GFD-E.160
DAIS Working Group

Steven Lynden, AIST
Mario Antonioletti (Corresponding Author), EPCC
Mike Jackson, EPCC
Sunil Ahn, KISTI
November 30, 2009

WS-DAI and WS-DAIR Implementations - Experimental Document

Status of This Document

This document presents the results of the interoperability testing process of two independent implementations of the WS-DAI (GFD.74) and WS-DAIR (GFD.76) specifications. Distribution is unlimited.

Abstract

The Data Access and Integration Services (DAIS) Working Group have defined three proposed recommendations within the Open Grid Forum (OGF). The OGF process and requirements document [GFD.152] states that two independent interoperable implementations are required for a proposed recommendation to become a full OGF recommendation. This OGF experimental document reports on interoperability testing of two implementations of WS-DAIR (GFD.76) – one from the OGSA-DAI[1] group at The University of Edinburgh and the other based on AMGA[2] from KISTI. In addition, as the WS-DAIR proposed recommendation is an extension to WS-DAI (GFD.74), the testing process has encompassed both the WS-DAIR and WS-DAI proposed recommendations. The tests documented in this OGF experimental document establish that it is indeed possible to obtain client-based interoperability for these two implementations. However, as a result of this interoperation process a number of changes are recommended for the WS-DAI and WS-DAIR documents before they gain full recommendation status.

It is important to note that this document establishes a set of interoperability requirements between two or more implementations of the WS-DAI specifications, taking into account the criteria established in GFD.77 [DAIS-Interop]. This document is not intended to establish a validation process to test the compliance of any particular implementation to any of the (proposed) DAIS recommendations.

[1] http://www.ogsadai.org.uk.
[2] http://amga.web.cern.ch.

dais-wg@ogf.org

Contents

1 Introduction

Three proposed recommendations [WS-DAI, WS-DAIR, WS-DAIX] have been defined by the OGF Database Access and Integration Services (DAIS) Working Group. This OGF experimental document reports on the interoperability testing results arising from two implementations of WS-DAIR (GFD.76), which also implement WS-DAI (GFD.74). Note that by interoperability here we mean interoperability of the clients accessing the DAIS defined services, i.e. not inter-service interoperability. To be more specific, the actual SOAP messages sent by a client, in this case a third party application, must produce the same behaviour and results for each of the DAIS-compliant services in order for those services to be defined as interoperable.

The implementations that participated in this exercise were:

- OGSA-DAI WS-DAIR implementation.
- AMGA WS-DAIR implementation.

The testing approach follows that defined in [DAIS-Interop] (GFD.77) with the additional two constraints:

- PortTypes MUST be bound to SOAP 1.1.
- The binding MUST use the “document/literal” style.

These constraints were first stipulated by the ByteIO working group for their interoperability tests as implementations using different versions of SOAP, or different bindings, are not interoperable. The rationale for taking this approach, as well as the precedent set by ByteIO, included:

- WSRF recommend this type of binding in their application notes[3].
- The document/literal style offers a certain degree of decoupling of the operations from its associated messages (the elements have to be there but these do not have to comply with a specific XML Schema)[4].

This document reports on the application of the process established in [DAIS-Interop] to validate the WS-DAI and WS-DAIR specifications. The WS-DAI specification is validated as part of this process as WS-DAIR extends this core specification.

2 Process and limitations

2.1 Overview

The DAIS-WG specifications define behaviours that may vary as a reflection of the different properties of the underlying data resources to which the services provide access (e.g. support for different query languages, support for concurrent access, etc.). As discussed in [DAIS-Interop], rather than write tests for each specific database management system, the aim of the testing process is to validate the specifications themselves and show that interoperability is achieved regardless of the database management system being used. It is not intended to test the compliance of any particular implementation to the WS-DAI or the WS-DAIR specifications. As such, the testing of every feature and of every operation is not mandated as this process is meant to just test interoperation of two implementations, rather than an implementation’s conformance to the specification, with a finite number of test cases.

[DAIS-Interop] states specifically that “the testing process *aspires* to test every mandatory feature in multiple implementations and every optional feature in at least one implementation”. A list of the mandatory operations and properties as defined by WS-DAI and WS-DAIR was presented in [DAIS-

[3] See the (non normative) "Application notes" document: http://docs.oasis-open.org/wsrf/wsrf-application_notes-1.2-cd-02.pdf which recommends document/literal (section 8.1). Thanks to Bernd Schuller for pointing this out.
[4] Thanks to Miguel Esteban Gutierrez for pointing this out.

Interop]. However, this document did not consider specific features of relational database management systems such as: support for query languages, user defined functions and stored procedures, which need to be taken into account when designing a test suite. The various features of WS-DAI and WS-DAIR that are not tested or depend on specific features of the relational database management system used are listed in the remainder of this section, with justification given where specific features cannot be tested.

2.2 WS-DAI untested features

2.2.1 Properties

The following optional parameter is not tested:

- `PreferredTargetService` of factory operations. This parameter provides a hint that can be ignored by the service, therefore testing it is impractical because there is no behaviour defined that is dependent on the value of the parameter.

The following fault is not tested:

- `DataResourceUnavailableFault` is not tested because its purpose is to indicate the unavailability of a data resource due to unanticipated reasons. The circumstances under which this fault will be generated are implementation specific.

The following properties:

- Implementations may or may not support concurrent access; hence the `ConcurrentAccess` property value is not tested. An optional test is included for implementations that have a specific limit on the number of concurrent requests that can be handled.

The following properties, which may appear in a `ConfigurationDocument` specified as a parameter with factory operations:

- `Readable` and `Writeable` properties – these are implementation-specific or related to the underlying data resource a service is exposing.
- `TransactionInitiation`, `TransactionIsolation`, `ChildSensitiveToParent` and `ParentSensitiveToChild` all define behaviours that may not be supported given the database management system or the implementation that is used and are therefore considered beyond the scope of the testing process.

The `CoreResourceList` portType is optional and is tested only by implementations that support it.

2.2.2 Testing ServiceBusyFault (optional)

This may be tested by implementations for which there is a reliable trigger condition that results in the generation of a `ServiceBusyFault`. It may be difficult to trigger a `ServiceBusyFault` in some implementations as its generation may be dependent on a number of variables including the time taken for requests to reach the service, the time taken to process requests, and the factors that limit the number of concurrent requests that can be processed.

2.2.3 Testing CoreDataAccess::GenericQuery (optional)

`GenericQuery` is an operation defined in the WS-DAI core specification that supports a general means of passing query documents to a data resource without mandating the use of a particular language. In this document, testing the `GenericQuery` operation assumes that this operation is implemented with support for SQL queries. As this is not a requirement of the specifications, which allows any query language to be implemented, the test is optional or may possibly be modified for implementations that support a different implementation of `GenericQuery`.

2.3 WS-DAIR untested features

According to the specifications, where factory operations take a configuration document as an OPTIONAL parameter the values specified in the configuration document are hints that may be complied with but can be ignored by the implementation. Given that the hints may be ignored, there is no correct behaviour with which different implementations must comply, and therefore tests involving the specification of different configuration document values are not included.

2.3.1 Features tested by implementations supporting modifications

It is possible that some implementations do not support modifications to the data contained in the database, i.e. SQL insert, update and delete statements. If modifications are not supported, the `SQLResponse::GetSQLUpdateCount` operation and `SQLUpdateCount` property cannot be tested.

2.3.2 Features tested by implementations supporting UDFs and stored procedures

If an implementation does not support user defined functions (UDFs), the `SQLResponse::GetSQLReturnValue` operation and `SQLReturnValue` property cannot be tested. If an implementation does not support stored procedures the `SQLResponse::GetSQLOutputParameter` operation and `SQLOutputParameter` property cannot be tested.

2.3.3 WS-DAIR features tested only by implementations supporting SQL Communications Area

Some implementations may not return errors using SQL Communications Areas, in which case the `SQLResponse::GetSQLCommunicationsArea` operation and `SQLCommunicationsArea` property cannot be tested.

2.4 Features shared by multiple operations

Where faults or parameters are shared by operations, the assumption is made that it is only necessary to test such features once, for example, for every WS-DAI operation, an `InvalidResourceNameFault` should be produced if an unknown resource name is specified using the `DataResourceAbstractName` parameter. This fault is assumed to have been tested if it is correctly produced by one test case applied to a single operation; it is not considered necessary to replicate this test for every operation.

2.5 Languages, schema and dataset formats

Languages, dataset and schema formats are implementation-specific and the tests do not define which formats are used, although it is suggested by the WS-DAIR specification that `WebRowSet` is used as the dataset format for SQL query results. Language and dataset formats may vary however, and where these features are relevant, the tests must be customised to the specific implementation being tested in order to validate that the datasets/schemas returned by the implementation are correct. Appendix 1 suggests the SQL queries that might be used or modified depending on the language used.

3 Implementations

This section briefly outlines the implementations that participated in this exercise.

OGSA-DAI WS-DAIR implementation[5]:

- Uses Apache Axis 1.4 and Java 1.4.
- Can potentially use any JDBC-enabled relational database as the underlying database management system, though it has only been tested with MySQL.
- Supports stored procedures if the underlying database management system does.
- Supported datasets: `WebRowSet`, comma-separated values.

[5] A version of this is available from https://sourceforge.net/projects/ogsa-dai/files. The 1.0 release will be updated to comply with the changes suggested in this document, see Section 6.1.1, once this document has gone through the OGF process.

- Security features: none.

AMGA WS-DAIR implementation[6]:

- Uses gSoap & C++.
- Supports the following underlying database management systems: PostgreSQL, MySQL, SQLite, and Oracle.
- Supported languages: SQL-92 entry level "direct data statement" and the AMGA metadata language.
  - Does not support stored procedures.
- Supported dataset: `WebRowSet`.
- Security Features: SSL, GSI, VOMS, permission, and ACL.

4 Testing Scenario

The test scenario is based on a database of contact details organized into a relational table. The scenario is initialized by constructing a relational database containing a set of entries.

4.1 Database Schema

A single table is created with the following schema:

```
CREATE TABLE IF NOT EXISTS littleblackbook
(
       id     INTEGER,
       name   VARCHAR(64),
       address VARCHAR(128),
       phone  VARCHAR(20)
)
```

This table is populated with the following 10 rows:

```
INSERT INTO littleblackbook VALUES (1,'Ally Antonioletti','101 Antonioletti Road, San
Jose','087192027');
INSERT INTO littleblackbook VALUES (2,'Amy Atkinson','70 Atkinson Crescent,
Southampton','0105931111');
INSERT INTO littleblackbook VALUES (3,'Bartosz Chue Hong','30 Chue Hong Gardens,
Winchester','04476816');
INSERT INTO littleblackbook VALUES (4,'Craig Dobrzelecki','72 Dobrzelecki Place,
Edinburgh','0311043554');
INSERT INTO littleblackbook VALUES (5,'David Hume','75 Hume Lane, San Jose','02628860');
INSERT INTO littleblackbook VALUES (6,'Elias Illingworth','46 Illingworth Avenue,
Southampton','0423436125');
INSERT INTO littleblackbook VALUES (7,'Kostas Jackson','52 Jackson Drive,
Winchester','01071062664');
INSERT INTO littleblackbook VALUES (8,'Malcolm Krause','72 Krause Street,
Edinburgh','0121741579');
INSERT INTO littleblackbook VALUES (9,'Mario Karasavvas','13 Karasavvas Road, San
Jose','07191274');
INSERT INTO littleblackbook VALUES (10,'Mike Theocharopoulos','51 Theocharopoulos Crescent,
Southampton','0291145557');
```

4.2 Stored Procedure and User Defined Function

If testing an implementation that supports stored procedures and user defined functions, the following should be defined (the syntax here is valid for MySQL, you may have to modify this for other DBMSs):

```
DELIMITER //
CREATE PROCEDURE proc_in_out(IN param1 INT, OUT param2
VARCHAR(30),OUT param3 INT)
BEGIN
declare loc VARCHAR(30);
```

[6] http://amga.web.cern.ch/amga/downloads/glite-amga-soapserver-2.0.0-1.SL4.i386.rpm.

```
select name from littleblackbook where id=param1 LIMIT 1 into param2;
select * from littleblackbook where id=param1+1;
select name from littleblackbook where id=param1+1 LIMIT 1 into loc;
update littleblackbook set name='nothing' where name=loc;
select count(id) from littleblackbook where name='nothing' into param3;
update littleblackbook set name=loc where name='nothing';
END;//
DELIMITER ;

DELIMITER //
CREATE FUNCTION func_in_out(param1 INT) RETURNS VARCHAR(30)
BEGIN
declare loc VARCHAR(30);
declare param2 VARCHAR(30);
select name from littleblackbook where id=param1 LIMIT 1 into param2;
select name from littleblackbook where id=param1+1 LIMIT 1 into loc;
update littleblackbook set name='nothing' where name=loc;
update littleblackbook set name=loc where name='nothing';
return(param2);
END;//
DELIMITER
```

4.3 Data resource

The database described above should be exposed using an externally managed data resource with the abstract name `dair:testresource`. However, this name may be changed as long as it is a valid URI and the new name is substituted for the `dair:testresource` wherever is appears in the tests.

4.4 ResultSet format

Although the WS-DAIR specification does not mandate the use of the JDBC `WebRowSet` [JSR114], it does implicitly suggest that this format should be supported by implementations. The tests are written for implementations supporting the `WebRowSet` format, identified by the dataset format URI '`http://java.sun.com/xml/ns/jdbc`'. In the event that an implementation that does not support `WebRowSet` is tested, the supported dataset format and dataset format URI values should be used in place of those currently defined in the tests.

5 Test suite

The test cases that comprise the WS-DAIR test suite are now listed. Tests should be executed in sequence using a tool such as soapUI (http://www.soapui.org) or a specially-written web services client. Regardless of how the tests are implemented or executed it should be demonstrable that the test suite is submitting requests that are valid with respect to the WS-DAI- and WS-DAIR-defined WSDL and XML Schema as well as checking that the responses are likewise valid with respect to the WSDL and XML Schema.

Section 6 presents the results of the execution of this test suite using soapUI, which explicitly uses SOAP requests to test for compliance of implementations with the specifications. Refer to Appendix 2 for more details about the implementation of the tests using soapUI.

5.1 Implementations supporting CoreResourceList

The following tests apply only to implementations supporting the optional `CoreResourceList` port type.

Test	Operation(s) tested	Description
1	`CoreResourceList:: GetResourceList`	Retrieves the list of data resources from the service. The test is passed if this list contains the resource with the abstract name `dair:testresource`. The resource may be available via a number of

		portTypes, however the arrangement of these portTypes can vary based on the implementation and the test checks only that this resource is returned.
2	`CoreResourceList:: Resolve`	Retrieves the list of data resource addresses for the abstract name `dair:testresource`. Passes if the `CoreResourceList` endpoint used to invoke the `Resolve` operation is contained in the list of addresses.

5.2 Mandatory tests

The following test cases test mandatory features of the specifications and apply to all WS-DAIR implementations.

Test	Operation(s) tested	Description
3	`CoreDataAccess:: GetDataResourcePropertyDocument`	Retrieves the data resource property document of the `dair:testresource` resource. The test is passed if the property document validates against its schema and the `DataResourceManagement` property is set to `ExternallyManaged`. Other properties cannot be tested as their values depend on the specifics of the underlying data resource exposed by the service.
4	`CoreDataAccess:: GetDataResourcePropertyDocument (InvalidResourceNameFault)`	Attempts to retrieve the `PropertyDocument` for an invalid resource, e.g. with resource name `dair:testresource`. The test passes if an `InvalidResourceNameFault` message is received.
5	`CoreDataAccess::Destroy (NotAuthorized Fault)`	Attempts to perform the `Destroy` operation on an externally managed `dair:testresource` resource. The test is passed if a `NotAuthorizedFault` message is received.
6	`SQLAccess:: GetSQLPropertyDocument`	Retrieves the SQL property document of the `dair:testresource` resource. The test fails if the document does not validate against its schema. The test needs to check the following properties are correct given the implementation specific features of the database management system: • `LanguageMap`, • `DatasetMap` and • `SchemaDescription` (if supported).
7	`SQLAccess::SQLExecute`	Execute a query to select 5 rows from the from the `dair:testresource` resource. See query 1 in Appendix 1. The `DataSetFormatURI` parameter should be specified as 'http://java.sun.com/xml/ns/jdbc'. The resulting `DatasetData` element of the

		response message is checked for its conformance to the `WebRowSet` schema and that it correctly returns the 5 rows selected by query 1.
8	`SQLAccess::SQLExecute` (test `InvalidDatasetFormatURI` fault)	Test 7 is repeated with one change, the `DatasetFormatURI` is specified as `dair:notsupporteddataset`. The test passes if an `InvalidDatasetFormatURI` fault is generated.
9	`SQLAccess::SQLExecute` (test `InvalidExpression` fault)	Test 7 is repeated with one change: an invalid query expression is specified. What constitutes an invalid expression depends on the implementation. The test passes if an `InvalidExpressionFault` is generated.
10	`SQLAccessFactory::SQLExecuteFactory`	Uses the `dair:testresource` resource to execute a query to select 5 rows (query 1), the resulting resource created by the factory operation being made available through the `SQLResponse` portType. The test passes if no fault is generated and a data resource address is returned. The created resource is used in subsequent tests.
11	`SQLResponse:: GetSQLResponsePropertyDocument`	Retrieve the property document from the resource created in test number 10. The property document is validated and it is checked that there is a single response item and the following property values exist: `NumberOfSQLRowsets=1`, `NumberOfSQLUpdateCounts=0`, `NumberOfSQLReturnValues=0`, `NumberOfSQLOutputParameters=0`, `NumberOfSQLCommunicationsAreas=0`.
12	`SQLResponse::GetSQLResponseItem`	Retrieve the single response item from the data resource created in test 10, specifying the parameters `position=0`, `count=1`. The `DatasetData` and `DatasetFormatURI` elements (which are implementation dependent) are checked to ensure that the correct query result is returned in the correct format and the `DatasetFormatURI` is correct. The test fails if the `SQLDataset` item contains any of the following elements: `SQLUpdateCount`, `SQLOutputParameter`, `SQLReturnValue`, `SQLCommunicationsArea`.
13	`SQLResponse::GetSQLResponseItem` (`InvalidPositionFault`)	Retrieve the single (Rowset) response item from the resource created in test

		number 10, specifying the parameters: `position=1, count=1`. The test passes if an Invalid `PositionFault` is returned.
14	`SQLResponse::GetSQLResponseItem` (`InvalidCountFault`)	Retrieve the single (Rowset) response item from the resource created in test number 10, specifying the parameters: `position=0, count=2`. The test passes if an `InvalidCountFault` is returned.
15	`SQLResponse::GetSQLRowSet`	Retrieve the Rowset from the data resource created in test 10, specifying the parameters: `position=0, count=1`. The `DatasetData`, and `DatasetFormatURI` elements are checked to ensure that the correct query result is returned in the correct format.
16	`SQLResponseFactory::SQLRowsetFactory`	Use the resource created by test 10 to create a data resource accessible by the `SQLRowset` portType. The parameters: `position=0 and count=1` should be used. The test passes if no fault is generated and a data resource address is returned. The created resource is used in subsequent tests.

17	`SQLRowset::GetSQLRowsetPropertyDocument`	The `RowsetPropertyDocument` is retrieved from the resource created in test number 16 and validated. The test fails if the value of the property `NoOfRows` is not equal to 5. The following elements of the `RowSchema` element are checked:
		`Column-count = 4`
		`Column 1: column-name = id`
		`Column 2: column-name = name`
		`Column 3: column-name = address`
		`Column 4: column-name = phone`

18	`SQLRowset::GetTuples`	Uses the resource created in test number 16 to execute `SQLRowset::GetTuples` specifying the parameters: `position=1` and `count=1`. The `DataFormatURI` should be specified as '`http://java.sun.com/xml/ns/jdbc`'. The test passes if the `DatasetData` element returned in the response message contains a valid `WebRowSet` containing the second row in the selected by query 1.
19	`SQLRowset::GetTuples` (test the `AccessMode` property)	Implementations will either support `Random` or `Forward` access modes, as indicated by the `AccessMode` property of the `SQLRowsetPropertyDocument` (which was retrieved by test 17). This test uses the resource created in test 16 to execute `GetTuples` with the parameters:

		position=0, count=1. If the AccessMode property is set to 'random', the test passes if a valid WebRowSet is returned containing the first row selected by query 1. If the AccessMode property is set to 'forward', the test is passed if an InvalidPositionFault is generated.
20	CoreDataAccess::Destroy (also tests InvalidResourceNameFault)	Performs the Destroy operation on the resource created in test 16. The test passes if no fault is received when executing the Destroy operation and a subsequent attempt to retry the operation described in test 17 fails with an InvalidResourceName fault.

5.3 Optional ServiceBusyFault test

Test	Operation(s) tested	Description
21	SQLAccess::SQLExecute (tests ServiceBusyFault)	Uses dair:testresource to execute multiple instances of test number 7 concurrently to trigger the ServiceBusyFault. The number of instances that must be simultaneously executed is an implementation-specific parameter of this test.

5.4 Optional GenericQuery test

Test	Operation(s) tested	Description
22	CoreDataAccess::GenericQuery	Execute a query to select 5 rows on the from the dair:testresource resource using the GenericQuery operation. See query 1 in Appendix 1. The DataSetFormatURI parameter should be specified as 'http://java.sun.com/xml/ns/jdbc'. The resulting DatasetData element of the response message is checked for its conformance to the WebRowSet schema and that it correctly returns the 5 rows selected by query 1.
23	CoreDataAccess::GenericQuery (tests InvalidLanguageFault)	GenericQuery is executed on dair:testresource specifying a language URI not supported by the service. The URI attribute of the GenericExpression parameter is specified as 'dair:notsupportedlanguage' in the request and the test passes if an InvalidLanguageFault is generated.

5.5 Tests for implementations supporting stored procedures and functions

Test	Operation(s) tested	Description
24	SQLAccess::SQLExecute	Uses the dair:testresource resource to execute the user defined function (see query 3 in Appendix 1). One in parameter is specified with the initial value '1'. The test passes if the correct return value is retrieved in the SQLOutputParameter element of the response.

25	SQLAccess::SQLExecute (tests InvalidSQLExpressionParameters fault)	Uses the dair:testresource resource to execute the user defined function (see query 3 in Appendix 1). Two in parameters are specified with arbitrary values. The test passes if an InvalidSQLExpressionParametersFault is generated, as the specified number of parameters (2) is not equal to the number of parameters supported by the function.
26	SQLAccessFactory::SQLExecuteFactory (test output parameter)	Uses the dair:testresource resource to execute the stored procedure proc_in_out (see query 2 in Appendix 1). The first (in) parameter should have the value '1', the next two (out) parameters can be initialised to arbitrary values. A resource is created, accessible through the SQLResponse portType. The test passes if no fault is generated and a data resource address is returned.
27	SQLResponse:: GetSQLOutputParameter	Retrieve an output parameter from the resource created in test 26. The following parameters should be used: position=0, count=1. The test passes if the correct output parameter is retrieved (see query 2 in Appendix 1).
28	SQLAccessFactory::SQLExecuteFactory (test return value)	Uses the dair:testresource resource to execute the function func_in_out (see query 3 in Appendix 1) creating a resource accessible through the SQLResponse portType. The test passes if no fault is generated and a data resource address is returned.
29	SQLResponse:: GetSQLReturnValue	Retrieve a return value from the resource created in test 28. The test passes if the correct return value (as specified in Appendix 1 for query 3) is retrieved.

5.6 Tests for implementations supporting modifications

Test	Operation(s) tested	Description
30	SQLAccess:SQLExecute	Uses the dair:testresource resource to execute an SQL INSERT statement to add a row to the database. See query 4 in Appendix 1. The test passes if the response message contains an SQLUpdateCount element containing the value 1.
31	SQLAccessFactory::SQLExecuteFactory	Uses the dair:testresource resource to execute an SQL INSERT statement to add a row to the database and make the response available through the SQLResponse portType. See query 5 in Appendix 1. The test passes if no fault is generated and a data resource address is returned.
		Retrieve the update count from the

32	`SQLResponse:: GetSQLUpdateCount`	resource created by test 31. The following parameters should be used `position=0, count=1`. The test passes if the update count value is equal to 1.

5.7 Tests for implementations supporting SQLCommunicationsArea

Test	Operation(s) tested	Description
33	`SQLAccessFactory::SQLExecuteFactory` (test `SQLCommunicationsArea`)	Uses the `dair:testresource` resource to execute the factory operation, specifying an erroneous SQL statement (in this case the specified table should not exist) resulting in error messages contained within the `SQLCommunicationsArea`. See query 6. The test passes if no fault is generated and a data resource address is returned.
34	`SQLResponse:: GetSQLCommunicationsArea`	Retrieve the `SQLCommunicationsArea` detail in the response item of the resource created by test 33. The following parameters should be used: `position=0, count=1`. The content is an implementation specific error message (table does not exist) that should be checked to be correct given the database management system used.

6 Results

This section presents the results of applying the test suite to two independent implementations of WS-DAIR:

- OGSA-DAI implementation of WS-DAIR
- AMGA implementation of WS-DAIR

These were implemented and executed using the soapUI web services development and test environment. Information about soapUI and the test suite implementation are in Appendix 2.

In some cases, issues arose during testing resulted in recommended alterations to the WS-DAI and WS-DAIR specifications. A list of these issues and their corresponding resolutions can be found in section 7.

6.1 OGSA-DAI implementation of WS-DAIR

OGSA-DAI WS-DAIR 1.0, released in December 2008, was used (https://sourceforge.net/projects/ogsa-dai/files/). Changes were made to the WSDL documents that bind the WS-DAI and WS-DAIR WSDL to SOAP/HTTP to specify a document/literal encoding rather than the rpc/literal one used in the 1.0 release. Additional changes made to the release to pass certain tests and reflect certain agreed clarifications in the specifications are documented below.

6.1.1 Changes made to OGSA-DAI WS-DAIR to pass certain tests

Changes were made to OGSA-DAI WS-DAIR to fix bugs highlighted by the tests.

WS-DAI/R namespaces – XPathMatch in TEST1, 2, 31 and PropertyTransfer2 after TEST10, 16, 31

XPath Matches failed if using:

```
declare namespace wsdai="http://www.ggf.org/namespaces/2005/12/WS-DAI"
(//wsdai:DataResourceAbstractName)
```

with error:

```
XPath assertion failed for path
[declare namespace wsdai="http://www.ggf.org/namespaces/2005/12/WS-
DAI"(//wsdai:DataResourceAbstractName)] : Exception: Missing content for xpath ... in
Response.
```

But worked if using:

```
declare namespace wsdai="http://www.ggf.org/namespaces/2005/12/WS-DAI/"
(//wsdai:DataResourceAbstractName)
```

Note the final "/"

Schema Compliance was valid though.

soapUI property transfers likewise failed. The problem affected any test that returned a WS-EPR. An example of a problematic fragment is:

```
<ns3:ReferenceParameters xmlns:ns3="http://www.w3.org/2005/08/addressing">
    <ns4:DataResourceAbstractName xmlns:ns4="http://www.ggf.org/namespaces/2005/12/WS-
DAI/">wsdai:BookDB</ns4:DataResourceAbstractName>
. . .
```

Note how the namespace has the final "/".

Referring to: [http://stackoverflow.com/questions/430990/what-is-the-significance-of-trailing-slashes-in-a-namespace-uri],

"A relative URI c from `http://a/b/` is `http://a/b/c` (a descendant) but from `http://a/b` it would be `http://a/c` (a sibling)."

This namespace with trailing "/" is not in any of the DAIS-WG or the OGSA-DAI XML Schema or WSDL sources. Nor is it in any WSDD or auto-generated Java class. The problem was in the server's WS-EPR construction. In `src/core-clientserver/uk/org/ogsadai/wsdai/core/CoreUtils.java` the WS-EPR is built and the `DataResourceAbstractName` is inserted in-code into the `ReferenceParameters` bean. The constant for the namespace is in `src/core-clientserver/uk/org/ogsadai/wsdai/core/CoreConstants.java`:

```
public static final String DAI_NS = "http://www.ggf.org/namespaces/2005/12/WS-DAI/";
```

with the trailing "/". This was changed to remove the trailing “/” as was the DAIR_NS constant in src`/dair-clientserver/uk/org/ogsadai/wsdai/dair/DAIRConstants.java` and, since the port mappings in config.txt use namespaces too, e.g:

```
wsdai.port.$SQL_RESPONSE_ID$.ResponseServiceResponseFactoryPT={http://www.ggf.org/name
spaces/2005/12/WS-DAIR/}SQLResponseFactoryPT
```

The trailing "/" was removed from these also.

General OGSA-DAI WS-DAIR bugs and changes

Position -1 can yield a:

`<faultstring>java.lang.NumberFormatException: Invalid unsigned int-1]</faultstring>`
This is not an issue. The number -1 is not a valid unsigned integer. Only 0..N are valid.

It was decided that `GetResourceList` should return all port-resource combinations for a service. Thus src`/core-server/uk/org/ogsadai/wsdai/core/executor/CoreResourceListExecutor.java` was changed to implement this.

There were incorrect WSDL/SOAP operation bindings. In `schema/wsdai/dai_service_bindings.wsdl`, the WSDL operation `GetResourceList` was mapped to a SOAP operation `GetDataResourceList`. Likewise in `schema/wsdair/coreresponse_bindings.wsdl`, the WSDL operations `GetSQLUpdateCount`, `GetSQLReturnValue`,`GetSQLOutputParameter`, `GetSQLCommunicationsArea` were mapped to `GetSQLRowsetFactory`. These have all been fixed.

6.1.2 Test run summary

This is a summary of the test runs on the soapUI test suite implementation:

- https://forge.gridforum.org/sf/wiki/do/viewAttachment/projects.dais-wg/wiki/IssuesWithTheWSDAIRProposedRecommendation/wsdair_project_amga 0.4.xml

It used OGSA-DAI WS-DAIR with document/literal encoding and the changed outlined in section 6.1.1. An “O” signifies a pass, “X” a failure.

Test	Results	Optional / Mandatory	Description
1	O	Optional	
2	O	Optional	
3	O	Mandatory	
4	O	Mandatory	
5	O	Mandatory	
6	O	Mandatory	
7	O	Mandatory	
8	O	Mandatory	
9	O	Mandatory	
10	O	Mandatory	
11	**X**	**Mandatory**	**SchemaCompliance failure.**
12	O	Mandatory	
13	O	Mandatory	
14	O	Mandatory	
15	O	Mandatory	
16	**X**	**Mandatory**	**SchemaCompliance failure. A warning about Missing operation GetSQLRowsetFactory. The operation is invoked successfully however.**
17	O	Mandatory	
18	O	Mandatory	
19	O	Mandatory	
20	O	Mandatory	
21	X	Optional	ServiceBusyFault is Not supported
22	X	Optional	GenericQuery is Not supported
23	X	Optional	GenericQuery is Not supported
24	O	Optional	

25	O	Optional	
26	O	Optional	
27	**X**	**Optional**	**XPath Match failure.**
28	O	Optional	
29	O	Optional	
30	**X**	**Optional**	**Schema Compliance failure. Invalid xsi:type qname: 'xsd:int'**
31	O	Optional	
32	O	Optional	
33	O	Optional	
34	**X**	**Optional**	**XPath Match failure.**

Comments on the failures and proposed solutions are as follows.

TEST11 – Schema Compliance

If the Schema Compliance assertion is enabled the following failure is reported.

```
line 33: Invalid xsi:type qname: 'ns19:SQLRowsetConfigurationDocumentType' in element
DefaultConfigurationDocument
```

This is the problematic excerpt from the SOAP response from OGSA-DAI WS-DAIR:

```
<ns15:ConfigurationMap xsi:type="ns15:ConfigurationMapType"
xmlns:ns15="http://www.ggf.org/namespaces/2005/12/WS-DAI">
    <ns15:MessageQName xmlns:ns16="http://www.ggf.org/namespaces/2005/12/WS-
DAIR/">ns16:SQLRowsetFactory</ns15:MessageQName>
    <ns15:PortTypeQName xmlns:ns17="http://www.ggf.org/namespaces/2005/12/WS-
DAIR/">ns17:SQLRowsetPT</ns15:PortTypeQName>
    <ns15:ConfigurationDocumentQName
xmlns:ns18="http://www.ggf.org/namespaces/2005/12/WS-
DAIR">ns18:SQLRowsetConfigurationDocumentType</ns15:ConfigurationDocumentQName>
    <DefaultConfigurationDocument xmlns="">
        <ns15:ConfigurationDocument xsi:type="ns19:SQLRowsetConfigurationDocumentType"
xmlns:ns19="http://www.ggf.org/namespaces/2005/12/WS-DAIR">
        <ns15:Readable>true</ns15:Readable>
        <ns15:Writeable>false</ns15:Writeable>
        <ns15:TransactionInitiation>NotSupported</ns15:TransactionInitiation>
        <ns15:TransactionIsolation>NotSupported</ns15:TransactionIsolation>
        <ns15:ChildSensitiveToParent>Sensitive</ns15:ChildSensitiveToParent>
        <ns15:ParentSensitiveToChild>Sensitive</ns15:ParentSensitiveToChild>
        </ns15:ConfigurationDocument>
    </DefaultConfigurationDocument>
</ns15:ConfigurationMap>
```

This is because OGSA-DAI WS-DAIR's `SQLResponsePropertyDocument` cites the `SQLRowsetConfigurationDocumentType`. The reason for this is that we need to expose the default `SQLRowset` configuration document in the `SQLResponse` property document – since `SQLResponse` can create `SQLRowset` resources. The XML Schema and WSDL for `SQLResponse` do not import `SQLRowset` types. In OGSA-DAI WS-DAIR we changed it so it does (otherwise Apache Axis could not handle such documents). These were the imports we added:

`SQLAccess` WSDL (`wsdair_sqlaccess_porttypes.wsdl`):

```
      <xsd:include schemaLocation="wsdair_sqlresponse_types.xsd"/>
      <xsd:include schemaLocation="wsdair_sqlrowset_types.xsd"/>
```

`SQLResponse` WSDL (`wsdair_sqlresponse_porttypes.wsdl`):

```
      <xsd:include schemaLocation="./wsdair_sqlrowset_types.xsd"/>
```

If the WSDL and XML Schema used by soapUI to validate SOAP requests and responses are changed use OGSA-DAI WS-DAIR's then the problem goes away. It is assumed that AMGA does not encounter this problem as their service does not include the problematic `SQLRowsetConfigurationDocumentType` element.

A resolution for this issue (recommendation 31 of section 7.2) is for the WS-DAI WSDL to include the additional imports above to reflect the implicit dependence of `SQLResponse` property documents on `SQLRowset` configuration document schema (and likewise `SQLAccess` property documents on `SQLResponse` configuration document schema).

TEST16 – GetSQLRowsetFactory

This test fails with:

```
Schema Compliance - FAILED
-> line1: Missing operation [GetSQLRowsetFactory] in wsdl definition
```

If the WSDL and XML Schema used by soapUI to validate SOAP requests and responses are changed use OGSA-DAI WS-DAIRs then the problem does not occur. Also, the problem does not occur if a new soapUI test case for this test is created, rather than using one pre-defined in a soapUI configuration file. In addition:

- The SOAP request is accepted by OGSA-DAI WS-DAIR.
- A SOAP response is received from OGSA-DAI WS-DAIR.
- AMGA's client can invoke this operation on an OGSA-DAI WS-DAIR service.
    - See https://forge.gridforum.org/sf/wiki/do/viewAttachment/projects.dais-wg/wiki/IssuesWithTheWSDAIRProposedRecommendation/AMGA_Client_To_OGSADAI_Server_TEST_20090527.doc
- OGSA-DAI WS-DAIR's client can invoke this operation on an AMGA service.
    - See https://forge.gridforum.org/sf/wiki/do/viewAttachment/projects.dais-wg/wiki/IssuesWithTheWSDAIRProposedRecommendation/OGSADAIClient_To_AMGA_Server_TEST_20090602.doc

In consequence, the authors believe this is a soapUI-related issue rather than an inter-operability issue.

TEST 27 and TEST 34 – XPath Match fails

An example of a TEST27 SOAP response is:

```
<SQLOutputParameter xsi:type="ns1:SQLOutputParameterType"
xmlns:ns1="http://www.ggf.org/namespaces/2005/12/WS-DAIR">
    <index xmlns="">2</index>
    <value xmlns="">Ally Antonioletti</value>
</SQLOutputParameter>
```

XPath Match fails for index and value, e.g.:

```
XPathContains assertion failed for path [declare namespace
wsdair="http://www.ggf.org/namespaces/2005/12/WS-DAIR"
(//wsdair:index)] : Exception:Missing content for xpath [declare namespace
wsdair="http://www.ggf.org/namespaces/2005/12/WS-DAIR"
(//wsdair:index)] in Response
```

Likewise, an example of a TEST34 SOAP response will contain:

```
<SQLState xmlns="">42S02</SQLState>
<VendorCode xmlns="">1146</VendorCode>
<MessageText xmlns="">Table 'wsdairinterop.tabledoesnotexist' doesn't
exist</MessageText>
```

And an XPath Match will fail for `SQLState`, `VendorCode` or `MessageContent`.

If the WS-DAIR and WS-DAIR XML Schema are changed and `elementFormDefault="qualified"` added to the `Core`, `SQLAccess`, `SQLResponse` and `SQLRowset` XML Schema (in OGSA-DAI WS-DAIR files `wsdai_core_types.xsd`, wsdair`_sqlaccess_types.xsd`, `wsdair_sqlresponse_types.xsd`, `wsdair_sqlrowset_types.xsd`), a new version of OGSA-DAI WS-DAIR built and deployed using these then the tests pass. This is recommendation 13 of section 7.1.

TEST30 - xsi:type="xsd:int"

SQLAccess XML Schema (`wsdair_sqlaccess_types.wsdl`) defines:

```
<xsd:element name="SQLUpdateCount" type="xsd:int"/>
```

It is used in:

```
<xsd:complexType name="SQLDatasetType">
  <xsd:complexContent>
    <xsd:extension base="wsdai:DatasetType">
      <xsd:sequence>
        <xsd:element ref="wsdair:SQLUpdateCount" minOccurs="0" maxOccurs="unbounded"/>
        <xsd:element ref="wsdair:SQLOutputParameter" minOccurs="0"
maxOccurs="unbounded"/>
        <xsd:element ref="wsdair:SQLReturnValue" minOccurs="0" maxOccurs="1"/>
        <xsd:element ref="wsdair:SQLCommunicationsArea" minOccurs="0"
maxOccurs="unbounded"/>
      </xsd:sequence>
    </xsd:extension>
  </xsd:complexContent>
</xsd:complexType>
```

TEST30 which submits an `SQLUpdate` gets back:

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
   <soapenv:Body>
      <SQLExecuteResponse xmlns="http://www.ggf.org/namespaces/2005/12/WS-DAIR">
         <SQLDataset>
            <ns1:DatasetFormatURI xmlns:ns1="http://www.ggf.org/namespaces/2005/12/WS-
DAI">http://java.sun.com/xml/ns/jdbc</ns1:DatasetFormatURI>
            <ns2:DatasetData xmlns:ns2="http://www.ggf.org/namespaces/2005/12/WS-
DAI"/>
            <SQLUpdateCount xsi:type="xsd:int">1</SQLUpdateCount>
         </SQLDataset>
      </SQLExecuteResponse>
   </soapenv:Body>
</soapenv:Envelope>
```

and the test fails at Schema Compliance which complains:

```
-> Invalid xsi:type qname: 'xsd:int' in element
SQLDataset@http://www.ggf.org/namespaces/2005/12/WS-DAIR
```

If, directly in the soapUI SOAP response window, the `xsi:type="xsd:int"` is edited out and a request made for it to be re-validated, validation succeeds.

If the XML Schema is changed to:

```
<xsd:complexType name="SQLDatasetType">
  <xsd:complexContent>
    <xsd:extension base="wsdai:DatasetType">
      <xsd:sequence>
        <xsd:element name="SQLUpdateCount" minOccurs="0" type="xsd:int"
maxOccurs="unbounded"/>
        <xsd:element ref="wsdair:SQLOutputParameter" minOccurs="0"
maxOccurs="unbounded"/>
        <xsd:element ref="wsdair:SQLReturnValue" minOccurs="0" maxOccurs="1"/>
        <xsd:element ref="wsdair:SQLCommunicationsArea" minOccurs="0"
maxOccurs="unbounded"/>
      </xsd:sequence>
    </xsd:extension>
  </xsd:complexContent>
</xsd:complexType>
```

Then the SOAP response contains

```
 ...
 <SQLUpdateCount xmlns="">1</SQLUpdateCount>
 ...
```

Enforcing `elementFormDefault="qualified"` resolves the empty namespace issue. However, the `xsi` issue remains.

Curiously, other occurrences of `xsd:int` give no problem, e.g. in `SQLAccess` the XML Schema (`wsdair_sqlaccess_types.xsd`) defines `index` and `value`:

```
  <xsd:complexType name="SQLOutputParameterType">
    <xsd:sequence>
      <xsd:element name="index" type="xsd:int"/>
      <xsd:element name="value" type="xsd:string"/>
    </xsd:sequence>
  </xsd:complexType>
  <xsd:element name="SQLOutputParameter" type="wsdair:SQLOutputParameterType"/>
```

And, then, if calling stored procedure `proc_in_out` via `SQLExecute`, the integers in the `SQLOutputParameter` give no problems:

```
<SQLUpdateCount xsi:type="xsd:int">1</SQLUpdateCount>
<SQLOutputParameter xsi:type="ns4:SQLOutputParameterType"
xmlns:ns4="http://www.ggf.org/namespaces/2005/12/WS-DAIR">
  <index xmlns="">2</index>
  <value xmlns="">Ally Antonioletti</value>
</SQLOutputParameter>
<SQLOutputParameter xsi:type="ns5:SQLOutputParameterType"
xmlns:ns5="http://www.ggf.org/namespaces/2005/12/WS-DAIR">
  <index xmlns="">3</index>
  <value xmlns="">1</value>
</SQLOutputParameter>
```

But the initial `SQLUpdateCount` element does cause the problem to arise.

Note though that `index` and `value` are defined within another type declaration rather than being a top-level type declaration. Likewise `SQLResponse` WSDL (`wsdair_sqlresponse_porttypes.wsdl`) defines:

```
<xsd:element name="GetSQLUpdateCountResponse">
  <xsd:complexType>
    <xsd:sequence>
      <xsd:element maxOccurs="unbounded" minOccurs="1" name="UpdateCount"
type="xsd:int"/>
    </xsd:sequence>
  </xsd:complexType>
</xsd:element>
```

This gives no problems at all, e.g. when calling `GetSQLUpdateCount`:

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
   <soapenv:Body>
      <GetSQLUpdateCountResponse xmlns="http://www.ggf.org/namespaces/2005/12/WS-
DAIR">
         <UpdateCount>1</UpdateCount>
      </GetSQLUpdateCountResponse>
   </soapenv:Body>
</soapenv:Envelope>
```

And for `SQLRowset` XML Schema (wsdair`_sqlrowset_types.xsd`), it too has a top-level `xsd:int`-typed element:

```
<xsd:element name="NoOfRows" type="xsd:int"/>

<xsd:complexType name="SQLRowsetPropertyDocumentType">
  <xsd:complexContent>
    <xsd:extension base="wsdai:PropertyDocumentType">
      <xsd:sequence>
        <xsd:element maxOccurs="1" minOccurs="1" ref="wsdair:RowSchema"/>
        <xsd:element maxOccurs="1" minOccurs="1" ref="wsdair:NoOfRows"/>
        <xsd:element maxOccurs="1" minOccurs="1" ref="wsdair:AccessMode"/>
      </xsd:sequence>
    </xsd:extension>
  </xsd:complexContent>
</xsd:complexType>
<xsd:element name="SQLRowsetPropertyDocument"
type="wsdair:SQLRowsetPropertyDocumentType"/>
```

This gives no problems at all, e.g. when calling `GetSQLRowsetPropertyDocument`:

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
   <soapenv:Body>
      <SQLRowsetPropertyDocument xmlns="http://www.ggf.org/namespaces/2005/12/WS-
DAIR">
         ...
         <NoOfRows>5</NoOfRows>
         <AccessMode>Forward</AccessMode>
      </SQLRowsetPropertyDocument>
   </soapenv:Body>
</soapenv:Envelope>
```

The exact cause of this problem is unclear but it is a tooling issue rather than an inter-operability issue and similar arguments apply as for is the TEST16 issue discussed previously.

Setting the following in the Apache Axis `server-config.wsdd` file should suppress `xsi` insertions:

```
<parameter name="sendXsiTypes" value="false"/>
```

But it does not. Others have e-mailed Axis about this problem over the years but to no response.

6.1.3 A second run

As stated in the investigation of the TEST11 and 16 failures, the soapUI suite validates requests against the WSDL and XML Schema defined by AMGA - if the test suite is changed to validate requests against the WSDL and XML Schema defined by OGSA-DAI then TEST11 and 16 pass.

Likewise, TEST27 and 34 pass if `elementFormDefault="qualified"` is used in the WS-DAI and WS-DAIR XML Schema.

Applying these changes and running the soapUI test suite implementation:

- https://forge.gridforum.org/sf/wiki/do/viewAttachment/projects.dais-wg/wiki/IssuesWithTheWSDAIRProposedRecommendation/wsdair_project_ogsadai_namespace 0.4.xml

gives the following results:

Test	Results	Optional / Mandatory	Description
1	O	Optional	
2	O	Optional	
3	O	Mandatory	
4	O	Mandatory	
5	O	Mandatory	
6	O	Mandatory	
7	O	Mandatory	
8	O	Mandatory	
9	O	Mandatory	
10	O	Mandatory	
11	O	Mandatory	
12	O	Mandatory	
13	O	Mandatory	
14	O	Mandatory	
15	O	Mandatory	
16	O	Mandatory	
17	O	Mandatory	
18	O	Mandatory	
19	O	Mandatory	
20	O	Mandatory	
21	X	Optional	ServiceBusyFault is Not supported
22	X	Optional	GenericQuery is Not supported
23	X	Optional	GenericQuery is Not supported
24	O	Optional	
25	O	Optional	
26	O	Optional	

27	O	Optional	
28	O	Optional	
29	O	Optional	
30	**X**	**Optional**	**Schema Compliance failure. Invalid xsi:type qname: 'xsd:int'**
31	O	Optional	
32	O	Optional	
33	O	Optional	
34	O	Optional	

6.2 AMGA implementation of WS-DAIR

6.2.1 A first run

This was run using AMGA version: 2.0.0pre2:

- http://amga.web.cern.ch/amga/downloads/glite-amga-soapserver-2.0.0-1_pre1.SL4.i386.rpm

The following test suite was used for this run:

- https://forge.gridforum.org/sf/wiki/do/viewAttachment/projects.dais-wg/wiki/IssuesWithTheWSDAIRProposedRecommendation/wsdair_project_amga 0.3.xml

The results were as follows:

Test	Results	Optional / Mandatory	Description
1	O	Optional	
2	O	Optional	
3	O	Mandatory	
4	O	Mandatory	
5	O	Mandatory	
6	O	Mandatory	
7	O	Mandatory	
8	O	Mandatory	
9	O	Mandatory	
10	O	Mandatory	
11	**X**	**Mandatory**	**Schema Compliance Issue (Namespace) - SequenceNumber is expected instead of wsdair: SequenceNumber**
12	O	Mandatory	
13	O	Mandatory	
14	O	Mandatory	
15	O	Mandatory	
16	O	Mandatory	
17	O	Mandatory	
18	O	Mandatory	
19	O	Mandatory	
20	O	Mandatory	
21	X	Optional	ServiceBusyFault is Not supported
22	O	Optional	
23	O	Optional	
24	X	Optional	Stored Procedure is Not supported
25	X	Optional	Stored Procedure is Not supported

26	X	Optional	Stored Procedure is Not supported
27	X	Optional	Stored Procedure is Not supported
28	X	Optional	Stored Procedure is Not supported
29	X	Optional	Stored Procedure is Not supported
30	O	Optional	
31	O	Optional	
32	O	Optional	
33	X	Optional	AMGA does not use Communication Area for a fault message
34	X	Optional	AMGA does not use Communication Area for a fault message

6.2.2 A second run

A second run used validating requests and responses against the XML Schema and WSDL used in OGSA-DAI WS-DAIR:

https://forge.gridforum.org/sf/wiki/do/viewAttachment/projects.dais-wg/wiki/IssuesWithTheWSDAIRProposedRecommendation/wsdair_project_ogsadai 0.4.1.xml

The results were as follows:

Test	Results	Optional / Mandatory	Description
1	O	Optional	
2	**X**	**Optional**	**The CoreResourceList endpoint used to invoke the Resolve operation is not contained in the list of addresses**
3	O	Mandatory	
4	O	Mandatory	
5	O	Mandatory	
6	O	Mandatory	
7	O	Mandatory	
8	O	Mandatory	
9	O	Mandatory	
10	O	Mandatory	
11	**X**	**Mandatory**	**Schema Compliance Issue (Namespace) - SequenceNumber is expected instead of wsdair: SequenceNumber**
12	O	Mandatory	
13	O	Mandatory	
14	O	Mandatory	
15	O	Mandatory	
16	O	Mandatory	
17	O	Mandatory	
18	O	Mandatory	
19	O	Mandatory	
20	O	Mandatory	
21	X	Optional	ServiceBusyFault is Not supported
22	O	Optional	
23	O	Optional	
24	X	Optional	Stored Procedure is Not supported
25	X	Optional	Stored Procedure is Not supported
26	X	Optional	Stored Procedure is Not supported
27	X	Optional	Stored Procedure is Not supported
28	X	Optional	Stored Procedure is Not supported

29	X	Optional	Stored Procedure is Not supported
30	O	Optional	
31	O	Optional	
32	O	Optional	
33	X	Optional	AMGA does not use Communication Area for a fault message
34	X	Optional	AMGA does not use Communication Area for a fault message

TEST2 : The CoreResourceList endpoint used to invoke the Resolve operation is not contained in the list of addresses

EndPoint: http://150.183.250.215:8844/CoreResourceList

Response:

```
<wsdai:DataResourceAddress xsi:type="wsdai:DataResourceAddressType">
            <wsa:Address
xsi:type="wsa:AttributedURIType">http://150.183.250.215:8844/SQLAccess/wsdair_test</ws
a:Address>
            <wsa:ReferenceParameters xsi:type="wsa:ReferenceParametersType">

<wsdai:DataResourceAbstractName>http://150.183.250.215:8844/SQLAccess/wsdair_test</wsd
ai:DataResourceAbstractName>
            </wsa:ReferenceParameters>
         </wsdai:DataResourceAddress>
      </wsdai:ResolveResponse>
```

6.2.3 A third run

A third run used validating requests and responses against the XML Schema and WSDL used in OGSA-DAI WS-DAIR with setting `elementFormDefault="qualified"`.

Changes to the AMGA WS-DAIR Implementation:

- `CoreResourceList::Resolve Operation` returns output compliant to TEST2 pass condition
- AMGA version: 2.0.0pre2: http://amga.web.cern.ch/amga/downloads/glite-amga-soapserver-2.0.0-1_pre2.SL4.i386.rpm.

New Test suite: `wsdair_test_0_1.xml`

- All the default namespaces are qualified.
- possible to test both AMGA and OGSA-DAI WS-DAIR interface.
- URL: https://forge.gridforum.org/sf/wiki/do/viewAttachment/projects.dais-wg/wiki/IssuesWithTheWSDAIRProposedRecommendation/wsdair_test_0_1.xml.

The results were as follows:

Test	Results	Optional / Mandatory	Description
1	O	Optional	
2	O	Optional	
3	O	Mandatory	
4	O	Mandatory	
5	O	Mandatory	
6	O	Mandatory	
7	O	Mandatory	

8	O	Mandatory	
9	O	Mandatory	
10	O	Mandatory	
11	O	Mandatory	
12	O	Mandatory	
13	O	Mandatory	
14	O	Mandatory	
15	O	Mandatory	
16	O	Mandatory	
17	O	Mandatory	
18	O	Mandatory	
19	O	Mandatory	
20	O	Mandatory	
21	X	Optional	ServiceBusyFault is Not supported
22	O	Optional	
23	O	Optional	
24	X	Optional	Stored Procedure is Not supported
25	X	Optional	Stored Procedure is Not supported
26	X	Optional	Stored Procedure is Not supported
27	X	Optional	Stored Procedure is Not supported
28	X	Optional	Stored Procedure is Not supported
29	X	Optional	Stored Procedure is Not supported
30	O	Optional	
31	O	Optional	
32	O	Optional	
33	X	Optional	AMGA does not use Communication Area for a fault message
34	X	Optional	AMGA does not use Communication Area for a fault message

6.3 Summary

In general, a number of tests were performed that required changes to the test suite itself and the implementations. The reasons for these changes have been documented in the corresponding sections. The final runs, test run 2 for OGSA-DAI and test run 3 for AMGA provide the final runs on which the final interoperability is based. For these:

- One test could not be executed with either implementation, Test 21 (`ServiceBusyFault`).

The two implementations never generated the `ServiceBusyFault` due to the nature of their support for concurrency. The difficulty associated with testing this feature has already been discussed in Section 2. It should be noted that `ServiceBusyFault` is defined in a manner consistent with other faults, meaning that the other tests do provide some validation of this untested feature. We do not consider the fact that neither of these implementations supported this fault to have compromised the interoperability between the two implementations in any way.

Each of the tests for optional features was executed by at least one implementation and all tests for mandatory features were executed by both implementations, which is consistent with the aims of the testing process discussed in Section 2. A number of issues were discovered which resulted in recommended alterations to the specifications, which are listed in Section 7. Several tests failed during initial runs of the test suite but were subsequently resolved as follows:

- Test 2 (`CoreResourceList::Resolve`): resolved with an implementation change to AMGA WS-DAIR.
- Test 11 (`SQLResponse::GetSQLResponsePropertyDocument`): passes if the proposed resolution to issue WS-DAIR 31 in Section 7 is carried out.
- Test 16 (`SQLResponseFactory::SQLRowsetFactory`): determined to be an issue related to soapUI's schema validation.

- Tests 27 (`SQLResponse::GetSQLOutputParameter`) and 34 (`SQLReponse::GetSQLCommunicationsArea`): pass if the proposed resolution to WS-DAI issue 13 (`use elementFormDefault="qualified"` throughout the specifications) is carried out.
- Test 30 (`SQLAccess::SQLExecute`): A known Apache Axis tooling issue, discussed in Section 6.1.2.

As each of the above test failures have resolutions listed in Section 7 or have been identified as tooling or test suite implementation issues, the conclusion reached from the interoperability testing process is that *two indepedent implementations of the WS-DAIR/WS-DAI specifications have been shown to be interoperable, subject to the recommended changes listed in Section 7 being made to the specifications*.

7 Specification errata and clarifications

This section summarises the errata and recommended changes to the proposed WS-DAI and WS-DAIR specifications that have been found in the process of implementing and testing the specifications. Some of the recognised issues recommend clarifications to the specification text; other issues may also require changes to the WSDL files specifying the interfaces. For issues that appear here, as the result of a failed test in Section 6, the corresponding resolutions have already been implemented in order to confirm that they result in test cases being passed. Therefore, if the resolutions to the issues listed in this section are made, it will reinforce the interoperability of the two independent implementations of the WS-DAIR specification.

Many of the clarifications and outstanding issues listed for the WS-DAI core specification have implications for the two other WS-DAI descendent specifications, WS-DAIX and WS-DAI-RDF. However, the suggestion here is that these will be noted as trackers for the WS-DAIX proposed recommendation or they can be taken into account by the WS-DAI-RDF specifications as these have not yet been submitted.

It is suggested is that these changes are made directly into the existing propsed WS-DAI and WS-DAIR recommendations or, if the OGF Editor deems it, new versions of the proposed recommendations will be produced obsoleting the previous versions of the WS-DAI and WS-DAIR documents. We are not aware of any other implementations of these proposed recommendations that would be affected by these changes.

7.1 Changes to the WS-DAI Core document

1. Page 9 states that:
   "When a data resource address is returned by a WS-DAI data service, for example, in the case of factory messages, the EPR ReferenceParameter element MUST contain the DataResourceAbstractName element that identifies the data resource to which the address refers."
   At OGF-22 it was concluded that violation of WS-EPR opacity of reference parameters is commonplace.
   Resolution: the following text should be added to the penultimate paragraph of p9:
   "In terms of WSRF realizations this violates the opacity of WS-resource qualified end-point references. But, in order to accommodate the WSRF-agnostic nature of the DAIS specifications this is a necessary transgression."
2. Inconsistency in the specs: on page 14 it states that `ServiceManaged` and `ExternallyManaged` are the two possible values for `DataResourceManagement` but on page 22 it introduces the value `InternallyManaged`. This is a typo as it does not appear in the normative WSDL.
   Resolution: change the `InternallyManaged` to `ServiceManaged`.
3. In the WSDL on p40, `GetDataResourcePropertyDocumentRequest` extends `RequestType` but it should extend `BaseRequestType` as this introduces `DatasetFormatURI` as an input to

`CoreDataAccess::GetDataResourcePropertyDocumentRequest` which is not required.
Resolution: change `GetDataResourcePropertyDocumentRequest` to extend from `BaseRequestType`.

4. The use of the `NotAuthorizedFault` on operations assumes that authorization is done within the service. Some architectures support/encourage authorization to be done before the operation is invoked. Information on the consumer, operation and arguments are intercepted and used to authorize a request before the service operation is even invoked. In such cases, this fault may be redundant. At OGF-22 it was explained that in other environments or implementations it may be down to the WS-DAI implementation to do the authorization in which case the fault is valid. Again this should be explicitly stated.
   Resolution: add the following text to the description of `NotAuthorizedFault` in Section 4.13:
   "Note: the NotAuthorizedFault on operations assumes that authorization is done by the service. In some cases invocation of a service operation will be authorised before it reaches the service. In such cases this fault may be rendered redundant."
   Additionally, for every operation that lists `NotAuthorizedFault` as a possible fault type, change the sentence:
   "NotAuthorizedFault - the consumer is not authorized to perform this operation at this time"
   to:
   "NotAuthorizedFault - service-level authorization was unsuccessful, indicating that the consumer is not authorized to perform this operation at this time".
5. Update the WS-Addressing reference in `wsdai_core_types.xsd` from:
   `http://www.w3.org/2005/08/addressing`
   to:
   `http://www.w3.org/2006/03/addressing/ws-addr.xsd`.
   Resolution: update the WS-Addressing reference in `wsdai_core_types.xsd` to use:
   `http://www.w3.org/2006/03/addressing/ws-addr.xsd`.
6. The specifications should state that if WSRF is being used then the properties in the property document MUST be exposed as individual WSRF resource properties.
   Resolution: add the following text to WS-DAI specification, section 6. WSRF Data Resource, sub-section 6.1.1 DataResourceProperties, p27.
   "For a WSRF-based implementation the properties defined in this section MUST be made available as individual WSRF resource properties."
7. In Section 5.3.1, the `DatasetFormat` element should be a `DatasetFormatURI` element. This appears in both the example XML fragment and the main text.
   Resolution: change all occurrences of `DatasetFormat` should be changed to `DatasetFormatURI` in Section 5.3.1.
8. On p26 defines `GetDataResourceList` and `GetDataResourceListResponse` but WSDL on Appendix 2 defines `GetResourceList` and `GetResourceListResponse`.
   Resolultion: the WSDL is normative, so change `GetDataResourceList` and `GetDataResourceListResponse` on p26 to `GetResourceList` and `GetResourceListResponse`.
9. On p27/p28, WS-Resource 1.2 is cited twice in the list.
   Resolution: remove the second bullet point that references it on p28.
10. On p42, `InvalidExpressionFault` message definition should use "`wsdai:InvalidLanguageFault`" element instead of using "`wsdai:InvalidExpressionFault`" element
    Resolution: on p42 change "`wsdai:InvalidExpressionFault`" to "`wsdai:InvalidLanguageFault`".
11. An optional fault should be added for requests that attempt to fetch too much data, for example an implementation may be able to tell that a direct data access request will result in more data being returned than the service can handle. In such scenarios a fault should be generated indicating to the client that indirect data access should be used. Provisional name for the proposed fault is `DatasetTooLargeFault`.
    Resolution: define `DatasetTooLargeFault` in WS-DAI Appendix A.1 – Core XML Schema

and add the fault to the following operations in the WSDL and document text for WS-DAI and WS-DAIR - WS-DAI Core: `GenericQuery`, WS-DAIR: `SQLExecute`, `SQLExecuteFactory`, `GetSQLResponseItem`, `GetSQLRowset`, `GetSQLRowsetFactory` and `GetTuples`. This resolution affects the core WS-DAI specification and WS-DAIR. Corresponding changes are also recommended to operations in the WS-DAIX and WS-DAI-RDF specifications (operations listed above are in WS-DAI and WS-DAIR, it needs to be decided where the fault is appropriate in the WS-DAIX and WS-DAIR-RDF specifications).

12. The specifications contain examples of various messages but no examples of fault messages. Some examples should be included as from the WSDL alone it can sometimes be difficult for developers to verify that the actual XML messages produced by an implementation are correct.
Resolution: add examples of fault messages to WS-DAI Core, section 4.1.3 Faults, p12:

```
<ns1:InvalidResourceNameFault
    xmlns:ns1="http://www.ggf.org/namespaces/2005/12/WS-DAI"/>
<ns1:NotAuthorizedFault
    xmlns:ns1="http://www.ggf.org/namespaces/2005/12/WS-DAI"/>
<ns1:InvalidDatasetFormatFault
    xmlns:ns1="http://www.ggf.org/namespaces/2005/12/WS-DAI"/>
<ns1:InvalidExpressionFault
   xmlns:ns1="http://www.ggf.org/namespaces/2005/12/WS-DAI"/>
```

13. There is an inconsistency in the WSDL ranging across the entire family of WS-DAI specifications with regards to `ElementFormDefault` being set in some places to "`qualified`" and in other places being implicitly set to "`unqualified`". The consensus is that "`qualified`" should be used throughout and namespaces should be explicitly set in all of the WSDL/XML Schema defined by any WS-DAI specification.
Resolution: explicitly set the `ElementFormDefault` across all specs to "`qualified`".
14. There is a certain degree of ambiguity and some inconsistencies regarding the `InvalidLanguageFault` and `InvalidExpressionFault`. The former could be interpreted as a subset of the latter and the exact conditions under which each fault is generated need to be clarified. The passing of `LanguageURI`s to query operations is related to this issue; currently only the `GenericQuery` operation allows for this, but it has been noted that multiple languages may be supported by specific realisations, e.g. different versions of SQL, so passing a `LanguageURI` to realisation specific query operations may be valid. However, this should be optional as many implementations will only support one language. The consensus is as follows:
    i. Specifying a `LanguageURI` that is not supported SHOULD result in an `InvalidLanguageFault`. This is the only condition under which an `InvalidLanguageFault` can be generated and any other problems with an expression must result in an `InvalidExpressionFault`.
    ii. When a query operation is invoked using a resource that supports only one language, any processing of the `LanguageURI` element that may appear in the message is optional. The implementation may choose to execute the expression using its supported language even if an invalid `LanguageURI` is specified. This ensures that `LanguageURI` and `InvalidLanguageFault` do not need to be supported by implementations that support only one language.

    Resolution: state in Section 5.1.7 that:
    "a LanguageURI that is not supported SHOULD result in an InvalidLanguageFault. This is the only condition under which an InvalidLanguageFault can be generated and any other problems with an expression must result in an InvalidExpressionFault. InvalidLanguageFault is an optional fault; it does not need to be supported by implementations that do not provide query operations supporting multiple languages".
15. Must configurable properties provided by a client in a configuration document be respected or can they all, or a subset of them, be ignored if the implementation precludes it? This should be clarified in the specifications – the configuration document is advisory not compulsory – the client should get information as to what was implemented or the client can query the property document.

Resolution: add to Section 5.2:
"Configuration properties passed to the service by a client are only an advisory, a service MAY choose to ignore these and use default values".

16. The `GetResourceList` operation is ambiguous: should one data resource address be provided for each possible port/resource combination to be returned or just one data resource address per resource? If the latter is the case, then how is the port to be chosen? Is this implementation-dependant?
    Consensus is for it to return all combinations.
    Resolution: add to Section 5.5.1:
    "In the case where a data resource exposed by a service via multiple addresses (e.g. via multiple ports) then all possible data resource-address combinations SHOULD be returned".
17. Experience of implementing WS-DAI with the the JAXB (http://jaxb.dev.java.net) framework for binding XML documents to Java classes has identified an issue with the way in which `DataResourceAddressType` is defined. Specifically, the problem is that the WS-Addressing `EndpointReferenceType` was not intended to be extended by other types and is therefore mapped to a final Java class by JAXB. As WS-DAI defines `DataResourceAddressType` as an extension of `EndpointReferenceType`, the JAXB framework maps the WS-DAI XML Schema to Java classes that will not compile (due to the illegal extension of a final class). Given the importance of producing specifications compatible with tooling and the fact that this problem has been experienced by multiple independent developers implementing WS-DAI, it is recommended that `EndpointReferenceType` is used directly. This change requires minimal modificatons to the specifications because even though `DataResourceAddressType` extends `EndpointReferenceType`, no extra elements are defined by `DataResourceAddressType`, it simply extends `EndpointReferenceType` without adding anything.
    Resolution: remove the definition of `DataResourceAddressType`. In all places where `DataResourceAddressType` is used, replace it with the WS-Addressing `EndpointReferenceType`. Note that this change has already been successfully implemented in two independent WS-DAI implementations in order to test the validity of the change.

7.2 Changes to the WS-DAIR document

1. `WebRowSet` reference – the appendices cite

```
<xsd:import namespace="http://java.sun.com/xml/ns/jdbc"
       schemaLocation="webrowset-jdbc150.xsd" />
```

   This can be accessed by visiting:
   - [JSR114] J. Bruce, JSR-000114 JDBC RowSet Implementations, Final Release, 07 April 2004.
   - http://jcp.org/aboutJava/communityprocess/final/jsr114.
   - JDBC(TM) RowSet Implementations 1.0.1
   - Select Reference Implementation then scroll down to JDBC Rowset Implementations 1.0.1 - Maintenance Release (July 16, 2004), select download, download, unzip the bundle, unzip the rowset.jar in the bundle and the XSD file is in: `jdbc_rowset_tiger1.0.1mrel-ri/javax/sql/rowset/webrowset.xsd`.

   Resolution: update the WS-DAIR [JSR114] reference on p27 to include the above information as to how the schema can be accessed.

2. On page 6 it states that `WebRowSet` is "one of the valid ResponseTypes", yet on page 7 it specifies that "All services adopting the `SQLAccess` interface MUST provide at least the following value to indicate that rowset databases can be returned in WebRowSet [JSR114] format." The support for `WebRowSet` format should be made stronger. The fact that the `WebRowSet` schema is

already imported in the SQLAccess XML Schema supports this course of action.
Resolution: state on p7 that:
"All services that have the the SQLAccess interface MUST provide support to return rowset databases in WebRowSet [JSR114] format. As a consequence of this statement they MUST also state that rowset databases can be returned in WebRowSet format."

3. It is important that the distinction between a SQLResponse resource and the associated SQLResponseAccess portType is respected and made clear. For example, p6 cites SQLResponseAccess but section 6.4 cites the messages as from SQLResponse, when it should also be SQLResponseAccess.
   Resolution: make the following changes:
   - p12, line before the XML fragment, change: "a SQLResponse interface" to "an SQL response via the SQLResponseAccess interface".
   - Section headings for 6.4, 6.4,1, 6.4.2, 6.4.3, 6.4.4, 6.4.5, 6.4.6, 6.4.7 change "SQLResponse" to "SQLResponseAccess".
4. Similar comments apply to the distinction made between a SQLRowset resource and the associated SQLRowsetAccess portType. For example, p6 cites SQLRowsetAccess but section 7.4 cites the messages to be from SQLRowset when it should be SQLRowsetAccess. Also, on p22 there is SQLRowset in the Section 7.4.1 heading but in the Section 7.4.2 heading it is (correctly) cited as SQLRowsetAccess.
   Resolutions: in Section headings for 7.4, 7.4.1, 7.4.2 change "SQLRowset" to "SQLRowsetAccess".
5. In the final paragraph on p8, SQLDescription is cited yet figure 1 cites SQLAccessDescription as does p9 section 5.4.1 paragraph 1.
   Resolution: change SQLDescription on p8 to SQLAccessDescription.
6. In p10 paragraph 2, DataseFormatURI should be DatasetFormatURI.
   Resolution: change DataseFormatURI to DatasetFormatURI on p10.
7. There is an inconsistency between the SQLExecuteRequestParameters on p10 and in the XML Schema of Appendix 2 where it is SQLParameter. SQLExpressionParameters also occurs.
   Resolution: change SQLExpressionParameters in the main text to SQLParameter.
8. On p13/14 the descriptions of the properties reads "The total number of … in the SQLExecuteResponse". But this section is describing an SQLReponse.
   Resolution: change SQLExecuteResponse to SQLReponse.
9. In p14 section 6.2, the reference to SQLAccess should be to SQLResponseAccess.
   Resolution: change SQLAccess to SQLResponseAccess on p14.
10. The comment on p15 section 6.4 paragraph 1 states:
    "This allows access to each SQLExecuteResponseType in the SQLExecuteResponse"
    which is confusing as the SQLExecuteResponseType has not been mentioned up to that point. Is it a typo and meant to read SQLExecuteResponseItemType? It would be better if the text also explained that it allows:
    "indirect access to an SQLResponse resource created via an SQLAccessFactory"
    which emphasises this is a different service for a different resource (rather than continually referring to SQLExecute and SQLExecuteResponse).
    Resolution: change:
    "This allows access to each SQLExecuteResponseType in the SQLExecuteResponse"
    to:
    "This allows access to each element in the SQLExecuteResponse providing indirect access to an SQLResponse resource created via the SQLAccessFactory interface".
11. In p16 section 6.4.2 paragraph 1, SQL Response should be replaced with SQLResponse for consistency.
    Resolution: change: "SQL Response", in Section 6.4.2, with "SQLResponse".
12. On p19, in the description of the "Count?" input parameter, "item" should be replaced with "reference".
    Resolution: in the description of count on p19 change "one item" by "one reference".

13. Page 21 cites `Rowset` data resource but elsewhere it is cited as `SQLRowset`.
    Reference: change `Rowset` to `SQLRowset`.
14. Page 20 cites `RowsetPropertyDocument` but on p21 and the example it is cited as `SQLRowsetPropertyDocument`.
    Resolution: in Section 7.1 on p20 change `RowsetPropertyDocument` to `SQLRowsetPropertyDocument`.
15. On p23, in the description of the "Count?" Input parameter, "item" should be replaced with "tuple".
    Resolution: on p23 in the description of Count change "item" with "tuple".
16. Sometimes "Set" starts with capital "S" and sometimes with small "s." For example, `SQLRowSet`, `SQLRowset`, `SQLRowSetAccess`, `SQLRowsetAccess`, `SQLDataSet`, `SQLDataset` - they need to be consistent.
    Resolution: "Set" should not start with an upper-case S in `SQLRowset`. All occurrences of `SQLRowset`, either standalone or in another word should use lower-case s. For example, `SQLRowset` is correct, but `SQLRowSet` is not. Similarly, `SQLDataSet` in Section 5.4.2 should be changed to `SQLDataset`. Note, `WebRowSet` should not be changed to `WebRowset`.
17. In p7, at the end of section 5.1.3, `/wsdair:SQLPropertyDocument` is written twice, e.g. `/wsdair:SQLPropertyDocument/wsdair:SQLPropertyDocument`. Also, in p21, at the end of section 7.1.3, `/wsdair:RowsetPropertyDocument` is written twice.
    Resolution: remove the second instance of "`/wsdair_SQLPropertyDocument`" for both cases.
18. In p7, at the last paragraph, `SQLDescription` should be replaced with `SQLAccessDesciption` for consistency with Figure 1.
    Resolution: change `SQLDescription` in p7 last paragraph with `SQLAccessDescription`.
19. In Figures 1, 2 and 3, `SQLRowSet` should be replaced with `DatasetData` to be consistent with section 5.4.2.
    Resolution: change `SQLRowSet` in Figures 1, 2 and 3 to `DatasetData`.
20. On p9, section 5.4.1, `GetSQLDocumentPropertyRequest` and `GetSQLDocumentPropertyResponse` should be replaced with `GetSQLPropertyDocumentRequest` and `GetSQLPropertyDocumentResponse` respectively.
    Resolution: change `GetSQLDocumentPropertyRequest` and `GetSQLDocumentPropertyResponse` in Section 5.4.1 to `GetSQLPropertyDocumentRequest` and `GetSQLPropertyDocumentResponse` respectively.
21. On p9, section 5.4.1, `PropertyDocument` is shown as the entity returned in `GetSQLDocumentPropertyResponse` but actually it is an `SQLPropertyDocument` that can be returned. Likewise for p15 is an `SQLResponsePropertyDocument` and for p23 an `SQLRowsetPropertyDocument`.
    Resolution: change `PropertyDocument` to be `SQLPropertyDocument` on p9, Section 5.4.1, to `SQLResponsePropertyDocument` on p15 and to `SQLRowsetPropertyDocument` on p23.
22. On p19, section 6.5.1, `SQLRowsetFactory`, `SQLRowsetFactoryRequest`, and `SQLRowsetFactoryResponse` should be replaced with `GetSQLRowsetFactory`, `GetSQLRowsetFactoryRequest`, and `GetSQLRowsetFactoryResponse` respectively.
    Resolution: change `SQLRowsetFactory`, `SQLRowsetFactoryRequest`, and `SQLRowsetFactoryResponse` with `GetSQLRowsetFactory`, `GetSQLRowsetFactoryRequest`, and `GetSQLRowsetFactoryResponse` respectively on p19, Section 6.5.1.
23. In Appendix A.2, SQLExpression should be declared not to be abstract as it can be instantiated.
    Resolution: remove the `abstract="true"` from the element declaration.
24. In Section 6.1.1, where `SQLResponseItem` is introduced, it is not clear whether the order of the items retrieved is important or not, i.e. if we should hold a list of items in the same order in which they are retrieved from a database or not. OGSA-DAI WS-DAIR opted to ignore the order in which items were retrieved, it therefore maintained a different list for each of the items (row sets, update counts, output values, communication areas, etc.). Clarification of this is required in the document as this impacts on operations such as `getSQLRowset(position, count)` - do the

parameters refer to a separate list of `SQLRowsets` or a list of response items which contain not only `SQLRowsets` but other items too? This would potentially result in different parameter values being used depending on how the implementation handles this. OGSA-DAI WS-DAIR opted for having one big list that has the retrieved objects in the following order:

- `SQLRowsets`
- `SQLUpdateCounts`
- `SQLOutputParameters`
- `SQLReturnValue`
- `SQLCommunicationsArea`

Resolution: in Section 6.4.2 (`GetSQLResponseItem`), state that:
"response items SHOULD be ordered as follows:

- SQLRowsets
- SQLUpdateCounts
- SQLOutputParameters
- SQLReturnValue
- SQLCommunicationsArea

as this is the order in which they are listed in the specification document".
In some cases other orderings may be used so this ordering cannot be guaranteed hence also explicitly state that:
"when retrieving a specific response item type (e.g. using GetSQLRowset, GetSQLUpdateCount etc.), the position and count parameters MUST use relative ordering with respect to items of that type."
An example should be provided to show how this works, for example:
"if there are the following items [Rowset1, Rowset2, UpdateCount1, UpdateCount2], the consumer should use GetUpdateCount with position=0 and count=1 to get UpdateCount1, i.e. not position=2".

25. In `SQLExecute` and in `SQLExecuteFactory`, the `SQLExpressionParameters` are assumed to appear only in stored procedures or functions ("…if it is a call to a stored procedure or function"). Does this mean that `SQLExpressionParameters` cannot be used for SQL parameterised queries? OGSA-DAI WS-DAIR opted against this as it seemed intuitive that parameterised queries should be supported as well.
    Resolution: remove the text:
    "if it is a call to a stored procedure or function."
    from the `SQLExpressionParameters` bullet of p10 and p12.
26. Compared to other similar operations in DAIR and DAIX, why does `SQLResponseFactory::SQLRowsetFactory` not throw the `InvalidPortTypeQNameFault` and `InvalidConfigurationDocumentFault` errors?
    Resolution: `SQLRowsetFactory` should support `InvalidPortTypeQNameFault` and `InvalidConfigurationDocumentFault` for consistency with the other factory operations.
27. Removal of the `InvalidGetTuplesRequestFault` associated with `SQLRowsetAccess::GetTuples` is recommended. This fault is for "XML syntax error or XML schema non-compliance" and if the specifications were consistent with this every operation would have an equivalent fault. It can be expected that tooling such as Axis/JAXB etc. would ensure schema compliance and such faults are redundant in the specifications.
    Resolution: remove `InvalidGetTuplesRequestFault` from the XML Schema and WSDL.
28. In the main body of the text, the WSDL and XML Schema the entity `SQLCommunicationsArea` is used. The related operation in the text is cited as `GetSQLCommunicationsArea`. However, the XML Schema and WSDL for this operation uses `GetSQLCommunicationsArea` (for its element, message parts and operation). According to "SQL, the complete reference" by James R. Groff, Paul N. Weinberg the "SQL Communications area" was pioneered by early IBM products. The most important part of it, the `SQLCODE` variable, became part of the SQL standard. The specification should be consistent with the book and within itself.
    Resolution: all occurrences of `SQLCommunicationArea` in the text, XML Schema and WSDL should be changed to `SQLCommunicationsArea`.

29. During testing it was found that `GetSQLPropertyDocument` sometimes failed with an operation not found exception on the server. This is thought to be related to the following. In the portType definition each input and output message is given a "name", e.g.

```
<wsdl:operation name="GetSQLPropertyDocument">
<wsdl:input message="wsdair:GetSQLPropertyDocumentRequest"
       name="GetSQLPropertyDocumentRequest"/>
<wsdl:output message="wsdair:GetSQLPropertyDocumentResponse"
name="GetSQLPropertyDocumentResponse"/>
```

The same comments apply to `GetSQLResponsePropertyDocument` and `GetSQLRowsetPropertyDocumentRequest`. All other input and output message definitions do not use the name attribute, e.g.

```
<wsdl:operation name="SQLExecute">
<wsdl:input message="wsdair:SQLExecuteRequest"/>
<wsdl:output message="wsdair:SQLExecuteResponse"/>
…
```

Resolution: the `name="..."` attributes of the `GetSQLPropertyDocument` operation messages in the WSDL should be removed. This will make them consistent with other operations as defined by the WS-DAI-* WSDL documents. A check for consistency across the board (including other realisations) should be done.

30. There are uses of both http://www.sqlquery.org/sql-92 and http://www.sql.org/sql-92. Googling http://www.sqlquery.org/sql-92 throws up the example in WS-DAIR as the fourth hit. The first three hits have no trace of this URL, which does not exist. An exact search only throws up one hit - WS-DAIR! The use of this URL is not recommended as an example language URI for SQL in the specification document.
Resolution: use http://www.sql.org/sql-92 as a language URI for SQL in all examples in the WS-DAIR document. Delete http://www.sqlquery.org/sql-92 wherever it appears.
31. Some stored procedures may return only return values and no actual datasets. It therefore seems logical to make the `<DatasetData>` element of the `SQLDataset` (a part of the `SQLResponse` message) optional.
Resolution: `<DatasetData>` is to be made an optional element of the `SQLDataset` element.
32. At times, an implementation may embed the default `SQLRowset` configuration document in the `SQLResponse` property document – as the `SQLResponse` operation can create `SQLRowset` resources a client could submit a default configuration document for their new `SQLRowset`. The WS-DAIR XML Schema and WSDL for `SQLResponse` do not import `SQLRowset` types. The OGSA-DAI WS-DAIR implementation changed this so that it does (otherwise Apache Axis could not handle such documents). These were the imports added:
`SQLAccess` WSDL (`wsdair_sqlaccess_porttypes.wsdl`):

```
 <xsd:include schemaLocation="wsdair_sqlresponse_types.xsd"/>
 <xsd:include schemaLocation="wsdair_sqlrowset_types.xsd"/>
```

`SQLResponse` WSDL (`wsdair_sqlresponse_porttypes.wsdl`)

```
 <xsd:include schemaLocation="./wsdair_sqlrowset_types.xsd"/>
```

Resolution: modify the WSDL to include the additional imports above to reflect the implicit dependence of `SQLResponse` property documents on `SQLRowset` configuration document schema (and likewise `SQLAccess` property documents on `SQLResponse` configuration document schema).

8 Conclusion

This document has reported the results of interoperability testing for two implementations of the WS-DAIR specification. The two implementations satisfy the DAIS-WG interoperability testing criteria in [DAIS-Interop]. A number of recommended changes to the WS-DAI and WS-DAIR specifications are suggested which should improve clarity and interoperability between the existing and future implementations.

9 Security Considerations

This document does not address security issues. Security was not used for any of the interoperability tests.

10 Contributors

Steven Lynden,
Information Technology Research Institute,
National Institute of Advanced Industrial Science and Technology (AIST),
Central 2, Umezono 1-1-1,
Tsukuba,
Ibaraki 305-8568
Japan.
steven.lynden@aist.go.jp

Mario Antonioletti (Corresponding Author),
EPCC,
JCMB,
The King's Buildings,
Mayfield Road,
Edinburgh EH9 3JZ,
United Kingdom.
mario@epcc.ed.ac.uk

Mike Jackson,
EPCC,
JCMB,
The King's Buildings,
Mayfield Road,
Edinburgh EH9 3JZ,
United Kingdom.
michaelj@epcc.ed.ac.uk

Sunil Ahn,
KISTI,
Gwahak-ro 335, Yuseong-gu,
Daejeon 305-806
Rep. of Korea
siahn@kisti.re.kr

11 Acknowledgements

Part of this work was made possible through resources provided by OMII-UK. OMII-UK is funded by EPSRC through the UK e-Science Core Programme and through the JISC. We would also like to acknowledge the National Institute of Advanced Industrial Science and Technology (AIST) and the Korea Institute of Science and Technology Information (KISTI) for supporting this work.

We would also like to thank Elias Theocharopoulos, one of the developers of the OGSA-DAI DAIR implementation from the OGSA-DAI team, who spotted a fair number of the issues/problems with the proposed recommendation that are addressed in this document.

12 Intellectual Property Statement

13 Disclaimer

14 Full Copyright Notice

15 References

[GFD.152]
C. Catlett, C. de Laat, D. Martin, G. Newby and D. Skow, Open Grid Forum Documents Process and Requirements. Open Grid Forum. June 2009. http://www.ogf.org//documents/GFD.152.pdf.

[DAIS-Interop]
S. Lynden, N. Paton, D. Pearson, Interoperability Testing for DAIS Working Group Specifications. Draft, Global Grid Forum, 19 June 2006. http://www.ogf.org/documents/GFD.77.pdf.

[WS-DAI]
M. Antonioletti et al. Web Services Data Access and Integration (WS-DAI) Specification Version 1.0., Global Grid Forum, 21 June 2006. http://www.ogf.org/documents/GFD.74.pdf.

[WS-DAIR]
M. Antonioletti et al. Web Services Data Access and Integration – The Relational Realisation (WS-DAIR) Specification Version 1.0. Draft, Global Grid Forum, 21 June 2006. http://www.ogf.org/documents/GFD.76.pdf.

[WS-DAIX]
M. Antonioletti et al. Web Services Data Access and Integration – The XML Realisation (WS-DAIX) Specification Version 1.0. Draft, Global Grid Forum, 21 June 2006. http://www.ogf.org/documents/GFD.75.pdf.

[JSR114]
J. Bruce, JSR-000114 JDBC RowSet Implementations, Final Release, 07 April 2004.http://jcp.org/aboutJava/communityprocess/final/jsr114.

Appendix 1 SQL queries

The SQL queries referenced in the test suite executed using MySQL.

Query 1

```
select * from littleblackbook where id < 6;

+------+-------------------+----------------------------------+------------+
| id   | name              | address                          | phone      |
+------+-------------------+----------------------------------+------------+
|    1 | Ally Antonioletti | 101 Antonioletti Road, San Jose  | 087192027  |
|    2 | Amy Atkinson      | 70 Atkinson Crescent, Southampton | 0105931111 |
|    3 | Bartosz Chue Hong | 30 Chue Hong Gardens, Winchester  | 04476816   |
|    4 | Craig Dobrzelecki | 72 Dobrzelecki Place, Edinburgh   | 0311043554 |
|    5 | David Hume        | 75 Hume Lane, San Jose           | 02628860   |
+------+-------------------+----------------------------------+------------+
```

Query 2

```
call proc_in_out(1,@var1,@var2)

+------+--------------+----------------------------------+------------+
| id   | name         | address                          | phone      |
+------+--------------+----------------------------------+------------+
|    2 | Amy Atkinson | 70 Atkinson
Crescent, Southampton | 0105931111 |
+------+--------------+----------------------------------+------------+
```

Query 3

```
Select func_in_out('1');

+--------------------+
| func_in_out('1')  |
+--------------------+
| Ally Antonioletti |
+--------------------+
```

Query 4

```
INSERT INTO littleblackbook VALUES (11,'Mike Hume','123 Atkinson Road, Winchester','0871231227');
Query OK, 1 row affected (0.00 sec)
```

Query 5

```
INSERT INTO littleblackbook VALUES (12,'Richard Smith','133 Highfield Road,
Manchester','087837464');
Query OK, 1 row affected (0.00 sec)
```

Query 6

```
select * from tabledoesnotexist;

SQL Communications Area details (obtained using OGSA-DAI WS-DAIR implementation):

SQLState: 42S02
Vendor Code: 1146
Message: Table 'wsdair.tabledoesnotexist' doesn't exist
```

Appendix 2 soapUI Test Suite

The WS-DAIR inter-operability test suite used by AMGA and OGSA-DAI is implemented in soapUI (www.soapui.org). SoapUI is a SOAP-based web services test and development tool.

You can use this suite with your own deployed services as described below. Examples of the settings used for AMGA and OGSA-DAI WS-DAIR are given.

Prerequisites

These instructions assume that you have created a test database as described in Appendix 1 and have configured a WS-DAIR service to expose this. It assumes that this service exposes at least one SQLAccess-compliant ExternallyManaged resource.

Download

1. Download soapUI from http://www.soapui.org, and install it on your computer.
   The test suite uses SoapUI version 3.0.1. All the details explained here assume that a user installed the SoapUI tool on a Windows platform.

2. Download the WS-DAIR test suite from the DAIS-WG grid forge site:
   - http://forge.gridforum.org//sf/wiki/do/viewAttachment/projects.dais-wg/wiki/IssuesWithTheWSDAIRProposedRecommendation/wsdair_test_0_1.zip

   The downloaded file includes 5 files and 1 directory:

   - wsdair_test_0_1.xml: main test suite
   - property_amga.txt: properties defined for AMGA WS-DAIR (eg. service endpoints)
   - property_ogsa.txt: defined for OGSA-DAI WS-DAIR (eg. service endpoints)
   - runTestAMGA.bat: test executable for AMGA WS-DAIR
   - runTestOGSA.bat: test executable for OGSA-DAI WS-DAIR
   - schema: wsdl and xsd files included in the WS-DAI & WS-DAIR specification and some of wsdl files for OGSA-DAI WS-DAIR implementation.

Configure

1. Create a property file for your own implementation. There are two examples included in the test suite distribution. The following is an example property file (property_amga.txt) for AMGA WS-DAIR.

```
DRAN=http://150.183.250.215:8844/SQLAccess/wsdair_test
RandomAccess=1
EP_SQLAccess=http://150.183.250.215:8844/SQLAccess
EP_SQLAccessFactory=http://150.183.250.215:8844/SQLAccess
EP_SQLResponse=http://150.183.250.215:8844/SQLResponse
EP_SQLResponseFactory=http://150.183.250.215:8844/SQLResponse
EP_SQLRowset=http://150.183.250.215:8844/SQLRowset
EP_CoreResourceList=http://150.183.250.215:8844/CoreResourceList
EP_CoreDataAccess_SQLAccess=http://150.183.250.215:8844/CoreDataAccess
EP_CoreDataAccess_SQLRowset=http://150.183.250.215:8844/CoreDataAccess
```

- DRAN: the Data Resource Abstract Name for the initial default resource.
- RandomAccess: Set to 1 if the SQLResponse service supports “Random Access”.
- EP_SQLAccess: the endpoint for the SQLAccess portType.
- EP_SQLAccessFactory: the endpoint for SQLAccessFactory portType.
- EP_SQLResponse: the endpoint for the SQLResponse portType.
- EP_SQLResponseFactory: the endpoint for SQLResponseFactory portType.
- EP_SQLRowset: the endpoint for the SQLRowset portType.

- EP_CoreResourceList: the endpoint for the CoreResourceList portType.
- EP_CoreDataAccess_SQLAccess: the endpoint for the CoreDataAccess portType for an SQLAccess resource.
- EP_CoreDataAccess_SQLRowset: the endpoint for the CoreDataAccess portType for an SQLRowset resource.

2. Create a batch script for your own implementation. The following is an example batch script (runTestAMGA.bat) for AMGA WS-DAIR

```
set DRAN=http://150.183.250.215:8844/SQLAccess/wsdair_test
set RandomAccess=1
set EP_SQLAccess=http://150.183.250.215:8844/SQLAccess
set EP_SQLAccessFactory=http://150.183.250.215:8844/SQLAccess
set EP_SQLResponse=http://150.183.250.215:8844/SQLResponse
set EP_SQLResponseFactory=http://150.183.250.215:8844/SQLResponse
set EP_SQLRowset=http://150.183.250.215:8844/SQLRowset
set EP_CoreResourceList=http://150.183.250.215:8844/CoreResourceList
set EP_CoreDataAccess_SQLAccess=http://150.183.250.215:8844/CoreDataAccess
set EP_CoreDataAccess_SQLRowset=http://150.183.250.215:8844/CoreDataAccess

C:\"Program Files"\eviware\soapui-3.0.1\bin\testrunner.bat -fresult -r -PDRAN=%DRAN% -
PRandomAccess=%RandomAccess% -PEP_SQLAccess=%EP_SQLAccess% -
PEP_SQLAccessFactory=%EP_SQLAccessFactory% -PEP_SQLResponse=%EP_SQLResponse% -
PEP_SQLResponseFactory=%EP_SQLResponseFactory% -PEP_SQLRowset=%EP_SQLRowset% -
PEP_CoreResourceList=%EP_CoreResourceList% -
PEP_CoreDataAccess_SQLAccess=%EP_CoreDataAccess_SQLAccess% -
PEP_CoreDataAccess_SQLRowset=%EP_CoreDataAccess_SQLRowset% -r wsdair_test_0_1.xml
```

It is necessary to modify the property values and the path of “testrunner.bat” program properly, these are underlined at the above example, according to your WS-DAIR deployment, used test platform and the installed location of soapUI.

Run the tests with command line interface

1. Execute the created batch file at the configure step 2.

```
C:\wsdair_test> runTestAMGA.bat
```

2. Check “result” directory to see whether there are failed TESTs.

```
C:\wsdair_test> dir /w result
[.]
[..]
15OptionalStoredProcedure-Function-TEST24-0-FAILED.txt
15OptionalStoredProcedure-Function-TEST25-0-FAILED.txt
15OptionalStoredProcedure-FunctionSQLResponse-TEST28-0-FAILED.txt
15OptionalStoredProcedure-FunctionSQLResponse-TEST29-0-FAILED.txt
15OptionalStoredProcedure-Procedure-TEST26-0-FAILED.txt
15OptionalStoredProcedure-Procedure-TEST27-0-FAILED.txt
16OptionalSQLCommunicationArea-SQLCommunicationArea-TEST33-0-FAILED.txt
16OptionalSQLCommunicationArea-SQLCommunicationArea-TEST34-0-FAILED.txt
```

Run the tests with GUI interface

1. Start soapUI and Load the test suite :
   - Select File=>Import Project
   - Select the WS-DAIR test suite XML file. (wsdair_test_0_1.xml)

2. Import property definition file:
   - Double click on the project name (wsdair_test_0_1)

- In the Name-Value table, click “Load Property” icon, and select the created property file at configure step 1. Now you should see that all the properties have proper values for your implementation.

3. Run the tests

To run a test suite, double click on the suite name and press the green arrow in the window that appears. Some tests assume that previous ones have run, so run the suites in the following order:

- 1_1_Optional_CoreResourceList – if applicable.
- 1_2_Mandatory.
- 1_4_Optional_GenericQuery – if applicable.
- 1_5_Optional_StoredProcedure – if applicable.
- 1_6_Optional_Modification – if applicable.
- 1_6_Optional_SQLCommunicationArea – if applicable.

Possible issues

An error that may occur if validating SOAP requests is:

```
line 9: Element not allowed:
DatasetFormatURI@http://www.ggf.org/namespaces/2005/12/WS-DAI in element
SQLExecuteRequest@http://www.ggf.org/namespaces/2005/12/WS-DAIR.
```

The WSDL and XML Schema allow this, as do certain WSDL and XML Schema validators. The authors conclude that this is a soapUI issue.

TEST16 – GetSQLRowsetFactory might fail with an error:

```
Schema Compliance - FAILED
-> line1: Missing operation [GetSQLRowsetFactory] in wsdl definition
```

As described in section 6.1.2 the authors are of the opinion that this is a soapUI issue rather than necessarily an implementation issue.