

JLITE: A LIGHTWEIGHT JAVA API FOR GLITE¹

O.V. Sukhoroslov

*Centre for Grid Technologies and Distributed Computing,
Institute for Systems Analysis, Russian Academy of Sciences,
Prosp. 60-let Oktyabrya 9, 117312 Moscow, Russia
os@isa.ru*

We present jLite, a Java library providing simple API for accessing gLite based grids in a platform-independent way. jLite application areas include development of grid-enabled Java applications, cross-platform tools, grid portals and services.

1 Motivation and Approach

Existing Java APIs for gLite services are scattered among several packages with complex external dependencies. These libraries expose mostly low-level service operations and lack detailed documentation. Scarce API usage examples often imply the presence of gLite User Interface (UI) environment installed on Scientific Linux. This complicates the usage of these APIs for development of cross-platform grid applications.

jLite is addressing these problems by providing a high-level Java API with functionality similar to gLite UI commands which is sufficient for building many grid applications. This API hides the complexity of underlying middleware and its configuration. jLite is easy to install because it includes all external dependencies and does not require installation of gLite UI. The library is pure Java and can be used on any Java-capable platform including Windows.

2 Current Implementation

The “early preview” release of jLite is distributed under Apache License 2.0 at [1]. Current implementation supports complete gLite job management lifecycle including VOMS proxy creation, job registration, transfer of job sandbox files, job startup and status monitoring.

jLite is built on top of existing gLite Java APIs and many other libraries, such as Apache Axis, Java CoG Kit, Condor ClassAd, etc. The distribution includes all of the numerous external dependencies which have been tested to work together smoothly. The use of other versions of external libraries can cause errors due to compatibility issues.

3 Installation and Configuration

The installation of jLite is done by unpacking the distribution to any directory (referred further as JLITE_HOME) and configuring it to use desired grid user credentials and virtual organization (VO) services.

By default, jLite looks for user credentials in a home directory of the current user which is the standard location used by grid middleware. The location of user credentials can also be specified during the run time via jLite API.

¹ The work is supported by the RFBR (grant 08-07-00430-a), RAS Presidium (Programme 15II) and FASI (contract 02.514.11.4053).

For each VO to be accessed via jLite the user provides standard VOMS server configuration file, VOMS server certificate and WMProxy server location. Currently, only one WMProxy sever per VO is supported. The jLite distribution includes configuration files for GILDA and RGSTEST VOs which can be used as a reference. The locations of configuration files can also be set during the run time via jLite API.

In order to authenticate grid services jLite also needs certificates of grid certification authorities (CA). The location of directory with trusted CA certificates is set via jLite API. The jLite distribution includes CA certificates needed for GILDA and RGSTEST VOs.

To use jLite API in a Java application a developer should add all jars from JLite_HOME/lib directory to the application classpath.

4 Running a Grid Job with jLite

The central class of jLite API is GridSession which represents a grid session with certain user credentials and provides methods for VOMS proxy creation, job submission, etc. Consider a typical scenario of running a grid job via jLite which includes the following steps:

1. Configure and create grid session;
2. Create user proxy;
3. Delegate user proxy to WMProxy server;
4. Load job description;
5. List matched resources;
6. Submit job to Grid;
7. Monitor job status;
8. Download job output.

Step 1 is to set configuration parameters and instantiate a grid session. Grid session is configured by a GridSessionConfig object and instantiated via a GridSessionFactory as exemplified below:

```
GridSessionConfig config = new GridSessionConfig();

// user's private key passphrase
config.setUserKeyPasswd("*****");

// path to CA certificates
config.setCADir("etc/certs/ca");

// paths to VOMS configuration files and certificates
config.setVOMSDir("etc/vomses");
config.setVOMSCertDir("etc/certs/voms");

// path to WMProxy configuration files
config.setWMSDir("etc/wms");

// create Grid session
GridSession session = GridSessionFactory.create(config);
```

Step 2 is to create a VOMS user proxy by specifying VO name and proxy lifetime in seconds as exemplified below:

```
GlobusCredential proxy = session.createProxy("gilda", 43200);
```

By default jLite stores proxy certificate in \$HOME/x509up_u_\$USER file.

Step 3, required before any job submissions can be made, is to delegate proxy credentials to the WMPProxy server by specifying a delegation identifier as follows:

```
session.delegateProxy("myId");
```

The delegation identifier can be an arbitrary string. jLite grid session remembers the specified identifier and automatically uses it for subsequent calls to the WMPProxy server.

Step 4 is to load job description in JDL format from a file. This task can be accomplished, for example, with gLite API from org.glite.jdl package:

```
JobAd jad = new JobAd();
jad.fromFile("test/test.jdl");
String jdl = jad.toString();
```

The optional step 5 is to match available grid resources (computing elements) to requirements specified in the job description. This can be done to assure that required resources exist and provide a list of matching resources to a user as exemplified below:

```
List<String> ces = session.listMatchedCE(jdl);
System.out.println("Matched Computing Elements:");
for (String ce : ces) {
    System.out.println("\t" + ce);
}
```

Step 6 is the job submission which is done by specifying the job description and a path to job input files:

```
String jobId = session.submitJob(jdl, "test/input");
```

Inside the submitJob() method jLite performs all the low-level steps needed in order to submit a job: registers job in WMPProxy server, gets input sandbox destination URI, uploads input directory to sandbox via GridFTP, and finally starts the job. The method returns a job identifier which is used for monitoring job status, downloading job output, canceling job, etc. Current implementation supports only normal gLite jobs.

Step 7 is monitoring job status until the job is done or aborted as exemplified below:

```
String jobState = "";
do {
    Thread.sleep(10000);
    jobState = session.getJobState(jobId);
    System.out.println(jobState);
} while ( !jobState.equals("DONE") &&
        !jobState.equals("ABORTED") );
```

The final step 8 is to download the job output to a specified local directory as follows:

```
if (jobState.equals("DONE")) {
    String outputDir = "test/" + Util.getShortJobId(jobId);
    session.getJobOutput(jobId, outputDir);
}
```

The complete source code of job submission example can be found in JLITE_HOME/src/jlite/examples/Demo.java.

5 jLite CLI

The jLite distribution includes a demo application which is a command line interface (CLI) similar to gLite UI commands. jLite CLI provides the following set of commands: proxy-init, proxy-info, proxy-delegate, job-match, job-submit, job-status, job-output, job-cancel, proxy-destroy. The semantics of these commands maps directly to jLite API methods so the CLI implementation is straightforward. Despite that, jLite CLI is not a mere demo but a real application which can be used as a cross-platform alternative to gLite UI on any Java capable operating system including Windows.

6 Related Work

Simple API for Grid Applications (SAGA) [2] is an ongoing effort to provide a unified API (think MPI) with C++ and Java bindings for popular grid middleware platforms. In comparison to SAGA, jLite focuses on both one middleware platform (gLite) and one programming language (Java) which simplifies API and implementation.

There are several efforts to port gLite UI on platforms different from Scientific Linux among which Windows is the most requested. GILDA VO provides a virtual machine with preinstalled gLite UI [3], while Grid2Win project [4] uses Cygwin to directly compile gLite UI under Windows. Both approaches have their problems: virtualization overhead in the first case and intricate compilation issues in the second. jLite CLI represents an alternative approach which doesn't suffer from these problems. Though it doesn't provide all functionality of gLite UI it is sufficient for many cases and can be further enhanced along with jLite development.

gEclipse [5] is a cross-platform grid user workbench based on Eclipse platform. It provides a feature rich GUI-based alternative to a classic CLI. gEclipse demonstrates that gLite based grid can be accessed with pure Java in a platform-independent way. This inspired development of jLite since gEclipse doesn't provide an API for building grid applications.

7 Conclusion and Future Work

We presented a simple Java API for accessing gLite based grids which is platform-independent and easy to install. The API provides methods with functionality similar to gLite User Interface commands which is sufficient for building many grid applications. A cross-platform command-line interface based on jLite enables grid users to submit jobs from their desktop machines.

While jLite is still in its infancy it is considered for use in several grid applications. We plan to continue jLite development in accordance with real-world application requirements, so we are looking forward for any feedback, bug reports and feature requests. Among the proposed future work are support for different job types, information service queries and data management operations.

References

- [1] jLite. <http://jlite.googlecode.com/>
- [2] Simple API for Grid Applications. <http://saga.cct.lsu.edu/>
- [3] GILDA VM UI. <https://gilda.ct.infn.it/VirtualServices.html>
- [4] Grid2Win. <http://grid2win.gilda-forge.ct.infn.it/>
- [5] gEclipse. <http://www.geclipse.eu/>