Iam creating my certificate from scratch exactly as follow (if this example works, il wil use the certificate authority later) :

Enabling client authentication for SSL

To exchange certificates and allow only "trusted" clients to use the Talend Runtime Container HTTP service, you need to follow the following instructions.

1. Enable the HTTP client auth support in the Karaf-based Talend Runtime Container.

   When you install the HTTP feature, the container leverages Pax-Web to provide HTTP OSGi service:

   ```
   karaf@trun> feature:install http
   ```

2. Add a custom etc/org.ops4j.pax.web.cfg file with the following content:

   ```
   org.osgi.service.http.port=8181

   org.osgi.service.http.port.secure=9001
   org.osgi.service.http.secure.enabled=true
   org.ops4j.pax.web.ssl.keystore=./etc/keystores/keystore.jks
   org.ops4j.pax.web.ssl.password=password
   org.ops4j.pax.web.ssl.keypassword=password
   #org.ops4j.pax.web.ssl.clientauthwanted=false
   org.ops4j.pax.web.ssl.clientauthneeded=true
   ```

   The clientauthwanted and clientauthneeded properties are valid for Karaf 2.2.x which uses Pax Web 1.0.x. For more information about the version of Karaf your Talend Runtime Container is based on, see the Talend Installation Guide or the Release Notes.

   Thanks to the clientauthneeded property, the client is "forced" to be trusted.

Create the trusted client certificate
About this task

You are going to use a keytool (provided with the JDK) to manipulate the keys and certificates.

Procedure

1. Create two key pairs:

   - one for the server side (use for SSL),
   - one as an example of the client side (use for "trust", should be performed for each client, on the client side).

   ```
   mkdir -p etc/keystores
   cd etc/keystores
   keytool -genkey -keyalg RSA -validity 365 -alias serverkey -keypass password -storepass password -keystore keystore.jks
   keytool -genkey -keyalg RSA -validity 365 -alias clientkey -keypass password -storepass password -keystore client.jks
   ```

   These key are self-signed. In a production system, you should use a Certificate Authority (CA).
2. Export the client certificate to be imported in the server keystore:

   ```
   keytool -export -rfc -keystore clientKeystore.jks -storepass password -alias clientkey -file client.cer
   keytool -import -trustcacerts -keystore keystore.jdk -storepass password -alias clientkey -file client.cer
   ```

3. Check that the client certificate is trusted in our keystore:

```
keytool -list -v -keystore keystore.jks
...
Alias name: clientkey
Creation date: Dec 12, 2012
Entry type: trustedCertEntry
...
```

4. You can now remove the client.cer certificate.

Start the container and test with the WebConsole
Procedure

1. Start the Talend Runtime Container:
   - bin/trun for Linux
   - bin/trun.bat for Windows
2. Install the WebConsole feature:

   ```
   karaf@trun> feature:install webconsole
   ```

   If you try to access to the WebConsole (using a simple browser) using https://localhost:9001/system/console, you get the following message:

   ```
   An error occurred during a connection to localhost:9001.

   SSL peer cannot verify your certificate.
   (Error code: ssl_error_bad_cert_alert)
   ```

   Which is normal as the browser does not have any trusted certificate.
3. Add the client certificate in the browser.
   Firefox supports the import of PKCS12 keystore. So, you are going to "transform" the JKS keystore into a PKCS12 keystore:

   ```
   keytool -importkeystore -srckeystore clientKeystore.jks -srcstoretype JKS -destkeystore client.pfx -deststoretype PKCS12
   Enter destination keystore password:
   Re-enter new password:
   Enter source keystore password:
   Entry for alias clientkey successfully imported.
   Import command completed: 1 entries successfully imported, 0 entries failed or cancelled
   ```

   Now, you can import the client certificate in Firefox. To do so, in the Tools menu, click the Options entry, and click on the Advanced tab.
   You can go in Certificates tab and click on View Certificates button.
   In the Your Certificates tab, you can click on the Import... button and choose the client.pfx keystore file.
4. If you try to access https://localhost:9001/system/console again, you will have access as a trusted client and use it.

When i import certificate in Firefox for example it shows me Bad certificate every time.

For jetty ssl iam using the settings as follow :

Configuring jetty for SSL

To turn off having pax-web to directly create the connector, change the etc/org.ops4j.pax.web.cfg file as follows:

```
#org.osgi.service.http.port.secure=9001
#org.osgi.service.http.secure.enabled=true
....
org.ops4j.pax.web.config.file=${karaf.base}/etc/jetty.xml
```

In etc/jetty.xml, remove the connector already defined there ("org.eclipse.jetty.server.nio.BlockingChannelConnector") and replace it with the following one:

```
<Call name="addConnector">
    <Arg>
        <New class="org.eclipse.jetty.server.ssl.SslSelectChannelConnector">
            <Set name="port">9001</Set>
            <Set name="maxIdleTime">30000</Set>
            <Set name="keystore">./etc/keystores/keystore.jks</Set>
            <Set name="password">password</Set>
            <Set name="keyPassword">password</Set>
        </New>
    </Arg>
</Call>
```

Those settings puts the connector on port 9001 to use the SslSelectChannelConnector which provides working continuation support.

Regards