

OGSA-DAI WS-DAIX 1.0

Authors: Mike Jackson (michaelj@epcc.ed.ac.uk) and Elias Theocharopoulos (elias@nesc.ac.uk).

Project: The OGSA-DAI Project Team (info@ogsadai.org.uk).

Version: 1.0

Date: May 2008.

Contents

Contents.....	1
1. Introduction.....	3
1.1 OGF WS-DAI specifications	3
2. Services and resources.....	3
3. OGSA-DAI WS-DAIX architecture.....	5
3.1 Services.....	5
3.1.1 WSDL	6
3.1.2 WSDL and auto-code generation	6
3.1.3 WSRF.....	6
3.1.4 Service configuration and indices	7
3.2 portTypes and providers	7
3.3 OGSA-DAI server context	8
3.3.1 Specifying the OGSA-DAI workflow parser	8
3.4 Services / resource relationships	9
3.5 Executors.....	9
3.5.1 CoreDataAccessExecutor.....	9
3.5.2 CoreResourceListExecutor	9
3.5.3 XMLCollectionExecutor.....	10
3.5.4 XPathExecutor.....	12
3.5.5 XQueryExecutor	12
3.5.6 XUpdateExecutor	13
3.5.7 XMLSequenceExecutor.....	14
3.5.8 Data request execution resource.....	14
3.6 WS-DAIX activities	14
3.6.1 CharArraysToDOM	14
3.6.2 AddDocuments	15
3.6.3 GetDocuments	15
3.6.4 RemoveDocuments	16
3.6.5 CreateSubcollection	16
3.6.6 RemoveSubcollection	17
3.6.7 CreateXMLCollectionResource	17
3.6.8 CreateXMLDocumentResource	17
3.6.9 CreateXMLSequenceResource	18
3.6.10 WriteToXMLSequence.....	18
3.6.11 GetItems	18
3.6.12 XUpdate.....	19
3.7 OGSA-DAI 3.0 activities	19
3.8 Resources	19
3.8.1 Core resource commonality.....	20
3.8.2 XMLCollection and XMLDocument commonality	20
3.8.3 XMLCollection.....	21
3.8.4 XMLDocument.....	22

3.8.5	XMLSequence.....	23
3.8.6	Sequences and SequenceManagers	24
4.	Naming and data formats	25
4.1	ResourceSets.....	25
4.2	Resource IDs and data resource abstract names	25
4.3	Collection names and URIs	25
4.4	XQueryX, XPath, XQuery and XUpdate.....	26
4.5	Sequences.....	26
5.	Security	26
5.1	Logins, XMLCollections and XMLDocuments	26
5.2	Logins and resource properties	27
6.	OGSA-DAI WS-DAIX Specification Compliance	27
6.1	Notation.....	27
6.2	portTypes common to all services	27
6.2.1	CoreDataAccess.....	27
6.2.2	CoreResourceList.....	28
6.3	XMLCollection and XMLCollectionService	29
6.3.1	CoreDataDescription	29
6.3.2	XMLCollectionDescription	30
6.3.3	XMLCollectionAccess	30
6.3.4	XMLCollectionFactory	35
6.3.5	XPathAccess	36
6.3.6	XPathFactory	37
6.3.7	XQueryAccess	38
6.3.8	XQueryFactory	39
6.3.9	XUpdateAccess.....	40
6.4	XMLDocument and XMLDocumentService	40
6.4.1	CoreDataDescription	40
6.4.2	Other portTypes	41
6.5	XMLSequence and XMLSequenceService	41
6.5.1	CoreDataDescription	41
6.5.2	XMLSequenceDescription	41
6.5.3	XMLSequenceAccess	42
7.	Options for future work	42
7.1	OGSA-DAI WS-DAIX test coverage.....	42
7.2	Error reporting	43
7.3	Resource state	44
7.4	Resource properties.....	44
7.5	InMemorySequence	45
7.6	Executors.....	45
7.7	XMLDocuments and XPath, XQuery, XUpdate.....	45
7.8	Activities	46
7.9	Presentation layers	46
7.10	Security	47
7.11	DRER capacity and ServiceBusyFaultType	47
7.12	Saving workflows as XML files	47
7.13	Clients	47
8.	OGSA-DAI 3.0 limitations identified.....	47
8.1	A limitation for multiple presentation layers.....	48
9.	OGSA-DAI 3.0 ingestion	49

1. Introduction

This document describes an implementation of the OGF WS-DAIX specification using OGSA-DAI 3.0 done by the OGSA-DAI team of the University of Edinburgh.

The document assumes knowledge of the OGF WS-DAI specifications and also, in certain sections, knowledge of OGSA-DAI 3.0 components.

OGSA-DAI WS-DAIX is available from the OGSA-DAI WWW site at <http://www.ogsadai.org.uk/downloads> and the documentation is available at <http://www.ogsadai.org.uk/documentation>.

1.1 OGF WS-DAI specifications

The WS-DAI specifications implemented by OGSA-DAI WS-DAIX are:

- Web Services Data Access and Integration – The Core (WS-DAI) Specification, Version 1.0, 20th July 2006. GFD 74 (WS-DAI). <http://www.ogf.org/documents/GFD.74.pdf>
- Web Services Data Access and Integration – The XML Realization (WS-DAIX) Specification, Version 1.0, 2nd August 2006. GFD 75 (WS-DAIX). <http://www.ogf.org/documents/GFD.75.pdf>

In addition, the following specifies what MUST/SHOULD/MAY be implemented and the strategy for doing an inter-operability test.

- Interoperability Testing for DAIS Working Group Specifications, 5th September 2006. GFD 77. <http://www.ogf.org/documents/GFD.77.pdf>

The deadline for specification implementation from the DAIS-WG perspective is 09/2008.

2. Services and resources

Though not explicitly defined in the specifications, the following resources are implied by WS-DAIX and are therefore the ones supported by OGSA-DAI WS-DAIX:

- XMLCollection – represents a XML database collection.
- XMLDocument – represents a document in an XML database collection.
- XMLSequence – represents the results from an XPath or XQuery query on a document or collection.

The WS-DAIX specifications do not specify WS-DAIX services, only individual portTypes which can be composed into services. However in the specification examples there are standard groupings of portTypes into services that are suggested. These are as follows and are the ones used in OGSA-DAI WS-DAIX:

- XMLCollection service
 - CoreDataAccess, CoreResourceList

- CollectionAccess, CollectionFactory
- XQueryAccess, XQueryFactory
- XPathAccess, XPathFactory
- XUpdateAccess
- XMLDocument service
 - CoreDataAccess, CoreResourceList
 - XQueryAccess, XQueryFactory
 - XPathAccess, XPathFactory
 - XUpdateAccess
- XMLSequence service
 - CoreDataAccess, CoreResourceList
 - XMLSequenceAccess

The following figures show the relationship between services, portType providers and executors.

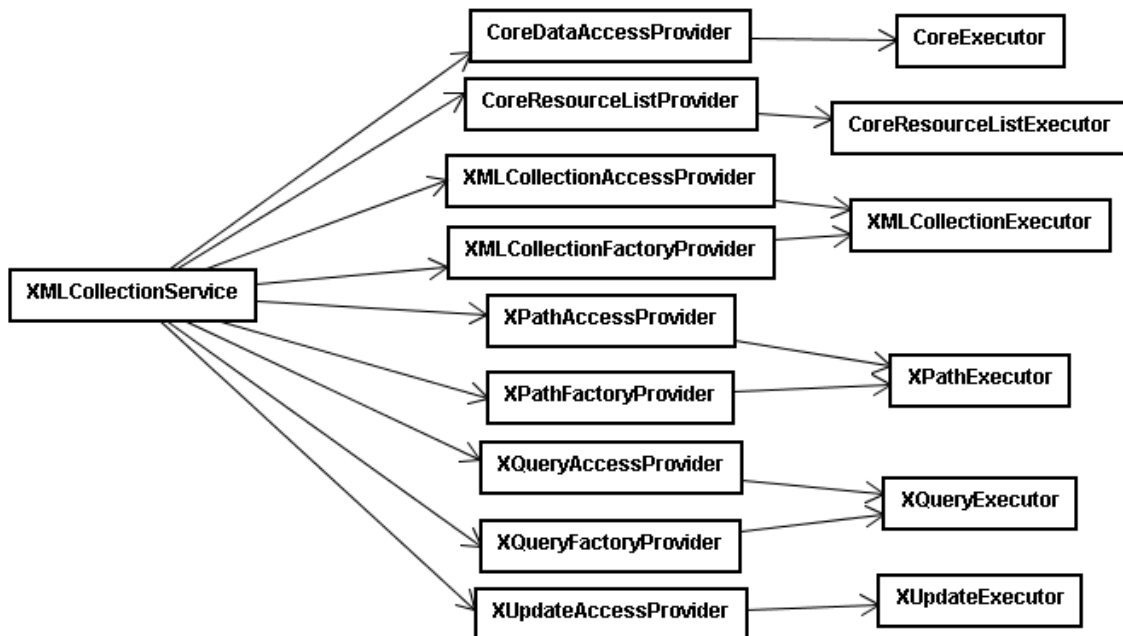


Figure 1: XMLCollection service, portType providers and executors

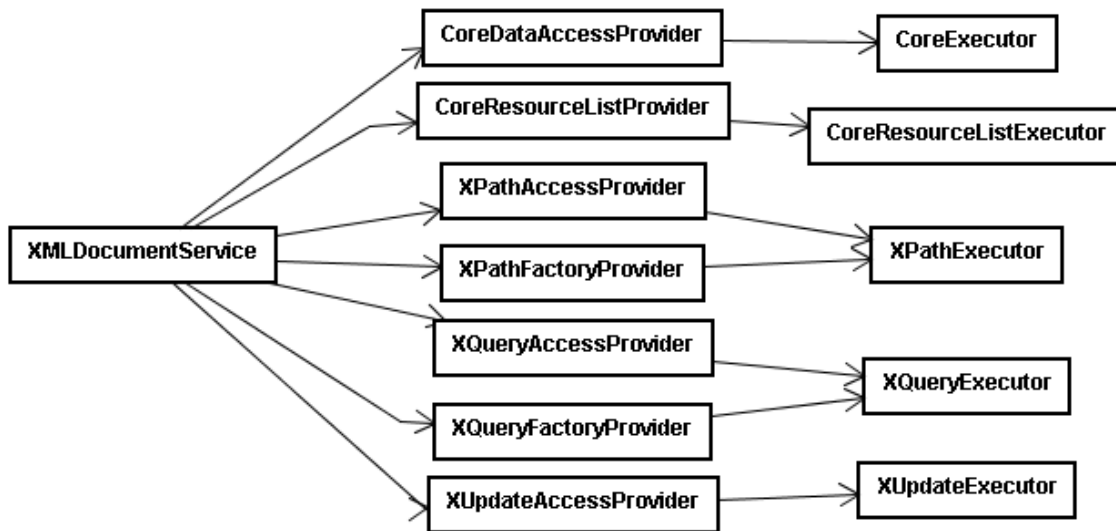


Figure 2: XMLDocument service , portType providers and executors

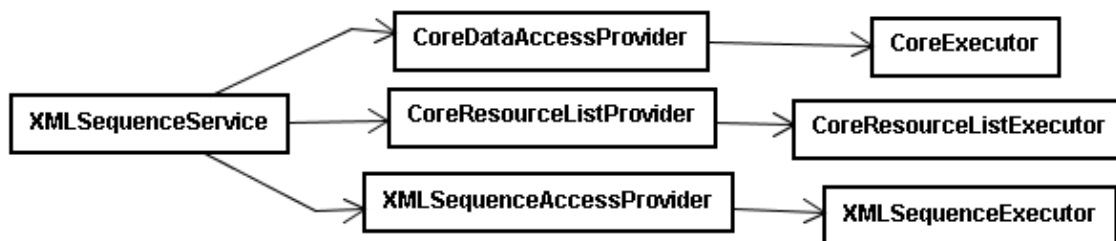


Figure 3: XMLSequence service, portType providers and executors

3. OGSA-DAI WS-DAIX architecture

This section outlines the OGSA-DAI WS-DAIX architecture and its main components and characteristics. The architecture consists of the following layers:

- Services – a thin Web services layer which delegates operations to portType providers.
- PortType providers – a layer of portType-specific components which gets resources and delegates responsibility for execution of operations on the resources to executors.
- Executors – a layer which implements WS-DAIX operations either directly, or by delegation to a resource, or by the execution of OGSA-DAI workflows.
- Resources – WS-DAIX resources which manage access to data (XML collections and documents and sequences of XML items).

3.1 Services

There are three types of service and so one class per WS-DAIX service. These are as follows:

- uk.org.ogsadai.wsdai.daix.service.XMLCollectionService
- uk.org.ogsadai.wsdai.daix.service.XMLDocumentService
- uk.org.ogsadai.wsdai.daix.service.XMLSequenceService

Each service:

- Implements all the operations of the associated portTypes by delegating to an associated portType provider, as described in the next section.
- Implements the javax.xml.rpc.server.ServiceLifecycle interface. By implementing this, services will be notified by the Web services container when they are first initialised. This, in turn, allows a service to bootstrap the OGSA-DAI WS-DAIX server (see section **Error! Reference source not found.**) when the service is first contacted and initialised.

3.1.1 WSDL

OGSA-DAI WS-DAIX services are declared in WSDL. A binding to SOAP over HTTP is used with an rpc/literal encoding.

While the service WSDL is accepted by Apache Axis WSDL2Java, it throws up validation errors if validated using the Eclipse WSDL validator. There are two validation errors:

- The part GetCollectionPropertyDocumentRequest has an invalid value 'GetDataResourcePropertyDocumentRequest' defined for its element. Element declarations must refer to valid values defined in a schema.
- The part 'GetXMLSequencePropertyDocumentRequest' has an invalid value 'GetDataResourcePropertyDocumentRequest' defined for its element. Element declarations must refer to valid values defined in a schema.

These arise in the WSDL as defined in the WS-DAIX specification, and not that defined by the OGSA-DAI WS-DAIX developers, so has been left unaltered.

3.1.2 WSDL and auto-code generation

Service interfaces for OGSA-DAI WS-DAIX services are generated using Apache Axis WSDL2Java. The argument and result classes auto-generated by WSDL2Java are used throughout OGSA-DAI WS-DAIX components (with the exception of the OGSA-DAI WS-DAIX activities of section 3.6).

3.1.3 WSRF

OGSA-DAI WS-DAIX services are not compliant with and do not support WSRF. This is an optional requirement in the WS-DAIX specifications.

3.1.4 Service configuration and indices

The OGSA-DAI WS-DAIX server configuration information is assumed to declare the following for each service:

- A unique service ID (of type `uk.org.ogsadai.common.ID` – basically a “.”-delimited string) for each deployed service e.g.: `ExampleXMLCollectionService`.
- A service type (of type `uk.org.ogsadai.common.ID` – basically a “.”-delimited string) which is of one of the following values:
 - `wsdai.XMLCollectionService`
 - `wsdai.XMLDocumentService`
 - `wsdai.XMLSequenceService`
- A URL prefix for all the service ports belonging to a service in the OGSA-DAI WS-DAIX server e.g. `http://coal:9020/dai/services/`
- A collection of portType-service port pairs which give the name of the port that implements each portType of the service
 - e.g. an `XMLCollection` service may have a port `CollectionServiceCollectionAccessPT` which implements the WS-DAIX portType `{http://www.ggf.org/namespaces/2005/12/WS-DAIX/}XMLCollectionAccessPT`.
 - A port appended to the URL prefix should specify a valid endpoint for the service.

OGSA-DAI WS-DAIX constructs, from this configuration information, a set of indices which maintain the relationships between services, ports and portTypes. This is required for constructing data resource addresses (WS-EPRs) for resources. This information is stored as maps in the OGSA-DAI server context (see section 3.3). The maps are as follows:

- Service type map - a mapping from service types to lists of service IDs. “Services S1, S2 and S3 all are of type ST”.
- portType map – a mapping from service ports to portTypes e.g. “Port P1 implements portType PT”.
- Service port map – a mapping from service IDs to lists of ports e.g. “Service S has ports P1, P2 and P3”.
- Port service map – a mapping from service ports to service IDs e.g. “Port P1 belongs to service S”.
- Service URL map – a mapping from service IDs to service URL prefixes e.g. “Service S has URL prefix `http://coal:9020/dai/services/`”.

3.2 portTypes and providers

A service is a collection of portTypes. Each service may implement multiple portTypes. More than one service may implement the same portType. To avoid duplication of functionality a “portType provider” model, as used in OGSA-DAI 3.0 and recommended by Globus, is used to implement services.

When a service operation is invoked the service class just forwards the arguments on to the corresponding method of the provider for the portType whose operation was invoked e.g. if the `DestroyDataResource` operation of an `XMLCollection` service is

invoked then this is passed onto a portType provider that handles operations of the CoreDataAccess portType. There is one provider class per WS-DAIX portType:

- uk.org.ogsadai.wsdai.core.provider.CoreDataAccessProvider
- uk.org.ogsadai.wsdai.core.provider.CoreResourceListProvider
- uk.org.ogsadai.wsdai.daix.provider.XMLCollectionAccessProvider
- uk.org.ogsadai.wsdai.daix.provider.XMLCollectionFactoryProvider
- uk.org.ogsadai.wsdai.daix.provider.XQueryAccessProvider
- uk.org.ogsadai.wsdai.daix.provider.XQueryFactoryProvider
- uk.org.ogsadai.wsdai.daix.provider.XPathAccessProvider
- uk.org.ogsadai.wsdai.daix.provider.XPathFactoryProvider
- uk.org.ogsadai.wsdai.daix.provider.XUpdateAccessProvider
- uk.org.ogsadai.wsdai.daix.provider.XMLSequenceAccessProvider

Certain providers have a constructor that takes as an argument the expected resource class. This allows the executors they call (see section 3.5) and that may be used by multiple services to filter resources according to the type of resource exposed by the associated service e.g. so that XPathAccessProvider and XPathExecutor only execute operations on XMLCollections when used by an XMLCollectionService.

As stated, the providers implement the operations of the associated portType. This includes:

- Getting, from the operation arguments, the ID of the resource on which the operation is to be executed.
- Getting the resource itself.
- Delegating the operation invocation to an associated executor (see section 3.5).
- Returning the results.

3.3 OGSA-DAI server context

OGSA-DAI 3.0 has a common context which is configured using Web services container-specific functionality. This context contains the main OGSA-DAI components including its resource manager, activity manager and configuration manager. This also, for OGSA-DAI WS-DAIX includes the service, port and portType maps of section 3.1.4, login providers (which manage database logins for XMLCollection and XMLDocuments) and sequence managers (which manage the sequences of XML fragments exposed by XMLSequences). In OGSA-DAI WS-DAIX the following class manages this bootstrapping process, populating the OGSA-DAI context from the Tomcat JNDI information:

uk.org.ogsadai.wsdai.core.context.WSDAIContextInitializer

This class is invoked by each of the service classes but ensures that the context is only populated once.

3.3.1 Specifying the OGSA-DAI workflow parser

OGSA-DAI WS-DAIX implements many WS-DAIX operations using OGSA-DAI workflows. This requires the OGSA-DAI activity framework to be initialised appropriately. This requires specification of a request factory, a class which maps workflows from a presentation layer-specific representation into the activity framework's internal representation. Since OGSA-DAI WS-DAIX constructs workflows using the activity framework's internal representation anyway, the request factory specified by the context initializer is one that does an identity mapping. Its class is:

`uk.org.ogsadai.resource.drer.SimpleRequestFactory`

3.4 Services / resource relationships

In OGSA-DAI WS-DAIX there is no explicit service-resource relationship maintained server-side. If one were to deploy two XMLCollection services then all XMLCollections on the server would be accessible via either of these services. Destroying a resource via any one service means it will be inaccessible by the other service.

3.5 Executors

Executors are classes that execute the WS-DAIX operations, being called by portType providers. They do this via operations on a resource directly, or via execution of OGSA-DAI workflows. The associated workflows use OGSA-DAI 3.0 activities plus a set of activities developed for OGSA-DAI WS-DAIX.

3.5.1 CoreDataAccessExecutor

The `uk.org.ogsadai.wsdai.core.executor.CoreDataAccessExecutor` executes the `CoreDataAccess` portType operations. It maintains a resource class to it can ensure that only resources consistent with the service that's ultimately using it are used. (e.g. `DestroyDataResource` can only be used with XMLCollections via an XMLCollection service).

CoreDataAccess::GetDataResourcePropertyDocument

This gets each property from the resource and assembles the data resource property document.

CoreDataAccess::DestroyDataResource

This tells the resource to destroy itself.

CoreDataAccess::GenericQuery

This is not implemented as it is optional in WS-DAIX.

3.5.2 CoreResourceListExecutor

The `uk.org.ogsadai.wsdai.core.executor.CoreDataAccessExecutor` executes the `CoreResourceList` portType operations. It maintains a resource class to it can ensure that only resources consistent with the service that's ultimately using it are used.

CoreResourceList::GetDataResourceList

This uses the resource type (i.e. the resource class), in conjunction with information on the services, ports and portTypes on the server to return the data resource address for all the resources of that type.

CoreResourceList::Resolve

This uses the resource ID and resource type, in conjunction with information on the services, ports and portTypes on the server (see XXXX) to return the data resource address for the given resource.

3.5.3 XMLCollectionExecutor

The `uk.org.ogsadai.wsdai.daix.executor.XMLCollectionExecutor` executes the `XMLCollectionAccess` and `XMLCollectionFactory` portType operations.

CoreDataAccess::GetCollectionPropertyDocument

This gets each property from the resource and assembles the data resource property document.

XMLCollectionAccess::AddDocuments

This runs the following workflow:

```
AddDocuments(XMLCollection) => DeliverToRequestStatus
```

The status information is retrieved from the request status.

XMLCollectionAccess::GetDocuments

This runs the following workflows:

```
CreateDataSource => DeliverToRequestStatus
```

```
CreateDataSource => DeliverToRequestStatus
```

```
GetDocuments(XMLCollection)
```

```
    => CharArraysToDOM => WriteToDataSource(DataSource)
```

```
    => WriteToDataSource(DataSource)
```

The status information and documents are retrieved from the data sources.

XMLCollectionAccess::RemoveDocuments

This runs the following workflow:

RemoveDocuments(XMLCollection) => DeliverToRequestStatus

The status information is retrieved from the request status.

XMLCollectionAccess::CreateSubcollection

This runs the following workflow:

CreateSubcollection(XMLCollection)

XMLCollectionAccess::RemoveSubcollection

This runs the following workflow:

RemoveSubcollection(XMLCollection)

XMLCollectionFactory::CollectionSelectionFactory

This runs the following workflow:

CreateXMLCollectionResource(XMLCollection) => DeliverToRequestStatus

The ID of the new resource is retrieved from the request status. Information on the services, ports and portTypes on the server (see section 3.1.4) along with the client's preferred service port and portType are used to construct a data resource address for the new resource.

XMLCollectionFactory::DocumentSelectionFactory

This runs the following workflow:

CreateXMLDocumentResource(XMLCollection) => DeliverToRequestStatus

The ID of the new resource is retrieved from the request status. Information on the services, ports and portTypes on the server (see section 3.1.4) along with the client's preferred service port and portType are used to construct a data resource address for the new resource.

XMLCollectionFactory::AddSchema

This is not implemented as it is optional in WS-DAIX.

XMLCollectionFactory::RemoveSchema

This is not implemented as it is optional in WS-DAIX.

XMLCollectionFactory::GetSchema

This is not implemented as it is optional in WS-DAIX.

3.5.4 XPathExecutor

The `uk.org.ogsadai.wsdai.daix.executor.XPathExecutor` executes the `XPathAccess` and `XPathFactory` portType operations. It maintains a resource class to it can ensure that only resources consistent with the service that's ultimately using it are used.

XPathAccess::XPathExecute

This runs the following workflows:

CreateDataSource => DeliverToRequestStatus

XPathQuery(XMLCollection or XMLDocument) => CharArraysToDOM => WriteToDataSource (DataSource)

The query results are retrieved from the data source.

Prior to running the workflow the query, in XQueryX format, is transformed to XPath via application of an XSL transform. A suitable XSL transform document is assumed to be available on the server (see section 4.4).

If used in conjunction with XMLDocuments the executor inserts the document name as an input to the XPathQuery activity.

XPathAccess::XPathExecuteFactory

This runs the following workflows:

CreateXMLSequenceResource => DeliverToRequestStatus

XPathQuery(XMLCollection or XMLDocument) => CharArraysToDOM => WriteToXMLSequence(XMLSequence)

Prior to running the workflow the query, in XQueryX format, is transformed to XPath via application of an XSL transform. A suitable XSL transform document is assumed to be available on the server (see section 4.4).

The ID of the new resource is retrieved from the request status. Information on the services, ports and portTypes on the server (see section 3.1.4) along with the client's preferred service port and portType are used to construct a data resource address for the new resource.

If used in conjunction with XMLDocuments the executor inserts the document name as an input to the XPathQuery activity.

3.5.5 XQueryExecutor

The `uk.org.ogsadai.wsdai.daix.executor.XQueryExecutor` executes the `XQueryAccess` and `XQueryFactory` portType operations. It maintains a resource class to it can ensure that only resources consistent with the service that's ultimately using it are used.

XQueryAccess::XQueryExecute

This runs the following workflows:

CreateDataSource => DeliverToRequestStatus

XQuery(XMLCollection or XMLDocument) => CharArraysToDOM => WriteToDataSource (DataSource)

The query results are retrieved from the data source.

Prior to running the workflow the query, in XQueryX format, is transformed to XQuery via application of an XSL transform. A suitable XSL transform document is assumed to be available on the server (see section 4.4).

If used in conjunction with XMLDocuments the executor inserts the document name as an input to the XQuery activity.

XQueryAccess::XQueryExecuteFactory

This runs the following workflows:

CreateXMLSequenceResource => DeliverToRequestStatus

XQuery(XMLCollection or XMLDocument) => CharArraysToDOM => WriteToXMLSequence(XMLSequence)

Prior to running the workflow the query, in XQueryX format, is transformed to XQuery via application of an XSL transform. A suitable XSL transform document is assumed to be available on the server (see section 4.4).

The ID of the new resource is retrieved from the request status. Information on the services, ports and portTypes on the server (see section 3.1.4) along with the client's preferred service port and portType are used to construct a data resource address for the new resource.

If used in conjunction with XMLDocuments the executor inserts the document name as an input to the XQuery activity.

3.5.6 XUpdateExecutor

The `uk.org.ogsadai.wsdai.daix.executor.XUpdateExecutor` executes the XUpdateAccess portType operations. It maintains a resource class to it can ensure that only resources consistent with the service that's ultimately using it are used.

XUpdateAccess::XUpdateExecute

This runs the following workflow:

XUpdate(XMLCollection or XMLDocument) => DeliverToRequestStatus

The update count is retrieved from the request status.

If used in conjunction with XMLDocuments the executor inserts the document name as an input to the XUpdate activity.

3.5.7 XMLSequenceExecutor

The `uk.org.ogsadai.wsdai.daix.executor.XMLSequence` executes the `XMLSequenceAccess` portType operations. It maintains a resource class to it can ensure that only resources consistent with the service that's ultimately using it are used.

XMLSequenceAccess::GetXMLSequencePropertyDocument

This gets each property from the resource and assembles the data resource property document.

XMLSequenceAccess::GetItems

This runs the following workflows:

CreateDataSource => DeliverToRequestStatus

GetItems(XMLSequence) => CharArraysToDOM => WriteToDataSource (DataSource)

The items are retrieved from the data source.

3.5.8 Data request execution resource

Each executor assumes a data request execution resource with ID "DataRequestExecutionResource" is available in the OGSA-DAI context's resource manager.

3.6 WS-DAIX activities

The following summarises the activities that were especially written for WS-DAIX to enable the implementation of the workflows.

3.6.1 CharArraysToDOM

CharArraysToDOM	Gets lists of char arrays and converts them into org.w3c.dom Document objects.	RMIA
		N/A

CharArrays ToDOM	Gets lists of char arrays and converts them into org.w3c.dom Document objects.			RMIA
				N/A
	<i>Name</i>	<i>Type</i>	<i>Optional</i>	<i>Multiple Occurrences</i>
Input	data	[char[]]	n	n
Output	result	org.w3c.Document	n	n

Implemented by uk.org.ogsadai.wsdai.core.activities.CharArraysToDOMActivity

3.6.2 AddDocuments

AddDocuments	Adds a number of documents to an XML collection. The documents are added to the XML collection associated with the target data resource, or a sub-collection of this if a value is given for the collection input. Each document must be given as a list of char arrays and have an associated name. The output is a status type which has one of five values: SUCCESS, DOES_NOT_VALIDATE, SCHEMA_DOES_NOT_EXIST, NOT_AUTHORIZED, OVERWRITTEN.			RMIA
				N/A
	<i>Name</i>	<i>Type</i>	<i>Optional</i>	<i>Multiple Occurrences</i>
Input	collection	String	y	n
Input	name	String	n	n
Input	data	[char[]]	n	n
Output	result	AddDocumentStatusType	n	n

Implemented by uk.org.ogsadai.wsdai.daix.activities.AddDocumentsActivity

3.6.3 GetDocuments

GetDocuments	Gets a number of documents from an XML collection. The documents are retrieved from the XML collection associated with the target data resource, or a sub-collection of this if a value is given for the collection input. The name input specifies the name of the document to get.			RMIA
				N/A
Each document is output as a list of char arrays and have. Also output is a status type which has one of three values: SUCCESS, DOES_NOT_EXIST, NOT_AUTHORIZED.				

GetDocuments	Gets a number of documents from an XML collection. The documents are retrieved from the XML collection associated with the target data resource, or a sub-collection of this if a value is given for the collection input. The name input specifies the name of the document to get.			RMIA
	Each document is output as a list of char arrays and have. Also output is a status type which has one of three values: SUCCESS, DOES_NOT_EXIST, NOT_AUTHORIZED.			N/A
	<i>Name</i>	<i>Type</i>	<i>Optional</i>	<i>Multiple Occurrences</i>
Input	collection	String	y	n
Input	name	String	n	n
Output	data	[char[]]	n	n
Output	result	GetDocumentStatusType	n	n

Implemented by uk.org.ogsadai.wsdai.daix.activities.GetDocumentsActivity

3.6.4 RemoveDocuments

RemoveDocuments	Removes a number of documents from an XML collection. The documents are removed from the XML collection associated with the target data resource, or a sub-collection of this if a value is given for the collection input. The name input specifies the name of the document to remove.			RMIA
	The activity outputs a status type which has one of three values: SUCCESS, DOES_NOT_EXIST, NOT_AUTHORIZED.			N/A
	<i>Name</i>	<i>Type</i>	<i>Optional</i>	<i>Multiple Occurrences</i>
Input	collection	String	y	n
Input	name	String	n	n
Output	result	RemoveDocumentStatusType	n	n

Implemented by uk.org.ogsadai.wsdai.daix.activities.RemoveDocumentsActivity

3.6.5 CreateSubcollection

CreateSubCollection	Creates a sub-collection in an XML collection. The sub-collection is created in the XML collection associated with the target data resource, or a sub-collection of this if a	RMIA
---------------------	---	-------------

				N/A
	<i>Name</i>	<i>Type</i>	<i>Optional</i>	<i>Multiple Occurrences</i>
Input	subCollection	String	y	n
Input	newCollection	String	n	n

Implemented by uk.org.ogsadai.wsdai.daix.activities.CreateSubCollectionActivity

3.6.6 RemoveSubcollection

RemoveSubCollection	Removes a sub-collection from an XML collection. The sub-collection is removed from the XML collection associated with the target data resource, or a sub-collection of this if a value is given for the subCollection input.			RMIA
				N/A
	<i>Name</i>	<i>Type</i>	<i>Optional</i>	<i>Multiple Occurrences</i>
Input	subCollection	String	y	n
Input	removeCollection	String	n	n

Implemented by uk.org.ogsadai.wsdai.daix.activities.RemoveSubCollectionActivity

3.6.7 CreateXMLCollectionResource

CreateXMLCollectionResource	Creates a new XMLCollection corresponding to the XML collection associated with the target data resource or a sub-collection of this if a value is given for the collectionPath input. The output is the ID of the new resource as a string.			RMIA
				N/A
	<i>Name</i>	<i>Type</i>	<i>Optional</i>	<i>Multiple Occurrences</i>
Input	collectionPath	String	n	n
Output	result	String	n	n

Implemented by uk.org.ogsadai.wsdai.daix.activities.CreateXMLCollectionResourceActivity

3.6.8 CreateXMLDocumentResource

CreateXMLDocumentResource	Creates a new XMLDocument corresponding to a document in the XML collection associated with the target data resource or a sub-collection of this if a value is given for the			RMIA
---------------------------	--	--	--	-------------

				N/A
	<i>Name</i>	<i>Type</i>	<i>Optional</i>	<i>Multiple Occurrences</i>
Input	collectionPath	String	n	n
Input	docName	String	n	n
Output	result	String	n	n

Implemented by
uk.org.ogsadai.wsdai.daix.activities.CreateXMLDocumentResourceActivity

3.6.9 CreateXMLSequenceResource

CreateXMLSequenceResource	Creates a new XMLSequence. The output is the ID of the new resource as a string.			RMIA
				N/A
	<i>Name</i>	<i>Type</i>	<i>Optional</i>	<i>Multiple Occurrences</i>
Output	result	String	n	n

Implemented by
uk.org.ogsadai.wsdai.daix.activities.CreateXMLSequenceResourceActivity

3.6.10 WriteToXMLSequence

WriteToXMLSequence	Inserts an XML fragment (represented as an org.w3c.dom.Node) into an XMLSequence.			RMIA
				N/A
	<i>Name</i>	<i>Type</i>	<i>Optional</i>	<i>Multiple Occurrences</i>
Input	doc	org.w3c.dom.Node	n	n

Implemented by uk.org.ogsadai.wsdai.daix.activities.WriteToXMLSequenceActivity

3.6.11 GetItems

GetItems	Gets XML items held in an XMLSequence as a list of char arrays. The items to be returned are indicated by the start position of the sequence and the count (i.e. number of			RMIA
----------	--	--	--	-------------

				N/A
	<i>Name</i>	<i>Type</i>	<i>Optional</i>	<i>Multiple Occurrences</i>
Input	start	Integer	n	n
Input	count	Integer	n	n
Output	output	[char[]]	n	n

Implemented by uk.org.ogsadai.wsdai.daix.activities.GetItemsActivity

3.6.12 XUpdate

XUpdate	Executes an XUpdate over an XML collection or a sub-collection therein (if the collection input is given) or a document (if the resourceID input is given) therein. The output is a count of the updated nodes.			RMIA
				N/A
	<i>Name</i>	<i>Type</i>	<i>Optional</i>	<i>Multiple Occurrences</i>
Input	collection	String	y	n
Input	resourceId	String	y	n
Input	expression	String	n	n
Output	count	Long	n	n

Implemented by uk.org.ogsadai.wsdai.daix.activities.XUpdateActivity

3.7 OGSA-DAI 3.0 activities

The following OGSA-DAI 3.0 activities are also used.

uk.org.ogsadai.activity.delivery.DeliverToRequestStatusActivity
uk.org.ogsadai.activity.delivery.WriteToDataSourceActivity
uk.org.ogsadai.activity.management.CreateDataSourceActivity
uk.org.ogsadai.activity.xmldb.XPathQueryActivity
uk.org.ogsadai.activity.xmldb.XqueryActivity

OGSA-DAI WS-DAIX executors assume that these activities are recorded in the OGSA-DAI context's activity manager and have been exposed with ID uk.org.ogsadai.ACTIVITY where the activity class-name is ACTIVITYActivity.

3.8 Resources

OGSA-DAI WS-DAIX provides implementations of XMLCollection, XMLDocument and XMLSequences as well as of resource-specific functionality common to all WS-DAI resources as specified in the WS-DAI Core specification.

3.8.1 Core resource commonality

Resource-specific functionality common to all WS-DAI resources as specified in the WS-DAI Core specification is represented in OGSA-DAI WS-DAIX by an interface – `uk.org.ogsadai.wsdai.core.resource.CoreResource` – and an implementing class – `uk.org.ogsadai.wsdai.core.resource.SimpleCoreResource`. This is a super-class of all WS-DAIX resources.

Resource configuration is handled on a resource's behalf by a `uk.org.wsdai.core.resource.CoreResourceState` object. This wraps a generic OGSA-DAI 3.0 `uk.org.ogsadai.resource.dataresource.DataResourceState` object, providing a WS-DAI-specific API for accessing configuration values. `CoreResourceState` manages the following configuration values:

- `ResourceID` – resource ID (see section 4.2).
- `Base resource ID` – ID of the resource's ancestor. If missing, it is assumed the resource has no ancestor.
- `Parent resource ID` – ID of the resource's parent. If missing, it is assumed the resource has no parent.
- `Parent resource port` – Service port through which the parent was requested to create this resource i.e. this port corresponds to a `portType` supporting the indirect access operation used to create the resource.

`CoreResourceState` manages the provision of the following WS-DAI resource properties:

- `DataResourceAbstractName` – constructed using the resource ID.
- `DataResourceManagement`
- `ParentDataResourceAddress` – constructed using the parent resource ID and parent resource port.
- `ConcurrentAccess`
- `DataResourceDescription`
- `Readable`
- `Writable`
- `TransactionInitiation` – hard-coded value of `NotSupported`.
- `TransactionIsolation` – hard-coded value of `NotSupported`.

The following resource properties are assumed to be managed by sub-classes:

- `DatasetMap`
- `ConfigurationMap`
- `LanguageMap`
- `ChildSensitiveToParent`
- `ParentSensitiveToChild`

3.8.2 XMLCollection and XMLDocument commonality

In OGSA-DAI WS-DAIX, XMLCollections and XMLDocuments share common configuration:

- XMLDB collection URI – URI to connect to database e.g. xmldb:exist://coal:9120/exist/xmlrpc.
- Base collection path – name of base collection for this resource e.g /db/littleblackbook.
- Driver class – database driver class name.
- Database login provider – ID of OGSA-DAI 3.0 login provider that provides database usernames and passwords.

This is represented by an interface - `uk.org.ogsadai.wsdai.daix.resource.XMLCollectionConfiguration` interface – and an implementing class - `uk.org.ogsadai.wsdai.daix.resource.XMLResourceState`. `XMLResourceState` extends `CoreResourceState` and manages the above configuration values.

Having XMLCollections and XMLDocuments each maintain driver classes and full collection URIs allows these resources to be used to interact with an XML database even if their parent XMLCollection (if applicable) is destroyed.

3.8.3 XMLCollection

XMLCollections are represented by the `uk.org.ogsadai.wsdai.daix.resource.XMLCollection` class, a sub-class of `CoreResource`. They provide access to collections in an XML database via provision of XMLDB Collection objects.

XMLCollection uses an `uk.org.ogsadai.wsdai.daix.resource.XMLCollectionState` object to manage its configuration. `XMLCollectionState` extends `XMLResourceState`. `XMLCollectionState` manages the provision of the following resource properties:

- `ChildSensitiveToParent` – hard-coded value of Sensitive.
- `ParentSensitiveToChild` – hard-coded value of Sensitive.

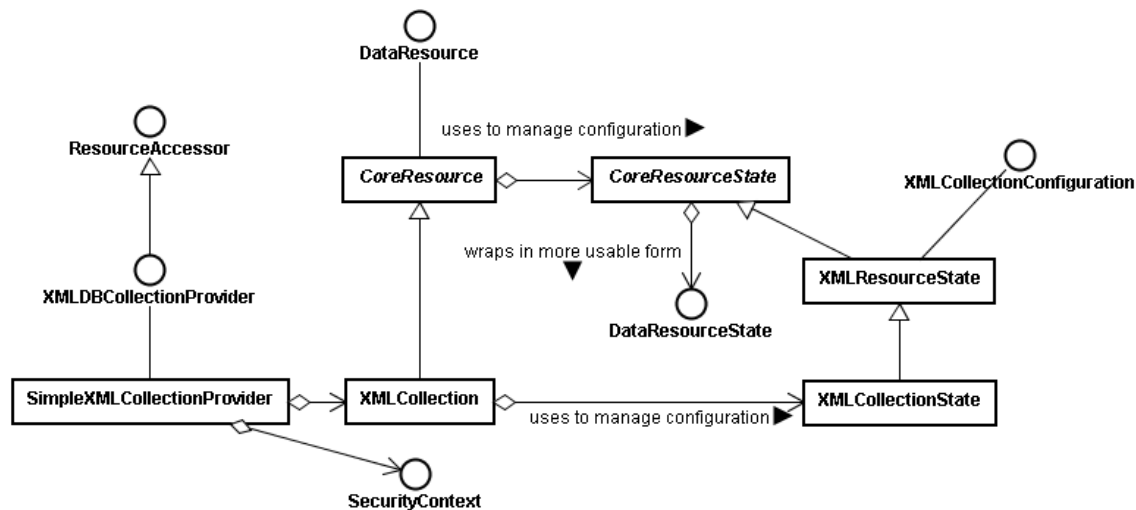
XMLCollection itself manages the provision of the following resource properties:

- `DatasetMap` – hard-coded.
- `LanguageMap` – hard-coded.
- `ConfigurationMap` – hard-coded.
- `TopLevelCollection` – determined via interrogation of the XML database.
- `NumberOfDocuments` – determined via interrogation of the XML database.
- `SupportsCollections` – hard-coded value of True.
- `SupportsCollectionNesting` – hard-coded value of True.
- `SupportsSchemas` – hard-coded value of False.

OGSA-DAI 3.0 resources have associated “providers” (also termed “accessors”). These are wrappers for the resources that associate the resource with a security context from a presentation layer. Activities are given references to providers rather than resources directly. XMLCollection can use this existing OGSA-DAI 3.0

XMLDBCollectionProvider wrapper interface since it too just provides access to an XMLDB Collection object. The class uk.org.ogsadai.wsdaix.resource.SimpleXMLCollectionProvider provides an implementation of XMLDBCollectionProvider for wrapping XMLCollections. This allows XMLCollections to be used within existing OGSA-DAI 3.0 XPath and XQuery activities.

The following diagram shows the relationship between WS-DAIX, WS-DAI and OGSA-DAI 3.0 resource, configuration and provider classes, for XMLCollections.



3.8.4 XMLDocument

XMLDocuments are represented by the uk.org.ogsadai.wsdaix.resource.XMLDocument class, a sub-class of CoreResource. They provide access to documents in collections in an XML database via provision of XMLDB Collection objects.

XMLDocument uses an uk.org.ogsadai.wsdaix.resource.XMLDocumentState object to manage its configuration. XMLDocumentState extends XMLResourceState.

XMLDocumentState manages the following configuration value:

- Document name – the name of a document in an XML collection.

XMLDocumentState manages the provision of the following resource properties:

- ChildSensitiveToParent – hard-coded value of Sensitive.
- ParentSensitiveToChild – hard-coded value of Sensitive.

XMLDocument itself manages the provision of the following resource properties:

- DatasetMap – hard-coded.
- LanguageMap – hard-coded.
- ConfigurationMap – hard-coded.

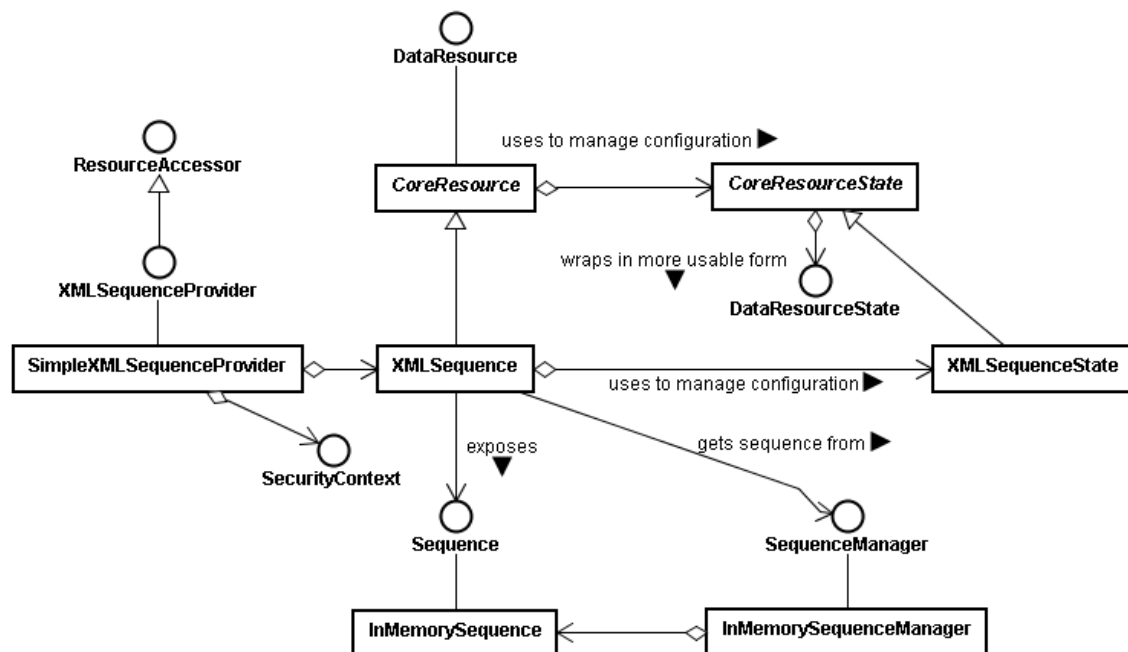
- ParentSensitiveToChild – hard-coded value of Insensitive.

XMLSequence itself manages the provision of the following resource properties:

- DatasetMap – hard-coded.
- LanguageMap – hard-coded value of null.
- ConfigurationMap – hard-coded value of null.
- NumberOfItems – determined via interrogation of the associated sequence.

The interface `uk.org.ogsadai.wsdai.daix.XMLSequenceProvider` and implementing class `uk.org.ogsadai.wsdai.daix.SimpleXMLSequenceProvider` provide a wrapper for the use of XMLSequences with XMLSequence-related activities (e.g. `GetItemsActivity`). This interface provides access to a Sequence object representing the sequence associated with the XMLSequence.

The following diagram shows the relationship between WS-DAIX, WS-DAI and OGSA-DAI 3.0 resource, configuration and provider classes, for XMLSequences.



3.8.6 Sequences and SequenceManagers

An XMLSequence exposes a set of XML items. This is represented via a `uk.org.ogsadai.wsdai.daix.Sequence` interface. A Sequence can be viewed as analogous to a JDBC Connection or XMLDB Collection object. OGSA-DAI WS-DAIX provide an implementation of this interface - `uk.org.ogsadai.wsdai.daix.resource.InMemorySequence` - which just stores the XML items in-memory.

A SequenceManager – represented by the interface `uk.org.ogsadai.wsdai.daix.resource.SequenceManager` – provides access to Sequences. A SequenceManager can be viewed as analogous to a JDBC or XMLDB

database driver. OGSA-DAI WS-DAIX provides an implementation of this interface - `uk.org.ogsadai.wsdai.daix.resource.InMemorySequenceManager` - which creates and manages an in-memory indexed collection of `InMemorySequence` objects.

A sequence manager is placed in the OGSA-DAI context when OGSA-DAI WS-DAIX is deployed.

4. Naming and data formats

This section outlines various features relating to data formats and naming in OGSA-DAI WS-DAIX.

4.1 ResourceSets

The results of `XPathExecute`, `XQueryExecute` and `GetItems` operations return data in the form of OGSA-DAI resource sets. The dataset format URI for an OGSA-DAI resource set is `http://ogsadai.org.uk/data/resourceset`.

The XML representation of a resource set is:

```
<resourceSet>
  <resource>
    CONTENT
  </resource>
  ...
</resourceSet>
```

4.2 Resource IDs and data resource abstract names

OGSA-DAI represents resource identifiers as so-called resource IDs (class `uk.org.ogsadai.resource.ResourceIDs`). These are "."-delimited identifiers. Example resource IDs include: `MyResource`, `org.MyOtherResource`, `uk.org.ogsadai.SomeDAIXResource`.

WS-DAIX represents resource identifiers as data resource abstract names. These are URIs.

In OGSA-DAI WS-DAIX the URI prefix is assumed to be "wsdai" and resource IDs are mapped to and from data resource abstract names as follows:

OGSA-DAI resource ID \Leftrightarrow wsdai:OGSA-DAI resource ID

For example resource ID `MyResource` becomes URI `wsdai:myResource` and resource ID `org.MyOtherResource` becomes URI `wsdai:org.MyOtherResource`.

4.3 Collection names and URIs

WS-DAIX represents collection names as URIs. Within OGSA-DAI WS-DAIX these are represented as strings.

In OGSA-DAI WS-DAIX the URI prefix is assumed to be “wsdai” and collection names are mapped between URIs and strings as follows:

CollectionName ⇔ wsdai:CollectionName

For example collection name /db/mycollection becomes URI wsdai:/db/mycollection.

4.4 XQueryX, XPath, XQuery and XUpdate

OGSA-DAI WS-DAIX uses an XSL transform to convert arguments to XPathExecute, XPathExecuteFactory, XQueryExecute and XQueryExecuteFactory, expressed in XQueryX, into XPath/XQuery expressions.

This XSL style-sheet is that provided by the W3C:

- <http://www.w3.org/TR/xqueryx/>
- <http://www.w3.org/TR/xqueryx/#Stylesheet>

4.5 Sequences

Each XMLSequence uses a sequence manager to manage its sequence of XML items. The sequence manager to use is expected to reside in the OGSA-DAI context and the XMLSequence configuration specifies the name of the sequence manager to use. The XML sequence itself, held by the sequence manager, is identified via a sequence URL, which is also part of the XMLSequence configuration. Sequence URLs are of form:

ogsadai:sequence://SEQUENCE-ID

For example:

ogsadai:sequence://118cc679e15

5. Security

OGSA-DAI WS-DAIX contains no support for securing services or resources.

5.1 Logins, XMLCollections and XMLDocuments

XML data resources in OGSA-DAI WS-DAIX use OGSA-DAI 3.0 login providers to map presentation layer security information (currently just an empty OGSA-DAI 3.0 security context) to database usernames and passwords. The OGSA-DAI context

contains a number of login providers which map security contexts from a presentation layer to database usernames and passwords. Each resource specifies as part of its configuration the login provider to use.

For OGSA-DAI WS-DAIX, XMLCollection and XMLDocuments created by indirect access operations keep a reference to their parent's resource ID and login provider. When the XMLCollection and XMLDocument need to access the associated XML database they use their parent's resource ID and login provider. Child resources adopt the access control of their parent.

5.2 Logins and resource properties

Certain resources need to connect to an XML database to determine resource property values e.g. XMLCollection and its TopLevelCollection property. However OGSA-DAI 3.0 resource property APIs don't allow for security contexts from the presentation layer to be associated with resource property requests. At present a null security context is used.

6. OGSA-DAI WS-DAIX Specification Compliance

This section describes OGSA-DAI WS-DAIX compliance to the WS-DAIX specifications.

6.1 Notation

PortType::Operation OR Input name OR Output name OR Fault name OR Property name	Comments on its implementation in OGSA-DAI DAIX.
* zero or more + one or more ? zero or one	

6.2 portTypes common to all services

The following portTypes are supported by all OGSA-DAI WS-DAIX services.

6.2.1 CoreDataAccess

CoreDataAccess::GetDataResourcePropertyDocument	
DataResourceAbstractName input	

DatasetFormatURI input	Ignored. This is in the specification WSDL but not in the specification text. Assumed it's a typo. - GetDataResourcePropertyDocumentRequest extends RequestType. We suspect this is meant to extend BaseRequestType.
PropertyDocument output	
InvalidResourceNameFault	Thrown if resource is unknown.
DataResourceUnavailableFault	Unused
NotAuthorizedFault	Unused
ServiceBusyFault	Unused

CoreDataAccess::DestroyDataResource	
DataResourceAbstractName input	
InvalidResourceNameFault	Thrown if resource is unknown or if resource has already been destroyed.
DataResourceUnavailableFault	Unused
NotAuthorizedFault	Thrown if resource's DataResourceManagement property is ExternallyManaged.
ServiceBusyFault	Unused

CoreDataAccess::GenericQuery	This is essentially an unimplemented no-op. This operation is optional in the specification.
DataResourceAbstractName input	
DatasetFormatURI input	
<i>GenericExpression input</i>	
(DatasetTypeData, DatasetFormatURI) output	
ServiceBusyFault	
NotAuthorizedFault	Thrown if resource isReadable property is false.
InvalidResourceNameFault	Thrown if resource is unknown.
DataResourceUnavailableFault	Unused
InvalidExpressionFault	Unused
InvalidLanguageFault	Unused
InvalidDatasetFormatFault	Unused

6.2.2 CoreResourceList

CoreResourceList::GetDataResourceList	This operation is optional in the specification.
DataResourceAddress* output	Returns a DataResourceAddress for every resource known to the service. The associated port is that of the CoreResourceList portType.
NotAuthorizedFault	Unused
ServiceBusyFault	Unused

CoreResourceList::Resolve	This operation is optional in WS-DAIX.
DataResourceAbstractName input	
(DataResourceAddress)+ output	Returns a DataResourceAddress of the form shown below. The associated port is picked at random from those available.
InvalidResourceNameFault	Thrown if resource is unknown.
NotAuthorizedFault	Unused
ServiceBusyFault	Unused

Example data resource address with service port and data resource abstract name highlighted.

```
<ns1:DataResourceAddress
  xmlns:ns1="http://www.ggf.org/namespaces/2005/12/WS-DAI/"
  xmlns:ns2="http://www.ggf.org/namespaces/2005/12/WS-DAI"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:type="ns2:DataResourceAddressType">
  <Address xmlns:ns3="http://www.w3.org/2005/08/addressing"
    xsi:type="ns3:AttributedURIType">
    http://coal:9020/dai/services/CollectionServiceXPathAccessPT
  </Address>
  <ReferenceParameters xmlns:ns4="http://www.w3.org/2005/08/addressing"
    xsi:type="ns4:ReferenceParametersType">
    <ns28:DataResourceAbstractName
      xmlns:ns28="http://www.ggf.org/namespaces/2005/12/WS-DAI/">
      wsdai:MyXMLCollection
    </ns28:DataResourceAbstractName>
  </ReferenceParameters>
  <Metadata xmlns:ns5="http://www.w3.org/2005/08/addressing"
    xsi:nil="true" xsi:type="ns5:MetadataType"/>
</ns1:DataResourceAddress>
```

6.3 XMLCollection and XMLCollectionService

6.3.1 CoreDataDescription

DataResourceAbstractName	Always has the resource name.
DataResourceManagement	ServiceManaged or ExternallyManaged.
ConcurrentAccess	Always has value True
DatasetMap	Always returns mappings: XPathExecute => http://ogsadai.org.uk/data/resourceset XQueryExecute => http://ogsadai.org.uk/data/resourceset
LanguageMap	Always returns mappings: XPathExecute => http://www.w3.org/2005/XQueryX XQueryExecute => http://www.w3.org/2005/XQueryX XUpdateExecute => "http://www.xmldb.org/xupdate

ConfigurationMap	
Readable	True OR false depending upon ConfigurationDocument provided to parent.
Writeable	True OR false depending upon ConfigurationDocument provided to parent.
TransactionInitiation	Always has value NotSupported.
TransactionIsolation	Always has value NotSupported.
ChildSensitiveToParent	Always has value Sensitive.
ParentSensitiveToChild	Always has value Sensitive.
ParentDataResource	Ommited for ExternallyManaged resources. Provided for ServiceManaged resources.
DataResourceDescription	Supported. Any XML fragment that can be represented as a single string in an OGSA-DAI WS-DAIX configuration file or is provided in a ConfigurationDocument passed to indirect access operations.

6.3.2 XMLCollectionDescription

TopLevelCollection.	Constructed by querying of the associated XML database. The absolute URIs of sub-collections are specified e.g. if the XMLCollection path is /db/band and this has a sub-collection bangles then this will be listed as /db/band/bangles. Information on schemas is ommited as these are unsupported.
NumberOfDocuments	Constructed by querying of the associated XML database.
SupportsCollections	Always has value true.
SupportsCollectionNesting	Always has value true.
SupportsSchemas	Always has value false.

6.3.3 XMLCollectionAccess

XMLCollectionAccess::GetCollectionPropertyDocument	
DataResourceAbstractName input	
PropertyDocument output	
InvalidResourceNameFault	Thrown if resource is unknown.
DataResourceUnavailableFault	Unused
NotAuthorizedFault	Unused
ServiceBusyFault	Unused

XMLCollectionAccess::AddDocuments	
DataResourceAbstractName input	

CollectionName input	<p>If omitted the collection associated with XMLCollection is assumed.</p> <p>If provided it is expected to be an absolute URI from the root of the XMLCollection. This is the way sub-collections are exposed in an XMLCollectionPropertyDocument.</p>
(DocumentName, XMLWrapper)+ input	Document is expected to be in the first MessageElement of each XMLWrapper's MessageElement array. i.e.: XMLWrapperType.get_any()[0];
(DocumentName, Success DoesNotValidate SchemaDoesNotExist NotAuthorized DocWithNameAlreadyExists, Details)+ output	<p>Details is unused.</p> <p>DocumentDoesNotValidate is unused.</p> <p>SchemaDoesNotExist is unused.</p> <p>NotAuthorized is unused</p>
InvalidResourceNameFault	Thrown if resource is unknown.
DataResourceUnavailableFault	Unused
InvalidCollectionNameFault	Thrown if CollectionName is not a known sub-collection of the collection represented by the resource.
NotAuthorizedFault	Thrown if resource isWriteable property is false. This fault is not cited in the WS-DAIX specification text.
ServiceBusyFault	Unused

XMLCollectionAccess::GetDocuments	
DataResourceAbstractName input	
CollectionName input	<p>If omitted the collection associated with XMLCollection is assumed.</p> <p>If provided it is expected to be an absolute URI from the root of the XMLCollection. This is the way sub-collections are exposed in an XMLCollectionPropertyDocument</p>
(DocumentName)+ input	
(DocumentName, Success NotAuthorized NotExist, XMLWrapper)+ output	<p>NotAuthorized is unused</p> <p>Each document is placed in the first MessageElement of each XMLWrapper's MessageElement array. i.e.: XMLWrapperType.get_any()[0];</p>
InvalidResourceNameFault	Thrown if resource is unknown.
DataResourceUnavailableFault	Unused
InvalidCollectionNameFault	Thrown if CollectionName is not a known sub-collection of the collection represented by the resource.
NotAuthorizedFault	Thrown if resource isReadable property is false. This fault is not cited in the WS-DAIX specification text.

ServiceBusyFault	Unused
------------------	--------

XMLCollectionAccess::RemoveDocuments	
DataResourceAbstractName input	
CollectionName input	If omitted the collection associated with XMLCollection is assumed. If provided it is expected to be an absolute URI from the root of the XMLCollection. This is the way sub-collections are exposed in an XMLCollectionPropertyDocument
(DocumentName)+ input	
(DocumentName, Success NotAuthorized NotExist, Details)+ output	Details is unused. NotAuthorized is unused.
InvalidResourceNameFault	Thrown if resource is unknown.
DataResourceUnavailableFault	Unused
InvalidCollectionNameFault	Thrown if CollectionName is not a known sub-collection of the collection represented by the resource.
NotAuthorizedFault	Thrown if resource isWriteable property is false. This fault is not cited in the WS-DAIX specification text.
ServiceBusyFault	Unused

XMLCollectionAccess::AddSchema	
	This is essentially an unimplemented no-op. This operation is optional in the specification.
DataResourceAbstractName input	
CollectionName input	
Schema input	
Response output	
InvalidResourceNameFault	Thrown if resource is unknown.
DataResourceUnavailableFault	
InvalidCollectionNameFault	
NotAuthorizedFault	
ServiceBusyFault	
SchemaAlreadyExistsFault	
SchemaAdditionMakesDocumentsInvalidFault	
SchemaInvalidFault	

XMLCollectionAccess::GetSchema	
	This is essentially an unimplemented no-op. This operation is optional in the specification.
DataResourceAbstractName input	
CollectionName input	

SchemaNamespace input	
Response output	
InvalidResourceNameFault	Thrown if resource is unknown.
DataResourceUnavailableFault	
InvalidCollectionNameFault	
NotAuthorizedFault	
ServiceBusyFault	
SchemaDoesNotExistFault	

XMLCollectionAccess::RemoveSchema	This is essentially an unimplemented no-op. This operation is optional in the specification.
DataResourceAbstractName input	
CollectionName input	
SchemaNamespace input	
Response output	
InvalidResourceNameFault	Thrown if resource is unknown.
DataResourceUnavailableFault	
InvalidCollectionNameFault	
NotAuthorizedFault	
ServiceBusyFault	
SchemaDoesNotExistFault	
SchemaRemovalMakesDocumentsInvalidFault	
SchemaRemovalMakesSchemaInvalidFault	

XMLCollectionAccess::CreateSubcollection	
DataResourceAbstractName input	
CollectionName input	If omitted the collection associated with XMLCollection is assumed. If provided it is expected to be an absolute URI from the root of the XMLCollection. This is the way sub-collections are exposed in an XMLCollectionPropertyDocument
SubcollectionName input	The URI prefix is ignored. Assumes a single path element e.g. "wsdai:littleblackbook", with no sub-paths. Recursive creation of sub-collections is not supported.
Response output	
InvalidResourceNameFault	Thrown if resource is unknown.
DataResourceUnavailableFault	Unused

InvalidCollectionNameFault	<p>Thrown if CollectionName is not a known sub-collection of the collection represented by the resource.</p> <p>Thrown if SubcollectionName specifies sub-sub-collections e.g. "wsdaix:/littleblackbook/books/"</p>
NotAuthorizedFault	Thrown if resource isWriteable property is false.
ServiceBusyFault	Unused
CollectionAlreadyExistsFault	Thrown if the new collection already exists.

XMLCollectionAccess::RemoveSubCollection	
DataResourceAbstractName input	
CollectionName input	<p>If omitted the collection associated with XMLCollection is assumed.</p> <p>If provided it is expected to be an absolute URI from the root of the XMLCollection. This is the way sub-collections are exposed in an XMLCollectionPropertyDocument</p>
SubcollectionName input	<p>The URI prefix is ignored.</p> <p>Assumes a single path element e.g. "wsdai:littleblackbook", with no sub-paths.</p> <p>Recursive removal of sub-collections is not supported.</p>
Response output	
InvalidResourceNameFault	Thrown if resource is unknown.
DataResourceUnavailableFault	Unused
InvalidCollectionNameFault	<p>Thrown if CollectionName is not a known sub-collection of the collection represented by the resource.</p> <p>Thrown if SubcollectionName specifies sub-sub-collections e.g. "wsdaix:/littleblackbook/books/"</p> <p>Thrown if the collection to remove doesn't exist.</p>
NotAuthorizedFault	Thrown if resource isWriteable property is false.
ServiceBusyFault	Unused

6.3.4 XMLCollectionFactory

XMLCollectionFactory::CollectionSelectionFactory	
DataResourceAbstractName input	
CollectionName input	<p>If omitted the collection associated with XMLCollection is assumed.</p> <p>If provided it is expected to be an absolute URI from the root of the XMLCollection. This is the way sub-collections are exposed in an XMLCollectionPropertyDocument</p>
PortTypeQName input	If none is provided then one from first ConfigurationMap for the resource is used.
ConfigurationDocument input	<p>If none is provided then the one from the ConfigurationMap – chosen based upon the PortTypeQName argument – is used.</p> <p>The new XMLCollection will inherit the isReadable, isWritable and Description property values only.</p>
PreferredTargetService input	<p>This is ignored if:</p> <ul style="list-style-type: none"> • A value is provided but there is no such service currently on the server. • If there is a service on the server but it doesn't support a port corresponding to the preferred portType. <p>If no value is given or a given value is ignored then another service will be selected.</p>
DataResourceAddress output	Returns a data resource address with the DataResourceAbstractName and a port that is one of those supported by an XMLCollection service available on the server.
InvalidResourceNameFault	Thrown if resource is unknown.
DataResourceUnavailableFault	Unused
InvalidCollectionNameFault	Thrown if CollectionName is not a known sub-collection of the collection represented by the resource.
NotAuthorizedFault	Thrown if resource isReadable property is false.
ServiceBusyFault	Unused
InvalidPortTypeQNameFault	Thrown if a PortTypeQName is provided but there is no associated ConfigurationMap for the resource with that portType.
InvalidConfigurationDocumentFault	Thrown if ConfigurationDocument is not an XMLCollectionConfigurationDocumentType.

XMLCollectionFactory::DocumentSelectionFactory	
---	--

DataResourceAbstractName input	
CollectionName input	<p>If omitted the collection associated with XMLCollection is assumed.</p> <p>If provided it is expected to be an absolute URI from the root of the XMLCollection. This is the way sub-collections are exposed in an XMLCollectionPropertyDocument</p>
PortTypeQName input	If none is provided then one from first ConfigurationMap for the resource is used.
ConfigurationDocument input	<p>If none is provided then the one from the ConfigurationMap – chosen based upon the PortTypeQName argument – is used.</p> <p>The new XMLDocument will inherit the isReadable, isWritable and Description property values only.</p>
PreferredTargetService input	<p>This is ignored if:</p> <ul style="list-style-type: none"> • A value is provided but there is no such service currently on the server. • If there is a service on the server but it doesn't support a port corresponding to the preferred portType. <p>If no value is given or a given value is ignored then another service will be selected.</p>
DocumentName input	
DataResourceAddress output	Returns a data resource address with the DataResourceAbstractName and a port that is one of those supported by an XMLCollection service available on the server.
InvalidResourceNameFault	Thrown if resource is unknown.
DataResourceUnavailableFault	Unused
InvalidCollectionNameFault	Thrown if CollectionName is not a known sub-collection of the collection represented by the resource.
DocumentDoesNotExistFault	Thrown if document doesn't exist within requested collection.
NotAuthorizedFault	Thrown if resource isReadable property is false.
ServiceBusyFault	Unused
InvalidPortTypeQNameFault	Thrown if a PortTypeQName is provided but there is no associated ConfigurationMap for the resource with that portType.
InvalidConfigurationDocumentFault	Never.

6.3.5 XPathAccess

XPathAccess::XPathExecute	This operation is optional in WS-DAIX if XQueryAccess is supported.
DataResourceAbstractName input	

DatasetFormatURI input	
XPathExpression input	Must correspond to XPath part of XQueryX
XMLSerializationParameters input	Ignored.
(DatasetFormatURI, DatasetData) output	All the items are combined into a single document placed in the first MessageElement of DatasetData's MessageElement array. i.e.: DatasetDataType.get_any()[0]; DatasetFormatURI is always http://ogsadai.org.uk/data/resourceset
InvalidResourceNameFault	Thrown if resource is unknown.
DataResourceUnavailableFault	Unused
InvalidDatasetFormatFault	Thrown if DatasetFormatURI input is provided but is not http://ogsadai.org.uk/data/resourceset
NotAuthorizedFault	Thrown if resource isReadable property is false.
ServiceBusyFault	Unused
InvalidExpressionFault	Thrown if there is a problem with the XQueryX syntax (particularly if the database returns an exception with message "unexpected token").
XPathFault	Thrown if something goes wrong that is the client's fault in some way and is not covered by InvalidExpressionFault.

6.3.6 XPathFactory

XPathFactory::XPathExecuteFactory	This operation is optional in WS-DAIX if XQueryFactory is supported.
DataResourceAbstractName input	
XPathExpression input	Must correspond to XPath part of XQueryX
PortTypeQName input	If none is provided then one from first ConfigurationMap for the resource is used.
ConfigurationDocument input	If none is provided then the one from the ConfigurationMap – chosen based upon the PortTypeQName argument – is used. The new XMLSequence will inherit the isReadable and Description property values only.
PreferredTargetService input	This is ignored if: <ul style="list-style-type: none"> • A value is provided but there is no such service currently on the server. • If there is a service on the server but it doesn't support a port corresponding to the preferred portType. If no value is given or a given value is ignored then another service will be selected.

DataResourceAddress output	Returns a data resource address with the DataResourceAbstractName and a port that is one of those supported by an XMLSequence service available on the server.
InvalidResourceNameFault	Thrown if resource is unknown.
DataResourceUnavailableFault	Unused
InvalidPortTypeQNameFault	Thrown if a PortTypeQName is provided but there is no associated ConfigurationMap for the resource with that portType.
InvalidConfigurationDocumentFault	Never thrown
NotAuthorizedFault	Thrown if resource isReadable property is false.
ServiceBusyFault	Unused
InvalidExpressionFault	Thrown if there is a problem with the XQueryX syntax (particularly if the database returns an exception with message "unexpected token").
XPathFault	Thrown if something goes wrong that is the client's fault in some way and is not covered by InvalidExpressionFault.

6.3.7 XQueryAccess

XQueryAccess::XQueryExecute	This operation is optional in WS-DAIX if XPathAccess is supported.
DataResourceAbstractName input	
DatasetFormatURI input	
XQueryExpression input	Must correspond to XQuery part of XQueryX
XMLSerializationParameters input	Ignored.
(DatasetFormatURI, DatasetData) output	All the items are combined into a single document placed in the first MessageElement of DatasetData's MessgeElement array. i.e.: DatasetDataType.get_any()[0]; DatasetFormatURI is always http://ogsadai.org.uk/data/resourceset
InvalidResourceNameFault	Thrown if resource is unknown.
DataResourceUnavailableFault	Unused
InvalidDatasetFormatFault	Thrown if DatasetFormatURI input is provided but is not http://ogsadai.org.uk/data/resourceset
NotAuthorizedFault	Thrown if resource isReadable property is false.
ServiceBusyFault	Unused
InvalidExpressionFault	Thrown if there is a problem with the XQueryX syntax (particularly if the database returns an exception with message "unexpected token").

XQueryFault	Thrown if something goes wrong that is the client's fault in some way and is not covered by InvalidExpressionFault.
-------------	---

6.3.8 XQueryFactory

XQueryFactory::XQueryExecuteFactory	This operation is optional in WS-DAIX if XPathFactory is supported.
DataResourceAbstractName input	
XQueryExpression input	Must correspond to XQuery part of XQueryX
PortTypeQName input	If none is provided then one from first ConfigurationMap for the resource is used.
ConfigurationDocument input	If none is provided then the one from the ConfigurationMap – chosen based upon the PortTypeQName argument – is used. The new XMLSequence will inherit the isReadable and Description property values only.
PreferredTargetService input	This is ignored if: <ul style="list-style-type: none"> • A value is provided but there is no such service currently on the server. • If there is a service on the server but it doesn't support a port corresponding to the preferred portType. If no value is given or a given value is ignored then another service will be selected.
DataResourceAddress output	Returns a data resource address with the DataResourceAbstractName and a port that is one of those supported by an XMLSequence service available on the server.
InvalidResourceNameFault	Thrown if resource is unknown.
DataResourceUnavailableFault	Unused
InvalidPortTypeQNameFault	Thrown if a PortTypeQName is provided but there is no associated ConfigurationMap for the resource with that portType.
InvalidConfigurationDocumentFault	Never thrown
NotAuthorizedFault	Thrown if resource isReadable property is false.
ServiceBusyFault	Unused

InvalidExpressionFault	Thrown if there is a problem with the XQueryX syntax (particularly if the database returns an exception with message "unexpected token").
XQueryFault	Thrown if something goes wrong that is the client's fault in some way and is not covered by InvalidExpressionFault.

6.3.9 XUpdateAccess

XUpdateAccess::XUpdateExecute	
DataResourceAbstractName input	
DatasetFormatURI input	Ignored
XUpdateExpression input	Must correspond to XQuery part of XQueryX
Count output	
InvalidResourceNameFault	Thrown if resource is unknown.
DataResourceUnavailableFault	Unused
InvalidDatasetFormatFault	Unused
NotAuthorizedFault	Thrown if resource isWriteable property is false.
ServiceBusyFault	Unused
InvalidExpressionFault	Thrown if there is a problem with the XUpdate syntax (particularly if the database returns an exception with message "unexpected token").
XUpdateSchemaInvalidationFault	Unused
XUpdateFault	Thrown if something goes wrong that is the client's fault in some way and is not covered by InvalidExpressionFault.

6.4 XMLDocument and XMLDocumentService

6.4.1 CoreDataDescription

DataResourceAbstractName	Always has the resource name.
DataResourceManagement	Always has value ServiceManaged.
ConcurrentAccess	Always has value True
DatasetMap	Always returns mappings: XPathExecute => http://ogsadai.org.uk/data/resourceset XQueryExecute => http://ogsadai.org.uk/data/resourceset
LanguageMap	Always returns mappings: XPathExecute => http://www.w3.org/2005/XQueryX XQueryExecute => http://www.w3.org/2005/XQueryX XUpdateExecute => " http://www.xmldb.org/xupdate
Readable	True OR false depending upon ConfigurationDocument provided to parent.

Writeable	True OR false depending upon ConfigurationDocument provided to parent.
TransactionInitiation	Always has value NotSupported.
TransactionIsolation	Always has value NotSupported.
ChildSensitiveToParent	Always has value Sensitive.
ParentSensitiveToChild	Always has value Sensitive.
ParentDataResource	Always provided.
DataResourceDescription	Supported. Any XML fragment that can be represented as a single string in an OGSA-DAI WS-DAIX configuration file or is provided in a ConfigurationDocument passed to indirect access operations.

6.4.2 Other portTypes

The support for other portTypes is as for those of XMLCollection and XMLCollection service as outlined in the previous section.

6.5 XMLSequence and XMLSequenceService

6.5.1 CoreDataDescription

DataResourceAbstractName	Always has the resource name.
DataResourceManagement	Always has value ServiceManaged.
ConcurrentAccess	Always has value True
DatasetMap	Always returns mappings: GetItems => http://ogsadai.org.uk/data/resourceset
LanguageMap	Always has value null.
ConfigurationMap	Always has value null.
Readable	True OR false depending upon ConfigurationDocument provided to parent.
Writeable	Always has value false.
TransactionInitiation	Always has value NotSupported.
TransactionIsolation	Always has value NotSupported.
ChildSensitiveToParent	Always has value Insensitive.
ParentSensitiveToChild	Always has value Insensitive.
ParentDataResource	Always provided.
DataResourceDescription	Supported. Any XML fragment that can be represented as a single string in an OGSA-DAI WS-DAIX configuration file or is provided in a ConfigurationDocument passed to indirect access operations.

6.5.2 XMLSequenceDescription

NumberOfItems	Constructed by querying of the associated sequence.
---------------	---

6.5.3 XMLSequenceAccess

XMLSequenceAccess::GetXMLSequencePropertyDocument	
DataResourceAbstractName input	
XMLSequencePropertyDocument output	
InvalidResourceNameFault	Thrown if resource is unknown.
DataResourceUnavailableFault	Unused
NotAuthorizedFault	Unused
ServiceBusyFault	Unused

XMLSequenceAccess::GetItems	
DataResourceAbstractName input	
DatasetFormatURI input	
StartPosition input	Assumes items are numbered from 0 onwards.
ItemCount input	
XMLSerializationParameters input	Ignored
(DatasetFormatURI, DatasetData) output	All the items are combined into a single document placed in the first MessageElement of DatasetData's MessageElement array. i.e.: DatasetDataType.get_any()[0]; DatasetFormatURI is always http://ogsadai.org.uk/data/resourceset
InvalidResourceNameFault	Thrown if resource is unknown.
DataResourceUnavailableFault	Unused
InvalidDatasetFormatFault	Thrown if DatasetFormatURI input is provided but is not http://ogsadai.org.uk/data/resourceset
NotAuthorizedFault	Thrown if resource isReadable property is false.
ServiceBusyFault	Unused
InvalidStartPositionFault	Thrown if StartPosition < 0 or > NumberOfItems property.
InvalidCountFault	Thrown if ItemCount < 0 or (StartPosition + ItemCount) > NumberOfItems

7. Options for future work

7.1 OGSA-DAI WS-DAIX test coverage

GetDataResourcePropertyDocument tests:

- Check the content of configuration maps in the property document.

GetXMLCollectionPropertyDocument tests:

- Check the content of configuration maps in the property document.

CollectionSelectionFactory tests:

- Check NotAuthorizedFaultType is thrown if the resource is not “readable”.

DocumentSelectionFactory tests:

- Check NotAuthorizedFaultType is thrown if the resource is not “readable”.

RemoveSubcollection tests:

- Check NotAuthorizedFaultType is thrown if the resource is not “writeable”.

CreateSubcollection tests:

- Check NotAuthorizedFaultType is thrown if the resource is not “writeable”.

RemoveDocuments tests:

- Check NotAuthorizedFaultType is thrown if the resource is not “writeable”.

AddDocuments tests:

- Check NotAuthorizedFaultType is thrown if the resource is not “writeable”.

GetDocuments tests:

- Check NotAuthorizedFaultType is thrown if the resource is not “readable”.

Destroy tests:

- Check NotAuthorizedFaultType is thrown if the resource is externally-managed.

Write CoreResource tests.

Write unit and activity tests.

7.2 Error reporting

When the OGSA-DAI 3.0 uk.org.ogsadai.util.xml.XML class is fixed to throw errors depending upon whether internal errors occur or whether the caller has provided badly-formed XML then update the components that use this class, to likewise provide improved error reporting.

Currently if a data resource description configuration property with invalid XML is provided, then a parse exception is gulped and an empty string returned by CoreResourceState.getDataResourceDescription(). Similarly if there is a problem in

converting an argument to `CoreResourceState.setDataResourceDescription()` to a string then an empty string is saved.

7.3 Resource state

Clean configuration maps in `XMLCollection`, `XMLSequence` and `XMLDocument` classes. Use static defaults in these classes or in `DAIXResourcePropertyUtils`. The values are partly dependant upon child resource implementations also e.g. `XMLCollection` and `XMLDocument` ones are dependant upon `XMLSequence`. `XMLCollection` is dependant upon `XMLDocument` e.g. for values such as whether schemas or nested collections are supported.

Allow N data resource description "documents" not just zero or one. Requires using resource-specific persistence and not just the OGSA-DAI 3.0 configuration layer which cannot handle such complexity.

Allow use of language maps, dataset maps and configuration maps provided via XML documents. Requires using resource-specific persistence, as above. Would allow implementations of `setLanguageMap`, `setDatasetMap` and `setConfigurationMap` which are currently no-ops in `Resource` and `ResourceState` classes.

Partition each `Resource` class into an `Resource` interface and `SimpleResource` class. Update configuration files and templates to use `SimpleResource`.

Partition each `ResourceState` class into an `ResourceState` interface and `SimpleResourceState` class. Update configuration files and templates to use `SimpleResourceState`.

Implement `XMLCollection` `SupportsCollections`, `SupportsCollectionNesting` and `SupportsSchemas` resource properties via calls to the database (if the XMLDB API allows access to such information from the database).

7.4 Resource properties

Change `SimpleCoreResource/XMLCollection/XMLDocument/XMLSequence` so that they use the OGSA-DAI 3.0 `RequestExecutionStatusPropertyCallback`-style model i.e. one callback class per resource property. This would avoid the massive IFs currently in the `get/setResourcePropertyValue()` methods. Each callback object should be created in the resource `initialize()` methods. An OGSA-DAI 3.0 example is in `uk/org/ogsadai/resource/request/RequestStatusPropertyCallback.java`

Change `XMLCollection/Document/Sequence` `get/setResourcePropertyValue` to use a default DN or username/password configuration property to provide a slight improvement on using a null security context,

Change `XMLCollection/Document/Sequence` `get/setResourcePropertyValue` to take a `SecurityContext` as an argument. This removes need for the above but requires an OGSA-DAI 3.0 API change.

Add to `OnDemandResourcePropertyCallbackException` a `(ResourceID, ResourcePropertyName, Throwable)` constructor and use this instead of `initCause()`. This requires an OGSA-DAI 3.0 API change.

Implement `SimpleDAISResourcePropertyValue` `getAsDOM()`. This requires implementing WS-DAI resource property value \Leftrightarrow DOM conversion. Lack of such an implementation means that WS-DAI resource properties cannot be accessed via `WS-ResourceProperties` if the resources are used in a standard OGSA-DAI 3.0 deployment (i.e. outwith the WS-DAI presentation layer).

7.5 *InMemorySequence*

Implement the unused `addItems(Node[])` method – it's currently a no-op.

7.6 *Executors*

Reassess use of singleton object for the executors within providers since this may cause problems if two threads access the executor at the same time.

Randomize candidate service selection if a preferred service is not selected by the client.

Change `WSDAIContextInitializer` and `Executors` to use not `SimpleRequestFactory` but an XML document-based request factory and store workflows as XML documents.

Devise a nicer approach to handling `DAIXRequestStatusExceptionHandler` which doesn't require assuming activity instance names based on whether they were XPath, XQuery or XUpdate activities.

7.7 *XMLDocuments and XPath, XQuery, XUpdate*

`XMLDocument` currently is used with `SimpleXMLDocumentProvider`, an `XMLDBCcollectionProvider-complaint` wrapper for use with OGSA-DAI 3.0 XML activities. This, in theory, allows `XMLDocument` to be successfully used by any activity applicable to `XMLDBCcollectionProvider-compliant` resources. However, this includes activities such as `XMLListCollections` which are not in the spirit of an `XMLDocument`.

An additional problem is the requirement for executors to provide the document name as an input to XPath, XQuery and XUpdate activities. This is a property of the resource so these activities should extract the ID from the resource.

One solution is to have these activities check for an additional interface e.g. a new `XMLDocumentProvider` interface which supports a `getDocumentName()` method. If the activity is given an instance of this interface then they use the document ID accessed via this interface and ignore any document name provided by an `inputLiteral`. However, the risk of using `XMLListCollections` with `XMLDocument` still remains.

One cause of this is the fact that the XMLDB Collection object provides the component to run XPath, XQuery or XUpdate statement upon them and then the specific document in the collection is specified via a method on the objects that implement these queries i.e.

```
import org.xmldb.api.base.Collection;
import org.xmldb.api.modules.XPathQueryService;

Collection myCollection = . . . ;
XPathQueryService xPath =
    (XPathQueryService) collection.getService("XPathQueryService","1.0");

xPath.query(expression);

xPath.queryResource(documentName, expression);
```

One solution therefore is to define three new XMLDB-related provider interfaces for resources – XPathProvider, XQueryProvider and XUpdateProvider – which provide methods to return XMLDB XPathQueryService, XQueryService and XUpdateQueryService respectively. Creation of the components to run the query is moved from the associated activities to the resource itself. This would mean a change in the XPath, XQuery and XUpdate activities to expect resource providers that implement these interfaces (or writing new versions of these activities). It would also mean a change to OGSA-DAI 3.0's SimpleXMLDBCollectionProvider to support these interfaces also. This means that OGSA-DAI 3.0's XMLDBDataResource could continue to be used with the updated activities.

For OGSA-DAI WS-DAIX, it would require changing XMLDocument, writing a new accessor/provider wrapper that only implemented these interfaces.

7.8 Activities

Define XQueryXtoXPath activity and move the conversion from XPathExcutor into the workflow.

Define XQueryXtoXQuery activity and move the conversion from XQueryExecutor into the workflow.

Define CharArraysToString activity. Allows:

- XSLTransform => CharArraysToString => XPath
- XSLTransform => CharArraysToString => XQuery

Implement application of XMLSerializationParameters as a transform activity.

Activities that create an XMLDB Collection should call XMLDBAccessor.release(Collection).

7.9 Presentation layers

Adapt presentation layer to use a security implementation e.g. OMII security.

Provide additional abstractions so resources, executors and utilities are not reliant upon auto-generated beans at all. This would make resources, executors and utilities indifferent to changes in auto-generated bean signatures which can arise due to differences in versions of Apache Axis. It would also allow use of these components under non-Axis-based presentation layers.

The portType providers provide a layer in which conversion to and from auto-generated beans can be done.

Provide a Globus Toolkit-compliant presentation layer.

7.10 Security

Dynamically add a mapping from each newly-created resource to a database username and password to a login provider.

7.11 DRER capacity and ServiceBusyFaultType

Change executors so that ServiceBusyFaultType is thrown if the DRER is processing its maximum number of requests and/or the queue is full i.e. RequestRejectedException is detected. Alternatively, DataResourceUnavailableFaultType could be thrown.

7.12 Saving workflows as XML files

Currently workflows are constructed in-code using activity framework objects and then passed directly to the activity framework via an OGSA-DAI. Another option is to store these server-side in XML and have a XML file-based request factory that, given a file name or workflow ID, loads in the file and converts this to activity framework objects.

7.13 Clients

Refactor the uk.org.ogsadai.wsdai.client.toolkit classes to ensure that the APIs exposed to applications developers do not expose any WSDL2Java auto-generated classes.

8. OGSA-DAI 3.0 limitations identified

Fix uk.org.ogsadai.util.xml.XML to throw errors depending upon whether internal errors occur or whether the caller has provided badly-formed XML. Currently all errors are thrown as IllegalArgumentException .

Change `get/setResourcePropertyValue` to take a `SecurityContext` as an argument.

Add to `OnDemandResourcePropertyCallbackException` a `(ResourceID, ResourcePropertyName, Throwable)` constructor and use this instead of `initCause()` in all dependant classes.

The server-side activity input/output name should be public to allow in-code server workflow construction e.g. `uk.org.ogsadai.activity.delivery.DeliverToRequestStatusActivity`'s `RESULT_NAME` field should be public.

Copy `DAIXUtils.nodeToString/2` to `XML.java`.

8.1 A limitation for multiple presentation layers

One desirable goal was that OGSA-DAI WS-DAIX services run in the same web application alongside standard OGSA-DAI 3.0 services i.e. the same resources could be accessed either via workflows and/or via WS-DAIX operations. However the current OGSA-DAI 3.0 architecture allows only one request factory in the OGSA-DAI server. So two presentation layers with two different request factories cannot co-exist at present.

The limitation arises in the `uk.org.ogsadai.resource.initialise.FactoryResourceStateVisitor.visitDRER` method. This does:

```
RequestFactory requestFactory =
    (RequestFactory)context.get(OGSADAIConstants.REQUEST_FACTORY);
mResource = new SimpleDRER(requestFactory);
mResource.initialize(resourceState);
```

So each DRER is committed to using the same request factory and there is only one request factory stored in the OGSA-DAI context.

A solution is to change `SimpleDRER`'s `initialise` method so that it gets a value from the DRER's configuration properties. This value could specify the name of the request factory the DRER is to use – the DRER then could get this from the OGSA-DAI context. If there is no request factory then the default one provided by `FactoryResourceStateVisitor` in the DRER's constructor can be used. Instead of one request factory per presentation layer there is one per DRER. This solution does yield a new version of `SimpleDRER` which can use accept an optional additional configuration property but doesn't entail any changes to method signatures. However if OGSA-DAI WS-DAIX were to co-exist with OGSA-DAI 3.0 then there would need to be two DRERs for the server – one for OGSA-DAI 3.0's presentation layer and using its request factory and one for the OGSA-DAI WS-DAIX.

A third solution is to write a request factory that aggregates other request factories and dispatches requests to the appropriate one. It could maintain a keyed collection of request factories with one as the default. When the aggregated request factory's

```
public Request createRequest(RequestDescriptor descriptor,  
                             RequestConfiguration context);
```

method is called it just delegates the call on to the createRequest method of the default request factory. However if in the RequestConfiguration it finds, for example, a REQUEST_FACTORY key-value pair then it delegates to the request factory whose key matches the given value. Thus each presentation layer can determine which request factory should be used. At initialisation time WSDAIContextInitializer would create this aggregated request factory and add the OGSA-DAI WS-DAIX-specific request factory to it. It would also add any current request factory in the context and set this as the default.

9. OGSA-DAI 3.0 ingestion

The following changes are required to allow the use of OGSA-DAI WS-DAIX resources and activities within a standard OGSA-DAI 3.0 deployment.

- Change CreateXMLSequenceResourceActivity to take an optional ResourceID input parameter.
- Change CreateXMLCollectionResourceActivity to take an optional ResourceID input parameter.
- Change CreateXMLDocumentResourceActivity to take an optional ResourceID input parameter.
- Change AddDocumentsActivity and remove the use of the status output.
- Change GetDocumentsActivity and remove the use of the status output and the dummy document.
- Change RemoveDocumentsActivity and remove the use of the status output.
- Change XMLCollection to not use NullSecurityContext but to take it in as an argument to get/setResourcePropertyValue – see section 7.4 and section 8.
- Change XMLSequence to not use NullSecurityContext but to take it in as an argument to get/setResourcePropertyValue – see section 7.4 and section 8.