CONNECTING
THE WORLD


NF
25
CONNECTING
THE WORLD
NIAGARA FORUM APAC | 9-10 SEPTEMBER 2025

Module Signing

Bo Wang

APAC Training Supervisor

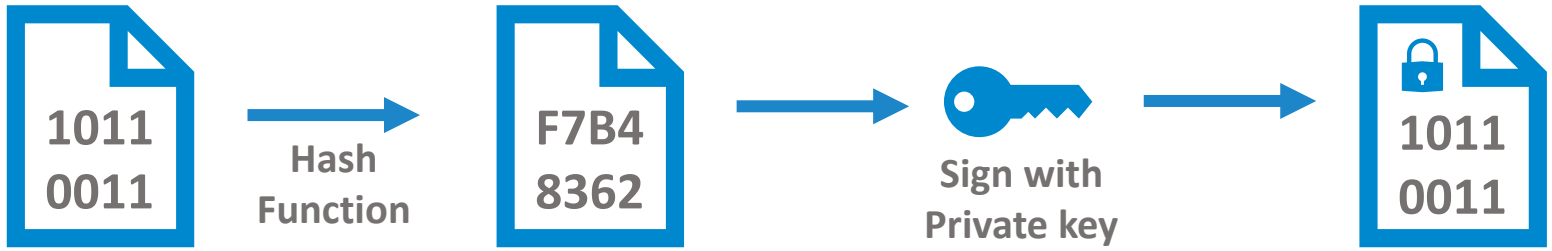
Tridium

Agenda

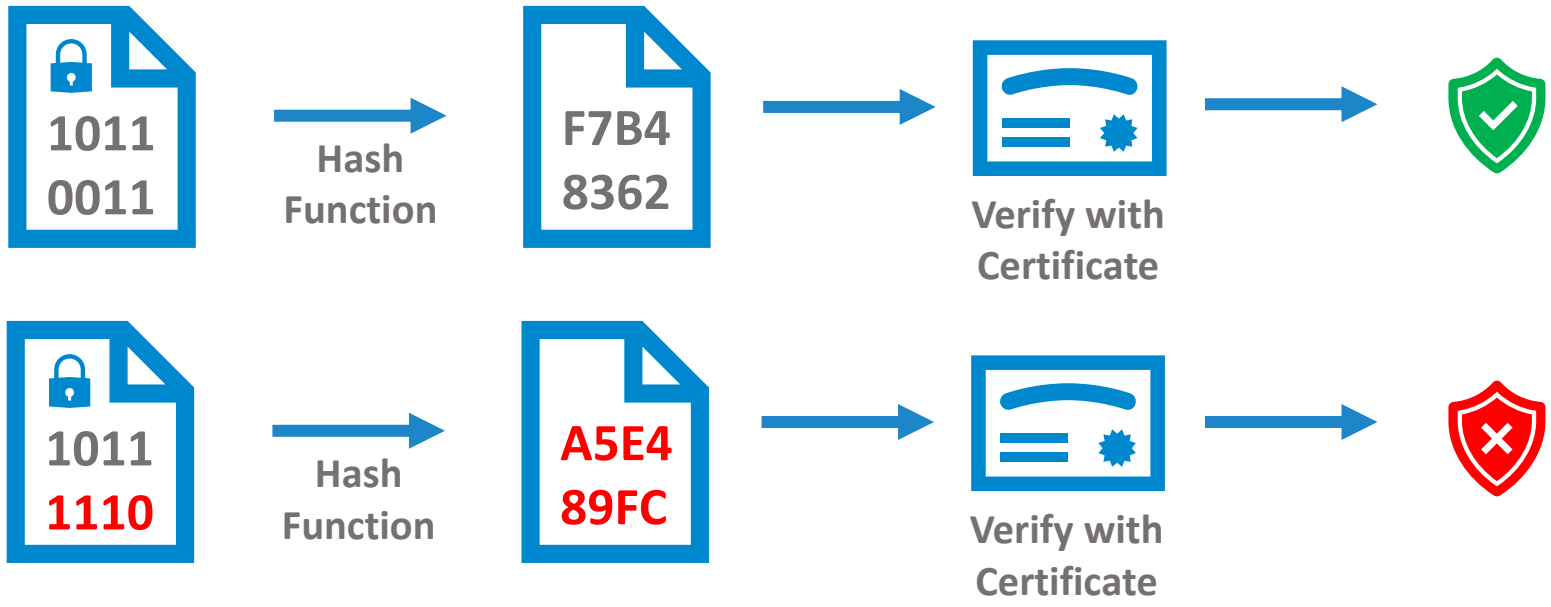
- What is code signing
- Niagara module verification mode
- Niagara Jar signer tool
- Code signing using certificate
- Code signing using HSMs

What is Code Signing

- Code signing is the process of applying a digital signature to a piece of code so that it can be later verified to ensure that it has not been modified after it was signed.



What is Code Signing



Benefits of Code Signing

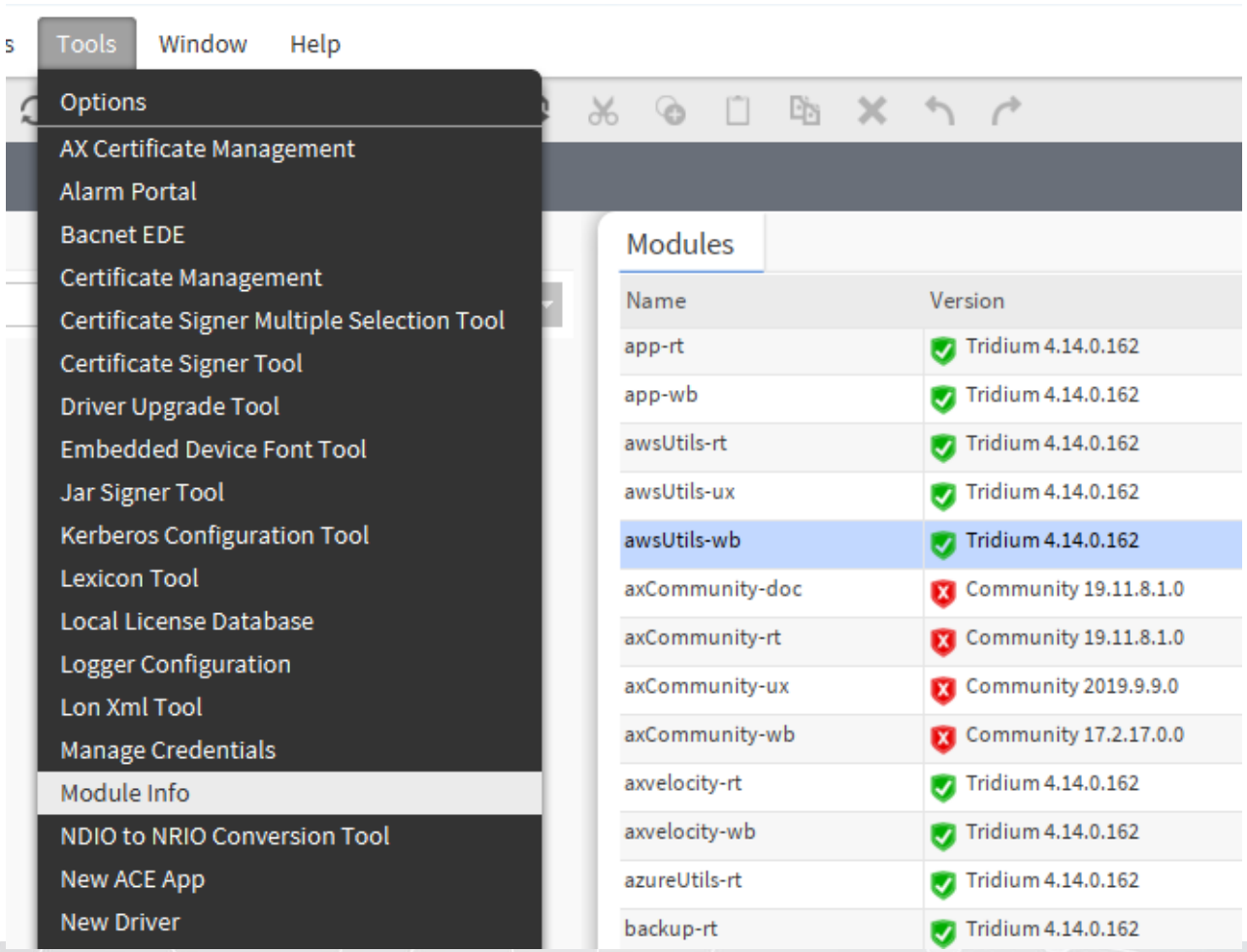
- Confirm code is from a trusted source
- Prevent installation of malicious code
- Satisfy security requirements

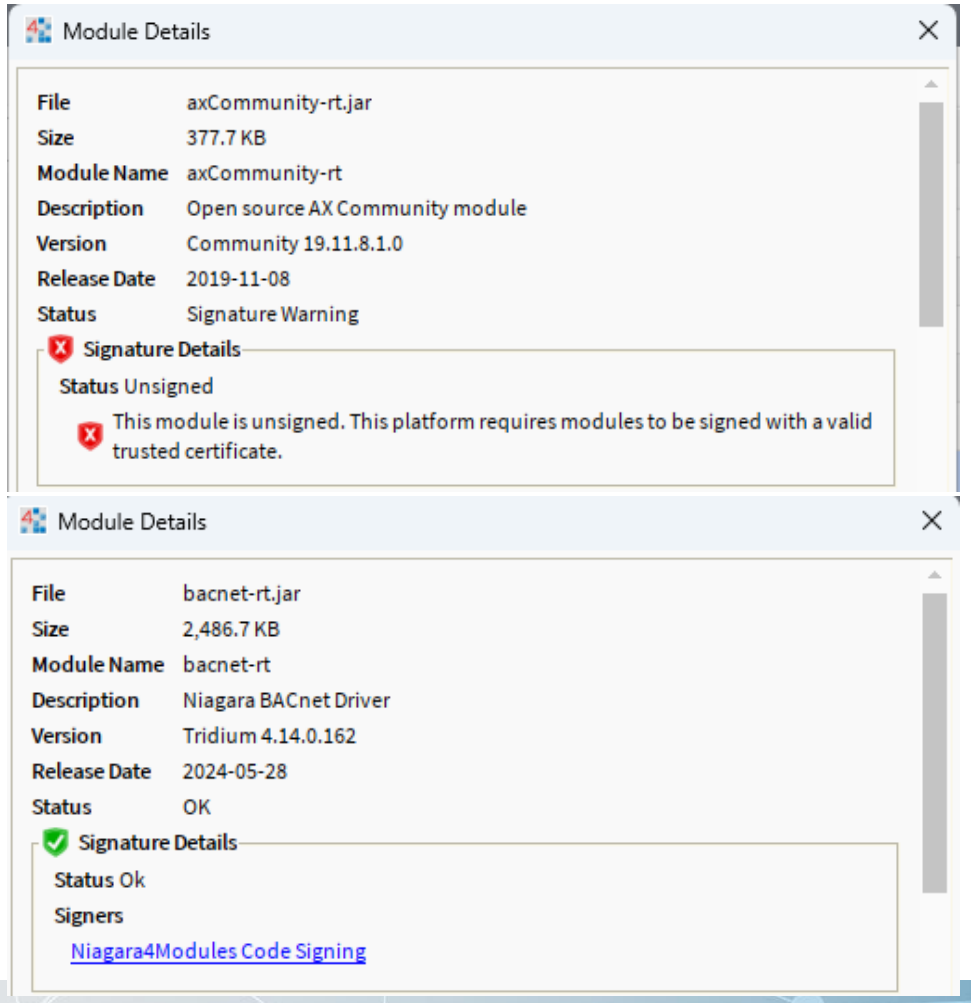
Verification Modes

Low	Medium	High
• Signing optional	• Modules must be signed by a trusted certificate • Self-signed certificates acceptable, but must be installed in user trust store	• Modules must be signed by a CA signed certificate • Internal CA is acceptable, but must be installed in user trust store

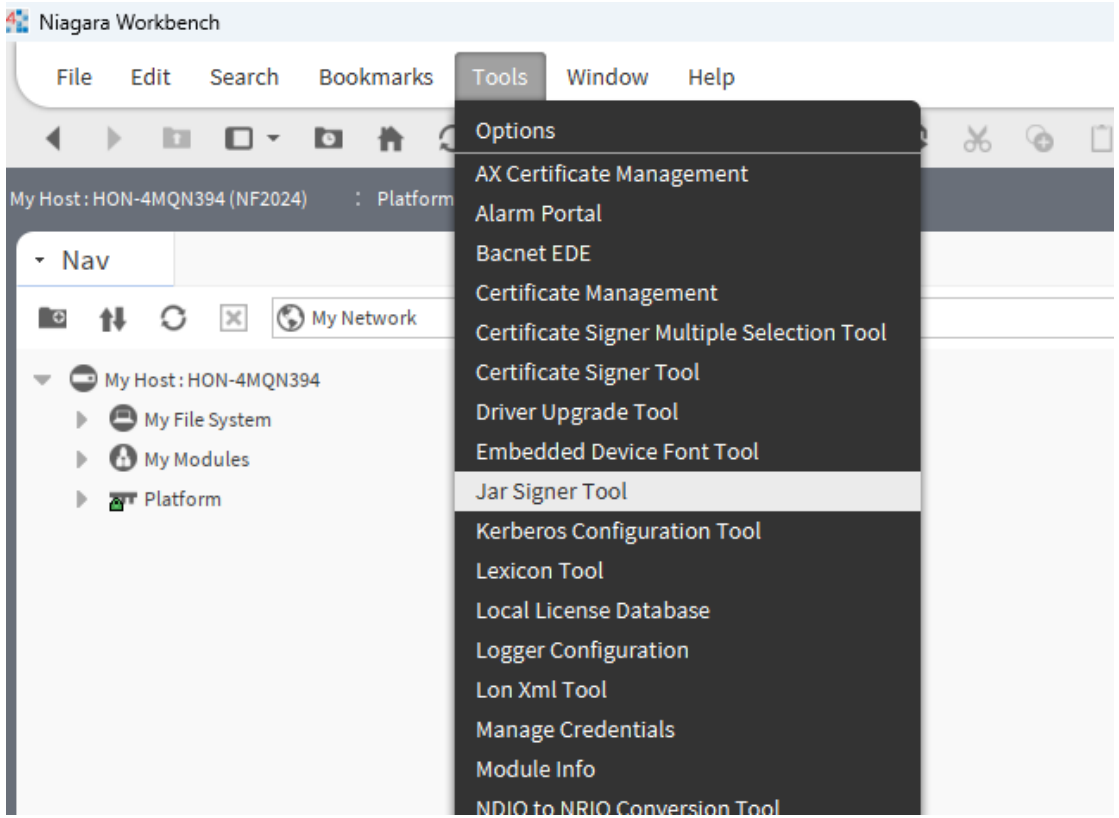
niagara.moduleVerificationMode=[**low, medium, high**] in *System.Properties* file

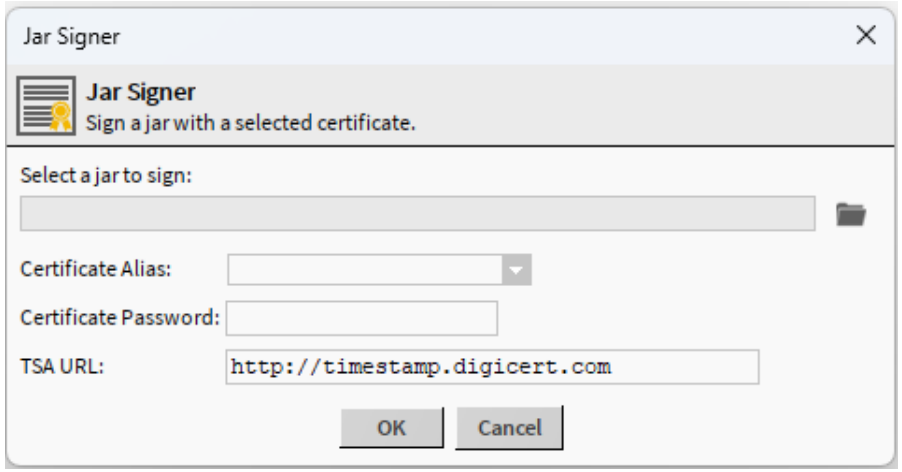
Module Information





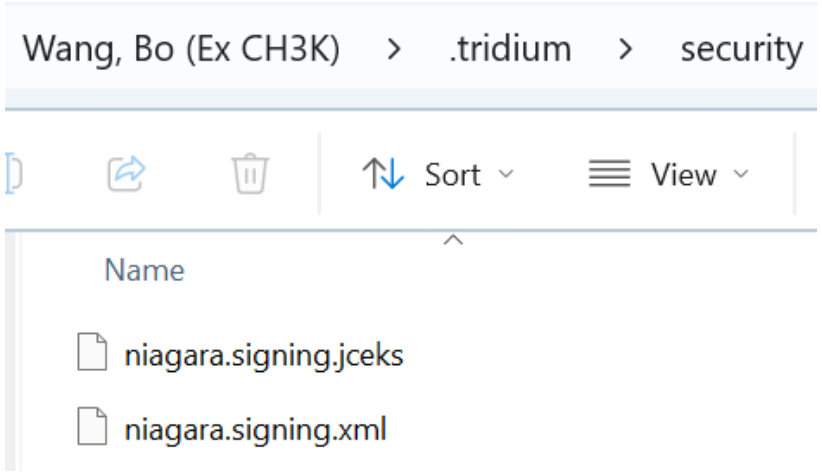
Niagara Jar Signer Tool





Default Signing Profile

- Starting in 4.6 any module you build will automatically be signed with a generic, auto-generated, self-signed certificate.
- Default Signing Profile is in USER_HOME/.tridium/Security



Creating a Signing Profile

```
gradlew :createProfile --profile-path path\to\my_signing_profile.xml
```

my_signing_profile.xml

```
<?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.storepass">1np209gr2f6niog5jms5</entry>
    <entry key="niagara.signing.validity">365</entry>
    <entry key="niagara.signing.keyalg">RSA</entry>
    <entry key="niagara.signing.dname">C=US,ST=Virginia,L=Richmond,O=TridiumUniversity,OU=Engineer
    <entry key="niagara.signing.profileType">com.tridium.gradle.plugins.signing.profile.Restricted
    <entry key="niagara.signing.storetype">JCEKS</entry>
    <entry key="niagara.signing.refresh">183</entry>
    <entry key="niagara.signing.keysize">3072</entry>
    <!--  The lines below will enable timestamping -->
    <entry key="niagara.signing.standardtsa">
        http://timestamp.digicert.com
    </entry>
</properties>
```

Generate Self-Signed Certificate

```
gradlew :generateCertificate --profile-path path\to\my_signing_profile.xml --alias MyCertificate
```

build.gradle.kts (helloNiagara)

```
46
47  signingServices { this: SigningServices
48      // Disable the use of the default profile; this will cause build failures instead
49      // of silently falling back to the default
50      signingProfileFactory { this: SigningProfileFactorySpec
51          allowDefaultProfile.set(false)
52      }
53  }
54
55  niagaraSigning { this: SigningExtension
56      aliases.set(listOf("MyCertificate"))
57      signingProfileFile.set(project.layout.projectDirectory.file( path: "path/to/my_signing_profile.xml"))
58  }
```

Get Certificate Signed by CA

- Create Certificate Signing Request

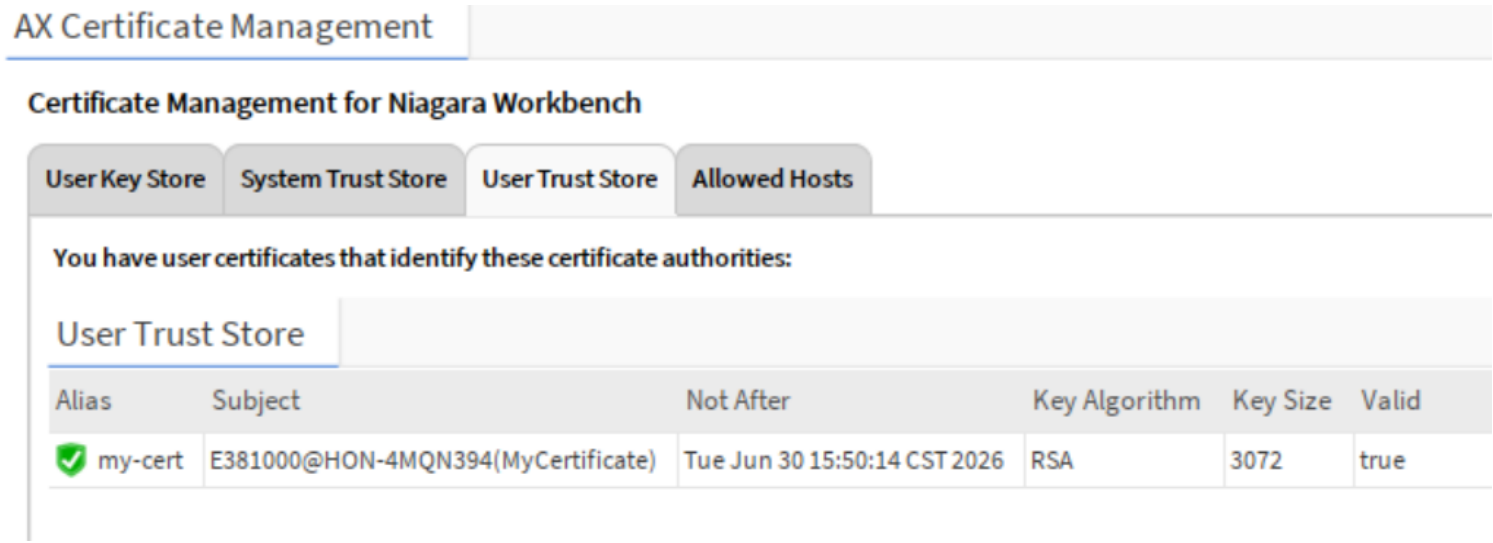
```
gradlew :exportCertificate --profile-path path\to\my_signing_profile.xml --alias MyCertificate --csr-only --pem-file my-cert.csr
```

- Import Signed Certificate

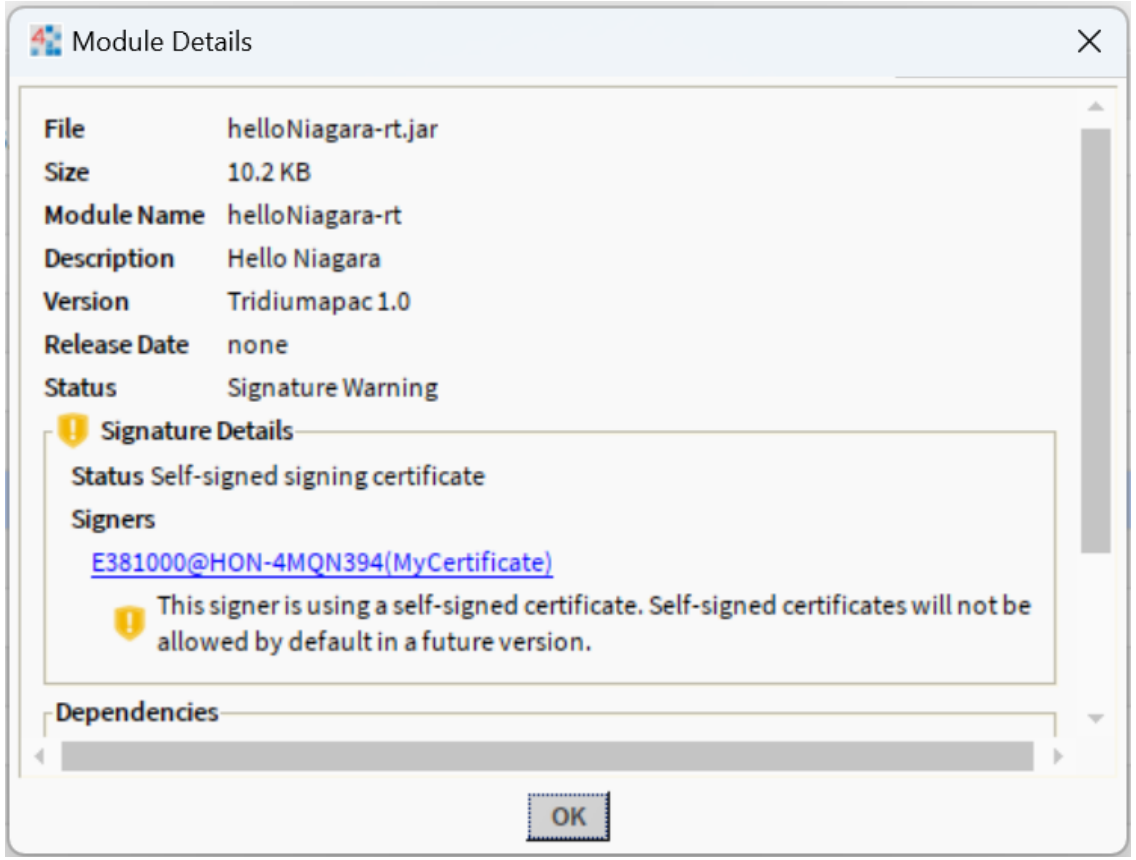
```
gradlew :importCertificate --profile-path path\to\my_signing_profile.xml --alias MyCertificate --pem-file my-cert.pem
```

Establishing Trust

```
gradlew :exportCertificate --profile-path path\to\my_signing_profile.xml --alias MyCertificate --pem-file my-cert.pem
```



Module Verification



Hardware Security Module (HSM)

- HSMs are hardened, tamper-resistant hardware devices used to secure cryptographic processes
- HSMs could be a card inside of a PC, a rack mounted device, a USB device or part of a cloud service
- As of mid-2023, all commercial certificate authorities require the use of a HSM to protect the private key of any code signing certificate
- Starting in 4.14 version, the gradle scripts are updated to support code signing compiled modules using a physical HSM device that supports the PKCS11 standard

Create a JarSignerSigningProfile

- A Java properties file
- Configure the command and arguments used to run *jarsigner*

```
# TODO Update with the TSA recommended by your certificate vendor and uncomment
# these lines
# jarsigner.args+=-tsa
# jarsigner.args+=http://timestamp.digicert.com

# jarsigner.args+=-tsadigestalg
# jarsigner.args+=SHA256

jarsigner.args+=-keystore
jarsigner.args+=NONE

jarsigner.args+=-storetype
jarsigner.args+=PKCS11

jarsigner.args+=-storepass
# TODO Update this if your credentials file specifies a different variable
jarsigner.args+=${jarsigner.storepass}

jarsigner.args+=-providerClass
jarsigner.args+=sun.security.pkcs11.SunPKCS11

# TODO Newer jarsigner versions may require the use of 'addProvider' instead of
# 'providerClass'; this may be the case if you get odd errors like
# 'java.security.NoSuchAlgorithmException: SHA256 MessageDigest not available'
# jarsigner.args+=-addProvider
# jarsigner.args+=SunPKCS11
jarsigner.args+=-providerArg

# TODO Update with the absolute path to your HSM .cfg file
```

Signing with JarSignerSigningProfile

- Set the configured Profile file as the default in *build.gradle.kts*

```
// Load the correct signing profile
niagaraSigning {
  aliases.set(listOf(TODO("Set the correct alias as reported by your HSM")))
  signingProfileFile.set(
    project.rootProject.layout.projectDirectory.file("security/niagara.signing.properties")
  ) ^niagaraSigning
}
```

Reference

- Niagara help doc

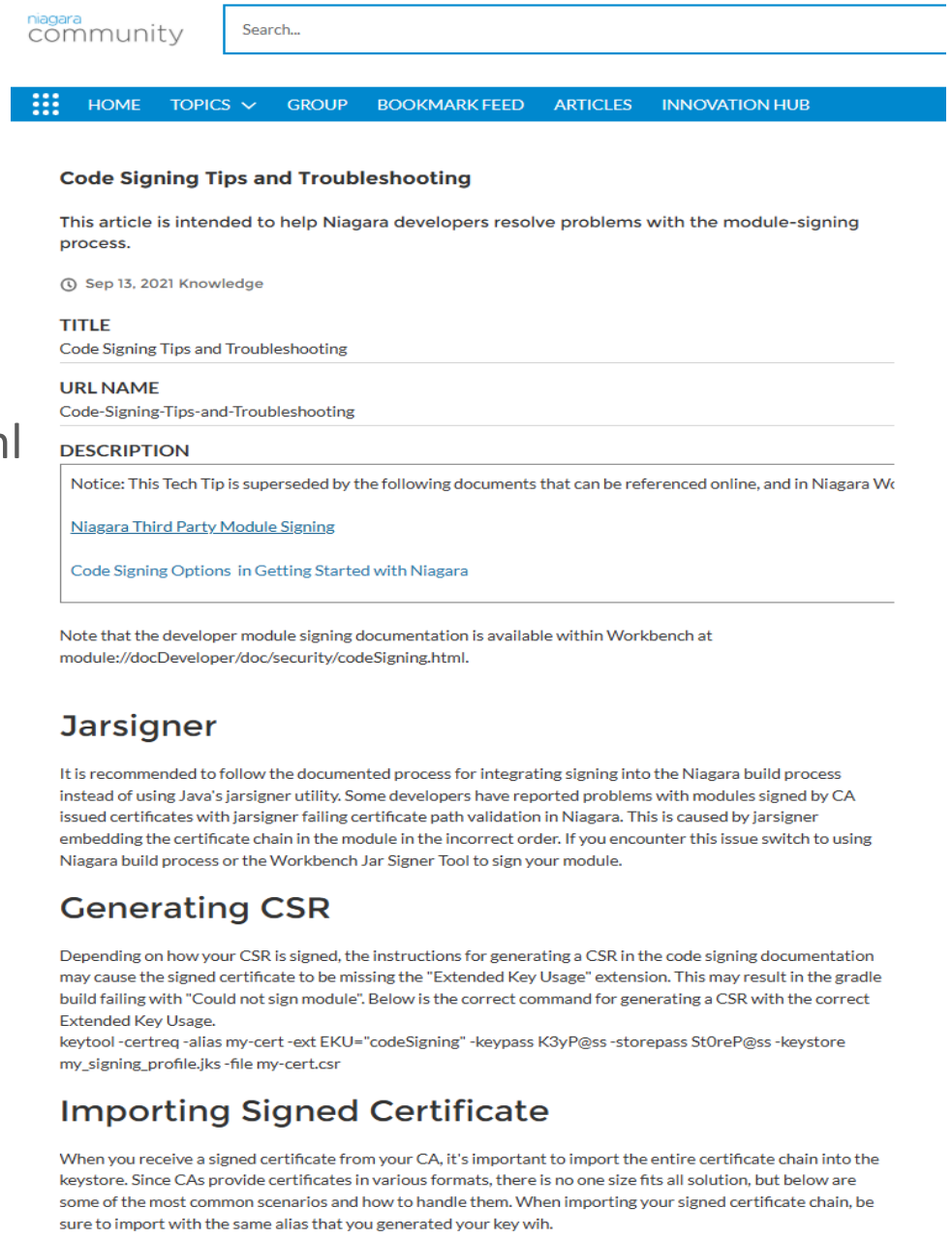
  local:|module://docDeveloper/doc/security/codeSigning.html

- Niagara community article

  https://www.niagara-community.com/s/article/Code-Signing-Tips-and-Troubleshooting

  https://www.niagara-community.com/s/article/Code-Signing-using-Hardware-Security-Modules

NIAGARA FORUM
CONNECTING THE WORLD



Summary

- Module signing is ***not optional***, it is a critical cybersecurity practice
- It verifies the ***who*** (publisher) and the ***what*** (integrity) of a software module
- It protects Niagara stations from malware and unauthorized code execution
- ***Always use signed modules*** from trusted sources to ensure system safety and reliability


THANK YOU
NF
25
CONNECTING
THE WORLD