HPC Developer’s Guide

Eclipse IDE Installation and Use

Version 1.0

Table of Contents

About this Guide

Informatics for Integrating Biology and the Bedside (i2b2) is one of the sponsored initiatives of the NIH Roadmap National Centers for Biomedical Computing (http://www.bisti.nih.gov/ncbc/). One of the goals of i2b2 is to provide clinical investigators broadly with the software tools necessary to collect and manage project-related clinical research data in the genomics age as a cohesive entity—a software suite to construct and manage the modern clinical research chart.

The guide provides installation and source code compilation and building steps for the High-Throughput Computing (HPC) Cell of the i2b2 hive. This cell allows researchers to directly access any remote computing resource (presumably, but not limited to, compute clusters).

Before using this guide, please install the HPC cell as per instructions provided in the document **HPC Installation Guide**.

1

Prerequisites and Third-Party Software

Downloads and Installation

a. i2b2 Workbench version 1.3

Download i2b2 Workbench version 1.3 (i2b2Workbench-src-13.zip) from https://www.i2b2.org/software/repository.html?t=demo&p=14. Follow installation and configuration instructions as given in the *i2b2 Workbench Developers' Guide v1.3* which can be found under the Docs tab.

b. Java JDK 5.0 – needed for i2b2 Workbench

This version of the DJK is needed for running the Eclipse Workbench. Download JDK 5.0 Update 11 (jdk-1_5_0_11-windows-i586-p.exe) from http://java.sun.com/products/archive/

Run the installer. Setup JAVA_HOME and CLASSPATH environment variables after installation.

c. Eclipse

You will need to use version 3.2.1 or later of the Eclipse SDK (e.g., eclipse-SDK-3.2.1-win32.zip), which can be found at http://archive.eclipse.org/eclipse/downloads. If you install Eclipse, be sure to install it in an area separate from any previous Eclipse installations.

To install, extract the zip file into a directory on local disk. Create a local desktop shortcut to eclipse.exe.

2

Install & Configure

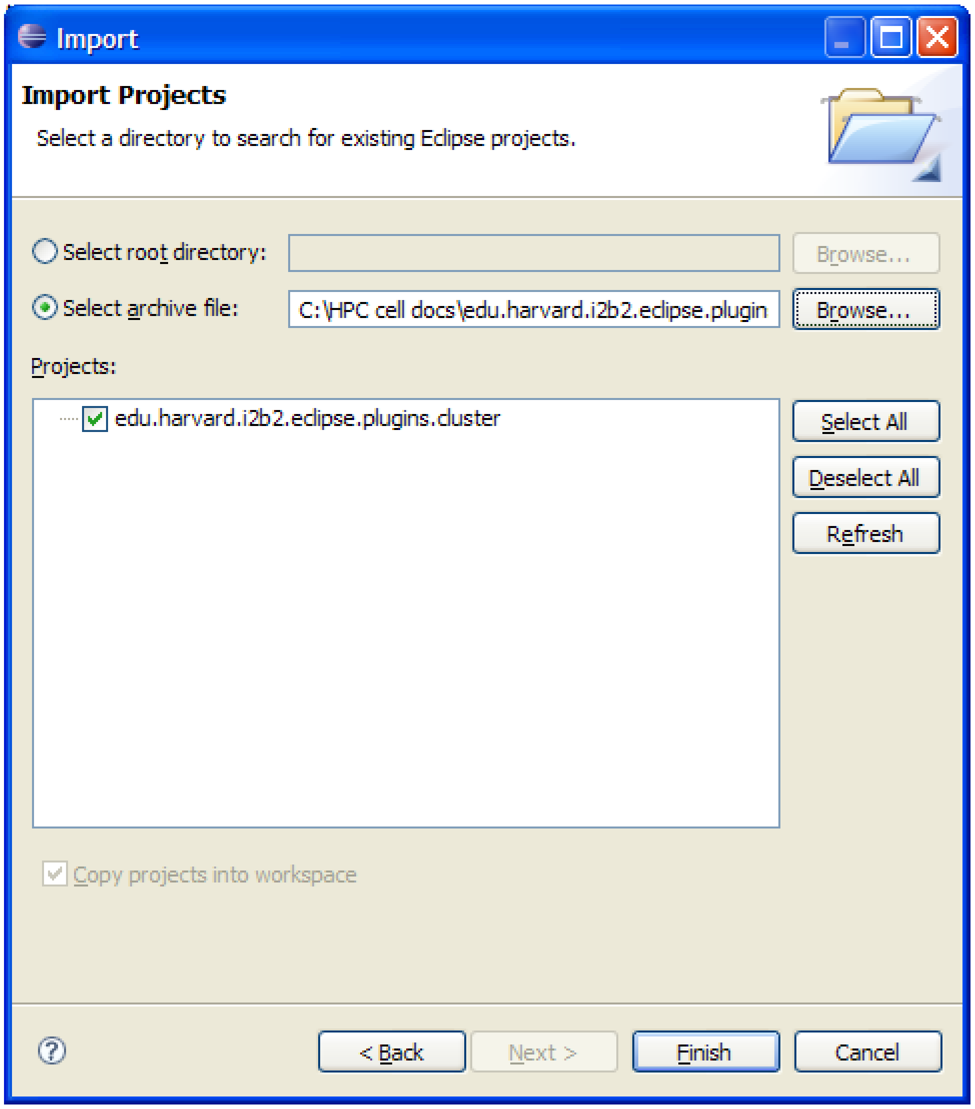
Setting up the HPC Cell within eclipse

You should have already installed the Eclipse 3.2.1 environment and imported the i2b2 Workbench v1.3 set of projects into it as detailed in the *i2b2 Workbench Developers' Guide v1.3.*

Importing the HPC project

This project software is contained in the Eclipse archive file called "edu.harvard.i2b2.eclipse.plugins.cluster.zip" which can be downloaded from the i2b2 web site at http://i2b2.org

From the File Menu of eclipse select Import->General->Existing Project into Workspace and navigate to the place where the archive was saved to disk. The archive contains a single Eclipse plug-in for the cluster access software.


Import

Import Projects

Select a directory to search for existing Eclipse projects.

( ) Select root directory:

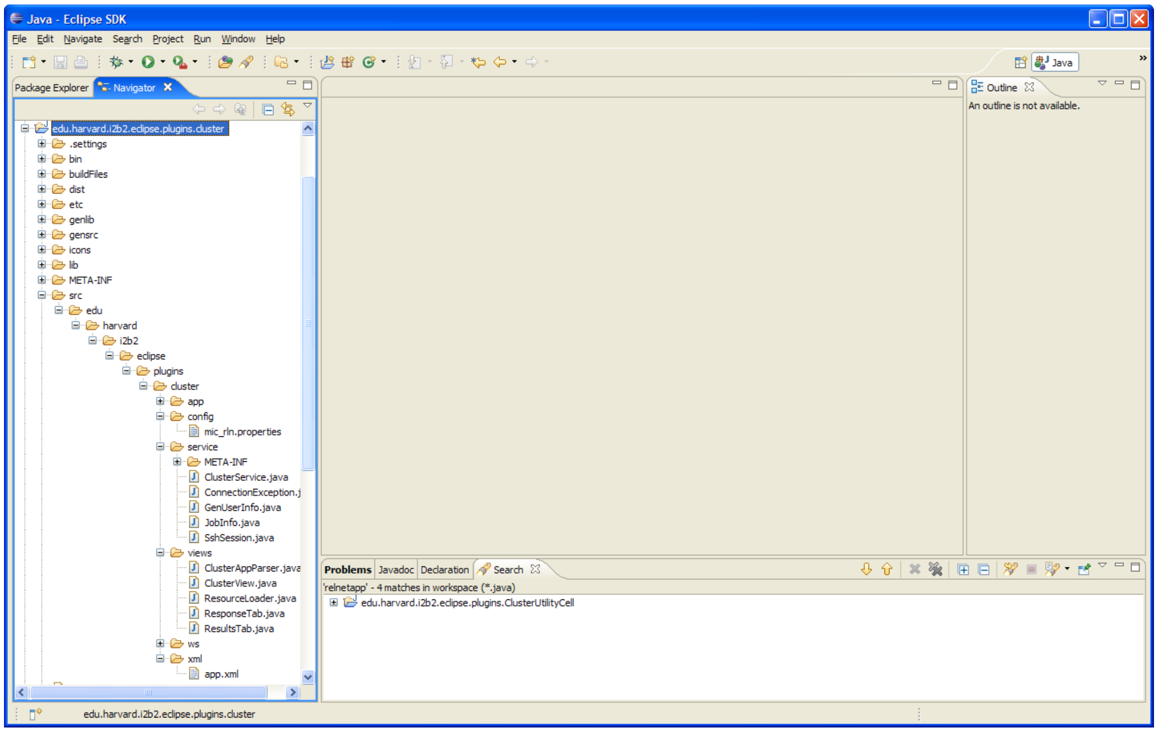
(●) Select archive file: C:\HPC cell docs\edu.harvard.i2b2.eclipse.plugin

Projects:

☑ edu.harvard.i2b2.eclipse.plugins.cluster

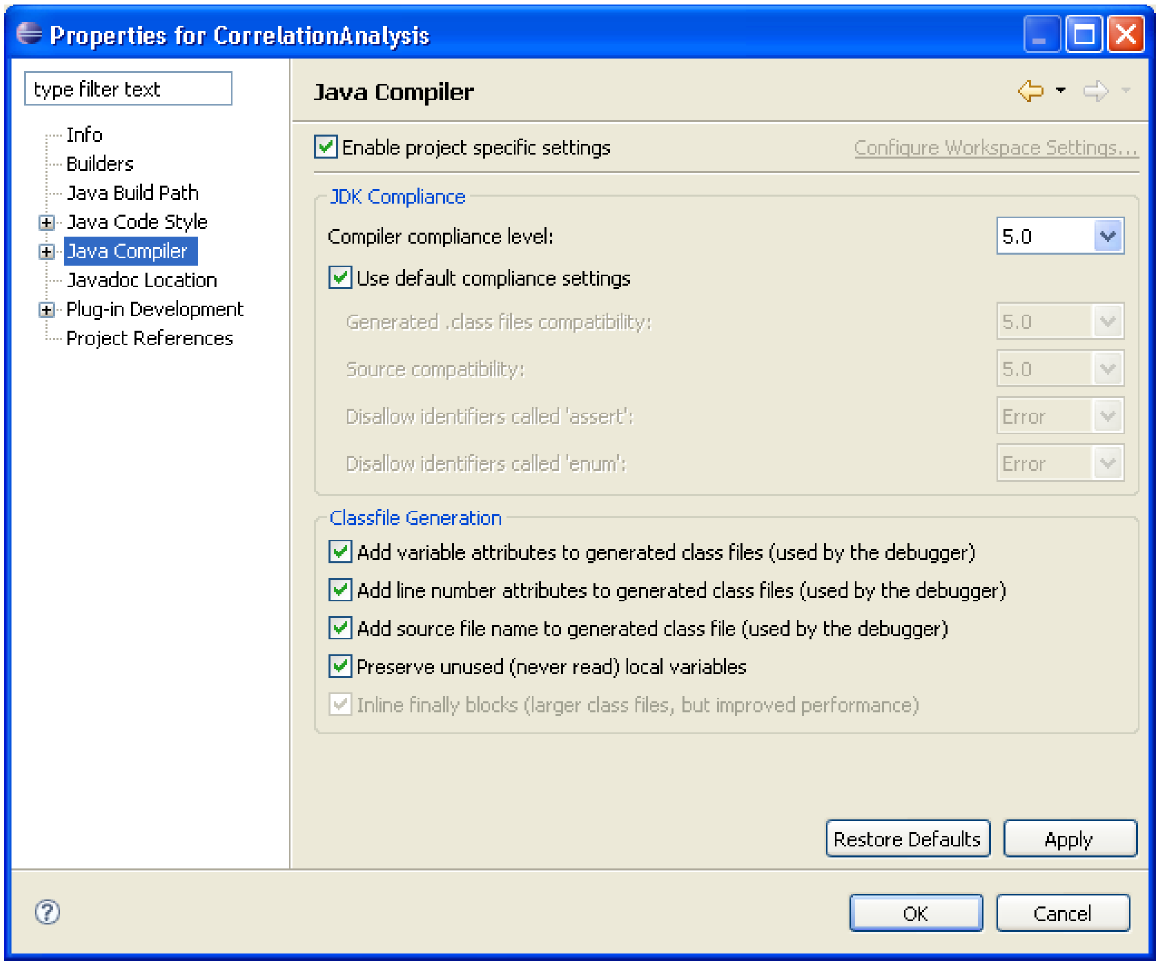
☑ Copy projects into workspace

Click Finish and let the i2b2 Workbench import the HPC archive. When done, you should see something similar to the screen-shot below.



Configuring the HPC Project within the i2b2 Workbench

Check your Java Complier settings. Choose Project->Properties from the main drop down menu. Next click on 'Java Compiler'. Verify that Compiler compliance level is set to 5.0. If not, select it from the drop-down menu and select Apply->OK.


Properties for CorrelationAnalysis

type filter text

- Info
- Builders
- Java Build Path
- Java Code Style
- Java Compiler
- Javadoc Location
- Plug-in Development
- Project References

Java Compiler

[✓] Enable project specific settings — Configure Workspace Settings...

JDK Compliance

Compiler compliance level: 5.0

[✓] Use default compliance settings

Generated .class files compatibility: 5.0

Source compatibility: 5.0

Disallow identifiers called 'assert': Error

Disallow identifiers called 'enum': Error

Classfile Generation

[✓] Add variable attributes to generated class files (used by the debugger)

[✓] Add line number attributes to generated class files (used by the debugger)

[✓] Add source file name to generated class file (used by the debugger)

[✓] Preserve unused (never read) local variables

[✓] Inline finally blocks (larger class files, but improved performance)

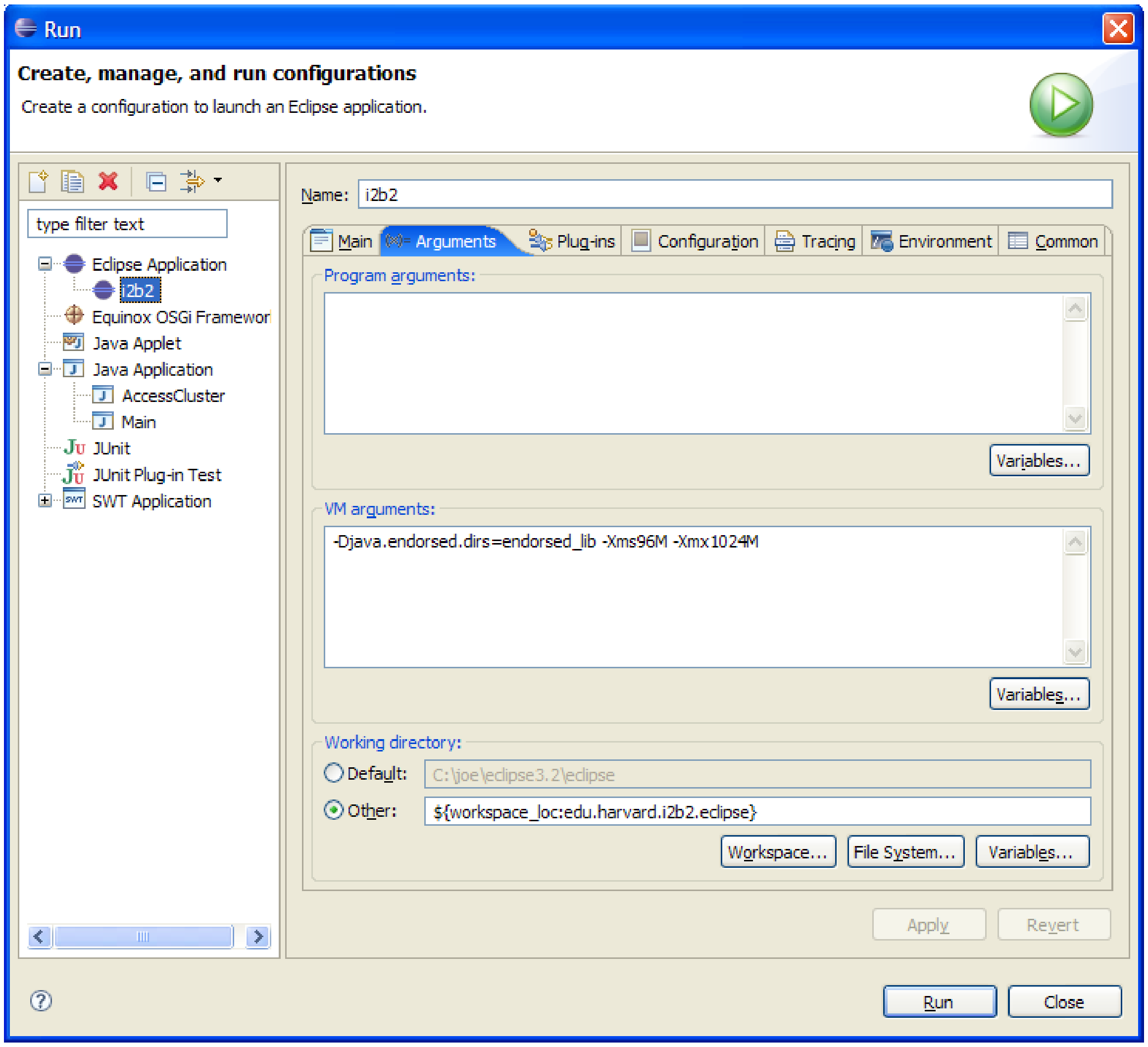
Change amount of memory allocated to JVM

Display current run-time configuration parameters by selecting Run from the i2b2 Workbench menus and choosing the Run… option. Select the Arguments tab on the right. Add these suggested memory management settings to the VM arguments area:

-Xms96M -Xmx1024M

First corresponds to the starting and second to maximum size of the heap size the JVM operates with. When not enough heap space, JVM will throw following error: Exception in thread "main" java.lang.OutOfMemoryError: Java heap space.

NOTE: the maximum heap size of 1024 M would normally correspond to a total of 1.5 Gb or more physical RAM available on a Windows-based platform. For other platforms or when different amount of physical memory is available, please make changes to these suggested numbers accordingly.


Run

Create, manage, and run configurations

Create a configuration to launch an Eclipse application.

Name: i2b2

VM arguments:

```
-Djava.endorsed.dirs=endorsed_lib -Xms96M -Xmx1024M
```

Working directory:

Default: C:\joe\eclipse3.2\eclipse

[✓ Other:] ${workspace_loc:edu.harvard.i2b2.eclipse}

Select Apply and Close.

Update plugin.xml file in **edu.harvard.i2b2.eclipse** directory

Make following change to the plugin.xml file in the main working directory so that the HPC Cell be selectable from the Window->Show view menu of the i2b2 Workbench.

Add

```
<viewShortcut
id="edu.harvard.i2b2.eclipse.plugins.cluster.views.clusterView"/>
```

in the section

```
<perspectiveExtension
targetID="edu.harvard.i2b2.eclipse.perspective">
```

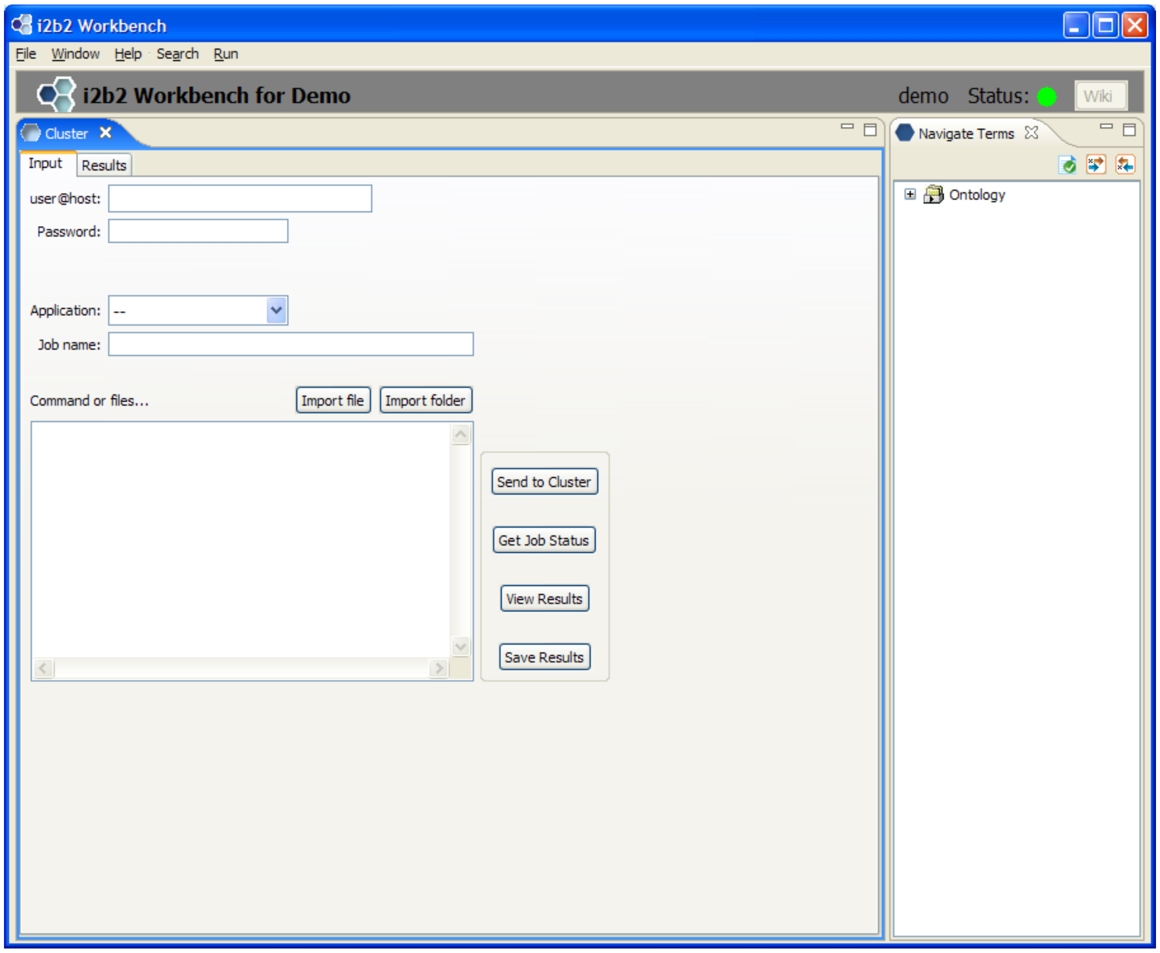
Building ClusterService.aar

```
Now, you should build the ClusterService.aar.  Edit the file 'jar-
service.xml' and specify your tomcat user, password, and directory.
Then run the jar-service.xml ant file by right-clicking and choosing
'Run As -> Ant Build'.  This will create the web service and deposit
into your tomcat installation.  You are now ready to run the HPC plug-
in client.
```

Running the HPC plug-in within the i2b2Workbench

Run the application by clicking on the **Run** menu button. One should see a similar screen to this screen shot configuration of windows.

If not, ensure the HPC view is visible: View -> Window -> Show View -> ClusterView



user@host:

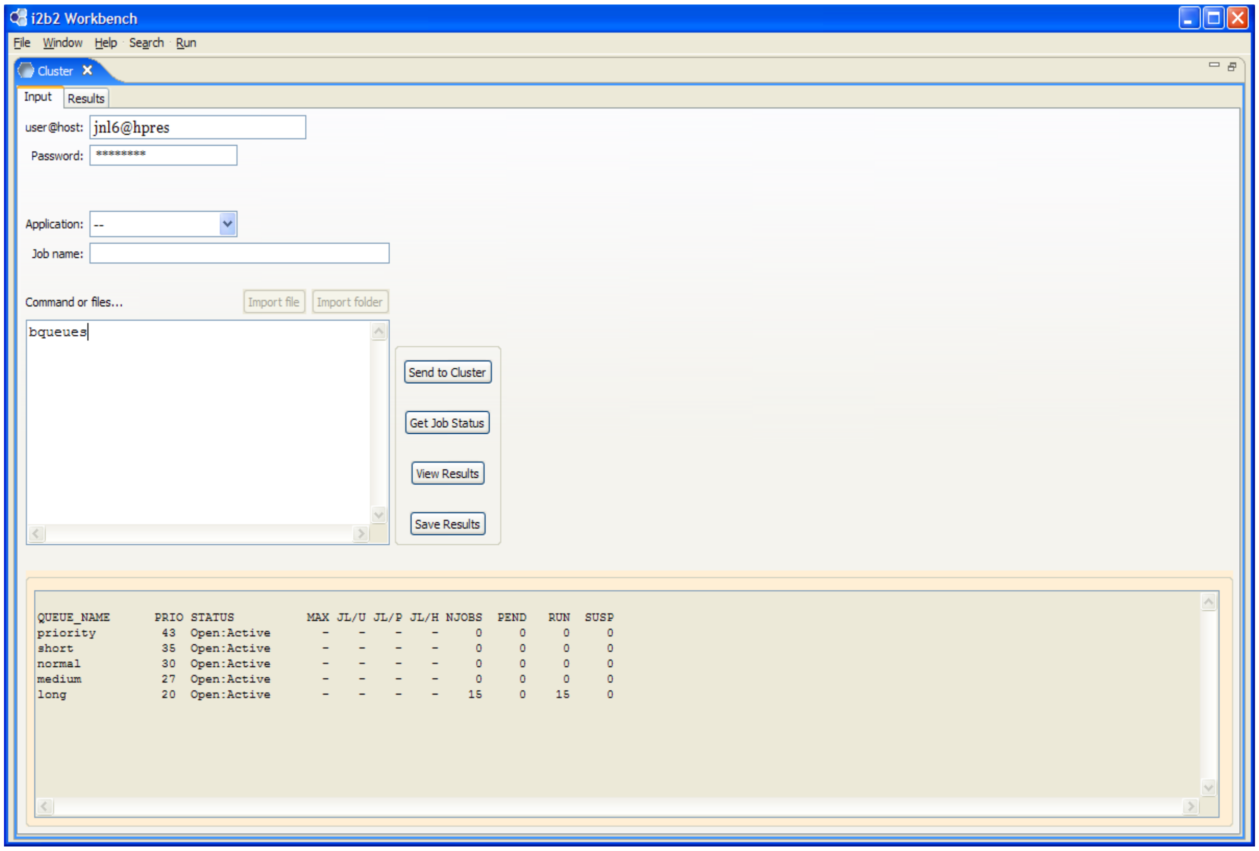
Password:

Application: --

Job name:

Command or files...

At this point, the user can access the cluster by entering a valid user@host and password in the appropriate fields. Leave the Application pull-down set to “--” and type in a command to run on the cluster, in this example, 'bqueues' to see the state of any running jobs on the cluster. Click 'send to cluster' to send the command and retrieve results:



user@host: jnl6@hpres
Password: ********
Application: --
Job name:
Command or files...

```
bqueues
```

```
QUEUE_NAME      PRIO STATUS          MAX JL/U JL/P JL/H NJOBS  PEND   RUN  SUSP
priority         43  Open:Active       -    -    -    -     0     0     0     0
short            35  Open:Active       -    -    -    -     0     0     0     0
normal           30  Open:Active       -    -    -    -     0     0     0     0
medium           27  Open:Active       -    -    -    -     0     0     0     0
long             20  Open:Active       -    -    -    -    15     0    15     0
```

For more involved applications, use the XML formatted messaging described below to define the scope of the application.

XML Definitions

In order to communicate with the HPC cell, the developer must provide an XML file describing the function(s) to be performed. HPC cell expects this xml file to be called app.xml and to reside in the HPC project's xml folder. More detailed instructions can be found in the **HPC XML Syntax Definition** document.

XML Syntax

We will use a simple "Hello World" example to demonstrate functionality. This example can be found in the app.xml file.

```
<App>
<ClusterApp name="HelloWorld">

 <!-- local tomcat web service data -->
 <local-resource>
  <server>localhost</server>
  <port>8080</port>
 </local-resource>

 <!-- remote resource -->
 <remote-resource>
  <server>hpres</server>
  <user>jnl6</user>
 </remote-resource>

 <preprocess>
  <concat>
   <command>mkdir pl</command>
  </concat>
  <ftp action="put" remotedir="pl">
   <fileset dir="c:\joe\perl">
    <include name="hello.pl" strip-return="true"/>
   </fileset>
  </ftp>
 </preprocess>

 <!-- aplication parameters -->
 <text label="iterations">10</text>

 <submit type="bsub">
  <app name="pl/hello.pl">
  </app>
 </submit>
</ClusterApp>
</App>
```

```
The root is <App>, while each separate cluster application is defined
in a <ClusterApp> element.  Within this element, a <local-resource>
element is required to define the user's local tomcat setup.  Also, a
```

<remote-resource> element is required to define the destination cluster resource resides.

What follows are all actions to be performed on the remote resource.

The <preprocess> element describes functions that need to be performed before application execution. Here, <concat> contains one or more <command> elements to be executed. Also, an <ftp> action to copy files is included. Here, we are simply ftp'ing a perl script:

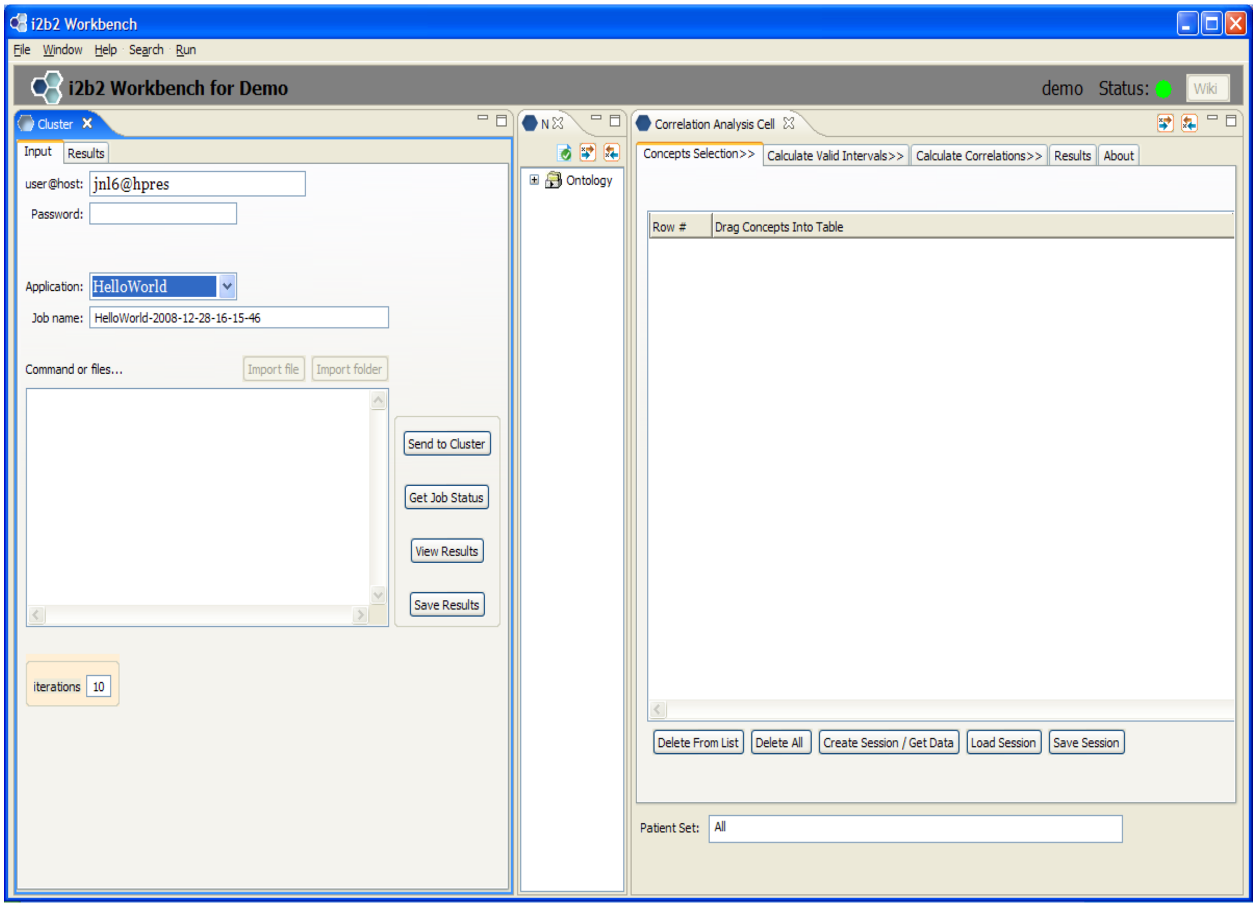
```
hello.pl:
#!/usr/bin/perl
$num = shift;
for $i (1 .. $num) {
      print "Hello, Cluster\n\n";
}
```

Be sure to place this in a file that you specify in the <fileset> element; the 'strip-cr' attribute is necessary to remove any MS carriage returns which will impair execution of the perl script on the remote machine.

Next, a <text> element will be displayed in the HPC cell, which is the input parameter to the perl script; it provides a default value of "10" which the user can modify in the cell.

The <submit> element describes that the task in question is to be submited via bsub (to the LSF queue on the cluster); the application is given in the <app> element.

Job submission occurs when the user chooses "HelloWorld" in the drop-down menu in the HPC cell, and then clicking the 'send to cluster' button.



user@host: jnl6@hpres
Password:
Application: HelloWorld
Job name: HelloWorld-2008-12-28-16-15-46
Command or files...
iterations 10
Patient Set: All

```
Clicking 'job status' will retrieve information about whether the job
is done or not.

One of the features of HPC cell is that it stores the last run you made
for a particular application, so that you can shut down i2b2 and return
at a later time, then choose your application from the drop-down menu;
when clicking 'get job status', HPC cell will populate the job name
field with the last job submitted and process the user's request.

The user could then either view the results of the run (basically the
stdout from the application) or save the results to a local file.
```