Source Code Compile Instructions

The source code will not compile as-is, since this application is meant to be used remotely if desired, and also because the web service itself must be built on the machine where it will be hosted.

Software Required:

- PostgreSQL
- Java SDK with Netbeans
- Java
- GlassFish

This project was created in NetBeans 8.1 using its built-in GUI builder for Swing applications. As a result, much of the code is auto-generated by the IDE and may not run properly in a different environment. Because this was built using NetBeans, any computer that can run NetBeans can run this software. The following hardware requirements are listed for NetBeans:

Minimum Hardware Configurations

- Microsoft Windows Vista SP1/Windows 7 Professional:
- Processor: 800MHz Intel Pentium III or equivalent
- Memory: 512 MB
- Disk space: 750 MB of free disk space
- Ubuntu 9.10:
- Processor: 800MHz Intel Pentium III or equivalent
- Memory: 512 MB
- Disk space: 650 MB of free disk space
- Macintosh OS X 10.7 Intel:
- Processor: Dual-Core Intel
- Memory: 2 GB
- Disk space: 650 MB of free disk space

Recommended Hardware Configurations

- Microsoft Windows 7 Professional/Windows 8/Windows 8.1:
- Processor: Intel Core i5 or equivalent
- Memory: 2 GB (32-bit), 4 GB (64-bit)
- Disk space: 1.5 GB of free disk space
- Ubuntu 15.04:
- Processor: Intel Core i5 or equivalent
- Memory: 2 GB (32-bit), 4 GB (64-bit)
- Disk space: 1.5 GB of free disk space

- OS X 10.10 Intel:
- Processor: Dual-Core Intel
- Memory: 4 GB
- Disk space: 1.5 GB of free disk space

Obtain Your Machine's IP Address

Open Command Prompt and type "ipconfig /all" on the host machine. Write down or copy to clipboard the **IPv4 Address** for later use ***if you are intending on hosting the web service on a different machine than the client.***

Program Installation

Before any of this is built, make sure to download and install:

- NetBeans & JDK
- GlassFish
- PostgreSQL
- Java

NetBeans Setup

NetBeans can be downloaded straight from Oracle at http://www.oracle.com/technetwork/articles/javase/jdk-netbeans-jsp-142931.html. This link will take you to a version that includes the JDK. If you download NetBeans without the JDK, keep in mind you will still need to download the JDK separately for any Java development. It also may be necessary to download the Plug-in to enable Java Web. To do this, go to Tools -> Plugins -> Available Plugins -> Search and look up "Java EE Base Plugin".

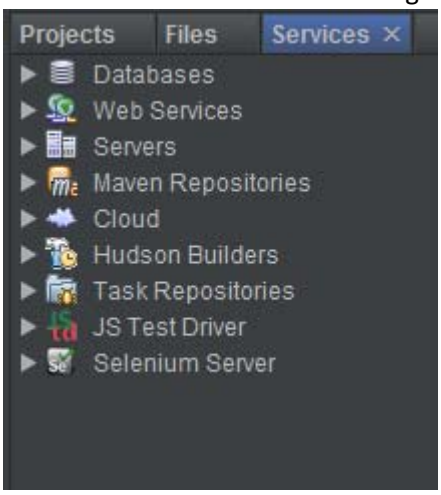
Java Setup

Navigate directly to https://java.com/en/download/ to download the latest version of Java from Oracle. This is also known as Java Runtime Environment and is necessary for launching Java executables.

GlassFish Setup

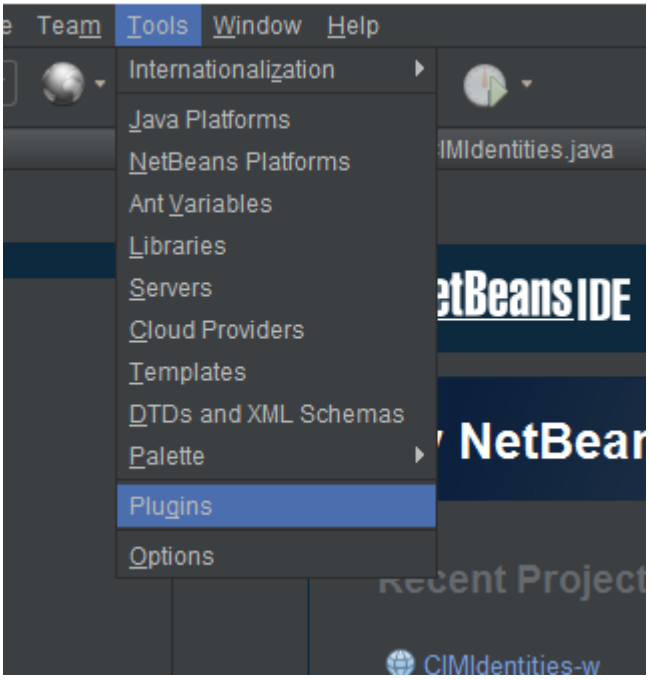
GlassFish is the server that will be used to host the webservice. It can be downloaded from Oracle, however the version on their website is not the most up-to-date version, so it's best to download it directly via NetBeans. To do so, follow these steps:

1. Select the “Server” tab in the navigator on the left side of the screen.



2. Right click “Servers”.
3. Select “add server”.
4. Choose GlassFish.
5. Check the box for the license agreement.
6. Select the latest version (version 4.1.1 or later as of present time).

Since GlassFish is being installed through NetBeans, the following plugins will need to be installed. Select Plugins from Tools:



Select “Available Plugins” and search for these 3 Plugins:

1. Jave EE Base
2. Oracle Cloud

3. EJB and EAR

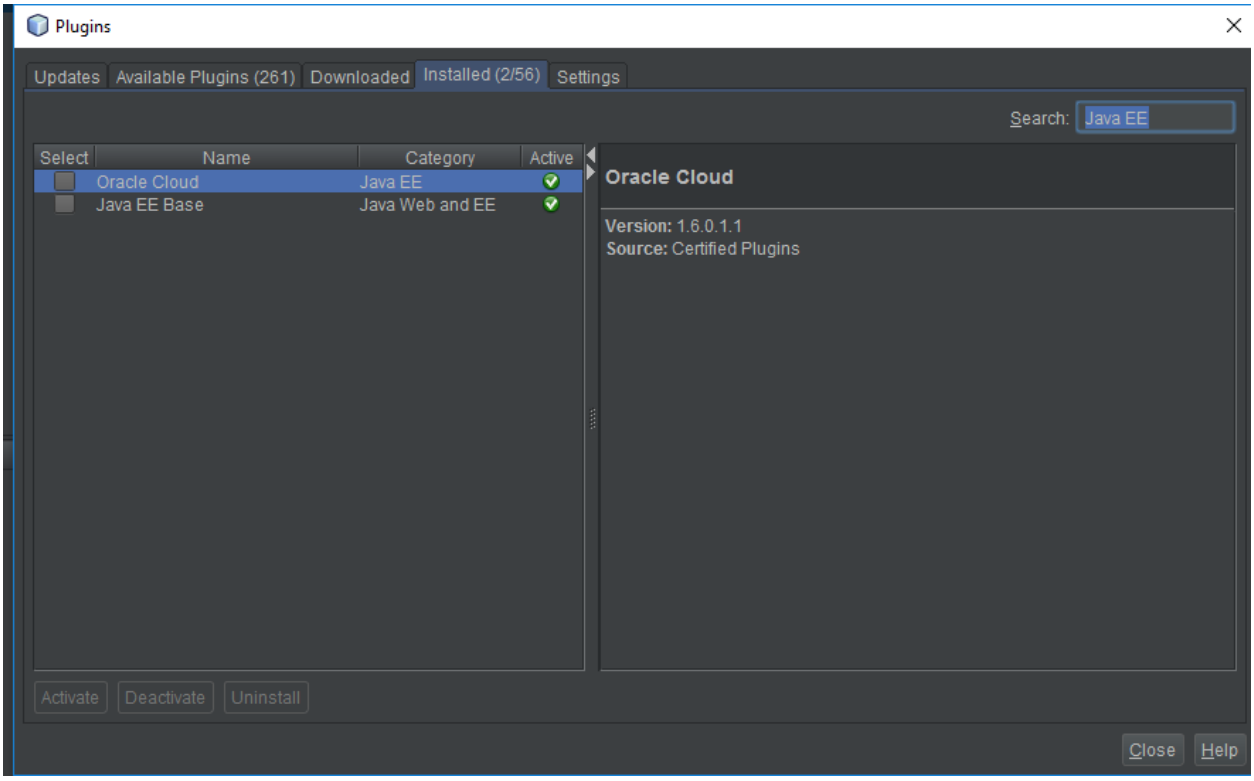


Figure 1: An example plugin menu.

PostgreSQL setup

If the installer on PostgreSQL isn't working (usually due to antivirus/Firewall settings), it may be necessary to download the binary zip file and set it up via Command Line. To do so, follow these instructions:

First create "log" and "data" folders in "pgsql" folder.

Note [POSTGRESQL_ROOT] is the full path to the pgsql folder when in command prompt. For example, if you saved Postgres in a folder called "psql" located directly on the C Drive, then you would replace [POSTGRESQL_ROOT] with "C:\psql".

1. Navigate to where the pgsql\bin folder is located. Note that the folder's name prior to "bin" could be different depending on whether you renamed the files as you saved them.
2. Run the following command:

   initdb -U postgres -A password -E utf8 -W -D [POSTGRESQL_ROOT]\data

   Note: "password" above means you are using password authentication. Do not type your actual intended password for the database here, as you will be prompted for that later.
3. You can start and stop the server by running (this is all on one line)

[POSTGRESQL_ROOT]/bin/pg_ctl -D [POSTGRESQL_ROOT]/data -l [POSTGRESQL_ROOT]/log/pgsql.log start

Or stop it by running (also all on one line):

[POSTGRESQL_ROOT]/bin/pg_ctl -D [POSTGRESQL_ROOT]/data -l [POSTGRESQL_ROOT]/log/pgsql.log stop

Note: If already in the binary folder when running this command, simply start the command with pg_ctl. Putting the entire file path is only necessary if you're currently in a different directory from the executable being used. The full path to the log folder must still be entered though.

After the server is up and running, modify the following files so allow outside connections to the database.

Configuring PostgreSQL for Remote Access

In your PostgreSQL folder, after the database server has been started, navigate to the "data" folder and open pg_hba.conf using Notepad. Scroll down the following section and make sure it matches what's shown below.

```
# IPv4 local connections:
host    all         all         127.0.0.1/32          md5
# IPv6 local connections:
host    all         all         ::1/128               md5
```

Make sure your pg_hba.conf configurations match what is shown above. Typically the command-line installation has slightly different settings. The example below is simply to show how it could be set up for a specific IP Address.

```
host    all         all         144.58.246.143/24     md5
```

Next, change the listen address in **postgresql.conf**, also located in the "data" folder. By default this is typically localhost. To get this to listen to all addresses, change the value for listen_addresses to '*'. Otherwise, list only desired addresses.

Note: It is highly recommended *not* to set listen_addresses to '*' if connections are known in advance (for example, if only work connections are allowed, it can be set up to only listen to IP addresses that follow a specific subset of IP addresses).

Setting up the CIMIdentity Database

1. After all 3rd party software is installed, first run **CreateTable.jar** (You can simply double click the icon for it on Windows Explorer) to set up the database tables. If an error message regarding a driver message pops up, navigate to CreateTable\dist\lib and launch the executable *postgresql9.2-1002.jdbc4* first.

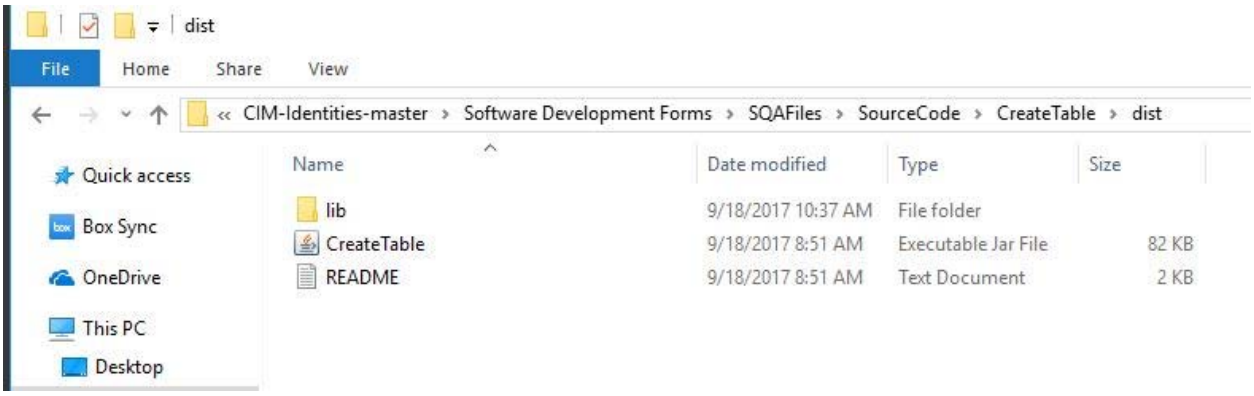


Figure 2: Java executables can be launched by double-clicking their icon in file explorer. Run the executable in the "lib" folder if a driver error occurs.

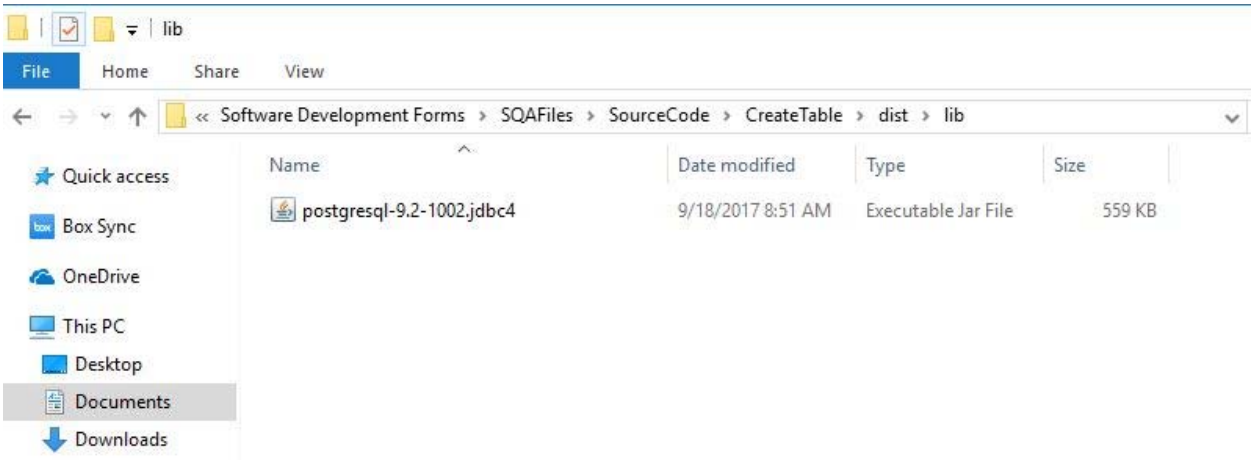


Figure 3: Launch this driver executable should a driver error message pop up. There is no confirmation message that appears after launching it.

2. Once that application has successfully run, set up the CIMIdentities Web Application as outlined below.
3. Once the Web Application is set up, either CIMIdentitiesClient or CIMIdentitiesJSP can be set up next. For the purpose of these instructions, CIMIdentitiesJSP is explained first. These are separate clients for interacting with the web service (Desktop application and web browser client respectively).

Launching the Web Service via NetBeans

The only lines of code in the web service that will need to be changed are those for the password to the database. First, however, the project must be opened in NetBeans. Navigate to File -> Open Project.

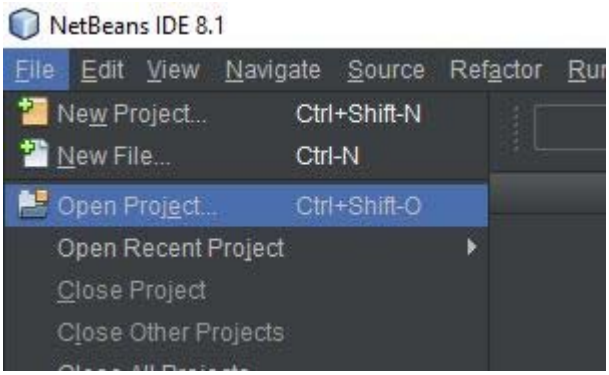


Figure 4: Open the entire project via NetBeans.

Then select the project "CIMIdentities-w"

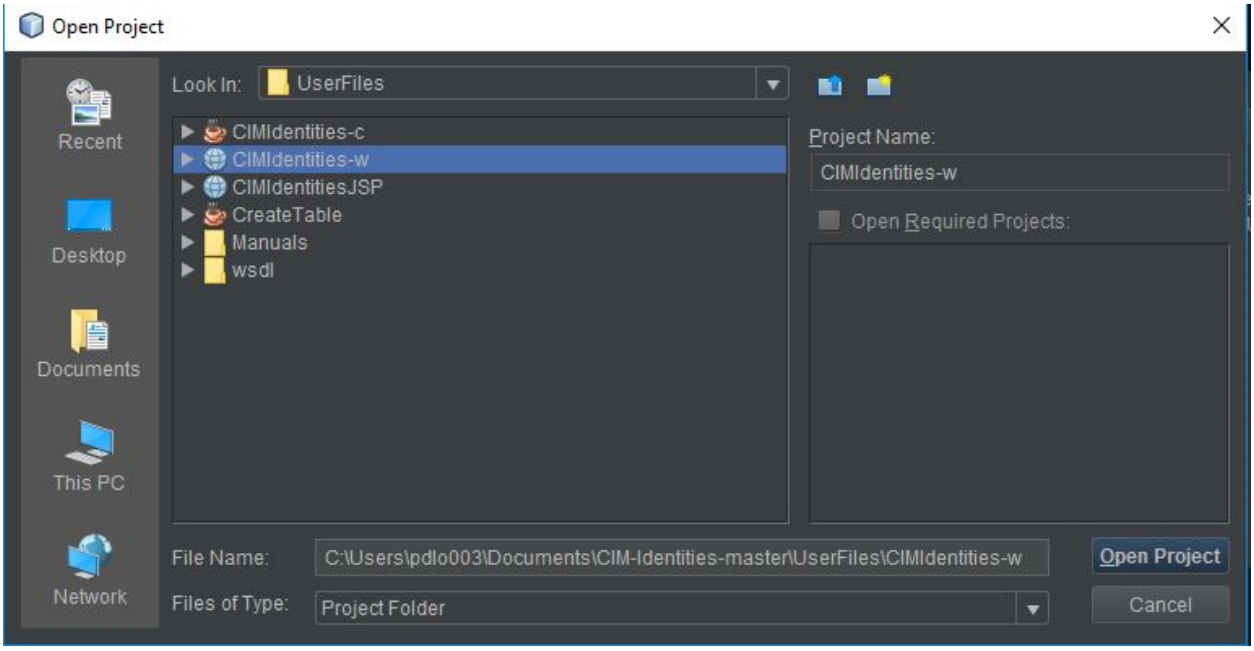


Figure 5: Start with setting up the Web Service.

CIMIdentities-w

- CIMIdentities.java, line 39, fill in the password setup for PostgreSQL inside the quotes
- GetCIMIdentities.java, line 44, fill in the password setup for PostgreSQL inside the quotes

After that's done, select "Clean and Build" for the Project, then right click it in the project navigator tab and selecty "Deploy".

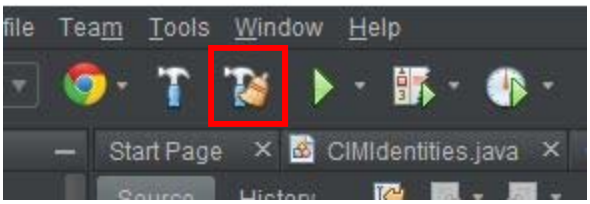


Figure 6: Clean and Build outlined in red above.

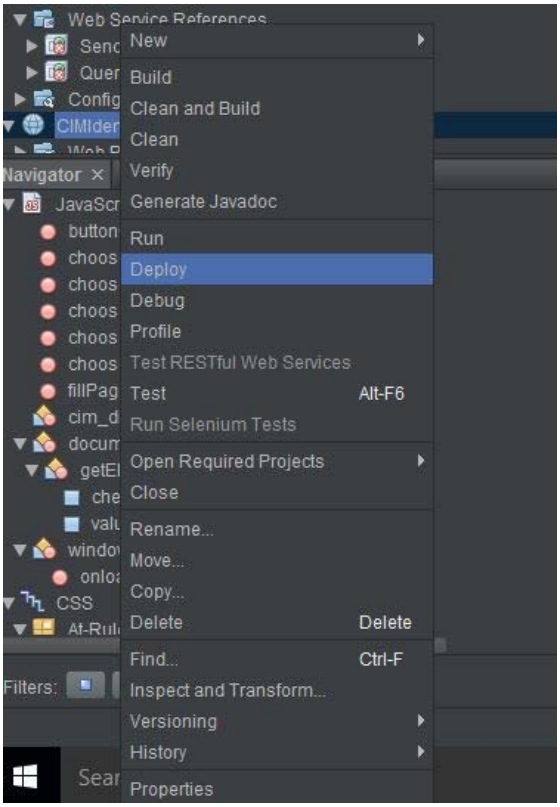


Figure 7: Deploy highlighted above.

CIMIdentities Clients – CIMIdentitiesJSP

First, open the Project by going to File -> Open Project and select CIMIdentitiesJSP.

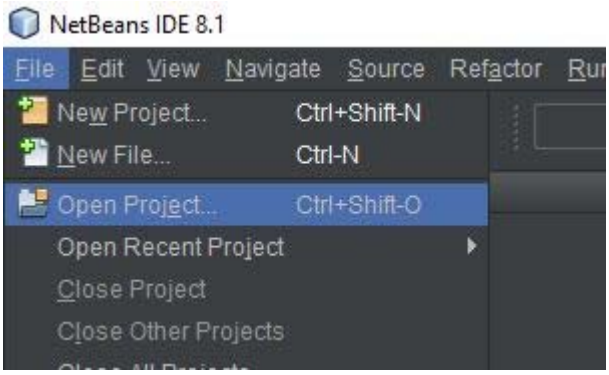


Figure 8: Again, select Open Project.

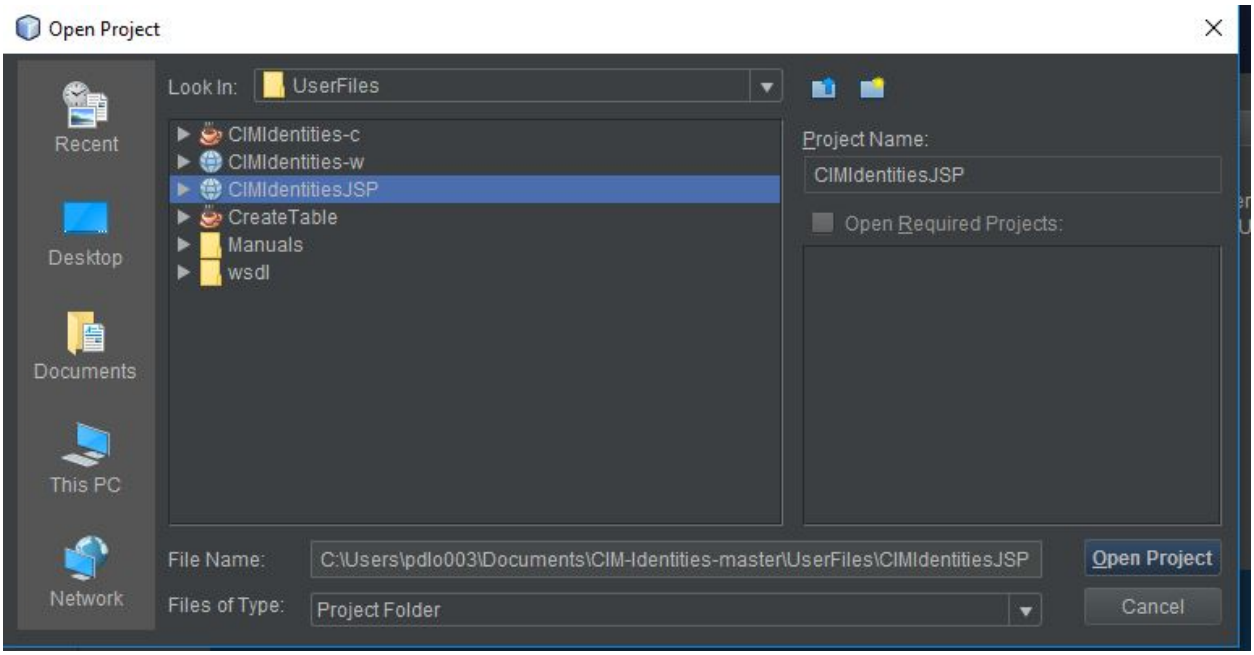


Figure 9: Open the JSP

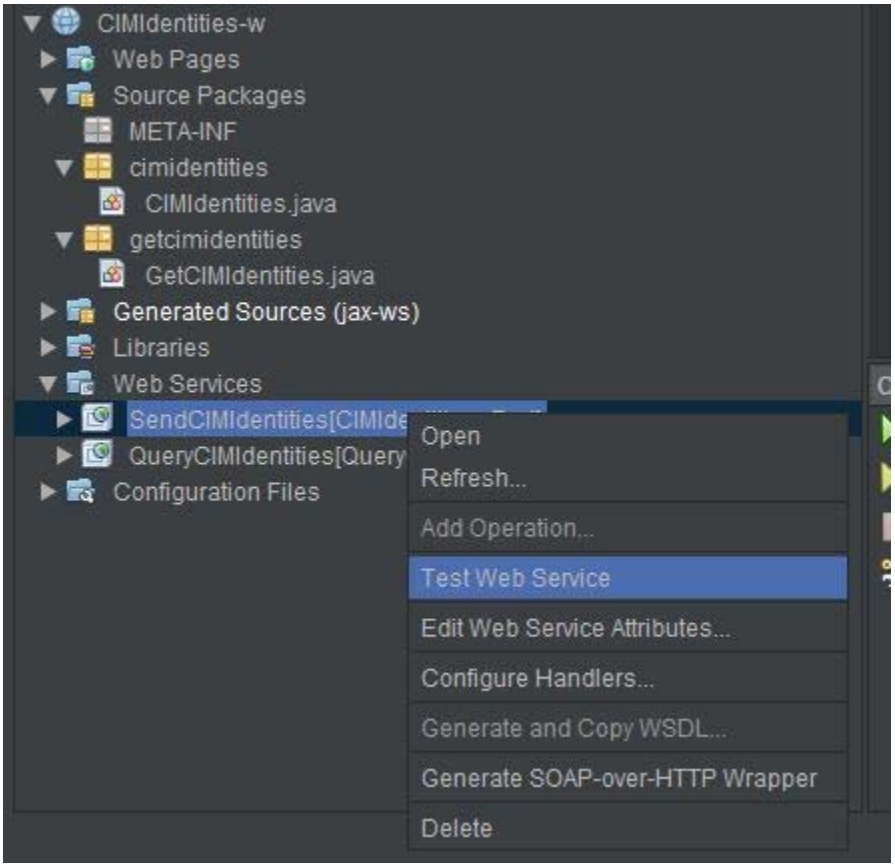
Making Changes to the Client

Note: If you are planning on keeping the host machine separate from the clients, then the clients must point to the host machine's IP address instead of localhost. Be aware that firewall and other security settings may prevent the host machine from being accessible outside of the local network.

The client setup can vary in difficulty depending on which ports localhost uses on your machine. **If the webservice uses port 8080 on your machine, then the clients will match up perfectly and can be run as-is. Otherwise, some changes will need to be made to the clients to make them run.**

Figure 10: An example error message if the Localhost address doesn't match up.

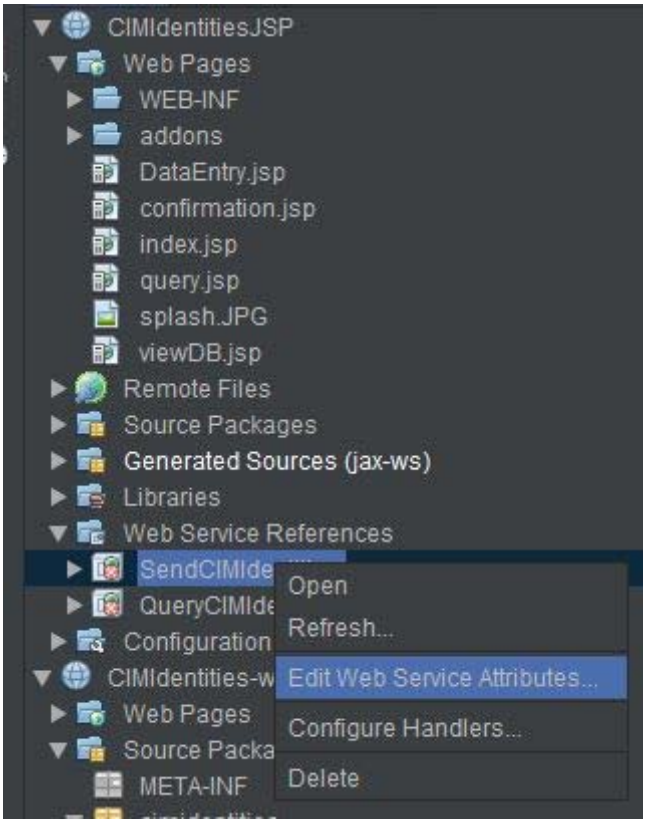
1. First, grab the WSDL address for your web service by right-clicking one of the web services in CIMIdentities-w as shown below.



This will bring up a tester page that contains a link to the WSDL file on the top of the screen. Click that link to obtain the address of the WSDL.

- Copy the WSDL link, which should be something like http://localhost:9090/CIMIdentities-w/SendCIMIdentities?WSDL
- Once the service is deployed, both SendCIMIdentities and QueryCIMIdentities will be deployed, and their WSDL address will follow standard conventions, so in the above example link, swapping the string “SendCIMIdentities” for “QueryCIMIdentities” will point to the other WSDL file.

2. Now that the WSDL address has been obtained, navigate to the “Web Service References” folder of the client, select either SendCIMIdentities or QueryCIMIdentities, and select “Edit Web Service Attributes” as shown below. **Note: If this option isn’t available, simply delete the web service clients and recreate them using the correct WSDL URL. You can do this by right-clicking the project, selecting new → web service client → select the option for WSDL URL → paste the URL for the SendCIMIdentities or QueryCIMIdentities. This must be done for both and you can do it in whichever order you prefer.**



3. Select the tab “Wsimport Options” and change the wsdl location to the address you copied earlier.

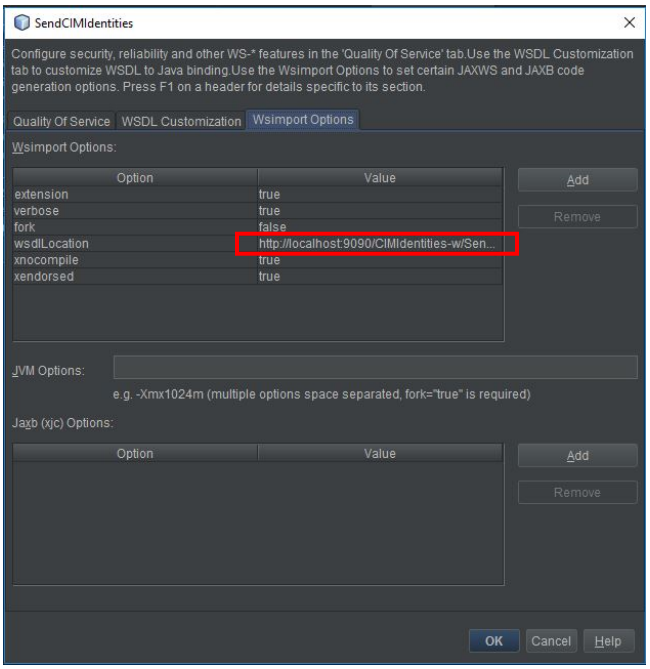


Figure 11: Change the WSDL location to match where your web service is actually being hosted, then select "OK"

4. In addition to changing the port localhost uses here, there are two other files where it will need to be changed: QueryCIMIdentities.wsdl and SendCIMIdentities.wsdl as shown below.

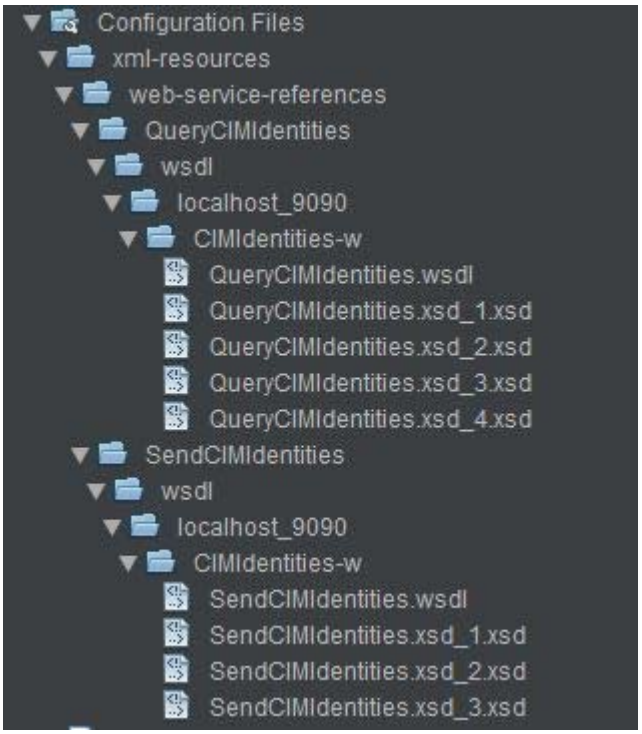


Figure 12: Open the WSDL files here and change every line where localhost's port doesn't match your version of the web service.

If no changes have been made, the following lines have port numbers that will need to be changed.

- QueryCIMIdentities.wsdl: Lines 4, 7, 44
- SendCIMIdentities.wsdl: Lines 3, 4, 120

Note: Be aware that when changing the port, a certificate message will pop up. Click "OK" for all of these.

5. After making these changes, **Clean and Rebuild** the project, then select "Run Project".

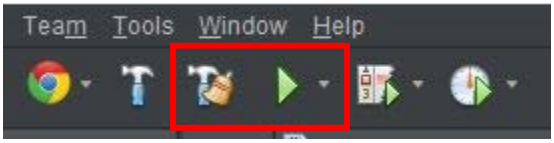


Figure 13: Clean and build first, then select Run Project on the right.

CIMIdentities Clients – CIMIdentitiesClient

For the desktop application, follow the process outlined above for CIMIdentitiesJSP.

For Step 4, look for the WSDL files to be modified under the “Source Packages” folder, shown below.

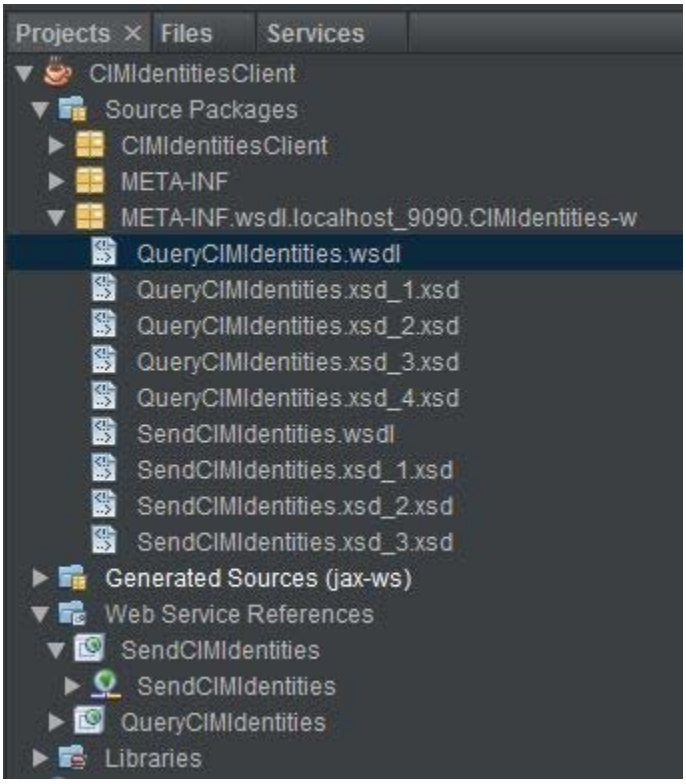


Figure 14: Modify the wsdl files here with the correct port number.

As before, modify the port on the following lines:

- QueryCIMIdentities: 4, 7, 44
- SendCIMIdentities: 3, 4, 120