaix:GetDocumentsResponse"/>
    </wsdl:message>
```

```
<!-- ############################ -->
<!-- ### AddDocuments Messages ### -->
<!-- ############################ -->
<wsdl:message name="AddDocumentsRequest">
  <wsdl:part name="AddDocumentsRequest" element="wsdaix:AddDocumentsRequest"/>
</wsdl:message>

<wsdl:message name="AddDocumentsResponse">
  <wsdl:part name="AddDocumentsResponse" element="wsdaix:AddDocumentsResponse"/>
</wsdl:message>

<!-- ############################### -->
<!-- ### RemoveDocuments Messages ### -->
<!-- ############################### -->
<wsdl:message name="RemoveDocumentsRequest">
  <wsdl:part name="RemoveDocumentsRequest" element="wsdaix:RemoveDocumentsRequest"/>
</wsdl:message>

<wsdl:message name="RemoveDocumentsResponse">
  <wsdl:part name="RemoveDocumentsResponse" element="wsdaix:RemoveDocumentsResponse"/>
</wsdl:message>

<!-- ################################### -->
<!-- ### CreateSubcollection Messages ### -->
<!-- ################################### -->
<wsdl:message name="CreateSubcollectionRequest">
  <wsdl:part name="CreateSubcollectionRequest" element="wsdaix:CreateSubcollectionRequest"/>
</wsdl:message>

<wsdl:message name="CreateSubcollectionResponse">
  <wsdl:part name="CreateSubcollectionResponse" element="wsdaix:CreateSubcollectionResponse"/>
</wsdl:message>

<wsdl:message name="CollectionAlreadyExistsFault">
  <wsdl:part name="CollectionAlreadyExistsFault"
             element="wsdaix:CollectionAlreadyExistsFault"/>
</wsdl:message>

<!-- ################################### -->
<!-- ### RemoveSubcollection Messages ### -->
<!-- ################################### -->
<wsdl:message name="RemoveSubcollectionRequest">
  <wsdl:part name="RemoveSubcollectionRequest" element="wsdaix:RemoveSubcollectionRequest"/>
</wsdl:message>

<wsdl:message name="RemoveSubcollectionResponse">
```

```
      <wsdl:part name="RemoveSubcollectionResponse" element="wsdaix:RemoveSubcollectionResponse"/>
    </wsdl:message>

    <wsdl:message name="CollectionDoesNotExistFault">
      <wsdl:part name="CollectionDoesNotExistFault"
                 element="wsdaix:CollectionDoesNotExistFault"/>
    </wsdl:message>

    <!-- ########################## -->
    <!-- ### AddSchema Messages ### -->
    <!-- ########################## -->
    <wsdl:message name="AddSchemaRequest">
      <wsdl:part name="AddSchemaRequest" element="wsdaix:AddSchemaRequest"/>
    </wsdl:message>

    <wsdl:message name="AddSchemaResponse">
      <wsdl:part name="AddSchemaResponse" element="wsdaix:AddSchemaResponse"/>
    </wsdl:message>

    <wsdl:message name="SchemaAlreadyExistsFault">
      <wsdl:part name="SchemaAlreadyExistsFault"
                 element="wsdaix:SchemaAlreadyExistsFault"/>
    </wsdl:message>

    <wsdl:message name="SchemaInvalidFault">
      <wsdl:part name="SchemaInvalidFault"
                 element="wsdaix:SchemaInvalidFault"/>
    </wsdl:message>

    <wsdl:message name="SchemaAdditionMakesDocumentsInvalidFault">
      <wsdl:part name="SchemaAdditionMakesDocumentsInvalidFault"
                 element="wsdaix:SchemaAdditionMakesDocumentsInvalidFault"/>
    </wsdl:message>

    <!-- ############################# -->
    <!-- ### RemoveSchema Messages ### -->
    <!-- ############################# -->
    <wsdl:message name="RemoveSchemaRequest">
      <wsdl:part name="RemoveSchemaRequest" element="wsdaix:RemoveSchemaRequest"/>
    </wsdl:message>

    <wsdl:message name="RemoveSchemaResponse">
      <wsdl:part name="RemoveSchemaResponse" element="wsdaix:RemoveSchemaResponse"/>
    </wsdl:message>

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

```
                element="wsdaix:SchemaRemovalMakesDocumentsInvalidFault"/>
    </wsdl:message>

    <wsdl:message name="SchemaRemovalMakesSchemaInvalidFault">
      <wsdl:part name="SchemaRemovalMakesSchemaInvalidFault"
                element="wsdaix:SchemaRemovalMakesSchemaInvalidFault"/>
    </wsdl:message>

    <!-- ############################# -->
    <!-- ### GetSchema Messages ### -->
    <!-- ############################# -->
    <wsdl:message name="GetSchemaRequest">
      <wsdl:part name="GetSchemaRequest" element="wsdaix:GetSchemaRequest"/>
    </wsdl:message>

    <wsdl:message name="GetSchemaResponse">
      <wsdl:part name="GetSchemaResponse" element="wsdaix:GetSchemaResponse"/>
    </wsdl:message>

    <!-- ########################################### -->
    <!-- ### CollectionSelectionFactory Messages ### -->
    <!-- ########################################### -->
    <wsdl:message name="CollectionSelectionFactoryRequest">
      <wsdl:part name="CollectionSelectionFactoryRequest" element="wsdaix:CollectionSelectionFactoryRequest"/>
    </wsdl:message>

    <wsdl:message name="CollectionSelectionFactoryResponse">
      <wsdl:part name="CollectionSelectionFactoryResponse" element="wsdaix:CollectionSelectionFactoryResponse"/>
    </wsdl:message>

    <!-- ######################################### -->
    <!-- ### DocumentSelectionFactory Messages ### -->
    <!-- ######################################### -->
    <wsdl:message name="DocumentSelectionFactoryRequest">
      <wsdl:part name="DocumentSelectionFactoryRequest" element="wsdaix:DocumentSelectionFactoryRequest"/>
    </wsdl:message>

    <wsdl:message name="DocumentSelectionFactoryResponse">
      <wsdl:part name="DocumentSelectionFactoryResponse" element="wsdaix:DocumentSelectionFactoryResponse"/>
    </wsdl:message>

    <wsdl:message name="DocumentDoesNotExistFault">
      <wsdl:part name="DocumentDoesNotExistFault"
                element="wsdaix:DocumentDoesNotExistFault"/>
    </wsdl:message>

<!-- WSDL PORT TYPES ############################################### -->
```

```
    <wsdl:portType name="XMLCollectionAccessPT">

        <wsdl:operation name="GetCollectionPropertyDocument">
          <wsdl:input  name="GetCollectionPropertyDocumentRequest"
                       message="wsdaix:GetCollectionPropertyDocumentRequest" />
          <wsdl:output name="GetCollectionPropertyDocumentResponse"
                       message="wsdaix:GetCollectionPropertyDocumentResponse" />
          <wsdl:fault  name="InvalidResourceNameFault"
                       message="wsdai:InvalidResourceNameFault" />
          <wsdl:fault name="DataResourceUnavailableFault"
                       message="wsdai:DataResourceUnavailableFault" />
          <wsdl:fault message="wsdai:NotAuthorizedFault"
                       name="NotAuthorizedFault"/>
          <wsdl:fault message="wsdai:ServiceBusyFault"
                       name="ServiceBusyFault" />
        </wsdl:operation>

        <wsdl:operation name="AddDocuments">
          <wsdl:input message="wsdaix:AddDocumentsRequest"/>
          <wsdl:output message="wsdaix:AddDocumentsResponse"/>
          <wsdl:fault  name="InvalidResourceNameFault"
                       message="wsdai:InvalidResourceNameFault" />
          <wsdl:fault name="DataResourceUnavailableFault"
                       message="wsdai:DataResourceUnavailableFault" />
          <wsdl:fault  name="InvalidCollectionNameFault"
                       message="wsdaix:InvalidCollectionNameFault" />
          <wsdl:fault message="wsdai:ServiceBusyFault"
                       name="ServiceBusyFault" />
        </wsdl:operation>

        <wsdl:operation name="GetDocuments">
          <wsdl:input message="wsdaix:GetDocumentsRequest"/>
          <wsdl:output message="wsdaix:GetDocumentsResponse"/>
          <wsdl:fault  name="InvalidResourceNameFault"
                       message="wsdai:InvalidResourceNameFault" />
          <wsdl:fault name="DataResourceUnavailableFault"
                       message="wsdai:DataResourceUnavailableFault" />
          <wsdl:fault  name="InvalidCollectionNameFault"
                       message="wsdaix:InvalidCollectionNameFault" />
          <wsdl:fault message="wsdai:ServiceBusyFault"
                       name="ServiceBusyFault" />
        </wsdl:operation>

        <wsdl:operation name="RemoveDocuments">
          <wsdl:input message="wsdaix:RemoveDocumentsRequest"/>
          <wsdl:output message="wsdaix:RemoveDocumentsResponse"/>
          <wsdl:fault  name="InvalidResourceNameFault"
```

```
                  message="wsdai:InvalidResourceNameFault" />
      <wsdl:fault name="DataResourceUnavailableFault"
                  message="wsdai:DataResourceUnavailableFault" />
      <wsdl:fault  name="InvalidCollectionNameFault"
                  message="wsdaix:InvalidCollectionNameFault" />
      <wsdl:fault message="wsdai:ServiceBusyFault"
                  name="ServiceBusyFault" />
    </wsdl:operation>

    <wsdl:operation name="CreateSubcollection">
      <wsdl:input message="wsdaix:CreateSubcollectionRequest"/>
      <wsdl:output message="wsdaix:CreateSubcollectionResponse"/>
      <wsdl:fault  name="InvalidResourceNameFault"
                  message="wsdai:InvalidResourceNameFault" />
      <wsdl:fault name="DataResourceUnavailableFault"
                  message="wsdai:DataResourceUnavailableFault" />
      <wsdl:fault  name="InvalidCollectionNameFault"
                  message="wsdaix:InvalidCollectionNameFault" />
      <wsdl:fault message="wsdai:NotAuthorizedFault"
                  name="NotAuthorizedFault"/>
      <wsdl:fault message="wsdai:ServiceBusyFault"
                  name="ServiceBusyFault" />
      <wsdl:fault  name="CollectionAlreadyExistsFault"
                  message="wsdaix:CollectionAlreadyExistsFault" />
    </wsdl:operation>

    <wsdl:operation name="RemoveSubcollection">
      <wsdl:input message="wsdaix:RemoveSubcollectionRequest"/>
      <wsdl:output message="wsdaix:RemoveSubcollectionResponse"/>
      <wsdl:fault  name="InvalidResourceNameFault"
                  message="wsdai:InvalidResourceNameFault" />
      <wsdl:fault name="DataResourceUnavailableFault"
                  message="wsdai:DataResourceUnavailableFault" />
      <wsdl:fault  name="InvalidCollectionNameFault"
                  message="wsdaix:InvalidCollectionNameFault" />
      <wsdl:fault message="wsdai:NotAuthorizedFault"
                  name="NotAuthorizedFault"/>
      <wsdl:fault message="wsdai:ServiceBusyFault"
                  name="ServiceBusyFault" />
    </wsdl:operation>

    <wsdl:operation name="AddSchema">
      <wsdl:input message="wsdaix:AddSchemaRequest"/>
      <wsdl:output message="wsdaix:AddSchemaResponse"/>
      <wsdl:fault  name="InvalidResourceNameFault"
                  message="wsdai:InvalidResourceNameFault" />
      <wsdl:fault name="DataResourceUnavailableFault"
```

```
                    message="wsdai:DataResourceUnavailableFault" />
      <wsdl:fault  name="InvalidCollectionNameFault"
                    message="wsdaix:InvalidCollectionNameFault" />
      <wsdl:fault message="wsdai:NotAuthorizedFault"
                    name="NotAuthorizedFault"/>
      <wsdl:fault message="wsdai:ServiceBusyFault"
                    name="ServiceBusyFault" />
      <wsdl:fault  name="SchemaAlreadyExistsFault"
                    message="wsdaix:SchemaAlreadyExistsFault" />
      <wsdl:fault  name="SchemaInvalidFault"
                    message="wsdaix:SchemaInvalidFault" />
      <wsdl:fault  name="SchemaAdditionMakesDocumentsInvalidFault"
                    message="wsdaix:SchemaAdditionMakesDocumentsInvalidFault" />
    </wsdl:operation>

    <wsdl:operation name="GetSchema">
      <wsdl:input message="wsdaix:GetSchemaRequest"/>
      <wsdl:output message="wsdaix:GetSchemaResponse"/>
      <wsdl:fault  name="InvalidResourceNameFault"
                    message="wsdai:InvalidResourceNameFault" />
      <wsdl:fault name="DataResourceUnavailableFault"
                    message="wsdai:DataResourceUnavailableFault" />
      <wsdl:fault  name="InvalidCollectionNameFault"
                    message="wsdaix:InvalidCollectionNameFault" />
      <wsdl:fault message="wsdai:NotAuthorizedFault"
                    name="NotAuthorizedFault"/>
      <wsdl:fault message="wsdai:ServiceBusyFault"
                    name="ServiceBusyFault" />
      <wsdl:fault  name="SchemaDoesNotExistFault"
                    message="wsdaix:SchemaDoesNotExistFault" />
    </wsdl:operation>

    <wsdl:operation name="RemoveSchema">
      <wsdl:input message="wsdaix:RemoveSchemaRequest"/>
      <wsdl:output message="wsdaix:RemoveSchemaResponse"/>
      <wsdl:fault  name="InvalidResourceNameFault"
                    message="wsdai:InvalidResourceNameFault" />
      <wsdl:fault name="DataResourceUnavailableFault"
                    message="wsdai:DataResourceUnavailableFault" />
      <wsdl:fault  name="InvalidCollectionNameFault"
                    message="wsdaix:InvalidCollectionNameFault" />
      <wsdl:fault message="wsdai:NotAuthorizedFault"
                    name="NotAuthorizedFault"/>
      <wsdl:fault message="wsdai:ServiceBusyFault"
                    name="ServiceBusyFault" />
      <wsdl:fault  name="SchemaRemovalMakesDocumentsInvalidFault"
                    message="wsdaix:SchemaRemovalMakesDocumentsInvalidFault" />
```

```
          <wsdl:fault  name="SchemaRemovalMakesSchemaInvalidFault"
                       message="wsdaix:SchemaRemovalMakesSchemaInvalidFault" />
          <wsdl:fault  name="SchemaDoesNotExistFault"
                       message="wsdaix:SchemaDoesNotExistFault" />
        </wsdl:operation>

    </wsdl:portType>

    <wsdl:portType name="XMLCollectionFactoryPT">
        <wsdl:operation name="CollectionSelectionFactory">
          <wsdl:input message="wsdaix:CollectionSelectionFactoryRequest"/>
          <wsdl:output message="wsdaix:CollectionSelectionFactoryResponse"/>
          <wsdl:fault  name="InvalidResourceNameFault"
                       message="wsdai:InvalidResourceNameFault" />
          <wsdl:fault name="DataResourceUnavailableFault"
                       message="wsdai:DataResourceUnavailableFault" />
          <wsdl:fault  name="InvalidCollectionNameFault"
                       message="wsdaix:InvalidCollectionNameFault" />
          <wsdl:fault message="wsdai:NotAuthorizedFault"
                       name="NotAuthorizedFault"/>
          <wsdl:fault message="wsdai:ServiceBusyFault"
                       name="ServiceBusyFault" />
          <wsdl:fault message="wsdai:InvalidConfigurationDocumentFault"
                       name="InvalidConfigurationDocumentFault" />
          <wsdl:fault message="wsdai:InvalidPortTypeQNameFault"
                       name="InvalidPortTypeQNameFault" />
        </wsdl:operation>

        <wsdl:operation name="DocumentSelectionFactory">
          <wsdl:input message="wsdaix:DocumentSelectionFactoryRequest"/>
          <wsdl:output message="wsdaix:DocumentSelectionFactoryResponse"/>
          <wsdl:fault  name="InvalidResourceNameFault"
                       message="wsdai:InvalidResourceNameFault" />
          <wsdl:fault name="DataResourceUnavailableFault"
                       message="wsdai:DataResourceUnavailableFault" />
          <wsdl:fault  name="InvalidCollectionNameFault"
                       message="wsdaix:InvalidCollectionNameFault" />
          <wsdl:fault message="wsdai:NotAuthorizedFault"
                       name="NotAuthorizedFault"/>
          <wsdl:fault message="wsdai:ServiceBusyFault"
                       name="ServiceBusyFault" />
          <wsdl:fault  name="DocumentDoesNotExistFault"
                       message="wsdaix:DocumentDoesNotExistFault" />
          <wsdl:fault message="wsdai:InvalidConfigurationDocumentFault"
                       name="InvalidConfigurationDocumentFault" />
          <wsdl:fault message="wsdai:InvalidPortTypeQNameFault"
                       name="InvalidPortTypeQNameFault" />
        </wsdl:operation>

    </wsdl:portType>
</wsdl:definitions>
```

Appendix B.1 – XQuery XML Schema

```
<?xml version="1.0" encoding="UTF-8" ?>
<xsd:schema targetNamespace="http://www.ggf.org/namespaces/2005/12/WS-DAIX" xmlns:xsd="http://www.w3.org/2001/XMLSchema"
    xmlns:wrs="http://java.sun.com/xml/ns/jdbc" xmlns:wsdai="http://www.ggf.org/namespaces/2005/12/WS-DAI"
    xmlns:wsdaix="http://www.ggf.org/namespaces/2005/12/WS-DAIX">
    <xsd:import namespace="http://www.ggf.org/namespaces/2005/12/WS-DAI" schemaLocation="./wsdai_core_types.xsd" />
    <!-- properties -->
    <xsd:simpleType name="YesNoType">
        <xsd:restriction base="xsd:string">
            <xsd:enumeration value="yes" />
            <xsd:enumeration value="no" />
        </xsd:restriction>
    </xsd:simpleType>
    <xsd:simpleType name="YesNoOmitType">
        <xsd:restriction base="xsd:string">
            <xsd:enumeration value="yes" />
            <xsd:enumeration value="no" />
            <xsd:enumeration value="omit" />
        </xsd:restriction>
    </xsd:simpleType>
    <xsd:element name="XMLSerializationParameters">
        <xsd:complexType>
            <xsd:sequence>
                <xsd:element name="byte-order-mark" type="wsdaix:YesNoType" />
                <xsd:element name="cdata-section-elements">
                    <xsd:simpleType>
                        <xsd:list itemType="xsd:QName" />
                    </xsd:simpleType>
                </xsd:element>
                <xsd:element name="doctype-system" type="xsd:string" minOccurs="0" />
                <xsd:element name="doctype-public" type="xsd:string" minOccurs="0" />
                <xsd:element name="encoding" type="xsd:string" />
                <xsd:element name="escape-uri-attributes" type="wsdaix:YesNoType" />
                <xsd:element name="include-content-type" type="wsdaix:YesNoType" />
                <xsd:element name="indent" type="wsdaix:YesNoType" />
                <xsd:element name="media-type" type="xsd:string" />
                <xsd:element name="normalize-unicode">
                    <xsd:simpleType>
                        <xsd:restriction base="xsd:string">
                            <xsd:enumeration value="NFC" />
                            <xsd:enumeration value="NFD" />
                            <xsd:enumeration value="NFKC" />
                            <xsd:enumeration value="NFKD" />
                            <xsd:enumeration value="fully-normalized" />
                            <xsd:enumeration value="none" />
                        </xsd:restriction>
                    </xsd:simpleType>
                </xsd:element>
                <xsd:element name="omit-xml-declaration" type="wsdaix:YesNoType" />
                <xsd:element name="standalone" type="wsdaix:YesNoOmitType" />
                <xsd:element name="undeclared-namespaces" type="wsdaix:YesNoType" />
                <xsd:element name="use-character-maps" type="xsd:string" />
                <xsd:element name="version" type="xsd:string" />
            </xsd:sequence>
        </xsd:complexType>
    </xsd:element>
    <!-- property and configuration documents -->
</xsd:schema>
```

Appendix B.2 – XQuery WSDL

```
<?xml version="1.0" encoding="UTF-8"?>
<wsdl:definitions name="wsdaix"
                  targetNamespace="http://www.ggf.org/namespaces/2005/12/WS-DAIX"
                  xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/"
                  xmlns:xsd="http://www.w3.org/2001/XMLSchema"
                  xmlns:wsdai="http://www.ggf.org/namespaces/2005/12/WS-DAI"
                  xmlns:wsdaix="http://www.ggf.org/namespaces/2005/12/WS-DAIX">

<!-- WSDL IMPORTS #################################################### -->
    <wsdl:import location="./wsdai_core_porttypes.wsdl"
                 namespace="http://www.ggf.org/namespaces/2005/12/WS-DAI"/>

<!-- WSDL TYPES ###################################################### -->
    <wsdl:types>
      <xsd:schema targetNamespace="http://www.ggf.org/namespaces/2005/12/WS-DAIX"
                  elementFormDefault="qualified">
       <xsd:import namespace="http://www.ggf.org/namespaces/2005/12/WS-DAI"
                   schemaLocation="./wsdai_core_types.xsd" />
       <xsd:include schemaLocation="./wsdaix_xquery_types.xsd" />
       <xsd:include schemaLocation="./wsdaix_xmlcollection_types.xsd" />


    <!-- ############################ -->
    <!-- ### Common Message Types ### -->
    <!-- ############################ -->
        <xsd:complexType name="XQueryExpressionType">
          <xsd:complexContent>
             <xsd:extension base="wsdai:ExpressionType">
               <xsd:sequence>
```

```
                <xsd:any namespace="http://www.w3.org/2005/XQueryX"/>
             </xsd:sequence>
           </xsd:extension>
        </xsd:complexContent>
      </xsd:complexType>
      <xsd:element name="XQueryExpression" type="wsdaix:XQueryExpressionType" abstract="true" />

      <xsd:complexType name="XQueryFaultType">
        <xsd:sequence>
          <xsd:element name="Detail" type="wsdaix:XMLWrapperType" minOccurs="0"/>
        </xsd:sequence>
      </xsd:complexType>
      <xsd:element name="XQueryFault" type="wsdaix:XQueryFaultType"/>

  <!-- ################################### -->
  <!-- ### XQueryExecute Message Types ### -->
  <!-- ################################### -1>

      <xsd:element name="XQueryExecuteRequest">
        <xsd:complexType>
          <xsd:complexContent>
            <xsd:extension base="wsdai:RequestType">
              <xsd:sequence>
                <xsd:element ref="wsdaix:XQueryExpression"  minOccurs="1" maxOccurs="1"/>
                <xsd:element ref="wsdaix:XMLSerializationParameters"  minOccurs="1" maxOccurs="1"/>
              </xsd:sequence>
            </xsd:extension>
          </xsd:complexContent>
        </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>

  <!-- ########################################## -->
  <!-- ### XQueryExecuteFactory Message Types ### -->
  <!-- ########################################## -->
      <xsd:element name="XQueryExecuteFactoryRequest">
        <xsd:complexType>
          <xsd:complexContent>
            <xsd:extension base="wsdai:FactoryRequestType">
              <xsd:sequence>
```

```
                  <xsd:element ref="wsdaix:XQueryExpression"  minOccurs="1" maxOccurs="1"/>
                </xsd:sequence>
              </xsd:extension>
            </xsd:complexContent>
          </xsd:complexType>
        </xsd:element>

        <!-- assumes that these messages result in a service/resource that contains all of -->
        <!-- the possible responses from a SQL execute (rowset, count, value, parameter etc) -->
        <xsd:element name="XQueryExecuteFactoryResponse" type="wsdai:DataResourceAddressType" />

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

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

    <wsdl:message name="XQueryFault">
      <wsdl:part name="XQueryFault"
                 element="wsdaix:XQueryFault"/>
    </wsdl:message>

    <!-- ############################## -->
    <!-- ### XQueryExecute Messages ### -->
    <!-- ############################## -->
    <wsdl:message name="XQueryExecuteRequest">
      <wsdl:part name="XQueryExecuteRequest" element="wsdaix:XQueryExecuteRequest"/>
    </wsdl:message>

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

    <!-- ##################################### -->
    <!-- ### XQueryExecuteFactory Messages ### -->
    <!-- ##################################### -->
    <wsdl:message name="XQueryExecuteFactoryRequest">
      <wsdl:part name="XQueryExecuteFactoryRequest" element="wsdaix:XQueryExecuteFactoryRequest"/>
    </wsdl:message>

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

<!-- WSDL PORT TYPES ################################################ -->
    <wsdl:portType name="XQueryAccessPT">

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

```
          <wsdl:input message="wsdaix:XQueryExecuteRequest"/>
          <wsdl:output message="wsdaix:XQueryExecuteResponse"/>
          <wsdl:fault  name="InvalidResourceNameFault"
                       message="wsdai:InvalidResourceNameFault" />
          <wsdl:fault name="DataResourceUnavailableFault"
                       message="wsdai:DataResourceUnavailableFault" />
          <wsdl:fault message="wsdai:InvalidExpressionFault"
                       name="InvalidExpressionFault"/>
          <wsdl:fault message="wsdai:InvalidDatasetFormatFault"
                       name="InvalidDatasetFormatFault"/>
          <wsdl:fault message="wsdai:NotAuthorizedFault"
                       name="NotAuthorizedFault"/>
          <wsdl:fault message="wsdai:ServiceBusyFault"
                       name="ServiceBusyFault" />
          <wsdl:fault  name="XQueryFault"
                       message="wsdaix:XQueryFault" />
        </wsdl:operation>

    </wsdl:portType>

    <wsdl:portType name="XQueryFactoryPT">

        <wsdl:operation name="XQueryExecuteFactory">
          <wsdl:input message="wsdaix:XQueryExecuteFactoryRequest"/>
          <wsdl:output message="wsdaix:XQueryExecuteFactoryResponse"/>
          <wsdl:fault  name="InvalidResourceNameFault"
                       message="wsdai:InvalidResourceNameFault" />
          <wsdl:fault name="DataResourceUnavailableFault"
                       message="wsdai:DataResourceUnavailableFault" />
          <wsdl:fault message="wsdai:InvalidExpressionFault"
                       name="InvalidExpressionFault"/>
          <wsdl:fault message="wsdai:InvalidPortTypeQNameFault"
                       name="InvalidPortTypeQNameFault"/>
          <wsdl:fault message="wsdai:InvalidConfigurationDocumentFault"
                       name="InvalidConfigurationDocumentFault"/>
          <wsdl:fault message="wsdai:NotAuthorizedFault"
                       name="NotAuthorizedFault"/>
          <wsdl:fault message="wsdai:ServiceBusyFault"
                       name="ServiceBusyFault" />
          <wsdl:fault  name="XQueryFault"
                       message="wsdaix:XQueryFault" />
        </wsdl:operation>

    </wsdl:portType>

</wsdl:definitions>
```

Appendix C.1 – XPath XML Schema

```
No types defines
```

Appendix C.2 – XPath WSDL

```
<?xml version="1.0" encoding="UTF-8"?>
<wsdl:definitions name="wsdaix"
                  targetNamespace="http://www.ggf.org/namespaces/2005/12/WS-DAIX"
                  xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/"
                  xmlns:xsd="http://www.w3.org/2001/XMLSchema"
                  xmlns:wsdai="http://www.ggf.org/namespaces/2005/12/WS-DAI"
                  xmlns:wsdaix="http://www.ggf.org/namespaces/2005/12/WS-DAIX">

<!-- WSDL IMPORTS ################################################## -->
    <wsdl:import location="./wsdai_core_porttypes.wsdl"
                 namespace="http://www.ggf.org/namespaces/2005/12/WS-DAI"/>

<!-- WSDL TYPES #################################################### -->
    <wsdl:types>
      <xsd:schema targetNamespace="http://www.ggf.org/namespaces/2005/12/WS-DAIX"
                  elementFormDefault="qualified">
       <xsd:import namespace="http://www.ggf.org/namespaces/2005/12/WS-DAI"
                   schemaLocation="./wsdai_core_types.xsd" />
       <xsd:include schemaLocation="./wsdaix_xpath_types.xsd" />
       <xsd:include schemaLocation="./wsdaix_xquery_types.xsd" />
       <xsd:include schemaLocation="./wsdaix_xmlcollection_types.xsd" />

    <!-- ############################ -->
    <!-- ### Common Message Types ### -->
    <!-- ############################ -->
        <xsd:complexType name="XPathExpressionType">
          <xsd:complexContent>
             <xsd:extension base="wsdai:ExpressionType">
               <xsd:sequence>
                 <xsd:any namespace="http://www.w3.org/2005/XQueryX"/>
               </xsd:sequence>
             </xsd:extension>
          </xsd:complexContent>
        </xsd:complexType>
        <xsd:element name="XPathExpression" type="wsdaix:XPathExpressionType" abstract="true" />

        <xsd:complexType name="XPathFaultType">
          <xsd:sequence>
            <xsd:element name="Detail" type="wsdaix:XMLWrapperType" minOccurs="0"/>
        </xsd:sequence>
      </xsd:complexType>
      <xsd:element name="XPathFault" type="wsdaix:XPathFaultType"/>

  <!-- ################################## -->
  <!-- ### XPathExecute Message Types ### -->
  <!-- ################################## -->

      <xsd:element name="XPathExecuteRequest">
        <xsd:complexType>
          <xsd:complexContent>
            <xsd:extension base="wsdai:RequestType">
              <xsd:sequence>
                <xsd:element ref="wsdaix:XPathExpression"  minOccurs="1" maxOccurs="1"/>
                <xsd:element ref="wsdaix:XMLSerializationParameters"  minOccurs="1" maxOccurs="1"/>
              </xsd:sequence>
            </xsd:extension>
          </xsd:complexContent>
        </xsd:complexType>
      </xsd:element>

      <xsd:element name="XPathExecuteResponse">
          <xsd:complexType>
              <xsd:sequence>
                  <xsd:element ref="wsdai:Dataset"  minOccurs="1" maxOccurs="1"/>
              </xsd:sequence>
          </xsd:complexType>
      </xsd:element>

  <!-- ##################################### -->
  <!-- ### XPathQueryFactory Message Types ### -->
  <!-- ##################################### -->
      <xsd:element name="XPathExecuteFactoryRequest">
        <xsd:complexType>
          <xsd:complexContent>
            <xsd:extension base="wsdai:FactoryRequestType">
              <xsd:sequence>
                <xsd:element ref="wsdaix:XPathExpression"  minOccurs="1" maxOccurs="1"/>
              </xsd:sequence>
            </xsd:extension>
          </xsd:complexContent>
        </xsd:complexType>
      </xsd:element>

      <!-- assumes that these messages result in a service/resource that contains all of -->
      <!-- the possible responses from a SQL execute (rowset, count, value, parameter etc) -->
      <xsd:element name="XPathExecuteFactoryResponse" type="wsdai:DataResourceAddressType" />
      </xsd:schema>
  </wsdl:types>

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

    <wsdl:message name="XPathFault">
      <wsdl:part name="XPathFault"
                 element="wsdaix:XPathFault"/>
    </wsdl:message>

    <!-- ########################### -->
    <!-- ### XPathQuery Messages ### -->
    <!-- ########################### -->
    <wsdl:message name="XPathExecuteRequest">
      <wsdl:part name="XPathExecuteRequest" element="wsdaix:XPathExecuteRequest"/>
    </wsdl:message>

    <wsdl:message name="XPathExecuteResponse">
      <wsdl:part name="XPathExecuteResponse" element="wsdaix:XPathExecuteResponse"/>
    </wsdl:message>

    <!-- ################################## -->
    <!-- ### XPathQueryFactory Messages ### -->
    <!-- ################################## -->
    <wsdl:message name="XPathExecuteFactoryRequest">
      <wsdl:part name="XPathExecuteFactoryRequest" element="wsdaix:XPathExecuteFactoryRequest"/>
    </wsdl:message>

    <wsdl:message name="XPathExecuteFactoryResponse">
      <wsdl:part name="XPathExecuteFactoryResponse" element="wsdaix:XPathExecuteFactoryResponse"/>
    </wsdl:message>


<!-- WSDL PORT TYPES ################################################ -->
    <wsdl:portType name="XPathAccessPT">

        <wsdl:operation name="XPathExecute">
          <wsdl:input message="wsdaix:XPathExecuteRequest"/>
          <wsdl:output message="wsdaix:XPathExecuteResponse"/>
          <wsdl:fault  name="InvalidResourceNameFault"
                       message="wsdai:InvalidResourceNameFault" />
          <wsdl:fault name="DataResourceUnavailableFault"
                       message="wsdai:DataResourceUnavailableFault" />
          <wsdl:fault message="wsdai:InvalidExpressionFault"
                       name="InvalidExpressionFault"/>
          <wsdl:fault message="wsdai:InvalidDatasetFormatFault"
```

```
                       name="InvalidDatasetFormatFault"/>
          <wsdl:fault message="wsdai:NotAuthorizedFault"
                       name="NotAuthorizedFault"/>
          <wsdl:fault message="wsdai:ServiceBusyFault"
                       name="ServiceBusyFault" />
          <wsdl:fault  name="XPathFault"
                       message="wsdaix:XPathFault" />
        </wsdl:operation>

    </wsdl:portType>

    <wsdl:portType name="XPathFactoryPT">

        <wsdl:operation name="XPathExecuteFactory">
          <wsdl:input message="wsdaix:XPathExecuteFactoryRequest"/>
          <wsdl:output message="wsdaix:XPathExecuteFactoryResponse"/>
          <wsdl:fault  name="InvalidResourceNameFault"
                       message="wsdai:InvalidResourceNameFault" />
          <wsdl:fault name="DataResourceUnavailableFault"
                       message="wsdai:DataResourceUnavailableFault" />
          <wsdl:fault message="wsdai:InvalidExpressionFault"
                       name="InvalidExpressionFault"/>
          <wsdl:fault message="wsdai:InvalidPortTypeQNameFault"
                       name="InvalidPortTypeQNameFault"/>
          <wsdl:fault message="wsdai:InvalidConfigurationDocumentFault"
                       name="InvalidConfigurationDocumentFault"/>
          <wsdl:fault message="wsdai:NotAuthorizedFault"
                       name="NotAuthorizedFault"/>
          <wsdl:fault message="wsdai:ServiceBusyFault"
                       name="ServiceBusyFault" />
          <wsdl:fault  name="XPathFault"
                       message="wsdaix:XPathFault" />
        </wsdl:operation>

    </wsdl:portType>
</wsdl:definitions>
```

Appendix D.1 – XUpdate XML Schema

```
No types defined.
```

Appendix D.2 – XUpdate WSDL

```
<?xml version="1.0" encoding="UTF-8"?>
<wsdl:definitions name="wsdaix"
                  targetNamespace="http://www.ggf.org/namespaces/2005/12/WS-DAIX"
                  xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/"
                  xmlns:xsd="http://www.w3.org/2001/XMLSchema"
                  xmlns:wsdai="http://www.ggf.org/namespaces/2005/12/WS-DAI"
                  xmlns:wsdaix="http://www.ggf.org/namespaces/2005/12/WS-DAIX">

<!-- WSDL IMPORTS #################################################### -->
    <wsdl:import location="./wsdai_core_porttypes.wsdl"
                 namespace="http://www.ggf.org/namespaces/2005/12/WS-DAI"/>

<!-- WSDL TYPES ##################################################### -->
    <wsdl:types>
      <xsd:schema targetNamespace="http://www.ggf.org/namespaces/2005/12/WS-DAIX"
                  elementFormDefault="qualified">
       <xsd:import namespace="http://www.ggf.org/namespaces/2005/12/WS-DAI"
                   schemaLocation="./wsdai_core_types.xsd" />
       <xsd:include schemaLocation="./wsdaix_xupdate_types.xsd" />
       <xsd:include schemaLocation="./wsdaix_xmlcollection_types.xsd" />

    <!-- ############################ -->
    <!-- ### Common Message Types ### -->
    <!-- ############################ -->
        <xsd:complexType name="XUpdateExpressionType">
          <xsd:complexContent>
             <xsd:extension base="wsdai:ExpressionType">
               <xsd:sequence>
                 <xsd:any namespace="http://www.xmldb.org/xupdate"/>
               </xsd:sequence>
             </xsd:extension>
          </xsd:complexContent>
        </xsd:complexType>
        <xsd:element name="XUpdateExpression" type="wsdaix:XUpdateExpressionType" abstract="true" />

        <xsd:complexType name="XUpdateFaultType">
          <xsd:sequence>
            <xsd:element name="Detail" type="wsdaix:XMLWrapperType" minOccurs="0"/>
          </xsd:sequence>
        </xsd:complexType>
        <xsd:element name="XUpdateFault" type="wsdaix:XUpdateFaultType"/>

        <xsd:complexType name="XUpdateSchemaInvalidationFaultType"/>
        <xsd:element name="XUpdateSchemaInvalidationFault" type="wsdaix:XUpdateSchemaInvalidationFaultType"/>

    <!-- ############################# -->
    <!-- ### XUpdate Message Types ### -->
    <!-- ############################# -->

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

```
          <xsd:complexType>
            <xsd:complexContent>
              <xsd:extension base="wsdai:RequestType">
                <xsd:sequence>
                  <xsd:element ref="wsdaix:XUpdateExpression"  minOccurs="1" maxOccurs="1"/>
                </xsd:sequence>
              </xsd:extension>
            </xsd:complexContent>
          </xsd:complexType>
        </xsd:element>

        <xsd:element name="XUpdateExecuteResponse">
            <xsd:complexType>
                <xsd:sequence>
                    <xsd:element name="Count" type="xsd:integer"  minOccurs="1" maxOccurs="1"/>
                </xsd:sequence>
            </xsd:complexType>
        </xsd:element>

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

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

    <wsdl:message name="XUpdateFault">
      <wsdl:part name="XUpdateFault"
                 element="wsdaix:XUpdateFault"/>
    </wsdl:message>

    <wsdl:message name="XUpdateSchemaInvalidationFault">
      <wsdl:part name="XUpdateSchemaInvalidationFault"
                 element="wsdaix:XUpdateSchemaInvalidationFault"/>
    </wsdl:message>

    <!-- ######################## -->
    <!-- ### XUpdate Messages ### -->
    <!-- ######################## -->
    <wsdl:message name="XUpdateExecuteRequest">
      <wsdl:part name="XUpdateExecuteRequest" element="wsdaix:XUpdateExecuteRequest"/>
    </wsdl:message>

    <wsdl:message name="XUpdateExecuteResponse">
      <wsdl:part name="XUpdateExecuteResponse" element="wsdaix:XUpdateExecuteResponse"/>
    </wsdl:message>

<!-- WSDL PORT TYPES ################################################ -->
    <wsdl:portType name="XUpdateAccessPT">
```

```
        <wsdl:operation name="XUpdateExecute">
          <wsdl:input message="wsdaix:XUpdateExecuteRequest"/>
          <wsdl:output message="wsdaix:XUpdateExecuteResponse"/>
          <wsdl:fault  name="InvalidResourceNameFault"
                       message="wsdai:InvalidResourceNameFault" />
          <wsdl:fault name="DataResourceUnavailableFault"
                       message="wsdai:DataResourceUnavailableFault" />
          <wsdl:fault message="wsdai:InvalidExpressionFault"
                       name="InvalidExpressionFault"/>
          <wsdl:fault message="wsdai:InvalidDatasetFormatFault"
                       name="InvalidDatasetFormatFault"/>
          <wsdl:fault message="wsdai:NotAuthorizedFault"
                       name="NotAuthorizedFault"/>
          <wsdl:fault message="wsdai:ServiceBusyFault"
                       name="ServiceBusyFault" />
          <wsdl:fault  name="XUpdateFault"
                       message="wsdaix:XUpdateFault" />
          <wsdl:fault  name="XUpdateSchemaInvalidationFault"
                       message="wsdaix:XUpdateSchemaInvalidationFault" />
        </wsdl:operation>

    </wsdl:portType>

</wsdl:definitions>
```

Appendix E.1 – XMLSequence XML Schema

```
<?xml version="1.0" encoding="UTF-8" ?>
<xsd:schema targetNamespace="http://www.ggf.org/namespaces/2005/12/WS-DAIX" xmlns:xsd="http://www.w3.org/2001/XMLSchema"
    xmlns:wrs="http://java.sun.com/xml/ns/jdbc" xmlns:wsdai="http://www.ggf.org/namespaces/2005/12/WS-DAI"
    xmlns:wsdaix="http://www.ggf.org/namespaces/2005/12/WS-DAIX">
    <xsd:import namespace="http://www.ggf.org/namespaces/2005/12/WS-DAI" schemaLocation="./wsdai_core_types.xsd" />
    <xsd:include schemaLocation="./wsdaix_xquery_types.xsd" />
    <!-- properties -->
    <xsd:element name="NumberOfItems" type="xsd:long" />
    <!-- property and configuration documents -->
    <xsd:complexType name="XMLSequencePropertyDocumentType">
        <xsd:complexContent>
            <xsd:extension base="wsdai:PropertyDocumentType">
                <xsd:sequence>
                    <xsd:element ref="wsdaix:NumberOfItems" />
                </xsd:sequence>
            </xsd:extension>
        </xsd:complexContent>
    </xsd:complexType>
    <xsd:element name="XMLSequencePropertyDocument" type="wsdaix:XMLSequencePropertyDocumentType" />
</xsd:schema>
```

Appendix E.2 – XMLSequence WSDL

```
<?xml version="1.0" encoding="UTF-8"?>
<wsdl:definitions name="wsdaix"
                  targetNamespace="http://www.ggf.org/namespaces/2005/12/WS-DAIX"
                  xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/"
                  xmlns:xsd="http://www.w3.org/2001/XMLSchema"
                  xmlns:wsdai="http://www.ggf.org/namespaces/2005/12/WS-DAI"
                  xmlns:wsdaix="http://www.ggf.org/namespaces/2005/12/WS-DAIX">

<!-- WSDL IMPORTS #################################################### -->
    <wsdl:import location="./wsdai_core_porttypes.wsdl"
                 namespace="http://www.ggf.org/namespaces/2005/12/WS-DAI"/>

<!-- WSDL TYPES ###################################################### -->
    <wsdl:types>
      <xsd:schema targetNamespace="http://www.ggf.org/namespaces/2005/12/WS-DAIX"
                  elementFormDefault="qualified">
       <xsd:import namespace="http://www.ggf.org/namespaces/2005/12/WS-DAI"
                   schemaLocation="./wsdai_core_types.xsd" />
       <xsd:include schemaLocation="./wsdaix_xmlsequence_types.xsd" />


    <!-- ############################ -->
    <!-- ### Common Message Types ### -->
    <!-- ############################ -->


    <!-- ################################# -->
    <!-- ### XMLSequence Message Types ### -->
    <!-- ################################# -->

        <xsd:complexType name="InvalidStartPositionFaultType"/>
        <xsd:element name="InvalidStartPositionFault" type="wsdaix:InvalidStartPositionFaultType"/>

        <xsd:complexType name="InvalidCountFaultType"/>
        <xsd:element name="InvalidCountFault" type="wsdaix:InvalidCountFaultType"/>

        <xsd:element name="GetItemsRequest">
          <xsd:complexType>
            <xsd:complexContent>
              <xsd:extension base="wsdai:RequestType">
                <xsd:sequence>
                  <xsd:element name="StartPosition" type="xsd:integer"/>
                  <xsd:element name="Count" type="xsd:integer"/>
                  <xsd:element ref="wsdaix:XMLSerializationParameters" minOccurs="1" maxOccurs="1"/>
                </xsd:sequence>
              </xsd:extension>
            </xsd:complexContent>
          </xsd:complexType>
        </xsd:element>

        <xsd:element name="GetItemsResponse">
            <xsd:complexType>
                <xsd:sequence>
                  <xsd:element ref="wsdai:Dataset"  minOccurs="1" maxOccurs="1"/>
                </xsd:sequence>
            </xsd:complexType>
        </xsd:element>

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

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

    <!-- ############################################### -->
    <!-- ### GetXMLSequencePropertyDocument Messages ### -->
    <!-- ############################################### -->
    <wsdl:message name="GetXMLSequencePropertyDocumentRequest">
      <wsdl:part name="GetXMLSequencePropertyDocumentRequest"
                 element="wsdai:GetDataResourcePropertyDocumentRequest" />
    </wsdl:message>

    <wsdl:message name="GetXMLSequencePropertyDocumentResponse">
      <wsdl:part name="GetXMLSequencePropertyDocumentResponse"
                 element="wsdaix:XMLSequencePropertyDocument" />
    </wsdl:message>

    <!-- ######################### -->
    <!-- ### GetItems Messages ### -->
    <!-- ######################### -->
    <wsdl:message name="GetItemsRequest">
      <wsdl:part name="GetItemsRequest" element="wsdaix:GetItemsRequest"/>
    </wsdl:message>

    <wsdl:message name="GetItemsResponse">
      <wsdl:part name="GetItemsResponse" element="wsdaix:GetItemsResponse"/>
    </wsdl:message>

    <wsdl:message name="InvalidStartPositionFault">
      <wsdl:part name="InvalidStartPositionFault"
                 element="wsdaix:InvalidStartPositionFault"/>
    </wsdl:message>
    <wsdl:message name="InvalidCountFault">
      <wsdl:part name="InvalidCountFault"
                 element="wsdaix:InvalidCountFault"/>
    </wsdl:message>

<!-- WSDL PORT TYPES ############################################### -->
    <wsdl:portType name="XMLSequenceAccessPT">

        <wsdl:operation name="GetXMLSequencePropertyDocument">
          <wsdl:input  name="GetXMLSequencePropertyDocumentRequest"
                       message="wsdaix:GetXMLSequencePropertyDocumentRequest" />
          <wsdl:output name="GetXMLSequencePropertyDocumentResponse"
                       message="wsdaix:GetXMLSequencePropertyDocumentResponse" />
          <wsdl:fault  name="InvalidResourceNameFault"
                       message="wsdai:InvalidResourceNameFault" />
          <wsdl:fault message="wsdai:NotAuthorizedFault"
                       name="NotAuthorizedFault"/>
          <wsdl:fault name="DataResourceUnavailableFault"
                       message="wsdai:DataResourceUnavailableFault" />
          <wsdl:fault  name="ServiceBusyFault"
                       message="wsdai:ServiceBusyFault" />
        </wsdl:operation>

        <wsdl:operation name="GetItems">
          <wsdl:input message="wsdaix:GetItemsRequest"/>
          <wsdl:output message="wsdaix:GetItemsResponse"/>
          <wsdl:fault  name="InvalidResourceNameFault"
                       message="wsdai:InvalidResourceNameFault" />
          <wsdl:fault name="DataResourceUnavailableFault"
                       message="wsdai:DataResourceUnavailableFault" />
          <wsdl:fault message="wsdai:InvalidDatasetFormatFault"
                       name="InvalidDatasetFormatFault"/>
          <wsdl:fault message="wsdai:NotAuthorizedFault"
                       name="NotAuthorizedFault"/>
          <wsdl:fault message="wsdai:ServiceBusyFault"
                       name="ServiceBusyFault" />
          <wsdl:fault  name="InvalidStartPositionFault"
                       message="wsdaix:InvalidStartPositionFault" />
          <wsdl:fault  name="InvalidCountFault"
                       message="wsdaix:InvalidCountFault" />
        </wsdl:operation>

    </wsdl:portType>

</wsdl:definitions>
```