The Glueing Service
(User Manual)
GSOC 2008

Table of Contents

The four main actors, which play a vital role in the completion of the Glueing Service, are:

- Grid Middleware setup
- Installation and Testing SAGA APIs
- Web Service Interface
- Application Client

The technical aspects of each of each of the above-mentioned actors are described in detail in this document.

1. Grid Middleware setup

The Glueing Service uses OMII middleware for submitting and executing the job over the Grid. The Grird Submission and Monitoring Service (GridSAM) is integrated with the OMII middleware.

There are two ways to set up OMII middleware. The first one is to install the Campus Grid Toolkit and the second is to install OMII stack.

We have set up the middleware using the second approach i.e. by installing the OMII stack. The following are the steps of installing OMII middleware using OMII stack installer.

1. The JAVA_HOME environment variable should be defined and should point to the Java installation directory.

2. By executing “OMIIStackInstall.pl” scrip starts the installation process of OMII server.
3. The web service component of the OMII sever must be installed after the installation of the database i.e. postgres sql.
4. Once the installation of OMII server is complete then GridSAM can be integrated into the OMII server.
5. The OMII_HOME environment variable should be defined before the installation of GridSAM and OMII_HOME should point to the installation directory of the OMII service.
6. The OMII service should be stopped and the apache ant should be in the path before installing the GridSAM service.
7. The GridSAM service can be installed by running the following command, after unpacking the service tar

ant install -Domii.server.home=${OMII_HOME} -Dtomcat.home=${OMII_HOME}/apache-tomcat-x.x.xx

8. Once the GridSAM is successfully installed, the OMII service can be started by running the “startomii.sh” script, present at $OMII_HOME/bin.
9. The successful installation of GridSAM service can be checked by visiting the following URL in the web browser

https://HOST:18443/.gridsam/services/gridsam

1.1 Setting up GridSAM Client

GridSAM client is used to test GridSAM service, integrated with the OMII middleware, by submitting and running a job. The following are the steps, which are required to setup a GridSAM client

1. The first step is to download the OMII client and unpack the archived setup.
2. Install the OMII client by running the “OMIIClientInstall.sh” script.
3. After following the installation procedure create an environment variable OMIICLIENT_HOME which points to the installation directory of OMII client.
4. The next step is to download and unpack the GridSAM client archive.
5. To install GridSAM client OMIICLIENT_HOME environment variable must be defined and apache ant must be installed.

6. The GridSAM client can be installed using the following command

ant install -Domii.client.home=${OMII_CLIENT_HOME}

7. Once the GridSAM client is installed the test jobs can be submitted to the OMII middleware.

1.2 Submitting Job to OMII Middleware

The GridSAM service should be running prior to submit a job to the OMII middleware, using GridSAM client, and the following link to the WSDL should be accessible.

https://HOST:18443/grisam/services/gridsam?wsdl.

The GridSAM client submits a job to the OMII middleware by defining the job in JSDL (Job Service Description Language). JSDL is an XML format of defining a job, job input parameters and other dependencies of the job. Once the JSDL is defined the "gridsam-sbumit" command, provided by GridSAM client, can be used to submit a job to the OMII middleware. The syntax of the command is shown below:

```
${OMIICLIENT_HOME}/gridsam/bin/gridsam-submit \
  -s "https://HOST:18443/gridsam/services/gridsam" \
  "PATH_TO_JSDL_FILE"
```

If a job is successfully submitted to the middleware, the "gridsam-submit" command returns a job identifier on the console form where the job is submitted. This job identifier can be used to monitor the status of the job and control the execution of the job. The command to monitor the status of the submitted job is "gridsam-status". The syntax of gridsam-status is as follows:

```
${OMIICLIENT_HOME}/gridsam/bin/gridsam-status -s
"https://HOST:18443/gridsam/services/gridsam" \
  -j "JOB_IDENTIFIER"
```

The GridSAM client also provides different other commands as well, for example "gridsam-terminate" command is used to terminate a job and "gridsam-ftp-server" is used to setup an anonymous FTP server for staging out the results after the execution of the job.

2. Installing and Testing Java SAGA APIs

The section 1 discusses the process and steps of setting up OMII middleware and submitting a job to the middleware using GridSAM client application. In this section we will discuss the process of submitting a job to the OMII middleware using SAGA APIs. The SAGA (Simple API for Grid Applications) is a set of middleware adaptors and high level APIs for the effortless programming of Grid applications (discussed can be found in the Design Document of the Glueing Service).

2.1 Installing SAGA Java Implementations

The archive of SAGA Java Implementations can be downloaded from the LSU's website. The followings are the steps to use SAGA libraries with the OMII middleware:

1. First unpack the archive and define an environment variable called SAGA_LOCATION, which point to the unpacked/installation directory of Java Implementations of SAGA. The $SAGA_LOCATION/lib directory contains SAGA libraries and middleware adaptors. The libraries can be used for compiling the applications that use SAGA APIs. The middleware adaptors can be loaded into the application to communicate with the middleware.
2. The $SAGA_LOCATION directory contains a file "crypto.properties". This file contains the middleware information. As we are using OMII middleware to access grid resources therefore we have to change "crypto.properties" file according to middleware specific properties. The important information in the "crypto.properties" file is path to the keystore file, generated during the installation of the OMII middleware, and the path of CRLs directory.
    a. The path of keystore file is $OMII_HOME/omii.ks
    b. The path of CRLs directory is $OMII_HOME/grid/CRLs
3. The $SAGA_LOCATION/bin contains some scripts which can by used for executing a simple application/class that uses SAGA API functions. The use of these scripts, such as "run_saga_app", is optional. The source can be simply compiled by providing all the libraries in the class path but for

executing the class the $SAGA_LOCATION/crypto.properties and $SAGA_LOCATION/saga.properties files must be in the provided in the path using "-D" switch of java executable.

2.2 Testing SAGA API with OMII Middleware

The SAGA APIs provide standard interfaces to build Grid applications. The Glueing Service intends to submit jobs to the OMII middleware using the standard interfaces of SAGA APIs and dynamically loading the middleware (GridSAM) adaptor.

The prerequisites of building a SAGA application for submitting the job on OMII middleware are:

1. The OMII service must be running.
2. An FTP Server must be setup, with write permissions to the FTP user.

The steps of building an application for submitting a job to the OMII middleware are:

1. The first step is to set preferences for the middleware adaptor. In the current scenario, we have to set the preferences for the OMII middleware. Setting a system property "jobservice.adaptor.name" equal to "gridsam" can achieve this.
2. The middleware service URL is also required for creating JobService instance (details are described in the Design Document of the Glueing Service). The middleware URL in the current scenario is the service URL of the OMII middleware and can be accessible only if the OMII service is running.
3. The next step is related to the job specific credentials. In this step the job is defined using different attributes. The important attributes of the job are job executable, job input parameters and an endpoint for staging out the execution results of the job.
4. Once the job is executed the job Metric API of SAGA Implementations can be used to determine the status of job and its performance, based on CPU time and memory size etc.

3. Web Service Interface

The Glueing Service is a web service interface of the SAGA API functions. The Glueing Service provides one to one correspondence of the SAGA API functions and exposes almost all the functionality in the form of a web service. The process of creating the Glueing Service is discussed as under:

1. The first step is to write a complete web service interface, which describes one to one correspondence of the SAGA API calls.
2. The service interface is provided with appropriate functionality such as submitting a job over the Grid middleware, as discussed in section 2.2.
3. The service interface, known as the Glueing Service, is deployed under Apache Tomcat web server using Apache Axis deployment engine. In this step the SAGA and Axis libraries are used to compile the Glueing Service. The SAGA libraries are made available to the deployment engine by specifying the paths of the SAGA libraries in the "Catalina.sh" script which is present at $TOMCAT_HOME/bin directory and is executed to start the web server.
4. If the service interface or the Glueing Service is compiled successfully, then a WSDL of the Glueing Service can be generated with Java2WSDL tool that comes with Apache Axis in "org.apache.axis.wsdl.Java2WSDL" package.
5. The Glueing Service, which is a JWS (Java Web Service) file, is published using Apache Axis and the service WSDL becomes available at the URL endpoint of service. For example http://HOST: PORT/axis/GS.jws?wsdl.

4. Application Client

The application client is a web service client that is used to test the Glueing Service functionality during the course of development. This test client application will be replaced by the Pipeline Service (discussed in the Design Document of the Glueing Service). The client application generates SOAP messages based on the two essential things, which are web service endpoint URL and the name of published function. The SOAP message is invoked by providing the input parameters of the published function of the web service. The web

service returns the output of the function in SOAP response. The SOAP response is de marshaled by the client and the function output is used inside the client application. The client application uses Apache Axis client API, therefore the Axis libraries are used to compile and execute the client application.