



WP 4.2 Functional specification of ILDG File Catalogue

Project Title: QCDgrid

Document Title: WP 4.2 Functional specification of ILDG File Catalogue

Document Identifier: QCDGRID2-FC-WS-SPEC

Document Filename: QCDGrid2-FCWS-Specification.doc

Distribution Classification: Public

Authorship: Radoslaw Ostrowski, George Beckett

Approval List: QCDgrid Project Management Board

Distribution List: Public

Document History:

<i>Personnel</i>	<i>Date</i>	<i>Summary</i>	<i>Version</i>
MGB/RHO	5/JAN/07	First release.	1.0
MGB	15/JAN/07	Minor updates.	1.0.1

Contents

1	Introduction	3
1.1	The QCDgrid Project	3
2	Background to the ILDG file catalogue.....	4
3	Specification	6
3.1	Access models	6
3.2	Service functionality	8
3.3	Security considerations.....	8
4	Conclusions.....	9
5	References.....	10
	Appendix A	11

1 Introduction

1.1 The QCDgrid Project

The QCDgrid project [12] is a core activity of UKQCD [11], 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 seven sites at: University of Columbia, University of Edinburgh (including the UoE Advanced Computing Facility), University of Liverpool, Rutherford Appleton Laboratories (RAL), University of Southampton, and University of Wales Swansea. The University of 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 [2], EGEE application stack [1], 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) [5], a group of like-minded scientists, working 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 by UKQCD 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 resources of each contributing collaboration (that is, regional grid such as the UKQCD Grid) for the benefit of the wider community. To achieve its objectives, ILDG has established two working groups:

- Metadata Working Group – to facilitate data sharing, through the standardisation of the format and content for Lattice QCD scientific data and associated metadata.
- Middleware Working Group – to produce a set of specifications that define an architecture for an international *Grid of Grids* for Lattice QCD.

This document describes the work of the Middleware Working Group to establish the form and function for the *ILDG File Catalogue*, a key component of the ILDG infrastructure. In Section 2, we explain the purpose of the file catalogue in the context of the wider infrastructure. Then in Section 3, we summarise key features of its specification and highlight any points of particular note for UKQCD. In Section 4, we draw together conclusions, based on the work to date, and establish the next steps towards a functioning implementation.

2 Background to the ILDG file catalogue

In a *grid* context, a file catalogue is a registry that binds unique file identifiers (commonly referred to as Logical Filenames (LFNs)) to one or more instances of (internet) locations, or URLs, where a copy of the specific file exists.

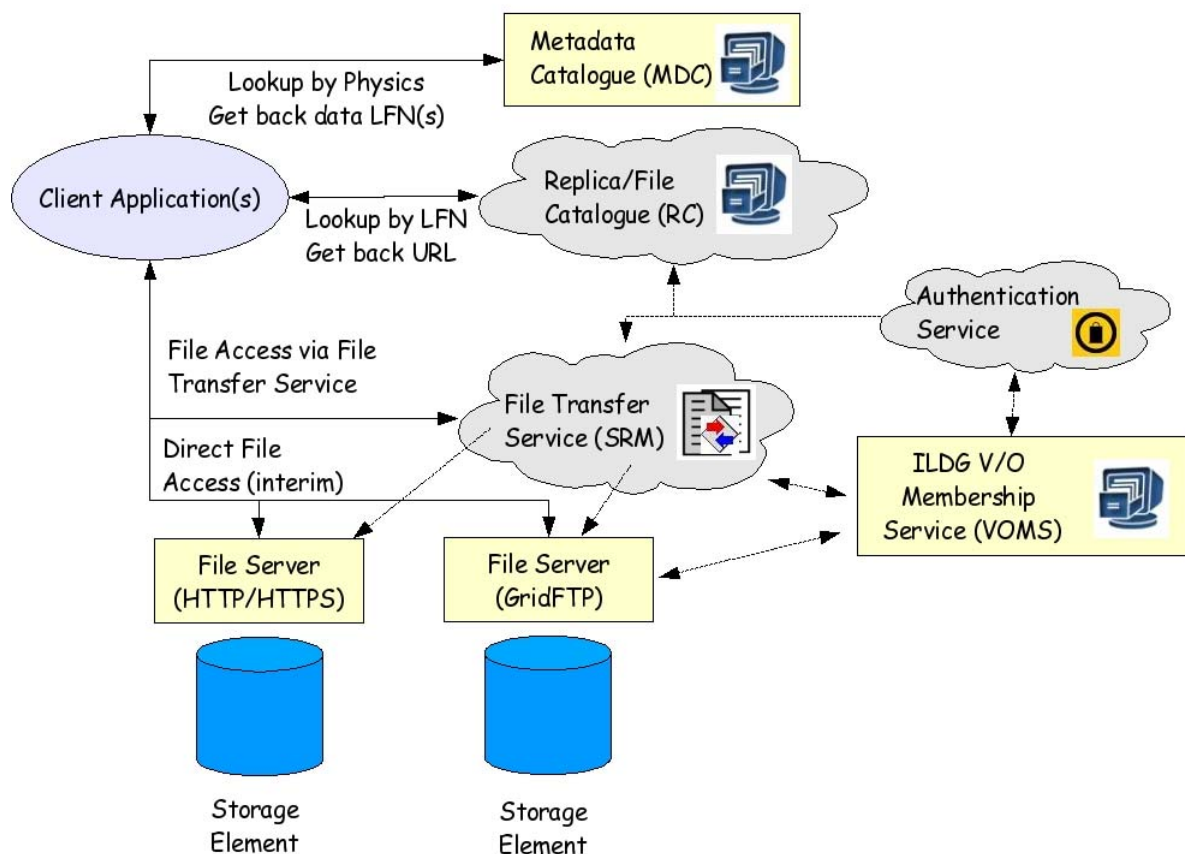


Figure 1: Modular overview of the ILDG infrastructure

A file catalogue is a core component of the ILDG infrastructure (see Figure 1) that is used to track the location of Lattice Gauge Configuration datasets within the storage resources provided by the regional grids. It is most powerful when combined with the ILDG Metadata Catalogue (see QCDgrid project deliverables from Work Package 3 [12]), which lists the lattice gauge configurations that are available, described using meaningful and standardised physics terminology and marked up using QCDML. The file catalogue contributes to the ILDG’s “Search and Retrieval” use cases [10][13].

At the time of writing, each regional grid has implemented – or is in the process of implementing – a local file catalogue to manage the datasets owned by the particular collaboration. Some groups, such as UKQCD, CSSM (Australia) and LDG (Germany), have elected to use a third-party catalogue: UKQCD and CSSM use the Globus Replica Location Service [2], while LDG use the LCG File Catalogue Service [1]. Other collaborations, such as USQCD (USA) and JLDG (Japan), have elected to build their own file catalogues from scratch.

Both of these approaches (using a third-party catalogue or writing one’s own from scratch) have advantages for the individual collaborations. However, for the ILDG as a whole, the different approaches imply that member collaborations present incompatible interfaces and provide different functionalities to the potential user. This is not a new situation for the middleware working group: it has previously been encountered and successfully overcome during the specification of the ILDG Metadata Catalogue. As then, the middleware working group propose to overcome these

incompatibilities by introducing a web service layer that will present a standard interface on top of the regional catalogues.

The determination of the form and function of this web service has been a focus of discussions within the working group over the previous 12 months. The QCDgrid project team, as representatives for UKQCD to the middleware working group, have been heavily involved in this process through both face-to-face meetings (Japan, October 2005; Germany, July 2006; and USA, December 2006) and monthly teleconferences of the working group. During the December 2006 meeting, the working group arrived at an agreed specification for the file catalogue that is to form the basis for the design and implementation activities of the regional grids.

3 Specification

As noted in Section 2, a first version of the specification for the file catalogue web service has been agreed [6]. In addition, a WSDL description is has been established (reproduced in Appendix A) and a prototype implementation has been developed by the LDG group, with contributions from UKQCD and USQCD. In this section, we summarise the key features of the specification and highlight any specific considerations for UKQCD.

3.1 Access models

Unlike the metadata catalogue, which has been specified by the ILDG Board to be generally accessible to users without the need for authentication, the file catalogue is subject to more complex authentication and authorisation requirements. Each regional grid has a different policy with regard to the security of their contribution to the catalogue. Because of this, it has been determined that the file catalogue web service needs to offer two different modes of access:

- **Unsecured** – For this access mode, the HTTP protocol is used for transport of messages between the server and client. The service does not authenticate the user, which implies that anonymous and public access is available to the catalogue. Furthermore, no assertion is given to the client that the correct server is contacted. This mode of access is only suitable for regional grids for which the underlying file catalogue does not require user authentication, and for which everyone is permitted to read the contents of the catalogue.

Advantages:

- No security-related overhead for setup, usage, and administration of the service.
- No requirement on client to support a security infrastructure (for example, Globus Security Infrastructure (GSI)) and consequently no need for the user to possess or provide an X.509 certificate.

Disadvantages:

- Cannot be used if the underlying file catalogue, on the server side, requires user authentication.
- There is no assurance for the client that it is contacting the intended server.
- **Secure** – The service uses the Secure Socket Layer (SSL) protocol (access over HTTPS) which encodes and decodes messages on both the client- and server-side, plus sends only encrypted messages over the network. The service requires mutual user and server authentication, which is carried out in the initial stages of the connection, commonly referred to as the *handshake*, when the identities of both client and server are confirmed basing on their X.509 certificates. An authorisation mechanism may also be implemented providing the capacity to attribute different permissions for different users.

Advantages:

- Required if the underlying file catalogue needs user authentication.
- Server is able to authenticate the client/user and perform an optional authorisation check on each user request.
- Client is able to confirm that it is talking to the intended server.

Disadvantages:

- There is an overhead in setting up, using and administering such a service.

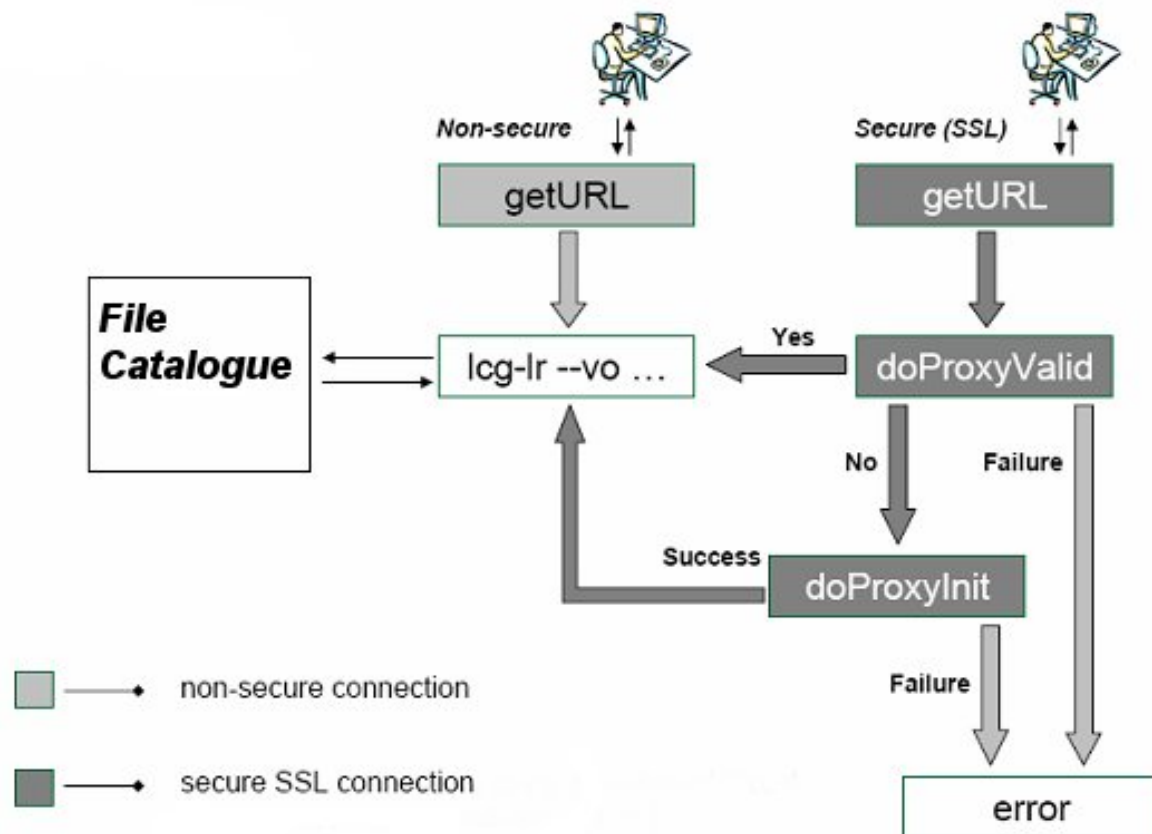


Figure 2: A solution to File Catalogue web service [8].

Figure 2 illustrates the flow of a typical query in both approaches:

- The left-hand client, which employs an unsecured connection over HTTP, queries the web service using the `getURL` method with the Logical Filename (LFN) for the data they are interested in as an argument. Assuming the service supports unsecured access, the request is actioned by spawning a query to the underlying file catalogue (in this illustration, a command-line invocation of the LCG File Catalogue) and the response (a, possibly empty, list of URLs representing addresses from which the data can be obtained) is returned to the client.
- The right-hand client accesses the web service in a secure manner, also using the `getURL` method (though this time over HTTPS), again passing the LFN for the data they are interested in as an argument. Assuming the service supports secure access, the client and server perform mutual authentication and open an SSL connection. For the authentication step to be successful, the user must have a valid X.509 certificate available on their local machine. The client must also delegate trust to the service to allow the web service to perform an authenticated query to the underlying file catalogue on behalf of the user. Once a chain of trust has been established, a query is made to the underlying file catalogue and the response (a list of URLs) is returned to the user, as for the unsecured access mode.

Each regional grid may implement either the unsecured access model, the secure access model, or both. As noted in Section 2, within UKQCD, the underlying file catalogue is the Globus Replica Location Service. The configuration of the UKQCD Grid demands authenticated access to the file catalogue using a valid X.509 certificate. In addition, an authorisation of the users could be implemented using “grid-mapfile” access control lists [3]. If the authorisation step was implemented, UKQCD would only support the secure access model for the ILDG File Catalogue web service.

3.2 Service functionality

A detailed description of the web service functionality is provided in the specification document [5]. For the secure access mode, the specification implies a trust delegation model in which the user uploads a proxy certificate to the remote service. It defines five operations: `getURL`, `proxyInit`, `proxyInfo`, `proxyDestroy` and `getVersion`. The most important operation is `getURL`, which takes as input a Markov Chain URI and returns a list of URLs from which the data may be available. Additionally, the following functions are provided for proxy management and web service maintenance:

- `proxyInit` – transfers a copy of a user’s proxy certificate to the server hosting the service. Note that the proxy certificate is not validated: it is permissible for the user to send an expired or invalid proxy.
- `proxyInfo` – returns information about the user’s proxy certificate stored at the server.
- `proxyDestroy` – destroys the user’s proxy certificate stored at the server.
- `getVersion` – returns version information for the file catalogue web service.

3.3 Security considerations

Delegation of authority by a user to a web service has significant security implications, both for the user and the service. These implications need to be considered carefully and, to this end, the working group have consulted with a number of security experts including: David Bianco, a cyber-security analyst from Jefferson Laboratories, USA; plus Rachana Ananthakrishnan and Charles Bacon from the Globus Security team.

At the time of writing, the proposed approach to secure access to a file catalogue requires the user to upload a valid proxy certificate to the web service, over an encrypted channel, using the proxy management functionality outlined in Section 3.2. This form of proxy delegation has an associated risk since the proxy is a full, though time-limited, representation of a user. Because of this, it must be appropriately protected in storage on the remote server.

The impact of the risk can be reduced by creating only a short-lived proxy, which is valid for little longer than the duration of the user’s interaction with the service. This approach is similar to the strategy followed by Kerberos [7] (but using a public-key infrastructure rather than Kerberos tickets).

The concerns about storage on the remote server may, in fact, be contained by a component such as the Globus Toolkit Delegation Service, which implements a standard called WS-Trust [4]. However, at the time of writing, WS-Trust is only supported by the Globus Toolkit (and specifically Version 4 of the toolkit) making it problematic to deploy for collaborations that use other forms of middleware such as LCG/gLite.

A middleware agnostic alternative could be provided by a MyProxy service [9]. This is a service that hosts proxy credentials on behalf of a user in a secure manner and manages delegation of trust to other services on their behalf. This approach would necessitate the deployment and maintenance of additional services within ILDG, something which is considered undesirable given the demands on resources.

4 Conclusions

A focus of the ILDG Middleware Working Group's activity during 2006 has been the form and function of the ILDG File Catalogue. This report has summarised the agreed specification for the first version of the file catalogue that supports the user to determine the location (URL) of lattice gauge configurations, identified by their Markov Chain URI. The working group are aiming to build on the experience gained during the successful specification and implementation of the ILDG Metadata Catalogue that has, at the time of writing, been deployed by five collaborations (CSSM, JLDG, LDG, UKQCD, and USQCD).

The specification of the file catalogue needs to accommodate more complex security requirements than the metadata catalogue and this has yielded some challenges in determining a robust process for trust delegation that is independent of the choice of middleware. A solution has been proposed that will be considered in more detail by the QCDgrid project team in the next work package (WP4.3) before implementation.

The working group have decided to combine resources to design and implement a common web service solution for all grids. This is a different approach to that employed for the design and implementation of the metadata catalogue. In that case, each collaboration completed their own implementation. It is hoped that a shared design and implementation will reduce development time and simplify the process of adoption for future members in ILDG.

Once the development of the file catalogue web service is complete, the QCDgrid project team will extend its ILDG Browser client (Figure 3), to interface with the file catalogue, completing the next step of the ILDG integration roadmap.

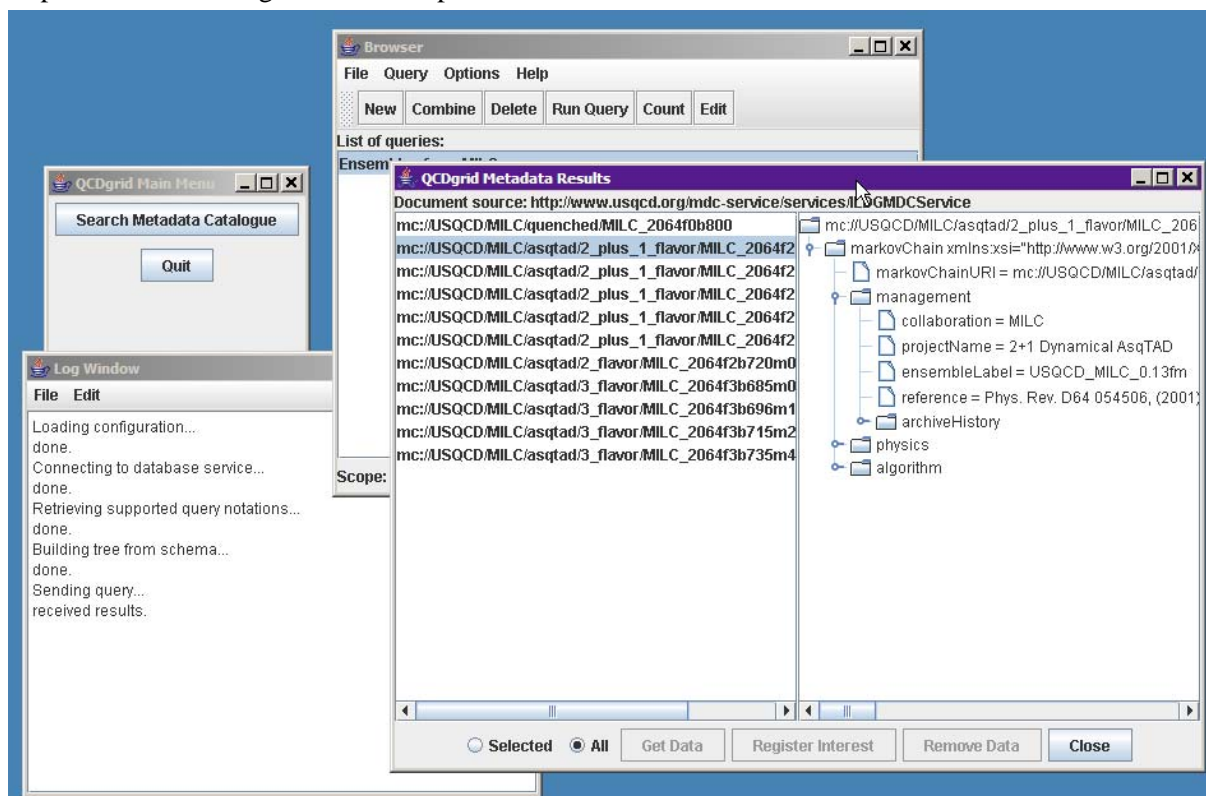


Figure 3: Screenshot of the ILDG browser showing the output from a query to the ILDG Metadata Catalogue for datasets generated by the US MILC collaboration.

5 References

- [1] EGEE, *gLite – Lightweight Middleware for Grid Computing*. Project homepage at <http://glite.web.cern.ch/glite/> (2006).
- [2] Globus Alliance, Globus Toolkit. Homepage at <http://www.globus.org/>.
- [3] Globus Alliance, *GT4.0: Security*. Part of the documentation for the Globus Toolkit 4.0 Release Manuals at <http://www.globus.org/toolkit/docs/4.0/>.
- [4] IBM, et al., *Web Services Trust Language (WS-Trust)*, 2006.
- [5] International Lattice Data Grid. Homepage at <http://www.lqcd.org/ildg/>.
- [6] ILDG Middleware Working Group, *ILDG FC-WS Operations (0.1)*, July 2006.
- [7] MIT, *Kerberos: The Network Authentication Protocol*. Homepage at <http://web.mit.edu/Kerberos/>, December 2006 .
- [8] D. Melkumyan. *Proxy Delegation: The prototype of the service “getURL”*, July 2006.
- [9] NCSA, *MyProxy Credential Management Service*. Homepage at <http://grid.ncsa.uiuc.edu/myproxy/>.
- [10] M. Sato, *Lattice QCD Data Grid Middleware: The Metadata Catalogue*, Presentation to ILDG Workshop (2004). Electronic copy available at <http://www.rcpp.tsukuba.ac.jp/workshop/ILDG-4/pdf/MitsuhisaSato.pdf>.
- [11] UKQCD Collaboration. Homepage at <http://ukqcd.epcc.ed.ac.uk/>.
- [12] UKQCD, *QCDgrid: Probing the building blocks of matter with the power of the Grid*, QCDgrid project homepage at <http://www.gridpp.ac.uk/qcdgrid/>.
- [13] T. Yoshie, *Status of ILDG Activity*, Presentation to ILFTN (Edinburgh, 2005). Electronic copy available from http://www.nesc.ac.uk/talks/464/Session7/ILFTN2_yoshie.pdf.

Appendix A

Reproduction of WSDL description for the ILDG File Catalogue [6].

```
<?xml version="1.0" encoding="UTF-8"?>
<wsdl:definitions targetNamespace="urn:QCDgrid" xmlns:apachesoap="http://xml.apache.org/xml-
soap" xmlns:impl="urn:QCDgrid" xmlns:intf="urn:QCDgrid"
xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/"
xmlns:tns1="http://responses.fc.qcdgrid.epcc.ed.ac.uk"
xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/"
xmlns:wsdlsoap="http://schemas.xmlsoap.org/wsdl/soap/"
xmlns:xsd="http://www.w3.org/2001/XMLSchema">
<!--WSDL created by Apache Axis version: 1.4
Built on Apr 22, 2006 (06:55:48 PDT)-->
  <wsdl:types>
    <schema targetNamespace="http://responses.fc.qcdgrid.epcc.ed.ac.uk"
xmlns="http://www.w3.org/2001/XMLSchema">
      <import namespace="urn:QCDgrid" />
      <import namespace="http://schemas.xmlsoap.org/soap/encoding/" />
      <complexType name="Response">
        <sequence>
          <element name="queryTime" nillable="true" type="xsd:string"/>
          <element name="statusCode" nillable="true" type="xsd:string"/>
        </sequence>
      </complexType>
      <complexType name="ResponseProxyStatus">
        <complexContent>
          <extension base="tns1:Response">
            <sequence>
              <element name="proxyStatus" nillable="true" type="xsd:string"/>
            </sequence>
          </extension>
        </complexContent>
      </complexType>
      <complexType name="ResponseGetURL">
        <complexContent>
          <extension base="tns1:ResponseProxyStatus">
            <sequence>
              <element name="numberOfResults" type="xsd:int"/>
              <element name="results" nillable="true" type="impl:ArrayOf_xsd_string"/>
            </sequence>
          </extension>
        </complexContent>
      </complexType>
      <complexType name="ResponseString">
        <complexContent>
          <extension base="tns1:Response">
            <sequence>
              <element name="string" nillable="true" type="xsd:string"/>
            </sequence>
          </extension>
        </complexContent>
      </complexType>
      <complexType name="ResponseProxyInfo">
        <complexContent>
          <extension base="tns1:ResponseProxyStatus">
            <sequence>
              <element name="proxyInfo" nillable="true" type="xsd:string"/>
              <element name="timeLeft" type="xsd:int"/>
            </sequence>
          </extension>
        </complexContent>
      </complexType>
    </schema>
    <schema targetNamespace="urn:QCDgrid" xmlns="http://www.w3.org/2001/XMLSchema">
      <import namespace="http://responses.fc.qcdgrid.epcc.ed.ac.uk" />
      <import namespace="http://schemas.xmlsoap.org/soap/encoding/" />
      <complexType name="ArrayOf_xsd_string">
        <complexContent>
          <restriction base="soapenc:Array">
            <attribute ref="soapenc:arrayType" wsdl:arrayType="xsd:string[]" />
          </restriction>
        </complexContent>
      </complexType>
  </wsdl:types>

```

```
</complexType>
</schema>
</wsdl:types>

<wsdl:message name="proxyInfoRequest">
</wsdl:message>

<wsdl:message name="getVersionResponse">
  <wsdl:part name="getVersionReturn" type="tnsl:ResponseString"/>
</wsdl:message>

<wsdl:message name="proxyInfoResponse">
  <wsdl:part name="proxyInfoReturn" type="tnsl:ResponseProxyInfo"/>
</wsdl:message>

<wsdl:message name="getURLRequest">
  <wsdl:part name="in0" type="xsd:string"/>
</wsdl:message>

<wsdl:message name="proxyInitResponse">
  <wsdl:part name="proxyInitReturn" type="tnsl:ResponseProxyStatus"/>
</wsdl:message>

<wsdl:message name="proxyDestroyResponse">
  <wsdl:part name="proxyDestroyReturn" type="tnsl:Response"/>
</wsdl:message>

<wsdl:message name="proxyDestroyRequest">
</wsdl:message>

<wsdl:message name="getVersionRequest">
</wsdl:message>

<wsdl:message name="proxyInitRequest">
  <wsdl:part name="in0" type="xsd:string"/>
</wsdl:message>

<wsdl:message name="getURLResponse">
  <wsdl:part name="getURLReturn" type="tnsl:ResponseGetURL"/>
</wsdl:message>

<wsdl:portType name="FCInterface">
  <wsdl:operation name="getURL" parameterOrder="in0">
    <wsdl:input message="impl:getURLRequest" name="getURLRequest"/>
    <wsdl:output message="impl:getURLResponse" name="getURLResponse"/>
  </wsdl:operation>
  <wsdl:operation name="getVersion">
    <wsdl:input message="impl:getVersionRequest" name="getVersionRequest"/>
    <wsdl:output message="impl:getVersionResponse" name="getVersionResponse"/>
  </wsdl:operation>
</wsdl:portType>
</wsdl:binding>
</wsdl:service>
</wsdl:definitions>
```

```
<wsdl:operation name="proxyInit" parameterOrder="in0">
  <wsdl:input message="impl:proxyInitRequest" name="proxyInitRequest"/>
  <wsdl:output message="impl:proxyInitResponse" name="proxyInitResponse"/>
</wsdl:operation>
<wsdl:operation name="proxyInfo">
  <wsdl:input message="impl:proxyInfoRequest" name="proxyInfoRequest"/>
  <wsdl:output message="impl:proxyInfoResponse" name="proxyInfoResponse"/>
</wsdl:operation>
<wsdl:operation name="proxyDestroy">
  <wsdl:input message="impl:proxyDestroyRequest" name="proxyDestroyRequest"/>
  <wsdl:output message="impl:proxyDestroyResponse" name="proxyDestroyResponse"/>
</wsdl:operation>
</wsdl:portType>
<wsdl:binding name="FCSoapBinding" type="impl:FCInterface">
  <wsdlsoap:binding style="rpc" transport="http://schemas.xmlsoap.org/soap/http"/>
  <wsdl:operation name="getURL">
    <wsdlsoap:operation soapAction="" />
    <wsdl:input name="getURLRequest">
      <wsdlsoap:body encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
        namespace="urn:QCDgrid" use="encoded"/>
    </wsdl:input>
    <wsdl:output name="getURLResponse">
      <wsdlsoap:body encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
        namespace="urn:QCDgrid" use="encoded"/>
    </wsdl:output>
  </wsdl:operation>
  <wsdl:operation name="getVersion">
    <wsdlsoap:operation soapAction="" />
    <wsdl:input name="getVersionRequest">
      <wsdlsoap:body encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
        namespace="urn:QCDgrid" use="encoded"/>
    </wsdl:input>
    <wsdl:output name="getVersionResponse">
      <wsdlsoap:body encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
        namespace="urn:QCDgrid" use="encoded"/>
    </wsdl:output>
  </wsdl:operation>
  <wsdl:operation name="proxyInit">
    <wsdlsoap:operation soapAction="" />
```

```
<wsdl:input name="proxyInitRequest">
  <wsdlsoap:body encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
namespace="urn:QCDgrid" use="encoded"/>
</wsdl:input>
<wsdl:output name="proxyInitResponse">
  <wsdlsoap:body encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
namespace="urn:QCDgrid" use="encoded"/>
</wsdl:output>
</wsdl:operation>
<wsdl:operation name="proxyInfo">
  <wsdlsoap:operation soapAction=""/>
  <wsdl:input name="proxyInfoRequest">
    <wsdlsoap:body encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
namespace="urn:QCDgrid" use="encoded"/>
  </wsdl:input>
  <wsdl:output name="proxyInfoResponse">
    <wsdlsoap:body encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
namespace="urn:QCDgrid" use="encoded"/>
  </wsdl:output>
</wsdl:operation>
<wsdl:operation name="proxyDestroy">
  <wsdlsoap:operation soapAction=""/>
  <wsdl:input name="proxyDestroyRequest">
    <wsdlsoap:body encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
namespace="urn:QCDgrid" use="encoded"/>
  </wsdl:input>
  <wsdl:output name="proxyDestroyResponse">
    <wsdlsoap:body encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
namespace="urn:QCDgrid" use="encoded"/>
  </wsdl:output>
</wsdl:operation>
</wsdl:binding>
<wsdl:service name="FCInterfaceService">
  <wsdl:port binding="impl:FCSoapBinding" name="FC">
    <wsdlsoap:address location="http://qcdgrid4.epcc.ed.ac.uk:8080/axis/services/FC"/>
  </wsdl:port>
</wsdl:service>
</wsdl:definitions>
```