GT 4.0 Tech Preview: OGSA-DAI

GT 4.0 Tech Preview: OGSA-DAI

Table of Contents

Chapter 1. Key Concepts

See the key concepts[1] from Data Management.

[1] http://www.globus.org/toolkit/docs/4.0/data/key/

Chapter 2. GT 4.0 Release Notes: OGSA-DAI

1. Component Overview

OGSA-DAI provides a pure Java data service framework for accessing and integrating data resources - such as files, relational and XML databases - on to Grids. Towards this end, OGSA-DAI:

- Exposes intrinsic data resource capabilities - such as the ability to perform SQL queries on relational resources or evaluate XPath statements on XML collections - through web service based interfaces, thus allowing data resources to be easily incorporated as first class citizens in grids.
- Allows additional functionality to be implemented at the service - such as transformation of data coming out of a data resource - so as to avoid unnecessary data movement.
- Provides a compact way of handling multiple potential interactions with a service within a single request via an XML document, called a perform document, where data is pipelined between different set of activities that operate on a data stream coming out of, or going into, a data resource.
- Allows developers to easily add or extend functionality within OGSA-DAI. The perform document, and underlying framework, are extensible allowing additional functionality to be added, or existing functionality to be customised, and still operate within the same framework.
- Allows metadata about data and the data resources in which it is found to be queried via a service interface.
- Facilitates the provision of data integration capabilities from various sources to obtain the required information.

This WS-RF distribution of OGSA-DAI has been designed to work with the Globus Toolkit 4 implementation of WS-RF; a WS-I distribution is also available from the OGSA-DAI web site, which is designed to work with the UK OMII web services infrastructure releases. Finally, an OGSI-based distribution of OGSA-DAI, designed to work with the GT3.2 releases, is also available from the project web site. Note: these releases are not designed to inter-operate. More information about these and OGSA-DAI in general is available from the project website at http://www.ogsadai.org.uk.

2. Feature Summary

Features new in release GT 4.0:

- Access to data is provided via an "OGSA-DAI data service". For those that have previously used the OGSI version of OGSA-DAI, this service amalgamates the capabilities of the Grid Data Service Factory (GDSF) and Grid Data Service (GDS) services (the metadata and configuration roles of the GDSF and the metadata and perform document processing aspects of the GDS).
- Each data service exposes zero or more data service resources. A data service resource manages exposure of and interaction with a data resource. A single data service resource can be viewed as offering capabilities analogous to a single GDS in the OGSI version of OGSA-DAI.
- Allows multiple data resources to be accessed through a single service. Data service resource identifiers, available from the data service's WS-Addressing endpoint reference, allow a client to target a specific data service resource.
- A listResources() operation is provided for a data service to list all the identifiers of the data service resources exposed by that service.

- The data service resource identifiers returned by a data service can subsequently be used by a client to obtain metadata and other information, about the data service resources corresponding to that identifier.
- Access to data service resource metadata (such as database schemas, request status, etc.) is provided by an implementation of the WS-ResourceProperties specification. In particular support for using the QueryResourceProperties, GetResourceProperties and GetMultipleResourceProperties portTypes is provided.
- Access to version information about the OGSA-DAI Data Service is available through the getVersion() operation.
- A WSRF version of the OGSA-DAI GridDataTransport portType supports asynchronous data delivery between data services.

Other Supported Features (features that continue to be supported from previous versions):

- Perform documents.
- Extensible activity framework.
- Statement, delivery and transformation activities.

Deprecated Features

- The OGSA-DAI Client Toolkit has not been updated yet to support interaction with WSRF-enabled OGSA-DAI services. This will be made available in a future version.
- The OGSA-DAI registry service, DAISGR, has not been updated for WSRF. This will be addressed in a future version or provided via a third party component.
- The old graphical demonstrator no longer works with this version of OGSA-DAI.

3. Technology Dependencies

OGSA-DAI depends on the following GT components:

- Java WS Core

OGSA-DAI depends on the following 3rd party software:

- Java 1.4.0 (OGSA-DAI WSRF has been tested on this version of Java though may work with other Java 1.4.x flavours).
- Jakarta ANT 1.5 (see http://ant.apache.org).

Depending on the underlying data resource that OGSA-DAI is going to expose, you may need one or more of the following:

- For relational databases such as MySQL, PostgreSQL, SQLServer, Oracle and DB2, the corresponding JDBC drivers are required.
- For XML databases such as Xindice, the corresponding XMLDB drivers are required.
- To use a full text search engine against flat files, Jakarta Lucene is required.

4. Supported Platforms

Tested Platforms for OGSA-DAI

- Sun Solaris 8 (SPARC)
- Microsoft Windows 2000 (X86)
- Microsoft Windows XP (X86)
- Redhat Linux 9 (X86)

We have not tested on other platforms, but if you do and have problems, we would like to hear about it.

OGSA-DAI has been tested with the following databases:

- MySQL 3.2.3 and above
- PostgreSQL 7.4
- IBM DB2
- Oracle 9i
- MS SQLServer 2000
- Derby
- Xindice 1.0

5. Backward Compatibility Summary

Protocol changes since GT version 3.2:

- Not backwards compatible with the previous OGSI version.

API changes since GT version 3.2:

- None

Exception changes since GT version 3.2:

- None

Schema changes since GT version 3.2:

- WSDL changes to work with new Java WS Core.

6. For More Information

Click here[1] for more information about this component.

[1] index.html

Chapter 3. GT 4.0.1 Incremental Release Notes: OGSA-DAI

1. Introduction

These release notes are for the incremental release 4.0.1. It includes a summary of changes since 4.0.0, bug fixes since 4.0.0 and any known problems that still exist at the time of the 4.0.1 release. This page is in addition to the top-level 4.0.1 release notes at http://www.globus.org/toolkit/releasenotes/4.0.1.

For release notes about 4.0 (including feature summary, technology dependencies, etc) go to the OGSA-DAI 4.0 Release Notes[1].

2. Changes Summary

No OGSA-DAI *WSRF-specific* changes have been made for this release.

The following general OGSA-DAI changes have been made for this release:

- Core OGSA-DAI functionality (the engine and data resource connection classes) has been abstracted out into a service-level independent module. OGSA-DAI OGSI, WSI and WSRF share exactly the same core code. Core OGSA-DAI functionality and activities are now Globus-independent.
- The namespaces of OGSA-DAI configuration, Perform and Response documents have been completely changed:
  - Perform and response documents are now declared within the http://ogsadai.org.uk/namespaces/2005/03/types namespace.
  - Data resource and activity configuration documents and roleMap documents are now declared within the http://ogsadai.org.uk/namespaces/2005/03/config namespace.
- The deliverToStream activity now delivers to a URL of form http://HOST:PORT/WEBAPP/servlet/DeliverToStreamServlet?url=http://HOST:PORT/WEBAPP/ogsadai/DataService&streamId=NAME - gsh has been renamed to url.
- Control flow constraints can now be expressed within Perform documents. This means you can ensure that one activity will not start until another has completed e.g. ensuring that a new table is not filled with data until it has been created.
- Server-side exceptions outwith client control and authorisation failures have been refactored. The problems are logged in detail server-side and tagged with a unique ID. This ID, with a request to contract the service deployer, is returned to the client.
- All OGSA-DAI data services now provide a getVersion operation which returns the OGSA-DAI flavour and version number.
- GUI installation tools now allow required JARs to be selected via a file browser.
- GUI installation tools now render yes/no options as check boxes.

[1] http://www.globus.org/toolkit/docs/4.0/techpreview/ogsadai/OGSA-DAI_Release_Notes.html

3. Bug Fixes

No OGSA-DAI *WSRF-specific* bug fixes have been made for this release.

The following general OGSA-DAI bug fixes have been made for this release:

- Client toolkit now raises an authorisation error if roleMapping fails server-side.
- sqlQueryStatement activity can now retrieve BLOB or IMAGE fields from databases.
- GridFTP-related delivery activities have had their JavaDoc, and the associated client toolkit JavaDoc revised to include details about how security credentials are used.
- WebRowSet processing code for SQL activities no longer has option to make results human-readable.
- Perform document schema property provided by OGSA-DAI service now includes required import statements.
- Response documents constructed in-code now meet the Response document XML Schema.
- User doc front page now includes link to JavaDoc in binary distributions.
- Errors in stringTokenizerActivity JavaDoc have been corrected.
- Various user doc revisions and corrections have been made.
- GUI installation tools now no longer occasionally freeze.
- Installation tools now no longer need to edit the Tomcat server configuration file so problems with its location or format no longer arise.

4. Known Problems

The following problems are known to exist for OGSA-DAI at the time of the 4.0.1 release:

- *Problems retrieving BLOBS*

  When trying to retrieve BLOBs from a relational database you will get zero rows returned (or an error). The following error message will be logged:

  ```
  Base64Provider has not been set up with a Base64 implementation
  ```

  This will be fixed in the next release. In the meantime if you have the source version of the release, you can correct this bug by changing two files and rebuilding OGSA-DAI. See BLOB Error Workaround[2] for details.

- *DeliverToStream activity is broken.*

  The DeliverToStream activity has a bug in the WSRF implementation. The service URL passed to the servlet is 'null' rather than the correct URL.

  Thus where the documentation says the data will be available from a URL of the form:

  ```
  http://host:port/ogsa/servlet/DeliverToStreamServlet?url=ht-
  tp://host:port/ogsa/ogsadai/DataService&streamId=NAME
  ```

[2] http://www.globus.org/toolkit/docs/4.0/techpreview/ogsadai/OGSA_DAI_BLOB_Error_Workaround.html

It will actually be available from a URL in the form:

```
http://host:port/ogsa/servlet/DeliverToStreamServlet?url=null&streamId=NAME
```

This will be fixed for the next release.

- *Client toolkit asynchronous result set broken.*

  The ResultSet object returned by the OutputStreamActivity class' getResultSet methods do not work for WSRF 1.0. This is because the code attempts to use the getNBlocks service operation which is not implemented in WSRF 1.0. This will be fixed in our next release.

  There are two possible work arounds to this problem:

  1. Use synchronous delivery. The ResultSet returned by the SQLQuery activity's getResultSet() method does work.

  2. Alter the client toolkit code to use the getBlock() service operation. Unfortunately this option will be very slow as only one row will be retrieved at a time.

5. For More Information

Click here[3] for more information about this component.

[3] index.html

Chapter 4. GT 4.0.2 Incremental Release Notes: OGSA-DAI

1. Introduction

These release notes are for the incremental release 4.0.2. It includes a summary of changes since 4.0.1, bug fixes since 4.0.1 and any known problems that still exist at the time of the 4.0.2 release. This page is in addition to the top-level 4.0.2 release notes at http://www.globus.org/toolkit/releasenotes/4.0.2.

For release notes about 4.0 (including feature summary, technology dependencies, etc) go to the OGSA-DAI 4.0 Release Notes[1].

2. Changes Summary

This section includes information for versions 2.0 and 2.1 of OGSA-DAI WSRF which is distributed with GT 4.0.2 (the GT4.0.1 release contained version 1.0 of the OGSA-DAI WSRF distribution).

- The documentation bundled with the OGSA-DAI distribution explicitly mentions GT4.0.1. Everything will work fine with GT4.0.2 too.
- The canonical documentation for the OGSA-DAI 2.1 release is contained in the release documentation so please use this for installation and usage.

2.1. Release 2.1

OGSA-DAI WSRF 2.1 is a WSRF-compliant version of OGSA-DAI and runs upon Globus Toolkit 4.0.2. It is not compatible with Globus Toolkit 3. The source release should still be compatible with Globus Toolkit 4.0. The main changes between this and the previous release are as follows:

OGSA-DAI WSRF 2.1 is almost identical to OGSA-DAI WSRF 2.0 which was released just one week earlier. This release was motivated by:

- A need to allow data resource accessors to be notified whenever a session is terminated, so that session-related resource properties can be removed.
- A bug fix to ensure that a single block of data can flow through a pipeline containing an inputStream activity and an outputStream activity.

If you have OGSA-DAI WSRF 2.0 and were not dependent on or concerned by these then there is no requirement for you to upgrade.

The additions / changes were as follows:

- In core/src/java/:
  - Changed uk.org.ogsadai.activity.ActivityRequest
  - Changed uk.org.ogsadai.activity.CallThroughExternalPipe

[1] http://www.globus.org/toolkit/docs/4.0/techpreview/ogsadai/OGSA-DAI_Release_Notes.html

- Changed uk.org.ogsadai.activity.CallThroughPipe
- Changed uk.org.ogsadai.service.resource.DataServiceResource
- Changed uk.org.ogsadai.sessions.SessionManagerFactory
- Changed uk.org.ogsadai.sessions.SessionConfigurationLoader
- Changed uk.org.ogsadai.sessions.impl.SessionConfiguration
- Changed uk.org.ogsadai.sessions.impl.TransientSessionManager
- Changed uk.org.ogsadai.sessions.impl.TransientSessionManagerFactory
- Added uk.org.ogsadai.dataresource.DataResourceListener
- Added uk.org.ogsadai.dataresource.DataResourceEvent
- Added uk.org.ogsadai.dataresource.DataResourceEventDispatcher
- Added uk.org.ogsadai.dataresource.TestDataResource
- Added uk.org.ogsadai.sessions.impl.TransientSessionConfiguration

- In doc:
  - Changed tutorials/dataresource/howTo.html
  - Changed clients/clienttoolkit/ObtainingMetaData.html
  - Changed misc/HowToUpgrade.html
  - Changed misc/KnownProblems.html
  - Changed clients/clienttoolkit/index.html
  - Changed clients/clienttoolkit/SecureServices.html

2.2. Release 2.0

OGSA-DAI WSRF 2.0 is a WSRF-compliant version of OGSA-DAI and runs upon Globus Toolkit 4.0.1. It is not compatible with Globus Toolkit 3. The source release should still be compatible with Globus Toolkit 4.0.

The major features of this release are as follows:

- OGSA-DAI WSRF 2.0 is compatible with Globus Toolkit 4.0.2 and the should still work with Globus Toolkit 4.0 (if you have the OGSA-DAI WSRF 2.0 binary distribution you'll need to get the source distribution and compile against Globus Toolkit 4.0).
- Data service resources support concurrent request execution and queueing of requests.
- All data service resources exposed by a specific service share the same settings for the number of concurrent requests that can be executed and the number that can be queued. These are specified at service deployment time.
- Sessions are now supported. Multiple perform documents can be submitted which operate within the context of the same session. State can be stored in session attributes. Data transport operations now operate within the scope of a session.

- Data service accessors have been provided. These manage access to a data resource on behalf of a data service resource. They each have their own specific configuration files.
- Configurable data services provide an undeploy operation so that a service can be instructed to no longer expose a specific data service resource. This is reflected in the client toolkit.
- There have been numerous performance improvements most notably in the refactoring of our java.sql.ResultSet to WebRowSet converters.
- Client and server-side now support transport-level security between clients and services.
- Client and server-side now support message-level security between clients and services.
- Services can be configured to register themselves with a Globus MDS Index service running in the same container.
- The OGSA-DAI data browser is now compatible with both OGSA-DAI WSRF 2.0 and OGSA-DAI WSI 2.0 services.

The main changes between this and the previous release are as follows:

- The date in all OGSA-DAI XML Schema namespaces have been updated i.e.: `http://ogsa-dai.org.uk/namespaces/2005/03/...` has been updated to: `http://ogsa-dai.org.uk/namespaces/2005/10/...`
- Data service resources now have a:
  - data resource class configuration file which specifies the data resource accessor to use.
  - session configuration file which specifies the maximum and default timeouts for sessions.
- Files data service resources no longer have associated role map documents.
- Data resource configuration files dataResource element no longer has the implementation attribute.
- Data resource configuration files for file data resources now have a new XML Schema. This is in schema/ogsa-dai/xsd/data_resource_config_files.xsd. The main changes are:
  - The driver element has been replaced by a rootDirectory element.
  - The rolemap is no longer needed.
  - The productInfo is no longer needed.
- Service interfaces have been completely refactored.
  - A terminate operation has been added. This is a placeholder and currently does nothing. It will be used to support request termination in future releases.
  - The updateRoleMaps operation for configurable data services no longer exists.
  - All OGSA-DAI-specific service operations throw one of five new fault types. The client toolkit has been completely refactored to reflect these fault types as Java exceptions.
- SQL activities no longer implicitly convert java.sql.ResultSets to WebRowSet. A new activity - sqlResultsToXML - has been provided to make this explicit. Whereas in previous releases you would use a sqlQueryStatement activity, for example, you now use a sqlQueryStatement activity connected to a sqlResultsToXML activity.
- Encrypted role map files are no longer supported and ogsadai-startup-tomcat.jar is no longer provided.

- OGSA-DAI tutorial examples are compiled as part of a source distribution build and are distributed within ogsadai-examples.jar.
- The Client Toolkit APIs have been modified to support sessions and concurrency. Some methods such as the poll method are now parameterised with a SessionID object. A user may attach session requirements to an ActivityRequest before performing it.
- The request status resource property available in previous versions is no longer available. Instead a session request status resource property is created dynamically whenever a new session is created. This indicates the processing status of the request that is currently joined to the session.

3. Bug Fixes

No OGSA-DAI *WSRF-specific* bug fixes have been made for this release.

4. Known Problems

The following problems are known to exist for OGSA-DAI at the time of the 4.0.2 release and will be addressed in the future. If you have any others that you feel should be added to this list then please let us know.

You may also want to consult the platform-specific FAQ[2] and general FAQ[3]. Additional information may be posted in the advisories page[4] which report any problems since the release, or the OGSA-DAI bugzilla[5] which may also have information on any problems with the current release.

- Performance:
  - Accessing the databaseSchema property for DB2/Cloudscape can sometimes retrieve unrelated meta-data and also cause OGSA-DAI to crash.
  - Accessing the databaseSchema property for Oracle causes a server-side java.lang.OutOfMemoryError.
  - Attempting to return a very large data set in a Response document can cause a server-side java.lang.OutOfMemoryError. This can be avoided by submitting an asynchronous request (one that uses an outputStream) and retrieving the results using a data service's data transport operations.
  - Queries of tables for millions of small rows can eventually cause a server-side java.lang.OutOfMemoryError even if using stream activities and data transport operations.
  - The fileWritingActivity attempts to read the whole file in a oner and so could cause a server-side java.lang.OutOfMemoryError for very large files.
  - Some databases, such as dBASE IV database on Microsoft Windows 2000, allow unusual characters, such as 0xC to be contained in certain field types. When field values containing this character, OGSA-DAI encodes them into invalid XML that subsequently causes Xerces-dependent components to raise an org.xml.sax.SAX-ParseException. Activities that have two outputs can sometimes cause java.lang.OutOfMemoryErrors to occur server-side if manipulating large amounts of data. This can arise when using the GZIP activities for example.
    - Such activities are driven by one of the two connected activities and the other activity can accumulate data without processing it. So, for example for the GZIP activities OGSA-DAI focuses on the GZIP meta-data

[2] http://www.ogsadai.org.uk/documentation/ogsadai-wsrf-2.1/doc/wsrf/FAQ.html
[3] http://www.ogsadai.org.uk/documentation/ogsadai-wsrf-2.1/doc/misc/FAQ.html
[4] http://www.ogsadai.org.uk/support/advisories
[5] http://bugs.ogsadai.org.uk/

output steam and data only flows from this stream when a file has been completely unzipped. Therefore for large files the whole contents of the file flows into the activity and is buffered without being sent on.

    - OGSA-DAI's GZIP activities will warn if memory usage is close to maximum and throw an OGSA-DAI exception server-side so that the web services container does not crash.

- Error and exception handling:
    - The failure of a single activity in a request can cause the status of all activities to be set to ERROR, even those that are not connected and succeeded.
    - The data transport activities can gulp exceptions. For example if an sqlQueryStatement with a incorrectly formed SQL query statment is connected to an outputStream activity then attempting to pull data from the outputStream using the data transport operations results in no data - no indication of an error is given to the client.
    - The uk.org.ogsadai.exception.DAILogger methods can sometimes log the wrong line numbers. Searching the logs will usually reveal where the problem actually arose.
- Security:
    - Encrypted role map files are currently unsupported.
    - If a client does not provide a certificate then deliverFromGridFTP and deliverToGridFTP in a request will fail.
- File activities:
    - The fileAccessActivity encodes text line-by-line if Base 64 encoding is being used. This may give different results from encoding a whole file in a oner.
    - In fileWritingActivity, EOF means "end of file" when perLine has value false but "end of line" when perLine has value true.
    - In fileWritingActivity only strings from its input stream are handled. Other block types e.g. bytes are ignored.
- General:
    - Note that when deploying new data service resources dynamically via configurable data service this only works if the JARs required by the data service resource (e.g. database driver JARs) are already within the web service container's library directories. If this is not the case then the container has to be restarted.
    - The data service terminate operation currently does nothing. It is intended as a placeholder for future development.
    - Meta-data from a database, returned in the databaseSchema property, can be case-sensitive depending upon the database. For example MySQL might return a table called littleblackbook while DB2 returns LITTLEBLACKBOOK.
    - The uk.org.ogsadai.common.BinaryLob class contains unimplemented methods which throw java.lang UnsupportedMethodException.
    - Conversion of java.sql.ResultSet to XML WebRowSet returns empty key-column and map properties in the properties element.
    - deliverFromGridFTP and deliverToGridFTP do not allow the setting of certain GridFTP parameters.
- There is no support in the client toolkit for certain activities. These include:

- directoryAccessActivity
- fileAccessActivity
- fileManipulationActivity
- deliverFromFTP
- deliverToFTP
- gzipDecompression

- The deliverToStream activity only works if services are deployed in Tomcat.

5. For More Information

Click here[6] for more information about this component.

[6] index.html

Chapter 5. GT 4.0.3 Incremental Release Notes: OGSA-DAI

1. Introduction

These release notes are for the incremental release 4.0.3. It includes a summary of changes since 4.0.2, bug fixes since 4.0.2 and any known problems that still exist at the time of the 4.0.3 release. This page is in addition to the top-level 4.0.3 release notes at http://www.globus.org/toolkit/releasenotes/4.0.3.

For release notes about 4.0 (including feature summary, technology dependencies, etc) go to the OGSA-DAI 4.0 Release Notes[1].

2. Changes Summary

The only thing that has changed for OGSA-DAI WSRF since GT 4.0.2 is that the new globus jars that we bundle with our distribution have been updated to reflect those contained in the 4.0.3 release.

3. Bug Fixes

No OGSA-DAI *WSRF-specific* bug fixes have been made for this release.

4. Known Problems

The following problems are known to exist for OGSA-DAI at the time of the 4.0.3 release and will be addressed in the future. If you have any others that you feel should be added to this list then please let us know. You may also want to consult the platform-specific FAQ[2] and general FAQ[3]. Additional information may be posted in the advisories page[4] which report any problems since the release, or the OGSA-DAI bugzilla[5] which may also have information on any problems with the current release.

- Performance:
    - Accessing the databaseSchema property for DB2/Cloudscape can sometimes retrieve unrelated meta-data and also cause OGSA-DAI to crash.
    - Accessing the databaseSchema property for Oracle causes a server-side java.lang.OutOfMemoryError.
    - Attempting to return a very large data set in a Response document can cause a server-side java.lang.OutOfMemoryError. This can be avoided by submitting an asynchronous request (one that uses an outputStream) and retrieving the results using a data service's data transport operations.
    - Queries of tables for millions of small rows can eventually cause a server-side java.lang.OutOfMemoryError even if using stream activities and data transport operations.

[1] http://www.globus.org/toolkit/docs/4.0/techpreview/ogsadai/OGSA-DAI_Release_Notes.html
[2] http://www.ogsadai.org.uk/documentation/ogsadai-wsrf-2.1/doc/wsrf/FAQ.html
[3] http://www.ogsadai.org.uk/documentation/ogsadai-wsrf-2.1/doc/misc/FAQ.html
[4] http://www.ogsadai.org.uk/support/advisories
[5] http://bugs.ogsadai.org.uk/

    - The fileWritingActivity attempts to read the whole file in a oner and so could cause a server-side java.lang.OutOfMemoryError for very large files.
    - Some databases, such as dBASE IV database on Microsoft Windows 2000, allow unusual characters, such as 0xC to be contained in certain field types. When field values containing this character, OGSA-DAI encodes them into invalid XML that subsequently causes Xerces-dependent components to raise an org.xml.sax.SAX-ParseException. Activities that have two outputs can sometimes cause java.lang.OutOfMemoryErrors to occur server-side if manipulating large amounts of data. This can arise when using the GZIP activities for example.
        - Such activities are driven by one of the two connected activities and the other activity can accumulate data without processing it. So, for example for the GZIP activities OGSA-DAI focuses on the GZIP meta-data output steam and data only flows from this stream when a file has been completely unzipped. Therefore for large files the whole contents of the file flows into the activity and is buffered without being sent on.
        - OGSA-DAI's GZIP activities will warn if memory usage is close to maximum and throw an OGSA-DAI exception server-side so that the web services container does not crash.

- Error and exception handling:
    - The failure of a single activity in a request can cause the status of all activities to be set to ERROR, even those that are not connected and succeeded.
    - The data transport activities can gulp exceptions. For example if an sqlQueryStatement with a incorrectly formed SQL query statment is connected to an outputStream activity then attempting to pull data from the outputStream using the data transport operations results in no data - no indication of an error is given to the client.
    - The uk.org.ogsadai.exception.DAILogger methods can sometimes log the wrong line numbers. Searching the logs will usually reveal where the problem actually arose.
- Security:
    - Encrypted role map files are currently unsupported.
    - If a client does not provide a certificate then deliverFromGridFTP and deliverToGridFTP in a request will fail.
- File activities:
    - The fileAccessActivity encodes text line-by-line if Base 64 encoding is being used. This may give different results from encoding a whole file in a oner.
    - In fileWritingActivity, EOF means "end of file" when perLine has value false but "end of line" when perLine has value true.
    - In fileWritingActivity only strings from its input stream are handled. Other block types e.g. bytes are ignored.
- General:
    - Note that when deploying new data service resources dynamically via configurable data service this only works if the JARs required by the data service resource (e.g. database driver JARs) are already within the web service container's library directories. If this is not the case then the container has to be restarted.
    - The data service terminate operation currently does nothing. It is intended as a placeholder for future development.

    - Meta-data from a database, returned in the databaseSchema property, can be case-sensitive depending upon the database. For example MySQL might return a table called littleblackbook while DB2 returns LITTLE-BLACKBOOK.
    - The uk.org.ogsadai.common.BinaryLob class contains unimplemented methods which throw java.lang UnsupportedMethodException.
    - Conversion of java.sql.ResultSet to XML WebRowSet returns empty key-column and map properties in the properties element.
    - deliverFromGridFTP and deliverToGridFTP do not allow the setting of certain GridFTP parameters.
- There is no support in the client toolkit for certain activities. These include:
    - directoryAccessActivity
    - fileAccessActivity
    - fileManipulationActivity
    - deliverFromFTP
    - deliverToFTP
    - gzipDecompression
- The deliverToStream activity only works if services are deployed in Tomcat.

5. For More Information

Click here[6] for more information about this component.

[6] index.html

Chapter 6. GT 4.0.4 Incremental Release Notes: OGSA-DAI

1. Introduction

These release notes are for the incremental release 4.0.4. It includes a summary of changes since 4.0.3, bug fixes since 4.0.3 and any known problems that still exist at the time of the 4.0.4 release. This page is in addition to the top-level 4.0.4 release notes at http://www.globus.org/toolkit/releasenotes/4.0.4.

For release notes about 4.0 (including feature summary, technology dependencies, etc) go to the OGSA-DAI 4.0 Release Notes[1].

2. Changes Summary

The only thing that has changed for OGSA-DAI WSRF since GT 4.0.3 is that the new globus jars that we bundle with our distribution have been updated to reflect those contained in the 4.0.4 release.

3. Bug Fixes

No OGSA-DAI *WSRF-specific* bug fixes have been made for this release.

4. Known Problems

The following problems are known to exist for OGSA-DAI at the time of the 4.0.4 release and will be addressed in the future. If you have any others that you feel should be added to this list then please let us know.

You may also want to consult the platform-specific FAQ[2] and general FAQ[3]. Additional information may be posted in the advisories page[4] which report any problems since the release, or the OGSA-DAI bugzilla[5] which may also have information on any problems with the current release.

- Performance:
    - Accessing the databaseSchema property for DB2/Cloudscape can sometimes retrieve unrelated meta-data and also cause OGSA-DAI to crash.
    - Accessing the databaseSchema property for Oracle causes a server-side java.lang.OutOfMemoryError.
    - Attempting to return a very large data set in a Response document can cause a server-side java.lang.OutOfMemoryError. This can be avoided by submitting an asynchronous request (one that uses an outputStream) and retrieving the results using a data service's data transport operations.
    - Queries of tables for millions of small rows can eventually cause a server-side java.lang.OutOfMemoryError even if using stream activities and data transport operations.

[1] http://www.globus.org/toolkit/docs/4.0/techpreview/ogsadai/OGSA-DAI_Release_Notes.html
[2] http://www.ogsadai.org.uk/documentation/ogsadai-wsrf-2.1/doc/wsrf/FAQ.html
[3] http://www.ogsadai.org.uk/documentation/ogsadai-wsrf-2.1/doc/misc/FAQ.html
[4] http://www.ogsadai.org.uk/support/advisories
[5] http://bugs.ogsadai.org.uk/

    - The fileWritingActivity attempts to read the whole file in a oner and so could cause a server-side java.lang.OutOfMemoryError for very large files.
    - Some databases, such as dBASE IV database on Microsoft Windows 2000, allow unusual characters, such as 0xC to be contained in certain field types. When field values containing this character, OGSA-DAI encodes them into invalid XML that subsequently causes Xerces-dependent components to raise an org.xml.sax.SAX-ParseException. Activities that have two outputs can sometimes cause java.lang.OutOfMemoryErrors to occur server-side if manipulating large amounts of data. This can arise when using the GZIP activities for example.
        - Such activities are driven by one of the two connected activities and the other activity can accumulate data without processing it. So, for example for the GZIP activities OGSA-DAI focuses on the GZIP meta-data output steam and data only flows from this stream when a file has been completely unzipped. Therefore for large files the whole contents of the file flows into the activity and is buffered without being sent on.
        - OGSA-DAI's GZIP activities will warn if memory usage is close to maximum and throw an OGSA-DAI exception server-side so that the web services container does not crash.

- Error and exception handling:
    - The failure of a single activity in a request can cause the status of all activities to be set to ERROR, even those that are not connected and succeeded.
    - The data transport activities can gulp exceptions. For example if an sqlQueryStatement with a incorrectly formed SQL query statment is connected to an outputStream activity then attempting to pull data from the outputStream using the data transport operations results in no data - no indication of an error is given to the client.
    - The uk.org.ogsadai.exception.DAILogger methods can sometimes log the wrong line numbers. Searching the logs will usually reveal where the problem actually arose.
- Security:
    - Encrypted role map files are currently unsupported.
    - If a client does not provide a certificate then deliverFromGridFTP and deliverToGridFTP in a request will fail.
- File activities:
    - The fileAccessActivity encodes text line-by-line if Base 64 encoding is being used. This may give different results from encoding a whole file in a oner.
    - In fileWritingActivity, EOF means "end of file" when perLine has value false but "end of line" when perLine has value true.
    - In fileWritingActivity only strings from its input stream are handled. Other block types e.g. bytes are ignored.
- General:
    - Note that when deploying new data service resources dynamically via configurable data service this only works if the JARs required by the data service resource (e.g. database driver JARs) are already within the web service container's library directories. If this is not the case then the container has to be restarted.
    - The data service terminate operation currently does nothing. It is intended as a placeholder for future development.

    - Meta-data from a database, returned in the databaseSchema property, can be case-sensitive depending upon the database. For example MySQL might return a table called littleblackbook while DB2 returns LITTLE-BLACKBOOK.
    - The uk.org.ogsadai.common.BinaryLob class contains unimplemented methods which throw java.lang UnsupportedMethodException.
    - Conversion of java.sql.ResultSet to XML WebRowSet returns empty key-column and map properties in the properties element.
    - deliverFromGridFTP and deliverToGridFTP do not allow the setting of certain GridFTP parameters.
- There is no support in the client toolkit for certain activities. These include:
    - directoryAccessActivity
    - fileAccessActivity
    - fileManipulationActivity
    - deliverFromFTP
    - deliverToFTP
    - gzipDecompression
- The deliverToStream activity only works if services are deployed in Tomcat.

5. For More Information

Click here[6] for more information about this component.

[6] index.html

Chapter 7. GT 4.0 OGSA-DAI (Tech Preview): System Administrator's Guide

1. Introduction

This guide contains advanced configuration information for system administrators working with OGSA-DAI. It provides references to information on procedures typically performed by system administrators, including installing, configuring, deploying, and testing the installation.

Important

This information is in addition to the basic Globus Toolkit prerequisite, overview, installation, security configuration instructions in the GT 4.0 System Administrator's Guide[1]. Read through this guide before continuing!

2. Building and installing

OGSA-DAI is not built and installed as part of the standard Globus Toolkit installation. You will find the installation bundle in the contributions directory. You may find a binary and/or a source distribution of the OGSA-DAI software. This documentation will outline how to build a binary distribution from the source. You will need to install the Java WS-Core from the Globus distribution.

2.1. Prerequisites

The deployment and use of the OGSA-DAI WSRF distribution will be easier if you are already familiar with:

- OGSA-DAI concepts and terms. Reading the user's guide included in the OGSA-DAI distribution should provide you with this information.
- WS-Addressing concepts and terms. An introduction to WS-Addressing is available at http://www-106.ibm.com/developerworks/library/specification/ws-add.
- Web Services Resource Framework (WSRF) concepts and terms. The WSRF specifications can be found at http://www.globus.org/wsrf.

2.1.1. Prerequisite Software

To use OGSA-DAI WSRF you will need the following software:

- Java 1.4.0: OGSA-DAI WSRF has been tested on this version of Java though may work with other Java 1.4.x flavours.
- Jakarta ANT 1.5: See http://ant.apache.org.
- Globus Toolkit Java WS Core
  - This provides a Java-based implementation of WSRF and is available at: http://www.globus.org/toolkit/downloads/4.0/.

[1] ../../admin/docbook/

- Information on Globus Toolkit 4.0 can also be found at: http://www.globus.org/toolkit/docs/4.0/GT4Facts.

3. Building OGSA-DAI WSRF

This section describes how to build OGSA-DAI WSRF. If you are using the binary distribution then you can move to the deploying OGSA-DAI WSRF section.

3.1. Insert the Prerequisite JAR Files

First, download the prerequisite JAR files (listed above):

- jakarta-oro-2.0.7.jar
- lucene-1.4-final.jar
- xmldb.jar

and put these into the OGSA-DAI distribution's lib directory.

3.2. Compile GT4 core

Now, you need to compile Globus Toolkit 4 Web Services Core. The GT4 instructions tell you how to do this.

3.3. Build OGSA-DAI WSRF

To build OGSA-DAI WSRF:

- Set a GLOBUS_LOCATION environment variable to point to the location of your GT4 distribution. For example, under UNIX, enter:

```
$ export GLOBUS_LOCATION=/path/to/Globus/directory
```

- Run the following from within the OGSA-DAI distribution directory:

```
$ ant createBinaryDistribution
```

  Progress messages will appear and if there are any problems - for example missing JARs - then you will be notified.

- When complete, you will have a ZIPped, TARred binary distribution within your OGSA-DAI source distribution directory. You will also have a binary directory containing the binary release.

OGSA-DAI WSRF will then be built. You should now change into the binary directory. Note that before you can use OGSA-DAI you must configure it. This is covered in the deployment section.

4. Configuring

This information is in addition to the basic configuration instructions in the GT4.0 System Administrator's Guide[2].

Please consult the Section 5, “Deploying” section in the Admin Guide for information on configuring OGSA-DAI services.

[2] http://www.globus.org/toolkit/docs/4.0/admin/docbook/

5. Deploying

This section describes how to install OGSA-DAI WSRF onto the GT4 container or Apache Tomcat.

This section assumes that you have built the WSRF version of OGSA-DAI from source or are using a binary distribution. Distributions of OGSA-DAI are bundled with the Globus Toolkit or are available from the project website www.ogsa-dai.org.uk[3]. Please consult the bundled documentation in any of these distributions for more details about OGSA-DAI.

The OGSA-DAI WSRF binary distribution can be deployed onto the GT4 container or Apache Tomcat:

- Either via the Command-line
- Or using a GUI

5.1. Deploying OGSA-DAI WSRF via the Command-line

5.1.1. Insert the Prerequisite JAR Files

First, download the prerequisite JAR files (listed in the building section):

- jakarta-oro-2.0.7.jar
- lucene-1.4-final.jar
- xmldb.jar

and put these into the OGSA-DAI binary distribution lib directory. (IF you build your binary from a source distribution you will already have these JARs and they will have been inserted into the binary lib directory as part of the build).

5.1.2. Deploying OGSA-DAI WSRF onto GT4

If you wish to deploy OGSA-DAI WSRF onto the GT4 Web services container then keep reading. If however you want to deploy OGSA-DAI WSRF onto Apache Tomcat then skip to the next section.

To deploy OGSA-DAI WSRF onto GT4:

- Set a GLOBUS_LOCATION environment variable to point to the location of your GT4 distribution. For example, under UNIX, enter:

```
$ export GLOBUS_LOCATION=/path/to/Globus/directory
```

- Run the following from within the OGSA-DAI binary distribution directory:

```
$ ant deployGT4
```

OGSA-DAI WSRF will then be deployed.

5.1.3. Deploying OGSA-DAI WSRF onto Tomcat

Before you can deploy OGSA-DAI WSRF onto Tomcat you will need to deploy GT4 onto Tomcat. You can do this as follows, if you have not already done so:

[3] http://www.ogsadai.org.uk

- Set a GLOBUS_LOCATION environment variable to point to the location of your GT4 distribution. For example, under UNIX, enter:

```
$ export GLOBUS_LOCATION=/path/to/Globus/directory
```

- Set a CATALINA_HOME environment variable to point to the location of Tomcat. For example, under UNIX, enter:

```
$ export CATALINA_HOME=/path/to/Tomcat/directory
```

- Run the following commands:

```
$ cd path/to/Globus/directory
$ ant -f share/globus_wsrf_common/tomcat/tomcat.xml deployTomcat
-Dtomcat.dir=/path/to/Tomcat
$ cd path/to/OGSA-DAI/binary/directory
```

- For more information see the GT4 installation instructions.

Now, to deploy OGSA-DAI WSRF onto Tomcat, run the following from within the OGSA-DAI binary distribution directory:

```
$ ant deployTomcat
```

OGSA-DAI WSRF will then be deployed to Tomcat.

Note

As an alternative to setting CATALINA_HOME you can specify the Tomcat location at the command-line as follows:

```
$ ant deployTomcat -Dtomcat.dir=/path/to/Tomcat/directory
```

5.2. Deploying OGSA-DAI WSRF using a GUI

5.2.1. Deploying OGSA-DAI WSRF onto GT4

If you wish to deploy OGSA-DAI WSRF onto the GT4 Web services container then keep reading. If however you want to deploy OGSA-DAI WSRF onto Apache Tomcat then skip to the next section.

To deploy OGSA-DAI WSRF onto GT4:

- Run the following from within the OGSA-DAI binary distribution directory:

```
$ ant guiDeployGT4
```

- A GUI will appear and you can select your GT4 distribution by pressing the Select... button. If GLOBUS_LOCATION is set then this will be offered as the default selection.

- When you are happy with the Path, press Next.

- You now have to select the prerequisite JAR files (listed in the building section):
  - jakarta-oro-2.0.7.jar
  - lucene-1.4-final.jar
  - xmldb.jar
- Using the file browser on the left select a JAR file. Press Add to add it to the list of current JARs on the right.
- If you make a mistake then select the JAR on the right and press Remove to remove it from the list.
- When you are happy with your selections, press Next.
- You will now be requested to press Next so that the tool can deploy OGSA-DAI WSRF. If any problems occur then the tool will notify you of these.
- When the deployment has completed a completion window will be displayed. Press Finish to exit the tool.

5.2.2. Deploying OGSA-DAI WSRF onto Tomcat

Before you can deploy OGSA-DAI WSRF onto Tomcat you will need to deploy GT4 onto Tomcat. You can do this as follows, if you have not already done so:

- Set a GLOBUS_LOCATION environment variable to point to the location of your GT4 distribution. For example, under UNIX, enter:

```
$ export GLOBUS_LOCATION=/path/to/Globus/directory
```

- Set a CATALINA_HOME environment variable to point to the location of Tomcat. For example, under UNIX, enter:

```
$ export CATALINA_HOME=/path/to/Tomcat/directory
```

- Run the following commands:

```
$ cd path/to/Globus/directory
$ ant -f share/globus_wsrf_common/tomcat/tomcat.xml deployTomcat
-Dtomcat.dir=$CATALINA_HOME
$ cd path/to/OGSA-DAI/binary/directory
```

- For more information see the GT4 installation instructions.

Now, to deploy OGSA-DAI WSRF onto Tomcat:

- Run the following from within the OGSA-DAI binary distribution directory:

```
$ ant guiDeployTomcat
```

- A GUI will appear and you can select your Tomcat distribution by pressing the Select... button. If CATALINA_HOME is set then this will be offered as the default selection.
- When you are happy with the Path press Next.

- You now have to select the prerequisite JAR files (listed in admin building section):
  - jakarta-oro-2.0.7.jar
  - lucene-1.4-final.jar
  - xmldb.jar
- Using the file browser on the left select a JAR file. Press Add to add it to the list of current JARs on the right.
- If you make a mistake then select the JAR on the right and press Remove to remove it from the list.
- When you are happy with your selections, press Next.
- You will now be requested to press Next so that the tool can deploy OGSA-DAI WSRF. If any problems occur then the tool will notify you of these.
- When the deployment has completed a completion window will be displayed. Press Finish to exit the tool.

5.3. Exposing Data Resources using OGSA-DAI WSRF

To expose a data resource via an OGSA-DAI WSRF data service is a three-step process:

1. Deploy an OGSA-DAI data service. This data service initially exposes 0 data service resources.
2. Deploy a data service resource. The data service resource contains information about a data resource and the activities clients can perform on that data resource via a data service.
3. Add the data service resource to a data service. This instructs the data service to expose the data service resource and so allows clients to interact with the data service resource - thereby interacting with a data resource.

For those who want a quick introduction to OGSA-DAI WSRF we recommend:

1. Deploy a new data service as described here.
2. Startup your Web services container.
3. Test the data service resource using the ListResources client and ensuring that the service is available and exposes 0 data service resources. The ListResources client is described here.
4. Shutdown your Web services container.
5. Deploy a new data data service resource as described here.
6. Add the data service resource to the data service as described here.
7. Startup your Web services container.
8. Submit a perform document to the data service resource exposed by your data service using the End-to-end client. The End-to-end client is described here.

5.4. Deploying Data Services

This section describes how to deploy an OGSA-DAI data service which initially will expose 0 data service resources.

You can:

- Deploy data services using the command-line.
- Deploy data services using a GUI.
- Check that a data service deployed correctly.

If you have already deployed some data service resources (as described in the next section) you can add these to the data service as described on a following section.

5.4.1. Deploying Data Services Using the Command-line

To deploy a data service under Tomcat:

- Run the following command from within the OGSA-DAI WSRF binary distribution directory:

```
$ ant deployDataServiceTomcat -Dtomcat.dir=/path/to/Tomcat/directory
-Ddai.service.path=service/path
```

  - `dai.service.path` specifies the local URL of the service. For example `-Ddai.service.path=ogsadai/DataService` specifies a data service whose URL will be `http://HOST:PORT/wsrf/services/ogsadai/DataService`.
  - `tomcat.dir` specifies the location of Tomcat. If this argument is omitted then the Tomcat location specified within CATALINA_HOME is used.
  - Providing the flag `-Ddai.service.config=y` requests that the data service support dynamic service configuration. See the sections on configurable data services, adding data service resources to data services and updating role maps for more information.
- The data service will be deployed onto Tomcat. You will need to shutdown and restart Tomcat before clients are able to access the new data service.

To deploy a data service under GT4 requires execution of the deployDataServiceGT4 target. This takes the same arguments as deployDataServiceTomcat. The exception is the tomcat.dir argument - this target instead accepts gt.dir which specifies the location of GT4. If this argument is omitted then the GT4 location specified within GLOBUS_LOCATION is used.

5.4.2. Deploying Data Services Using a GUI

To deploy a data service under Tomcat:

- Run the following command from within the OGSA-DAI WSRF binary distribution directory:

```
$ ant guiDeployDataServiceTomcat
```

- A GUI will appear and you can select your Tomcat distribution by pressing the Select... button. If CATALINA_HOME is set then this will be offered as the default selection.
- When you are happy with the Path press Next.
- Now you will be asked to enter:
  - The local URL of the data service, for example `ogsadai/DataService`, in the Service Path field. For example ogsadai/DataService specifies a data service whose URL will be `http://HOST:PORT/wsrf/services/ogsadai/DataService`.

    - Clicking on the Dynamically configurable? box requests that the data service support dynamic service configuration. See the sections on configurable data services, adding data service resources to data services and updating role maps for more information.

- Press Next when you have entered this information.
- You will now be requested to press Next so that the tool can deploy the data service. If any problems occur then the tool will notify you of these.
- When the deployment has completed a completion window will be displayed. Press Finish to exit the tool.
- You will need to shutdown and restart Tomcat before clients are able to access the new data service.

To deploy data services under GT4 requires execution of the `guiDeployDataServiceGT4` target.

5.5. Checking the OGSA-DAI Data Service Deployment

To check that a service has deployed under Tomcat:

- Restart Tomcat.
- Using a Web browser, visit the following URL:

```
http://TOMCAT.HOST:TOMCAT.PORT/wsrf/services
```

- Eventually a list of service URLs will be displayed. You should see one corresponding to your service. For example:

```
ogsadai/DataService (wsdl)
```

- Click on the (wsdl) link and the WSDL describing the service should appear. For example:

```
<?xml version="1.0" encoding="UTF-8" ?>
<wsdl:definitions name="DataService"
...
```

To check that a service has deployed under GT4:

- Start the GT4 container:

```
$ cd $GLOBUS_LOCATION
$ ./bin/globus-start-container -nosec
```

- Eventually a list of service URLs will be displayed. You should see one corresponding to your service. For example:

```
[22]: http://129.215.56.6:8080/wsrf/services/ogsadai/DataService
```

An alternative is to test the service using the OGSA-DAI WSRF listResourcesClient client. We describe how to use this client in the next section.

5.6. ListResources Client

OGSA-DAI comes with a client that allows you to list the data service resources exposed by an OGSA-DAI WSRF data service.

To list the data service resources exposed by a data service:

- Ensure that your Web services container is running.
- Ensure that the GLOBUS_LOCATION environment variable is set.
- Run the following command from within the OGSA-DAI WSRF binary distribution directory:

  ```
  $ ant listResourcesClient -Ddai.url=SERVICE-URL
  ```

  where `-Ddai.url=` specifies the URL of the data service. If omitted then a default URL of `http://localhost:8080/wsrf/services/ogsadai/DataService` is used.

  For example:

  ```
  $ ant listResourcesClient
   -Ddai.url=http://localhost:8080/wsrf/services/ogsadai/MyDataService
  ```

- The data service resources exposed by your service will be displayed. For example:

  ```
  [java] Contacting ... http://localhost:8080/wsrf/services/ogsadai/DataService
  [java] Service version: OGSA-DAI WSRF 1.0
  [java] This service exposes no resources!
  ```

  For example:

  ```
  [java] Contacting ... http://localhost:8080/wsrf/services/ogsadai/DataService
  [java] Service version: OGSA-DAI WSRF 1.0
  [java] Number of resources: 2
  [java] Resource: MySQLResource
  [java] Resource: AnotherResource
  ```

Note

As an alternative to setting CATALINA_HOME or GLOBUS_LOCATION you can specify the Tomcat or GT4 locations using the -Dtomcat.dir=/path/to/Tomcat/directory or -Dgt.dir=/path/to/Globus/directory flags respectively.

5.7. Deploying Data Service Resources

This section describes how to deploy a data service resource. Once deployed the data service resource can then be added to the set of those exposed by a data service as described in the next section.

The characteristics of a data service resources are specified in a data services resource file. We describe these below.

There are a number of ways to create a data service resource file and deploy a data service resource. You can:

- Write or edit a data service resource file using an editor then deploy the data service resource using the command-line.
- Create new data service resource files and (optionally) deploy the data service resources using a GUI
- Load and edit existing data service resource files and (optionally) deploy the data service resources using a GUI

5.7.1. Data Service Resource Files

A data service resource file is used specify the properties of your data service resource. This is a simple properties file consisting of argument=value values. You can reuse the same file to deploy the same data service resource under different names or to customise a specific configuration in small or major ways. Alternatively you can just write a new file each time.

Within the OGSA-DAI WSRF distribution directory we provide an example - data.service.resource.properties. The file is fully commented.

The properties are as follows:

- dai.resource.id= - name for the data service resource.
- dai.data.resource.type=[Relational | XML | Files] - the type of data resource to which the data service resource provides access.
- dai.product.name= - data resource product name (optional).
- dai.product.vendor= - data resource product vendor (optional).
- dai.product.version= - data resource product version (optional).
- dai.data.resource.uri= - data resource URI. This must be compatible with the driver class specified next.
- dai.driver.class= - data resource driver class name. This is optional only if dai.data.resource.type=Files.
- dai.credential= - Grid certificate credentials of a user permitted to access the data resource. If omitted then any user will be allowed access.
- dai.user.name= - data resource user name. Optional only if there is no user name required for a database.
- dai.password= - corresponding data resource password. Optional if there is no user name required, or if the password is null.

5.7.2. Deploying Data Service Resources Using the Command-line

To deploy a data service resource under Tomcat:

- Take a copy of data.service.resource.properties within the OGSA-DAI WSRF distribution directory.
- Load the file into an editor and provide values specifying your data service resource - the comments in the file should help you.
- When done, save the file.
- Put all the JARs implementing the data resource driver within the drivers directory within the OGSA-DAI WSRF binary distribution directory.

- Run the following command from within the OGSA-DAI WSRF binary distribution directory:

```
$ ant deployResourceTomcat -Dtomcat.dir=/path/to/Tomcat/directory
-Ddai.service.resource.file=DAI-SERVICE-RESOURCE-FILE
```

  - `tomcat.dir` specifies the location of Tomcat. If this argument is omitted then the Tomcat location specified within CATALINA_HOME is used.
  - `dai.service.resource.file` specifies the location of a data service resource properties file. If no argument is given then a default location of data.service.resource.properties (within the OGSA-DAI WSRF distribution directory) is assumed.

- The data service resource will be deployed. For example:

```
$ ant deployResourceTomcat -Ddai.service.resource.file=my.config
...
[echo] Reading properties file my.config
[echo] Data service resource ID: MySQLResource
[echo] Data resource type: Relational
[echo] Data resource product: MySQL
[echo] Data resource vendor: MySQL
[echo] Data resource version: 1.0
[echo] Data resource URI: jdbc:mysql://myhost.example.com:3306/ogsadai/DATABASE
[echo] Data resource credential:
[echo] Data resource user name: ogsadai
[echo] Data resource password: ogsadai
[echo] Data driver class: org.gjt.mm.mysql.Driver
[echo] Deploying data service resource MySQLResource...
...
[echo] Data service resource deployed!
```

- Values in the same data service resource file can be overridden at the command line via the use of -Dproperty=value flags.
  - For example to deploy another data service resource with the same configuration as held within my.config you can execute:

```
$ ant deployResourceTomcat -Ddai.service.resource.file=my.config
-Ddata.resource.id=AnotherResource
```

  - As another example, consider deploying a data service resource with the same configuration but with a different location:

```
$ ant deployResourceTomcat -Ddai.service.resource.file=my.config
-Ddata.resource.id=YetAnotherResource
-Ddai.data.resource.uri=jdbc:mysql://myhost:8080/some/other/DB
```

  - Alternatively, you can just edit your data service resource file.

To deploy data service resources under GT4 requires execution of the deployResourceGT4 target. This takes the same arguments as deployResourceTomcat. The exception is the tomcat.dir argument - this target instead accepts gt.dir which specifies the location of GT4. If this argument is omitted then the GT4 location specified within GLOBUS_LOCATION is used.

5.7.3. Creating new Data Service Resource Files and Deployment

To create a new data service resource file and (optionally) deploy the data service resource under Tomcat:

- Run the following command from within the OGSA-DAI WSRF binary distribution directory:

```
$ ant guiCreateResourceTomcat
```

- A GUI will appear and you can select your database product from those offered.
- When you are happy with your selection, press Next.
- You will now have to enter information about the data service resource and your database product. This information includes:
  - Data Resource URI - data resource URI(required). This must be compatible with the driver class specified next.
  - Data Resource Driver Class - data resource driver class name. This is optional only if you selected a Files and directories product on the previous window.
  - Vendor - data resource product vendor (optional).
  - Version - data resource product version (optional).
  - Database access credential - Grid certificate credentials of a user permitted to access the data resource. If omitted then any user will be allowed access.
  - Database user ID - data resource user name. Optional only if there is no user name required for a database.
  - Database password - corresponding data resource password. Optional if there is no user name required, or if the password is null.
  - Data Service Resource ID - name for the data service resource (required).
- When you are happy with your values, press Next.
- You now have to select the JARs that implement your database driver if these have not already been installed on the server.
  - Using the file browser on the left select a JAR file. Press Add to add it to the list of current driver JARs on the right.
  - If you make a mistake then select the JAR on the right and press Remove to remove it from the list.
- When you are happy with your selections, press Next.
- You should now enter a file name in which to save your values. These will be saved in a new data service resource file. Press the Select... button, enter a new file name in the File name field then press Select
- When you are happy with the Path press Next.
- You will now be given the choice of pressing Cancel to exit, or Next to proceed to deploy your data service resource.

    - If you press Cancel you can always deploy your resource later using via the command-line as described above or a GUI as described below.

- You can select your Tomcat distribution by pressing the Select... button. If CATALINA_HOME is set then this will be offered as the default selection.
- When you are happy with the Path press Next.
- You will now be requested to press Next so that the tool can deploy the data service resource. If any problems occur then the tool will notify you of these.
- When the deployment has completed a completion window will be displayed. Press Finish to exit the tool.

To create and deploy data service resources under GT4 requires execution of the guiCreateResourceGT4 target.

5.7.4. Editing Existing Data Service Resource Files and Deployment

To edit an existing data service resource file and (optionally) deploy the data service resource under Tomcat:

- Run the following command from within the OGSA-DAI WSRF binary distribution directory:

  ```
  $ ant guiDeployResourceTomcat
  ```

- A GUI will appear and you can select your data service resource file by pressing the Select... button.
- When you are happy with the Path press Next.
- You can now edit the information about the data service resource and your database product.
- When you are happy with your values, press Next.
- You can now change the current selection of the JARs that implement your database driver.

    - Using the file browser on the left select a JAR file. Press Add to add it to the list of current driver JARs on the right.

    - If you make a mistake or if some of the JARs are no longer valid or required then select the JAR on the right and press Remove to remove it from the list.

- When you are happy with your selections, press Next.
- You should now enter a file name in which to save your values. These will be saved in a new data service resource file. Press the Select... button, enter a new file name in the File name field then press Select
- When you are happy with the Path press Next.
- You can now select your Tomcat distribution by pressing the Select... button. If CATALINA_HOME is set then this will be offered as the default selection.
- When you are happy with the Path press Next.
- You will now be requested to press Next so that the tool can deploy the data service resource. If any problems occur then the tool will notify you of these.
- When the deployment has completed a completion window will be displayed. Press Finish to exit the tool.

To edit existing data service resource files and deploy data service resources under GT4 requires execution of the guiDeployResourceGT4 target.

5.8. Adding Data Service Resources to Data Services

This section describes how to add a data service resource to a data service, thereby allowing clients to access and interact with the data service resource. It assumes that both the data service and data service resource have been deployed.

You can:

- Add data service resources using the command-line.
- Add data service resources using a GUI.
- Dynamically add a data service resource using the command-line.

5.8.1. Adding Data Service Resources Using the Command-line

To add data service resources to data services under Tomcat:

- Run the following command from within the OGSA-DAI WSRF binary distribution directory:

```
$ ant addResourceTomcat -Ddai.service.path=service/path
-Ddai.resource.id=ResourceID -Dtomcat.dir=/path/to/Tomcat/directory
```

  - dai.service.path specifies the local URL of the service. For example -Ddai.service.path=ogsadai/DataService specifies a data service whose URL will be http://HOST:PORT/wsrf/services/ogsadai/DataService.
  - -Ddai.resource.id= is the ID of the data service resource that the data service is to expose. This must be the name of a data service resource which you have already deployed as described in the previous section. For example -Ddai.service.resource=MySQLResource.
  - tomcat.dir specifies the location of Tomcat. If this argument is omitted then the Tomcat location specified within CATALINA_HOME is used.

- You will need to shutdown and restart Tomcat before clients are able to access the new data service resource via the data service.

To add data service resources under GT4 requires execution of the addResourceGT4 target. This takes the same arguments as addResourceTomcat. The exception is the tomcat.dir argument - this target instead accepts gt.dir which specifies the location of GT4. If this argument is omitted then the GT4 location specified within GLOBUS_LOCATION is used.

After restarting your container, you can test whether the new data service resource was successfully added using the OGSA-DAI WSRF listResourcesClient client. We describe how to use this client in this section.

5.8.2. Adding Data Service Resources Using a GUI

To add data service resources to data services under Tomcat:

- Run the following command from within the OGSA-DAI WSRF binary distribution directory:

```
$ ant guiAddResourceTomcat
```

- A GUI will appear and you can select your Tomcat distribution by pressing the Select... button. If CATALINA_HOME is set then this will be offered as the default selection.

- When you are happy with the Path press Next.
- Now you will be asked to enter:
  - The local URL of the data service, for example ogsadai/DataService, in the Service Path field.
  - The ID of the data service resource, for example MyRelationalResource in the Data Service Resource ID field.
- Press Next when you have entered this information.
- You will now be requested to press Next so that the tool can add the data service resource to the data service. If any problems occur then the tool will notify you of these.
- When the addition has completed a completion window will be displayed. Press Finish to exit the tool.
- You will need to shutdown and restart Tomcat before clients are able to access the new data service resource via the data service.

To add data service resources from data services under GT4 requires execution of the guiAddResourceGT4 target.

After restarting your container, you can test whether the new data service resource was successfully added using the OGSA-DAI WSRF listResourcesClient client. We describe how to use this client in this section.

5.9. Dynamically Add a Data Service Resource using the Command-line

If your data service is configurable then you can request that the data service immediately expose the data service resource, without having to restart your Web services container. This is done as follows:

- Ensure that the GLOBUS_LOCATION environment variable is set.
- Run the following command from within the OGSA-DAI WSRF binary distribution directory:

```
$ ant dataServiceClient -Ddai.url=SERVICE-URL
-Ddai.resource.id=RESOURCE-ID -Ddai.action=deploy
```

  where

  - dai.url specifies the URL of the data service. If omitted then a default URL of http://localhost:8080/wsrf/services/ogsadai/DataService is used.
  - dai.resource.id specifies the name of the data service resource.

  For example:

```
$ ant dataServiceClient
-Ddai.url=http://localhost:8080/wsrf/services/ogsadai/MyDataService
-Ddai.resource.id=MyNewResource -Ddai.action=deploy
```

- The request will be forwarded to the data service. For example:

```
[echo] Deploying resource  MyNewResource...
[java] Contacting ... http://localhost:8080/wsrf/services/ogsadai/DataService
[java] Service version: OGSA-DAI WSRF 1.0
[java] Number of resources: 1
```

```
[java] Resource: MySQLResource
[java] Data Service Resource: MyNewResource
[java] About to invoke Deploy...
[java] Deploy completed!
```

- You can now use the listResourcesClient client (described in this section) to check if the deployment was successful.

During a deployment the service will attempt to update a data service resources file on the server. If this fails then a ConfigurationFaultType will occur with a corresponding message. The new data service resource will still be available via the data service but not if the container is shutdown and restarted. If such a failure occurs you can always add the data service resource to the data service using the first two approaches above.

5.10. Removing Data Service Resources from Data Services

This section describes how to remove a data service resource from a data service - the data service will no longer expose the data service resource. It assumes that both the data service and data service resource have been deployed and that the data service resource has been added to the data service (as described on the previously.

You can:

- Remove data service resources using the command-line
- Remove data service resources using a GUI

5.10.1. Removing Data Service Resources Using the Command-line

To remove data service resources from data services under Tomcat:

- Run the following command from within the OGSA-DAI WSRF binary distribution directory:

```
$ ant removeResourceTomcat -Dtomcat.dir=/path/to/Tomcat/directory
-Ddai.service.path=service/path -Ddai.resource.id=ResourceID
```

  - tomcat.dir specifies the location of Tomcat. If this argument is omitted then the Tomcat location specified within CATALINA_HOME is used.
  - dai.service.path specifies the local URL of the service. For example -Ddai.service.path=ogsadai/DataService specifies a data service whose URL will be http://HOST:PORT/wsrf/services/ogsadai/DataService.
  - dai.resource.id is the ID of the data service resource that the data service is no longer to expose. For example -Ddai.service.resource=MyRelationalResource

- You will need to shutdown and restart Tomcat before the changes take effect.

To remove data service resources from data services under GT4 requires execution of the removeResourceGT4 target. This takes the same arguments as removeResourceTomcat. The exception is the tomcat.dir argument - this target instead accepts gt.dir which specifies the location of GT4. If this argument is omitted then the GT4 location specified within GLOBUS_LOCATION is used.

5.10.2. Removing Data Service Resources Using a GUI

To remove data service resources from data services under Tomcat:

- Run the following command from within the OGSA-DAI WSRF binary distribution directory:

```
$ ant guiRemoveResourceTomcat
```

- The tool will appear and you can select your Tomcat distribution by pressing the Select... button. If CATALINA_HOME is set then this will be offered as the default selection.
- When you are happy with the Path press Next.
- Now you will be asked to enter:
  - The local URL of the data service, for example ogsadai/DataService, in the Service Path field.
  - The ID of the data service resource, for example MyRelationalResource in the Data Service Resource ID field.
- Press Next when you have entered this information.
- You will now be requested to press Next so that the tool can remove the data service resource from the data service. If any problems occur then the tool will notify you of these.
- When the removal has completed a completion window will be displayed. Press Finish to exit the tool.
- You will need to shutdown and restart Tomcat before the changes take effect.

To remove data service resources from data services under GT4 requires execution of the guiRemoveResourceGT4

5.11. Updating Data Service Resource Role Maps

If you have deployed a configurable data service then you can make changes to the role maps file of any data service resources exposed by that data service. You can then request that the data service resource exposed by the service read in and use the updated role maps.

To update the role maps that a data service resource exposed by a specific data service uses:

- Make the changes to the role maps document for the specific data service resource of interest:
  - Within the GT4 container the role maps for a specific data service resource can be found in:

    ```
    etc/ogsadai_wsrf/ResourceID/DatabaseRoles.xml
    ```

    where ResourceID is the ID of the data service resource.
  - Within Tomcat container the role maps for a specific data service resource can be found in:

    ```
    webapps/wsrf/WEB-INF/etc/ogsadai_wsrf/ResourceID/DatabaseRoles.xml
    ```

    where ResourceID is the ID of the data service resource.
- Ensure that the GLOBUS_LOCATION environment variable is set.
- Run the following command from within the OGSA-DAI WSRF binary distribution directory:

  ```
  $ ant dataServiceClient -Ddai.url=SERVICE-URL
                      -Ddai.resource.id=RESOURCE-ID -Ddai.action=updateRoleMaps
  ```

  where

- dai.url specifies the URL of the data service. If omitted then a default URL of http://localhost:8080/wsrf/services/ogsadai/DataService is used.
- dai.resource.id specifies the data service resource at which the request is targeted.

For example:

```
$ ant dataServiceClient
                    -Ddai.url=http://localhost:8080/wsrf/services/ogsadai/MyDataService
                    -Ddai.resource.id=MySQLResource -Ddai.action=updateRoleMaps
```

- The request will be forwarded to the data service. For example:

```
[echo] Updating roleMaps for resource MySQLResource...
                    [java] Contacting ... http://localhost:8080/wsrf/services/ogsadai/Da
                    [java] Service version: OGSA-DAI WSRF 1.0
                    [java] Number of resources: 1
                    [java] Resource: MySQLResource
                    [java] Data Service Resource: MySQLResource
                    [java] About to invoke UpdateRoleMaps...
                    [java] UpdateRoleMaps completed!
```

- If you see the following message:

```
[echo] Updating roleMaps for resource MySQLResource...
                    ...
                    [java] About to invoke UpdateRoleMaps...
                    [java] This service does not support the updateRoleMaps operation!
```

Then the data service is not configurable i.e. it does not allow data service resources to dynamically update their role maps.

5.12. End-to-end Client

OGSA-DAI comes with a client that allows you to:

- Contact a data service and get a list of the data service resources it exposes.
- Submit a Perform document to the data service and print the resulting Response document.

To submit a Perform document to a data service resource exposed by a data service and so perform some data resource-related activities (e.g. query a database):

- Ensure that your Web services container is running.
- Ensure that the GLOBUS_LOCATION environment variable is set.
- Run the following command from within the OGSA-DAI WSRF binary distribution directory:

```
$ ant dataServiceClient -Ddai.url=SERVICE-URL
                    -Ddai.resource.id=RESOURCE-ID
                    -Ddai.action=PERFORM-DOC-LOCATION
```

where

- -Ddai.url= specifies the URL of the data service. If omitted then a default URL of http://localhost:8080/wsrf/services/ogsadai/DataService is used.
- -Ddai.resource.id= specifies the data service resource at which the Perform document is targeted.
- -Ddai.action= specifies the location of an OGSA-DAI Perform document.

For example:

```
$ ant dataServiceClient
                    -Ddai.url=http://localhost:8080/wsrf/services/ogsadai/MyDataService
                    -Ddai.resource.id=MySQLResource
                    -Ddai.action=examples/GDSPerform/JDBC/query/select1Row.xml
```

- The Perform document will be forwarded to the data service and executed by the specified data service resource. For example:

```
[echo] Executing Perform document on resource One...
                    [java] Contacting ... http://localhost:8080/wsrf/services/ogsadai/My
                    [java] Service version: OGSA-DAI WSRF 1.0
                    [java] Number of resources: 2
                    [java] Resource: MySQLResource
                    [java] Resource: AnotherResource
                    [java] Data Service Resource: MySQLResource
                    [java] About to invoke Perform...
                    [java] Perform completed!
                    [java] Response:
                    [java] <?xml version="1.0" encoding="UTF-8"?>
                    [java] <ns1:response xmlns:ns1="http://ogsadai.org.uk/namespaces/200
                    [java] <request status="COMPLETED"
                    xmlns="http://ogsadai.org.uk/namespaces/2005/03/service/types"/>
                    ...
```

Note

As an alternative to setting CATALINA_HOME or GLOBUS_LOCATION you can specify the Tomcat or GT4 locations using the -Dtomcat.dir=/path/to/Tomcat/directory or -Dgt.dir=/path/to/Globus/directory flags respectively.

5.13. Configurable Data Services

Configurable data services are a special type of data service. Asides from all the operations and capabilities offered by services a configurable data service provides operations allowing the state of the service or the data service resources it exposes to be updated while the service is running.

These operations are intended for use by service deployers rather than clients of the service.

5.13.1. Configuration Operations

5.13.1.1. The Deploy Operation

This operation is not directed at a specific data service resource. Instead it contains the name of a data service resource which the data service is to expose. Upon completion of the operation clients will be able to access the named data service resource via the service.

It is the responsibility of the service deployer to ensure that the information required to configure the data service resource is available within the Web service container hosting the service.

5.13.1.2. The UpdateRoleMaps Operation

This operation is directed at a specific data service resource - (specified within the WS-Addressing Endpoint Reference). The operation requests that a data service resource re-read its role maps file which resides on the server.

This operation allows service deployers to change the list of clients authorised to access a data resource known to a data service resource and then to ensure that these changes take force immediately.

5.14. Undeploying OGSA-DAI WSRF

The OGSA-DAI WSRF binary distribution can be undeployed from the GT4 container or Apache Tomcat:

- Either via the Command-line
- Or using a GUI

Undeployment erases all OGSA-DAI XML Schema, OGSA-DAI-specific JAR files and configuration files from the destination.

5.14.1. Undeploying OGSA-DAI WSRF via the Command-line

5.14.1.1. Undeploying OGSA-DAI WSRF from GT4

To undeploy OGSA-DAI WSRF from GT4:

- Set a GLOBUS_LOCATION environment variable to point to the location of your GT4 distribution. For example, under UNIX, enter:

```
$ export GLOBUS_LOCATION=/path/to/Globus/directory
```

- Run the following from within the OGSA-DAI binary distribution directory:

```
$ ant undeployGT4
```

OGSA-DAI WSRF will then be undeployed.

5.14.1.2. Undeploying OGSA-DAI WSRF from Tomcat

To undeploy OGSA-DAI WSRF from Tomcat:

- Set a CATALINA_HOME environment variable to point to the location of Tomcat. For example, under UNIX, enter:

```
$ export CATALINA_HOME=/path/to/Tomcat/directory
```

- Run the following from within the OGSA-DAI binary distribution directory:

```
$ ant undeployTomcat
```

OGSA-DAI WSRF will then be undeployed from Tomcat.

Note

As an alternative to setting CATALINA_HOME you can specify the Tomcat location at the command-line as follows:

```
$ ant undeployTomcat -Dtomcat.dir=/path/to/Tomcat/directory
```

5.14.2. Undeploying OGSA-DAI WSRF using a GUI

5.14.2.1. Undeploying OGSA-DAI WSRF from GT4

To undeploy OGSA-DAI WSRF from GT4:

- Run the following from within the OGSA-DAI binary distribution directory:

```
$ ant guiUndeployGT4
```

- A GUI will appear and you can select your GT4 distribution by pressing the Select... button. If GLOBUS_LOCATION is set then this will be offered as the default selection.
- When you are happy with the Path press Next.
- You will now be requested to press Next so that the tool can deploy OGSA-DAI WSRF. If any problems occur then the tool will notify you of these.
- When the undeployment has completed a completion window will be displayed. Press Finish to exit the tool.

5.14.2.2. Undeploying OGSA-DAI WSRF from Tomcat

To undeploy OGSA-DAI WSRF from Tomcat:

- Run the following from within the OGSA-DAI binary distribution directory:

```
$ ant guiUndeployTomcat
```

- A GUI will appear and you can select your Tomcat distribution by pressing the Select... button. If CATALINA_HOME is set then this will be offered as the default selection.
- When you are happy with the Path press Next.
- You will now be requested to press Next so that the tool can deploy OGSA-DAI WSRF. If any problems occur then the tool will notify you of these.
- When the undeployment has completed a completion window will be displayed. Press Finish to exit the tool.

5.15. Where Does Stuff Go?

This section describes the locations of OGSA-DAI related files after deployment on Tomcat or GT4.

Under GT4, OGSA-DAI resides in the following places:

- share/schema/ogsadai/ - OGSA-DAI XML Schema and WSDL.
- lib/ - OGSA-DAI JAR files.
- etc/ogsadai_wsrf/ - OGSA-DAI configuration files - see below.

Under Tomcat, OGSA-DAI resides in the following places:

- webapps/wsrf/share/schema/ogsadai/ - OGSA-DAI XML Schema and WSDL.
- webapps/wsrf/WEB-INF/lib/ - OGSA-DAI JAR files.
- webapps/wsrf/WEB-INF/etc/ogsadai_wsrf/ - OGSA-DAI configuration files - see below.

The OGSA-DAI WSRF configuration files residing in the etc/ogsadai_wsrf directories are as follows:

- server-config.wsdd - deployment descriptors for the current OGSA-DAI services.
- jndi-config.xml - JNDI deployment descriptors for the current OGSA-DAI services (see the GT4 documentation for more information).
- Data service resource files for each current data service. There will be one of these XML files for each current OGSA-DA service. These files maintain a list of the current data service resources for each data service. The file name is derived from the service path e.g. a service with relative service path ogsadai/DataService has a file _ogsadai_DataService.dsr.xml.
- Data service resource directories. There is one such directory for each deployed data service resource - the directory shares the name of the data service resource e.g. MyRelationalResource Each directory contains:
  - dataResourceConfig.xml - data resource configuration file for the data service resource.
  - activityConfig.xml - activity configuration file for the data service resource.
  - DatabaseRoles.xml - roleMaps file for the data service resource.

6. Testing

Please see the deployment section for information on testing and configuring OGSA-DAI.

7. Security considerations

OGSA-DAI does not provide any security over and above that already provided by the Globus Toolkit. However, consideration must be given to the role mapping which converts a grid credential to a database username/password. OGSA-DAI comes with two basic role mappers: a simple role mapper which accesses a plain text file with this information and one which encrypts the username/password information. More details can be found in the documentation bundled with the documentation. If neither of these schemes are secure enough then it is possible to replace it by implementing a new role mapper interface.

8. Troubleshooting

Generating verbose log output is a critical aid in troubleshooting of the DRS and is useful when communicating problems to Globus support lists. To increase logging detail, add the following line to the `$GLOBUS_LOCATION/container-log4j.properties` file.

```
...
log4j.category.org.globus.replica=DEBUG
...
```

8.1. I am using MySQL and keep on getting a java.sql.SQLException: Cannot connect to MySQL server on localhost:3306 exception - what is the problem?

The MySQL JDBC driver recommended in the OGSA-DAI documentation is not compatible with the latest versions of MySQL. This is the case for MySQL 4.1.6 and 4.1.7 and may be the case for other versions. This problem usually causes the following error message to be logged:

```
java.sql.SQLException: Cannot connect to MySQL server on localhost:3306.
```

8.2. Is there a MySQL server running on the machine/port you are trying to connect to? (java.lang.NumberFormatException)

To address this problem all you need to do is use a newer version of the MySQL JDBC driver JAR which can be downloaded from:

http://www.mysql.com/products/connector/j/

If you have already deployed services using the incompatible JAR, follow these steps:

- Stop your container.
- Replace the JAR file in the lib directory in the container. See Where Does Stuff Go? above.
- Restart the container.

8.3. Why do I get Exception in thread "main" java.lang.IllegalAccessError: tried to access field...?

This error can arise due to clashes of Xalan shipped with certain versions of Java and that shipped with the Globus Toolkit. Please see the OGSA-DAI FAQ at http://www.ogsadai.org.uk/docs/R6.0/doc/misc/FAQ.html#IllegalAccessError for more information.

Please check up the OGSA-DAI website (www.ogsadai.org.uk[4]) for any updates. In particular the OGSA-DAI FAQ although note that this also caters for the non-WSRF distributions of OGSA-DAI.

[4] http://www.ogsadai.org.uk

Chapter 8. GT 4.0 OGSA-DAI: User's Guide

1. Introduction

The OGSA-DAI User's Guide provides general end user-oriented information.

2. Command-line tools

Please consult the documentation included in the OGSA-DAI distribution.

3. Usage scenarios

Please see the documentation available in the OGSA-DAI distribution.

4. Troubleshooting

Please check with the Administration Troubleshooting section for some common errors encountered with OGSA-DAI. More information will also be available from the OGSA-DAI web site (http://www.ogsadai.org.uk).

Chapter 9. GT 4.0 OGSA-DAI: Developer's Guide

1. Introduction

This guide contains information of interest to developers working with OGSA-DAI. It provides reference information for application developers, including APIs, architecture, and procedures for using the APIs.

2. Before you begin

2.1. Feature summary

Features new in release GT 4.0:

- Access to data is provided via an "OGSA-DAI data service". For those that have previously used the OGSI version of OGSA-DAI, this service amalgamates the capabilities of the Grid Data Service Factory (GDSF) and Grid Data Service (GDS) services (the metadata and configuration roles of the GDSF and the metadata and perform document processing aspects of the GDS).
- Each data service exposes zero or more data service resources. A data service resource manages exposure of and interaction with a data resource. A single data service resource can be viewed as offering capabilities analogous to a single GDS in the OGSI version of OGSA-DAI.
- Allows multiple data resources to be accessed through a single service. Data service resource identifiers, available from the data service's WS-Addressing endpoint reference, allow a client to target a specific data service resource.
- A listResources() operation is provided for a data service to list all the identifiers of the data service resources exposed by that service.
- The data service resource identifiers returned by a data service can subsequently be used by a client to obtain metadata and other information, about the data service resources corresponding to that identifier.
- Access to data service resource metadata (such as database schemas, request status, etc.) is provided by an implementation of the WS-ResourceProperties specification. In particular support for using the QueryResourceProperties, GetResourceProperties and GetMultipleResourceProperties portTypes is provided.
- Access to version information about the OGSA-DAI Data Service is available through the getVersion() operation.
- A WSRF version of the OGSA-DAI GridDataTransport portType supports asynchronous data delivery between data services.

Other Supported Features (features that continue to be supported from previous versions):

- Perform documents.
- Extensible activity framework.
- Statement, delivery and transformation activities.

Deprecated Features

- The OGSA-DAI Client Toolkit has not been updated yet to support interaction with WSRF-enabled OGSA-DAI services. This will be made available in a future version.
- The OGSA-DAI registry service, DAISGR, has not been updated for WSRF. This will be addressed in a future version or provided via a third party component.
- The old graphical demonstrator no longer works with this version of OGSA-DAI.

2.2. Tested platforms

Features new in release GT 4.0:

- Access to data is provided via an "OGSA-DAI data service". For those that have previously used the OGSI version of OGSA-DAI, this service amalgamates the capabilities of the Grid Data Service Factory (GDSF) and Grid Data Service (GDS) services (the metadata and configuration roles of the GDSF and the metadata and perform document processing aspects of the GDS).
- Each data service exposes zero or more data service resources. A data service resource manages exposure of and interaction with a data resource. A single data service resource can be viewed as offering capabilities analogous to a single GDS in the OGSI version of OGSA-DAI.
- Allows multiple data resources to be accessed through a single service. Data service resource identifiers, available from the data service's WS-Addressing endpoint reference, allow a client to target a specific data service resource.
- A listResources() operation is provided for a data service to list all the identifiers of the data service resources exposed by that service.
- The data service resource identifiers returned by a data service can subsequently be used by a client to obtain metadata and other information, about the data service resources corresponding to that identifier.
- Access to data service resource metadata (such as database schemas, request status, etc.) is provided by an implementation of the WS-ResourceProperties specification. In particular support for using the QueryResourceProperties, GetResourceProperties and GetMultipleResourceProperties portTypes is provided.
- Access to version information about the OGSA-DAI Data Service is available through the getVersion() operation.
- A WSRF version of the OGSA-DAI GridDataTransport portType supports asynchronous data delivery between data services.

Other Supported Features (features that continue to be supported from previous versions):

- Perform documents.
- Extensible activity framework.
- Statement, delivery and transformation activities.

Deprecated Features

- The OGSA-DAI Client Toolkit has not been updated yet to support interaction with WSRF-enabled OGSA-DAI services. This will be made available in a future version.
- The OGSA-DAI registry service, DAISGR, has not been updated for WSRF. This will be addressed in a future version or provided via a third party component.
- The old graphical demonstrator no longer works with this version of OGSA-DAI.

2.3. Backward compatibility summary

Protocol changes since GT version 3.2:

- Not backwards compatible with the previous OGSI version.

API changes since GT version 3.2:

- None

Exception changes since GT version 3.2:

- None

Schema changes since GT version 3.2:

- WSDL changes to work with new Java WS Core.

2.4. Technology dependencies

OGSA-DAI depends on the following GT components:

- Java WS Core

OGSA-DAI depends on the following 3rd party software:

- Java 1.4.0 (OGSA-DAI WSRF has been tested on this version of Java though may work with other Java 1.4.x flavours).
- Jakarta ANT 1.5 (see http://ant.apache.org).

Depending on the underlying data resource that OGSA-DAI is going to expose, you may need one or more of the following:

- For relational databases such as MySQL, PostgreSQL, SQLServer, Oracle and DB2, the corresponding JDBC drivers are required.
- For XML databases such as Xindice, the corresponding XMLDB drivers are required.
- To use a full text search engine against flat files, Jakarta Lucene is required.

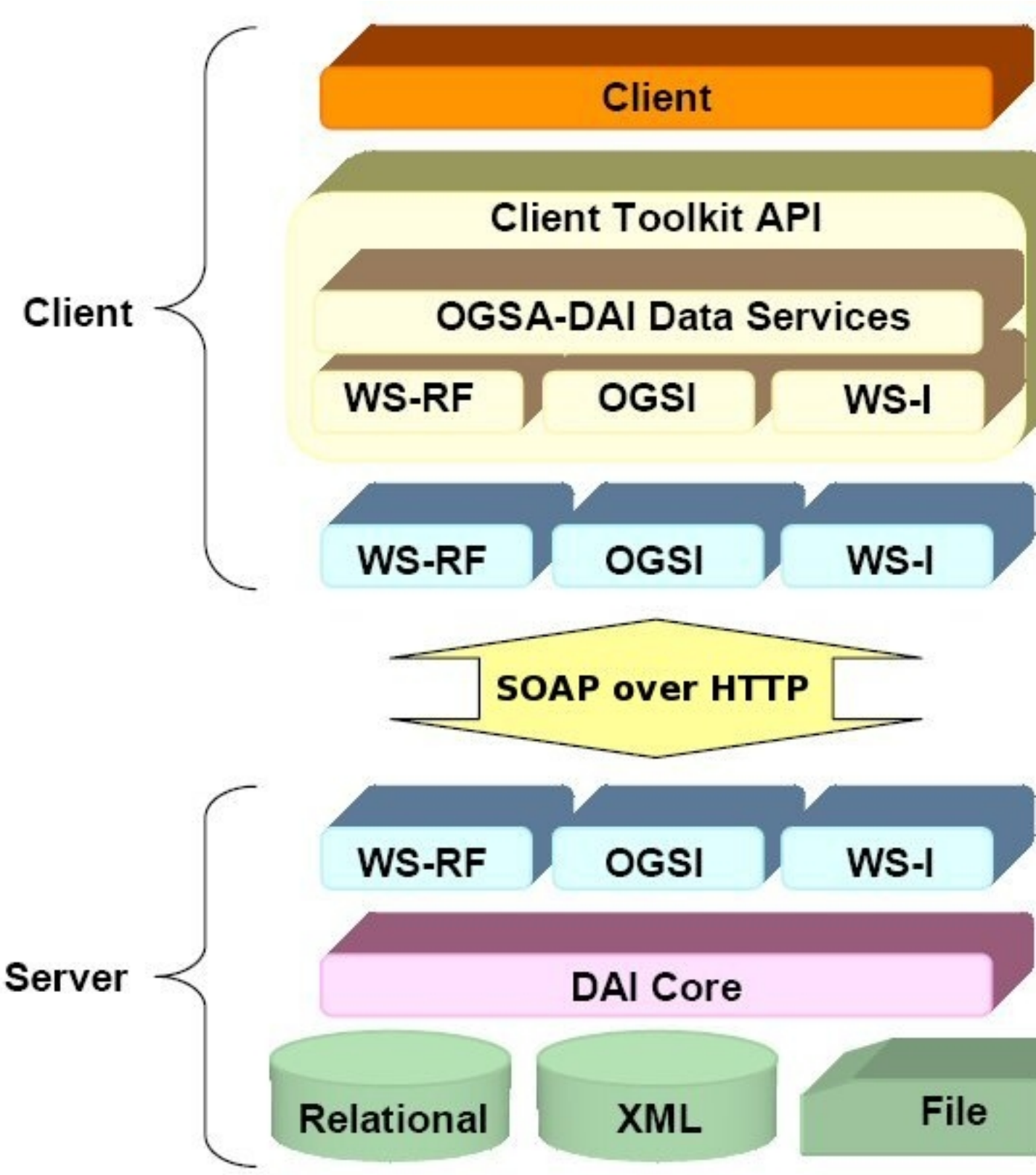
2.5. Security considerations

OGSA-DAI does not provide any security over and above that already provided by the Globus Toolkit. However, consideration must be given to the role mapping which converts a grid credential to a database username/password. OGSA-DAI comes with two basic role mappers: a simple role mapper which accesses a plain text file with this information and one which encrypts the username/password information. More details can be found in the documentation bundled with the documentation. If neither of these schemes are secure enough then it is possible to replace it by implementing a new role mapper interface.

3. Architecture and design overview

A high-level schematic representation of the OGSA-DAI architecture is shown in the diagram below. This is an end goal. All the components may not yet be available within one of the OGSA-DAI distributions. When you download

an OGSA-DAI distribution you will only get components specific to one of the WS-I, WSRF or OGSI based infrastructures. The version of OGSA-DAI included in the GT4.0 release only includes the WSRF components. Also note that there is no client toolkit for WSRF in this release. For information about and availability of other releases please visit the project website www.ogsadai.org.uk.



OGSA-DAI High-level Architecture

The different components in this diagram are explained below, working from the bottom of the diagram up to the top.

3.1. Data Layer

The data layer consists of data resources which can be exposed via OGSA-DAI. Currently these include:

- Relational data resources, e.g. MySQL, SQL Server, DB2, Oracle.
- XML data resources, e.g. Xindice.
- Files data resources, e.g. files and directories, OMIM, SWISSPROT and EMBL.

3.2. Data Layer-Business Logic Layer Interface

This interface communicates information between the business logic and data layers. This interface is provided by OGSA-DAI classes which invoke JDBC drivers, XMLDB drivers, or other OGSA-DAI classes to manage communications to and from data resources.

3.3. Business Logic Layer

This layer encapsulates the core functionality of OGSA-DAI. This layer consists of components which manage:

- Execution of Perform documents which encapsulate a pipelined sequence of activities to be executed at the service. These could consist of queries or updates that operate on a data resource and/or data transformation and/or delivery operations acting on the incoming or outgoing data streams.
- Preparation of responses to client requests for data resource query, update, transformation and delivery activities. Responses include execution status information and can also include data. Responses are in the form of Response documents.
- Data transformation and delivery management.
- Connection to, management of and interaction with data resources.

The business logic later is termed the DAI-Core.

3.4. Presentation Layer-Business Logic Layer Interface

This interface communicates information between the presentation and business logic layers. This interface supports invocation of OGSA-DAI functionality within the business logic layer in a way that is independent of any Web or Grid environment, i.e. a way that is also suitable to allow non-Web-enabled clients to access OGSA-DAI functionality directly. This interface provides the following:

- Components to extract information for the business logic layer from requests arriving via:
  - OGSA-DAI WSRF services.
  - OGSA-DAI WS-I services.
  - OGSA-DAI OGSI services.
- Note that WS-I is being used in a very specific way - services that only require those standards that are addressed in the WS-I Basic Profile 1.1 document. Components to extract information from the business logic layer and build responses to be provided via:

- OGSA-DAI WSRF services.
- OGSA-DAI WS-I services.
- OGSA-DAI OGSI services.

3.4.1. Information From Presentation Layer to Business Logic Layer

- Client proxy certificates and credentials in a Web- and Grid-independent format.
- Received data.
- Perform documents from clients.
- DAI-Core configuration information including data resource drivers, data resource URIs, database user names and passwords, information on supported activities and the legal form of Perform documents.

3.4.2. Information From Business Logic Layer to Presentation Layer

- Response documents.
- Data for delivery.
- Data resource schema.
- Information on data resource-related activities that can be requested by the user within Perform documents.

3.5. Presentation Layer

This layer encapsulates the functionality relating to exposing OGSA-DAI to a Grid via Web- or Grid-enabled interfaces. For each realisation there is associated WSDL and XML Schema describing the Web- or Grid-enabled interfaces. The following presentation layer interfaces are supported:

- OGSA-DAI OGSI-compliant services based on the Globus Toolkit 3.2.
- OGSA-DAI WSRF-compliant services based on the Globus Toolkit 4.0.
- OGSA-DAI WS-I-compliant services based on Apache Axis 1.2.

3.6. Clients

OGSA-DAI can support access by any suitable OGSI-, WSRF or WS-I-compliant client, depending on the OGSA-DAI presentation layer deployed at the server which the client is trying to access.

OGSA-DAI provides a Client Toolkit which provides a higher-level of interaction with OGSA-DAI services than that supported by exchanging Perform and Response documents. This however is not yet supported by the WSRF version of OGSA-DAI. There will be a version available for the next WSRF version of OGSA-DAI that will be available from the OGSA-DAI web site and will be included in the next Globus Toolkit release.

4. Public interface

The semantics and syntax of the APIs for this component can be found in the public interface guide[1].

[1] OGSA_DAI_Public_Interfaces.html

5. Usage scenarios

5.1. Interacting with Data Resources

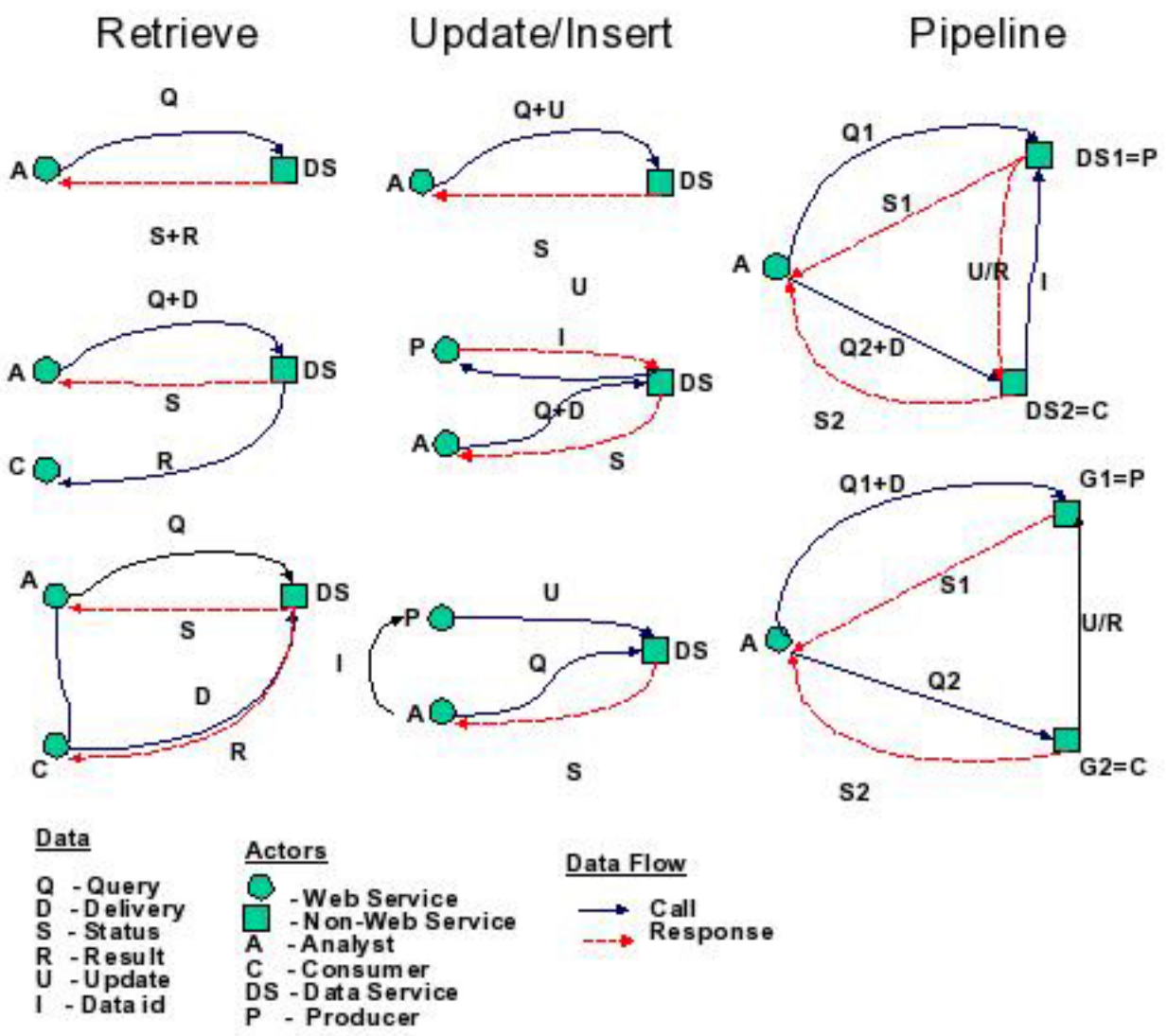
OGSA-DAI supports interaction with data resources, and other data manipulation operations, via a document-oriented interface. The basic building blocks for doing this are:

- Activities - form the basic data resource manipulation, data transformation and delivery operations that a client may want to perform. OGSA-DAI comes with a bunch of pre-defined activities, see the documentation in the distribution for more detail on what is available, and if you do not find the activity that you own then you can write your own and plug it into the existing framework. Activities are the basic building block of Perform documents.

- Perform Documents - allow clients to specify a collection of operation that they would like to perform on a data resource. Activities are collected together in the perform document with a simple data flow going from one activity to the next. In principle a simple operation could be to perform an SQL query on a relational database, take the results and perform an XSL Transform - assuming that the results are returned in XML format and then delivered to a third party using ftp. A number of activities could be linked together to process the data before it is delivered. The object here is to take the computation to the data and avoid as many client-service interaction. Perform documents are submitted to data service resources exposed by OGSA-DAI data services - a data service resource manages the exposure of and interaction with a data resource.

- Response Documents - are returned by data service resources via OGSA-DAI services to clients to inform them as to the execution status of their Perform documents and, often, to also return data directly back to a client if third party delivery is not being used.

Using these fairly basic properties it is possible to achieve fairly complex interaction patterns as illustrated in the figure below.



Possible OGSA-DAI Scenario Configurations

Thus, one is able to do a simple query-response interaction where the data goes directly back to the analyst querying a data service resource exposed by a data service. A third party delivery where the data is returned to a third party consumer and the analyst receives the status of the the perform document. Another scenario has the client passing on the details of the data service resource (and the data service which exposes it) to a third party who then pulls the data from the data service resource via the data service.

The process is relatively similar for updates where the analyst can submit the update directly to a data service resource (via a data service), with the data being transferred in the same message. Alternatively the data for an insert may be pulled or pushed from a third party.

More complex scenarios are also possible with the ability to perform service-to-service communication. However, although it is possible to do this within OGSA-DAI it has been found that performing data transfers using SOAP is not very efficient. More work remains to be done in this area.

Thus based on the simple concept of activities, perform and response documents fairly complex scenarios can be achieved using OGSA-DAI.

6. Debugging

There is no particular debugging advice at this time.

7. Troubleshooting

Please check with the Administration Troubleshooting section for some common errors encountered with OGSA-DAI. More information will also be available from the OGSA-DAI web site (www.ogsadai.org.uk[2]).

8. Related Documentation

OGSA-DAI related presentations can be found at the OGSA-DAI web sited under: www.ogsadai.org.uk/docs[3] - note that this contains information about the different flavours of OGSA-DAI and not just the WSRF version. You also will find links to courses and tutorials at www.ogsadai.org.uk/courses/[4].

[2] http://www.ogsadai.org.uk
[3] http://www.ogsadai.org.uk/docs
[4] http://www.ogsadai.org.uk/courses/

Chapter 10. GT 4.0 Component Fact Sheet: OGSA-DAI (Tech Preview)

1. Brief component overview

OGSA-DAI provides a pure Java data service framework for accessing and integrating data resources - such as files, relational and XML databases - on to Grids. Towards this end, OGSA-DAI:

- Exposes intrinsic data resource capabilities - such as the ability to perform SQL queries on relational resources or evaluate XPath statements on XML collections - through web service based interfaces, thus allowing data resources to be easily incorporated as first class citizens in grids.
- Allows additional functionality to be implemented at the service - such as transformation of data coming out of a data resource - so as to avoid unnecessary data movement.
- Provides a compact way of handling multiple potential interactions with a service within a single request via an XML document, called a perform document, where data is pipelined between different set of activities that operate on a data stream coming out of, or going into, a data resource.
- Allows developers to easily add or extend functionality within OGSA-DAI. The perform document, and underlying framework, are extensible allowing additional functionality to be added, or existing functionality to be customised, and still operate within the same framework.
- Allows metadata about data and the data resources in which it is found to be queried via a service interface.
- Facilitates the provision of data integration capabilities from various sources to obtain the required information.

This WS-RF distribution of OGSA-DAI has been designed to work with the Globus Toolkit 4 implementation of WS-RF; a WS-I distribution is also available from the OGSA-DAI web site, which is designed to work with the UK OMII web services infrastructure releases. Finally, an OGSI-based distribution of OGSA-DAI, designed to work with the GT3.2 releases, is also available from the project web site. Note: these releases are not designed to inter-operate. More information about these and OGSA-DAI in general is available from the project website at http://www.ogsadai.org.uk.

2. Summary of features

Features new in release GT 4.0:

- Access to data is provided via an "OGSA-DAI data service". For those that have previously used the OGSI version of OGSA-DAI, this service amalgamates the capabilities of the Grid Data Service Factory (GDSF) and Grid Data Service (GDS) services (the metadata and configuration roles of the GDSF and the metadata and perform document processing aspects of the GDS).
- Each data service exposes zero or more data service resources. A data service resource manages exposure of and interaction with a data resource. A single data service resource can be viewed as offering capabilities analogous to a single GDS in the OGSI version of OGSA-DAI.
- Allows multiple data resources to be accessed through a single service. Data service resource identifiers, available from the data service's WS-Addressing endpoint reference, allow a client to target a specific data service resource.
- A listResources() operation is provided for a data service to list all the identifiers of the data service resources exposed by that service.

- The data service resource identifiers returned by a data service can subsequently be used by a client to obtain metadata and other information, about the data service resources corresponding to that identifier.
- Access to data service resource metadata (such as database schemas, request status, etc.) is provided by an implementation of the WS-ResourceProperties specification. In particular support for using the QueryResourceProperties, GetResourceProperties and GetMultipleResourceProperties portTypes is provided.
- Access to version information about the OGSA-DAI Data Service is available through the getVersion() operation.
- A WSRF version of the OGSA-DAI GridDataTransport portType supports asynchronous data delivery between data services.

Other Supported Features (features that continue to be supported from previous versions):

- Perform documents.
- Extensible activity framework.
- Statement, delivery and transformation activities.

Deprecated Features

- The OGSA-DAI Client Toolkit has not been updated yet to support interaction with WSRF-enabled OGSA-DAI services. This will be made available in a future version.
- The OGSA-DAI registry service, DAISGR, has not been updated for WSRF. This will be addressed in a future version or provided via a third party component.
- The old graphical demonstrator no longer works with this version of OGSA-DAI.

3. Backward compatibility summary

Protocol changes since GT version 3.2:

- Not backwards compatible with the previous OGSI version.

API changes since GT version 3.2:

- None

Exception changes since GT version 3.2:

- None

Schema changes since GT version 3.2:

- WSDL changes to work with new Java WS Core.

4. Technology dependencies

OGSA-DAI depends on the following GT components:

- Java WS Core

OGSA-DAI depends on the following 3rd party software:

- Java 1.4.0 (OGSA-DAI WSRF has been tested on this version of Java though may work with other Java 1.4.x flavours).
- Jakarta ANT 1.5 (see http://ant.apache.org).

Depending on the underlying data resource that OGSA-DAI is going to expose, you may need one or more of the following:

- For relational databases such as MySQL, PostgreSQL, SQLServer, Oracle and DB2, the corresponding JDBC drivers are required.
- For XML databases such as Xindice, the corresponding XMLDB drivers are required.
- To use a full text search engine against flat files, Jakarta Lucene is required.

5. Tested platforms

Tested Platforms for OGSA-DAI

- Sun Solaris 8 (SPARC)
- Microsoft Windows 2000 (X86)
- Microsoft Windows XP (X86)
- Redhat Linux 9 (X86)

We have not tested on other platforms, but if you do and have problems, we would like to hear about it.

OGSA-DAI has been tested with the following databases:

- MySQL 3.2.3 and above
- PostgreSQL 7.4
- IBM DB2
- Oracle 9i
- MS SQLServer 2000
- Derby
- Xindice 1.0

6. Associated standards

In essence, OGSA-DAI uses the Globus Toolkit Java WS Core component; therefore, any standards that apply to this apply equally well to OGSA-DAI. In addition, the base framework used is based on a GGF DAIS base specification presented at GGF 7: Grid Data Service Specification.

7. For More Information

Click here[1] for more information about this component.

[1] index.html

Chapter 11. GT 4.0 Component Guide to Public Interfaces: OGSA-DAI

1. Semantics and syntax of APIs

1.1. Programming Model Overview

Please consult the documentation included in the OGSA-DAI distribution.

1.2. Component API

Please consult the documentation included in the OGSA-DAI distribution.

2. Semantics and syntax of the WSDL

Please consult the documentation included in the OGSA-DAI distribution.

2.1. Resource properties

Supported resource properties for OGSA-DAI include:

- `productInfo`: Information on a data resource, including name, vendor and version.
- `databaseSchema`: Data resource schema (for relational data resources).
- `collectionSchema`: Data resource collection schema (for XML data resources).
- `activityTypes`: Activities the data service resource can be requested to perform on the data resource - the legal activities in Perform documents that can be sent to the data service resource via the data service.
- `performDocumentSchema`: XML Schema specifying the legal structure of Perform documents that can be sent to the data service resource via the data service.
- `requestStatus`: Status of execution of the current or more recent Perform document sent to the data service resource.

3. Command-line tools

Please consult the documentation included in the OGSA-DAI distribution.

4. Overview of Graphical User Interface

Please consult the documentation included in the OGSA-DAI distribution for information about GUI interfaces to access services.

5. Semantics and syntax of domain-specific interface

5.1. Interface introduction

Please consult the documentation included in the OGSA-DAI distribution.

5.2. Syntax of the interface

Please consult the documentation included in the OGSA-DAI distribution.

6. Configuration interface

Please consult the Section 5, “Deploying” section in the Admin Guide for information on configuring OGSA-DAI services.

7. Environment variable interface

Please consult the documentation included in the OGSA-DAI distribution.

Chapter 12. GT 4.0 OGSA-DAI: Quality Profile

1. Test coverage reports

This information is not publicly available for OGSA-DAI yet.

2. Code analysis reports

There are no code analysis reports for this version of OGSA-DAI yet.

3. Outstanding bugs

At the present time we are not aware we are not aware of any outstanding bugs in this distribution. If you find any bugs or want to know if any have been reported please consult www.ogsadai.org.uk/support[1].

4. Bug Fixes

This release does not explicitly fix bugs in the GT 3.2 OGSA-DAI contribution.

5. Performance reports

This information is not publicly available for OGSA-DAI yet.

[1] http://www.ogsadai.org.uk/support

Chapter 13. GT 4.0 Migrating Guide for OGSA-DAI

The following provides available information about migrating from previous versions of the Globus Toolkit.

1. Migrating from GT2

OGSA-DAI was not a GT2 component. There is no upgrade path from OGSA-DAI.

2. Migrating from GT3

The version of OGSA-DAI distributed with the Globus Toolkit 4.0.1 adds the new client toolkit libraries which were absent in the OGSA-DAI version distributed with Globus Toolkit 4.0.0, as well as a number of other additions which brings the functionality close to that found in the previous OGSI versions of OGSA-DAI.

The two remaining issues outstanding in this distribution are that message level security is not supported 'out of the box' and there is no support for concurrent requests, due to the removal of the OGSI based Grid Data Service Factory. Both of these issues will be remedied in the next release.

This release is suitable for evaluation and development of OGSA-DAI on the GT4.0.x platform, and the upgrade path to future releases for production systems should be straightforward.

Chapter 14. GT 4.0 OGSA-DAI: BLOBs Error Workaround

When trying to retrieve BLOBs from a relational database you will get zero rows returned (or an error). The following error message will be logged:

```
Base64Provider has not been set up with a Base64 implementation
```

This will be fixed in the next release. In the meantime if you have the source version of the release, you can correct this bug by changing the following two files and rebuilding OGSA-DAI.

1. Editing DataServiceImpl.java

At the bottom of the initialise() method in `uk/org/ogsadai/service/wsrf/dataservice/impl/DataServiceImpl.java`, make the following changes:

OLD VERSION

```
...
    LOG.error(msg);
    }
    }
    if (LOG.isDebugEnabled()) {

    LOG.debug("Exiting initialise.");
    }
    mInitialised = true;
    }
```

NEW VERSION

```
...
    LOG.error(msg);
    }
    }

    // Create a PlatformConfigurator to use for Base64 encoding
    new uk.org.ogsadai.common.wsrf.WSRFPlatformConfigurator();

    if (LOG.isDebugEnabled()) {

    LOG.debug("Exiting initialise.");
    }
    mInitialised = true;
    }
```

2. Editing WSRFDataService.java

In the two contructor methods in `uk/org/ogsadai/client/toolkit/wsrf/WSRFDataService.java`, make the following changes:

OLD VERSION

```
public WSRFDataService(String handle)
throws MalformedURLException
{
mService = new WSRFDataServiceStub(handle);
mTransport = null;
mResource = null;
}

/**
*
* @param handle
* @param id
*/
public WSRFDataService(String handle, ResourceID resourceID)
throws ServiceCommsException, MalformedURLException {
mService = new WSRFDataServiceStub(handle);
mResource = resourceID;
}
```

NEW VERSION

```
public WSRFDataService(String handle)
throws MalformedURLException
{
mService = new WSRFDataServiceStub(handle);
mTransport = null;
mResource = null;
new uk.org.ogsadai.common.wsrf.WSRFPlatformConfigurator();
}

/**
*
* @param handle
* @param id
*/
public WSRFDataService(String handle, ResourceID resourceID)
throws ServiceCommsException, MalformedURLException {
mService = new WSRFDataServiceStub(handle);
mResource = resourceID;
new uk.org.ogsadai.common.wsrf.WSRFPlatformConfigurator();
}
```