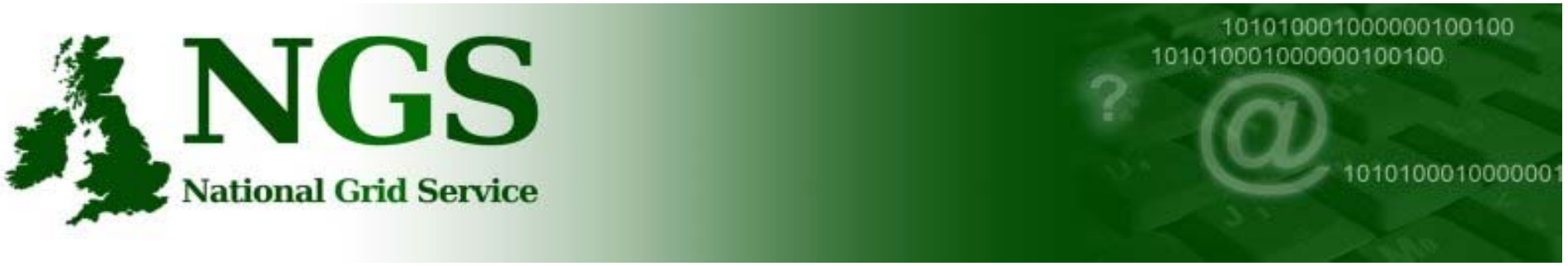


The NGS Grid Portal

David Meredith
NGS + Grid Technology Group, e-Science Centre,
Daresbury Laboratory, UK

d.j.meredith@dl.ac.uk

CCLRC

NGS Portal

The NGS portal can be used to access and interact with the HPC and Data resources available on the Grid via SSO (Certificates + myproxy):

- Browse for different applications available on a Grid this includes your own personal applications and pre-configured applications available on a particular Grid (e.g. the NGS is currently publishing applications within the NGS portal to be made easily available for its users).
- Submit/monitor compute jobs/applications.
- Access and move data around the Compute and Data Grid (Gridftp, srb).

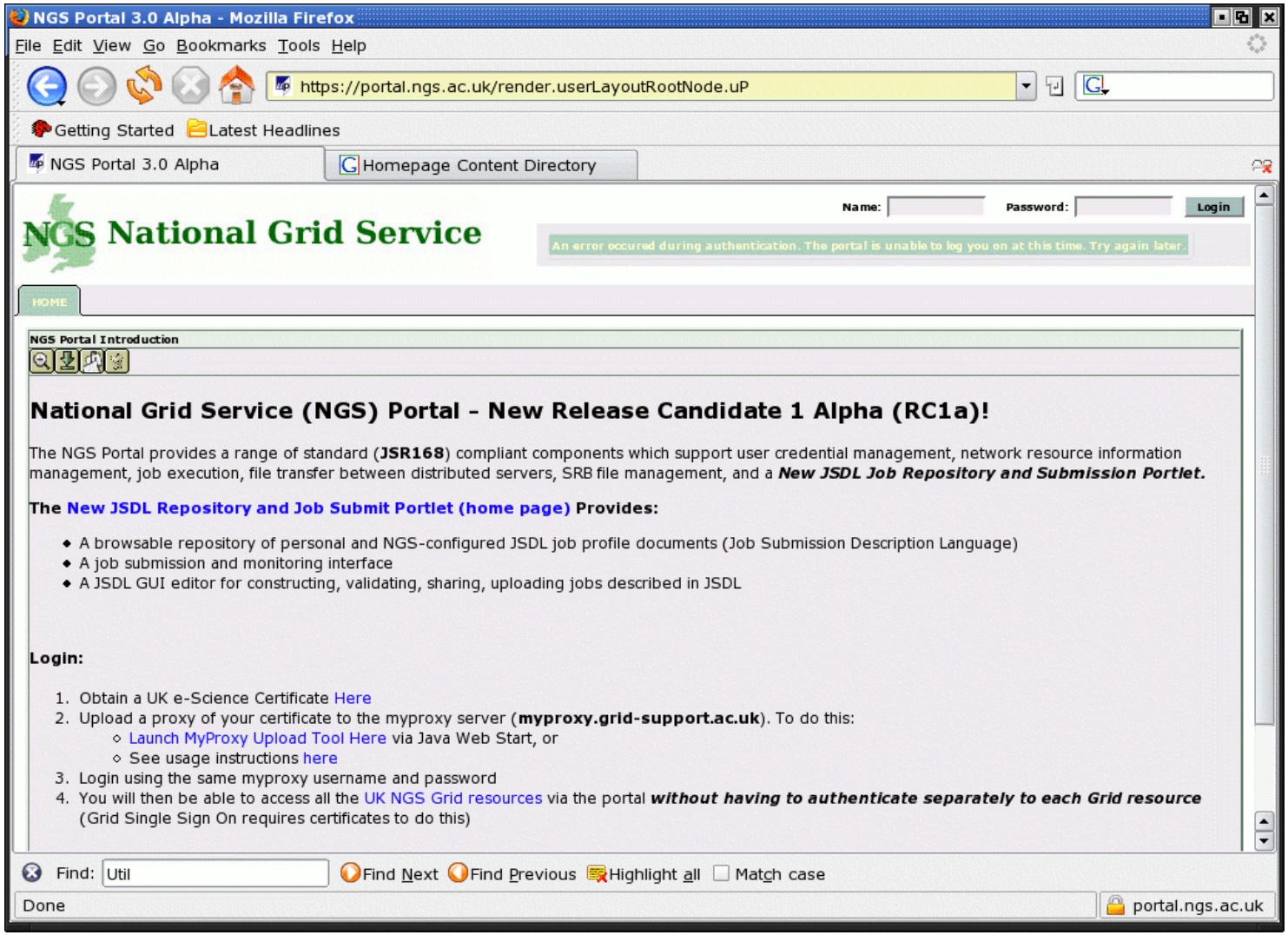
Portal and Portlets

- The NGS Grid portal extends a JSR-168 compliant portal container which hosts a selection of ‘portlets.’
- Portlets are online-accessible applications that are hosted and managed within the portal container.
- The list of portlets that are deployed to the portal make up the portal’s overall functionality (users may be interested only in a selection of portlets)
- Main Benefit: Portlets facilitate the sharing and re-use of applications (168-compliant 3rd party portlets can be used within a portal as required).
- NGS and CCLRC e-Science Centre are developing a collection of portlets designed for the Computational and Data Grid.
- The NGS portal is a current implementation of these Grid portlets.

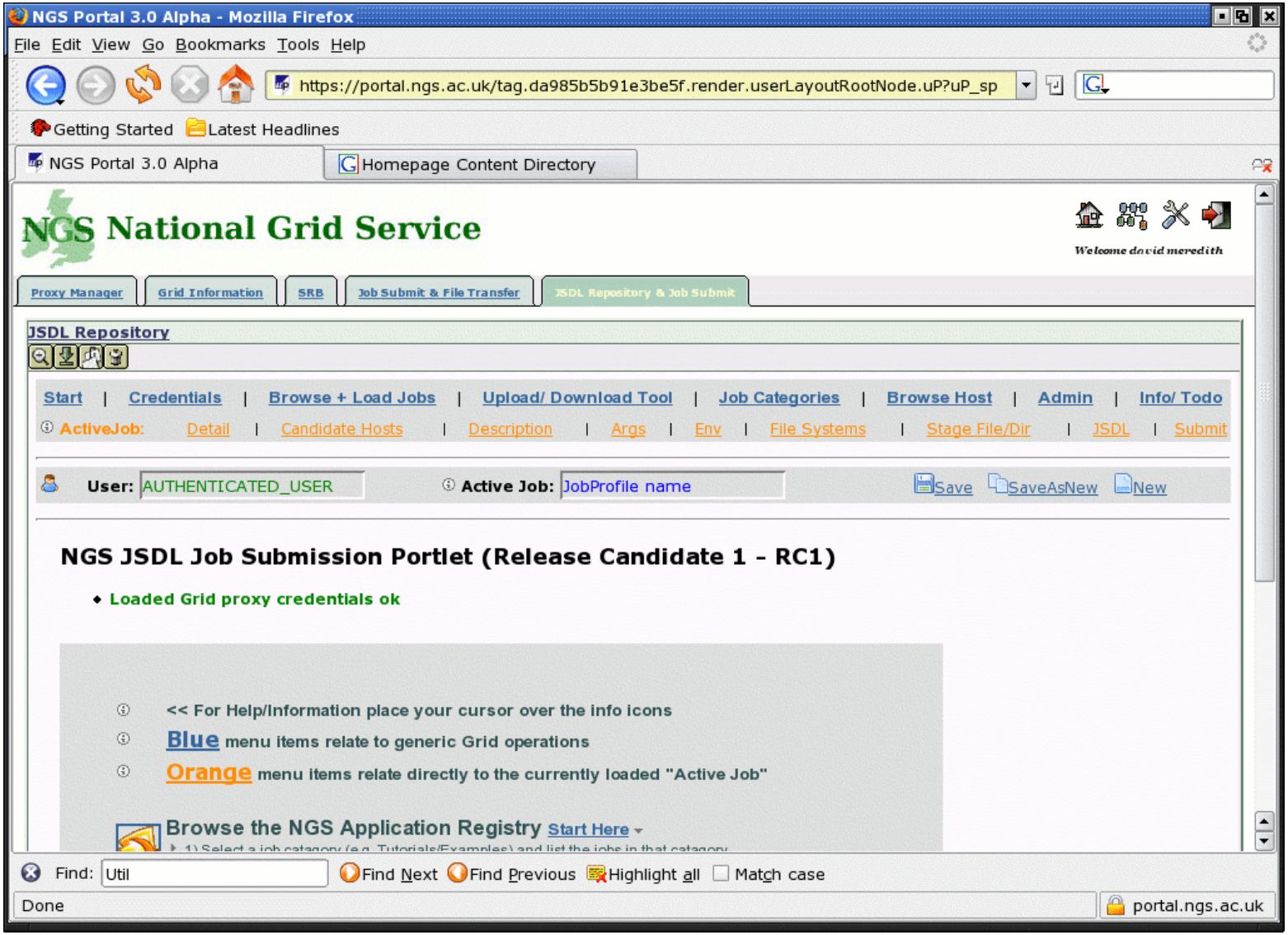
A Famous Portal + Portlets

Select Portlets of Interest (customisation)

NGS portal


NGS Portal 3.0 Alpha - Mozilla Firefox
File Edit View Go Bookmarks Tools Help
https://portal.ngs.ac.uk/render.userLayoutRootNode.uP
Getting Started
Latest Headlines
NGS Portal 3.0 Alpha
Homepage Content Directory
NGS National Grid Service
Name:
Password:
Login
An error occured during authentication. The portal is unable to log you on at this time. Try again later.
HOME
NGS Portal Introduction
National Grid Service (NGS) Portal - New Release Candidate 1 Alpha (RC1a)!
The NGS Portal provides a range of standard (JSR168) compliant components which support user credential management, network resource information management, job execution, file transfer between distributed servers, SRB file management, and a New JSDL Job Repository and Submission Portlet.
The New JSDL Repository and Job Submit Portlet (home page) Provides:
A browsable repository of personal and NGS-configured JSDL job profile documents (Job Submission Description Language)
A job submission and monitoring interface
A JSDL GUI editor for constructing, validating, sharing, uploading jobs described in JSDL
Login:
1. Obtain a UK e-Science Certificate Here
2. Upload a proxy of your certificate to the myproxy server (myproxy.grid-support.ac.uk). To do this:
Launch MyProxy Upload Tool Here via Java Web Start, or
See usage instructions here
3. Login using the same myproxy username and password
4. You will then be able to access all the UK NGS Grid resources via the portal without having to authenticate separately to each Grid resource (Grid Single Sign On requires certificates to do this)
Find: Util
Find Next
Find Previous
Highlight all
Match case
Done
portal.ngs.ac.uk

After Login – Grid portlets



JSDL Repository / Job Submit Portlet

1. A browsable JSDL database of personal and shared JSDL job profile documents (Job Submission Description Language, i.e. 'job recipes/templates').
   - JSDL can be browsed for, selected and loaded in order to run applications on the Grid (loaded either 'out-of-the-box' or, more usually loaded and modified/tweaked as required).
   - JSDL can be searched for by category of interest in the portal (e.g bioinformatics, chemistry, tutorials/examples).
   - JSDL documents can be pre-configured and published by the portlet administrator(s) to be made available to all other users.
2. A JSDL GUI editor for constructing, validating, sharing, uploading jobs described in JSDL.
3. A Grid job submission and monitoring application (currently, only Globus but more Grid middleware providers are being added, e.g. GridSam/WSRF).

JSDL – Job Submission Description Language

```
<jsdl:Application>
        <jsdl:ApplicationName>gnuplot</jsdl:ApplicationName>
        <jsdl-posix:POSIXApplication>
           <jsdl-posix:Executable>
              /usr/local/bin/gnuplot
           </jsdl-posix:Executable>
           <jsdl-posix:Argument>control.txt</jsdl-posix:Argument>
<jsdl-posix:Argument>DavesControlFile.txt</jsdl-posix:Argument>
           <jsdl-posix:Input>input.dat</jsdl-posix:Input>
           <jsdl-posix:Output>output1.png</jsdl-posix:Output>
        </jsdl-posix:POSIXApplication>
     </jsdl:Application>
     <jsdl:Resources>
        <jsdl:IndividualPhysicalMemory>
<jsdl:LowerBoundedRange>2097152.0</jsdl:LowerBoundedRange>
        </jsdl:IndividualPhysicalMemory>
        <jsdl:TotalCPUCount>
           <jsdl:Exact>1.0</jsdl:Exact>
        </jsdl:TotalCPUCount>
     </jsdl:Resources>
      ....
```

1. XML Schema language for describing compute jobs in a platform independent language (XML).
2. Is agnostic of middleware - no dependencies on Globus, WSRF, gLite (portal that is generic and not tied to any particular set of Grid technologies).
3. GGF / OGF Standard.
4. JSDL documents can be validated against the JSDL and JSDL POSIX XSD Schema to ensure its correctness

JSDL – Job Submission Description Language XSD Schema

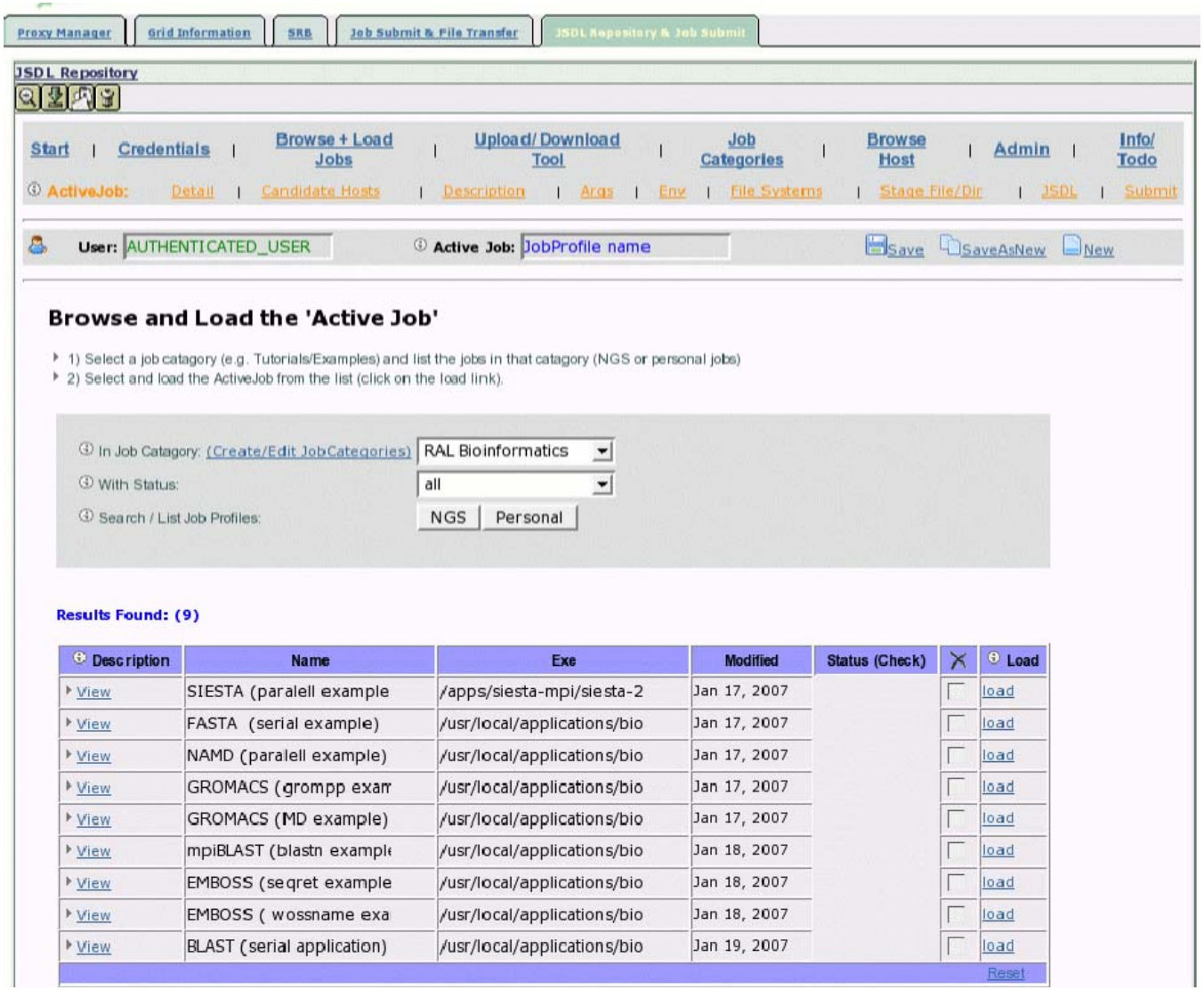
```
<!--=============================================-->
 <xsd:complexType name="Environment_Type">
        <xsd:simpleContent>
                <xsd:extension base="xsd:string">
                        <xsd:attribute name="name" type="xsd:NCName" use="required"/>
                        <xsd:attribute name="filesystemName" type="xsd:NCName" use="optional"/>
                        <xsd:anyAttribute namespace="##other" processContents="lax"/>
                </xsd:extension>
        </xsd:simpleContent>
 </xsd:complexType>
 <!--=======================================-->
 <xsd:complexType name="Argument_Type">
        <xsd:simpleContent>
                <xsd:extension base="xsd:normalizedString">
                        <xsd:attribute name="filesystemName" type="xsd:NCName" use="optional"/>
                        <xsd:anyAttribute namespace="##other" processContents="lax"/>
                </xsd:extension>
        </xsd:simpleContent>
 </xsd:complexType>
```

JSDL Repository / Database

Select category of interest – e.g. ‘RAL Bioinformatics.’

Browsing the JSDL database for public and personal job profiles.

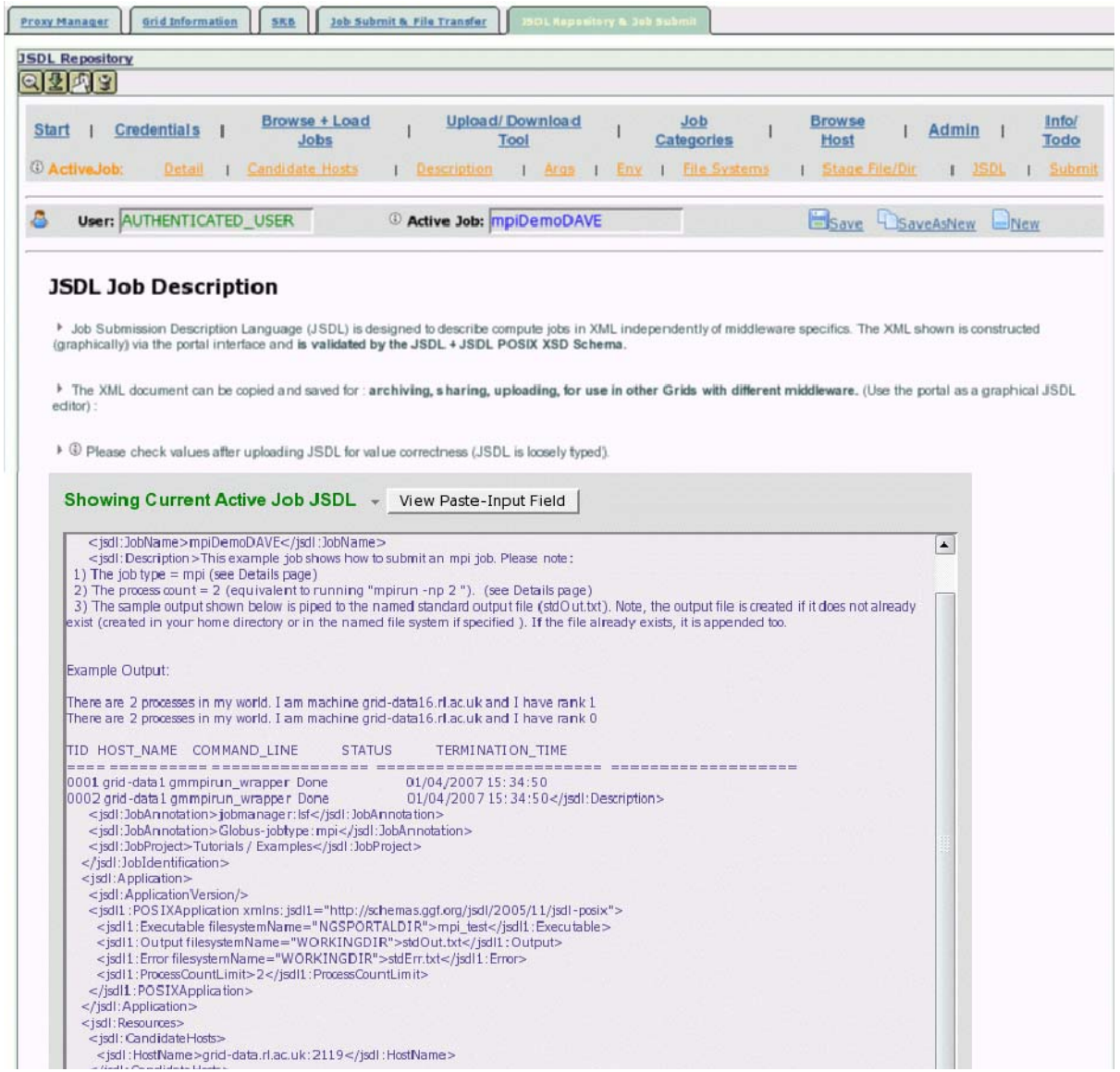
List jobs, read job descriptions and load a job to initialise the ‘Active Job.’



Active Job's JSDL

The Active Job JSDL is automatically created, updated and validated by the portal by changing parameters in the portal GUI.

The portlet acts as a JSDL GUI editor

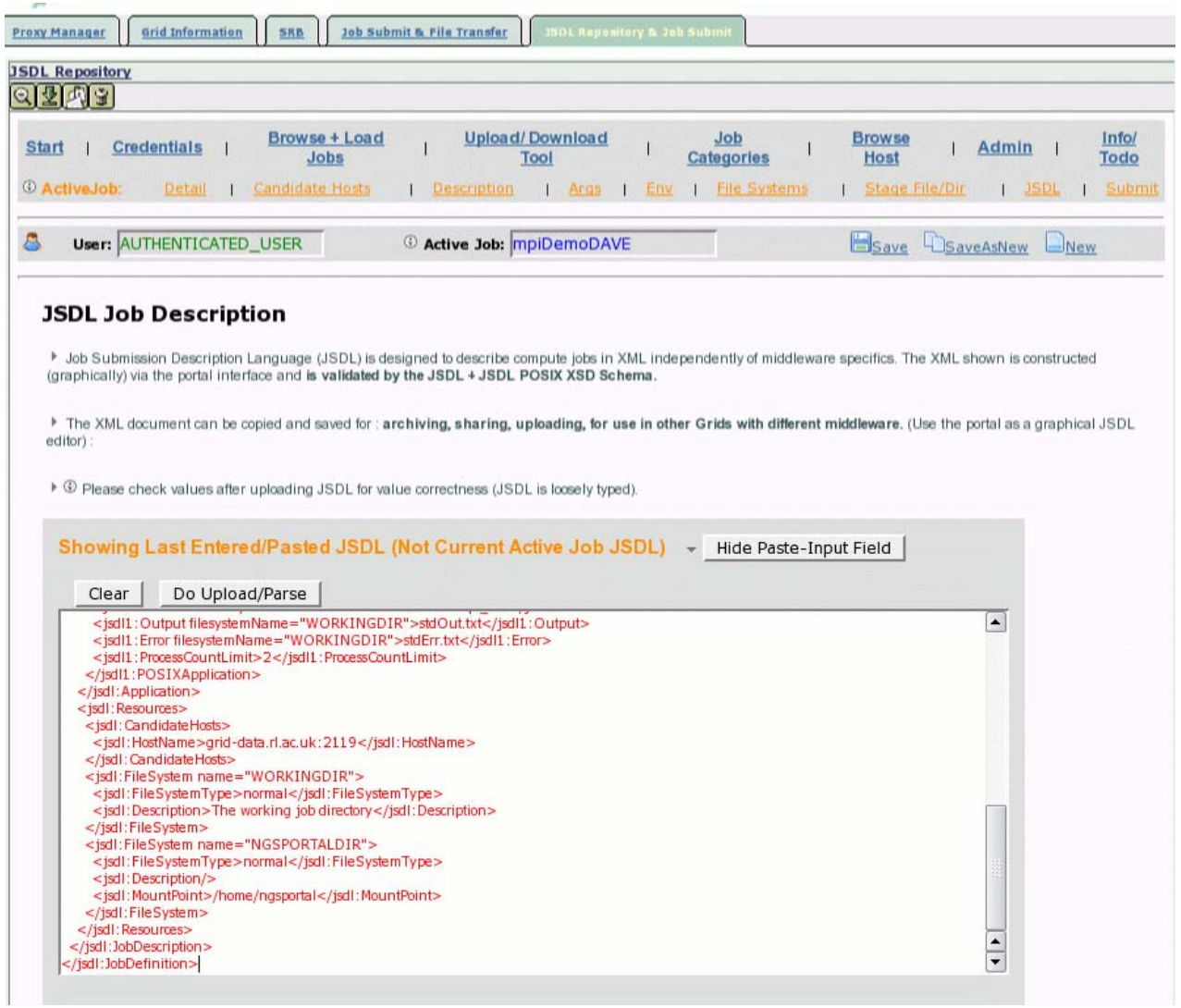


Upload / Share JSDL

JSDL documents can be uploaded to the portal in order to initialise the Active Job.

Validation errors and messages are displayed in the interface when uploading JSDL.

The portlet allows the sharing of job profiles and JSDL between users and user communities.

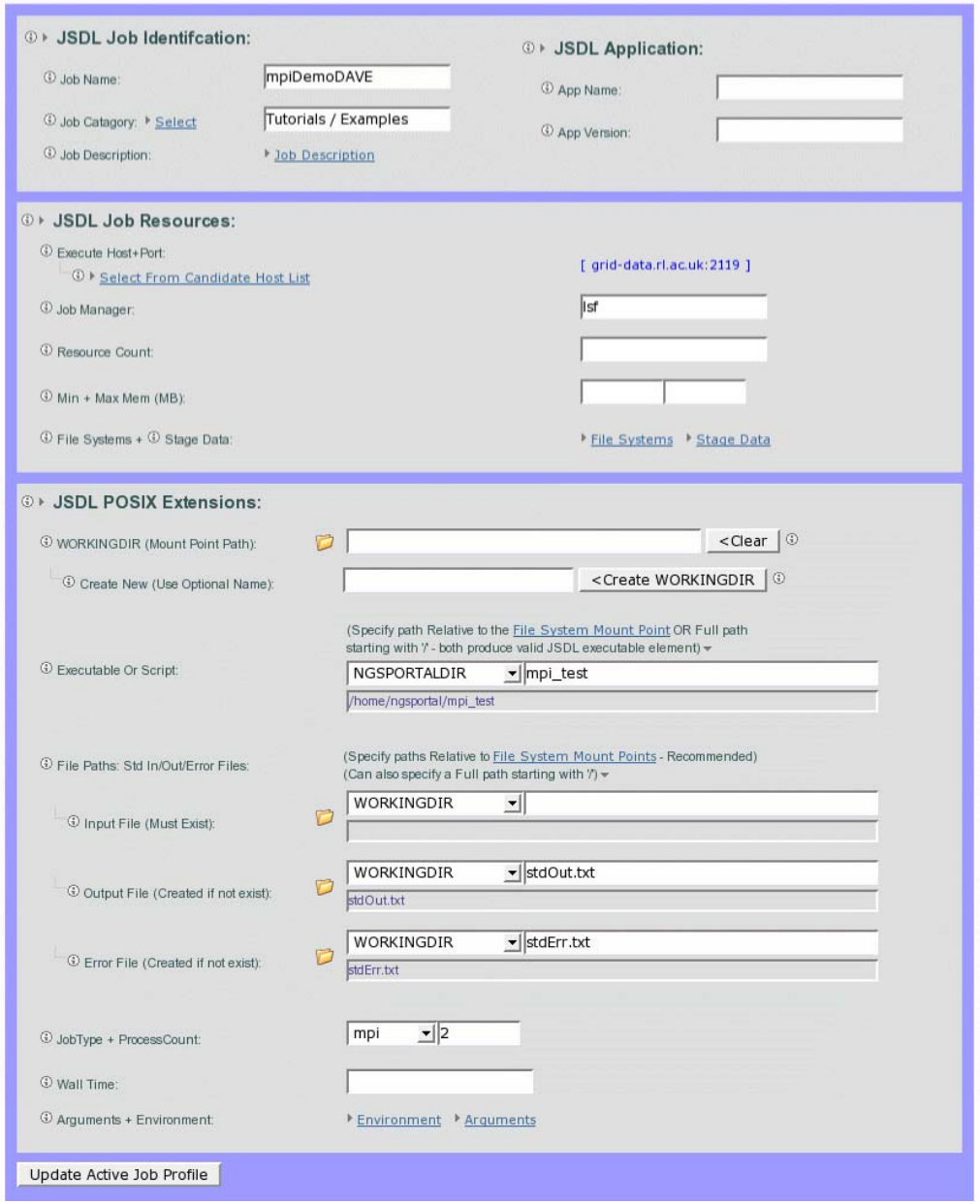


Active Job Detail

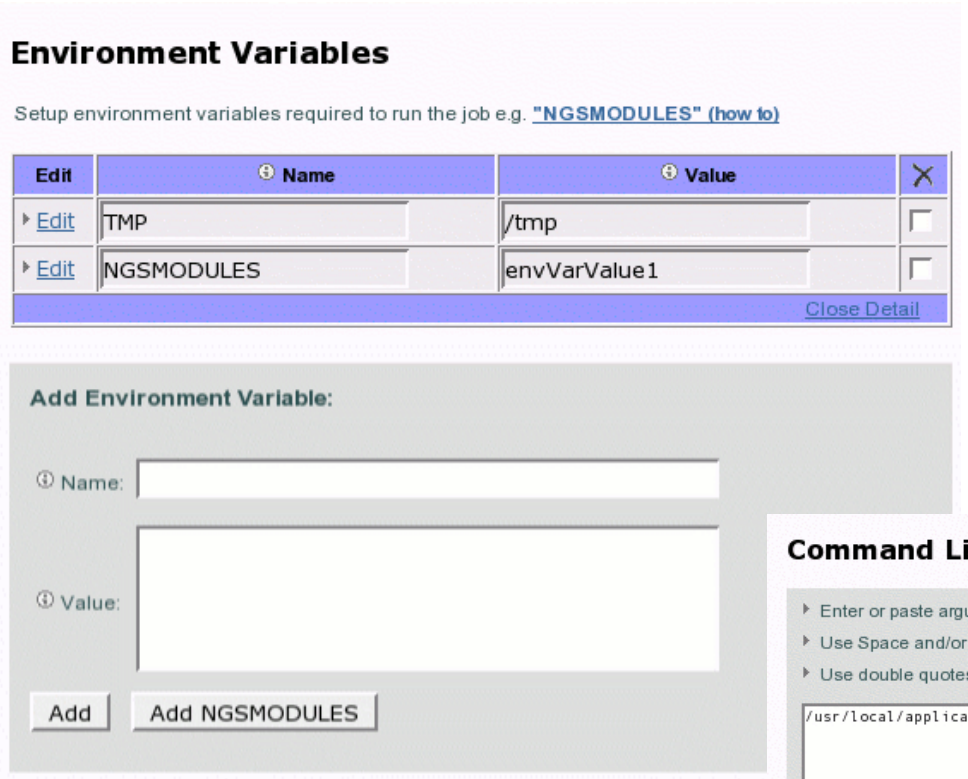
Input fields are filled out for pre-configured applications.

Changes to the parameters in the GUI will update the generated JSDL automatically.

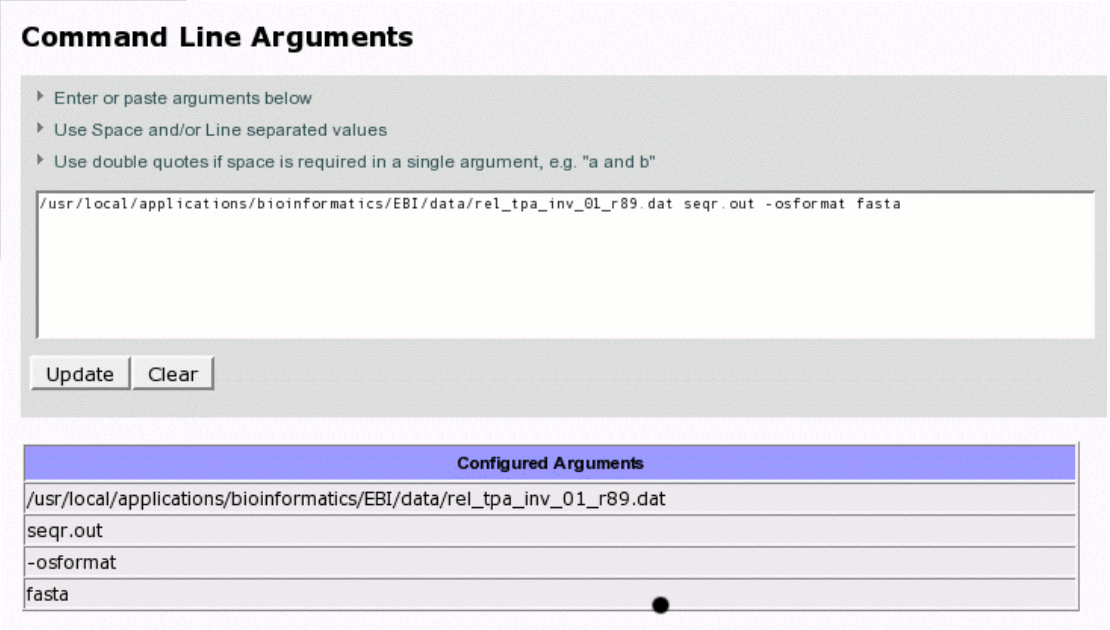
Input fields are taken from the JSDL and JSDL-POSIX extension schemas.



Environment Variables / Arguments



Add required env vars, e.g. ‘NGSMODULES’ – used to configure application environment



Paste and parse command line arguments (space and/or line separated values)

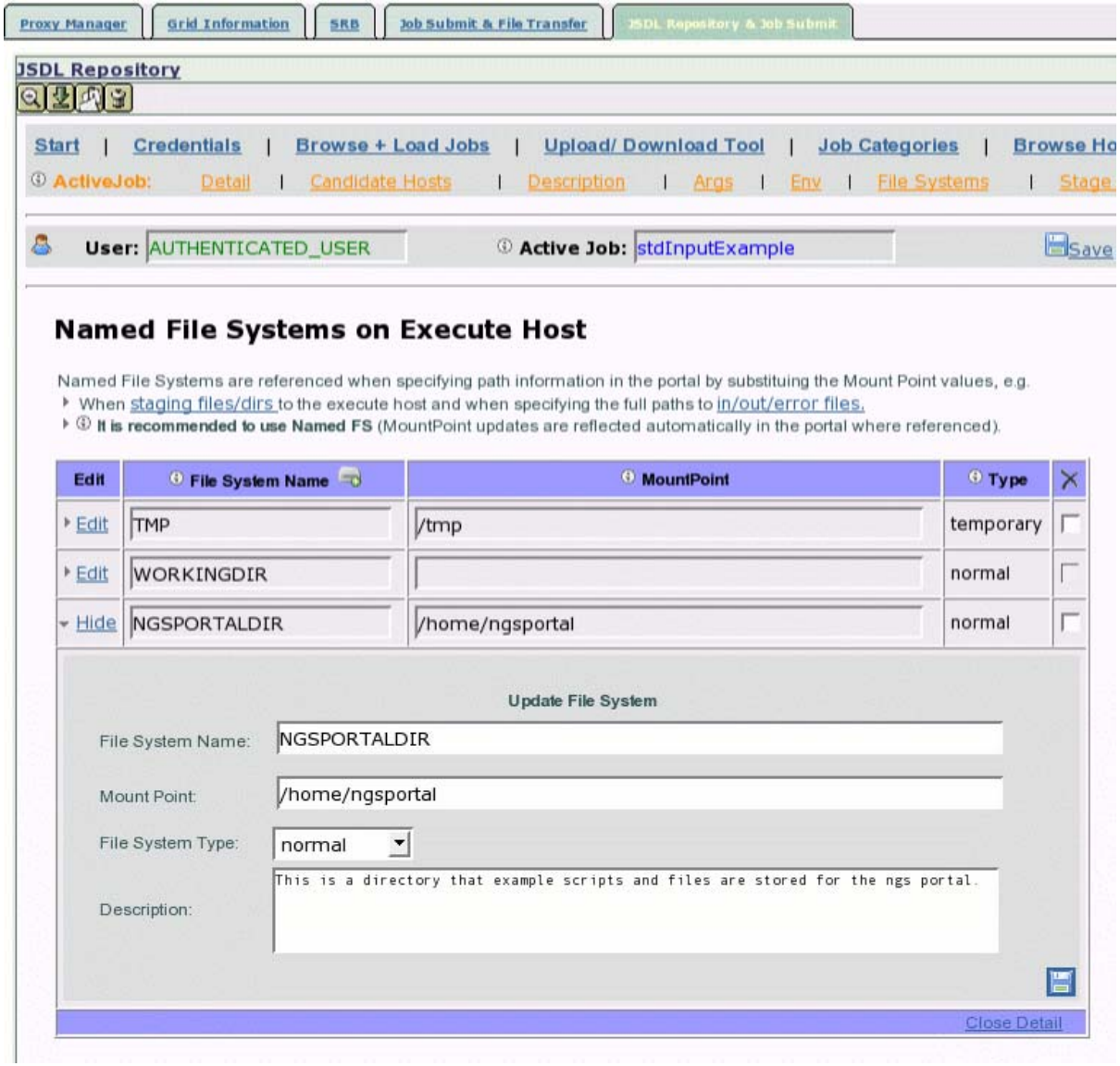
Named File Systems

Named file systems used to declare mount points that are required on the consuming system.

File system names are referenced throughout the portlet (and JSDL doc) for substituting mount points where required.

Changes to a FS mount point will be updated automatically throughout the portal/JSDL.

Used when specifying path info e.g., locations to files/dirs, stage data locations etc.

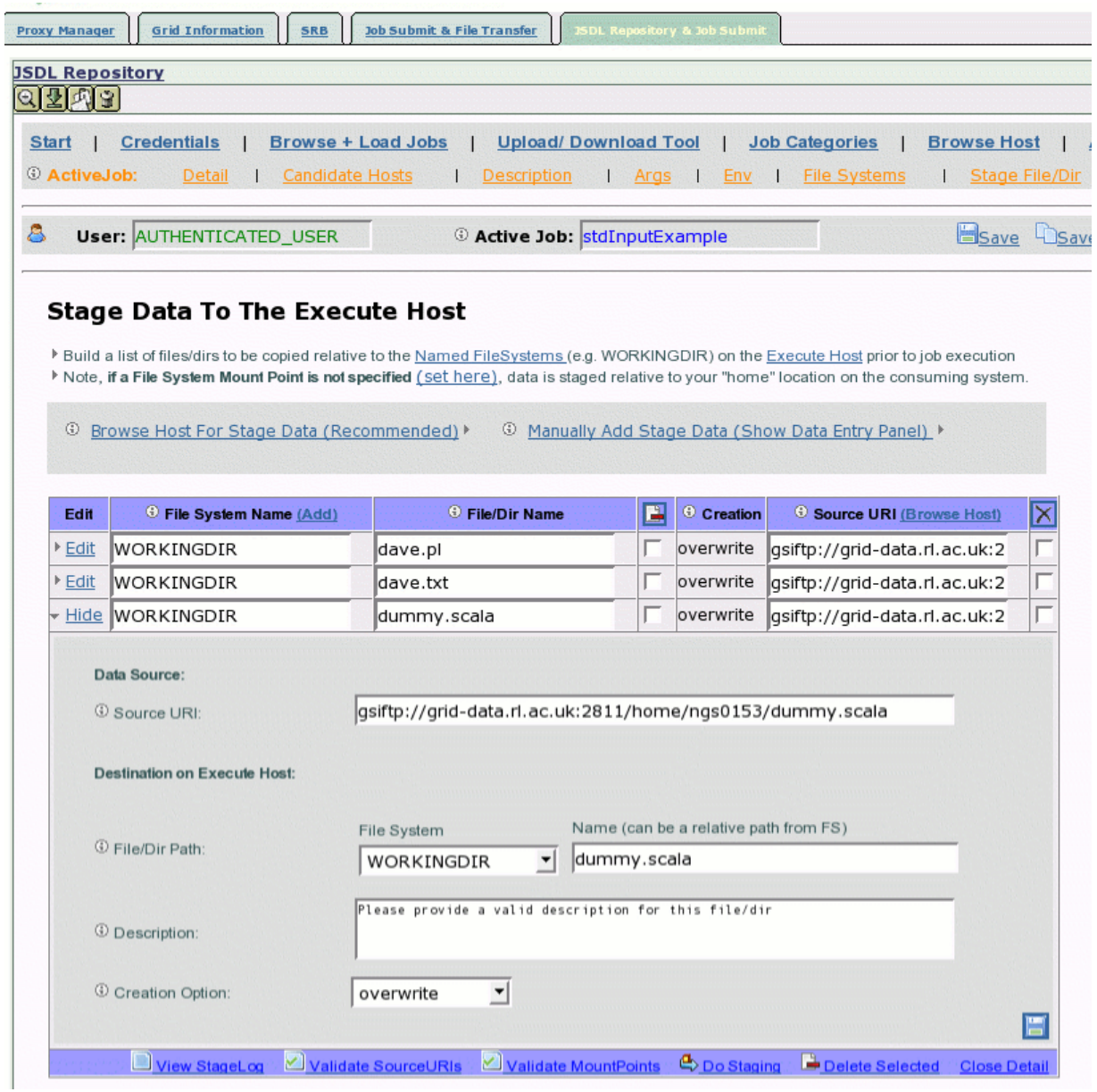


Stage Data

Compile a list of required data (i.e. data that should be copied to the consuming system from remote locations across the Grid prior to job execution).

Data is staged relative to named file systems.

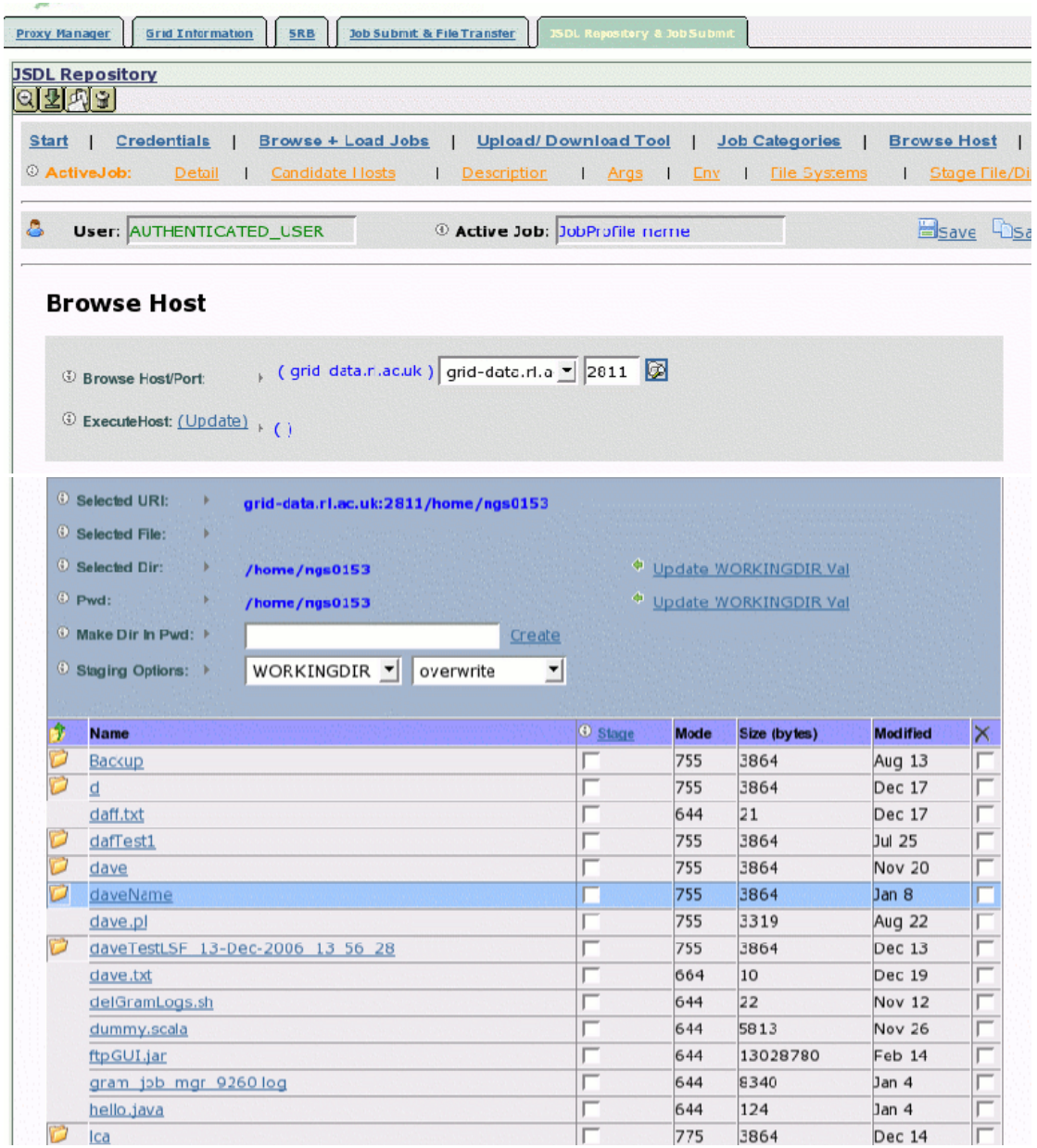
The source URI can be either specified manually or, more normally, browsed for in the 'Browse Host' page.



Browse Hosts

Browse remote Grid hosts for stage data.

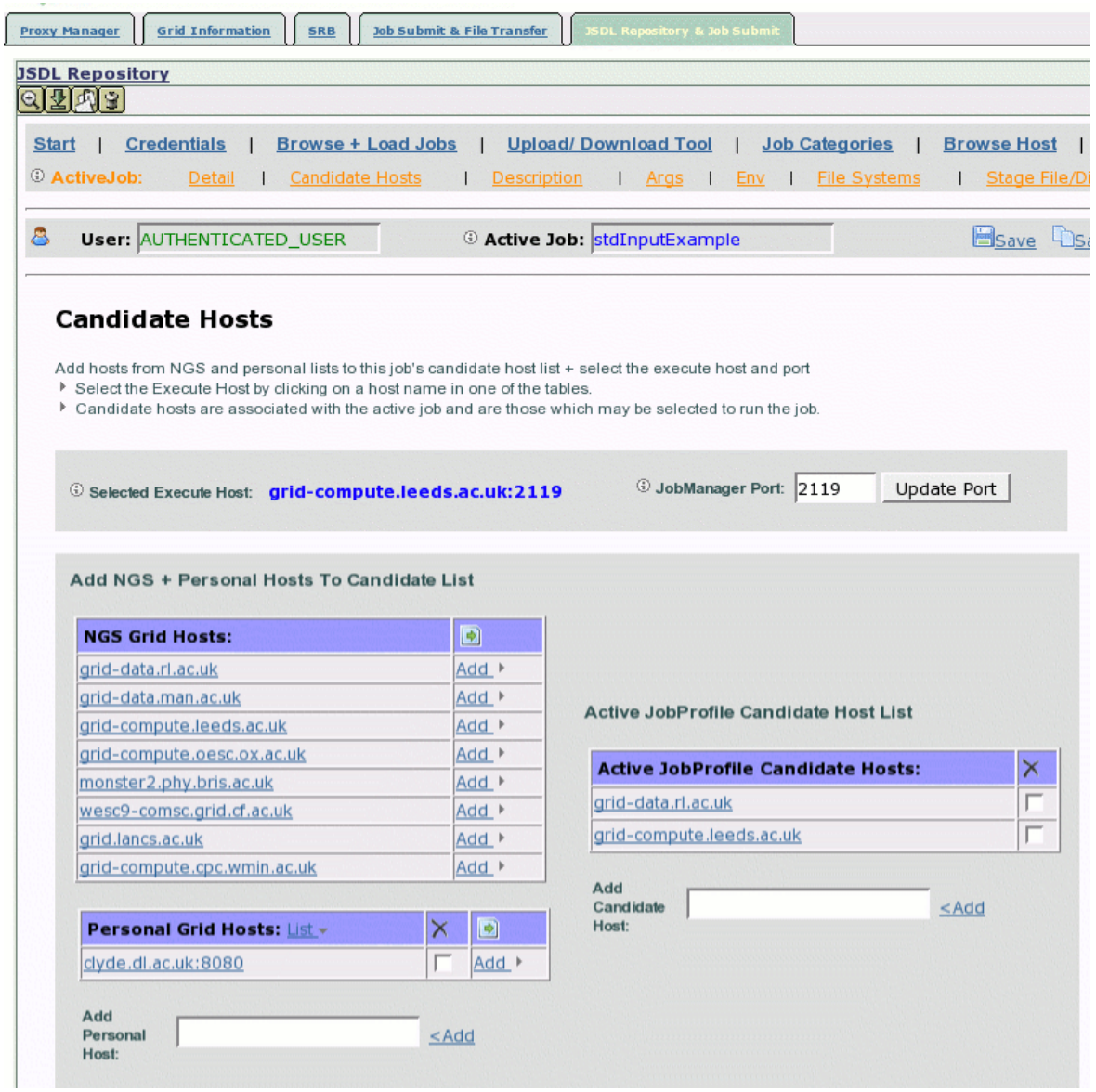
Select files and directories that should be copied to the consuming system via GsiFtp (more protocols to be supported inc srb, ftp, webdav).



Candidate Hosts

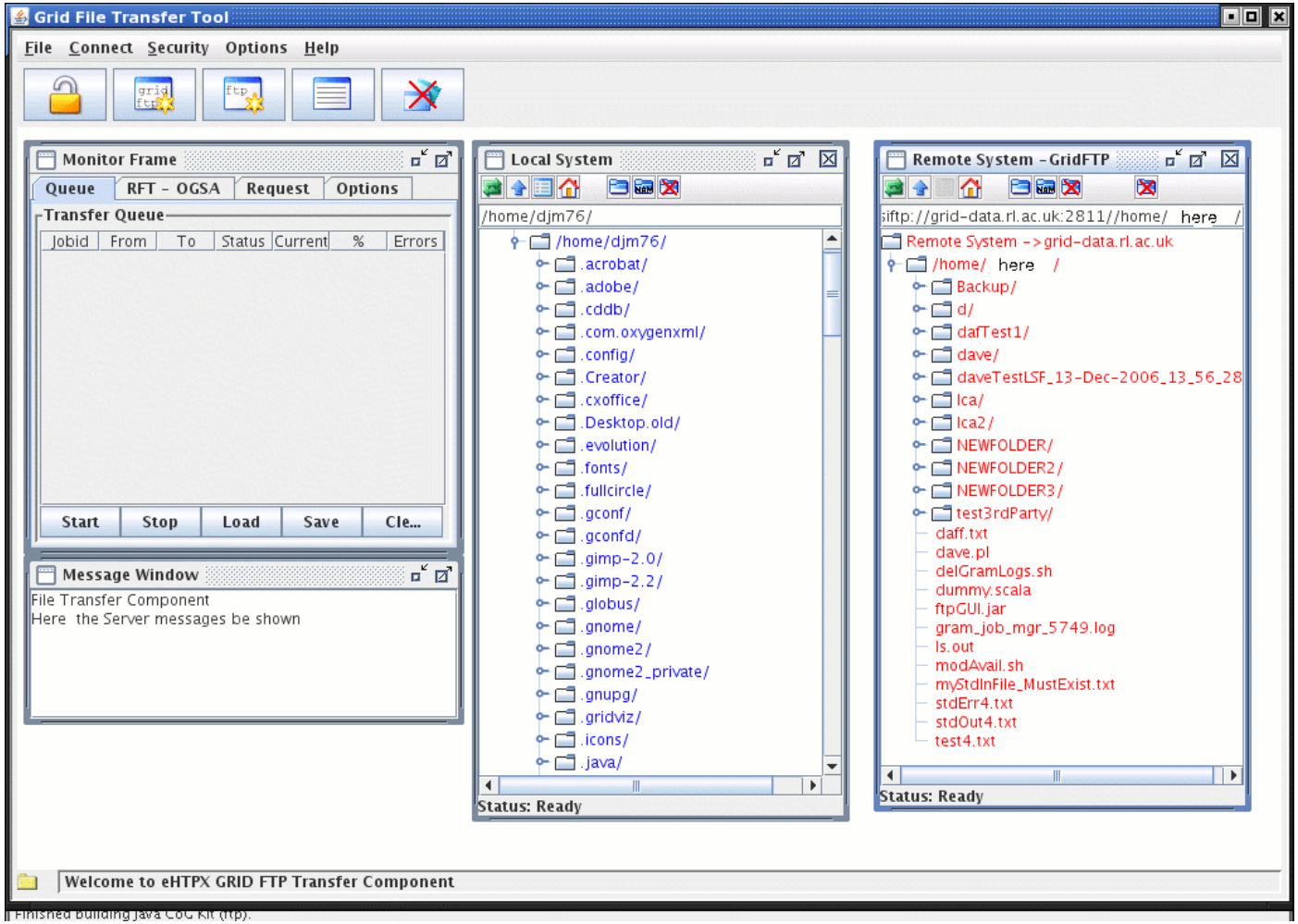
Candidate hosts are consuming systems that can be nominated to run the Active Job.

The candidate host list can be compiled from a personal host list and from a default host list (available to all users).



GridFtp Upload / Download / File Transfer Tool

Transfer files to/from your desktop and a remote Grid ftp server via ‘drag-n-drop’ – akin to Windows explorer for the Grid



Summary

1. Core of the application is designed to be generic and not tied to any particular set of Grid technologies (facilitated using JSDL). Middleware dependencies emerge at job-submit/monitor time when the specifics of the middleware have to be accommodated (e.g. parsing the JSDL into RSL, adding mw-specific parameters, e.g. RSL JobType).
2. Application can be deployed as a JSR-168 portlet or as a standalone Web application. This helps deployment – e.g. openPortal that was designed to show users what is available before having to 'log-in' (encourage users).
3. Currently, application only supports Globus, but GridSAM will be added shortly (more on GridSAM next few days).

TODO / Future

1. Extend application to support more Data Grid + Web protocols for data staging (SRB, WebDav). This will involve browsing / interacting with different data protocols in the interface (e.g. browse SRB), but deciding who actually performs the stating is currently being decided (manage in application, or leave to job submission service). Crossing protocols adds some complexity.
2. Growing list of improvements / suggestions to refine interface – HCI (Human computer interaction).
3. Extend the interface for Data visualisation via the portal interface.
4. Release the portlets for use in other projects / Grids.
5. Longer term - Extend the portlet to support the registration of new interfaces, i.e. register application specific interfaces designed for specific applications. (Note, new interfaces will build JSDL in same way – will use underlying portlet functionality in same way).
6. A portal is only as good as the underlying deployed infrastructure…. portal development often involves debugging the underlying consuming systems and middleware

Software Stack

JSF (Java Server Faces) interface + MVC control layer (Http session and request scope data)

Spring v2.0 managed business objects (singleton + prototype injected object graphs, declarative transaction demarcation, data source management).

C3p0 db connection pooling

ORM (object relational mapping) - JPA (Java persistence API) + Hibernate 3.2 for domain model.

Java CogKit for Globus API

Apache XMLBeans for JSDL xml-object data binding framework