Open Source Military GPS How-To

4.1 The Project How-To

4.1.1 Getting Java:

The only requirements to run Tomcat is of Java Development Kit (JDK).

Installing Java SDK :

We chose to install Sun's Java 2 Platform, Standard Edition, which can be downloaded from http://java.sun.com/j2se/. we have downloaded the *java_ee_sdk-5_0-beta-linux* self-extracting binary file.

Change to the directory where you downloaded the SDK and make the self-extracting binary executable:

```
chmod  +x  java_ee_sdk-5_0-beta-linux.bin
```

Run the self-extracting binary:

```
./ java_ee_sdk-5_0-beta-linux.bin
```

There should now be a directory called java_ee_sdk-5_0-beta-linux in the download directory. Move the SDK directory to where you want it to be installed. we chose to install it in /opt/java. Create /opt/java if it doesn't exist. Here is the command we used from inside the download directory:

```
mv java_ee_sdk-5_0-beta-linux /opt/java
```

Set the JAVA_HOME environment variable, by modifying /etc/profile so it includes the following:

```
export JAVA_HOME=/opt/java/java_ee_sdk-5_0-beta-linux
```

/etc/profile is run at startup and when a user logs into the system, so you will need to log out and log back in for JAVA_HOME to be defined.

Check to make sure JAVA_HOME is defined correctly using the command below. You should see the path to your Java SDK.

```
echo $JAVA_HOME
```

4.1.2 Installing Tomcat :

Download the latest release binary build from http://www.apache.org/dist/jakarta/tomcat5/.

We have downloaded the apache-tomcat-5.5.16.tar.gz file. To unzip this use the following command.

```
tar xvzf  apache-tomcat-5.5.16.tar.gz
```

This will create a directory called jakarta-tomcat-5.5.16. Move this directory to wherever you would like to install Tomcat. we choose /opt. Here is the command I issued from inside the download directory:

```
mv apache-tomcat-5.5.16.tar.gz /opt
```

The directory where Tomcat is installed is referred to as CATALINA_HOME in the Tomcat documentation. In this installation CATALINA_HOME=/opt/apache-tomcat-5.5.16.Thn create a symbolic link to that directory.

```
ln  -s  apache-tomcat-5.5.16 tomcat
```

In order to start and shutdown tomcat you need to add an environment variable pointing to tomcat directory- CATALINA_HOME.

Tomcat is installed in /opt/ apache-tomcat-5.5.16 and linked as tomcat.
Insert the following line in /etc/profile.

```
export CATALINA_HOME=/opt/tomcat
export JAVA_HOME=/opt/jdk.1.4.7/bin
```

Now save the file and reboot the system to ensure that all changes take effect.

Starting up and Shutting down Tomcat :

Before move further we have to ensure that CATALINA_HOME and JAVA_HOME are correctly set. To do this open a terminal and type the following.

```
echo  $JAVA_HOME
echo  $CATALINA  HOME
```

If u get a blank line, or if the directory points anywhere besides where it is supposed to , you will have to correct these environment variable, before continuing.

If everything fine, you can start your tomcat with the following command as root.

```
$CATALINA_HOME/bin/startup.sh
```

To check if tomcat is running fine, open a browser and point the URL to http://localhost:8080, you would see the default welcome page of tomcat.

To stop the tomcat , as root , type the command as indicated below

```
$CATALINA_HOME/bin/shutdown.sh
```

If tomcat does not start, the cause is due to permissions.Ensure that the following files are executable inside the $CATALINA_HOME /bin directory.

```
Chmod +x startup.sh
Chmod +x shutdown.sh
Chmod +x tomcat.sh
```

After you have made the files executable try starting & stopping Tomcat again.

4.1.3 Installing MySQL on Linux

To install the mysql you need two files .you can download it from http://www.mysql.com/downloads

MySQL-client-standard-5.0.18-0.i386

MySQL-server-standard-5.0.18-0.i386

If u want to install directly from internet use the following command(yum utility)

```
yum  install  MySql
```

Creating an administrative password :

We need to create one administrative user account so that we can manage the mysql database server. Here we are going to add root user as an administrator account

First, use the mysql program to connect to the server as the MySQL root user:

```
mysql --user=root mysql
```

If you have assigned a password to the root account, you'll also need to supply a --password or -p option for this mysql command and also for those later in this section.

After connecting to the server as root, you can add new accounts. The following statements use GRANT to set up ‘miladmin’ account:

```
mysql> GRANT ALL PRIVILEGES ON *.* TO 'miladmin'@'localhost' IDENTIFIED
BY 'authorize' WITH GRANT OPTION;


mysql> GRANT ALL PRIVILEGES ON *.* TO 'miladmin'@'%' IDENTIFIED BY
'authorize' WITH GRANT OPTION;


mysql> GRANT RELOAD,PROCESS ON *.* TO 'miladmin'@'localhost';
```

Starting the MySql services

To start the service of the MySql server, issue the following command from the terminal window.

```
 /etc/init.d/mysqld start
or
service mysqld start
```

Creating the database

To create tables we need to first create database so issue the following command.

```
create database HCD;
```

Displaying the databases

You can now see the name of all the existing databases, use the following SQL command.

```
Show databases;
```

This shows three databases *mysql* and *test* and *HCD* database has been created by us.

Mysql and test are the two initial databases. The *mysql* database contains all the information about database access. The *test* database is provided for easy initial testing. It can be accessed with any user name and empty password.

Changing the databases

To work with the database we have to make it the current database. By default, current database is ‘mysql’ .Apply the following SQL statement to use our database.

```
use HCD;
```

‘Database changed’ message will appear on your screen, this means that your current database is HCD.

Creating a table

Type the following command to create a table named as *clearance*.

```
mysql>create table clearance
        - >(
        - >AId int not null primary key,
        - > Code int,
        - > Fname char(10),
        - >Lname  char(10),
        - > R int,
        - > W int,
        - > X int
        - >);
```

4.1.4 Configure the JNDI(Java Naming & directory Interface) DataSource

Here we want to create a connection pool hence we are using JNDI for getting the resources available to the Base Server .The type="javax.sql.DataSource specifies that we are going for connection pooling.Connection pooling makes the server available with pool of few databases connection and hence provides the clients with faster access .Latency is mitigated by use of connection pooling and since our application in real time hence reqires fast execution.

To enable connection pooling in Tomcat replace the context.xml($CATALINA_HOME/conf/context.xml) file with the file provided below:

```
<Context path="/HCD" docBase="HCD"
        debug="5" reloadable="true" crossContext="true">

  <Resource name="jdbc/HCD" auth="Container" type="javax.sql.DataSource"
               maxActive="100" maxIdle="30" maxWait="10000"
               username="miladmin" password="authorize"
driverClassName="com.mysql.jdbc.Driver"
               url="jdbc:mysql://localhost:443/miladmin?autoReconnect=tr
ue"/>

</Context>
```

Now creating a WEB-INF/web.xml for this(Militarygps) application.

```
<web-app xmlns="http://java.sun.com/xml/ns/j2ee"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="http://java.sun.com/xml/ns/j2ee
http://java.sun.com/xml/ns/j2ee/web-app_2_4.xsd"
    version="2.4">
  <description>Military Gps Appication</description>
  <resource-ref>
      <description>Highly Confidential Database(HCD)
Connection</description>
      <res-ref-name>jdbc/HCD</res-ref-name>
      <res-type>javax.sql.DataSource</res-type>
      <res-auth>Container</res-auth>
  </resource-ref>
</web-app>
```

4.1.5 Building OpenSSL on Linux for Military Gps application

1. Download the latest OpenSSL distribution from http://www.openssl.org/source/.

Unpack the distribution

```
tar zxvf openssl-0.9.7e.tar.gz
```

2. Configure OpenSSL

```
cd openssl-0.9.7e
./config -fPIC --prefix=/usr --openssldir=/usr/openssl
```

If you omit the --prefix and --openssldir parameters, openssl will install into /usr/local/openssl.

3. Compile OpenSSL

```
make
make test
```

The "make test" step is optional, but useful to make sure all works as it is supposed to.

4. Install OpenSSL

```
make install
```

4.1.6 Configuring Tomcat 5 for SSL (with OpenSSL)

1. Generate an RSA key for signing the certificate:

```
openssl genrsa -out MilitaryGpsKey.pem 2048
```

2. Generate a certificate using the new key:

```
openssl req -new -x509 -key MilitaryGpsKey.pem -out
MilitaryGpsCert.pem -days 365
```

Enter your name, organization name and address as prompted.

In this example, we've created a key file, MilitaryGpsKey.pem, and a self-signed certificate. Normally, you want a certificate from a "certificate authority" or CA. Using a self-signed certificate is not for production.

3. Since the certificate is in PEM format, convert it to PKCS12 for Tomcat:

```
openssl pkcs12 -export -in MilitaryGpsCert.pem -inkey
MilitaryGpsKey.pem -out MilitaryGpsCert.p12 -name tomcat
```

You MUST specify an export password! Tomcat expects one. We have used “authorize” as the password.

4. Edit $CATALINA_HOME/conf/server.xml and undefine the SSL connector and comment the connector for port 8080 :

```
   <Connector port="8443"
            maxThreads="150" minSpareThreads="25"
   maxSpareThreads="75"
            enableLookups="false"
   disableUploadTimeout="true"
            acceptCount="100" debug="0" scheme="https"
   secure="true"
            clientAuth="false" sslProtocol="TLS"
   keystoreType="PKCS12"
            keystoreFile="/opt/openssl/MilitaryGpsCert.p12"
   keystorePass="authorize"/>
```

4.1.7 JAVA RMI :

Remote Method Invocation is used to make the client as a thin client The files that are required to access the RMI procedures are :

MilitaryGpsImpl

MilitaryGpsIntf

MilitaryGpsServer

MilitaryGpsClient

The first three files in our case /home/miladmin/MilitaryGpsBase.

The fourth file(MilitaryGpsClient) located in \opt\apache-tomcat-5.5.16\webapps\MilitaryGps\WEB-INF\classes

(The above directories are specific to our case)

MilitaryGpsImpl

This file contains the implementation of the RMI procedure .There are two procedures :

a) Information :-- Information procedure will return the satellite map imaginary as per the current co-ordinates(long & lat) of the remote user

b)Authorize :--this procedure will authenticate the remote user by checking the authorization id & code(password) from the HCD database & will return a flag specifying acceptance or denial of GPS access.

After compiling the MIlitaryGpsImpl.java we need to run the rmic with MIlitaryGpsImpl.class as the parameter.s

```
rmic MIlitaryGpsImpl
```

After running above statement two files will be generated named as MilitaryGpsImpl_Skel.class & MilitaryGpsImpl_Stub.class in the present working directory.

To run the server we required MilitaryGpsImpl_Skel.class to be present on the server side & the client will be needing MilitaryGpsImpl_Stub.class .

MilitaryGpsIntf

Interface contains the name of the remote procedure & the arguments that will be passed to them.the file looks like this.

```
//interface for the Military client access
import java.rmi.*;
public interface MilitaryGpsIntf extends Remote {
String information(int lati,int longi) throws RemoteException;
int authorize(int Aid,int Code) throws RemoteException;
}
```

MilitaryGpsServer

This is the base server which will serves the remote user.this can be executed using :

```
Java  MilitaryGpsServer
```

Note : before running the above command you have to start the rmiregistry using :

```
rmiregistry &
```

MilitaryGpsClient

The client is actually a servlet which is accessing the MilitaryGps server & invoking the remote procedures using Interface for the remote procedures.

4.1.8 Configuring the jdk for elliptic curve cryptography (ECC)support:

1. Download the latest mustang version of JAVA available at java.net community .

2. Get the cryptographic jar file from bouncycstle and copy these files to the extracted jdk.

3. To enable ecc suport in jdk we have to get the jurisdiction policy from sun.com After extracting the policy copy these files to:

$JAVA_HOME/jre/lib/security

4. Open the java.security file from($JAVA_HOME/jre/lib/security) and append a line in the file stating the bouncycastle cryptographic provider.The file will look like this after appending the following line:

```
security.provider.9=org.bouncycastle.jce.provider.BouncyCastleProvider
```

4.1.9 Bastion Host

We have configured a firewall (firestarter) to have a policy mechanism working. There are two kind of policy which we have a configured :

a. Inbound policy

b. Outbound policy

Inbound policy deals with the connection that are being made by other hosts to bastion host.

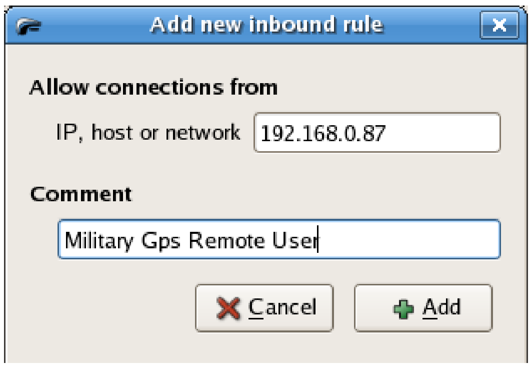
Where as Outbound policy describes the mechanism for LAN clients to the outside.

Military-Gps Bastion Host policy :

note : the ips stated here are specific in our case and can be changed accordingly.

Inbound policies for MilitaryGps

1.At first we have to define the policy for MilitaryGps Remote Users in which we allow those ips who want to connect to the Military Base.(like in our case we set the remote user ip as 192.168.0.87)


Add new inbound rule

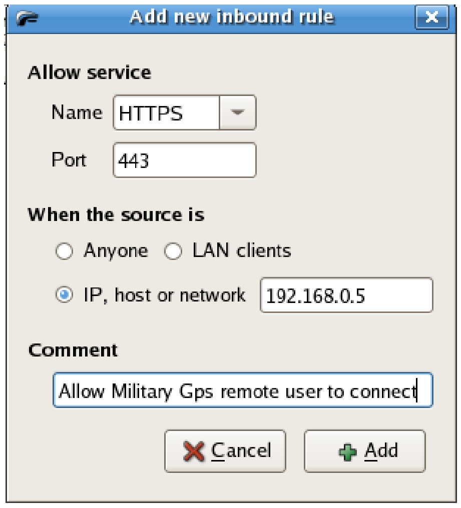
Allow connections from

IP, host or network: 192.168.0.87

Comment: Military Gps Remote User

Snapshot 4.1.9 a

2.Secondly we allow Military Gps remote users to access https service at port 443(Default) to connect to the Base server


Add new inbound rule

Allow service

Name: HTTPS

Port: 443

When the source is

○ Anyone ○ LAN clients

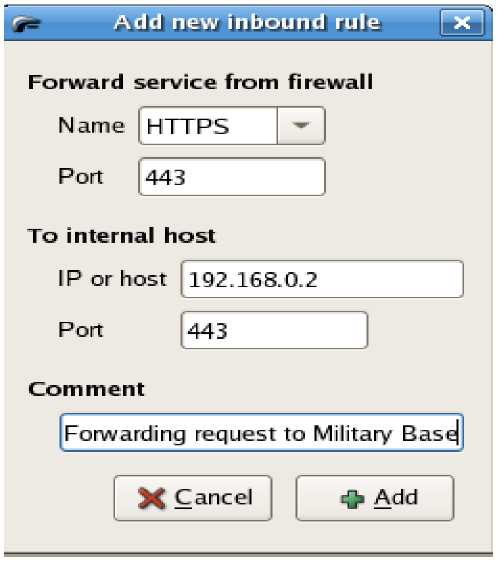
● IP, host or network: 192.168.0.5

Comment

Allow Military Gps remote user to connect

Snapshot 4.1.9 b

3.Finally the bastion host redirects the request to the base server once it verifies the integrity of the remote user.


Add new inbound rule

Forward service from firewall

Name: HTTPS

Port: 443

To internal host

IP or host: 192.168.0.2

Port: 443

Comment

Forwarding request to Military Base

Snapshot 4.1.9 c

Out Bound Policy

The bastion host is not allowed to make external connections with the outside world. This will enforce complete security to the base server since any connection to the outside world will be denied with the firewall.

Check the radio button which says *Restrictive by default* and remove all the default allowed services.