Installing GSAC on CentOS 6.3

by James Matykiewicz, UNAVCO
8 Oct 2014

A prototype GSAC was created to make sure there are no problem developing GSAC on CentOS. This first 'repository' application is coconetgsac.war

For this application, I followed updated instruction in both README files and from additional notes on the UNAVCO GSAC website In general, everything was OK, except I had one issue with the generated code, identified in the note#6 here:

GSAC-60 - Authenticate to see issue details

I used a populated example DB and the service runs in Tomcat here: http://coconetab:8080/coconetgsac

The code is stored in a Subversion repository here: https://fac-svn/svn/repos/dataworks/trunk

There were a few issues dealing with GSAC not running properly - this is a major point about generating and configuring an application from templates or prototype code -> the developer really has to pay attention to the detail of getting all parts of the application development correct. In this case, .properties files, build.xml and Java source code. A failure to exactly match text, can lead to random failures.

In addition, this is a remote server, and Jetty is hard coded to localhost, so It cannot be used is a simple Linux setup. Of course, one could go through the trouble of setting up ssh to X11Forward, and install a browser on the host, then use the ssh-X command remotely; but this was a work-around. The correct solution is to set the evn var jetty.host=hostname. The trick here is 'what process?'. Currently, I have added code the the core GSL code org.gsac.gsl.GsacServer.java. However, this should be parameterized in the gsac.properties file and parsed upon start-up.

Here are some notes while creating a new GSAC code base via prototype development on CentOS:

1) make sure to have Oracle Java installed, default is OpenJDK (even though we are not having any problems so far)
2) Follow the READMEs (1&2) very closely, small mistakes between matching text in xml, properties and java code can lead to obscure failures.
3) always build the main GSL code from the root directory using $>ant or with a specific target. You may have to edit and rebuild this core code (gsacws.jar) several times if you are updating or debugging - just make sure to rebuild from the root, so the specific Dataworks GSAC will use this latest version
4) you must cd to the {root}/src/org/{dataworks}/gsac directory to run the proper ant build.xml in order to compile the Dataworks GSAC. Edit this build.xml if you want new targets, tasks, parameters, etc.
5) To run the Dataworks GSAC with the embedded Jetty:
5.1) modify the local build.xml 'runserver' target and change the default port to 9090, or 9010, just not 8080
5.2) For a remote build and test edit the core org.gsac.gsl.GsacServer.java code, and in the constructor add the lines to set the property 'jetty.host', where the local hostname here is coconetab:

```
String host = "coconetab";
System.setProperty("jetty.host", host);
```

and modify the following line to use the parameter 'host':

```
System.out.println(
 "Running stand-alone GSAC server at: http://"+host+":" + port
+ gsacServlet.getRepository().getUrlBase());
```

Now you can connect your personal machine's browser to http://cococnet:9090/dataworksgasc (or what ever your hostname, port and applications' base url is set to in the configurations.

6) Bug: in my early attempt, I was not getting the XXXDatabaseManager object to instantiate, so I had to modify the XXXRepository object to force the doMakeDatabaseManager()

7) Bug: in the next application being developed, there was an error in the {root}/src/org/dataworks/gsac/resource/gsac.properties, where the 'baseurl' property was not being set correctly. Symptoms are:

7.1) a request to the base application(http://coconetab:8080/dataworksgsac) would return a Tomcat 404 error

7.2) even if you know a webservice url (one with correct parameters) you might get a result back, but not be able to navigate any additional pages.

Installing GSAC on CentOS 6.3

by Susanna Gross, UNAVCO
23 May and 2 June 2014

I got GSAC built and running on CentOS 6.3 with JDK 1.7. I had to install some extra packages, alter the GSAC build scripts, and install apache-ant with extensions that appear to be missing in CentOS.

CentOS comes with ant, but evidently it lacks the javascript extensions necessary to build the repository.

First, quite a few packages are required for installation. On the test system the following packages were added. It is not clear that all of these are necessary. For missing packages, the command is "yum install package_name".

```
sudo yum install svn
sudo yum install ant
sudo yum install ant-scripts
sudo yum install ant-apache
sudo yum install ant-apache-bcel
sudo yum -y install ant-apache-bsf
sudo yum -y install ant-apache-log4j
sudo yum -y install ant-apache-oro
sudo yum -y install ant-apache-regexp
sudo yum -y install ant-apache-resolver
sudo yum install bsf
sudo yum install bsf-javadoc
sudo yum install rhino-javadoc  (actually this is for javascript support, not
javadoc!)

sudo yum -y install ant-javadoc ant-contrib-javadoc antlr-javadoc objectweb-
anttask-javadoc

sudo yum -y install ant-commons-logging

sudo yum -y install ant-antlr ant-commons-net ant-contrib ant-javamail ant-jdepend
ant-jmf ant-jsch ant-junit ant-manual ant-swing ant-trax

sudo yum -y install jss-javadoc jss

sudo yum install java-1.7.0-openjdk java-1.7.0-openjdk-demo java-1.7.0-openjdk-
```

```
devel java-1.7.0-openjdk-javadoc java-1.7.0-openjdk-src

sudo yum install javax
```

Next, here is a summary of the GSAC installation commands and related commands I ran.

Working through the default GSAC installation commands:

```
mkdir GSAC
cd GSAC
svn export svn://svn.code.sf.net/p/gsac/code/trunk gsac-code
cd ~/GSAC/gsac-code/
```

This is a csh script to modify GSAC build scripts to work with JDK 1.7 on CentOS:

```
#!/bin/csh
foreach buildfile (src/*/*/*/build.xml src/gov/nasa/cddis/gsac/build.xml)
   sed 's/target="1.5"/target="1.7"/' $buildfile > a.t
   mv a.t $buildfile
end
```

Now ant (the ant as supplied from CentOS) can use the build.xml files:

```
ant
```

Going on, restoring modified config files:

```
cp ~/repositorymap.js src/org/gsac/gsl/htdocs/repositorymap.js
ant
cd  src/org/gsac/
cp ~/awstest.properties  template/
```

Now (to use the javascript in the build.xml files), you need to use this for ant, /usr/local/apache-ant, for extensions that appear to be missing in the CentOS default ant.

```
setenv ANT_HOME /usr/local/apache-ant-1.9.4/
```

Trying the next ant step in GSAC installation:

```
ant -propertyfile template/awstest.properties makerepository
```

(Without /usr/local/apache-ant, this is my error:

BUILD FAILED
java.lang.ClassNotFoundException:
org.apache.bsf.engines.javascript.JavaScriptEngine)

Going on,

```
cd ~/GSAC/gsac-code/src/org/roipac/gsac
```

Adding DB username and password, which were selected in mysql setup (as in README):

```
vi dbresources/gsacdb_test.properties
```

```
ant
```

And MySQL needs to be set up at this point (see below)

```
ant tables
```

And adding ".database" to the end of the third line (as in README):

```
vi Tables.java
mkdir database
mv Tables.java  database
ant
ant runserver
```

Setting up the mysql database and user:

```
yum install mysql
sudo /etc/init.d/mysqld start
sudo -u mysql mysql -u root

mysql> source /home/sjg/Prototype_GSAC_Database_with_data.sql
mysql> CREATE USER gsac_user@localhost IDENTIFIED BY 'your_password_here';
mysql> GRANT ALL PRIVILEGES ON Prototype_GSAC_Database_with_data.* to
gsac_user@localhost;
```

Setting up apache-ant, I followed instructions at http://ant.apache.org/manual/index.html :

```
cd

wget http://www.gaidso.com/apache//ant/binaries/apache-ant-1.9.4-bin.tar.gz
```

(or wget the latest version)

```
cd /usr/local
sudo tar -xzvf ~/apache-ant-1.9.4-bin.tar.gz
setenv ANT_HOME /usr/local/apache-ant-1.9.4/
cd apache-ant-1.9.4
sudo ant -f fetch.xml -Ddest=system
```