

OGSA-DAI WS-DAIR 1.0

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

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

Version: 1.0

Date: December 2008.

Contents

Contents.....	1
1. Introduction.....	2
1.1 OGF WS-DAI specifications	2
2. Services and resources.....	3
3. OGSA-DAI WS-DAIR architecture.....	4
3.1 Services.....	4
3.1.1 WSDL	5
3.1.2 WSDL and auto-code generation	5
3.1.3 WSRF.....	5
3.1.4 Service configuration and indices	5
3.2 portTypes and providers.....	6
3.3 OGSA-DAI server context	7
3.3.1 Specifying the OGSA-DAI workflow parser.....	7
3.4 Services / resource relationships	7
3.5 Executors.....	8
3.5.1 CoreDataAccessExecutor.....	8
3.5.2 CoreResourceListExecutor	8
3.5.3 SQLAccessExecutor.....	9
3.5.4 SQLResponseExecutor	10
3.5.5 SQLRowsetExecutor	12
3.5.6 Data request execution resource.....	12
3.6 WS-DAIR activities	12
3.6.1 CharArraysToDOM	13
3.6.2 SQLStatement.....	13
3.6.3 WriteToSQLResponse	13
3.6.4 ReadFromSQLResponse	14
3.6.5 CreateSQLResponseResource	15
3.6.6 CreateSQLRowsetResources	15
3.6.7 GetTuples.....	16
3.7 OGSA-DAI 3.1 activities	16
3.8 Resources	17
3.8.1 Core resource commonality.....	17
3.8.2 SQLAccess.....	18
3.8.3 SQLResponse	19
3.8.4 SQLResponseData and SQLResourceManagers	20
3.8.5 SQLRowset	21
3.8.6 Rowset.....	22
4. Naming and data formats	22
4.1 WebRowSet and CSV	23
4.2 Resource IDs and data resource abstract names	25
4.3 SQLResponseData	26
4.4 Rowset	26

5.	Security	26
5.1	Logins, SQLAccesses	26
5.2	Logins and resource properties	26
6.	OGSA-DAI WS-DAIR 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	SQLAccess and SQLAccessService	29
6.3.1	CoreDataDescription	29
6.3.2	SQLAccessDescription	29
6.3.3	SQLAccess	29
6.3.4	SQLAccessFactory	30
6.4	SQLResponse and SQLResponseService	31
6.4.1	CoreDataDescription	31
6.4.2	SQLResponseDescription	32
6.4.3	SQLResponse	32
6.4.4	SQLResponseFactory	35
6.5	SQLRowset and SQLRowsetService	36
6.5.1	CoreDataDescription	36
6.5.2	SQLRowsetDescription	36
6.5.3	SQLRowsetAccess	36
7.	Options for future work	37
7.1	Error reporting	37
7.2	Resource state	37
7.3	Resource properties	38
7.4	Executors	39
7.5	Presentation layers	39
7.6	Security	39
7.7	DRER capacity and ServiceBusyFaultType	39
7.8	Saving workflows as XML files	39
7.9	Clients	40
7.10	Additional SQL Exception handlers	40
8.	OGSA-DAI 3.1 limitations identified	40
9.	OGSA-DAI 3.1 ingestion	40

1. Introduction

This document describes an implementation of the OGF WS-DAIR specification using OGSA-DAI 3.1 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 3.1 components.

OGSA-DAI WS-DAIR 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-DAIR 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 Relational Realization (WS-DAIR) Specification, Version 1.0, 20th July 2006. GFD 76 (WS-DAIR). <http://www.ogf.org/documents/GFD.76.pdf>

2. Services and resources

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

- SQLAccess – represents a relational database.
- SQLResponse – represents the results from an SQL execution on a relational database.
- SQLRowset – represents a result set which is being held by a SQLResponse resource.

The WS-DAIR specifications do not specify WS-DAIR 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-DAIR:

- SQLAccess service
 - CoreDataAccess, CoreResourceList
 - SQLAccess, SQLAccessFactory
- SQLResponse service
 - CoreDataAccess, CoreResourceList
 - SQLResponse, SQLResponseFactory
- SQLRowset service
 - CoreDataAccess, CoreResourceList
 - SQLRowsetAccess

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

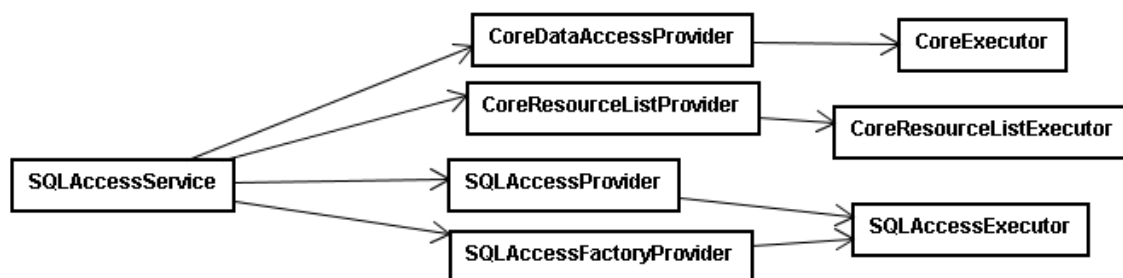


Figure 1: SQLAccess service, portType providers and executors

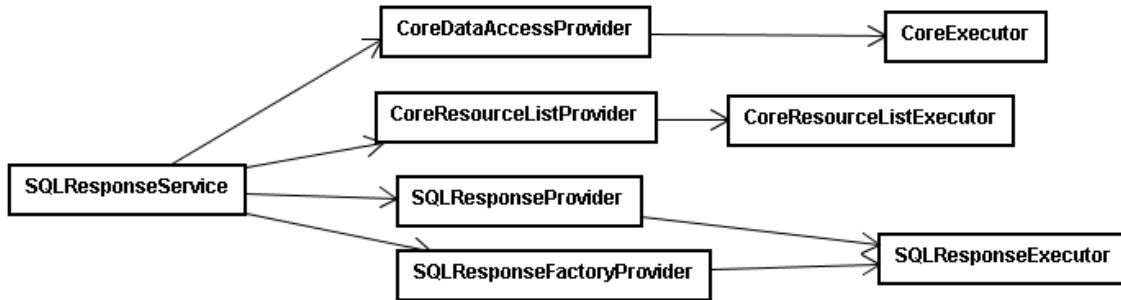


Figure 2: SQLResponse service , portType providers and executors

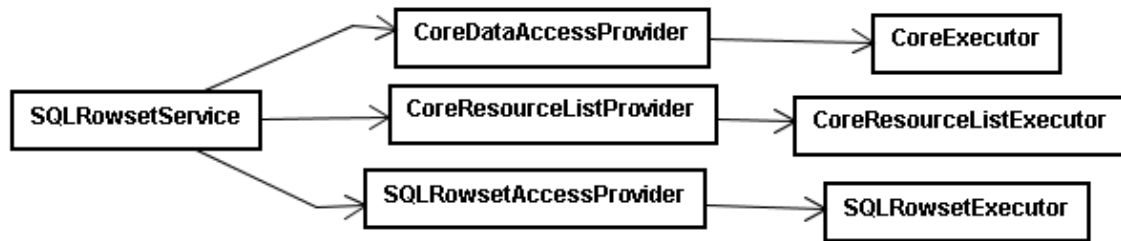


Figure 3: SQLRowset service, portType providers and executors

3. OGSA-DAI WS-DAIR architecture

This section outlines the OGSA-DAI WS-DAIR 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-DAIR operations either directly, or by delegation to a resource, or by the execution of OGSA-DAI workflows.
- Resources – WS-DAIR resources which manage access to data (Relational databases and responses and rowsets).

3.1 Services

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

- uk.org.ogsadai.wsdai.dair.service.SQLAccessService
- uk.org.ogsadai.wsdai.dair.service.SQLResponseService
- uk.org.ogsadai.wsdai.dair.service.SQLRowsetService

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-DAIR server (see section **Error! Reference source not found.**) when the service is first contacted and initialised.

3.1.1 WSDL

OGSA-DAI WS-DAIR 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 the following five validation errors:

- The part 'GetSQLPropertyDocumentRequest' has an invalid value 'GetDataResourcePropertyDocumentRequest' defined for its element. Element declarations must refer to valid values defined in a schema.
- The part 'GetSQLResponsePropertyDocumentRequest' has an invalid value 'GetDataResourcePropertyDocumentRequest' defined for its element. Element declarations must refer to valid values defined in a schema.
- Cannot resolve the name 'wrs:metadata' to a(n) 'element declaration' component.
- The part 'GetSQLRowsetPropertyDocumentRequest' has an invalid value 'GetDataResourcePropertyDocumentRequest' defined for its element. Element declarations must refer to valid values defined in a schema.

These issues will be communicated to the WS-DAIR specification authors. They do not affect the use of OGSA-DAI WS-DAIR via Apache Axis-based clients.

3.1.2 WSDL and auto-code generation

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

3.1.3 WSRF

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

3.1.4 Service configuration and indices

The OGSA-DAI WS-DAIR 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.: `ExampleSQLAccessService`.
- A service type (of type `uk.org.ogsadai.common.ID` – basically a “.”-delimited string) which is of one of the following values:
 - `wsdai.SQLAccessService`
 - `wsdai.SQLResponseService`
 - `wsdai.SQLRowsetService`
- A URL prefix for all the service ports belonging to a service in the OGSA-DAI WS-DAIR 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 `SQLAccess` service may have a port `AccessServiceAccessPT` which implements the WS-DAIR portType `{http://www.ggf.org/namespaces/2005/12/WS-DAIR/}SQLAccessPT`.
 - A port appended to the URL prefix should specify a valid endpoint for the service.

OGSA-DAI WS-DAIR 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 3.1 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 a `SQLAccess` 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-DAIR portType:

- `uk.org.ogsadai.wsdai.core.provider.CoreDataAccessProvider`

- uk.org.ogsadai.wsdai.core.provider.CoreResourceListProvider
- uk.org.ogsadai.wsdai.dair.provider.SQLAccessProvider
- uk.org.ogsadai.wsdai.dair.provider.SQLAccessFactoryProvider
- uk.org.ogsadai.wsdai.dair.provider.SQLResponseAccessProvider
- uk.org.ogsadai.wsdai.dair.provider.SQLResponseFactoryProvider
- uk.org.ogsadai.wsdai.dair.provider.SQLRowsetProvider

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 3.1 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-DAIR includes the service, port and portType maps of section 3.1.4, login providers (which manage database logins for SQLAccess), resource managers (which manage the results exposed by SQLResponses) and exception handlers (which manage which vendor-specific SQLExceptions will be thrown as exceptions or will be part of the SQL execution response). In OGSA-DAI WS-DAIR 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-DAIR implements many WS-DAIR 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-DAIR 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-DAIR there is no explicit service-resource relationship maintained server-side. If one were to deploy two SQLAccess services then all SQLAccesses 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-DAIR 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 3.1 activities plus a set of activities developed for OGSA-DAI WS-DAIR.

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 used with `SQLResponses` via an `SQLResponse` 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-DAIR.

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 to return the data resource address for the given resource.

3.5.3 SQLAccessExecutor

The uk.org.ogsadai.wsdai.dair.executor.SQLAccessExecutor executes the SQLAccess and SQLAccessFactory portType operations.

SQLAccess::GetSQLPropertyDocument

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

SQLAccess::SQLExecute

This runs the following workflow:

CreateDataSource => DeliverToRequestStatus

CreateDataSource => DeliverToRequestStatus

CreateDataSource => DeliverToRequestStatus

CreateDataSource => DeliverToRequestStatus

CreateDataSource => DeliverToRequestStatus

Then

```
=> ListRemove => TupleToWebRowSet => CharArraysToDOM =>
    WriteToDataSource(DataSource)
=> WriteToDataSource (DataSource)
SQLStatement(SQLAccess)
=> WriteToDataSource (DataSource)
=> WriteToDataSource (DataSource)
=> WriteToDataSource (DataSource)
```

OR (depending on the datasetFormatURI)

```
=> ListRemove => TupleToCSV => WriteToDataSource (DataSource)
=> WriteToDataSource (DataSource)
SQLStatement(SQLAccess)
=> WriteToDataSource (DataSource)
=> WriteToDataSource (DataSource)
=> WriteToDataSource (DataSource)
```

This basically uses one data source for each type of result item e.g. result sets, update counts, output parameters, return values, communication areas). The results from the SQL execution are retrieved from the data sources.

SQLAccessFactory::SQLExecuteFactory

This runs the following workflows:

CreateSQLResponseResource => DeliverToRequestStatus

=> WriteToSQLResponse(SQLResponse)
=> WriteToSQLResponse

SQLStatement(SQLAccess)

=> WriteToSQLResponse
=> WriteToSQLResponse
=> WriteToSQLResponse

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 list of data resource addresses for the new resource.

3.5.4 SQLResponseExecutor

The `uk.org.ogsadai.wsdai.dair.executor.SQLResponseExecutor` executes the `SQLResponse` and `SQLResponseFactory` portType operations.

SQLResponse::GetSQLResponsePropertyDocument

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

SQLResponse::GetSQLRowset

This runs the following workflows:

CreateDataSource => DeliverToRequestStatus

then

ReadFromSQLResponse(SQLResponse) => ListRemove => TupleToCSV =>
WriteToDataSource (DataSource)

OR (depending on the datasetFormatURI)

ReadFromSQLResponse(SQLResponse) => ListRemove => TupleToWebRowSet =>
CharArraysToDOM => WriteToDataSource (DataSource)

The items are retrieved from the data source.

SQLResponse::GetSQLUpdateCount

This runs the following workflows:

CreateDataSource => DeliverToRequestStatus

ReadFromSQLResponse(SQLResponse)=> WriteToDataSource (DataSource)

The items are retrieved from the data source.

SQLResponse::GetSQLOutputParameter

This runs the following workflows:

CreateDataSource => DeliverToRequestStatus

ReadFromSQLResponse(SQLResponse)=> WriteToDataSource (DataSource)

The items are retrieved from the data source.

SQLResponse::GetSQLReturnValue

This runs the following workflows:

CreateDataSource => DeliverToRequestStatus

ReadFromSQLResponse(SQLResponse)=> WriteToDataSource (DataSource)

The items are retrieved from the data source.

SQLResponse::GetSQLCommunicationArea

This runs the following workflows:

CreateDataSource => DeliverToRequestStatus

ReadFromSQLResponse(SQLResponse)=> WriteToDataSource (DataSource)

The items are retrieved from the data source.

SQLResponse::GetSQLOutputParameter

This runs the following workflows:

CreateDataSource => DeliverToRequestStatus

ReadFromSQLResponse(SQLResponse)=> WriteToDataSource (DataSource)

The items are retrieved from the data source.

SQLResponse::GetSQLResponseItem

This operation does not run directly any workflows; it calls the methods implementing the GetSQLRowset, GetSQLUpdateCount, GetSQLOutputParameter, GetSQLReturnValue, GetSQLCommunicationArea operations to construct the SQLResponseItem.

SQLResponseFactory::SQLRowsetFactory

This runs the following workflow:

```
CreateSQLRowsetResources => DeliverToRequestStatus
```

The IDs of the new resources are 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 list of data resource addresses for the new resources.

3.5.5 SQLRowsetExecutor

The `uk.org.ogsadai.wsdai.dair.executor.SQLRowset` executes the `SQLRowsetAccess` portType operations.

SQLRowsetAccess::GetSQLRowsetPropertyDocument

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

SQLRowsetAccess::GetTuples

This runs the following workflows:

```
CreateDataSource => DeliverToRequestStatus
```

then

```
GetTuples(SQLRowset) => TupleToWebRowSet => CharArraysToDOM =>  
WriteToDataSource (DataSource)
```

OR (depending on the datasetFormatURI)

```
GetTuples(SQLRowset) => TupleToCSV => WriteToDataSource (DataSource)
```

The items are retrieved from the data source.

3.5.6 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-DAIR activities

The following summarises the activities that were especially written for WS-DAIR 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
	<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 SQLStatement

SQLStatement	It executes any SQL statement based on the given SQL expression and the optional SQL parameters.			RMIA
				N/A
	<i>Name</i>	<i>Type</i>	<i>Optional</i>	<i>Multiple Occurrences</i>
Input	expression	String	n	n
Input	parameters	[uk.org.ogsadai.wsdai.dair.activities.SQLParameter]	y	n
Output	data	[[tuple]]	n	n
Output	count	[Integer]	n	n
Output	value	String	n	n
Output	outputParameters	HashMap	n	n
Output	commAreas	[SQLException]	n	n

Implemented by uk.org.ogsadai.wsdai.dair.activities.SQLStatementActivity

3.6.3 WriteToSQLResponse

WriteToSQLResponse	Inserts the results of an SQL execution into a SQLResponse.			RMIA
				N/A

WriteToSQLResponse	Inserts the results of an SQL execution into a SQLResponse.			RMIA
				N/A
	<i>Name</i>	<i>Type</i>	<i>Optional</i>	<i>Multiple Occurrences</i>
Input	data	[[tuple]]	n	n
Input	count	[Integer]	n	n
Input	value	String	n	n
Input	outputParameters	HashMap	n	n
Input	commAreas	[SQLException]	n	n

Implemented by uk.org.ogsadai.wsdai.dair.activities.WriteToSQLResponseActivity

3.6.4 ReadFromSQLResponse

ReadFromSQLResponse	Retrieves some of the items being held by the SQLResponse. The type of items to be returned must be specified by providing on the corresponding input one of: "data", "updateCounts", "outputParameters", "returnValue", "sqlException". The items to be returned are indicated by the start position and the count (i.e. the number of items to be returned). As a result the output depends on the requested types of items to be retrieved.			RMIA
				N/A
	<i>Name</i>	<i>Type</i>	<i>Optional</i>	<i>Multiple Occurrences</i>
Input	start	Integer	y	n
Input	count	Integer	y	n
Input	type	String	n	n

ReadFromSQLResponse	Retrieves some of the items being held by the SQLResponse. The type of items to be returned must be specified by providing on the corresponding input one of: "data", "updateCounts", "outputParameters", "returnValue", "sqlException". The items to be returned are indicated by the start position and the count (i.e. the number of items to be returned). As a result the output depends on the requested types of items to be retrieved.			RMIA
				N/A
Output	output	One of: [[tuple]], [Integer], String, HashMap, [SQLException]	n	n

Implemented by uk.org.ogsadai.wsdai.dair.activities.ReadFromSQLResponseActivity

3.6.5 CreateSQLResponseResource

CreateSQLResponseResource	Creates a new SQLResponse. The output is the ID of the 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.dair.activities.CreateSQLResponseResourceActivity

3.6.6 CreateSQLRowsetResources

CreateSQLRowsetResources	Creates a number of new SQLRowset resources. The output is a list of the IDs of the resources as Strings. Since the SQLRowset resources represent rowsets being held by a SQLResponse, the activity has start and count inputs that indicate which items are to be returned.			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

CreateSQLRowsetResources	Creates a number of new SQLRowset resources. The output is a list of the IDs of the resources as Strings. Since the SQLRowset resources represent rowsets being held by a SQLResponse, the activity has start and count inputs that indicate which items are to be returned.			RMIA
				N/A
Output	result	[String]	n	n

Implemented by `uk.org.ogsadai.wsdai.dair.activities.CreateSQLRowsetResourcesActivity`

3.6.7 GetTuples

GetTuples	Gets a number of tuples being stored in a SQLRowset resource. Therefore start and count inputs are needed to indicate which tuples to be retrieved. There is also an accMode input whose values can be either "forward" or "random" and indicates if the tuples can be navigated in forward only or random access mode.			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
Input	accMode	String	n	n
Output	output	[tuple]	n	n

Implemented by `uk.org.ogsadai.wsdai.dair.activities.GetTuplesActivity`

3.7 OGSA-DAI 3.1 activities

The following OGSA-DAI 3.1 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.transform.TupleToCSVActivity`
`uk.org.ogsadai.activity.transform.TupleToWebRowSetCharArraysActivity`
`uk.org.ogsadai.activity.transform.TableMetadataToXMLCharArraysListActivity`
`uk.org.ogsadai.activity.block.ListRemove`
`uk.org.ogsadai.activity.sql.ExtractTableSchemaActivity`

OGSA-DAI WS-DAIR 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-DAIR provides implementations of `SQLAccess`, `SQLResponse` and `SQLRowsets` 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-DAIR 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-DAIR 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 3.1 `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`
- `Writeable`
- `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 SQLAccess

In OGSA-DAI WS-DAIR, SQLAccess has the following configuration:

- SQL Database URI – URI to connect to database e.g. jdbc:mysql://localhost:3307/ogsadai
- Driver class – database driver class name.
- Database login provider – ID of OGSA-DAI 3.1 login provider that provides database usernames and passwords.
- SQL exception handler – ID of the exception handler that is managing the SQLExceptions. The value of this ID is then picked up through the OGSA-DAI context.

This is represented by an implementing class - uk.org.ogsadai.wsdai.dair.resource.SQLAccessResourceState which extends CoreResourceState and manages the above configuration values.

SQLAccess resources are represented by the uk.org.ogsadai.wsdai.dair.resource.SQLAccessResource class, a sub-class of CoreResource. They provide access to relational databases via provision of JDBC objects.

SQLAccess uses the aforementioned uk.org.ogsadai.wsdai.dair.resource.SQLAccessResourceState object to manage its configuration. SQLAccessResourceState also manages the provision of the following resource properties:

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

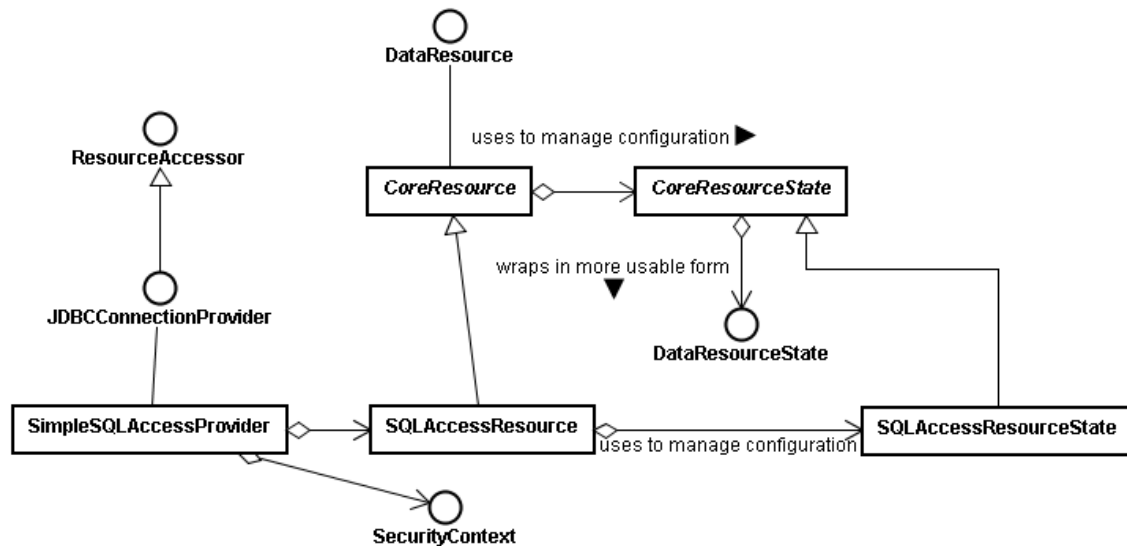
SQLAccess itself manages the provision of the following resource properties:

- DatasetMap – hard-coded.
- LanguageMap – hard-coded.
- ConfigurationMap – hard-coded.
- SchemaDescription – determined via execution of the following workflow against the relational database:
`ExtractTableSchemaActivity => TableMetadataToXMLCharArraysListActivity
=> DeliverToRequestStatusActivity`

OGSA-DAI 3.1 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. SQLAccess can use this existing OGSA-DAI 3.1 JDBCConnectionProvider wrapper interface since it too just provides access to a

relational database. However since we have the additional configuration property of the sql-exception-handler we introduce the SQLAccessProvider interface that extends the JDBCConnectionProvider and the class uk.org.ogsadai.wsdai.dair.resource.SimpleSQLAccessProvider that provides an implementation of SQLAccessProvider for wrapping SQLAccess resources.

The following diagram shows the relationship between WS-DAIR, WS-DAI and OGSA-DAI 3.1 resource, configuration and provider classes, for SQLAccess resources.



3.8.3 SQLResponse

SQLResponses are represented by the uk.org.ogsadai.wsdai.dair.resource.SQLResponse class, a sub-class of CoreResource. They provide access to the results of an SQL statement execution.

SQLResponse uses an uk.org.ogsadai.wsdai.dair.resource.SQLResponseState object to manage its configuration. SQLResponseState extends CoreResourceState.

SQLResponseState manages the following configuration values:

- Resource manager ID – ID of a resource manager in the OGSA-DAI context which stores the SQLResponseData associated with this SQLResponse.
- Response URL – URL which identifies this resource’s SQLResponseData in the resource manager. SQLResponseManager and SQLReponseData are described in the next section.

SQLResponseState manages the provision of the following resource properties:

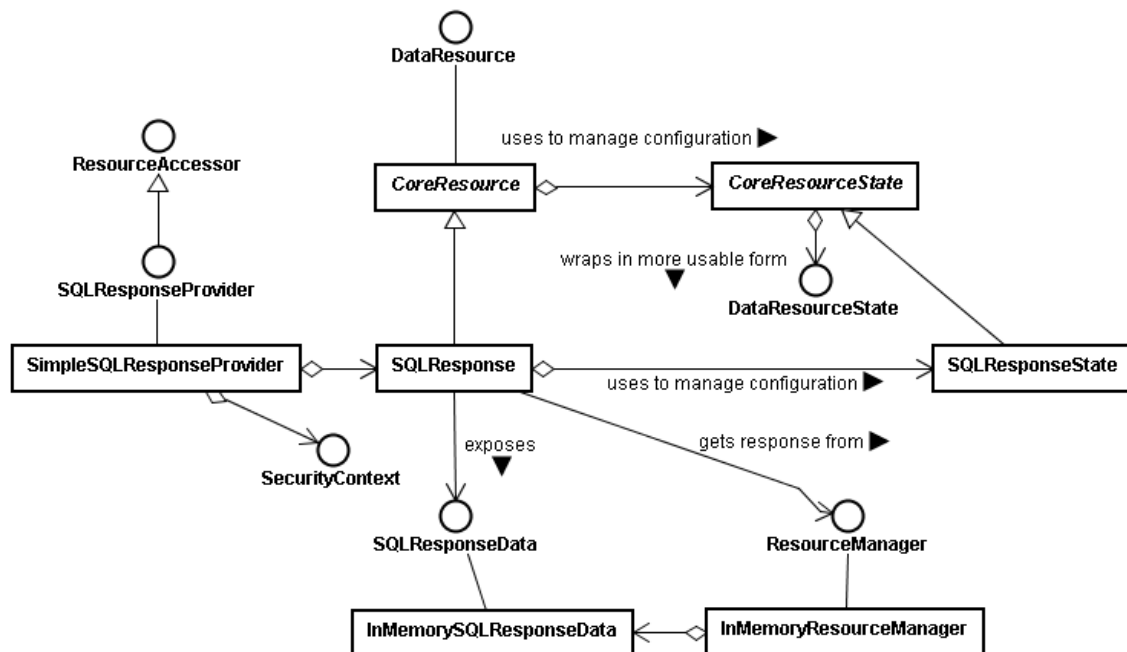
- ChildSensitiveToParent – hard-coded value of Insensitive.
- ParentSensitiveToChild – hard-coded value of Insensitive.

SQLResponse itself manages the provision of the following resource properties:

- DatasetMap – hard-coded.
- LanguageMap – hard-coded value of null.
- ConfigurationMap – hard-coded value.
- NumberOfRowsets – determined via interrogation of the associated SQLResponseData.
- NumberOfUpdateCounts - determined via interrogation of the associated SQLResponseData.
- NumberOfReturnValues - determined via interrogation of the associated SQLResponseData.
- NumberOfOutputParameters - determined via interrogation of the associated SQLResponseData.
- NumberOfCommunicationAreas - determined via interrogation of the associated SQLResponseData.
- SQLResponseItem - determined via interrogation of the associated SQLResponseData.

The interface `uk.org.ogsadai.wsdai.dair.SQLResponseProvider` and implementing class `uk.org.ogsadai.wsdai.dair.SimpleSQLResponseProvider` provide a wrapper for the use of SQLResponses with SQLResponse-related activities (e.g. `ReadFromSQLResponseActivity`). This interface provides access to a `SQLResponseData` object representing the data associated with the SQLResponse.

The following diagram shows the relationship between WS-DAIR, WS-DAI and OGSA-DAI 3.1 resource, configuration and provider classes, for SQLResponses.



3.8.4 SQLResponseData and SQLResourceManagers

A `SQLResponse` exposes a set of items derived from an execution of an SQL statement. This is represented via a `uk.org.ogsadai.wsdai.daie.SQLResponseData` interface. A `SQLResponseData` can be viewed as analogous to a JDBC Connection or

XMLDB Collection object. OGSA-DAI WS-DAIR provide an implementation of this interface - `uk.org.ogsadai.wsdai.dair.resource.InMemorySQLResponseData` - which just stores the results in-memory.

A `SQLResourceManager` – represented by the interface `uk.org.ogsadai.wsdai.dair.resource.SQLResourceManager` – provides access to `SQLResponseData`. A `SQLResourceManager` can be viewed as analogous to a JDBC or XMLDB database driver. OGSA-DAI WS-DAIR provides an implementation of this interface - `uk.org.ogsadai.wsdai.dair.resource.InMemorySQLResourceManager` - which creates and manages an in-memory indexed collection of `InMemorySQLResponseData` objects.

A `SQLResourceManager` is placed in the OGSA-DAI context when OGSA-DAI WS-DAIR is deployed.

3.8.5 SQLRowset

`SQLRowsets` are represented by the `uk.org.ogsadai.wsdai.dair.resource.SQLRowset` class, a sub-class of `CoreResource`. They provide access to the tuples of the resultsets being held by an `SQLResponse` resource.

`SQLRowset` uses an `uk.org.ogsadai.wsdai.dair.resource.SQLRowsetState` object to manage its configuration. `SQLRowsetState` extends `SQLResponseState` so it inherits the Response URL and the Resource Manager ID configuration properties.

`SQLRowsetState` manages the following configuration values:

- Rowset URL – URL which identifies this resource's rowset in the `SQLResponseData` object. In more detail, this means that a `SQLResponseData` holds a `Rowset` object with a specific URL. Once a new `SQLRowset` resource is created this rowset URL attaches the resource to the in-memory object that holds the tuples of the resultset.

`SQLRowsetState` manages the provision of the following resource properties:

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

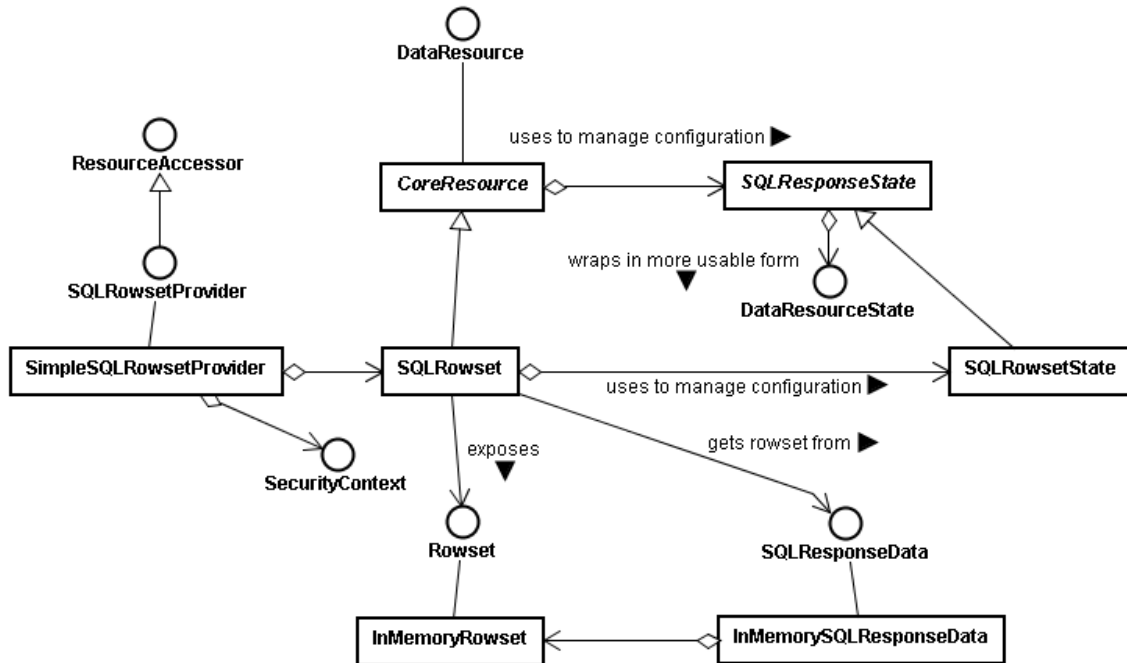
The sensitivity derives from the fact that the Rowsets of `SQLRowset` resources are effectively being held by the `SQLResponseData` of `SQLResponse` resources. As a consequence, if a `SQLResponse` resource is destroyed, all its attached `SQLRowset` resources will also be destroyed.

`SQLRowset` itself manages the provision of the following resource properties:

- `DatasetMap` – hard-coded.
- `LanguageMap` – hard-coded value of null.
- `ConfigurationMap` – hard-coded value of null.
- `NumberOfRows` – determined via interrogation of the associated Rowset.
- `RowsetSchema` - determined via interrogation of the associated Rowset.
- `AccessMode` - determined by the client request.

The interface `uk.org.ogsadai.wsdai.dair.SQLRowsetProvider` and implementing class `uk.org.ogsadai.wsdai.dair.SimpleSQLRowsetProvider` provide a wrapper for the use of `SQLRowsets` with `SQLRowset`-related activities (e.g. `GetTuplesActivity`). This interface provides access to a `Rowset` object representing the tuples associated with the `SQLRowset`.

The following diagram shows the relationship between WS-DAIR, WS-DAI and OGSA-DAI 3.1 resource, configuration and provider classes, for `SQLRowsets`.



3.8.6 Rowset

A `SQLRowset` exposes a set of tuples derived from a resultset being held by a `SQLResponse`. This is represented via an `uk.org.ogsadai.wsdai.dair.Rowset` interface. A `SQLResponseData` object can be holding one or more `Rowset` objects. OGSA-DAI WS-DAIR provides an implementation of this interface - `uk.org.ogsadai.wsdai.dair.resource.InMemoryRowset` - which just stores the results in-memory. Once a new `SQLRowset` is created the URL of the `Rowset` object is attached to this resource.

4. Naming and data formats

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

4.1 *WebRowSet and CSV*

The results of `SQLExecute`, `GetSQLResponseItem`, `GetSQLRowset`, `GetTuples` operations return data in the form of either webrowset format ([JSR114] J. Bruce, JSR-000114 JDBC RowSet Implementations, Final Release, 07 April 2004) or in CSV (comma-separated values) format.

The dataset format URI of webrowset is `http://java.sun.com/xml/ns/jdbc`.

The dataset format URI of one of csvCSV is `http://ogsadai.org.uk/data/csv`.

The XML representation of a csv representation looks like:

```
<csv>
1,Ally Antonioletti,101 Antonioletti Road, San Jose,087192027
</csv>
```

Whereas the webrowset equivalent would look like:

```
<webRowSet xmlns="http://java.sun.com/xml/ns/jdbc"
xmlns:ns4="http://java.sun.com/xml/ns/jdbc"
xsi:schemaLocation="http://java.sun.com/xml/ns/jdbc
http://java.sun.com/xml/ns/jdbc/webrowset.xsd">
  <properties>
    <command/>
    <concurrency/>
    <datasource/>
    <escape-processing>true</escape-processing>
    <fetch-direction>1000</fetch-direction>
    <fetch-size>0</fetch-size>
    <isolation-level>0</isolation-level>
    <key-columns/>
    <map/>
    <max-field-size>0</max-field-size>
    <max-rows>0</max-rows>
    <query-timeout>0</query-timeout>
    <read-only>true</read-only>
    <rowset-type>1003</rowset-type>
    <show-deleted>>false</show-deleted>
    <table-name/>
    <url/>
    <sync-provider>
      <sync-provider-name/>
      <sync-provider-vendor/>
      <sync-provider-version/>
      <sync-provider-grade/>
      <data-source-lock/>
    </sync-provider>
  </properties>
  <metadata>
    <column-count>4</column-count>
    <column-definition>
```

```

    <column-index>1</column-index>
    <auto-increment/>
    <case-sensitive/>
    <currency/>
    <nullable>1</nullable>
    <signed/>
    <searchable/>
    <column-display-size>11</column-display-size>
    <column-label>id</column-label>
    <column-name>id</column-name>
    <schema-name/>
    <column-precision>11</column-precision>
    <column-scale>0</column-scale>
    <table-name>littleblackbook</table-name>
    <catalog-name/>
    <column-type>4</column-type>
    <column-type-name>INTEGER</column-type-name>
  </column-definition>
  <column-definition>
    <column-index>2</column-index>
    <auto-increment/>
    <case-sensitive/>
    <currency/>
    <nullable>1</nullable>
    <signed/>
    <searchable/>
    <column-display-size>64</column-display-size>
    <column-label>name</column-label>
    <column-name>name</column-name>
    <schema-name/>
    <column-precision>64</column-precision>
    <column-scale>0</column-scale>
    <table-name>littleblackbook</table-name>
    <catalog-name/>
    <column-type>12</column-type>
    <column-type-name>VARCHAR</column-type-name>
  </column-definition>
  <column-definition>
    <column-index>3</column-index>
    <auto-increment/>
    <case-sensitive/>
    <currency/>
    <nullable>1</nullable>
    <signed/>
    <searchable/>
    <column-display-size>128</column-display-size>
    <column-label>address</column-label>
    <column-name>address</column-name>
    <schema-name/>
    <column-precision>128</column-precision>
    <column-scale>0</column-scale>
    <table-name>littleblackbook</table-name>
    <catalog-name/>

```

```

        <column-type>12</column-type>
        <column-type-name>VARCHAR</column-type-name>
    </column-definition>
    <column-definition>
        <column-index>4</column-index>
        <auto-increment/>
        <case-sensitive/>
        <currency/>
        <nullable>1</nullable>
        <signed/>
        <searchable/>
        <column-display-size>20</column-display-size>
        <column-label>phone</column-label>
        <column-name>phone</column-name>
        <schema-name/>
        <column-precision>20</column-precision>
        <column-scale>0</column-scale>
        <table-name>littleblackbook</table-name>
        <catalog-name/>
        <column-type>12</column-type>
        <column-type-name>VARCHAR</column-type-name>
    </column-definition>
</metadata>
<data>
    <currentRow>
        <columnValue>1</columnValue>
        <columnValue>Ally Antonioletti</columnValue>
        <columnValue>101 Antonioletti Road, San Jose</columnValue>
        <columnValue>087192027</columnValue>
    </currentRow>
</data>
</webRowSet>

```

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.SomeDAIRResource`.

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

In OGSA-DAI WS-DAIR 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 *SQLResponseData*

Each `SQLResponse` uses a resource manager to manage its set of items. The resource manager to use is expected to reside in the OGSA-DAI context and the `SQLResponse` configuration specifies the name of the resource manager to use. The `SQLResponseData` held by the resource manager, is identified via a response URL, which is also part of the `SQLResponse` configuration. Response URLs are of form:

`ogsadai:response://RESPONSE-ID`

For example:

`ogsadai:response://118cc679e15`

4.4 *Rowset*

Each `SQLRowset` is related to a `Rowset` object that holds the set of tuples. This `Rowset` object which belongs to the `SQLResponseData` is identified via a rowset URL, which is also part of the `SQLRowset` configuration. Rowset URLs are of form:

`ogsadai:rowset://ROWSET-ID`

For example:

`ogsadai:rowset://118cc679e15`

5. Security

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

5.1 *Logins, SQLAccesses*

Relational data resources in OGSA-DAI WS-DAIR use OGSA-DAI 3.1 login providers to map presentation layer security information (currently just an empty OGSA-DAI 3.1 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.

5.2 *Logins and resource properties*

Certain resources need to connect to a relational database to determine resource property values e.g. `SQLAccess` and its `SchemaDescription` property. However OGSA-DAI 3.1 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-DAIR Specification Compliance

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

6.1 Notation

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

6.2 *portTypes common to all services*

The following portTypes are supported by all OGSA-DAI WS-DAIR 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-DAIR.
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/AccessServiceAccessPT
  </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:MySQLCollection
    </ns28:DataResourceAbstractName>
  </ReferenceParameters>
  <Metadata xmlns:ns5="http://www.w3.org/2005/08/addressing"
    xsi:nil="true" xsi:type="ns5:MetadataType"/>
</ns1:DataResourceAddress>

```

6.3 SQLAccess and SQLAccessService

6.3.1 CoreDataDescription

DataResourceAbstractName	Always has the resource name.
DataResourceManagement	ExternallyManaged.
ConcurrentAccess	Always has value True
DatasetMap	Always returns mappings: SQLExecute => http://java.sun.com/xml/ns/jdbc SQLExecute => http://ogsadai.org.uk/data/csv
LanguageMap	Always returns mappings: SQLExecute => http://www.sqlquery.org/sql-92
ConfigurationMap	Always returns a mapping of: ConfigurationDocument – SQLResponsePT – SQLExecuteFactory
Readable	True OR false depending upon the service deployer.
Writable	True OR false depending upon the service deployer.
TransactionInitiation	Always has value NotSupported.
TransactionIsolation	Always has value NotSupported.
ChildSensitiveToParent	Always has value Insensitive.
ParentSensitiveToChild	Always has value Insensitive.
ParentDataResource	Ommited as it is an ExternallyManaged resource.
DataResourceDescription	Depends solely on the service deployer.

6.3.2 SQLAccessDescription

SchemaDescription	Constructed by querying of the associated database.
-------------------	---

6.3.3 SQLAccess

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

SQLAccess::SQLExecute	
DataResourceAbstractName input	
SQLExpression (SQLExpression,SQLParameters*) input	
DatasetFormatURI? input	
SQLDataset(DatasetFormatURI, DatasetData, SQLUpdateCount*, SQLOutputParamter*, SQLCommunicationsArea*) output	Each resultset document is placed in the first MessageElement of each DatasetData MessageElement array. i.e.: DatasetDataType.get_any()[0].
InvalidResourceNameFault	Thrown if resource is unknown.
DataResourceUnavailableFault	Unused
InvalidExpressionFault	Thrown if something is wrong with the SQL expression provided.
InvalidSQLExpressionParametersFault	Thrown if something is wrong with the SQL parameters provided.
InvalidLanguageFault	Unused
NotAuthorizedFault	Thrown if resource isReadable property is false.
ServiceBusyFault	Unused

6.3.4 SQLAccessFactory

SQLAccessFactory::SQLExecuteFactory	
DataResourceAbstractName input	
SQLExpression (SQLExpression,SQLParameters*) input	
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 SQLResponse 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.
DataResourceAddressList output	Returns a list with one data resource address with the DataResourceAbstractName and a port that is one of those supported by an SQLAccess service available on the server.
InvalidResourceNameFault	Thrown if resource is unknown.
DataResourceUnavailableFault	Unused.
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	If the client provided a configuration document that is of a type unsupported by the resource (inconsistent with that specified in the configuration map).
InvalidExpressionFault	Thrown if something is wrong with the SQL expression provided.
InvalidSQLExpressionParametersFault	Thrown if something is wrong with the SQL parameters provided.
InvalidLanguageFault	Unused
InvalidDatasetFormatFault	Thrown if DatasetFormatURI input is provided but is unsupported.

6.4 SQLResponse and SQLResponseService

6.4.1 CoreDataDescription

DataResourceAbstractName	Always has the resource name.
DataResourceManagement	Always has value ServiceManaged.
ConcurrentAccess	Always has value True
DatasetMap	Always returns mappings: SQLResponseItem => http://java.sun.com/xml/ns/jdbc SQLResponseItem => http://ogsadai.org.uk/data/csv SQLRowset => http://java.sun.com/xml/ns/jdbc SQLRowset => http://ogsadai.org.uk/data/csv
LanguageMap	Always has value null.

ConfigurationMap	Always returns a mapping of: SQLRowsetConfigurationDocument – SQLRowsetPT - SQLRowsetFactory
Readable	True OR false depending upon ConfigurationDocument provided to parent.
Writable	Always 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-DAIR configuration file or is provided in a ConfigurationDocument passed to indirect access operations.

6.4.2 SQLResponseDescription

SQLResponseItem	Constructed by querying of the associated SQLResponseData.
NumberOfSQLRowsets	Constructed by querying of the associated SQLResponseData.
NumberOfSQLUpdateCounts	Constructed by querying of the associated SQLResponseData.
NumberOfSQLReturnValues	Constructed by querying of the associated SQLResponseData.
NumberOfSQLOutputParameters	Constructed by querying of the associated SQLResponseData.
NumberOfSQLCommunicationAreas	Constructed by querying of the associated SQLResponseData.

6.4.3 SQLResponse

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

SQLResponse::GetSQLResponseItem	
DataResourceAbstractName input	
DatasetFormatURI? input	
Position input	Assumes items are numbered from 0 onwards.

Count? input	If not provided one item will be returned (the first).
SQLDataset(DatasetFormatURI, DatasetData, SQLUpdateCount*, SQLOutputParamter*, SQLCommunicationsArea*) output	Each resultset document is placed in the first MessageElement of each DatasetData MessageElement array. i.e.: DatasetDataType.get_any()[0].
InvalidResourceNameFault	Thrown if resource is unknown.
DataResourceUnavailableFault	Unused
InvalidDatasetFormatFault	Thrown if DatasetFormatURI input is provided but is unsupported.
NotAuthorizedFault	Thrown if resource isReadable property is false.
ServiceBusyFault	Unused
InvalidStartPositionFault	Thrown if StartPosition < 0 or > the sum of the values of the NumberOfSQLRowset, NumberOfSQLUpdateCounts, NumberOfSQLReturnValue, NumberOfSQLOutputParameter, NumberOfSQLCommunicationAreas properties.
InvalidCountFault	Thrown if ItemCount < 0 or (StartPosition + ItemCount) > the sum of the values of the NumberOfSQLRowset, NumberOfSQLUpdateCounts, NumberOfSQLReturnValue, NumberOfSQLOutputParameter, NumberOfSQLCommunicationAreas properties.

SQLResponse::GetSQLRowset	
DataResourceAbstractName input	
DatasetFormatURI? input	
Position input	Assumes items are numbered from 0 onwards.
Count? input	If not provided one item will be returned (the first).
Dataset(DatasetFormatURI, DatasetData) output	Each resultset document is placed in the first MessageElement of each DatasetData MessageElement array. i.e.: DatasetDataType.get_any()[0].
InvalidResourceNameFault	Thrown if resource is unknown.
DataResourceUnavailableFault	Unused
InvalidDatasetFormatFault	Thrown if DatasetFormatURI input is provided but is unsupported.
NotAuthorizedFault	Thrown if resource isReadable property is false.
ServiceBusyFault	Unused
InvalidStartPositionFault	Thrown if StartPosition < 0 or > the value of NumberOfSQLRowset property.
InvalidCountFault	Thrown if ItemCount < 0 or (StartPosition + ItemCount) > the value of the NumberOfSQLRowset property.

SQLResponse::GetSQLUpdateCount	
DataResourceAbstractName input	
Position input	Assumes items are numbered from 0 onwards.
Count? input	If not provided one item will be returned (the first).
SQLUpdateCount? output	
InvalidResourceNameFault	Thrown if resource is unknown.
DataResourceUnavailableFault	Unused
NotAuthorizedFault	Thrown if resource isReadable property is false.
ServiceBusyFault	Unused
InvalidStartPositionFault	Thrown if StartPosition < 0 or > NumberOfSQLUpdateCounts.
InvalidCountFault	Thrown if ItemCount < 0 or (StartPosition + ItemCount) > NumberOfSQLUpdateCounts.

SQLResponse::GetSQLReturnValue	
DataResourceAbstractName input	
SQLReturnValue? output	
InvalidResourceNameFault	Thrown if resource is unknown.
DataResourceUnavailableFault	Unused
NotAuthorizedFault	Thrown if resource isReadable property is false.
ServiceBusyFault	Unused

SQLResponse::GetSQLOutputParameter	
DataResourceAbstractName input	
Position input	Assumes items are numbered from 0 onwards.
Count? input	If not provided one item will be returned (the first).
SQLOutputParameter? output	
InvalidResourceNameFault	Thrown if resource is unknown.
DataResourceUnavailableFault	Unused
NotAuthorizedFault	Thrown if resource isReadable property is false.
ServiceBusyFault	Unused
InvalidStartPositionFault	Thrown if StartPosition < 0 or > NumberOfSQLOutputParameters.
InvalidCountFault	Thrown if ItemCount < 0 or (StartPosition + ItemCount) > NumberOfSQLOutputParameters.

SQLResponse::GetSQLCommunicationArea	
DataResourceAbstractName input	
Position input	Assumes items are numbered from 0 onwards.

Count? input	If not provided one item will be returned (the first).
SQLCommunicationArea? output	
InvalidResourceNameFault	Thrown if resource is unknown.
DataResourceUnavailableFault	Unused
NotAuthorizedFault	Thrown if resource isReadable property is false.
ServiceBusyFault	Unused
InvalidStartPositionFault	Thrown if StartPosition < 0 or > NumberOfSQLCommunicationArea.
InvalidCountFault	Thrown if ItemCount < 0 or (StartPosition + ItemCount) > NumberOfSQLCommunicationArea.

6.4.4 SQLResponseFactory

SQLResponseFactory::SQLRowsetFactory	
DataResourceAbstractName input	
Position input	Assumes items are numbered from 0 onwards.
Count? input	If not provided one item will be returned (the first).
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 SQLRowset resources will inherit the isReadable and Description property values only.</p> <p>If the ConfigurationDocument provided is a SQLRowsetConfigurationDocument the client can provide the access mode to navigate through the created SQLRowset resources.</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>
DataResourceAddressList output	Returns a list with data resource addresses with each having the DataResourceAbstractName and a port that is one of those supported by an SQLResponse service available on the server.
InvalidResourceNameFault	Thrown if resource is unknown.
DataResourceUnavailableFault	Unused.

NotAuthorizedFault	Thrown if resource isReadable property is false.
ServiceBusyFault	Unused.
InvalidStartPositionFault	Thrown if StartPosition < 0 or > NumberOfSQLRowsets.
InvalidCountFault	Thrown if ItemCount < 0 or (StartPosition + ItemCount) > NumberOfSQLRowsets.

6.5 SQLRowset and SQLRowsetService

6.5.1 CoreDataDescription

DataResourceAbstractName	Always has the resource name.
DataResourceManagement	Always has value ServiceManaged.
ConcurrentAccess	Always has value True
DatasetMap	Always returns mappings: GetTuples => http://java.sun.com/xml/ns/jdbc GetTuples => http://ogsadai.org.uk/data/csv
LanguageMap	Always has value null.
ConfigurationMap	Always has value null.
Readable	True OR false depending upon ConfigurationDocument provided to parent.
Writable	Always has value false.
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-DAIR configuration file or is provided in a ConfigurationDocument passed to indirect access operations.

6.5.2 SQLRowsetDescription

RowSchema	Constructed by querying of the associated rowset.
NoOfRows	Constructed by querying of the associated rowset.
AccessMode	Provided by the client through the SQLRowsetConfigurationDocument.

6.5.3 SQLRowsetAccess

SQLRowsetAccess::GetSQLRowsetPropertyDocument	
DataResourceAbstractName input	
SQLRowsetPropertyDocument output	

InvalidResourceNameFault	Thrown if resource is unknown.
DataResourceUnavailableFault	Unused
NotAuthorizedFault	Unused
ServiceBusyFault	Unused

SQLRowsetAccess::GetTuples	
DataResourceAbstractName input	
DatasetFormatURI? input	
StartPosition input	Assumes items are numbered from 0 onwards.
ItemCount? input	If not provided one item will be returned (the first).
Dataset(DatasetFormatURI, DatasetData) output	Each resultset document is placed in the first MessageElement of each DatasetData MessageElement array. i.e.: DatasetDataType.get_any()[0].
InvalidResourceNameFault	Thrown if resource is unknown.
DataResourceUnavailableFault	Unused
InvalidDatasetFormatFault	Thrown if DatasetFormatURI input is provided but is not supported.
NotAuthorizedFault	Thrown if resource isReadable property is false.
ServiceBusyFault	Unused
InvalidStartPositionFault	Thrown if StartPosition < 0 or > NoOfRows property.
InvalidCountFault	Thrown if ItemCount < 0 or (StartPosition + ItemCount) > NoOfRows.

7. Options for future work

7.1 Error reporting

When the OGSA-DAI 3.1 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. This in WSDAIR implementation applies when using the CharArraysToDOMActivity.

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.

7.2 Resource state

Clean configuration maps in SQLAccess, SQLResponse and SQLRowset classes. Use static defaults in these classes or in DAIRResourcePropertyUtils. The resource state of SQLRowset depends on the presence and existence of the parent SQLResponse resource. This introduces dependency between them but it was the preferred and most convenient approach during the implementation taken that the specification doesn't preclude this!

Allow N data resource description "documents" not just zero or one. Requires using resource-specific persistence and not just the OGSA-DAI 3.1 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.

7.3 Resource properties

Change SimpleCoreResource/SQLAccess/SQLResponse/SQLRowset so that they use the OGSA-DAI 3.1 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 3.1 example is in uk/org/ogsadai/resource/request/RequestStatusPropertyCallback.java

Change SQLAccess/SQLResponse/SQLRowset get/setResourcePropertyValue to use a default DN or username/password configuration property to provide a slight improvement on using a null security context,

Change SQLAccess/SQLResponse/SQLRowset get/setResourcePropertyValue to take a SecurityContext as an argument. This removes need for the above but requires an OGSA-DAI 3.1 API change.

Add to OnDemandResourcePropertyCallbackException a (ResourceID, ResourcePropertyName, Throwable) constructor and use this instead of initCause(). This requires an OGSA-DAI 3.1 API change.

Implement SimpleDAISResourcePropertyValue getAsDOM(). This requires implementing WS-DAI resource property value <=> 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 3.1 deployment (i.e. outwith the WS-DAI presentation layer).

7.4 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.

7.5 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.

7.6 Security

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

7.7 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.

7.8 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.

7.9 Clients

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

7.10 Additional SQL Exception handlers

As discussed earlier on `SQLAccess` resources have a `SQLExceptionHandler` property which indicates the ID of the exception handler in the OGSA-DAI context. This handler parses any `SQLExceptions` thrown and decides if indeed an `SQLException` should be thrown or the exception will be part of the response wrapped as `SQLCommunicationArea`. Since the messages, error codes are vendor specific we opted for providing a `MySQLExceptionHandler`. Providing additional `SQLExceptionHandler` implementations for other vendor databases that are supported by OGSA-DAI 3.1 would be a nice extension.

8. OGSA-DAI 3.1 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.

9. OGSA-DAI 3.1 ingestion

The following changes are required to allow the use of OGSA-DAI WS-DAIR resources and activities within a standard OGSA-DAI 3.1 deployment.

- Change `CreateSQLResponseResourceActivity` to take an optional `ResourceID` input parameter.
- Change `CreateSQLRowsetResourcesActivity` to take an optional list of `ResourceIDs` input parameter.
- Changing `SQLStatement` needs some thorough consideration to be ported to the OGSA-DAI 3.1 activity model.

- Change SQLAccess, SQLResponse, SQLRowset to not use NullSecurityContext but to take it in as an argument to get/setResourcePropertyValue – see section 7.3 and section 8.