



## WG-7 Simulation specific portal components and documentation<sup>1</sup>

Deliverable	7.5
Authors	Oliver Wehrens
Date	April 7, 2008
Document Version	1.0
Current Version	1.0
Previous Versions	0.2

### A: Status of this Document

Final version.

### B: Reference to project plan

This deliverable refers to the task TA VII-4 "*Erstellung von Benutzerschnittstellen fuer Deployment, Ueberwachung und Steuerung von Grid-Simulationen*".

### C: Abstract

This document describes the needed steps for creating a portal interface for a use case in AstroGrid-D.

---

<sup>1</sup>This work is part of the AstroGrid-D project and D-Grid. The project is funded by the German Federal Ministry of Education and Research (BMBF).

**D: Changes History**

<b>Version</b>	<b>Date</b>	<b>Name</b>	<b>Brief summary</b>
0.1.0	January 23 2008	Oliver Wehrens	Working Draft Creation
0.2.0	February 5 2008	Oliver Wehrens	Changes to style and some corrections
1.0.0	March 28 2008	Oliver Wehrens	Final version

E:

## Contents

<b>1</b>	<b>Introduction</b>	<b>4</b>
<b>2</b>	<b>How to prepare a use case for the portal</b>	<b>5</b>
<b>3</b>	<b>API for your Grid Portlets</b>	<b>7</b>
3.1	Credentials . . . . .	7
3.2	Job submission . . . . .	7
<b>4</b>	<b>Use case Clusterfinder</b>	<b>9</b>
4.1	Running <i>Clusterfinder</i> from the commandline . . . . .	9
4.2	The Portal Interface . . . . .	11
4.3	The Portal Code . . . . .	11
4.4	Remarks . . . . .	13
	<b>References</b>	<b>14</b>

## 1 Introduction

Within the AstroGrid-D community project, there was a need for developing interfaces for existing grid enabled use cases to provide an easy to use access.

This document describes

- requirements to the portal from the community
- what is needed from your use case to enable it for the portal
- API for programming the portlet
- an example use case *Clusterfinder*

## 2 How to prepare a use case for the portal

This section describes how a use case can be prepared to ease the development of the corresponding portlet.

The G4P API in the portal is basically emulating an RSL script. It constructs out of the given parameters by the portlet a valid job description and uses `globusrun-ws` to submit the job.

If the application can be started with something like

```
globusrun-ws -submit -f my.rsl -J -S -F mymachine.astrogrid.de
```

then the part on the use case itself is almost done and the portal developer can take over to work on the portlet.

One thing is not yet implemented by the API in the portal which is the deletion of files in the RSL script. This has yet to be done.

A description of the usual workflow with the application will be very helpful to the portal developer, like:

- How many input files you have
- How many parameters you have
- Where should they be staged in/out to

Some of that can be answered from looking at the RSL script. But there are also other questions which need to be answered

- How big are the files usually
- What does the scientist do with them
- Does he need to download the files to his local machine (or some other vis machine maybe)
- Does he do the same kind of runs often and only with minimal changing parameters
- How long is a run typically (or a range)
- Should he share his results with others
- Should he share parameters with others
- Should only he see his runs

These questions (among others) are very important to think about in the first place. They can substantially influence the code written by the portal developer. If there are already some ideas on features which might be useful in the future it is necessary to communicate that before the coding begins. Depending on the requests they can be harder and harder to implement.

If the use case in question is not (yet) a scriptable solution it is required to make sure the job can be started with a RSL script. This will avoid development problems in the portal if it turns out that the script does not work in the first place.

Assumptions about a certain environment can be made very rarely, e.g. the RSL script contains something like:

```
<directory>MYUSECASE</directory>
```

it can not be guaranteed that this directory exists.

A general advice is to stage 2 files to the target machine: one scriptfile (shell) and one tarfile.

Once it is staged the scriptfile gets executed and usually unpacks the tarfile and does all the steps needed to run your usecase (e.g. creating a directory)

To be really safe one needs to give this script/tar file a unique name (like appending a random number) to avoid conflicts on those clusters where you can run multiple jobs at the same time (otherwise both jobs would use the same directory, and depending on your directory structure and usecase, they can conflict with each other). This also helps when a job is submitted to a metascheduler like GridGateWay where one just doesn't know beforehand where the job will end up.

If the code needs to be checked out of SVN (by the staged-in script), it is advisable to make sure the job can read the SVN repository without any authorisation (unless a password to that SVN in the commandline is specified in that script or a `.svnpass` was copied over to the target machine as well).

### 3 API for your Grid Portlets

The Grid 4 Portals (G4P) add-on for GridSphere offers an API for accessing the main functionality in a Globus 4 based grid. This work is based on GridPortlets 1.4 and was ported to work with GridSphere 3.

This section describes some examples of the API to show what is possible in the portal to achieve.

#### 3.1 Credentials

If G4P is deployed one can configure that a user at login automatically retrieves his credential after he specified the parameters once. From there on he will get a valid credential from the myproxy[1] server upon login.

Since the credential can be expired it is advisable to check for a valid token.

Listing 1: Credential check

```
1 protected boolean checkCredentials(User user) {
2     List<CredentialContext> credentials =
3         credentialManagerService.getCredentialContexts(user);
4     for (CredentialContext credential : credentials) {
5         if (credential.isActive()) return true;
6     }
7     return false;
8 }
```

#### 3.2 Job submission

The G4P API enables the programmer to submit jobs to Globus 4 based resources.

Listing 2: Jobsubmission

```
1 /**
2  * Provides an example of submitting "/bin/lis" for a given user to
3  * a given host with a given directory. Illustrates how to set the
4  * directory in which to run a job and get the job output as a string.
5  * Returns null if the job status is failed when the job ends.
6  * @param user The user
7  * @param host The host on which to run /bin/lis
8  * @param directory The directory in which to run /bin/lis
9  * @return Returns the value return by the job, null if the job failed.
10  * @throws PortletException If a exception occurs during job submission
11  * or getting job output.
12  */
13 public String doBinLis(User user, String host, String directory)
14     throws PortletException {
15     PortletLog log = SportletLog.getInstance(getClass());
16     String answer = null;
17     try {
18         // Create a job spec using the generic job type
```

```
19 // Note one could specify a particular job type if so desired
20 log.debug("Creating job spec for " + user.getUserName());
21 JobSpec jobSpec = jobSubmissionService.
22     createJobSpec(JobType.INSTANCE);
23 jobSpec.setUser(user);
24 FileLocation executable = new FileLocation("/bin/ls");
25 jobSpec.setDirectory(directory);
26 jobSpec.setExecutableLocation(executable);
27 jobSpec.setHostName(host);
28 log.debug("Submitting /bin/ls to " + host + " in " + directory);
29 Job job = jobSubmissionService.submitJob(jobSpec);
30 job.waitFor();
31 if (job.getTaskStatus().equals(TaskStatus.FAILED)) {
32     log.error("Job failed with message " +
33         job.getTaskStatusMessage());
34     return null;
35 }
36 FileLocation stdout = job.getStdoutLocation();
37 if (stdout != null) {
38     log.debug("Reading stdout from " + stdout.getUrl());
39     FileHandle fileHandle = new FileHandle(stdout);
40     answer = fileHandle.readContents(user);
41 }
42 } catch (JobException e) {
43     log.error("Unable to submit job", e);
44     throw new PortletException(e.getMessage());
45 } catch (MalformedURLException e) {
46     log.error("Unable to create file location", e);
47     throw new PortletException(e.getMessage());
48 } catch (IOException e) {
49     log.error("Unable to read stdout location", e);
50     throw new PortletException(e.getMessage());
51 }
52 log.debug("answer = " + answer);
53 return answer;
54 }
```

Here a method is specified to return the content of a given directory on a remote host. The code is easy to understand and self explanatory. As one can see it just takes a couple of lines to submit a job (in this case /bin/ls) and return a result.

On the developer pages<sup>[3]</sup> of Gridsphere there are more examples of the API usage.

This documentation was written for GridPortlets 2.x. It does however apply for G4P as well since nothing but the package name changed.

## 4 Use case Clusterfinder

The use case *Clusterfinder* from the AstroGrid-D partner MPE was the first for which we developed a portlet interface to demonstrate the possibilities within the AstroGrid-D community.

### 4.1 Running Clusterfinder from the commandline

The code needs several settings to be run correctly.

So far a user had to do the following to execute the code:

- check the code out of the AstroGrid-D SVN with username and password
- set several environment variables
- run two make command with certain options

The parameters for the search are specified in an input file and describe a rectangle of the sky.

To automate this procedure it was needed to develop a script solution. The main idea behind this is that two files are staged in. There is a tar file containing the source and a shellsript doing the necessary settings. It was choosen to stage in a tar file containing the sources because checking out the code out of SVN requires a username/password which can't be supplied and it was not possible to develop a solution with a real username/password to allow checking out the code.

The bootstrap shell script looks like this:

Listing 3: Shellsript Clusterfinder

```
1 #!/bin/sh
2
3 # unpack the source
4 export dir=clusterfinder
5
6 if [ -d "$dir" ]; then echo "$dir exists.";
7   else tar xzf clusterfinder.tgz; fi
8
9 cd clusterfinder
10
11 export ID=$1
12 export ra0=$2
13 export ra1=$3
14 export de0=$4
15 export de1=$5
16
17 export X509_USER_PROXY='$GLOBUS_LOCATION/bin/grid-proxy-info \
18   | grep x509|awk '{print $3}'
19
20 export FC=gfortran
21 export F_PORTABILITY_FLAGS=-DPLANCK_GFORTRAN
22
23 make CAT=rass PAT=$1 inputdata
```

```

24 # wait for a few seconds until OGSA DAI has transfered the input data
25 sleep 5
26
27 # copy input file
28 mkdir -p parameters/rass/
29
30 sed "s/%%RA0%%/$ra0/" ~/ClusterFinderParamTemplate > parameters/rass/$ID.tmp1
31 sed "s/%%RA1%%/$ra1/" parameters/rass/$ID.tmp1 > parameters/rass/$ID.tmp2
32 sed "s/%%DE0%%/$de0/" parameters/rass/$ID.tmp2 > parameters/rass/$ID.tmp3
33 sed "s/%%DE1%%/$de1/" parameters/rass/$ID.tmp3 > parameters/rass/$ID
34 rm parameters/rass/$ID.tmp?
35 make CAT=rass PAT=$ID map

```

The shellscript has 5 parameters which will be set by the portal. The first is the name of the parameterfile and the last 4 describe the rectangle coordinates which it will search the sky for.

If the source code does not yet exist the tarfile will be unpacked, and several environment variables will be set in order to build and execute the code (lines 6-21).

The template parameter file will be copied to the corresponding directory and with the help of the sed commands the correct paramters will be subsituted in the file (lines 30-33).

At the end the program will be executed via the make command.

Now it is possible to run the script from the commandline with a specified RSL script.

This script looks like the following and needs to be emulated by the portal in order to run the use case from a webbrowser.

Listing 4: Clusterfinder RSL script

```

1
2 <?xml version="1.0" encoding="UTF-8"?>
3 <job>
4   <executable>clusterfinder.sh</executable>
5   <argument>OK</argument>
6   <argument>14.02</argument>
7   <argument>-1.22</argument>
8   <argument>-1.22</argument>
9   <stdout>portal-stdOut</stdout>
10  <stderr>portal-stdError</stderr>
11  <jobType>single</jobType>
12
13  <fileStageIn>
14    <transfer>
15      <sourceUrl>
16        gsiftp://buran.aei.mpg.de/tmp/clusterfinder.tgz
17      </sourceUrl>
18      <destinationUrl>
19        file:///${GLOBUS_USER_HOME}/clusterfinder.tgz
20      </destinationUrl>
21    </transfer>
22    <transfer>
23      <sourceUrl>
24        gsiftp://buran.aei.mpg.de/tmp/clusterfinder.sh

```

```

25     </sourceUrl>
26     <destinationUrl>
27         file:///${GLOBUS_USER_HOME}/clusterfinder.sh
28     </destinationUrl>
29 </transfer>
30 <transfer>
31     <sourceUrl>
32         gsiftp://buran.aei.mpg.de/tmp/ClusterFinderParamTemplate
33     </sourceUrl>
34     <destinationUrl>
35         file:///${GLOBUS_USER_HOME}/ClusterFinderParamTemplate
36     </destinationUrl>
37 </transfer>
38 </fileStageIn>
39
40 <fileStageOut>
41     <transfer>
42     <sourceUrl>
43         file:///${GLOBUS_USER_HOME}/clusterfinder/target/runs/RS22/maps/rass
44     </sourceUrl>
45     <destinationUrl>
46         gsiftp://buran.aei.mpg.de/tmp/RESULT-RS22/
47     </destinationUrl>
48 </transfer>
49 </fileStageOut>
50 </job>

```

All the parameters given to the shell script and the machine are included. Those have to be replaced by the portal with the input given by the user.

## 4.2 The Portal Interface

The corresponding portal code is fairly simply for the basic functionality. From the use case description it is required to have 4 input fields and a selector for a machine. Furthermore a 'run name' needs to be specified for each execution.

These parameters then are taken to construct the RSL file within the portal.

The visual interface is not very complicated and looks like figure 1.

If a job is submitted it will show up in the *Job submission portlet* which comes with Grid 4 Portals (figure 2).

## 4.3 The Portal Code

The main code for handling the form input is easy to understand:

Listing 5: Handling form input and submitting a job

```

1  public void submitJob(ActionFormEvent event) {
2      String hostname = event.getListBoxBean("hosts").getSelectedValue();
3

```

Figure 1: Running a clusterfinder job

Task submitted	Description	Host	Status
<input type="checkbox"/> Thu Jan 24 11:09:58 CET 2008	Clusterfinder run (OW, 14.02, 14.12 , -1.22 , -0.12)	hydra.ari.uni-heidelberg.de	Job completed with message success
<input type="checkbox"/> Thu Jan 24 10:51:22 CET 2008	Dir	hydra.ari.uni-heidelberg.de	Job completed with message success

Figure 2: Running a clusterfinder job

```

4   JobSpec jobSpec = jobSubmissionService.createJobSpec(
5       WsGramJobType.INSTANCE);
6   String stageoutHost = event.getListBoxBean("stageoutHosts").
7       getSelectedValue();
8   try {
9       jobSpec.setHostName(hostname);
10      jobSpec.setUser(getUser(event.getActionRequest()));
11      jobSpec.setExecutableLocation(new FileLocation("clusterfinder.sh"));
12      jobSpec.setExecutionMethod(ExecutionMethod.SINGLE);
13      jobSpec.setStderrLocation(new FileLocation("portal-stderr"));
14      jobSpec.setStdoutLocation(new FileLocation("portal-stdout"));
15      jobSpec.setArguments(event.getTextFieldBean("runname").getValue());
16      jobSpec.setArguments(event.getTextFieldBean("rastart").getValue());
17      jobSpec.setArguments(event.getTextFieldBean("raend").getValue());
18      jobSpec.setArguments(event.getTextFieldBean("destart").getValue());
19      jobSpec.setArguments(event.getTextFieldBean("deend").getValue());
20
21      jobSpec.addFileStageInParameter(
22          new FileLocation("gsiftp://buran.aei.mpg.de/tmp/clusterfinder.tgz"),
23          "clusterfinder.tgz");
24      jobSpec.addFileStageInParameter(
25          new FileLocation("gsiftp://buran.aei.mpg.de/tmp/clusterfinder.sh"),
26          "clusterfinder.sh");
27      // copy default template over
28      jobSpec.addFileStageInParameter(new FileLocation(
29          "gsiftp://buran.aei.mpg.de/tmp/ClusterFinderParamTemplate"),
30          "ClusterFinderParamTemplate");
31      jobSpec.addFileStageOutParameter(
32          new FileLocation("gsiftp://" + stageoutHost + "/" + "${GLOBUS_USER_HOME}"/),
33          "clusterfinder/target/runs/" +

```

```
34         event.getTextFieldBean("runname").getValue()+"/maps/rass");
35     jobSpec.setDescription(
36         "Clusterfinder run (" +event.getTextFieldBean("runname").getValue()
37         +", "+event.getTextFieldBean("rastart").getValue()+", "+
38         event.getTextFieldBean("raend").getValue()+", "
39         +event.getTextFieldBean("destart").getValue()+", "+
40         event.getTextFieldBean("deend").getValue()+")");
41     jobSubmissionService.submitJob(jobSpec);
42     appendInfo(event, "Job submitted.");
43 } catch (MalformedURLException e) {
44     appendError(event, "Job was not submitted.");
45     e.printStackTrace();
46 }
47 }
```

In line 3 the hostname of the target machine is saved and in line 4 a job specification object is obtained. This will hold all the values for the job and is submitted later to the job submission service.

Lines 9 to 19 set some variables like the script to be executed, the location of standard error and out, the parameters for the script.

Starting line 21 the staging parameters are set.

The job submission happens in line 41.

#### 4.4 Remarks

The code still needs a final touch and is not yet complete, e.g. the directory which the result files will be staged to needs to be configurable (or not). There is still an ongoing discussion with the clusterfinder people to decide how to implement some details and what will be the best way to do so. Here is a list of things which need some improvements:

- Places for the tar file and the shell script
- Naming for the standard out and standard error files
- The result file is not yet copied correctly since there is some misunderstanding where this will be stored on the computing machine (in progress)
- How the scientist will access the result file, so far it is copied to a gridftp resource, but I do imagine this might not be ideal

To resolve these questions a close collaboration between the stakeholders and the programmers is necessary. Only if both parties use a common language to describe the domain of the problem a satisfying solution will emerge.

## F: References / Bibliography

### References

- [1] MyProxy  
<http://grid.ncsa.uiuc.edu/myproxy/>
- [2] GridSphere Portal Framework  
<http://www.gridsphere.org>
- [3] GridPortlets Developer Documentation  
<http://www.gridsphere.org/gridsphere/docs-GS-2.2.X/gridportlets/docbook/DeveloperGuide/DeveloperGuide.html>
- [4] Working Group 2 of Astrogrid  
<http://www.gac-grid.de/project-overview/workpackages/wp2.html>