**Creating a Signing Profile**

To create a signing profile, we first need a keystore that contains our signing keys. To create a new certificate and import it into a new keystore, execute the following command.

keytool -genkey -alias my-cert -keyalg RSA -ext EKU="codeSigning" -validity 365 -keypass K3yP@ss -storepass St0reP@ss -keystore my\_signing\_profile.jks

And follow the instruction to generate .jks file

Keytool is a standard Java utility. These options can be altered as necessary. Run keytool -help for more information.

Next, we need to create the xml file to store our signing profile settings. This must have the same name as the keystore file, and must be a peer to it, so create a file named my\_signing\_profile.xml with the following contents:

<?xml version="1.0" encoding="UTF-8" standalone="no"?>

<!DOCTYPE properties SYSTEM "http://java.sun.com/dtd/properties.dtd">

<properties>

<comment>Code Signing Properties</comment>

<entry key="niagara.signing.keypass.my-cert">K3yP@ss</entry>

<entry key="niagara.signing.storepass">St0reP@ss</entry>

<entry key="niagara.signing.profileType">com.tridium.signing.RestrictedSigningProfile</entry>

<!-- The lines below will enable timestamping -->

<entry key="niagara.signing.standardtsa">

http://timestamp.digicert.com

</entry>

</properties>

This file stores the key password for the signing key we just generated, and the keystore password for the keystore we created. It also sets the signing profile to the RestrictedSigningProfile. The default signing profile is the LocalSigningProfile, which will automatically regenerate a generic certificate when the existing one gets close to expiration. This is useful if you are using generic self-signed certificates, but since we are using custom certificates, we do not want it to be automatically regenerated.

Copy the keystore files (both jks and xml) into a location your Gradle has access permission to. (e.g.: C:\Users\BGite\.tridium\security would be a good place but this is up to you)

The file names, cert alias, storepass, and keypass may all be changed from this example. Be sure that the xml and jks files have the same name and are in the same directory, and the alias and passwords are updated in all commands as well as in the xml file. Some special characters in the passwords require additional quoting and escaping in commands and xml. To avoid this you can refrain from using $, &, \, :, ', ", <, and > in the passwords.

For this new signing profile to take effect, we need to set the niagara.signing.profile system property to the full path of the profile’s xml file. We can do this either by passing a system property argument to gradle commands from the command line

gradlew jar -Pniagara.signing.profile=/path/to/my\_signing\_profile.xml

Or you can add it to your gradle.properties create file if not exist file at USER\_HOME/.gradle/gradle.properties. Note: It may be necessary to restart the Gradle daemon for gradle.properties changes to take effect. The daemon can be restarted by running gradlew --stop

niagara.signing.profile=/path/to/my\_signing\_profile.xml

If you specify the path using backslashes, be sure to escape them with double backslashes (C:\\path\\to\\my\_signing\_profile.xml)

# Specifying Certificate Alias

By default, modules will be signed using the certificate in your signing profile with the alias Niagara4Modules. Since we used a different alias for our certificate, we will have to specify that alias in our build environment. The alias can be specified globally by adding the following block to your NIAGARA\_USER\_HOME/build.gradle file. Or you can add into your project build.gradle file see Screenshot for project build.gradle file

A picture containing table

Description automatically generated

...

subprojects { Project p ->

p.pluginManager.withPlugin('com.tridium.niagara-module') {

p.niagaraModule {

certAlias = "my-cert"

}

}

}

...

This should be sufficient in most cases, but if you discover you need to sign certain modules with a different certificate, you can specify the alias on a module specific basis by adding the following to your module’s gradle file.

...

niagaraModule {

moduleName = "module"

preferredSymbol = "mod"

runtimeProfile = "rt"

certAlias = "my-cert"

...

}

...

If you find you need to sign any modules with multiple certificates, you can specify the certAlias property in either of these locations with a comma separated list.

certAlias = "my-cert,my-other-cert"

# Creating internal CA Signed certificate

The first two approaches described above will require you to get your certificate signed by a CA and import it back into your keystore. To do this, you first need to generate a certificate signing request with the following command

keytool -certreq -alias my-cert -ext EKU="codeSigning" -keypass K3yP@ss -storepass St0reP@ss -keystore my\_signing\_profile.jks -file my-cert.csr

Document correction

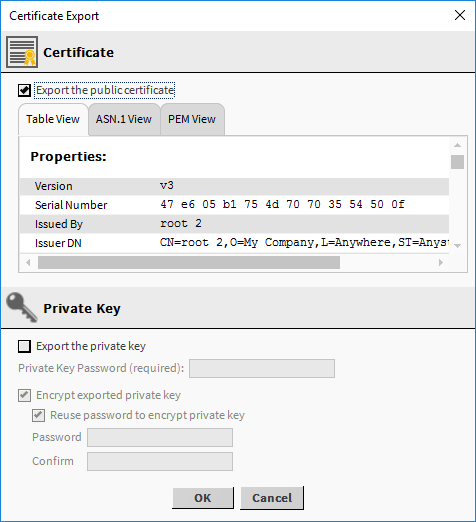
<https://www.niagara-community.com/s/article/Niagara-code-signing-documentation-correction>

Open workbench go to certificate management from tools menubar, fill the details as mentioned filled in below ss, you can enter your org details and select CA Option.

Graphical user interface, application

Description automatically generated

Select the new root CA certificate and click Export



To create the root CA certificate that will reside in each client’s User Trust Store, click **OK**.

The Certificate Export window opens with the file ready to export as a .pem file.

Graphical user interface, text, application

Description automatically generated

[Root and intermediate certificate checklist (niagara-community.com)](https://docs.niagara-community.com/bundle/StationSecurity/page/SSLCertificateChecklist.html)

Select Tools -> Certificate Signer Tool from menubar see below image for more details

Graphical user interface, text, application

Description automatically generated

Select Created CSR file change expiry dates if required longer certificate expiry see below screenshot for same.

Graphical user interface, text, application, email

Description automatically generated

On Click ok Save generated .pem file into certificate management/or any location in computer.

Restart the computer.

# Import certificate

To import certificate for Niagara modules, follow below steps

1. Restart workbench, stop running station and daemon certificate to take set properly.
2. Open certificate management from tools -> certificate management
3. Click on user trust store tab and import generated PEM certificated file signed by root CA

See below SS for more details.

Graphical user interface

Description automatically generated

1. Open build module in palette to verify.
2. To drag and drop or use component in station import. pem certificate in station certificate management.
   1. Open Platform service from station services
   2. PlatformService->CertManagerService->Click on User Trust Store
   3. Repeat step 3 & 4
3. Restart station and workbench if showing cert chain issue.