

# Technical Document

## **Niagara Third Party Module Signing**

**August 3, 2020**



# Niagara Third Party Module Signing

## **Tridium, Inc.**

3951 Westerre Parkway, Suite 350  
Richmond, Virginia 23233  
U.S.A.

## **Confidentiality**

The information contained in this document is confidential information of Tridium, Inc., a Delaware corporation ("Tridium"). Such information and the software described herein, is furnished under a license agreement and may be used only in accordance with that agreement.

The information contained in this document is provided solely for use by Tridium employees, licensees, and system owners; and, except as permitted under the below copyright notice, is not to be released to, or reproduced for, anyone else.

While every effort has been made to assure the accuracy of this document, Tridium is not responsible for damages of any kind, including without limitation consequential damages, arising from the application of the information contained herein. Information and specifications published here are current as of the date of this publication and are subject to change without notice. The latest product specifications can be found by contacting our corporate headquarters, Richmond, Virginia.

## **Trademark notice**

BACnet and ASHRAE are registered trademarks of American Society of Heating, Refrigerating and Air-Conditioning Engineers. Microsoft, Excel, Internet Explorer, Windows, Windows Vista, Windows Server, and SQL Server are registered trademarks of Microsoft Corporation. Oracle and Java are registered trademarks of Oracle and/or its affiliates. Mozilla and Firefox are trademarks of the Mozilla Foundation. Echelon, LON, LonMark, LonTalk, and LonWorks are registered trademarks of Echelon Corporation. Tridium, JACE, Niagara Framework, and Sedona Framework are registered trademarks, and Workbench are trademarks of Tridium Inc. All other product names and services mentioned in this publication that are known to be trademarks, registered trademarks, or service marks are the property of their respective owners.

## **Copyright and patent notice**

This document may be copied by parties who are authorized to distribute Tridium products in connection with distribution of those products, subject to the contracts that authorize such distribution. It may not otherwise, in whole or in part, be copied, photocopied, reproduced, translated, or reduced to any electronic medium or machine-readable form without prior written consent from Tridium, Inc.

Copyright © 2021 Tridium, Inc. All rights reserved.

The product(s) described herein may be covered by one or more U.S. or foreign patents of Tridium.

# Contents

<b>About this guide .....</b>	<b>5</b>
Document change log .....	5
Related documentation .....	5
<b>Chapter 1 Overview .....</b>	<b>7</b>
Verification modes.....	7
Staged roll-out .....	7
New tools .....	8
Changes to existing tools .....	8
Signature statuses .....	14
<b>Chapter 2 Signing A Third Party Module.....</b>	<b>15</b>
Creating a self-signed certificate.....	15
Using the Jar Signer Tool.....	17
Importing the self-signed certificate into a platform.....	19
<b>Chapter 3 Common tasks.....</b>	<b>21</b>
Checking module signatures on a remote platform .....	21
Checking module signatures on your local platform .....	22
Installing modules with signature warnings.....	23
Obtaining properly signed modules .....	24
<b>Chapter 4 Troubleshooting.....</b>	<b>27</b>
Signature warnings or errors when installing module.....	27
Signature warnings in the Application Director .....	27
Station fails with error message .....	27
Workbench fails to start after installing a module.....	27
<b>Index.....</b>	<b>29</b>



## About this guide

This topic contains important information about the purpose, content, context, and intended audience for this document.

### Product Documentation

This document is part of the Niagara technical documentation library. Released versions of Niagara software include a complete collection of technical information that is provided in both online help and PDF format. The information in this document is written primarily for Systems Integrators. To make the most of the information in this book, readers should have some training or previous experience with Niagara software, as well as experience working with JACE network controllers.

### Document Content

This document discusses code-signing for third-party modules and the phased levels of enforcement over the next few releases. Sections in this guide include common tasks, code-signing concepts, and reference information. Also included are images and descriptions of the primary software user interface windows involved.

## Document change log

Changes to this document are listed in this topic.

### August 3, 2020

- Added the chapter, "Signing a third party module".

### March 21, 2019

- Initial publication

## Related documentation

Additional information on Niagara systems, devices and protocols is available in the following documents.

- *Getting Started with Niagara*
- *Niagara Developer Guide*
- *Niagara Station Security Guide*



# Chapter 1 Overview

## Topics covered in this chapter

- ◆ Verification modes
- ◆ Staged roll-out
- ◆ New tools
- ◆ Changes to existing tools
- ◆ Signature statuses

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. This can help establish that the code came from a trusted source and reduce the risk of installing malicious code. In Niagara, all stock modules are signed and verified at runtime.

In Niagara 4.8 and later, there is added support for the signing and verification of third party modules. Third party module signing is still optional in most cases, but this will gradually shift to a requirement over the course of the next three releases.

**CAUTION:** A security best practice is to encourage the original authors of a module to sign their own code. Signing a module on behalf of a 3rd party should be done only as a last option, and only if you trust the authors.

For more details on code signing, see “Code Signing” in the *Niagara Developer Guide* (Niagara 4.6 and later).

## Verification modes

Niagara supports three different verification modes for third party modules. Module verification is based on the `moduleVerificationMode`, which can be configured to Low, Medium or High.

Each mode has the following behavior:

- Low — Any modules that are not signed or are signed with an untrusted or expired certificate will cause warnings but will still function normally. Errors will occur if a signed module is modified after it was signed and installation of such modules is not allowed.
- Medium — All modules must be signed by a valid, trusted certificate, but this certificate can be self-signed. Installation of unsigned or invalidly signed modules is not allowed.
- High — All modules must be signed with a CA signed certificate. An internal CA can be used, but in this case, the CA certificate must be imported into the user trust store. Installation of modules signed with self-signed certificates is not allowed.

The current default verification mode is Low, but this can be changed if desired. Change the verification mode for a device by adding the following line to the `system.properties` file. Where, you would enter only one of the options shown between the square brackets.

```
niagara.moduleVerificationMode=[low, medium, high]
```

## Staged roll-out

The module signing requirement will be enforced in stages. This gradual increase in the default module verification mode gives customers time to adjust to the new requirement. Although subject to change, the planned roll-out schedule will span several releases with the following steps.

- Niagara 4.8 — Default verification mode is Low. Customers can increase the verification mode as needed for testing or to improve security posture.

- Niagara 4.9 — Default verification mode is Medium. Customers can decrease the verification mode to “low” if they are not ready for module signing yet, or up to “high” if they want to enforce more strict requirements.
- Unspecified future Niagara 4.x release — Default verification mode is High. Customers can decrease verification mode to “medium” if they still have some modules signed with self-signed certificates. Decreasing verification mode to “low” will no longer be allowed except for developer customers.

## New tools

In Niagara 4.8 and later, there are added Workbench tools to support module signing.

The **Module Info View** checks the signatures of modules installed on your local machine. The view lists the currently installed modules, each with a signature status icon.

The **Jar Signer Tool** is useful for non-developers or anyone using an unsigned legacy module that is no longer supported by the vendor. The Jar Signer allows you to “sign” an unsigned \*.jar file using a code signing certificate.

For more details, see “About Workbench tools” in *Getting Started with Niagara*.

## Changes to existing tools

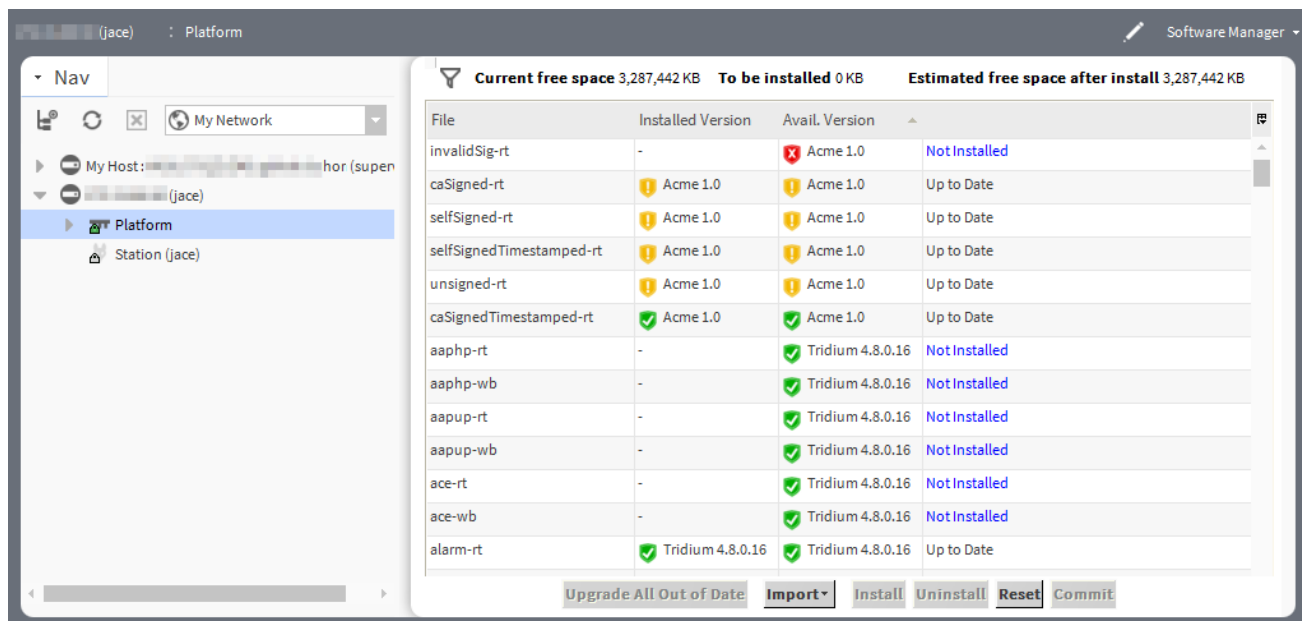
Several existing tools have been modified to accommodate module signing.

### Software Manager/Commissioning Wizard

Both the **Software Manager** and **Commissioning Wizard** show an added icon in the Installed Version and Available Version columns indicating the signature status of the installed and available modules, as shown here.

**NOTE:** A red icon (❌) there are errors that are not acceptable for the current module verification mode. A yellow icon (⚠️) there are warnings, but they are acceptable for the current module verification mode. A green icon (✅) indicates that the module is signed with a Certificate Authority (CA). For details, see “Signature statuses” in this chapter.

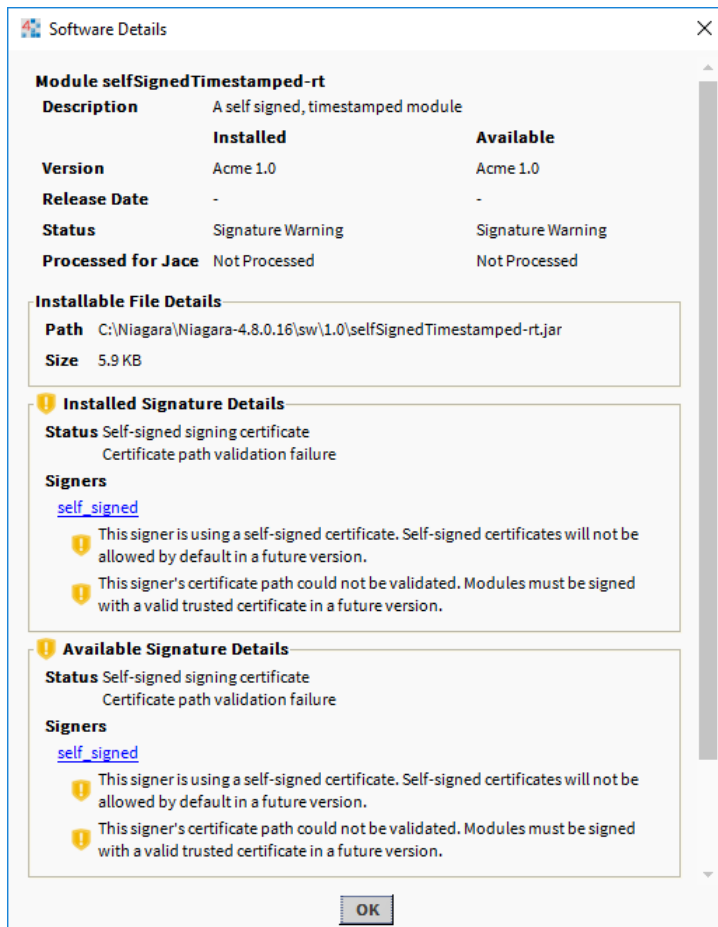
Figure 1 Software Manager view with signature status icons





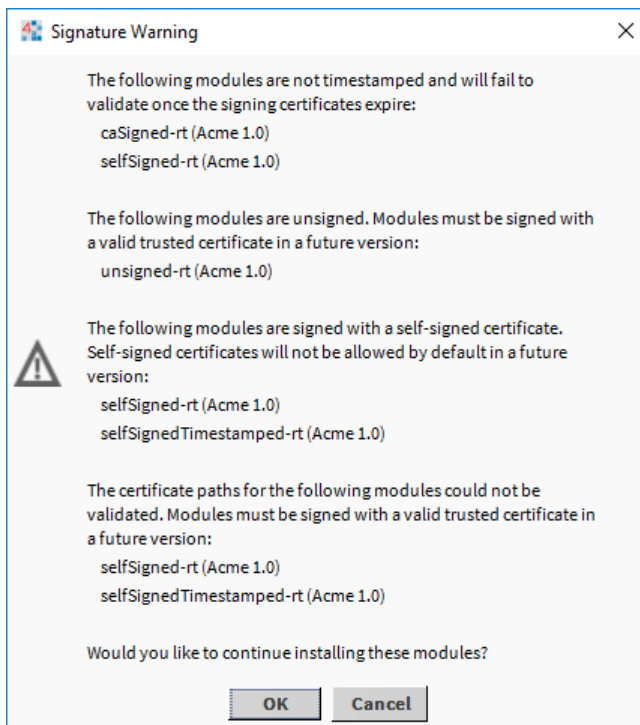
The **Software Details** window also includes signature status details for installed and available modules. This window displays all the signature statuses of the selected module with additional details. It will also provide a link to view the certificate that the module is signed with.

Figure 2 Example Software Details dialog



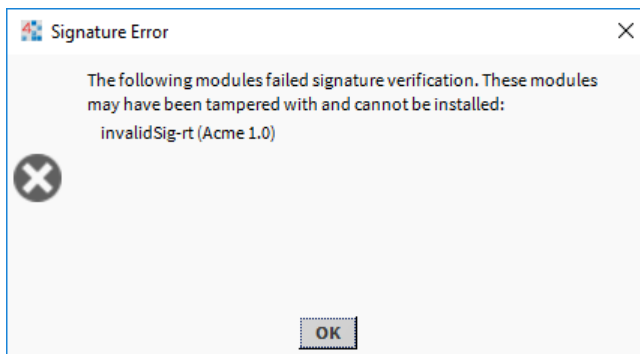
Attempting to install modules with signature warnings will cause a **Signature Warning** dialog to be displayed.

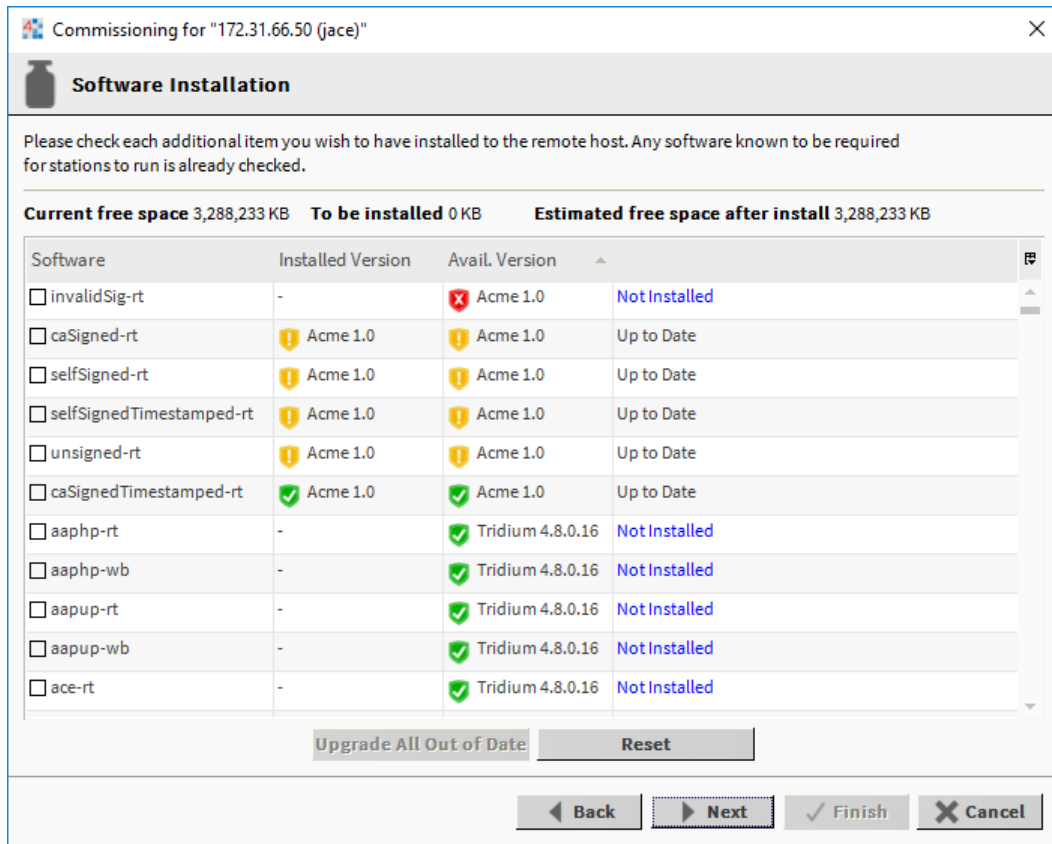
Figure 3 Example Software Manager signature warning

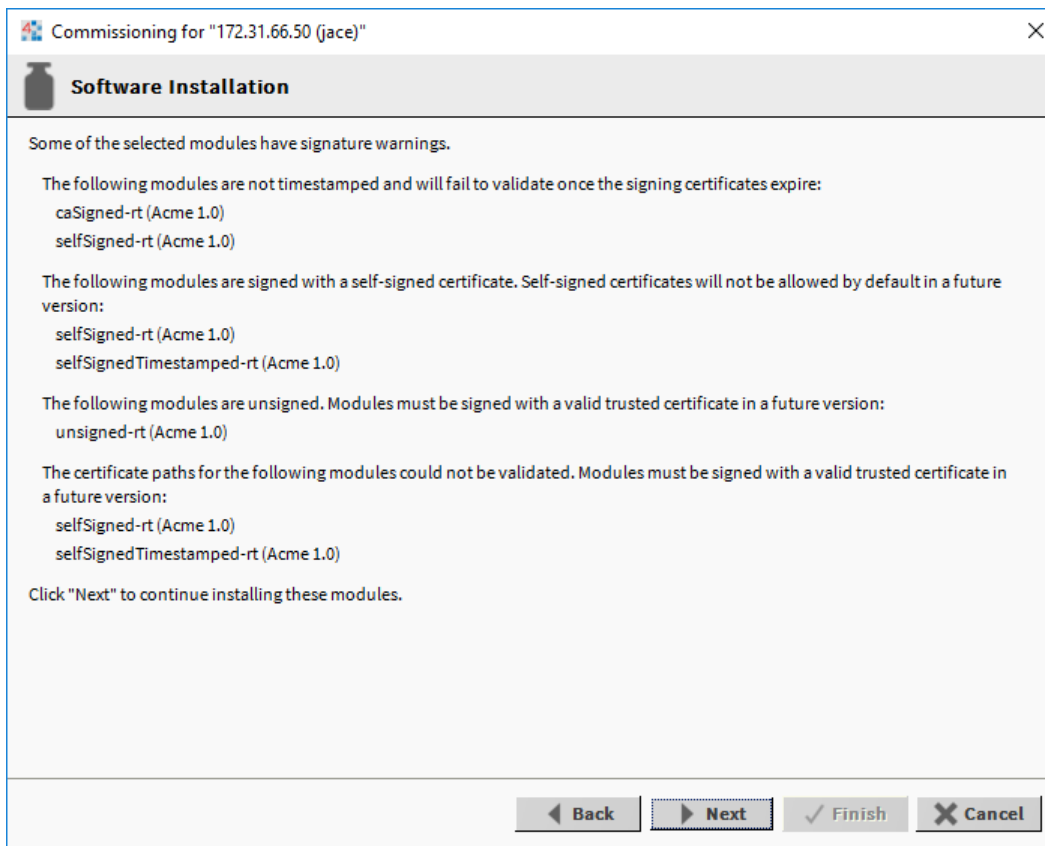


**NOTE:** Attempting to install modules with signature errors will cause the installation to fail.

Figure 4 Example Software Manager signature error



**Figure 5** Commissioning Wizard software installation window shows module signature status icons

**Figure 6** Commissioning Wizard signature warning

## Station Copier changes

When installing a station that requires additional dependencies to be installed using the **Station Copier**, any signature warnings for the dependencies will be displayed in a **Signature Warning** dialog, and any signature errors will cause station copy to fail.

## Provisioning Install Software step changes

The Provisioning **Install Software** step dialog shows signature status icons next to each version of each available module. Selecting modules with signature warnings will cause a warning dialog to display.

Figure 7 Provisioning Install Software dialog

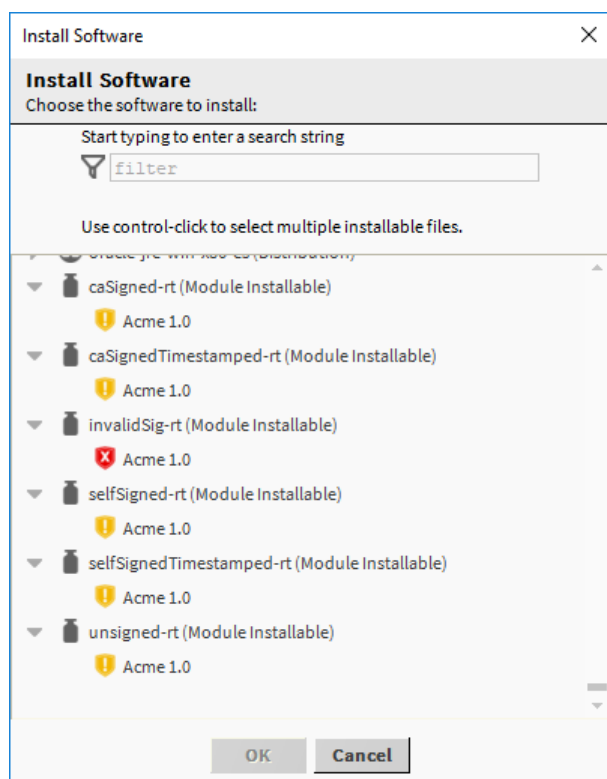
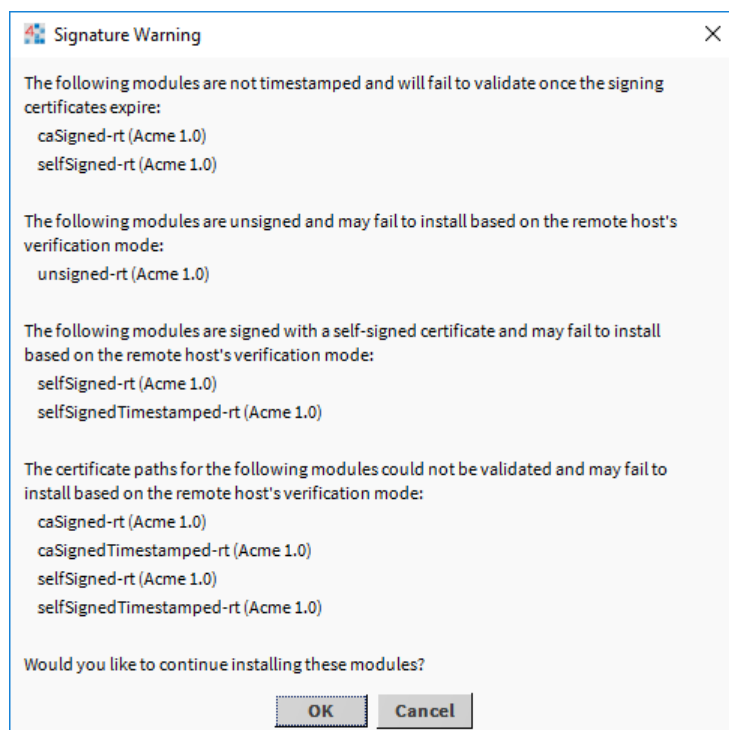


Figure 8 Install Software signature warning



**NOTE:** Even if the warnings are accepted, the modules may fail to install on the remote host(s) since the verification mode and trust store of the remote host(s) cannot be determined until the job is run. The job log will provide more details for any failures.

## Signature statuses

Following is a list of all possible signature statuses. Each module can have one or more of these signature statuses.

- **Ok** – The module is signed with a trusted CA (Certification Authority) signed certificate and is validly timestamped.
- **Not timestamped** – The module is not timestamped. This status is allowed in all verification modes, but verification will fail once the signing certificate expires.
- **Self-signed signing certificate** – The module is signed with a self-signed certificate. This status is allowed in medium verification mode as long as the certificate is installed in the user trust store, but will not be allowed in high verification mode.
- **Self-signed timestamp certificate** – The module is timestamped by a TSA (Time Stamping Authority) that uses a self-signed certificate. This status is allowed in medium verification mode as long as the certificate is installed in the user trust store, but will not be allowed in high verification mode.
- **Certificate path validation failure** – The module is signed with a certificate that failed validation. This usually happens when there is not a trust anchor for the certificate in one of the trust stores, or the certificate has expired and the module is not timestamped. This status is only allowed in low verification mode.
- **Unsigned** – The module is not signed. This status is allowed only in low verification mode.
- **Invalid signature** – The module failed signature verification. This usually means the module was modified after it was signed, possibly by a malicious party. This status is not allowed in any verification mode.
- **Unknown** – The module's signature status is not available, either because the remote platform's version is earlier than Niagara 4.8 where signature verification is not supported, or the signature needs to be re-loaded. Rebooting the platform will reload signatures on supported versions.

# Chapter 2 Signing A Third Party Module

## Topics covered in this chapter

- ◆ Creating a self-signed certificate
- ◆ Using the Jar Signer Tool
- ◆ Importing the self-signed certificate into a platform

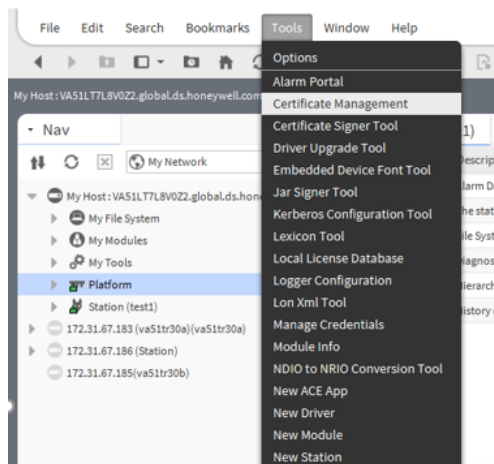
Code signing is the process of applying a digital signature to a piece of code so that it can be verified to ensure that it has not been modified after it was signed.

## Creating a self-signed certificate

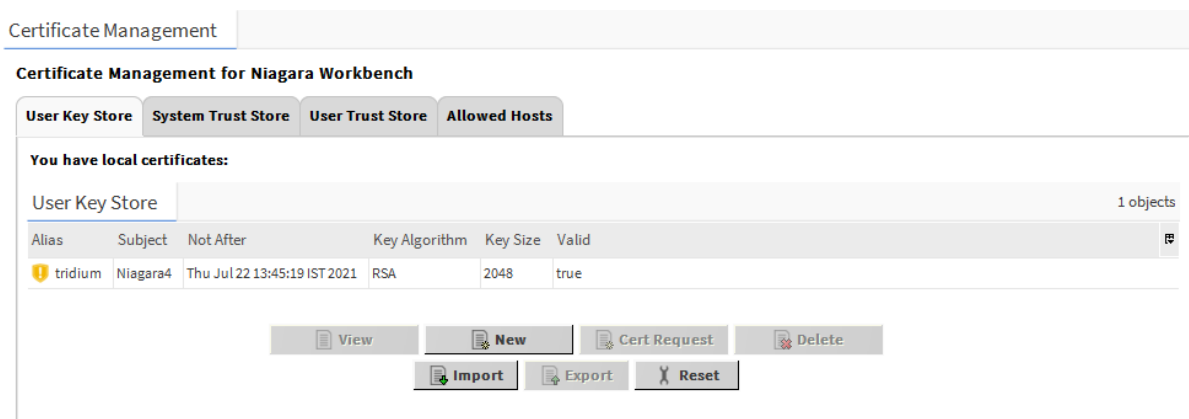
This topic explains how to generate a self-signed certificate.

**Prerequisites:** Workbench is running on your PC.

**Step 1** In Workbench click **Tools**→**Certificate Management**.

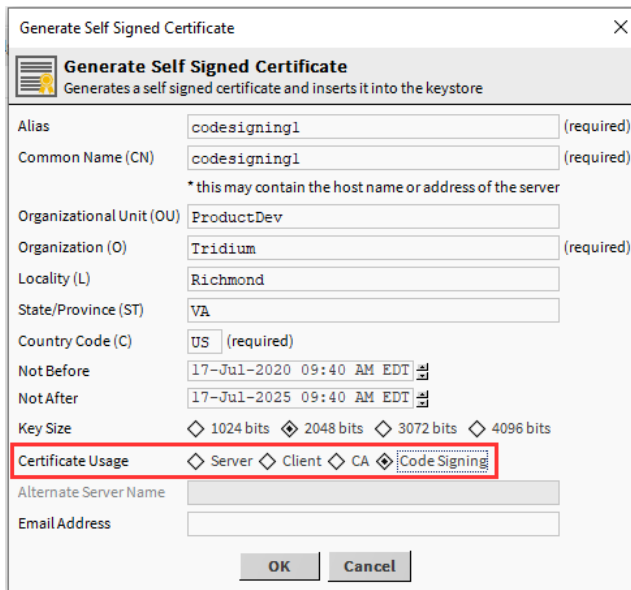


**Step 2** Click **New**.



The **Generate Self Signed Certificate** window appears.

**Step 3** Fill in all the properties, click the **Code Signing** option for **Certificate Usage** (as shown here), and click **OK**.



**Generate Self Signed Certificate**  
Generates a self signed certificate and inserts it into the keystore

Alias: codesigning1 (required)  
 Common Name (CN): codesigning1 (required)  
 \* this may contain the host name or address of the server  
 Organizational Unit (OU): ProductDev  
 Organization (O): Tridium (required)  
 Locality (L): Richmond  
 State/Province (ST): VA  
 Country Code (C): US (required)  
 Not Before: 17-Jul-2020 09:40 AM EDT  
 Not After: 17-Jul-2025 09:40 AM EDT  
 Key Size: 1024 bits 2048 bits 3072 bits 4096 bits  
**Certificate Usage: Server Client CA Code Signing**  
 Alternate Server Name:  
 Email Address:

OK Cancel

The **Private key Password** window opens.



**Private Key Password**  
Private Key Password

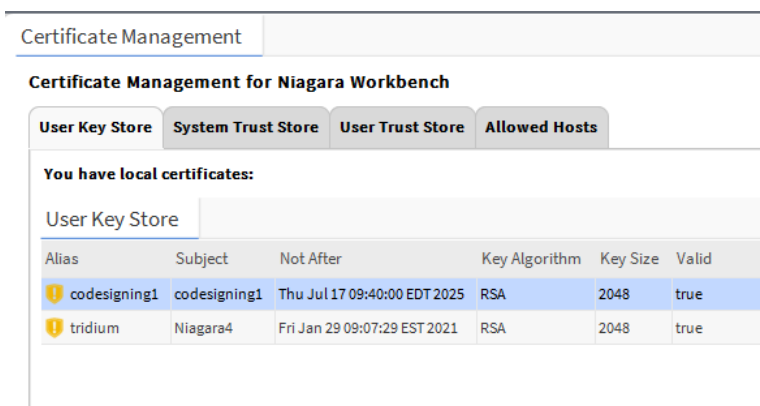
Private Key Password (required):  
 Password:   
 Confirm:   
 OK Cancel

Step 4

Enter a password and click **OK**

**NOTE:** The password should be at least 10 characters. At least one character should be a digit, one must be lowercase and one must be uppercase letter.

Step 5 The certificate is now visible under User Key Store.



**Certificate Management**

**Certificate Management for Niagara Workbench**

User Key Store System Trust Store User Trust Store Allowed Hosts

**You have local certificates:**

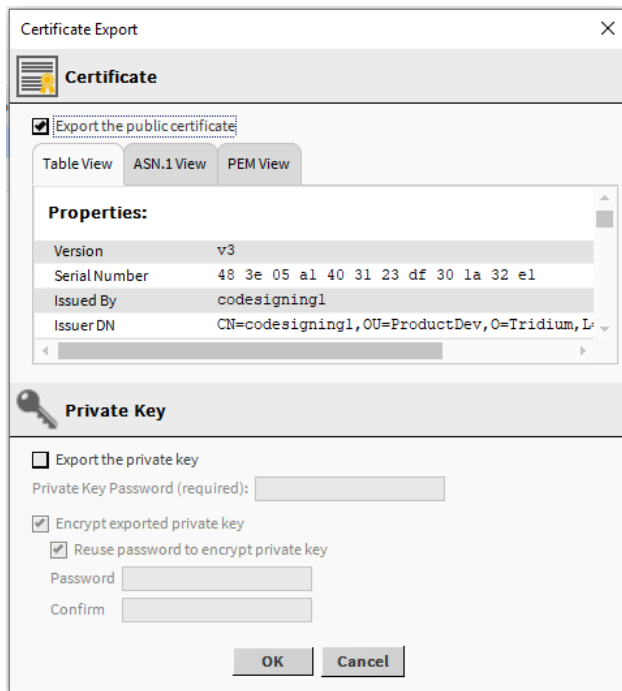
User Key Store

Alias	Subject	Not After	Key Algorithm	Key Size	Valid
codesigning1	codesigning1	Thu Jul 17 09:40:00 EDT 2025	RSA	2048	true
tridium	Niagara4	Fri Jan 29 09:07:29 EST 2021	RSA	2048	true

Step 6 Doubleclick the certificate.

A **Certificate Export** window opens.



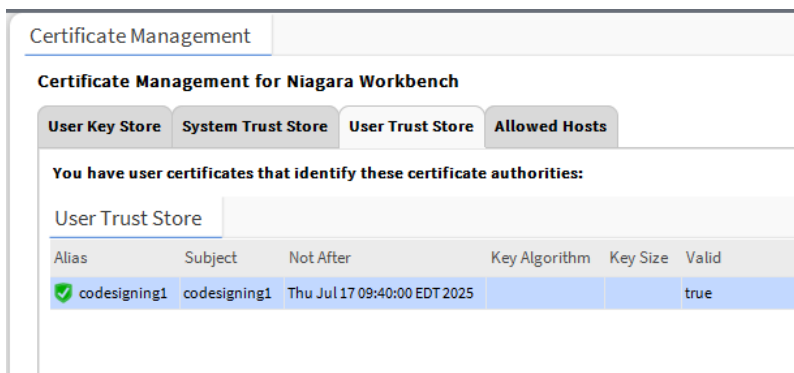


Step 7 Click **OK** to export the certificate.

Step 8 Click **User Trust Store** tab and click import. To import the certificate.

Step 9 In the Certificate Import window, navigate to the certificate, click to select it, and click Open to import it.

You can see the certificate with a green icon under **User Key Store**

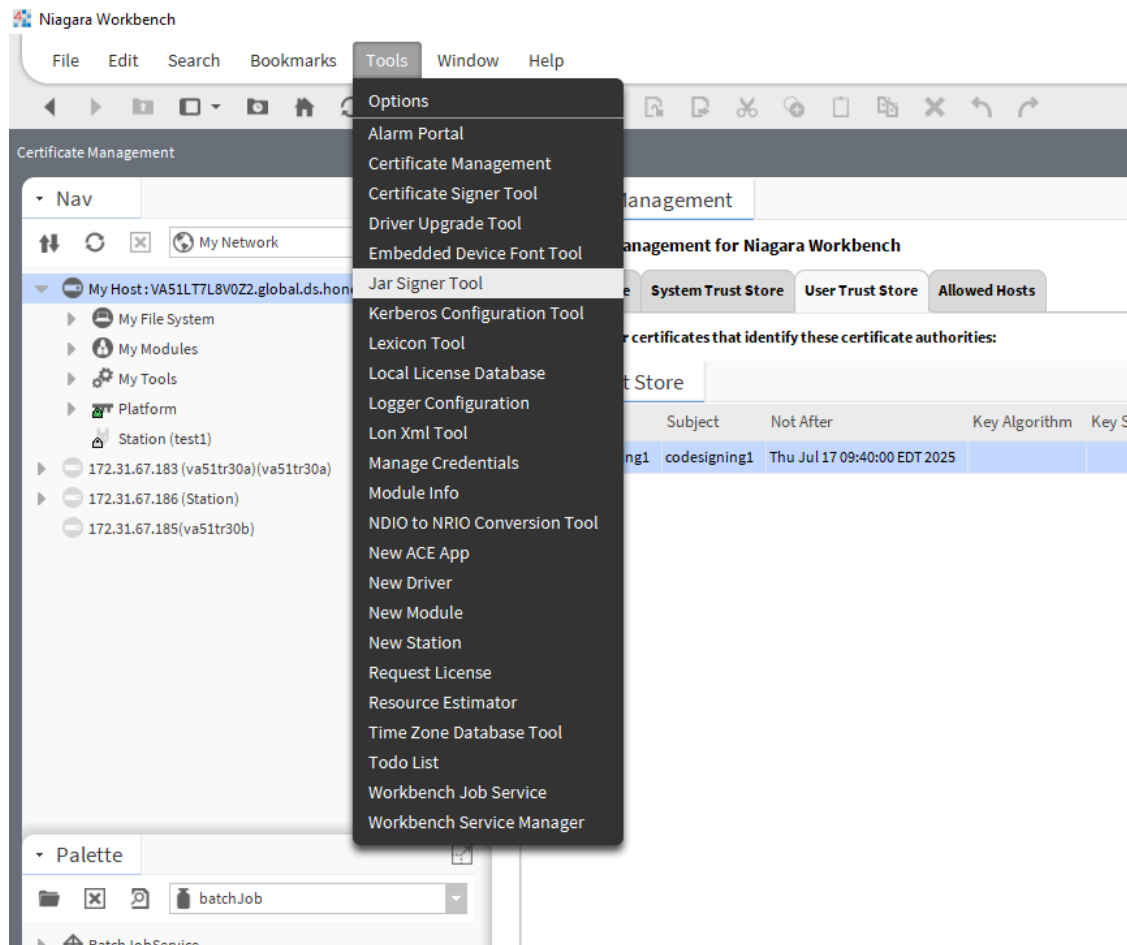


## Using the Jar Signer Tool

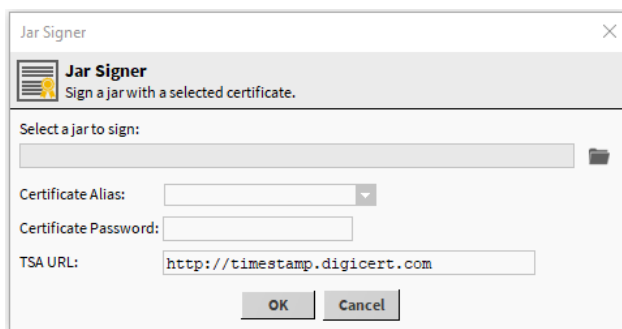
This topic explains how to sign the third party modules using the Jar Signing Tool.

**CAUTION:** A security best practice is to encourage the original authors of a module to sign their own code. Signing a module on behalf of a 3rd party should be done only as a last option, and only if you trust the authors.

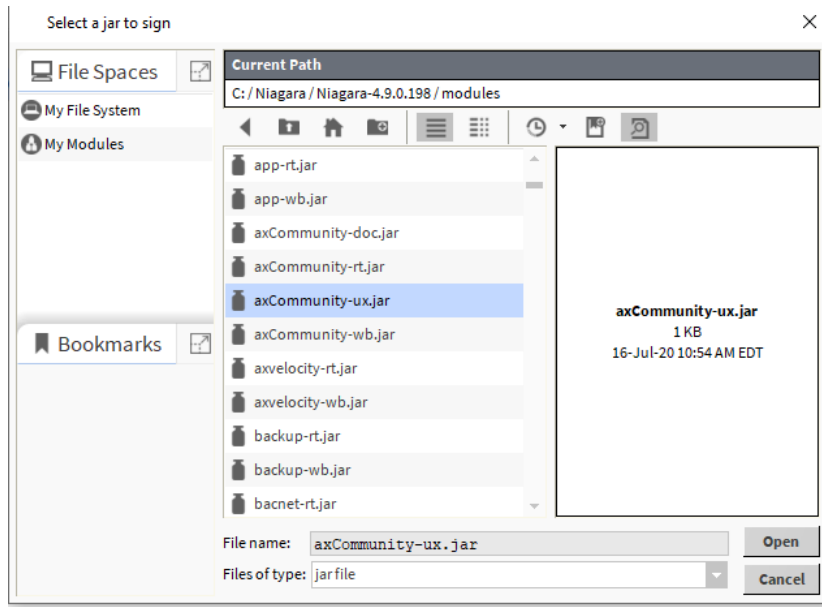
Step 1 In Workbench click **Tools**→**Jar Signer Tool**.



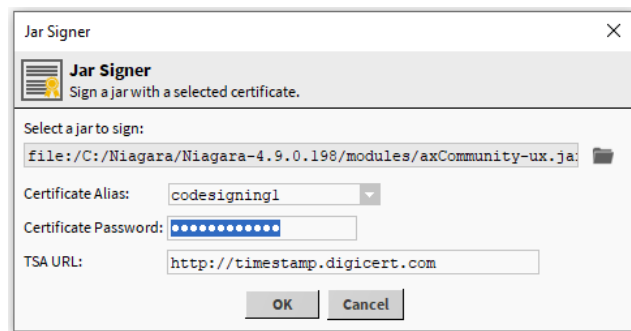
The Jar Signer window opens.



Step 2 Click the Folder icon to navigate to the jar file to be signed, and click open.



Step 3 In the Certificate Alias field, select the specific certificate from the drop-down list and click open.



Step 4 Click **OK**.

Step 5 In the **Save signed jar as** window, navigate to Niagara-4.x.x/modules folder on the system , and click **Save**.

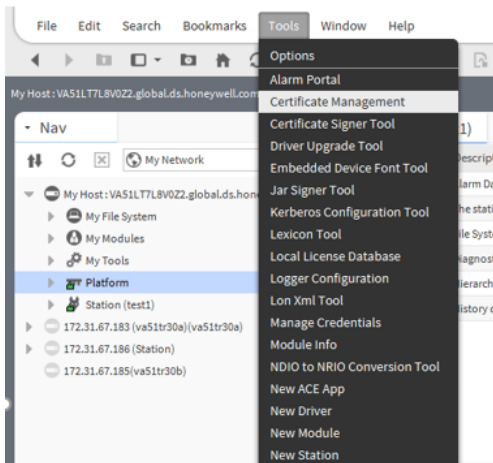
Step 6 A confirmation window appears click **OK** to confirm.

## Importing the self-signed certificate into a platform

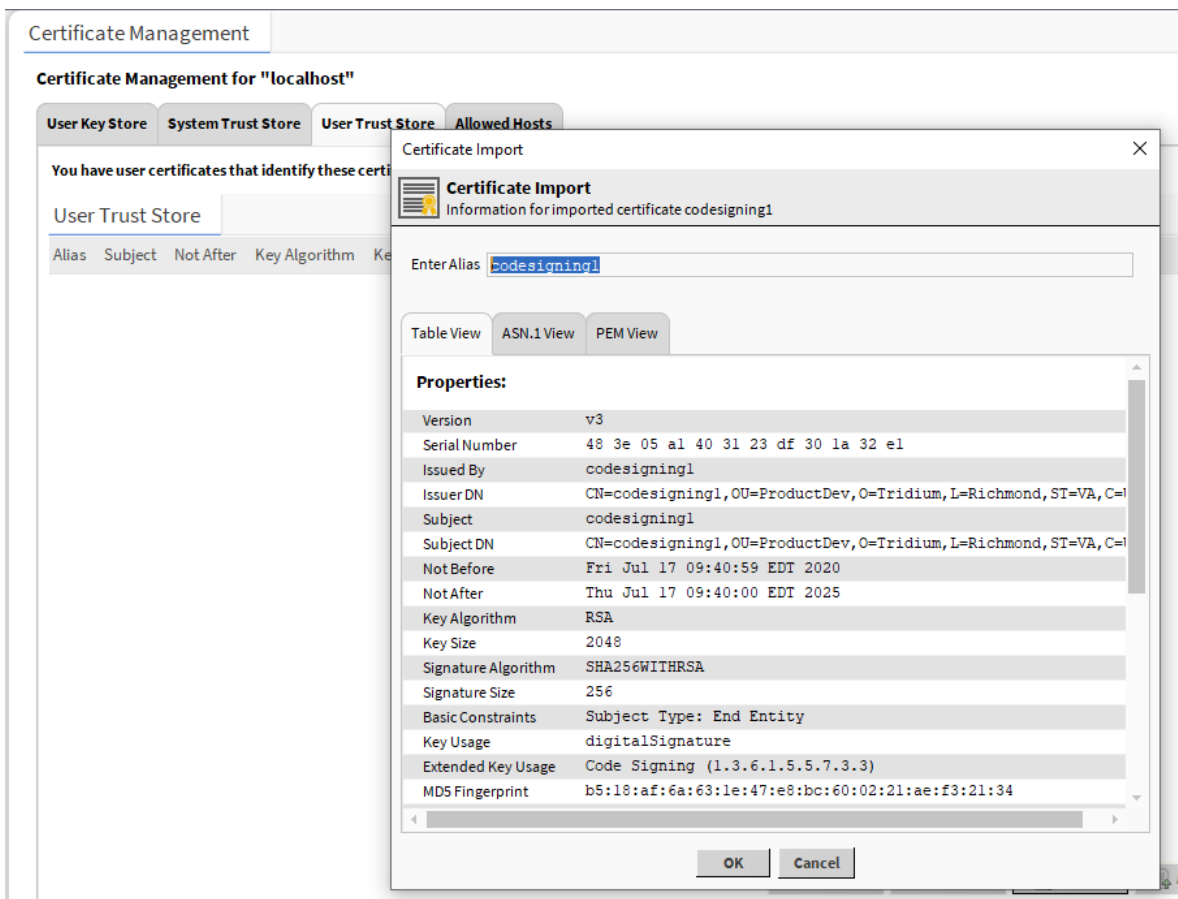
This topic explains how to import a self-signed certificate into the platform where the signed 3rd party modules will be used.

Step 1 Establish a Workbench connection to the platform.

Step 2 Open **Tools**→**Certificate Management** on the platform.



Step 3 Click the **User Trust Store** tab, click import, and import the previously created self-signed certificate..



Once the certificate is imported you should be able to use the signed 3rd party modules on this platform.

# Chapter 3 Common tasks

## Topics covered in this chapter

- ◆ Checking module signatures on a remote platform
- ◆ Checking module signatures on your local platform
- ◆ Installing modules with signature warnings
- ◆ Obtaining properly signed modules

Procedures related to third party module signing include using Software Manager to check module signatures on a remote platform, using Module View to check module signatures on your local machine, installing signed modules, and information on obtaining properly signed modules.

## Checking module signatures on a remote platform

You can easily check for the status of the module signatures for a remote platform by opening the **Software Manager** view of the platform. In Niagara 4.8 and later, the installed version and available version columns contain a shield icon indicating module signature status by color.

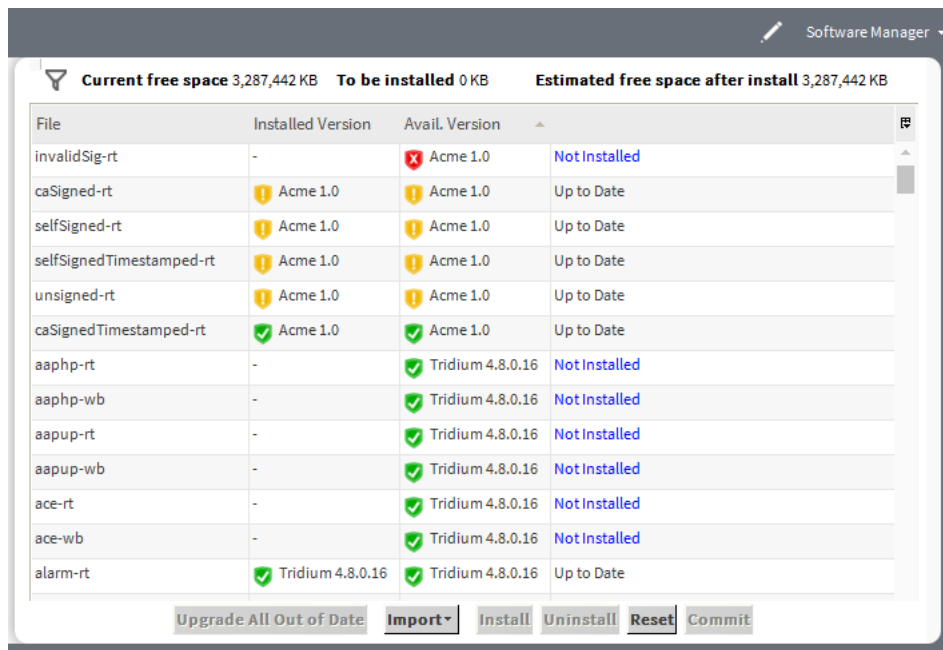
### Prerequisites:

- The remote platform is running Niagara 4.8.

Step 1 Open a connection to the remote platform.

Step 2 In the Nav Tree (or Nav Container View) double-click **Software Manager**.

The **Software Manager** view opens, displaying module details similar to those shown in this example.



Software Manager

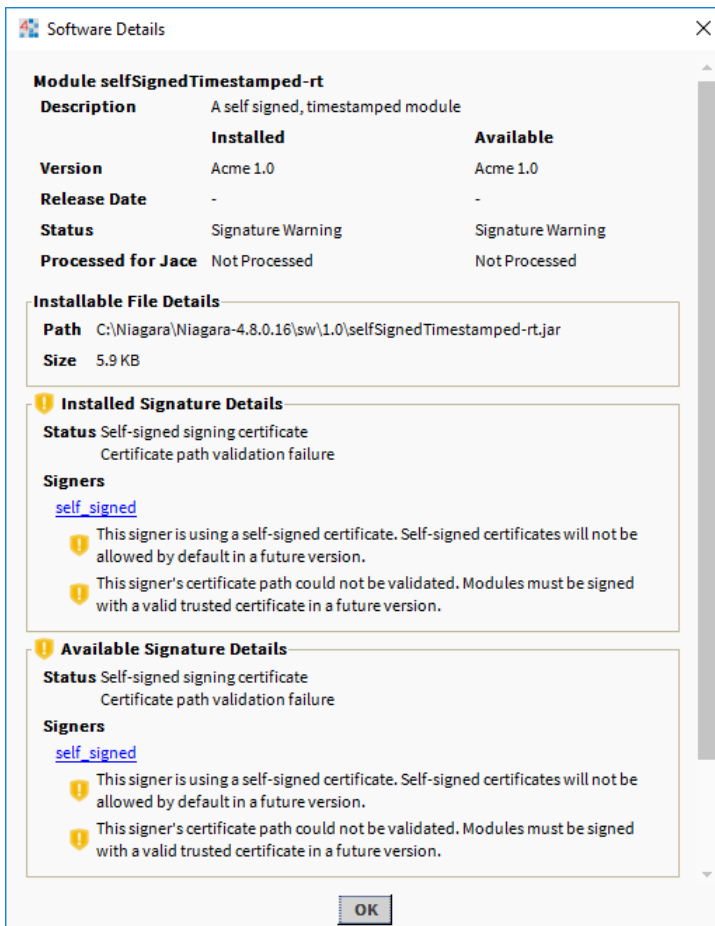
Current free space 3,287,442 KB To be installed 0 KB Estimated free space after install 3,287,442 KB

File	Installed Version	Avail. Version	
invalidSig-rt	-	Acme 1.0	Not Installed
caSigned-rt	Acme 1.0	Acme 1.0	Up to Date
selfSigned-rt	Acme 1.0	Acme 1.0	Up to Date
selfSignedTimestamped-rt	Acme 1.0	Acme 1.0	Up to Date
unsigned-rt	Acme 1.0	Acme 1.0	Up to Date
caSignedTimestamped-rt	Acme 1.0	Acme 1.0	Up to Date
aapbp-rt	-	Tridium 4.8.0.16	Not Installed
aapbp-wb	-	Tridium 4.8.0.16	Not Installed
aapup-rt	-	Tridium 4.8.0.16	Not Installed
aapup-wb	-	Tridium 4.8.0.16	Not Installed
ace-rt	-	Tridium 4.8.0.16	Not Installed
ace-wb	-	Tridium 4.8.0.16	Not Installed
alarm-rt	Tridium 4.8.0.16	Tridium 4.8.0.16	Up to Date

Upgrade All Out of Date Import Install Uninstall Reset Commit

**NOTE:** A red icon (❌) there are errors that are not acceptable for the current module verification mode. A yellow icon (⚠️) there are warnings, but they are acceptable for the current module verification mode. A green icon (✅) indicates that the module is signed with a Certificate Authority (CA). For details, see “Signature statuses” in this chapter.

**Step 3** For any module showing the signature warning icon or signature error icon, double-click the row to view details.



The **Software Details** window displays all the signature statuses of the module with additional details. It also provides a link to view the certificate that the module is signed with.

## Checking module signatures on your local platform

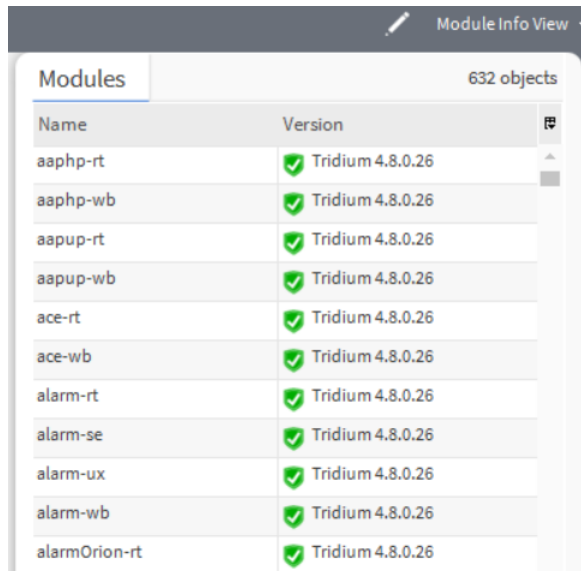
In Niagara 4.8 and later, you can easily check for the status of the module signatures for your local platform using the Workbench tools.

### Prerequisites:

- The local platform is running Niagara 4.8.

**Step 1** In Workbench, click **Tools→Module Info**.

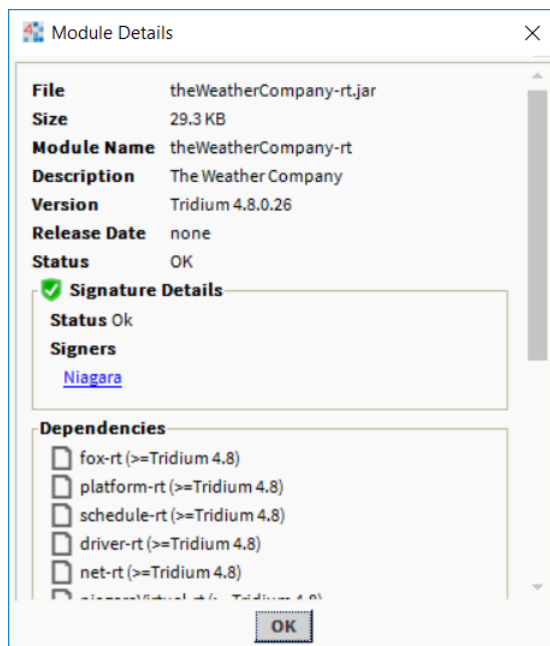
**Step 2** In the **Module Info** view, review the list of currently installed modules and the signature status icons.



Name	Version
aaphp-rt	Tridium 4.8.0.26
aaphp-wb	Tridium 4.8.0.26
aapup-rt	Tridium 4.8.0.26
aapup-wb	Tridium 4.8.0.26
ace-rt	Tridium 4.8.0.26
ace-wb	Tridium 4.8.0.26
alarm-rt	Tridium 4.8.0.26
alarm-se	Tridium 4.8.0.26
alarm-ux	Tridium 4.8.0.26
alarm-wb	Tridium 4.8.0.26
alarmOrion-rt	Tridium 4.8.0.26

**NOTE:** A red icon (❌) there are errors that are not acceptable for the current module verification mode. A yellow icon (⚠️) there are warnings, but they are acceptable for the current module verification mode. A green icon (✅) indicates that the module is signed with a Certificate Authority (CA). For details, see “Signature statuses” in this chapter.

**Step 3** For any module listed, double-click the row to view more details.

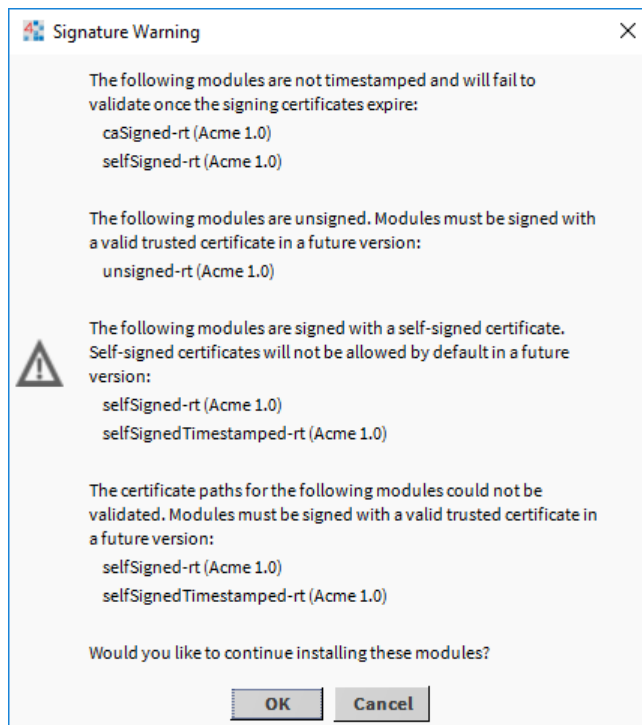


The **Module Details** window displays all the signature statuses of the module with additional details. It also provides a link to view the certificate that the module is signed with.

## Installing modules with signature warnings

When installing modules through any standard tool, such as **Software Manager**, **Commissioning Wizard**, **Station Copier**, or **Provisioning**, any modules with signature warnings will result in a warning dialog.

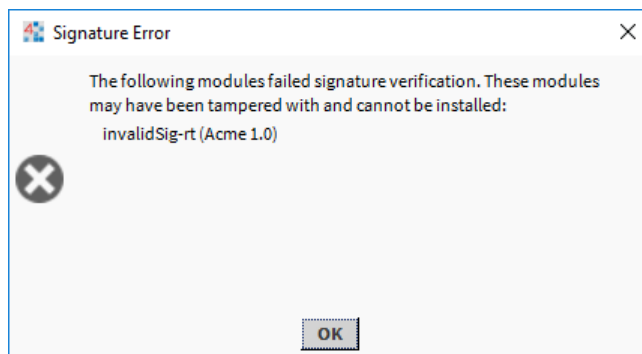
Figure 9 Example Software Manager signature warning dialog



If presented with a warning dialog, click **OK** to accept the warning and continue installing the module(s).

**NOTE:** Any module with signature errors will cause a signature error dialog to display and the installation will fail. Click **OK** to close the window.

Figure 10 Example Software Manager signature error dialog



## Obtaining properly signed modules

To ensure your systems continue to function properly and securely in future versions, you need to make sure all the third party modules you use are properly signed. If you don't develop any modules yourself, you shouldn't have to sign any modules, with a few exceptions, but you will need to make sure all of your module vendors provide you with validly signed modules.

The ideal situation is for your vendors to provide you with modules signed with a public CA (Certification Authority) issued certificate. This will give the module an "OK" signing status and show a green icon (✓) in the **Software Manager** view, and will successfully install in any future version installation without any additional steps.



The next best option is a module that is signed with a certificate issued by an internal CA. This module will still work in all verification modes, but you will have to import the CA certificate into the trust store of every device you install the module on. The vendor can provide you with a certificate \*.pem file to import. A module signed with a self-signed certificate will still work, but for later releases where the verification mode default is changed to “high”, you will have to set your verification mode back to “medium” for the module to work.

There are two cases where non-developers may still have to sign modules. First is if you use program objects or program modules. Second, is if you use any legacy modules that are no longer supported by the vendor. For information on program objects, see “Signing program objects” in *Getting Started with Niagara*, and for details on signing legacy modules, see “Jar Signer Tool” in the same guide.

If you do develop your own modules, signing is incorporated into the build process. For more information, see “Code signing” in the online help version of the *Niagara Developer Guide*.



# Chapter 4 Troubleshooting

## Topics covered in this chapter

- ◆ Signature warnings or errors when installing module
- ◆ Signature warnings in the Application Director
- ◆ Station fails with error message
- ◆ Workbench fails to start after installing a module

Following is a list of typical problems that you may encounter, and tips on how to resolve them.

## Signature warnings or errors when installing module

For cases where signature warnings or errors occur when installing a module, refer to the following list of recommended actions.

- For certificate path validation failures, ensure that the certificate that the module is signed with is valid, and import the certificate or CA certificate into the user trust store of the device you are installing the module to.
- If you do not have the certificate to install, request it from the module's vendor.
- For any other warnings or errors, contact the module's vendor for a solution.
- For a temporary solution, accept warnings or lower verification mode settings to resolve errors.

## Signature warnings in the Application Director

Signature warnings are logged in the **Application Director** when loading modules that have signature warnings.

To resolve warnings or accept them, see listed recommendations in "Signature warnings or errors when installing module".

## Station fails with error message

Troubleshooting recommendations for cases where the station fails with the "Unable to load ..." message in Application Director.

This can result in the following situations.

- If a module is installed via non-standard tools.
- The certificate that the module is signed with is removed from the trust store.
- The certificate expired and the module is not time-stamped.

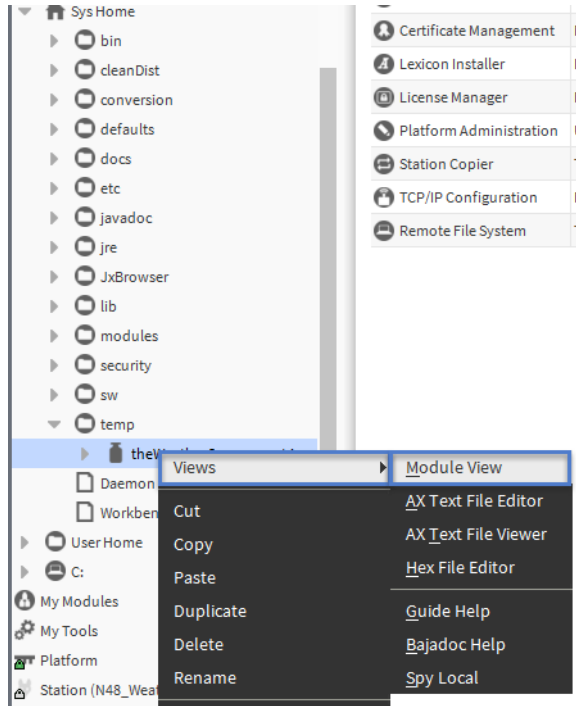
To resolve this, open the **Software Manager** which lists any modules with signature errors, describing the problem. To resolve the errors, see listed recommendations in "Signature warnings or errors when installing module"..

## Workbench fails to start after installing a module

Troubleshooting recommendations for the case where Workbench fails to start after installing one or more modules.

This situation can occur if the module is failing verification. Follow the steps below to resolve the error.

1. Remove the recently installed module, which is probably causing the problem.
2. Workbench should be able to start after removing the module, but run it from a Console window so that if the problem persists it will generate more debug details.
3. In the Workbench Nav Tree, navigate to wherever you have the module stored in the file system (outside of the !modules directory) and right-click on the module and click **Views→ModuleView**. In the **Module View**, check the module's signatures and resolve any errors. For additional details, see listed recommendations in "Signature warnings or errors when installing module".



# Index

## C

changes to tools .....	8
checking module signatures	
local platform .....	22
remote platform .....	21
code signing .....	15

## D

document change log .....	5
---------------------------	---

## I

installing modules	
signature warnings .....	23

## J

Jar Signing Tool	
using .....	17

## M

module signatures .....	21–22
module signing	
third party module .....	15

## O

obtaining signed modules .....	24
overview .....	7

## R

related documentation .....	5
-----------------------------	---

## S

self-signed certificate	
creating .....	15
importing to a platform .....	19
signature statuses .....	8, 14
staged roll-out .....	7

## T

tools	
Jar Signer Tool .....	8
Module Info View .....	8
troubleshooting .....	27

signature errors .....	27
signature warnings .....	27
signature warnings in Application Director .....	27
station fails .....	27
unable to load message .....	27
Workbench fails to start .....	27

## V

verification modes .....	7
--------------------------	---