Tech-X Corporation www.txcorp.com

5621 Arapahoe Ave, Suite A
Boulder, CO 80303
phone: 303-448-0727
fax: 303-448-7756

SBIR Phase I Final Progress Report

July 11, 2005

An Abstract Job Handling Grid Service for Dataset Analysis

Contract: DE-FG02-04ER84095

David A. Alexander, Principle Investigator
alexanda@txcorp.com
303-448-7751

Abstract

For Phase I of the Job Handling project, Tech-X has built a Grid service for processing analysis requests, as well as a Graphical User Interface (GUI) client that uses the service. The service is designed to generically support High-Energy Physics (HEP) experimental analysis tasks. It has an extensible, flexible, open architecture and language. The service uses the Solenoidal Tracker At RHIC (STAR) experiment as a working example. STAR is an experiment at the Relativistic Heavy Ion Collider (RHIC) at the Brookhaven National Laboratory (BNL). STAR and other experiments at BNL generate multiple Petabytes of HEP data. The raw data is captured as millions of input files stored in a distributed data catalog. Potentially using thousands of files as input, analysis requests are submitted to a processing environment containing thousands of nodes. The Grid service provides a standard interface to the processing farm. It enables researchers to run large-scale, massively parallel analysis tasks, regardless of the computational resources available in their location.

System Description

The major components of the system software are the Grid service and the GUI client. All the components are written in Java, using Globus Toolkit 4.0, the industry standard Grid implementation, as the service framework and communications layer. Globus provides a great deal of standard functionality, including a Simple Object Access Protocol (SOAP) message layer. All communication between the client and server occurs via SOAP messages, which are Extensible Markup Language (XML) documents serialized and sent over Hyper-Text Transfer Protocol (HTTP).

The GUI client is a user-friendly interface for creating and running Grid service requests. It is built using standard Java Swing components, and so may be run on virtually any platform. Requests are written in the Request Definition Language (RDL). RDL is a extensible, human-readable, open language. Its format is specified by a standard XML schema. The client obtains the schema from the service, uses it to generate and validate a RDL request, and submits it to the service.

The RDL handling service (RDLService) runs within a Globus Web Service Resource Framework (WSRF) container; essentially, an application server. RDLService stores, runs, and monitors user requests. It uses the pre-existing STAR Scheduler to submit requests to the BNL processing farm. The Scheduler is a highly configurable job engine, able to process requests using a variety of policies, dispatchers, and queues. The RDLService also is capable of accepting requests in Job Description Language (JDL), the predecessor to RDL. Additional schemas may be added to the service, permitting the definition of additional request languages.

The main software components and their relationships are shown in Figure 1. Multiple clients can connect simultaneously, each having a separate instance of the RDLService. The service instances share a common, persistent state object, which stores and manages user requests. The service state also creates and manages instances of the Scheduler as requests are run.

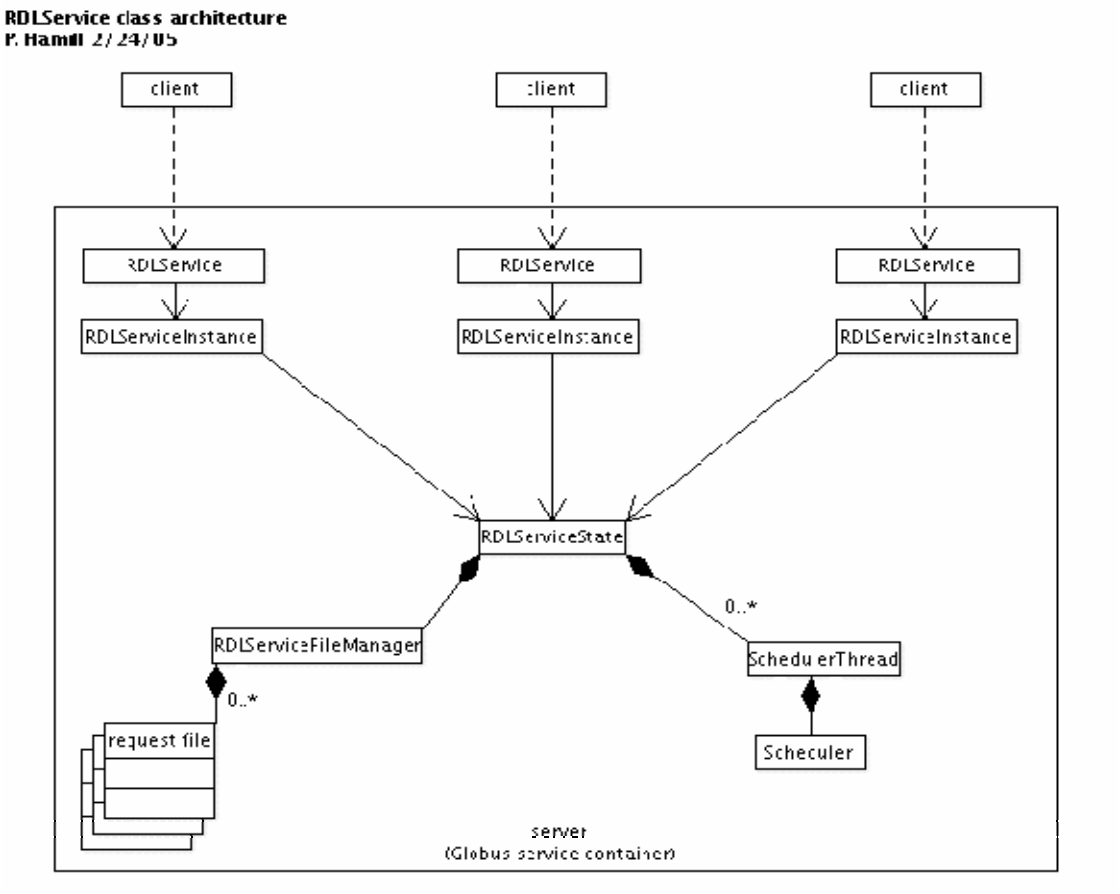

Figure 1: The main RDL client and service components.

Usage Example

To illustrate the functionality of the RDL service and client, a usage example is helpful. This example outlines the typical usage of the service by one of BNL's remote collaborators.

A researcher is designing an experimental analysis of a subset of the STAR data. He or she picks an input dataset, which includes both files on disk and files extracted from a data catalog. An application will be used to process the dataset. The application can be a standard program or script, or custom software. Tasks also are defined for the application. In this case, the researcher's application is Java, and the task is to run an algorithm contained in a Java class. The researcher defines processing parameters, such as the number of processing nodes to use (i.e., degree of parallelization), time and memory constraints, and the output handling. Once the analysis is outlined, the researcher uses a client tool to create the request. The GUI client's user-friendly point-and-click interface can be used to define the request, or the request XML may be manually edited using a text or XML editor.

After creating the request, the researcher uses the client to connect to the RDL service. The service's address is a standard Universal Resource Identifier (URI), so it is accessible from anywhere on the Internet. The service obtains and validates the user's credentials, a US Department of Energy (DOE) security certificate. Once authorized, the user sends the RDL request to the service. The service validates the request, stores it, and submits it to the STAR Scheduler. The Scheduler dispatches the request to a queue and runs it. As the request runs, the service monitors its status. The user may close the client and reconnect later to check the status of their request, which may take seconds or hours, depending on system load, queue status, and the size of the job. When the request finishes, the service records its completion state. The researcher can then study the results and perform further analysis. He or she also may modify and re-submit the original request.

Development of RDL

A key piece of this project is the development of Request Definition Language (RDL).

GUI Client

The GUI client is a tool both for creating requests and for working with the service. The GUI interface permits request input using either standard point-and-click graphical controls, or direct editing of the request XML. Requests can be saved in local files and edited without a service connection. The user also can use the client to connect to the service and run, monitor, and control requests on the Grid.

The request editing interface is a set of panes accessed by tab controls. When the client is started, the XML Text Editor pane appears (Figure 2.) It displays the XML content of the current RDL request. The user can manually edit the XML, or use the graphical controls on the other panes to modify the request. The File menu is used to load and save RDL files.

The text area below the XML Text Editor pane is separate from the tabbed panes, and is used to display system messages such as service status. At the bottom of the GUI client window, a status line displays the service connection state.

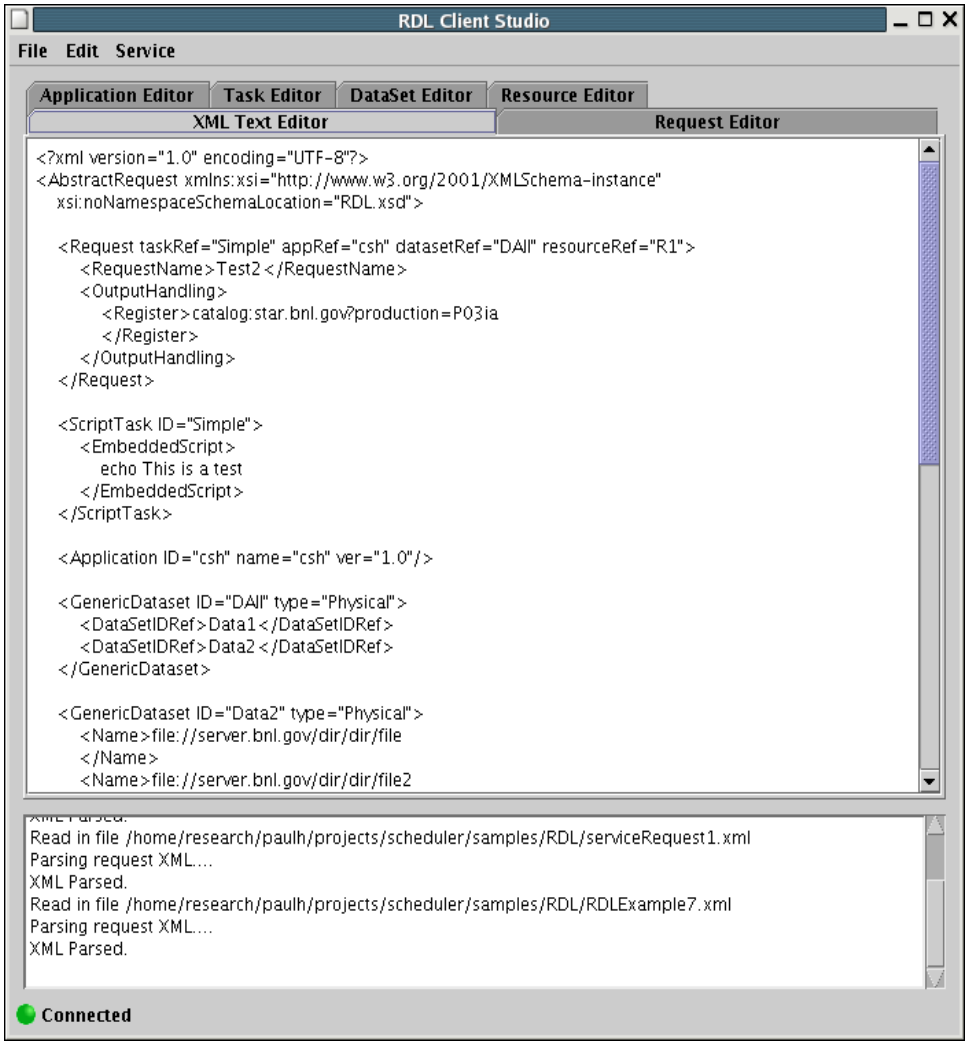

Figure 2: The XML Text Editor pane, showing a request's XML source.

The Request Editor pane is used to specify the top-level request attributes such as name, description, and output handling (Figure 3.)

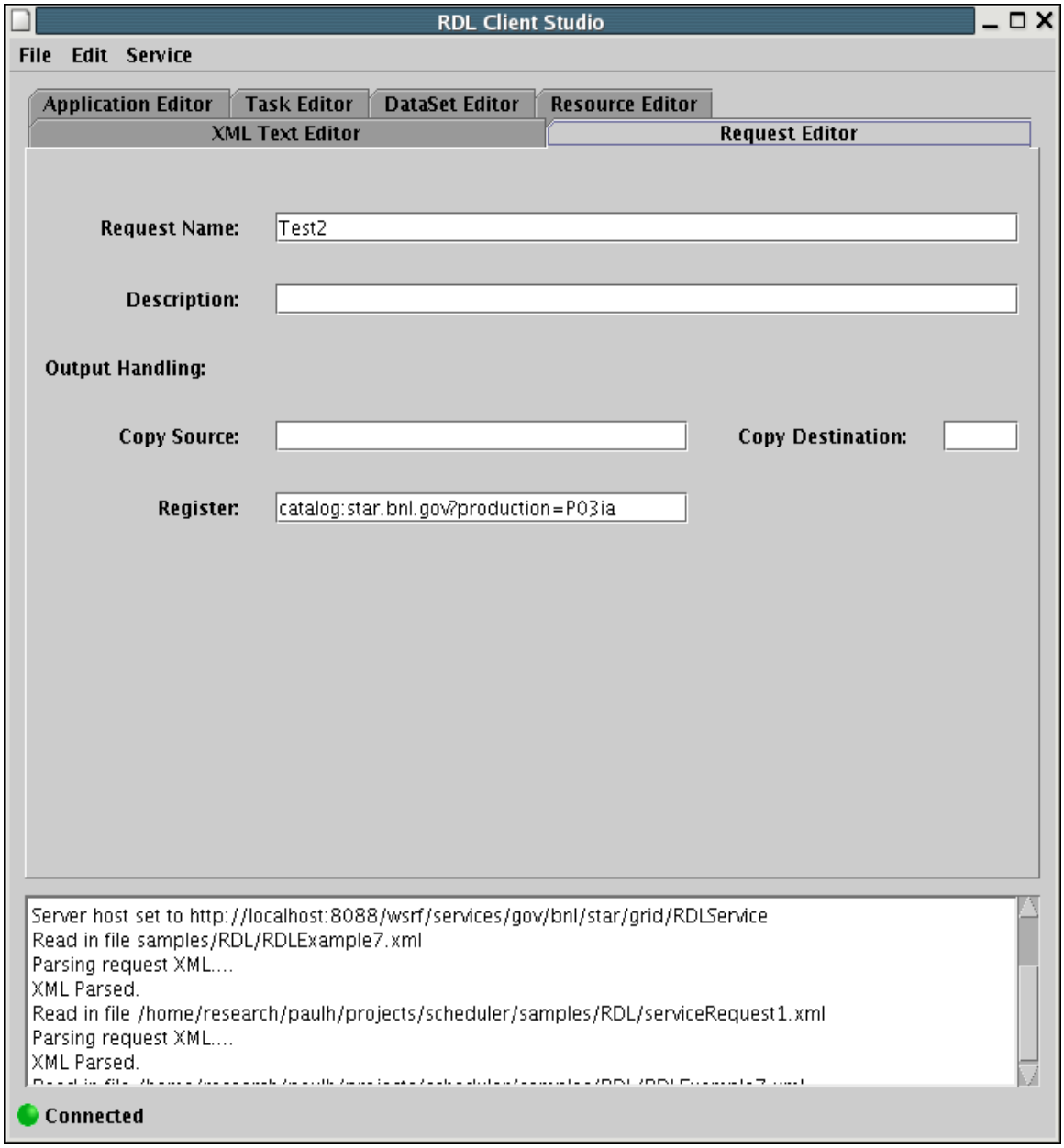

Figure 3: The Request Editor pane.

The Application pane lets the user specify the name of the application to be run on the Grid nodes (Figure 4). Currently it has a single control allowing selection of an application name, such as 'csh', 'root4star', or 'java'. Future versions could set additional application attributes, such as installation location and version information.

Figure 4: The Application Editor pane.

The Task Editor pane lets the user specify a task for the application (Figure 5). The nature of a task depends on the application, but it could be an embedded script, a script name, a Java class file name, or a set of arguments. The task can set the system streams STDIN, STDOUT, and STDERR, allowing all task output to be redirected to a specific file.

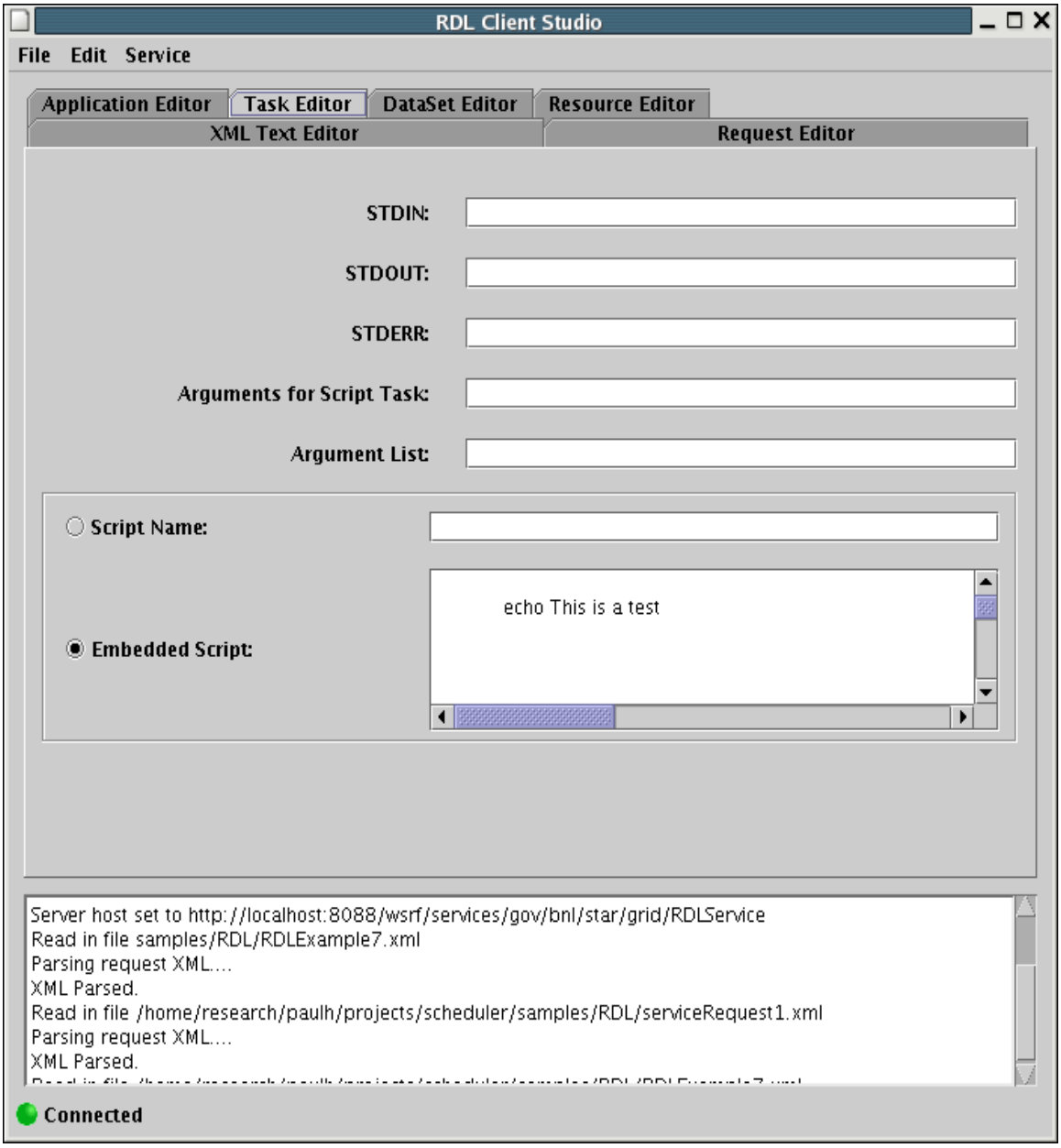

Figure 5: The Task Editor pane.

A request has an input dataset. The user specifies the dataset on the Dataset Editor pane (Figure 6.) A dataset may be based on a catalog query or a list of file names.

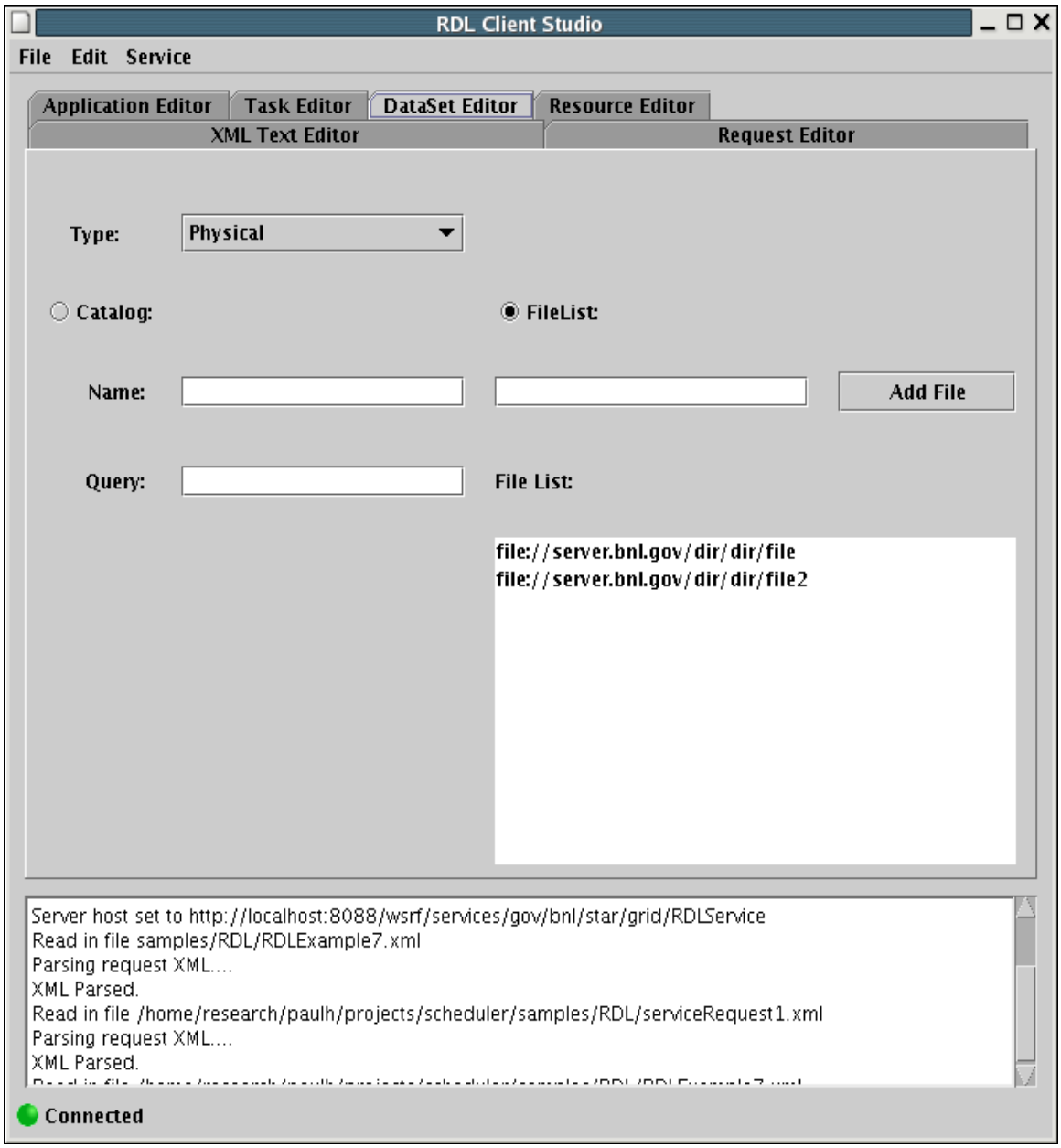

Figure 6: The Dataset Editor pane.

The Resource Editor pane (Figure 7) allows the user to specify processing resources and parameters such as time, memory limits, and number of input or output files.

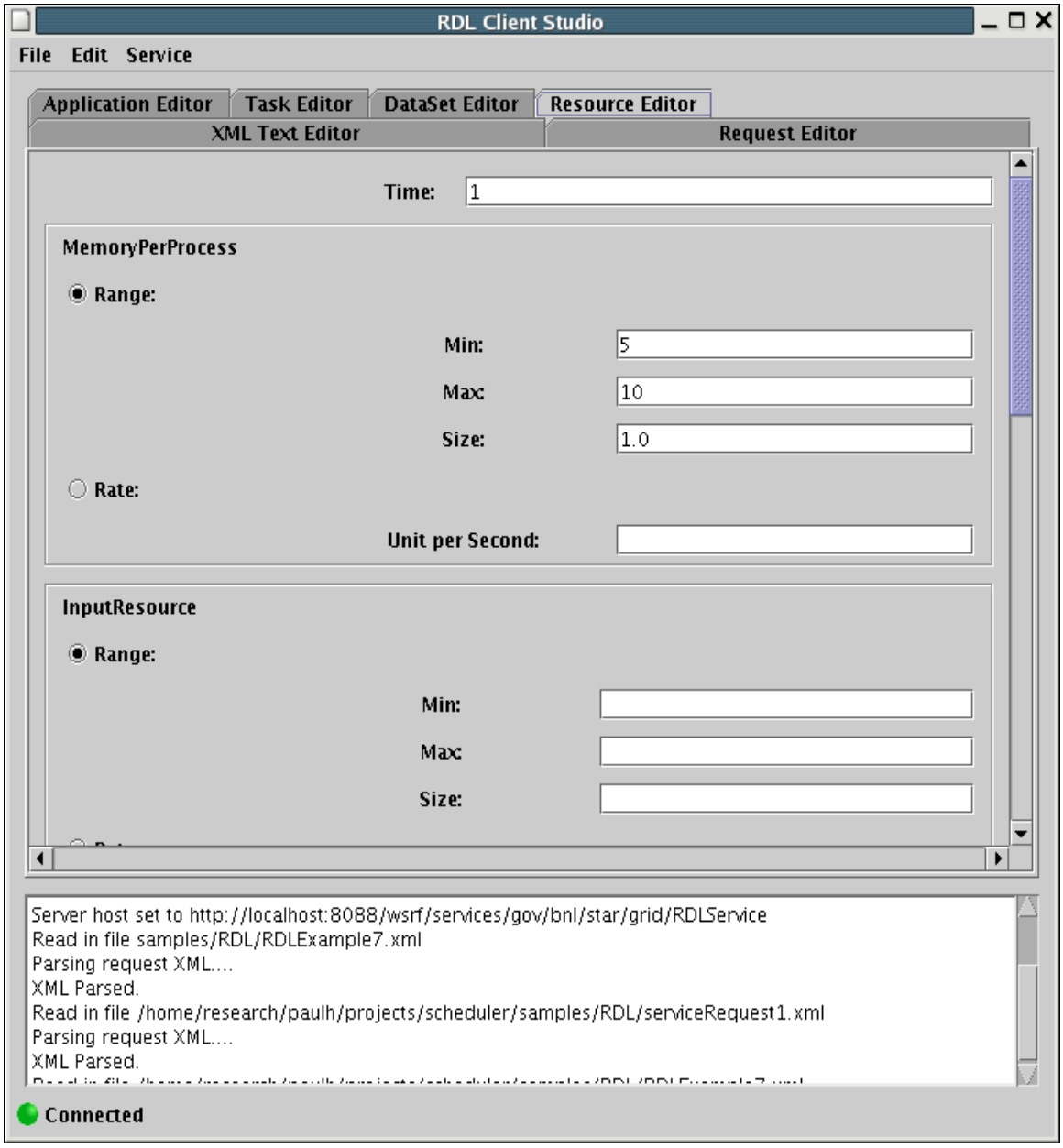

Figure 7: The Resource Editor pane.

Using the Service menu, the user can connect to the service, submit the current request, and open the Monitor window (Figure 8). The Monitor shows the service's current set of requests and their status, and allows the user to stop or delete requests.

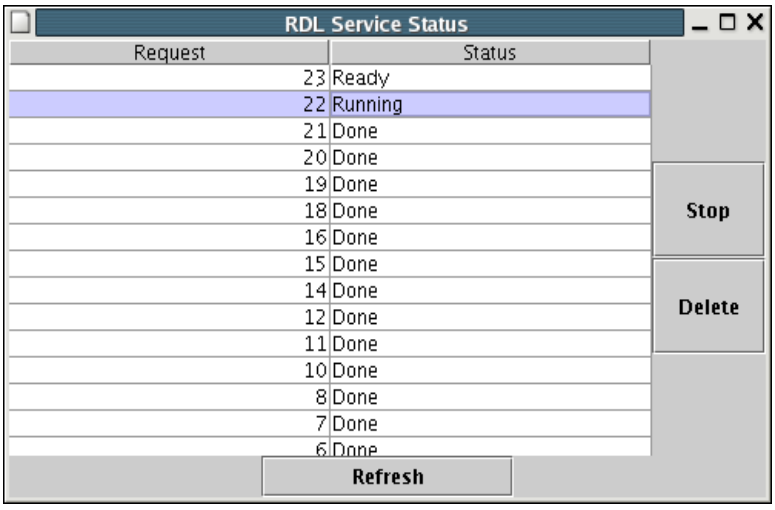

Figure 8: The Monitor window, used to check service status and manage requests.