



QCDGrid2 Metadata Catalogue Web Service Browser Design

Project Title: QCDGrid2

Document Title: QCDGrid2 Metadata Catalogue Web Service Browser Design

Document Identifier: QCDGRID_MDWS_BROWSER_DESIGN

Document Filename: QCDGrid2-MDC-ILDGBrowser-Design.doc

Distribution Classification: Public

Authorship: Daragh Byrne

Approval List: QCDgrid Development Team

Distribution List: UKQCD Collaboration

Document History:

<i>Personnel</i>	<i>Date</i>	<i>Summary</i>	<i>Version</i>
DJB	9 June 2006	Version 1.1	1.1
DJB	16 August 2006	Version 1.2	1.2

Contents

1	Introduction	3
1.1	The QCDgrid project.....	3
1.2	Document purpose	3
2	Design goals	4
3	Requirements	5
4	Features of the current browser	6
5	Proposed design - ILDG browser	7
5.1	Modes of operation	7
5.2	Refactoring the query execution code	7
5.2.1	Current implementation	7
5.2.2	Desired modifications	9
5.3	Making the mode of the browser configurable	12
5.4	Security	13
	References	14

1 Introduction

1.1 The QCDgrid project

The QCDgrid project is a core activity of UKQCD, a collaboration of UK academics and researchers that aims to procure and jointly exploit computing facilities for lattice field theory (commonly referred to as Lattice QCD) calculations. The primary aim is to increase the predictive power of the *Standard Model of elementary particle interactions* through numerical simulation of *Quantum Chromodynamics*. Such numerical simulations produce significant amounts of data and the purpose of the QCDgrid project is to provide software and supporting infrastructure that simplifies the management, storage, and manipulation of this data.

In the first three years of the project (2002 – 2004), software engineers at EPCC developed QCDgrid—a data management system that combines the distributed resources of the collaborators into a robust facility called the *UKQCD Grid*. The result is a multi-terabyte storage facility over six UK sites at: Edinburgh (including the University of Edinburgh Advanced Computing Facility), Liverpool, RAL, Southampton, and Swansea. Glasgow is also a member of the consortium.

The facility is based on commodity hardware and open-source software. The hardware consists primarily of high specification, PC-based servers running the Linux operating system and managing large RAID storage arrays. On top of this infrastructure, the QCDgrid software (built using components from the Globus Toolkit, EGEE application stack, and an XML database) provides *Datagrid* management and user functionality – furnishing a simple and intuitive environment that hides the complexities of the underlying grid and presents a standard file system to the user. It incorporates a robustness metric that automatically disperses datasets across the grid, providing a resilience that ensures data is not affected by the loss of one (or possibly more) storage nodes. Security is leveraged from the Globus Toolkit, based on X.509 digital certificates issued by an approved Certificate Authority. The result is a reliable, secure data management system.

UKQCD is an important contributor to the International Lattice Data Grid (ILDG), a group of like-minded scientists around the world who aim to share their data to accelerate scientific progress in the field of Lattice QCD. The ILDG was initiated in 2002 and, at the time of writing, has significant representation from research groups in Australia, France, Germany, Italy, Japan, UK and USA.

The ILDG infrastructure is being assembled as a web services layer that will aggregate the particular resources of contributing collaborations (for example, the UKQCD Grid) for the benefit of the wider community. To achieve its objectives, ILDG has established working groups to:

- Facilitate data sharing, through the standardisation of the form and content for Lattice QCD scientific data and associated metadata.
- Produce a set of specifications that define an architecture for an international *Grid of Grids* for Lattice QCD.

1.2 Document purpose

One of the core components of the ILDG architecture is the metadata catalogue web service. This web service allows ILDG members to search the metadata catalogue of each collaboration. The specification of this web service is addressed in [2].

The web service specification has already been implemented by a number of the members of ILDG (UKQCD, USQCD, LQCD, and JLDG). A need for client tools that are capable of interacting with these web service implementations is clear. The QCDgrid project has produced a browser application that interacts directly with the UKQCD eXist XML database (which is the back-end storage mechanism for the UKQCD web service implementation). This document investigates how this browser tool could be adapted to interact with the web service implementations.

2 Design goals

This document seeks to explore the following areas:

- The current features of the existing browser that are desirable in an ILDG version of the browser;
- The current features of the existing browser that are undesirable/unnecessary in an ILDG version of the browser;
- How existing features need to be modified to interact with web services instead of XML databases;
- How unnecessary existing features could be “turned off” in the ILDG version of the browser.

3 Requirements

The following are the high-level requirements for a new version of the browser:

- The browser must assist users in constructing queries to be executed against metadata catalogues;
- The browser must allow such queries to be executed against a number of metadata catalogue web services;
- The browser must be capable of presenting results from queries across multiple sites;
- In the future, the browser may allow retrieval of QCD data associated with results (however this is beyond the scope of this particular document).

4 Features of the current browser

At present, the browser has the following features:

- Submit QCD data and metadata (about configurations and ensembles) to the UKQCD data grid system (data grid and metadata catalogue);
- Allows the user to construct queries to be issued against configuration or ensemble metadata documents, stored in the metadata catalogue, using a graphical query builder interface;
- Execute queries directly against the metadata catalogue (eXist database driver);
- Display the results of these queries;
- Retrieve data associated with query results;
- Administer the data grid (when the user has sufficient privilege) – although this appears as a feature of the browser, it is not actually implemented yet.

The user can specify the location of the metadata catalogue to use via a popup at start-up, or via the *.browserrc* configuration file. Security is not implemented at the moment.

When the browser is run, the user is presented with a screen that allows access to the rest of the functionality via three buttons: “Search Metadata Catalogue”, “Submit Data” and “Administer Grid” (as noted above, the administration functions have not yet been implemented). Clicking on one of these allows access to the appropriate functionality.

5 Proposed design - ILDG browser

5.1 Modes of operation

We propose that the existing browser application be partly rewritten in such a manner that it can operate in either “ILDG mode” or “UKQCD mode”. The mode will be determined by use of a configuration file, and the browser will alter its runtime behavior (for example, disabling particular functionality) as needed.

The modifications to be made to the browser to implement this dual-mode capability are:

- Refactoring the code that runs user queries against the catalogue in a manner that will allow queries to be run both via the web service and in the existing manner;
- Making the mode of operation configurable. Specifically, this should address:
 - Changing the mechanism by which the queries are issued depending on the operation mode;
 - Disabling/enabling the browser features depending on the mode.

5.2 Refactoring the query execution code

5.2.1 Current implementation

At present the browser application interacts with the metadata catalogue for the following reasons:

1. To download information used to configure the browser, specifically information about the query types supported and the name of schema files used;
2. To issue XPath queries. Distinction between configuration and ensemble metadata is made at the level of running an individual query - the browser does have an understanding of the query types that are allowed via the `QueryTypesManager` class. Some queries are specific to QCDgrid (QCDgrid specific queries); others are general queries that would be issued by any client (user queries);
3. To retrieve metadata documents. There are generic `getDocument` and `getSchema` methods that allow this – again, there is no distinction between document types. Documents are currently accessed by file name rather than LFN or URI.

A detailed analysis of the browser query code is presented in order to highlight the changes that are required. The nature of each interaction is noted. `CONFIGURATION` means an interaction that downloads browser configuration information. `QUERY-USER` is used to denote an interaction that issues queries constructed by the user. `QUERY-QCDGRID-SPECIFIC` is used to denote an interaction that issues a query specific to QCDgrid activities (e.g. checking that a particular LFN is represented in the UKQCD catalogue). `DOCUMENT` is used to denote downloading a metadata document.

`ExistDatabaseDriver` is used in the following classes.

MainGUIFrame

- Instantiates via constructor;
- Uses `getQueryNotations` to retrieve the supported query notations (i.e. XPath/SQL – only XPath supported now) (`CONFIGURATION`)
- Interacts with `XPathQueryBuilder` to issue queries (`QUERY-USER`).

QueryType

- Uses the `getDriver` static method to obtain an instance of the driver;
- Uses the `getSchema` method to download the schema associated with a query type (CONFIGURATION);

QueryTypeManager

- Uses the `getDriver` static method to obtain an instance of the driver;
- Uses the `getSchema` method to download the `QueryTypes.xml` file, which contains details of the query types supported (CONFIGURATION);

This class is used in `MainGUIFrame`, `QueryTypeSelector` and `XPathQueryBuilder`.

QCDgridFrame

- Uses the `getDriver` static method to obtain an instance of the driver;
- Uses `getDocument` to retrieve document contents by name (DOCUMENT).
- This class is used by `QCDgridResultHandler`.

QCDgridMetadataClient

- Instantiated directly using a string representing the database location;
- Runs queries to see if particular LFNs are represented in the catalogue (uses `runQuery`) (QUERY-QCDGRID-SPECIFIC);
- Runs queries to see if particular URIs are represented in the catalogue (uses `runQuery`) (QUERY-QCDGRID-SPECIFIC);
- Provides a general `getSchema` method which replicates the functionality of the `getSchema` method provided by `ExistDatabaseDriver` (CONFIGURATION);
- Provides a general `getDocument` method which replicates the functionality of the `getDocument` method provided by `ExistDatabaseDriver` (CONFIGURATION);

`QCDgridMetadataClient` is used in the following classes:

SchemaInfo

- Passed an instance in the constructor, which is never used – this should be fixed;

QCDgridSubmitter

- Uses many of the methods to submit metadata to the catalogue.

ConfigurationMetadataValidator

- To get the schema used to carry out validation operations;
- To check if LFNs are represented in the MDC using `isLFNRepresentedInMDC` (QUERY-QCDGRID-SPECIFIC);
- To check if ensemble URIs are represented in the MDC using `isEnsembleRepresentedInMDC` (QUERY-QCDGRID-SPECIFIC);

EnsembleMetadataValidator

- To get the schema used to carry out validation operations (CONFIGURATION);

- To check if ensemble URIs are represented in the MDC using `isEnsembleRepresentedInMDC` (QUERY-QCDGRID-SPECIFIC);

MetadataValidator

- Maintains a reference for subclasses.

Any proposed refactoring of the access classes will require that at least the above functionality be maintained.

XPathQueryBuilder

- Uses an instance of the `DatabaseDriver` interface to issue queries. In this case, the instance is always of type `ExistDatabaseDriver` (QUERY-USER).

This class is used in the `Configuration` and `MainGUIFrame` classes.

5.2.2 Desired modifications

The browser needs to be modified as follows:

- The main query-related features of the browser, such as the query builder and the way it configures itself, must be maintained;
- XPath queries must be executable against multiple data sources, wrapped by instances of the ILDG metadata catalogue web service, instead of a single database server.

5.2.2.1 Running user queries

There are two possible strategies to manage the changes noted above:

- A complete rewrite of all the data access parts of the browser to use the web service;
- A rewrite of the code such that things like configuration and UKQCD-specific queries remained the same but that queries and document fetches are done via the web service. This recognises the fact that there are two types of query (QCDGRID-SPECIFIC and USER as noted above).

It is anticipated that the second option will require less work. In any event, direct interaction with the UKQCD metadata catalogue will be necessary so there is no point in undoing all of the work that has already been done.

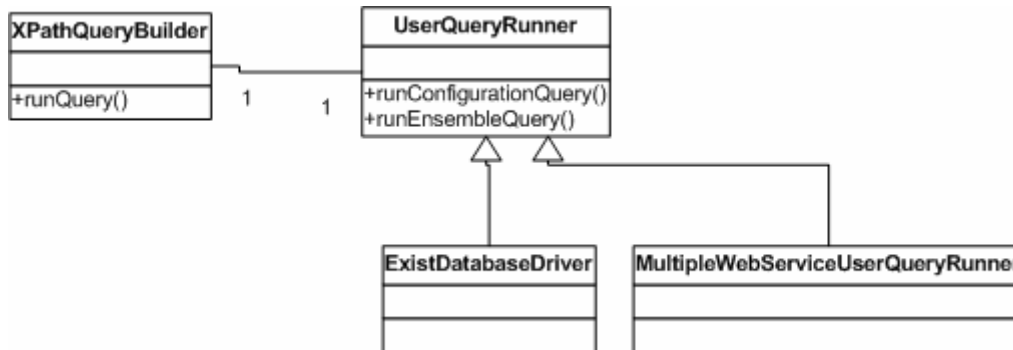
We propose that an interface called `UserQueryRunner` be introduced to represent situations where user queries are issued. An implementation of this that aggregates queries across multiple web services interfaces will be created (`MultipleWebServiceUserQueryrunner`) (this will be used in the ILDG browser). An implementation of `UserQueryRunner` that passes calls to `ExistDatabaseDriver` in the existing manner will be created (most likely by having `ExistDatabaseDriver` implement `UserQueryRunner`). `UserQueryRunner` will have two methods: `runConfigurationUserQuery` and `runEnsembleUserQuery`.

The main changes need to take place in the `runQuery` method of the `XPathQueryBuilder` class. This method needs to be changed in the following manner:

- The method currently takes an instance of the `DatabaseDriver` abstract class; this should be changed to `UserQueryRunner`;

- The method needs to take a decision on which of the methods on `UserQueryRunner` to call. Currently an instance of `XPathQueryExpression` is used to contain information about the query to be executed. This may need to be modified to have a getter that allows the query type (ensemble or configuration) to be determined.

The following class diagram attempts to summarise the design.



The combine method of `XPathQueryBuilder` may also need to be altered so that the query type is included in the instance of `XPathQueryExpression` it returns.

At present, the browser uses the `uk.ac.ed.epcc.qcdgrid.metadata.catalogue.QueryResults` class to handle query results. The result handler classes deal with objects of this type. This presents a difficulty, as the web service returns results wrapped in an instance of `org.lqcd.ildg.mdc.QueryResults`. In addition, the internal `QueryResults` class can expose the results via the `XMLDB ResourceSet` class, which makes no sense when the query results come from web services. It is proposed that the following be carried out to allow rectify the situation:

- `uk.ac.ed.epcc.qcdgrid.metadata.catalogue.QueryResults` is to be renamed `QCDgridQueryResults` to reflect the fact that it represents the results of a query against the QCDgrid catalogue (i.e. the eXist database server). This method is only exposed in order to make the GUI more responsive (see `QCDgridFrame` for details);
- The `individualResults` method of `QCDgridQueryResults` should be renamed to `resultsAsXMLDBResourceSet`, as this is what it actually returns;
- The constructor of `QCDgridQueryResults` should allow the type of query issued to be set – i.e. ensemble query or configuration query (this will be useful when retrieving result documents);
- A new method, `ResultInfo [] getIndividualResultInfoObjects` will be introduced that allows the query results to be exposed as `ResultInfo` objects (see below).

In addition, a new `ILDGQueryResults` class will be created to represent the results of querying against multiple metadata catalogue web services. This class will directly subclass `QCDgridQueryResults` and will override all the public methods as appropriate. In particular, `resultsAsXMLDBResourceSet` will be overridden to throw an exception if it is called, as it is not appropriate for this class as noted above.

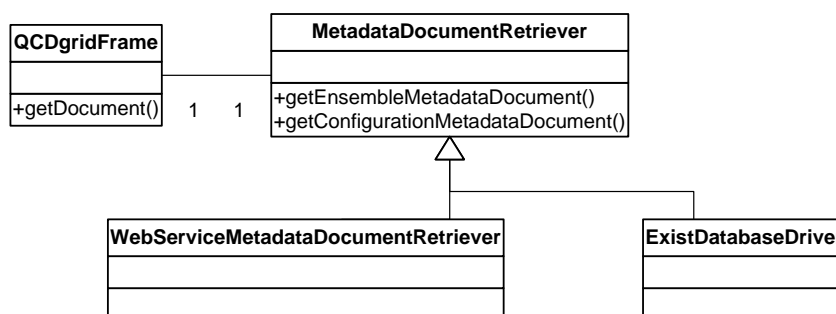
A `ResultInfo` class will be created that stores an atomic query result. This will have two public properties:

- `Result`, which is the actual query result (in the UKQCD mode, the XMLDB resource ID relating to the query result, in the ILDG mode, the ensemble URI or gauge configuration LFN returned by the query);

- Source, which is the source of the result (in the UKQCD mode, the string “UKQCD”, in the ILDG mode, a URL representing the web service from which the query result was returned – this can be used later for retrieving the appropriate metadata document);

In order to leverage the above changes, the following modifications need to be made:

- In `QCDgridListBuilder`, which is responsible for populating the GUI components used to display the results of the query, the `run` method needs to be modified to react differently depending on whether the query results object it interacts with is of type `QCDgridQueryResults` or `ILDGQueryResults`. In case of the former, it should continue taking the action it takes now. In case of the latter, it should use the `getIndividualResultInfoObjects` method to populate the list;
- The `getDocument` method of `QCDgridFrame` needs to be modified to react differently when the query result it deals with is of type `QCDgridQueryResults` or `ILDGQueryResults`. In both cases, an instance of a class that implements the `MetadataDocumentRetriever` interface should be used to retrieve the document. `ExistDatabaseDriver` will be used in the UKQCD case; otherwise a new class called `WebServiceMetadataDocumentRetriever` (that uses the auto-generated Axis stubs to download the appropriate metadata documents) will be used. See the following class diagram for further details;



The above changes should allow the existing query handler classes to be used to handle the ILDG query results in a similar manner to how they are handled now.

MultipleWebServiceUserQueryRunner

`MultipleWebServiceUserQueryRunner` will load a local copy of the `ILDG Services.xml` file when it is instantiated. This file will be cached locally. It is up to the user to make sure they have an up to date copy. The file will be parsed to gain information about the location of the ILDG services

The implementations of `runConfigurationQuery` and `runEnsembleQuery` will start threads that query each of the available metadata catalogues, using the Axis-generated Java stubs to execute the queries. A query to an individual web service will be abstracted in the `MetadataCatalogueWebServiceQuery` class, which will implement `Runnable`. Exceptions will be handled in each thread; a `setError` method will allow errors to be noted without throwing an exception.

5.2.2.2 Running QCDgrid specific queries

The features that run QCDgrid specific queries will be disabled. This code will remain untouched, barring a number of refactorings which we propose here.

The `ExistDatabaseDriver` class acts as a QCDgrid specific interface to an eXist database driver. It should be renamed to `QCDgridExistDatabase`, as this reflects its role within both the ILDG and QCDgrid browser configurations more accurately.

The `getSchema` method will be renamed to `getBrowserConfigurationDocument` to reflect the fact that this is what it does.

5.2.2.3 Making the mode of the browser configurable

The following need to be addressed:

- Changing the mechanism by which the queries are issued depending on the operation mode;
- Disabling/enabling the browser features depending on the mode.

We propose that each of the operating modes of the browser will be packaged separately. A configuration file in each distribution will enable and disable browser features as required.

As noted, there are three main features in the current browser:

- Search metadata catalogue;
- Submit metadata;
- Administer grid (although as pointed out, this is a placeholder feature and has not really been implemented yet).

At present, most of the browser configuration is handled by the class `uk.ac.ed.epcc.qcdgrid.browser.Configuration.Configuration`. This uses the `.browserrc` file to store information such as the queries that users have saved; the current selected result handler class and so on. It is probably more appropriate to use a separate file to store feature configuration information, since `.browserrc` is not created until the browser has been run for the first time, and we do not wish to overwrite existing `.browserrc` files. This new file will be called `browser.properties` and will live in the root of the distribution. It will use the Java properties file format (key-value pairs), and the Java Properties class will be used to parse the file. For the ILDG configuration the file will contain the following:

```
browsertype = ILDG
```

For the QCDgrid configuration `browser.properties` will contain the following.

```
browsertype = UKQCD
```

The value of the `BROWSER_HOME` environment variable will be used to locate this file. The default mode of the browser in the absence of this file will be “UKQCD”.

We propose a `uk.ac.ed.epcc.qcdgrid.browser.Configuration.BrowserFeatureConfiguration` class that has a `getBrowserType` method. This class will be used in the `QCDgridGUI` class to determine which top-level buttons to display. It will also be used in the `MainGUIFrame` to determine which type of `UserQueryRunner` to instantiate and pass to `XPathQueryBuilder` (i.e. `HomeMetadataCatalogue` in the case of the QCDgrid browser or `MultipleWebServiceQueryRunner` in the case of the ILDG browser).

5.2.2.4 Security

This section has been removed from the public version of the design document, since it contains information that may compromise the integrity of the metadata catalogue. [MGB, 19/AUG/06]

References

- [1] QCDGrid: Probing the building blocks of matter with the power of the Grid, QCDgrid Project Homepage available on-line at <http://www.gridpp.ac.uk/qcdgrid/>.
- [2] QCDgrid2 Metadata Catalogue Web Service Specification, available from the QCDgrid team on request.