Technical Document

Niagara Station Security Guide

June 19, 2023

niagara4

Niagara Station Security Guide

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

Confidentiality

Trademark notice

Copyright and patent notice

Contents

About this guide **7**
Document change log 7
Related documents 10

Chapter 1 About station security **11**
Security precautions 11
Security best practices 12
Security Dashboard overview 13
Vulnerability management tools 14

Chapter 2 Secure communication **17**
Client/server relationships 17
Certificates 18
Self-signed certificates 19
Keys 20
How certificates verify identity 21
Encryption 21
Naming convention 21
Certificate stores 22
Accessing the stores 22
Stores folder structure 23
Stores file names 24
CSR folder structure 24
Creating a CSR folder structure 24
Certificate set up 25
Root and intermediate certificate checklist 25
Supervisor/engineering PC checklist 26
Platform and station checklist 26
Opening a secure platform connection (niagarad) 26
Enabling the NiagaraNetwork 28
Confirming client/server relationships 28
Creating a root CA certificate 28
Creating a Client Certificate for Syslog configuration 33
Creating a server certificate 35
Creating a code-signing certificate 37
Creating a CSR 39
Signing a certificate 39
Importing the signed certificate back into the User Key Store 41
Installing a root certificate in the Windows trust store 41
Exporting a certificate 42
Manually importing a certificate into a User Trust Store 43
Installing a certificate 43
Station health confirmation 44
Viewing session information 44

Allowed hosts management....45
When a certificate expires....46
Deleting a certificate....48
Certificate Wizard....48
Generating a CA certificate and signed Server certificate using the Certificate Wizard....49
Recommended verifications....54
Signing multiple certificates....56
Configuring secure platform communication....59
Configuring secure station communication....59
Enabling clients and configuring them for the correct port....60
Installing a station copy on another platform....60
Securing email....61
Secure communication troubleshooting....62
Default TCP/IP ports....64
Certificate management when replacing a controller....65
Accepting a self-signed certificate after a change....65

Chapter 3 User authentication....67
User authentication checklist....67
Authentication schemes....68
Client certificate authentication setup....69
Admin workflow for client certificate authentication....69
User workflow for client certificate authentication....71
Logging in via a browser using client certificate authentication....76
Logging in via Workbench using client certificate authentication....78
Enabling a kiosk-like mode using ClientCertAuth....79
Setting up Google authentication....79
Single Sign On....80
How SAML SSO works....80
Creating a User Prototype for SAML Authentication....81
Configuring the SAML Authentication Scheme....82
Customizing SAML attribute mapping....83
Logging in with SAML SSO....84
About the SAML IdP Service....85
Users....90
User prototypes....90
Station-to-station users....91
Adding or editing a user....92
Adding roles and permissions....92
Assigning roles to users....95
Assigning authentication schemes to users....95
Password management....95
Setting up password strength....95
Setting up password options....96
Setting up a user's password....96
Logging on to a station....96

Station Auto Logoff 98
Changing your password 98
User authentication troubleshooting 99

Chapter 4 Authorization management 101
Component permissions checklist 102
Component permission level 102
Changing a component Config flag 102
UserService permission levels 103
Categories 104
Basic categories 104
Adding and editing a basic category 104
Assigning a component to a basic category 105
Deleting a category name 106
Roles and permissions 107
Adding roles and permissions 107
Adding a component 109
Editing roles and permissions 110
Assigning roles to users 110
Confirming access security 110
Reviewing permissions 111
Ancestor permissions 111
File permissions 111
History permissions 112
Component permissions troubleshooting 112

Chapter 5 Components, views and windows 113
Components 113
Client Cert Auth Scheme (clientCertAuth-ClientCertAuthScheme) 113
SecurityService (nss-SecurityService) 114
Certificate Folder (nss-CertificateFolder) 116
Certificate Info (nss-CertificateInfo) 116
Certificate Expiry Point (nss-CertificateExpiryPoint) 117
Google authentication scheme (gauth-GoogleAuthenticationScheme) 118
S A M L Attribute Mapper (saml-SAMLAttributeMapper) 118
SAML Authentication Scheme (saml-SAMLAuthenticationScheme) 119
SAML CoT Prototypes (saml-SAMLCoTPrototypesMixIn) 120
S A M L Id P Service (saml-SAMLIdPService) 121
Circle of Trust Editor (saml-CircleOfTrust) 122
Prototype Folder (saml-SAMLUserPrototypeFolder) 124
Station Service Provider Folder (saml-StationServiceProviderFolder) 124
Saml Xml Decrypter (samlEncryption-SamlXmlDecrypter) 124
Saml Xml Encrypter (samlEncryption-SamlXmlEncrypter) 125
Plugins (views) 125

Session Token Manager View (signingService-SessionTokenUxManager) 125
Security Dashboard View (nss-SecurityDashboardView) 126
Workbench Certificate Management (platCrypto-CertManagerTool) 129
Platform Certificate Management (platCrypto-CertManagerService) 129
HTML–5 Certificate Ux Mangement View 135
Certificate Signer Multiple Selection (platCrypto-CertificateSignerMultipleSelectionTool) 136
Certificate Parameter (platCrypto-CertificateParameter) 137
Common Name Template (platCrypto-CommonNameTemplate) 138
Circle Of Trust Editor (saml-CircleOfTrustEditor) 139
Category Browser (wbutil-CategoryBrowser) 140
Category Manager (wbutil-CategoryManager) 141
wbutil-CategorySheet 142
wbutil-PermissionsBrowser 143
Role Manager (wbutil-RoleManager) 144
Windows 145
Certificate Signing window 145
Certificate Export windows 146
Generate Self-Signed Certificate window 148
New category windows 151
New/Edit roles window 152
Platform Authentication window 152
Platform Connect 153
Platform TLS settings 154
Permissions map 155
Session Info window 156
Station Authentication window 157
Station Connect window 157

Glossary 159

Index 161

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 provides guidance and best practices designed to help you configure and maintain secure stations while taking advantage of the benefits that internet connectivity offers.

Document change log

This log provides the date this document was released and lists any subsequent document updates that have occurred.

June 19, 2023

- Added "Email Authenticator" options to "Securing emails" chapter (as of Niagara 4.13).
- Added JACE-9000 reference.
- Added "nss-SecurityDashboardDeviceExt", "nss-ReachableStationSecurityDashboardExt", and "nss-SecurityDashboardDataImport" component topics.
- Updated "Security Dashboard Overview" on certificate data.
- In "Platform TLS settings", added details about updated "Certificate Alias", "Certificate Password", and "Use Global Certificate Password" property.
- In "Creating a server certificate", added details about the unique private key password and global certificate password.
- Added "Creating a Client Certificate for Syslog configuration" topic to Secure Communication chapter.
- Removed some component topics that are present in the *Getting Started with Niagara*.
- Added new topic "HTML–5 Certificate Ux Mangement View" to the "Plugins" chapter.

October 14, 2021

Added property description to **`WebService`** component **`Https Min Protocol`** property regarding addition of TLSv1.3 option.

May 19, 2021

Added Same Site property in web-WebService component topic.

April 29, 2021

In topic "Security best practices" changed the jpeg files to png.

January 26, 2021

Replaced the Screen capture of "Generate Self-Signed Certificate window" and added the Key usage. Updated the "Security Dashboard feature" and "nss-SecurityDashboardView" topics with added details on the **System View** feature.

October 13, 2020

Updated to include details on bulk certificate signing using the Workbench Certificate Signing Multiple Selection tool, available in Niagara 4.10 and later.

July 10, 2020

Added a new component Http Header Providers in web-WebService topic.

June 29, 2020

Added the topic: "Vulnerability management tools"

April 23, 2020

Minor edit to the topic, "Preliminary steps" in the section on setting up SAML IdP Service.

April 1, 2020

Added the procedure, "Installing a station copy on a different platform". Also, edited this document for consistency and clarity.

March 6, 2020

Added the procedure, "Setting up alarming for certificate expiration", in the "Secure communication" chapter. Also added the following component topics to support online help: nss-ExpiryAlarmExt and nss-CertificateInfo.

January 22, 2020

Updated for Niagara 4.9.

- Added new content on the SAML IdP Service in the "Single Sign On" section and in the "Components" section.
- Also, replaced references to "applet" and "WebStart" with "Web Launcher".
- In the topic, "Security precautions", added a caution note alerting customers to restrict access to all computers, devices, field buses, components, etc., that manage their building model.
- Added section on the Certificate Wizard platform tool.

September 25, 2019

In Chapter 2, added content on using the **Certificate Wizard**.

July 25, 2019

Many changes throughout to support the Niagara 4.8 release, including the following edits:

- In Chapter 1, added a section on the "Security Dashboard feature".
- In Chapter 3, edited "Configuring a user for Client Certificate Authentication", and added a new procedure for "Enabling a kiosk-like mode using client certificate authentication".
- In Chapter 4, added a note to the "Reviewing permissions" procedure.
- In Chapter 5, added a component topic on the "nss-SecurityService", and view topic on the "nss-SecurityDashboard". Also added information about the Station Link Config property to the "nss-SecurityService" component topic. Edited the "wbutil-PermissionsBrowser" topic, to add information on improvements to the **Permissions Browser** view.

February 11, 2019

- In Chapter 3, edited the "Authentication Schemes" topic to add information on the Client Certificate Authentication Scheme and the GoogleAuthenticationScheme.
- In Chapter 3, added two sections with associated procedures: the "Admin workflow for client certificate authentication" and "User workflow for client certificate authentication".

- Added these component topics to Chapter 5: “clientCertAuth-ClientCertificateAuthenticationScheme” and “gauth-GoogleAuthenticationScheme”.

November 14, 2018

- Edited the component topic, “saml-SAMLAttributeMapper”, to add information on a recent configuration change to handle multiple values returned from the IdP for the prototypeName attribute.
- Edited the component topic, “saml-SAMLAuthenticationScheme”, to add information on SAML metadata URL which can automatically generate the station's SAML metadata XML.
- Added the component topic, “saml-SamlXmlDecrypter” which you can add to a SAMLAuthentication-Scheme to configure a certificate for decryption.
- Edited component topics, “wbutil-CategoryBrowser” and “wbutil-CategorySheet” to add note on behavior new in Niagara 4.8.
- Minor changes in the procedure, “Customizing SAML attribute mapping".

August 8, 2018

Correction to specified SP metadata in prerequisites for “Configuring the SAML Authentication Scheme” procedure.

May 17, 2018

Added a caution regarding giving admin write permissions on the Role Service to the following topics: Roles and permissions”, “Role Service”, and “Role Manager”.

March 2, 2018

In the “Single Sign On” and “Components” sections, added information on the baja-UserPrototype which is required for SAML authentication; also added the procedure, “Creating a User Prototype for SAML authentication”.

February 15, 2018

Edited the procedure, Configuring the SAML Authentication scheme,” to add information on required SAML SP metadata that must be shared with the SAML IdP. Expanded on information provided in the “saml-SAMLAttributeMapper” component topic, and added a procedure for “Customizing SAML attribute mappings”.

January 24, 2018

Changed the topic title “Auto Logoff" to “Station Auto Logoff” and clarified wording throughout. Also edited property descriptions for Auto Logoff settings in the “baja-UserService” component topic.

November 13, 2017

In the topic, “About station security”, under authorization management list item, deleted a note discussing unsupported tagged categories.

October 12, 2017

- In the User Authentication chapter, edited the “Authentication Schemes” topic; added the “Single Sign On” and “Auto Logoff” topics; and added these procedures: “Configuring the SAML Authentication Scheme” and “Logging in with SSO”.
- In the Components chapter, added the “saml-SAMLAttributeMapper”, and “saml-SAMLAuthenticationScheme” topics; and edited the “baja-SSOConfiguration”, and “baja-UserService” topics.
- Significantly edited the topics in the Secure Communication chapter, rewriting “Creating a server certificate,” adding “Creating a root CA certificate, and “Creating a code-signing certificate.”
- Added “Provisioning a job to install a certificate” to the same chapter.
- Added references to code-signing certificates through the chapter.

- Added "Certificate Export windows" to the Components, views and windows chapter.

September 20, 2017

- Added the topic "When a certificate expires" to the "Certificate Setup" chapter.
- Updated multiple topics in the "Certificate Setup" chapter to include the code-signing certificate.

September 14, 2017

Updated the WebService property description in web-WebService topic.

September 13, 2017

Updates to WebService properties and descriptions in the web-WebService component topic

August 31, 2017

The following list of modifications are included in this update:

- In the topic baja-UserService, added the description about "Effect of property changes on user session"
- In the topic "Configuring Secure Platform Communication" and "Platform TLS Setting" modified the description for Platform TLS setting window.
- Created new topic WebService cacheControl under the chapter Components, views, and windows.
- Restored Network User content (formerly found only in legacy documentation) and updated that content to reflect user synchronization feature support currently in Niagara 4.
- Added baja-AuthenticationService components to the Components section.
- Revised the Preface section to remove content which now makes up the chapter, "About station security."
- In the User Authentication chapter, updated several topics to update the name of the LegacyDigestScheme which changed to AXDigestScheme in Niagara 4.

July 13, 2016

Updated to support rebranding (minor changes throughout)

November 6, 2015

Updates to WebService properties description in web-WebService component topic

August 23, 2015

Initial release document

Related documents

Following is a list of related guides.

- *Niagara Drivers Guide*
- *Niagara Platform Guide*
- *Niagara 4 Hardening Guide* (available on *https://www.tridium.com/us/en/services-support/library*)

Chapter 1 About station security

Topics covered in this chapter

♦ Security precautions
♦ Security best practices
♦ Security Dashboard overview
♦ Vulnerability management tools

Security begins with the way you configure and monitor each station. It involves setting up secure communication, secure email, secure user credentials, and configuring components, categories, hierarchies, and roles to grant users access only to the system objects they need to do their jobs. Ultimately, your system will only be secure if you take full advantage of Niagara 4's security features, and if you configure your network effectively. Although the defaults are designed to be as secure as possible, your system will remain vulnerable if you rely solely on factory defaults. The aspects of station security that require configuration are settings for secure communication, user authentication and authorization management.

- *Secure communication* provides:
  - Server identity verification, which prevents man-in-the-middle and spoofing attacks. To set up the digital certificates that verify server identity. you use the **Certificate Manager** view.
  - Data encryption (foxs/https/platformtls), which prevents eavesdropping during the actual transmission of data. You define the key size used to encrypt data transmission when you create each certificate.
  - Secure email communication. To configure email security, you use the **`EmailService`**.
- *User authentication* protects against malicious access by ensuring that only legitimate users (human or station) can log in using Workbench or a web browser. You use the **`AuthenticationService`** to activate the authentication schemes the station needs, and the **`UserService`** to assign the authentication scheme and login credentials to individual users (human or another station). You can add multiple schemes, each of which may be used by a different user.
- *Authorization management* involves defining which component slots, files and histories are accessible, which users may modify them, and what modifications users may make Niagara uses role-based access control, where users are assigned roles that are mapped to component permissions. You use the **`CategoryService`** to set up component categories (groups of components), the **`RoleService`** to assign permissions, and the **`UserService`** to assign roles to users.

NOTE: Platform security is beyond the scope of this document.

Security precautions

Whether you are protecting assets in a single building or in a large, multi-site application, station security is critical. The practical implementation of a secure device network relies on basic common sense.

Do not connect any station directly to the Internet. If you need remote access, use a VPN (Virtual Private Network) solution where your devices are protected behind a fire wall, but remotely accessible. Your VPN solution should incorporate RSA (Rivest-Shamir-Adleman) two-factor authentication.

Do not share accounts (log in using someone else's credentials). Always log in as yourself.

Do not create a certificate (and key pair) on a local computer and download the certificate into the **User Key Store** of each remote controller. Each host requires its own unique certificate, public and private keys, which should be generated by the controller and should reside only in the

controller or on a backup medium that is physically protected. Transmitting a certificate with its private key exposes the key to the risk of capture during transmission.

Do not commission a remote controller over the Internet. If it becomes necessary to replace a controller, physically travel to the location, take the controller off the network, connect a cross-over cable, and import the backed-up stores. While the Key and Trust Stores are backed up with the station, they are not part of a station copy.

Do not mix secure platforms with platforms that are not secure on the same network. All controllers and Supervisor stations must be secure.

Do not use self-signed certificates. In a CA-signed certificate, the **`Issued By`** property is not the same as the **`Subject`**.

Do not use guest accounts. They are easy to hack.

Do not use default passwords or passwords that can be easily guessed by attackers, such as birth dates, short words, and real words. Use different passwords for each entity. For example, use different usernames and passwords for your system password, platform credentials and station credentials. Implement strong passwords and change them frequently. Store and use passwords securely, strictly controlling access to file systems.

Do not rely on an NTP (Network Time Protocol) server that you do not directly control. If your Niagara network depends on an external NTP server for the time of day, and that server is compromised or spoofed, your Niagara system may be harmed. For example, locks may be turned off, the alarm system disabled, etc. If you use an NTP server, it must be an internal server that is physically controlled by your trusted organization.

Be warned. If your Niagara system is dependent on an external weather service, and if that weather service is compromised or spoofed, any logic in your system that uses the temperature for heating or cooling, or any other purpose may be harmed.

CAUTION: Protect against unauthorized access by restricting physical access to the computers and devices that manage your building model. Set up user authentication with strong passwords, and secure components by controlling permissions. Failure to observe these recommended precautions could expose your network systems to unauthorized access and tampering.

Security best practices

In today's world, ensuring the security of your device network is extremely important. While managing digital certificates and passwords may seem like an excessive burden, the cost of the alternative is so substantial that you must assign resources and take the time to implement the best practices covered by this topic.

Always upgrade your platform and station to the latest software version. Install all patches and software updates.

Physical security is crucial. Secure all computer equipment in a locked room. Make sure that each station is only accessible by authorized users.

Physically protect wiring to prevent an unauthorized person from plugging in to your network.

Use digital certificates to secure data transmission over wires or wireless connections. If you must connect a host station directly to the public Internet, make sure you are using CA-signed certificates.

If your company is acting as its own CA (Certificate Authority), your root CA certificate must be separately installed in each station's **User Trust Store** and each browser.

Physically protect the medium (usually a USB thumb drive) you use to back up and transport exported certificates.

Install browsers using only a trusted installation program. The program you use installs third-party certificates from CAs, such as VeriSign and Thawte. These must be trustworthy certificates.

For high-traffic stations (especially stations that provide public access to a controller network), secure **`niagarad`** with a separate certificate from that used for your **`FoxService`** and **`WebService`**.

Back up each station regularly. Embedded systems, such as JACE controllers write audit information to a rolling buffer. To avoid losing a station's audit trail, regularly export audit histories to a Supervisor station.

Security Dashboard overview

In Niagara, the Security Dashboard feature provides (for admin and other authorized users) a bird's eye view of the security configuration of your station. This allows you to easily monitor the security configuration in many station services, and identify any security configuration weaknesses on the station.

CAUTION: The **Security Dashboard View** may not display every possible security setting, and should not be considered as a guarantee that everything is configured securely. In particular, third party modules may have security settings that do not register to the dashboard.

The **Security Dashboard** view is the default view on the station's **SecurityService**. It alerts you to security weaknesses such as the following:

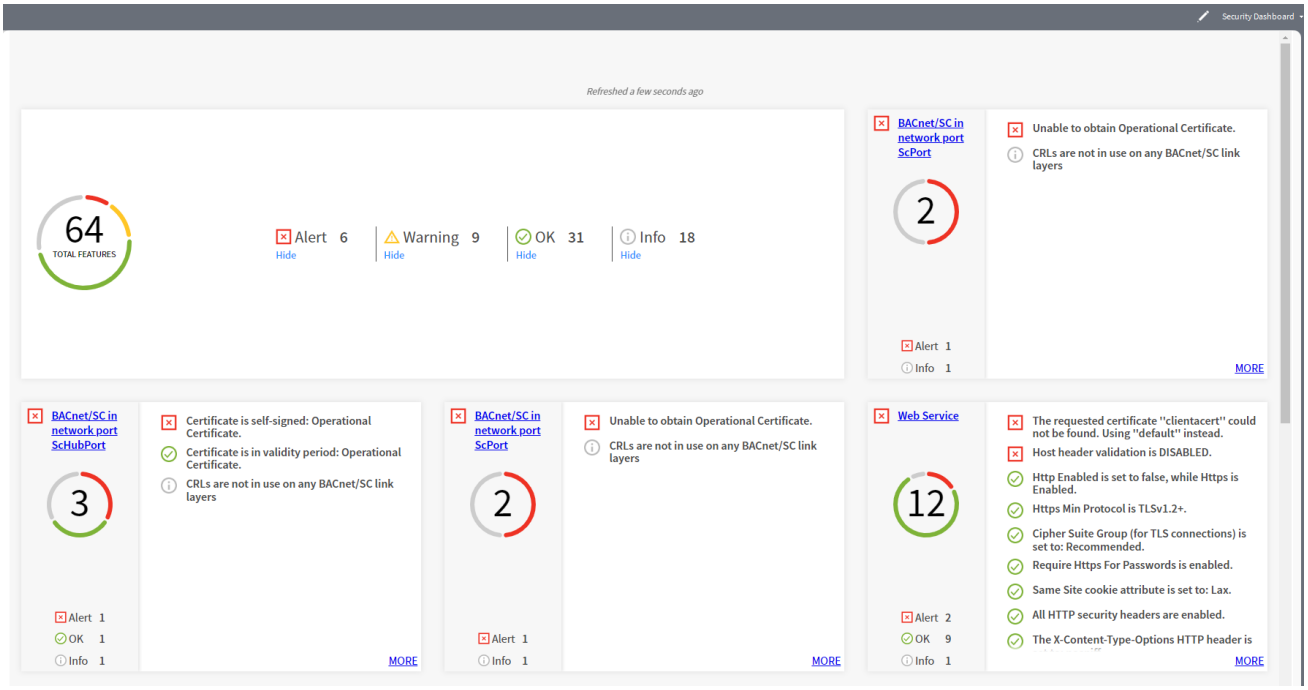
- poor password strength settings
- expired, self-signed, unobtained or invalid certificates
- unencrypted transport protocols indicating areas where the configuration should be more secure

Other reported data include:

- system health
- information on certificate usage and associated security warnings, for example, certificates that are not encrypted with a unique password
- number of active accounts
- inactive accounts
- number of accounts with super-user permissions

Optionally, you may set the **`System`** attribute on the **Security Dashboard** license feature to `true`. This setting enables the **System View** of the station that provides security details for each subordinate station in the **NiagaraNetwork**, as shown here:

Figure 1 Example Security Dashboard View



The summary pane in the upper left corner of the view summarizes the number and type of security status messages that each station service generates. Each pane in the view provides security configuration data for a particular service. Typically, this includes a hyperlink to the service (or to a component) so that you can easily change a configuration. In cases where there is no component to link to, the pane provides no hyperlink.

Vulnerability management tools

Increasingly, vulnerability management tools are used to search computers, networks and applications for potential security breaches. Niagara 4.9 introduced a number of changes intended to allow a Niagara-based hardware platform to appropriately respond to the scanning utilities while continuing operation.

These tools can cause Niagara platforms, such as a JACE-8000 or JACE-9000 or Edge 10, to become unresponsive or to reboot via an Engine Watchdog Timeout. This behavior is not acceptable for the critical applications that Niagara facilitates.

Beginning with Niagara 4.9, the platform daemon responds in these ways to the scanning utilities:

- **Recognition of non-Niagara traffic on the platform**: The daemon recognizes non-Niagara traffic on the platform over a period of time, shuts down the connection, if necessary, and waits for a pre-determined amount of time before connecting again.

  The scanner may report a denial of service. In fact, Niagara disables the communication mechanism by which the scanner attempts its interrogation. This affects normal platform communication, however, the platform and station continue to run.

- **Prioritization of internal vs. external communication on the station**: Niagara prioritizes internal vs. external (scanner) communication on the station. An interrogation from a scanner may cause an Engine Watchdog Timeout, which stops and restarts the station's **WebService**.

  The scanner may report that the Niagara instance abruptly stopped communicating, and may have encountered a denial of service. Normal client web connections to the station are affected; however, the platform and station continue to run.

Vulnerability management tools continue to evolve to protect against threats. As a best practice, the scanning of building automation systems should be scheduled during down time. Any findings observed during down time are just as legitimate as those reported during normal operation.

It may be prudent to work with the scanning tools to configure an appropriate scan priority. The intensity with which you scan a production multicore, failover redundant webserver host, is likely not the best choice for scanning a single core controller or other embedded device.

Chapter 2 Secure communication

Topics covered in this chapter

- ♦ Client/server relationships
- ♦ Certificates
- ♦ Certificate stores
- ♦ CSR folder structure
- ♦ Certificate set up
- ♦ Certificate Wizard
- ♦ Signing multiple certificates
- ♦ Configuring secure platform communication
- ♦ Configuring secure station communication
- ♦ Enabling clients and configuring them for the correct port
- ♦ Installing a station copy on another platform
- ♦ Securing email
- ♦ Secure communication troubleshooting

A Public Key Infrastructure (PKI) supports the distribution and identification of public encryption keys used to protect the exchange of data over networks, such as the Internet. PKI verifies the identity of the other party and encodes the actual data transmission. Identity verification provides non-repudiated assurance of the identity of the server. Encryption provides confidentiality during network transmission. Requiring signed code modules ensures that only expected code runs in the system.

To provide secure networks using PKI, Niagara supports the TLS (Transport Layer Security) protocol, versions 1.0, 1.1, 1.2 and 1.3. TLS replaces its predecessor, SSL (Secure Sockets Layer).

Each Niagara installation automatically creates a default certificate, which allows the connection to be encrypted immediately. However, these certificates generate warnings in the browser and Workbench and are generally not suitable for end users. Creating and signing custom digital certificates allows a seamless use of TLS in the browser, and provides both encryption as well as server authentication.

Beyond communication security, each module of computer code that runs in the system is protected with a digital signature. Added program objects require this signature or they do not run.

NOTE: Verifying the server, encrypting the transmission and ensuring that only signed code runs do not secure data stored on a storage device. You still need to restrict physical access to the computers and controllers that manage your building model, set up user authentication with strong passwords, and secure components by controlling permissions.

Niagara supports and uses secure communication and signed code by default. You do not need to purchase an additional license.

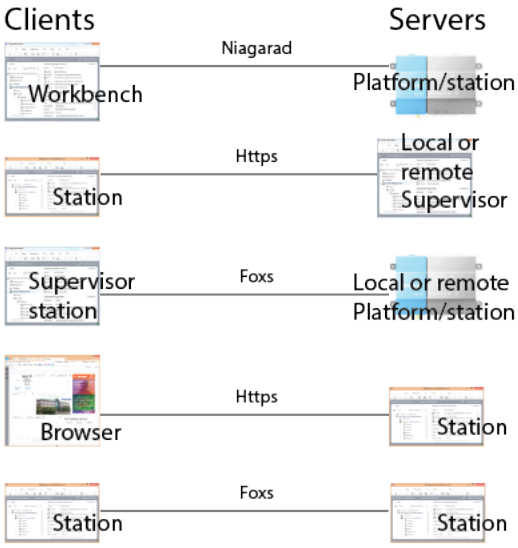
Security is an ongoing concern. While you will find much valuable information in the secure communication topics, expect future updates and changes.

Client/server relationships

Client/server relationships identify the connections that require protection.Workbench client/server relationships vary depending on how you configure and use a system.

Workbench is always a client. A platform is always a server. A station may be a client and a server.

Figure 2 Relationships



The system protocols that manage communications between these entities are:

- Platform connections from Workbench (client) to controller or Supervisor PC platform daemon (server) use niagarad. A secure platform connection is sometimes referred to as platformtls. You enable platformtls using the Platform Administration view.
- Local station connections (Supervisor and platform) use Foxs. You enable these connections in a station's FoxService (**Config→Services→FoxService**).
- Browser connections use Https, as well as Foxs if you are using Web Launcher with a WbWebProfile. You enable these connections using the station's WebService (**Config→Services→WebService**).
- Client connections to the station's email server, if applicable. You enable secure email using the station's EmailService (**Config→Services→EmailService**).

These relationships determine an entity's certificate requirements. For example, a station requires a signed server certificate, which it uses when it functions as a server, and a copy of the root CA certificate, which it uses when it functions as a client. Setting up digital certificates for identity verification involves creating separate certificates to verify the identity of each server. Each server's unique certificate, signed by a CA (Certificate Authority), resides in its **User Key Store**. Each client requires the root CA certificate used to sign each server certificate. The root CA certificate resides in the platform/station **System** or **User Trust Store**.

Certificates

A certificate is an electronic document that uses a digital signature to bind a *public key* with a person or organization. Certificates may serve a variety of purposes depending on how you configure the certificate's **`Key Usage`** property. Their primary purpose in this system is to verify the identity of a server so that communication can be trusted.

Niagara supports these types of certificates:

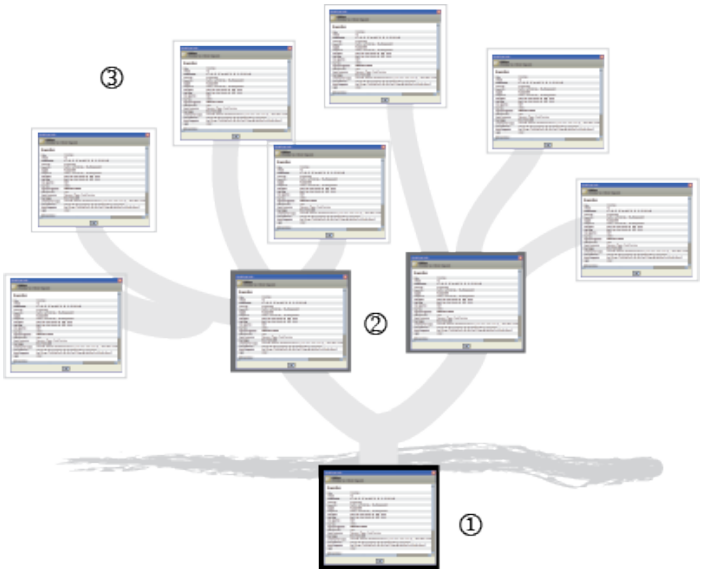
- A *CA (Certificate Authority) certificate* is a self-signed certificate that belongs to a CA. This could be a third party or a company serving as its own CA.
- A *root CA certificate* is a self-signed CA certificate whose private key is used to sign other certificates creating a trusted certificate tree. With its private key, a root CA certificate may be exported, stored on a USB thumb drive in a vault, and brought out only when certificates need to be signed. A root CA certificate's private key requires the creation of a password on export and the provision of the same password when you use it to sign other certificates.

- An *intermediate certificate* is a CA certificate signed by a root CA certificate that is used to sign server certificates or other intermediate CA certificates. Using intermediate certificates isolates a group of server certificates.
- A *server certificate* represents the server-side of a secure connection.

  While you may set up a separate certificate for each protocol (Foxs, Https, Webs). While you may configure a platform and station (as server) with separate server certificates, for simplicity most systems usually use the same server certificate.
- A *code-signing* certificate is a certificate used to sign program objects and modules. Systems integrators use this certificate to prevent the introduction of malicious code when they customize the framework.

Identity verification uses multiple certificates in a trusted *certificate tree* Setting up identity verification may involve a third-party CA (Certificate Authority) or you may decide to serve as your own CA.

Figure 3 Visualizing the certificate tree



In the illustration above:

1. Below the ground is the root CA certificate.
2. The major branches represent intermediate certificates.
3. The leaves are server certificates.

How many certificates you need depends on your configuration. At a minimum you need a unique server certificate for each server (controller) and a single root CA certificate to sign your server certificates. If your company is large, you may need an intermediate certificate for each geographical division or location. An individual server may have multiple certificates: one each to secure its Fox, Http and niagarad (platformtls) connections. Although each platform and station usually share the same certificate, you may create a separate platform certificate and a different station certificate.

If your network is large and getting thousands of certificates signed would be difficult, you may sign a wildcard certificate. Instead of identifying a specific IP or domain (for example, server1.domain.com), a wildcard certificate uses *.domain.com.

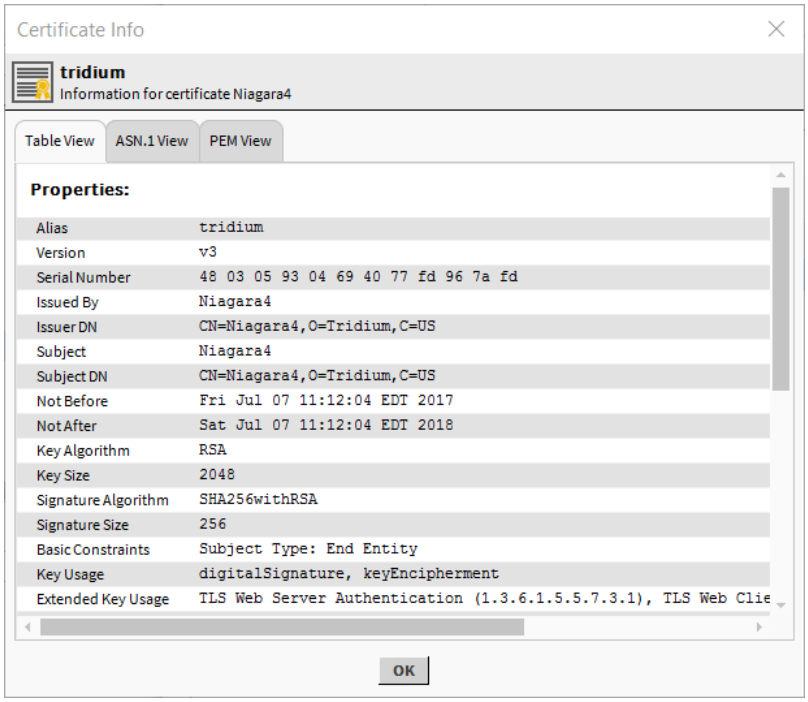
Self-signed certificates

A self-signed certificate is one that is signed by default using its own private key rather than by the private key of a root CA (Certificate Authority) certificate.

The system supports two types of self-signed certificates:

- A *root CA certificate* is implicitly trusted because there is no higher authority than the CA (Certificate Authority) that owns this certificate. For this reason, CAs, whose business it is to endorse other people's certificates, closely guard their root CA certificate(s) and private keys. Likewise, if your company is serving as its own CA, you should closely guard the root CA certificate you use to sign other certificates.
- *A default, self-signed certificate*: The first time you start an instance of Workbench, a platform or a station after installation (commissioning), the system creates a default, self-signed server certificate with the alias of tridium.

Figure 4 Self-signed certificate



Since the `Issuer DN` (Distinguished Name) and `Subject DN` are the same, the certificate is said to be self-signed using its own 2048–bit, private key. The purpose of a self-signed certificate is to allow secure access to the platform and station before a trusted certificate tree with signed server certificates is established. Since a client cannot validate this type of certificate, it is not recommended for robust, long-term security.

When presented with a self-signed certificate, always confirm that it is the expected certificate before you manually approve its use. Once approved, you do not have to approve the certificate each time you make a connection to the server.

NOTE: Do not export this certificate and import it into any store of another platform or station. Although possible, doing so decreases security and increases vulnerability.

To minimize the risk of a man-in-the-middle attack when using self-signed certificates, all your platforms should be contained in a secure private network, off line, and without public access from the Internet.

CAUTION: To use self-signed certificates, before you access the platform or station from Workbench for the first time, make sure that your PC and the platform are not on any corporate network or the Internet. Once disconnected, connect the PC directly to the platform, open the platform from Workbench, and approve its self-signed certificate. Only then should you reconnect the platform to a corporate network.

Keys

A pair of asymmetric *keys* (one public and the other private) makes server verification and encryption possible. The term "asymmetric" means that the two keys are different, but related. The system can use the private key to read messages encrypted with the public key and vice versa.

The signing of certificates with the private key is required to verify authenticity. Both keys are required to encrypt information. In advance, key generation software running on remote controller or station generates this pair of asymmetric keys.

- A *public key* is a string of bytes included in the certificate. This key resides in the server's **System** or **User Trust Store** and is used to identify the authenticity of the connecting client certificate.
- A *private key* is also a string of bytes that resides on the server. The root CA certificate's private key must be physically protected for a certificate tree to remain secure. A private key must not be sent via email, and, if necessary, should be physically transported (on a thumb drive or other medium that is not connected to the Internet).

How certificates verify identity

Once you set up a certificate tree, identity verification takes place during the client/server handshake, before transmission begins and before the system authenticates each user by prompting for credentials (user name and password).

This is how digital certificates verify identity:

1. A unique server certificate resides with its public and private keys in the **User Key Store** of each server (platform/station and Supervisor).
2. When a client connects to a server, the server sends its certificate to the client.
3. The client station validates the server certificate against a root CA certificate in its **System** or **User Trust Store** by matching keys, ensuring that the `Subject` of the root CA certificate is the same as the `Issuer` of a server certificate, and confirming other factors. A client browser does the same. Each browser has a trust store of root CA certificates.
4. If the server certificate is valid, the system establishes a trusted connection between the server and client, and encrypted communication begins. If the certificate is not valid, the station or browser notifies the client and communication does not begin.
5. You may choose to approve a rejected certificate if you know that, although unsigned, it can be trusted.

   NOTE: Always verify the issuer name on any certificate presented by the system as untrusted. Do not approve a certificate from an entity that you do not recognize.

Encryption

Encryption is the process of encoding data transmission so that it cannot be read by untrusted third parties. TLS uses encryption to transmit data between the client and server. While it is possible to make an unencrypted connection using only the fox or http protocols, you are strongly encouraged not to pursue this option. Without encryption, your communications are potentially subject to an attack. Always accept the default Foxs or Https connections.

The following summarizes how encryption works:

1. At the start of a TLS session, the system encrypts the server/client handshake using the client and server certificates' key pairs.
2. During the handshake the system verifies server identity and negotiates encryption keys.
3. Once communication is established, identity verification is no longer needed, and encrypted data transmission begins using the negotiated keys.

Key size is directly related to encryption security. The larger (more complex) the key, the more secure the data transmission. Large keys do not slow encryption, but they do take longer to initially generate.

Naming convention

The **User Key Store**, **User Trust Store**, and **System Trust Store** form the heart of the configuration. Certificates look a lot alike, and the various default self-signed certificates are named identically. While developing

a naming convention is not a requirement (the system will function just fine if the certificates are called “cert1,” “cert2,” etc.), a consistent naming scheme can make the process much easier to follow.

Consider using the **`Alias`** of each certificate to identify the certificate’s purpose. Certificate aliases might include:

- The words "root," "intermediate," “server,” “client,” “code-signing”
- The geographic location of the remote controllers protected by the certificate
- The host name of the server
- The IP address of the server

Certificate stores

Certificate management uses four stores to manage certificates: a **User Key Store**, **System Trust Store**, **User Trust Store** and **Allowed Hosts** list.

The **User Key Store** is associated with the server side of the client-server relationship. This store holds certificates, each with its public and private keys. In addition, this store contains the self-signed certificate initially created when you launched Workbench or booted the platform for the first time.

The **User** and **System Trust Stores** are associated with the client side of the client-server relationship. The **System Trust Store** comes pre-populated with standard public certificates: root CA certificates from well-known Certificate Authorities, such as VeriSign, Thawte and Digicert. The **User Trust Store** holds root CA and intermediate certificates for companies who serve as their own certificate authority.

The **Allowed Hosts** list contains server certificate(s) for which no trusted root CA certificate exists in the client’s **System** or **User Trust Stores**, but the server certificates have been approved for use anyway. This includes servers for which the host name of the server is not the same as the `Common Name` in the server certificate. You approve the use of these certificates on an individual basis. While communication is secure, it is better to use signed server certificates.

Accessing the stores

The system supports two sets of stores: a set for Workbench, and another shared set for each platform and station. Workbench provides access to each set of stores.

Step 1 Launch Workbench or Workbench in the browser.

Step 2 Open a supervisor or remote platform/station.

Step 3 Do one of the following:

a. To access the Workbench stores, click **Tools→Certificate Management**.

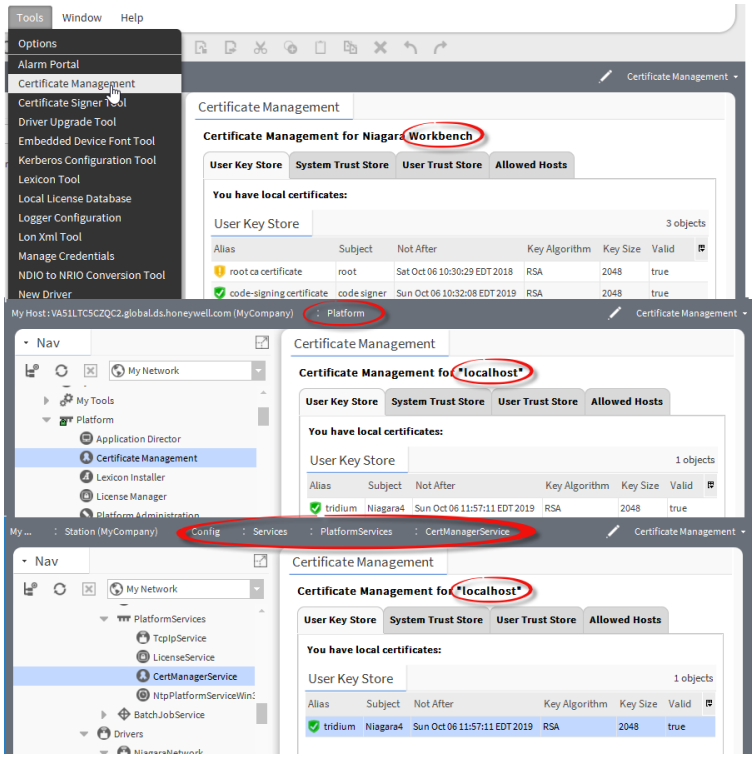
b. To access the platform/station stores, expand **Platform** and double-click **Certificate Management** in the Nav tree. To access the stores this way, the station must be idle.

Since a platform/station may function at different times as a client or a server, it must have a server certificate in its shared **User Key Store** and a trusted root CA certificate in its **System** or **User Trust Store**.

c. If your station is running, you can access the platform/station stores by expanding **Station→Config→Services→PlatformServices** and double-clicking **CertManagerService** in the Nav tree.

The station shares the root CA certificate with the platform, but may have its own server certificate (when it functions as a server) located in the same **User Key Store**.

Step 4 To confirm that you have accessed the correct stores, check the name in the screen title and above the stores.



The platform and station stores are actually the same stores.

Stores folder structure

The Workbench and platform/station stores reside in separate locations on a Supervisor and remote platform.

The following table lists the default user homes that contain the stores, which are considered configuration files. If the folder paths have been changed, these locations no longer apply. `username` is the Windows user name of the person starting the Workbench application.

Table 1 Default user homes

Stores	Nav tree node	Default folder path
Workbench[1]	**My Host→My File System→User Home→security**	`C:\Users\username\Niagara4.x\security`
Supervisor or engineering workstation[2]	**Platform→RemoteFileSystem→User Home (Read Only)→security**	`ProgramData\Niagara4.x\security`
remote controller[3]	**Platform→RemoteFileSystem→User Home (Read Only)→security**	`\home\niagara\security`

1. Each Workbench user has their own User Home.
2. These platform and station stores (same stores) are in the Platform daemon User Home of the Supervisor or any engineering workstation.
3. These platform and station stores (same stores) are in the Platform daemon User Home of the remote controller.

Stores file names

The folders that contain the Workbench and platform/station stores each contain a set of three data files, one per type of store.

NOTE: Only the appropriate Workbench, platform or station tools may be used to modify these data files. Attempts to modify them by other means renders them corrupt and unusable.

- `keystore.jceks` is the **User Key Store**. In Workbench it contains a company's root CA, intermediate, and code-signing certificates. In a server, it contains the server certificate.
- `cacerts.jceks` is the **User Trust Store**. In a client it contains the root CA and intermediate certificates with only their public keys.
- `exemptions.tes` is the **Allowed Hosts** list. In a client it contains the certificate for hosts (servers) with whom the client may securely communicate even though the client either:
  - does not have a root CA certificate in its **System** or **User Trust Store** for the server, or
  - may have a matching root CA certificate, but the `Common Name` or `Alternate Server Name` of the server certificate is not the same as the host name of the server being authenticated.
- `.bks` is the FIP compliance file for access control.

NOTE: A certificate in the Workbench **User Key Store** may have the same name as a certificate in a platform/station **User Key Store**, but they may not be the same certificate. Similarly, files in these stores may have differing alias names, and, in fact, contain the same public keys. It is the public/private key pair that defines a certificate, not the certificate's name.

CSR folder structure

You may create CSRs (Certificate Signing Requests), store them in, and import signed certificates (.pem files) from any folder on your Supervisor or engineering PC. The default location is a working folder in the user file space.

The first time you access the Certificate Management view from Workbench, the system creates an empty **certManagement** folder in the following location:

`c:\Users\username\Niagara4.x\certManagement`, where `username` is the name you use to log in to the computer.

In the Nav tree, this location is: **My File System→User Home→certManagement**.

This folder, in Workbench's user space, is a working folder for storing CSRs and .pem files exported and imported by the **Certificate Manager**. Within this folder, you may create a structure for managing exports and imports or you may use a different location for exports and imports.

NOTE: Do not confuse the `certManagement` folder with the `certificates` folder that stores one or more certificates used to validate the authenticity of Niagara system licensing files. The `certificates` folder has nothing to do with secure communication.

The platform and station folders do not have a `certManagement` folder.

Creating a CSR folder structure

A CSR (Certificate Signing Request) folder structure helps to organize a large number of server certificates for easy retrieval. You create this structure under the **`certManagement`** folder, which is an automatically-created folder in your personal niagara_user_home.

Step 1 Using Windows Explorer, locate your niagara_user_home: `C:\Users\username\niagara4.x\brand\certManagement`, where

`username` is your name or other text used to identify you as the user of your computer.

`brand` is your company name or other name used to label personal information.

Step 2 Create folders under `certManagement`.

For example:

```
certManagement
rootCertificate
intermediateCertificates
serverCertificates
code-signingCertificates
```

Certificate set up

Configuring a network for secure communication using digital certificates involves accessing the appropriate stores; creating certificates and certificate signing requests; signing certificates; importing them into hosts **User Key Store**s; and importing the root CA certificate (or intermediate certificate) into client **User Trust Store**s.

CAUTION: If the private key of your root CA and intermediate certificates fall into the wrong hands, your entire network can be in danger of a significant cyber attack. To ensure security, always create the root CA and intermediate certificates, and use them to sign other certificates inWorkbench running on a secure computer, which is located under lock and key. Use this computer for only one purpose: to manage and sign certificates. Never connect this computer to the Internet, and ever access it over your company network. Carefully protect any thumb drive that contains any certificate with its private key.

You may use a third-party CA (Certificate Authority), such as VeriSign or Thawte to sign your certificates, or you may serve as your own CA.

Unless absolutely necessary, do not use a Supervisor or engineering PC to access a controller remotely for the purpose of generating a server certificate and CSR. The preferred best practice is to set up certificates before distributing each controller to its remote location. If controllers are already in the field, travel to the remote location, take the controller off the Internet and corporate LAN, then connect your engineering PC directly to the controller using a cross-over cable.

Root and intermediate certificate checklist

This checklist assumes that you are serving as you own CA (Certificate Authority). It summarizes the steps for setting up digital certificates using the Workbench **User Key Store** of a physically and electronically secure computer.

Use the check list to make sure you perform all necessary configuration tasks.

☐ Computer and device network disconnected from the company LAN and global Internet.

☐ Needed certificates identified: one root CA certificate, two or more intermediate certificates (optional) and one server certificate per controller. You need a code-signing certificate if you will be customizing the system by adding program objects. .

☐ Logical certificate naming convention established (a naming convention is not required, but it will help you differentiate among your certificates).

☐ CSR folder structure under the **certManagement** folder in the niagara_user_home created.

☐ Root CA certificate and any intermediate certificates created.

☐ CSR for each intermediate and code-signing certificate created.

☐ Any intermediate and code-signing certificates signed using the root CA certificate.

☐ Any signed intermediate certificates imported back into the Workbench User Key Store where they were originally created.

☐ Backup of the root CA certificate and the signed intermediate certificates created.

☐ Root CA certificate with only its public key exported in preparation to import it into the platform/station Trust Stores.

Supervisor/engineering PC checklist

Use this checklist to verify that you completed all required tasks to set up a Supervisor or engineering PC platform and station.

- ☐ **NiagaraNetwork** enabled.
- ☐ Remote controller(s) set up as Supervisor/engineering PC client(s).
- ☐ Server certificate for the Supervisor/engineering PC platform/station created.
- ☐ CSR for the server certificate generated.
- ☐ Server certificate CSR signed by the root CA or intermediate certificate's private key.
- ☐ Signed server certificate imported back into the platform/station's **User Key Store**.
- ☐ The existence of the third-party's root CA certificate in the platform/station's **System Trust Store** confirmed or root CA certificate imported into the platform/station's **User Trust Store**.
- ☐ Certificate .pem file deleted.
- ☐ Confirmed platform (Niagarad) enabled.
- ☐ Secure **`FoxService`** confirmed (the default) and **`Foxs Cert`** selected.
- ☐ Secure **`WebService`** confirmed (the default) and **`Https Cert`** selected.

Platform and station checklist

Use this checklist to verify that you completed all required tasks to set up a new platform and station.

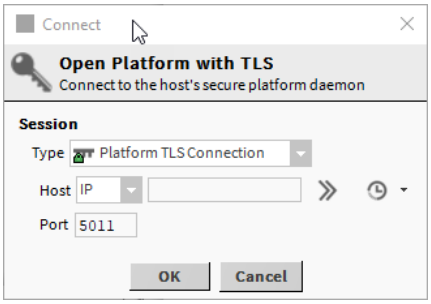
- ☐ **NiagaraNetwork** enabled.
- ☐ Supervisor set up as controller client..
- ☐ Server certificate(s) created on each platform/station. If needed, a separate server certificate created for each communication protocol: foxs, https, and platformtls created. .
- ☐ CSR(s) generated on each platform/station.
- ☐ CSR(s) signed by the root CA or intermediate certificate's private key. .
- ☐ Signed server certificate(s) imported back into the platform/station **User Key Store**.
- ☐ Certificate(s) .pem files deleted.
- ☐ If using a third-party CA, the existence of the third-party's root CA certificate in the platform/station's **System Trust Store** confirmed.
- ☐ If serving as the CA, company's root CA certificate imported into the platform/station's **User Trust Store** either one at a time, or using a provisioning job..
- ☐ Confirmed platform (Niagarad) enabled and correct certificate assigned.
- ☐ Secure **`FoxService`** confirmed (the default) and **`Foxs Cert`** selected. For the specific procedure.
- ☐ Secure **`WebService`** confirmed (the default) and **`Https Cert`** selected. For the specific procedure.

Opening a secure platform connection (niagarad)

Even before you configure digital certificates to provide server identity verification, every connection you make from a client to a server can be secure because you can manually verify the authenticity of the server.

Step 1 Right-click **My Host** (for Supervisor) or an IP address (for a controller) and click **Open Platform**.

The **Connect** window opens with `Platform TLS Connection` already selected.



This window identifies the entity to which you are connecting: your local computer, a Supervisor platform, or a controller with an IP address.

Step 2 If needed, enter the host IP and click **OK**.

If you are accessing the platform for the first time, the system displays an identity verification warning and a self-signed, default certificate.

This message and certificate are expected for these reasons:

- The `Subject` or `CN` (Common Name) of the default certificate (`Niagara4`) does not match the host name, which is usually the host IP address or domain name.
- The default certificate's `Issued By` and `Subject` are the same indicating that the certificate is self-signed. No third-party CA (Certificate Authority) has verified the server's authenticity.
- The certificate is signed, but no root CA certificate in the client's **User** or **System Trust Store** can verify its signature.

Step 3 If you are presented with this warning and a certificate, make sure you recognize the certificate's `Issued By` and `Subject` properties.

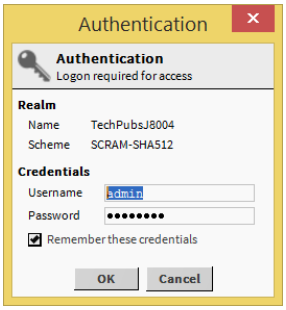
CAUTION: Do not approve a certificate if you do not recognize these properties.

Step 4 Assuming that this is the default `tridium` certificate, which can be trusted, click **Accept**.

Accepting the certificate creates an approved host exemption in the platform/station **Allowed Hosts** list.

NOTE: Although the name of the default certificate is the same for each controller and for Workbench, the content of each certificate is unique. Do not attempt to export and use the same `default` certificate for each controller in your network.

The system asks you to enter or confirm your platform credentials.

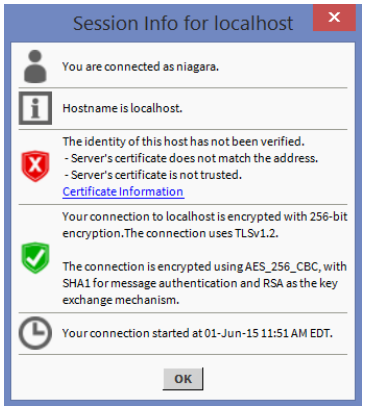


Step 5 Enter your platform credentials and click **OK**.

The platform is now connected over a secure connection. All data transmitted are encrypted. If you logged on for the first time and accepted the default certificate, only the server's identity cannot be validated.

Step 6 To confirm that you are using the self-signed certificate, right-click **Platform** in the Nav tree and click `Session Info`.

The system displays session information.



- The red shield with the X indicates that the handshake was unable to verify the authenticity of the server's certificate. To view the certificate, click the link (`Certificate Information`).
- The green shield with the check mark indicates that encryption is enabled. In this example, the secure connection is using `TLSv1.2` as the protocol and data is encrypted using `AES_256_CBC` (Advanced Encryption Standard) with `SHA1` (hash function) and `RSA` (Rivest-Shamir-Adleman), the most widely used public key cryptography algorithm.

Step 7 Click **OK**.

The tiny lock on the platform icon in the Nav tree indicates a secure, encrypted connection.

Enabling the NiagaraNetwork

The **NiagaraNetwork** provides the physical connections for data transmission. Secure communication ensures that data are transmitted securely between trusted entities.

Step 1 Right-click the node in the **Drivers** container and click **Views→Property Sheet**.

Step 2 Expand the **`NiagaraNetwork`** property.

Step 3 Confirm that the `true` check box is selected for **`Enabled`**.

Confirming client/server relationships

At any given time the Supervisor station may be the client of a controller station and vice versa. This procedure confirms that a client for the Supervisor station exists in the controller station and a client for the controller station exists in the Supervisor station.

Step 1 Expand the **Drivers→NiagaraNetwork** node in the Supervisor Nav tree. It should contain a node for each controller.

Step 2 Expand the **Drivers→NiagaraNetwork** in a controller Nav tree. It should contain a node for the Supervisor station.

Step 3 If either node does not exist, discover the station.

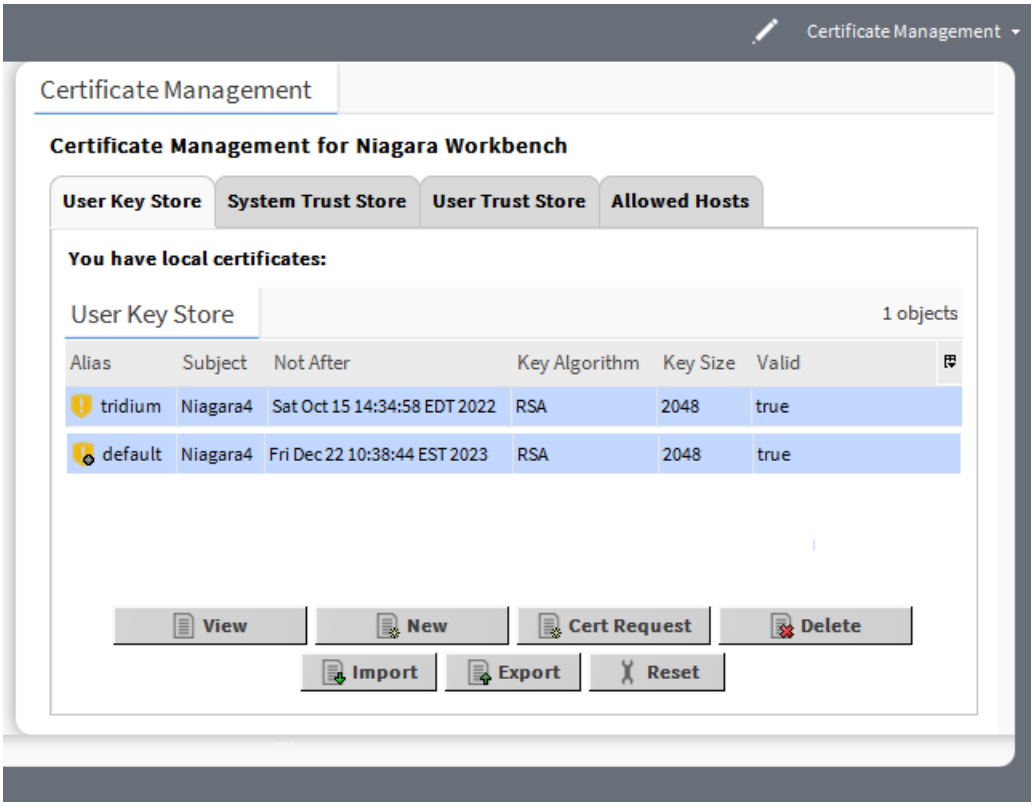
Creating a root CA certificate

A company's root CA certificate is a self-signed certificate. Companies that serve as their own CA use its private key to sign their intermediate, server, client and code-signing certificates. The root CA certificate resides in the Workbench Certificate Management **User Key Store** with both its public and private keys. You export it with only its public key so that you can import it into each platform/station's **User Trust Store**.

Prerequisites: You have the required authority to create certificates. You are working in Workbench on a computer that is dedicated to certificate management, is not on the Internet or the company's LAN and is physically secure in a vault or other secure location.

Step 1 Access the Workbench **Certificate Management** view by clicking **Tools→Certificate Management**.

The **Certificate Management** view opens to the **User Key Store**.

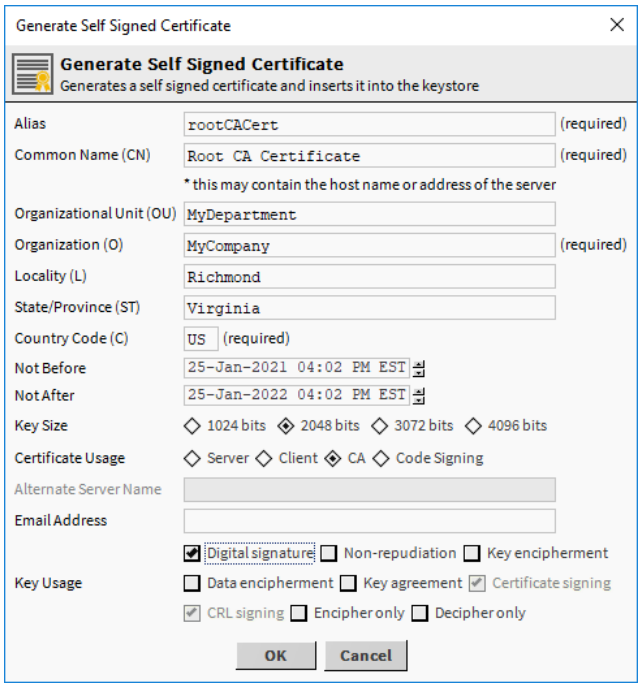


This key store contains a default, self-signed `tridium` certificate. As of Niagara 4.13, the key store also contains the self-signed `default` certificate that is mainly used for recovery purposes and cannot be deleted.

Step 2 Confirm that you opened the Workbench **User Key Store** and click the **New** button at the bottom of the view.

NOTE: If you opened the platform/station **Certificate Management** view by mistake, you can still create a root CA certificate, but it will not be available to sign the other certificates.

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



All certificates begin as self-signed certificates. Only the root CA certificate remains self-signed because it sits at the top of the certificate chain.

Step 3 Fill in the form and click **OK**.

- Use **Alias** to identify this as a root certificate.
- The **Common Name (CN)** becomes the **Subject** (also known as the Distinguished Name). For a root CA certificate, the **Common Name (CN)** may be the same as the **Alias**.
- **Organization** should be the name of the company.
- Although **Locality** and **State/Province** are not required and are arbitrary, leaving them blank generates a warning message.
- The two-character **Country Code** is required and must be a known value, such as: US, IN, CA, FR, DE, ES, etc. (refer to the ISO CODE column at countrycode.org).
- Based on the **Not Before** and **Not After** dates, certificate validity defaults to a year. A longer period is not recommended and not tolerated by some browsers. Changing to a new certificate annually or even within a year makes it more likely that your certificate contains the latest cryptographic standards, and reduces the number of old, neglected certificates that can be stolen and re-used for phishing and drive-by malware attacks.
- **Key Size** defaults to 2048. A larger key improves security and does not significantly affect communication time. The only impact it has is to lengthen the time it takes to create the certificate initially.
- For **Certificate Usage**, select `CA`.

The **Private Key Password** window opens.

Step 4 Enter and confirm a strong password, and click **OK**.

The system informs you that the certificate has been submitted. Soon the certificate appears behind the Info message in the **User Key Store** table.

Step 5 To continue, click **OK**.

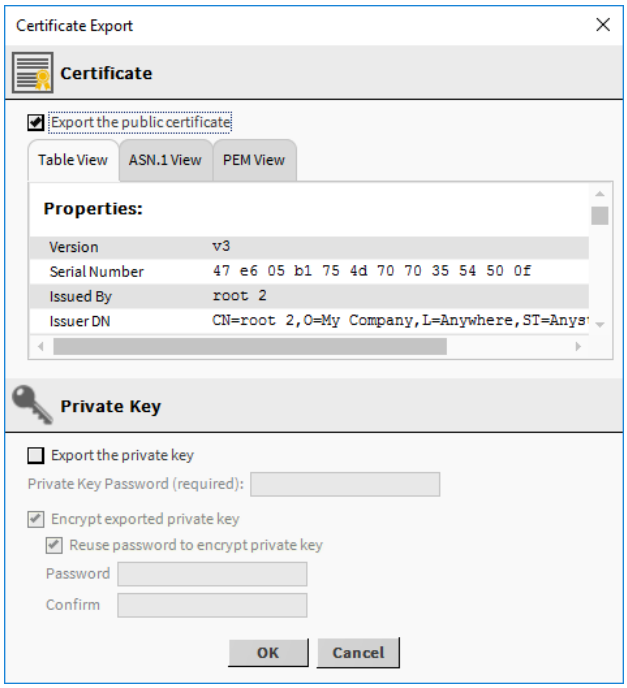
The root CA certificate now exists with both its keys in the Workbench **User Key Store**. From this location you can use it to sign other certificates (intermediate, server, client and code-signing).

NOTE: Since this certificate is not signed by any higher certificate authority, it is always identified with an exclamation icon ( ). As the self-signed certificate, it cannot be trusted. This is why you must protect the computer (and thumb drive) on which it resides by keeping the computer off the Internet, corporate LAN, and most securely, in a locked physical location.

For this certificate to authenticate the certificates it signs, you now need to export it with only its public key and import it into the **User Trust Store** of each client computer and platform/station.

Step 6 Select the new root CA certificate and click **Export**.

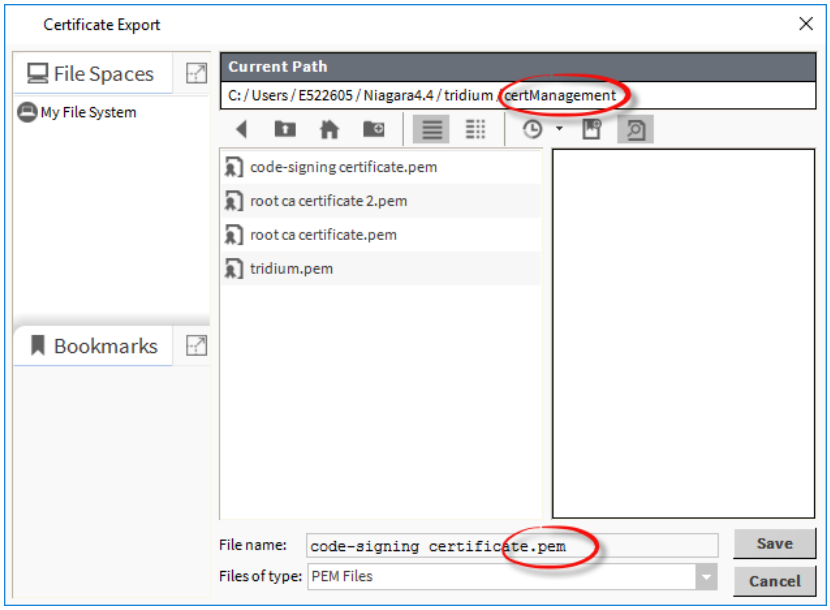
The **Certificate Export** window opens.



CAUTION: Do not click the check box to **`Export the private key`**.The only time you click this check box is when you are backing up the certificate to another location for safe keeping.

Step 7 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.



Notice the Current Path. This is where the system stores the exported certificate.

Step 8 Navigate to a `rootcert` folder or location on a thumb drive, and click **Save**.

The system reports that it exported the certificate successfully.

Step 9 To complete the export, click **OK**.

When exported with only its public key, the root CA certificate may be freely distributed. You are ready to manually import the root CA certificate with only its public key into the **User Trust Store** of the computer, usually a Supervisor (or engineering) computer, from which to either manually, or with a provisioning job, install this certificate in the **User Trust Store** of all remote platforms/stations.

Password strength

To protect each certificate's private key you must supply a private key password. When backing up certificate private keys by exporting certificates, you may use an additional encryption password. (The default encryption password is the same as the private key password.) To prevent unauthorized access, your passwords need to be strong.

A strong platform or station password:

- Has 10 or more characters.
- Includes letters, punctuation, symbols, and numbers.
- Is unique for each set of credentials.

  CAUTION: Do not reuse passwords.
- Avoids dictionary words in any language, words spelled backwards or words that use common misspellings and abbreviations, sequences or repeated characters, personal information such as your birthday, driver's license, passport number, etc.

These precautions were adapted from information at microsoft.com, which provides a secure password checker you can use to test the strength of any password. Niagara 4 allows you to control password strength for user authentication. The password strength configuration for user authentication does not apply to certificate passwords.

Creating a Client Certificate for Syslog configuration

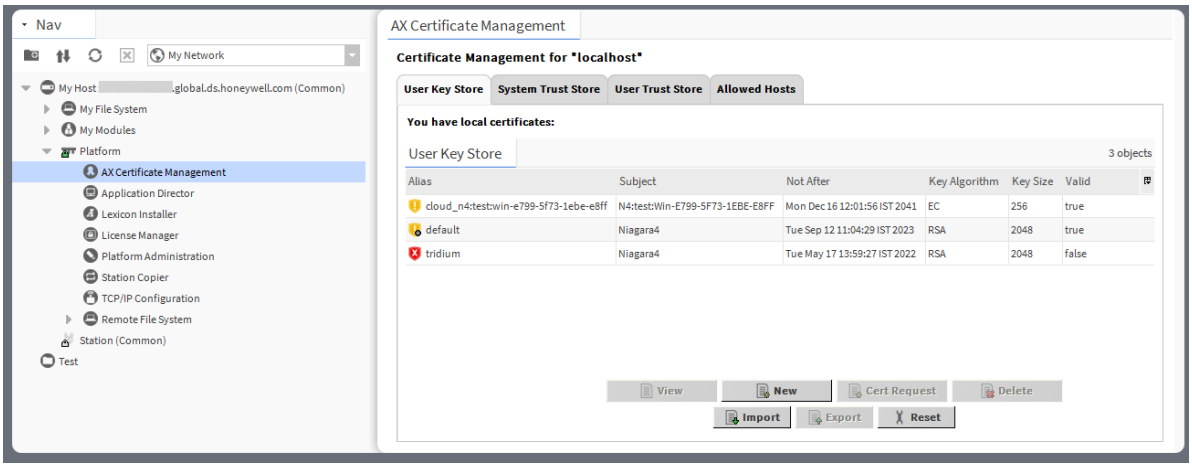
The server configurations require clients to support a client-signed certificate by the approved CA certificate. This CA certificate has been added to the appropriate folder in the server to support only allowing authorized clients to send messages. Use this certificate in the `Client Alias` field for Syslog configurations.

Prerequisites: You have the required authority to create and manage certificates. You are either running Workbench on your PC or laptop.

Step 1 To open the certificate stores, do one of the following in the Nav tree:

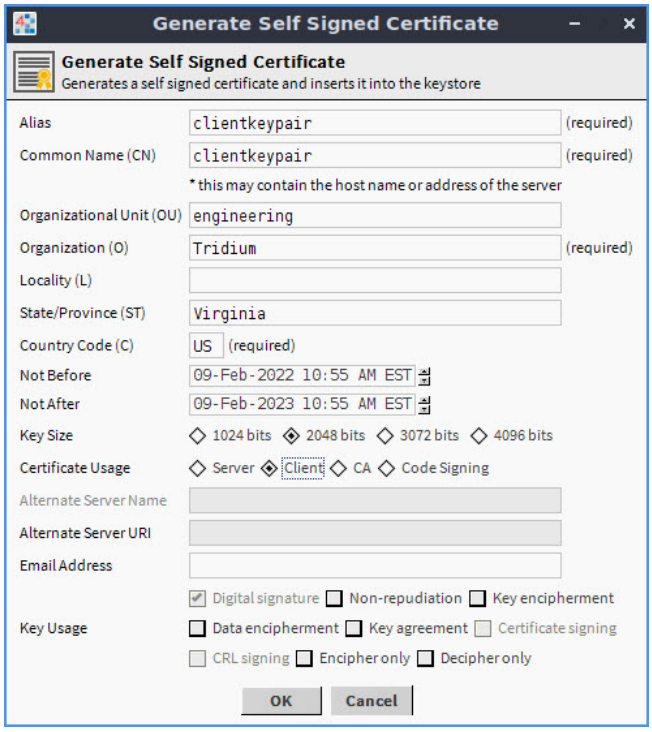
- Expand **Platform** and double-click **Certificate Management**.
- Expand **Station→Config→Services→PlatformServices** and double-click **CertManagerService**.

Both steps open the same stores. Which to use depends on how you are connected to the platform/station.



Step 2 Click the **New** button at the bottom of the view.

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



Step 3 Fill in the form and click **OK**.

- Use **Alias** to identify this as a client certificate.
- The **Common Name (CN)** becomes the **Subject** (also known as the Distinguished Name).
- **Organization** should be the name of the company.
- Although **Locality** and **State/Province** are not required and are arbitrary, leaving them blank generates a warning message.
- The two-character **Country Code** is required and must be a known value, such as: US, IN, CA, FR, DE, ES, etc. (refer to the ISO CODE column at countrycode.org).
- For **Certificate Usage**, select `Client`.

Step 4 Create a CA certificate. Refer to “Creating a root CA certificate”.

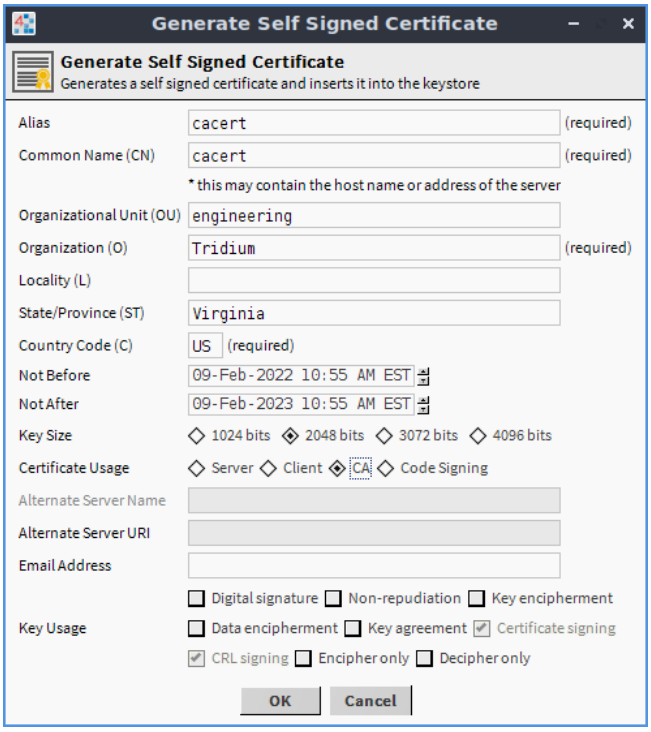
Step 5 Click **Cert Request** to create a certificate signing request.

Step 6 Navigate to **Tools→Certificate Signer Tool**.

The **Certificate Signing** window opens.

Step 7 Select the certificate signing request and enter the password set while creating the CA certificate.

Step 8 Sign the client certificate with the CA certificate in Workbench.



Step 9 Import the signed certificate back into Workbench, replacing the self-signed certificate.

Step 10 Export the CA certificate and key as a PEM separately from Workbench.

This CA certificate and key can be added to the CA certificate folder in the Syslog server and be used to sign the server certificate while configuring the mutual authentication.

Creating a server certificate

Each Supervisor PC, engineering laptop, remote controller, and remote station requires a server certificate for those times when it functions as a server. If you plan to use as a server certificate the default, self-signed, certificate that you approve when you log in to Workbench or boot a platform for the first time, you may skip this procedure. If it is important to you for each certificate to identify the Locality and State, use this procedure to make a new certificate for each server.

Prerequisites: You have the required authority to create and manage certificates. You are either running Workbench on your PC or laptop, or are connected to the remote controller on which you are creating the certificate.

TIP: While not a requirement when creating a remote server certificate, as a best practice, you should disconnect both your computer and the controller platform from the Internet and company LAN, then connect your Workbench computer to the platform using a crossover cable.

Step 1 To open the certificate stores do one of the following in the Nav tree:

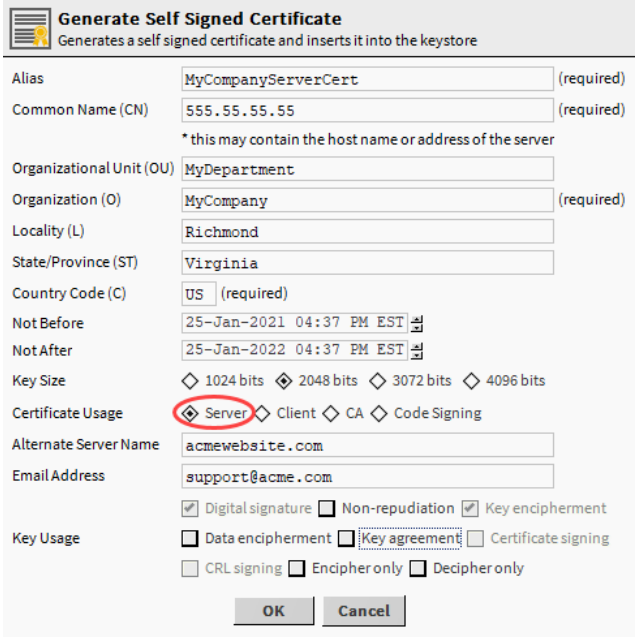
- Expand **Platform** and double-click **Certificate Management**.
- Expand **Station→Config→Services→PlatformServices** and double-click **CertManagerService**.

Both steps open the same stores. Which to use depends on how you are connected to the platform/station.

Step 2 Confirm that the title at the top of the view identifies the host for which you are creating the server certificate. For a remote controller, this is the IP address.

Step 3 Click the **New** button at the bottom of the view.

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



Step 4 Give the certificate at least an **Alias**, **Common Name (CN)**, **Organization**, **Locality**, **State/Province**, and **Country Code**.

- Use **Alias** to identify this as a server certificate, including in the name the company, geography or department.
- **Common Name (CN)** should be the same as the host name, which is how a server identifies itself. The common name becomes the **Subject** (also known as the Distinguished Name). The IP address of a controller or its Fully Qualified Domain Name (FQDN) are appropriate **Alias** and **Common Name**s for a remote controller or Supervisor station.

  An FQDN is the **Hostname** plus the **Primary Dns Suffix**. For a computer, you can see this name in My Computer Properties: "Full computer name." For a controller, there is no good place to see this name, but it would be something like: mycontroller.mydomain.com or mycontroller.mydomain.net.

  NOTE: Do not use the same name for **Common Name (CN)** of a server certificate that you use for a root or intermediate certificate's **Common Name (CN)**.
- Although **Locality** and **State/Province** are not required and are arbitrary, leaving them blank generates a warning message. Third-party CAs may not sign certificates without these properties defined.
- The two-character **Country Code** is required and must be a known value, such as: US, IN, CA, FR, DE, ES, etc. (refer to the ISO CODE column at countrycode.org).
- **Not Before** and **Not After** define the period of validity for the certificate.
- **Key Size** defaults to 2048. A larger key improves security and does not significantly affect communication time. The only impact it has is to lengthen the time it takes to create the certificate initially.

  If a third-party will sign the certificate, consult with your CA (Certificate Authority) to determine the acceptable key size. Some CAs support a limited number of key sizes.
- For **Certificate Usage**, select **Server** for a platform/station.
- For server certificates, if **Common Name** is an IP address, use a Fully Qualified Domain Name for the **Alternate Server Name**.

The **OK** button activates when all required information is provided.

Step 5 To create the certificate, click **OK**.

The **Private Key Password** window opens.

Step 6 Enter a strong password for a unique password or select the **Use global certificate password** check box.

Your password must be at least 10 characters long. At least one character must be a digit; one must be lower case; and one must be upper case.

The system submits the certificate for processing in the background. A pop-up window in the lower right of your screen advises you regarding the time it may take to generate the certificate. The length of time it takes depends on the key size and the platform's processing capability. When created, the certificate appears as a row in the **User Key Store** table.

Step 7 To view the certificate from the platform/station's **User Key Store**, double-click it or select it and click **View**.

Notice that the `Issuer` and `Subject` are the same and the certificate is identified with a yellow shield icon (). These factors indicate that this is a self-signed certificate.

Step 8 Confirm that the information is correct.

NOTE: To change a certificate you just created, delete it and create a new certificate. Do not delete a certificate that is already in use.

Repeat this procedure to create additional certificates.

Creating a code-signing certificate

The system signs code objects using a code-signing certificate that is password protected. This procedure explains how to generate a code-signing certificate using Workbench. You may use a third-party tool to generate a code-signing certificate followed by importing it into your Workbench **User Key Store**. Such an imported certificate must have a code-signing set as its extended key usage. This is a standard certificate extension.

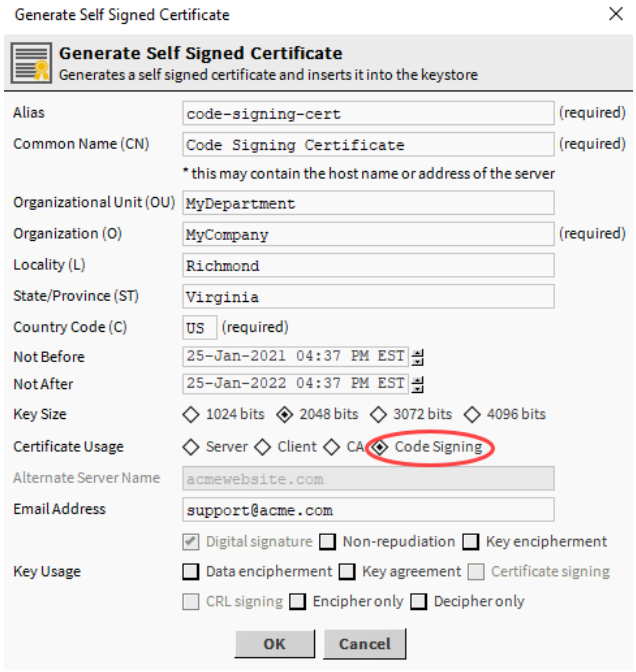
Prerequisites: You are working in Workbench running on a Supervisor or engineering PC.

Step 1 Click **Tools→Certificate Management**.

The **Certificate Management** view opens.

Step 2 Click the **New** button at the bottom of the view.

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



Step 3 Fill in the properties.

In addition to the required properties, define your **`Locality (L)`** (city) and **`State/Province (ST)`**. Without these properties the system reports an error message.

`Country Code` is a two-character ISO code (refer to the ISO CODE column at countrycode.org).

Choose `Code Signing` for **`Certificate Usage`**.

The **OK** button activates when all required information is provided.

Step 4 To create the certificate, click **OK**.

The **Private Key Password** window opens.

Step 5 Enter a strong password and click **OK**.

Your password must be at least 10 characters long. At least one character must be a digit; one must be lower case; and one must be upper case.

The system submits the certificate for processing in the background.

Step 6 When the certificate has been created, click **OK**.

This self-signed certificate appears as a row in the **`User Key Store`** tab, identified by a yellow shield icon ( ).

Protect this certificate and password! If someone steals your certificate and knows your password, they could damage your operation by using your certificate to sign their own malicious code.

Creating a CSR

A CSR (Certificate Signing Request) creates a .csr file for each intermediate, server, and code-signing certificate. This file can be signed by a root or intermediate CA certificate.

Prerequisites: For creating intermediate and code-signing certificates you are viewing the Workbench stores. For creating server certificates you are viewing the platform/station stores.

Step 1 Select the certificate to sign, and click **Cert Request**.

The **Certificate Request Info** window opens.

Step 2 Confirm that the certificate properties are correct and click **OK**.

One of the following happens:

- If you are preparing a CSR for a server certificate, the system displays the **certManagement** folder for you to choose the location to store the CSR.
- If you are creating a CSR for a CA certificate (root or intermediate) or a code-signing certificate, the **Certificate Manager** prompts you for the private key password. Enter the password and click **OK**. The system displays the **certManagement** folder for you to choose the location to store the CSR.

The **`Alias`** for the certificate is used as the file name of the CSR.

Step 3 Use the default folder, or select a different folder in which to store the CSR and click **Save**.

The system displays, `CSR generation complete`.

Step 4 To confirm completion, click **OK**.

NOTE: Once you create a CSR, do not delete the original certificate from which you created the CSR. Later in the process you will import a signed certificate back into the **User Key Store** where its public key must match the private key of the original certificate. Creating a new certificate with the same name does not generate the same key pair and results in errors when you try to import the signed certificate. If it is absolutely necessary (for example, if the computer on which the certificate is stored is vulnerable), you may export the original certificate with its private key and import it into the **User Key Store** when you receive the signed certificate. But, ideally, you should leave the original certificate in the **User Key Store** of the original secure host.

Step 5 If an external CA, such as VeriSign or Thawte, will sign your server certificates, follow the CSR submission procedure as required by the CA.

The CA verifies that you are who you claim to be, that each certificate is for a server your organization actually maintains, and other important information. They then return a signed server certificates (one for each server).

The CA may compress both the new signed certificate and a copy of the root CA certificate containing only the public key with password protection, put both on a website, email the links to you, and phone you with the password for the compressed, password-protected files. The root CA certificate with its public key does not have to be protected and can be sent via email.

Signing a certificate

Signing a certificate is the job of a CA (Certificate Authority). A variety of certificate-signing software tools are available. You are not required to use the framework and Workbench to sign certificates. This procedure documents how to sign certificates. It applies to companies who serve as their own CA. In a large installation, you use your root CA certificate to sign any intermediate certificates and the intermediate certificates to sign your server and code-signing certificates. In a small installation, you may use your root CA certificate to sign all certificates.

Prerequisites:

- You are working in Workbench on a physically and electronically secure PC that is never connected to the Internet, and is used exclusively to sign certificates.

- The root CA or intermediate certificate that will do the signing is in the Workbench **`User Key Store`**.
- You know the password of the CA signing certificate (root or intermediate) that will sign the certificate(s).
- You have one or more CSR files (signing requests) ready to sign.

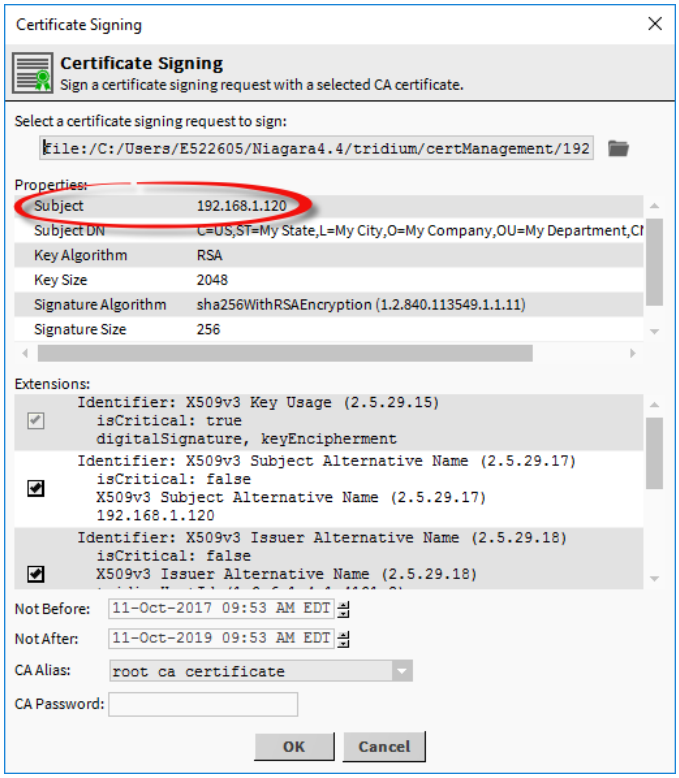
NOTE: To ensure network security, always sign certificates using Workbench on a computer that is disconnected from the Internet and from the company LAN. Maintain this computer in a physically secure location.

Step 1 In Workbench on your physically and electronically secure (and never connected to the Internet) PC that is used exclusively to sign certificates, click **Tools→Certificate Signer Tool**.

The **Certificate Signing** window opens.

Step 2 Click the folder icon, locate, and open the CSR for the certificate you wish to sign.

The **Certificate Signing** window expands to show certificate details.



Step 3 Confirm that this is the correct CSR by checking the **Subject**.

Step 4 Select the date on which the certificate becomes effective (**`Not Before`**) and the date after which it expires (**`Not After`**).

Step 5 For **CA Alias**, use the drop-down list to select the certificate (root or intermediate) whose private key will sign this certificate.

Step 6 Supply the CA certificate's password and click **OK**.

Signing is done by the private key of the root or intermediate certificate.

The same file folder, `C:/Users/[username]/Niagara4.x/certManagement`, displays with the file name (extension: .pem) filled in for you.

You may modify this file structure to aid in the management of these files.

Step 7 To complete the signing, click **Save**.

Step 8 Copy the signed certificate .pem file to a thumb drive and import it back into the **User Key Store** of the computer that created the certificate and generated the CSR.

You can repeat this procedure for each CSR.

NOTE: In Niagara there is added support for bulk certificate signing. For more details refer to the "Signing multiple certificates" topic.

Importing the signed certificate back into the User Key Store

Signing a certificate creates a .pem file, which is only intended for importing back into the **User Key Store** that contains the original certificate with the matching private key. For a server certificate this is the platform/station **User Key Store** that originally created the certificate and CSR. For an intermediate certificate or a code-signing certificate, this is, most likely, the Workbench **User Key Store** on the secure computer, which you use to sign other certificates.

Prerequisites: You have the signed .pem files. The focus is on the User Key Store in the appropriate stores location (Workbench or platform/station).

Step 1 Click **Import**.

Step 2 Locate and select the signed certificate's .pem file (the output of the certificate signer or the .pem file you received from a third-party CA) and click **Open**.

The **Certificate Import** window opens.

Step 3 Confirm that you are importing the correct certificate and click **OK**.

If the `Alias` of the certificate you are importing is not the same as the `Alias` of the certificate you are replacing, the system prompts you for the `Alias` of the certificate to replace.

Step 4 If needed, enter the **Alias** and click **OK**.

The green shield icon (✅) replaces the yellow shield icon (⚠) next to the certificate `Alias` in the **User Key Store** tab.

Step 5 Using the operating system, delete the .pem file(s) from the secure Workbench computer.

Installing a root certificate in the Windows trust store

For communication to be secure when accessing a station using the web UI, a company's root certificate must be imported into the Windows trust store.

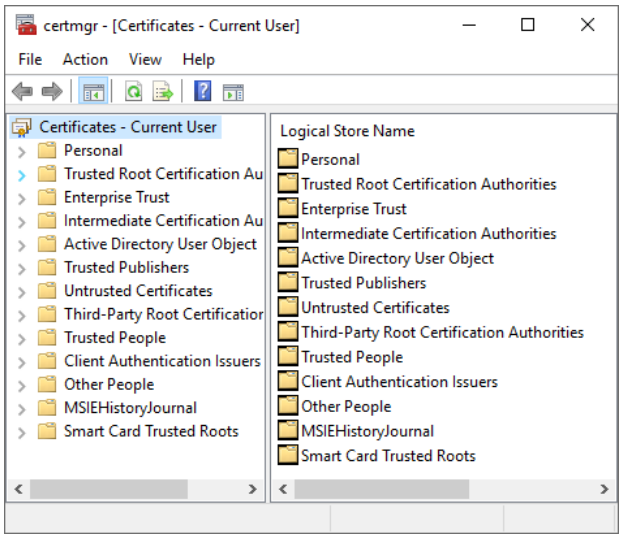
Prerequisites: Using Workbench, you exported the root certificate into a directory you can access from the PC. (The extension for a root certificate file name is .pem).

Step 1 On the PC, open a Windows command prompt by clicking **Start** and typing `cmd`.

The **Command Prompt** window opens.

Step 2 Type `certmgr.msc` and press **Enter**.

The certificate manager window opens.



Step 3 In the column to the right, right-click the **Trusted Root Certificate Authorities** folder and click **All Tasks→Import**.

the Certificate Import wizard window opens.

Step 4 Follow the steps of the wizard to browse to the User Home where the root certificate .pem file is located, select to view All Files..., select the file and click **Open**.

Step 5 Finish the wizard.

Exporting a certificate

There are two reasons to export certificates: 1) to create a root CA certificate with only its public key for each client's **User Trust Store** and browser, and 2) to create a backup, for safe keeping, of all certificates with their private keys.

As soon as you finish importing all certificate .pem files back into their respective **User Key Store**s, make a backup of all of certificates and store the backup on a thumb drive in a separate, physically secure location. You back up each certificate one at a time.

NOTE: To protect your backups create strong passwords and store backup media in a vault. These backups contain the key(s) used to sign all server certificates.

Step 1 Open the stores that contain the certificate(s) to export.

Step 2 On the **User Key Store** tab, select the certificate and click **Export**.

The system opens the **Certificate Export** window.

Step 3 Do one of the following:

In addition to the private key password, you should use an encryption password to provide double-password protection. The default encryption password is the same as the private key password.

- To create a CA certificate (root or intermediate) for importing into a client **User Trust Store**, just click **OK** (do not select `Export the private key`).
- To back up a certificate with its private key, click `Export the private key`, deselect `Reuse password to encrypt private key` under `Encrypt exported private key`, and supply the additional password.

Step 4 Navigate to a location on a thumb drive and click **Save**.

The system reports that the export was successful.

Step 5 To complete the action, click **OK**.

Manually importing a certificate into a User Trust Store

If your **System Trust Stores** already contain the root CA certificate of the CA (Certificate Authority) that signed your intermediate, server or code-signing certificates, you do not need to import anything. If you are serving as your own CA, you must import the root CA certificate you exported (without its private key) into the **User Trust Store** of each client. Each platform and station share the same stores. This procedure documents how to manually import the root CA certificate into an individual client.

Prerequisites: The focus is on the User Trust Store in the appropriate stores location, which would be the platform/station User Trust Store for a server certificate and the Workbench User Trust Store for a code-signing certificate.

NOTE: There is no need to import the server certificates or the intermediate CA certificates to any **User Trust Store**. Each signed server certificate carries within it any intermediate and root CA certificate information.

Step 1 Select the **User Trust Store** tab.

Step 2 Click **Import**.

Step 3 Locate and select the root CA certificate's .pem file on the thumb drive and click **Open**.

The **Certificate Import** window opens.

Step 4 Confirm that the **Subject** property identifies the correct certificate and click **OK**.

The system imports the certificate, identifying it with a green shield icon (✅)

You may repeat this procedure for each client platform/station, or use a provisioning job to automate the process.

Installing a certificate

If the **System Trust Store**s in your remote stations already contain the root CA certificate from the CA (Certificate Authority) that signed your intermediate, server or code-signing certificates, you do not need to run a provisioning job. If your company serves as its own CA, you must install a root CA or intermediate certificate in the **User Trust Store**s of each platform/station that serves as a client. Installing this certificate can be useful before running a signed provisioning robot on several stations.

Prerequisites: The **BatchJobService** is available under **Services**. The **ProvisioningNwExt** component is available under the **NiagaraNetwork**. The **Niagara Network Job Builder** (one-time) or **Niagara Network Prototype View** (reoccurring job) is open.

This is a recurring provisioning job that uses the **NiagaraNetworkJobPrototype**.

Step 1 On your Supervisor, expand **Config→Services→PlatformServices**, double-click **CertManagerService** and click the **User Trust Store** tab.

NOTE: You cannot complete this procedure if you import the certificate into the Workbench **User Trust Store** of your Supervisor PC.

Step 2 Click the **Import** button, navigate to the location on the thumb drive that contains the root CA certificate and click **Open**.

Step 3 Confirm that the `Subject` of the certificate identifies it as the root CA certificate and click **OK**.

The system imports the certificate in preparation for the provisioning job.

Step 4 Navigate to the location in the station where you manage provisioning jobs.

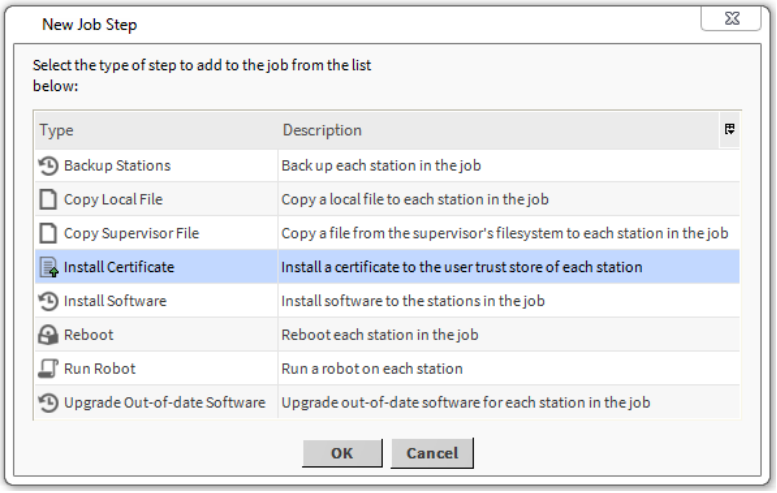
Step 5 Drag a **NiagaraNetworkJobPrototype** component to this location and name the component something like, "Root CA certificate provisioning."

Step 6 Double-click the new component.

The **Niagara Network Prototype View** opens.

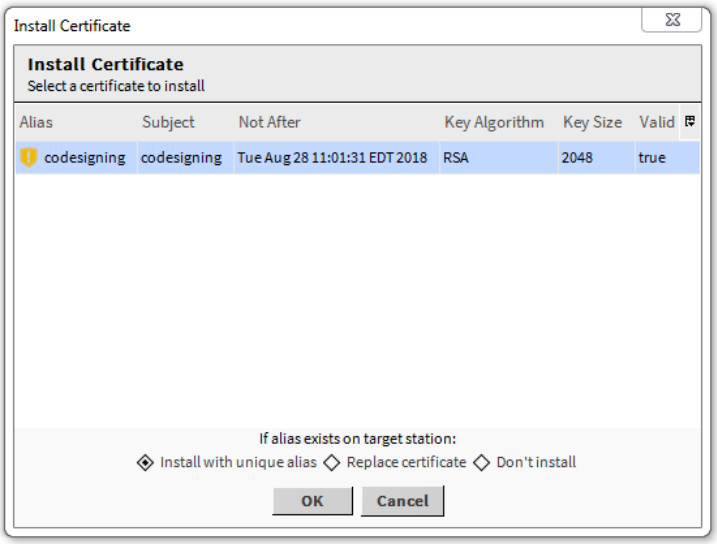
Step 7 In the top pane, **Provisioning steps to run**, click ⊕ (Add).

The **New Job Step** window opens.



Step 8 Select **`Install Certificate`** and click **OK**.

The **Install Certificate** window opens.



Step 9 To complete the installation, select the root CA certificate and click **OK**.

Step 10 Define the stations to include in the job.

The system copies the certificate from the Supervisor station's **User Trust Store** to the **User Trust Stores** of the other clients.

Station health confirmation

When you finish configuring a client or server, stop and restart a secure station and check station health. The system does not validate existing connections against new certificates until you restart the station. The system does not automatically change connections from Http and Fox to Https and Foxs, even when you enable **`Https Only`** and **`Foxs Only`**, until you reestablish the connection.

Viewing session information

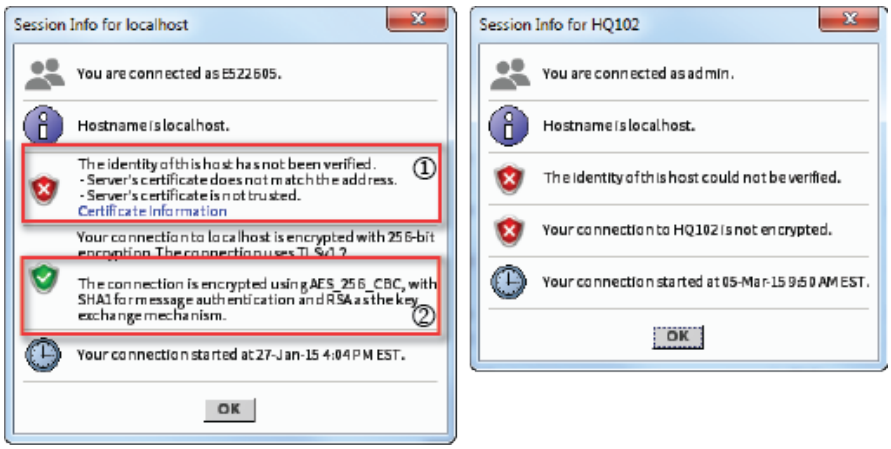
The **Session Info** window provides useful information and a graphical representation of certificate status (green and red icons).

Prerequisites: Workbench is running.

Step 1 To view session information, do one of the following:

- Click the Session Info icon ( ) in the row of icons at the top of the page.
- Right-click the station name in the Nav tree and click **Session Info**.

The system displays one of two **Session Info** windows.



- The **Session** Info message on the left indicates that you have made a secure connection.

  For the Server identity section (1), a red shield with a white X indicates that the client is unable to verify the authenticity of the Fox host. There are multiple reasons why this host may not be authentic.

  A green shield with a white check mark indicates that a root CA certificate in the client's **System** or **User Trust Store** was able to validate the signature on the server certificate, verifying the authenticity of the Fox host.

  For the Connection encryption section (2), a red shield with a white X indicates that the Fox session connection is not sufficiently encrypted.

  A green shield with a white check mark indicates that the Fox session connection is sufficiently encrypted.

  Communication is the most secure when both shields are green.

- The Session Info message on the right indicates that you have made a regular connection (a connection that is not secure). Communication is the least secure when both shields are red.

Step 2 Click the `Certificate Information` link.

The system displays the details of the Fox server certificate.

Allowed hosts management

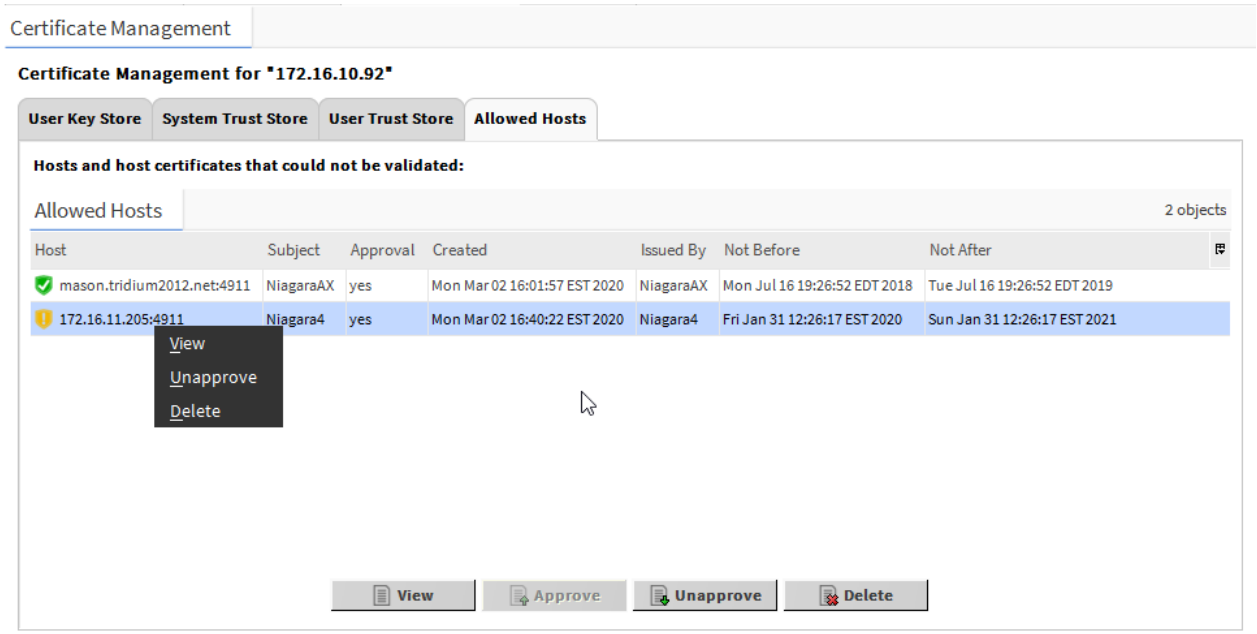
An allowed host, which you approve by accepting its self-signed certificate, makes it possible to continue your workflow uninterrupted. However, once you get started, take the time to set up signed certificates, which will prevent future hackers from compromising your system.

- To access the Workbench**Allowed Hosts** list, click **Tools→Certificate Management**, and click the **Allowed Hosts** tab.
- To access the platform/station **Allowed Hosts** list, expand **Platform** and double-click **Certificate Management** in the Nav tree. Then, click the **Allowed Hosts** tab.
- You may also access the platform/station stores by expanding **Station→Config→Services→PlatformServices** and double-clicking **CertManagerService** in the Nav tree.

More than one exemption may be allowed in your **Allowed Hosts** list. Once you have set up signed certificates for all hosts, delete the exemptions from each **Allowed Hosts** list (Workbench platform/station).

If, after you approve an exemption, its public key changes, the system negates the certificate and the green shield icon changes to the yellow shield icon with an exclamation mark ( ).

Figure 5 A changed certificate on the Allowed Hosts tab.



To approve this change, right-click the certificate row, click **View** and click **Accept** in the **Detected Public Key Change** window.

The orange text in the **Detected Public Key Change** window indicates the change.

When a certificate expires

Each root, intermediate, server, and code-signing certificate remains valid for a specific period of time (**`Valid From`** and **`Valid To`** dates). When a certificate expires, system users receive error messages.

Ensuring continued secure system access requires advance planning. There is no certificate renewal process. For each expiring certificate, you must create a new, replacement certificate, get it signed, import it into the **`User Key Store`**, and ensure that the root CA certificate used to sign it is in each station's **`User Trust Store`**. If your company uses a third-party CA, the whole process can take a couple of weeks. As a best practice, keep track of each certificate expiration date, and plan ahead to replace old certificates before they expire.

The code-signing certificate provides an exception to this rule. As long as your code-signing certificate is time-stamped, you may continue to use it even after it expires.

Additional details

- FOXS connections between stations that are using Allowed Hosts exemptions will still connect even when a certificate has expired.
- FOXS connections between stations not using Allowed Hosts exemptions will fail to reconnect. Certificates must be reissued for successful connections.
- Browser connections will start showing messages that the certificate is no longer trusted, but will still connect.
- Workbench will connect even though a certificate has expired.

NOTE: Using Allowed Hosts exemptions is not as secure as using signed certificates without exemptions. The use of signed certificates means that each certificate will need to be re-issued before they expire to avoid connection problems. It is important to note that using signed certificates without exemptions provides a much more secure environment.

Setting up alarming for certificate expiration

This procedure describes the steps to add an alarm extension on server certificate(s) under the **SecurityService** to notify you 30 days prior to expiration so that you can begin the certificate re-issue process before expiration occurs.

Prerequisites: The **SecurityService** is already present in the station's **Service** node. If not, you can add it from the **nss** palette.

Typically, server certificates have an expiration date set for two years from the date of creation (the date can vary). When the certificate expires, connection between the Niagara Network and the station fails. The expired certificate will need to be re-issued.

NOTE: The **Certificates** folder under **SecurityService** is automatically populated with any server certificates installed on the platform.

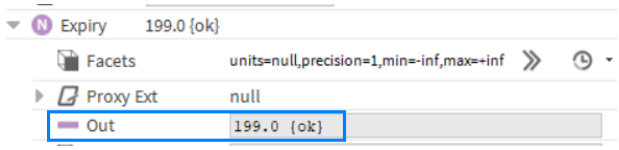
Step 1 In the NavTree, expand the station's **Config→Services→SecurityService→Certificates** node.

Step 2 Expand the folder for the certificate that you will add the alarm extension to.

Step 3 Open the **nss** palette and expand the **CertificateAlarms** folder.

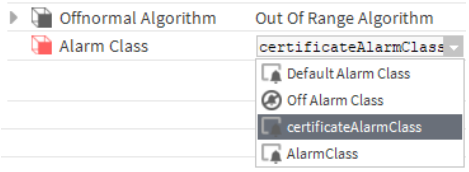
Step 4 From the palette, drag the **ExpiryAlarmExt** object onto the **Expiry** slot of the certificate.

NOTE: The **Expiry** component's **OUT** value displays the number of days remaining until the certificate will expire.



Step 5 Open a **Property Sheet** view of the **ExpiryAlarmExt** and configure the **Alarm Class** property as desired.

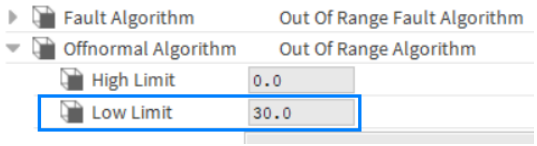
For example, you might create an **Alarm Class** (under the AlarmService) just for this purpose named **certificateAlarmClass** and configure it to route the alarm to a **ConsoleRecipient**. Shown here, the ExpiryAlarmExt is configured to use that **certificateAlarmClass**.



Step 6 Optionally, you can enter preferred alarm text in the **To Offnormal Text** property.

On completion, when an Expiry alarm is generated it is routed to the alarm class, which routes the alarm to all recipients. If routed to a Console Recipient the alarm will show up in the **Alarm Console** view 30 days prior to the certificate's expiration date. This allows sufficient time to have the certificate re-issued.

NOTE: By default, the **ExpiryAlarmExt** is configured for 30 days (fixed value) prior to expiration.



Deleting a certificate

As a general rule, third-party certificates may be renewed but not changed. Some CAs will not allow any changes once the certificate is signed. If you need to make a change, delete the certificate and start again with a new certificate.

ATTENTION: Do not delete a certificate until its replacement is in place and configured. If you delete a certificate that is in use, the platform, **FoxService** or **WebService** could fail to restart. If you have the services configured for **Https Only** a secure platform connection using TLS (Platform TLS settings) or **Foxs Only**, a missing certificate could prohibit connectivity using encrypted connections. Workbench gives no warning if you delete a certificate that is currently being used by Workbench or the platform/station.

Step 1 Connect to the platform.

Step 2 Do one of the following:

- To access the Workbench stores for managing the root CA, intermediate, and code-signing certificates, click **Tools→Certificate Management**.
- To access the platform/station stores, expand **Platform** and double-click **Certificate Management** in the Nav tree. To access the stores this way, the station must be idle.

Step 3 Select the certificate in the **User Key Store** and click **Delete**.

The system asks you to confirm the deletion.

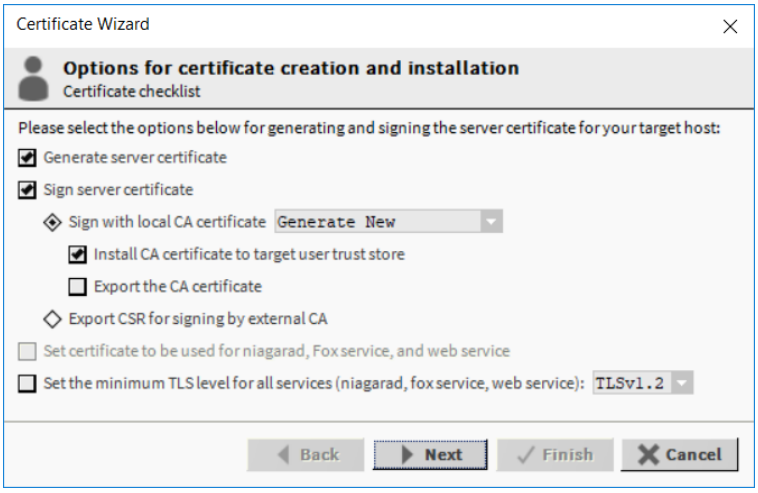
Step 4 Carefully consider your action and click **Yes** (or **No**).

Certificate Wizard

The latest Niagara version supports a **Certificate Wizard**. This wizard, available as a view on the platform root, provides a complete, continuous workflow that helps you properly set up certificates to harden a platform and station against cyber attack.

The wizard assumes prior experience in various types of certificate setup and a reasonable level of confidence in performing such procedures as are commonly done using the **Certificate Management** tool. That being the case, you will find that the **Certificate Wizard** simplifies the certificate setup process for a station by combining several steps into a continuous workflow. If you are unfamiliar with certificates, work through individual setup procedures using the **Certificate Management** tool as described elsewhere in this guide. The individual procedures will help you gain a better understanding of the steps involved.

Figure 6 Certificate Wizard platform tool with default selections



As an alternative to using the **Certificate Management** tabs to create and install a CA root certificate, the **Certificate Wizard** generates the root CA certificate and exports it with only its public key in preparation to install in a browser.

NOTE: The **Certificate Wizard** is intended to be used for a single platform at a time, not for provisioning multiple platforms.

The **Certificate Wizard** may be configured to perform some or all of the following tasks:

- Generate a new root CA certificate on the local host that can be used to sign all server certificates.
- Generate a new server certificate for a host platform.
- Sign a server certificate with a new or existing root CA certificate.
- Export a server certificate CSR (signing request) for signing by an external certificate authority.
- Install a root CA certificate into the **User Trust Store** of a platform/station selected from the daemon directory.

Once the **Certificate Wizard** generates the files, they must still be installed into all of the appropriate devices. This is accomplished via the **Certificate Management** tool found in the Workbench **Tools** menu.

Generating a CA certificate and signed Server certificate using the Certificate Wizard

This procedure describes how to use the **Certificate Wizard** workflow to complete a series of certificate-related steps for a platform and/or station.

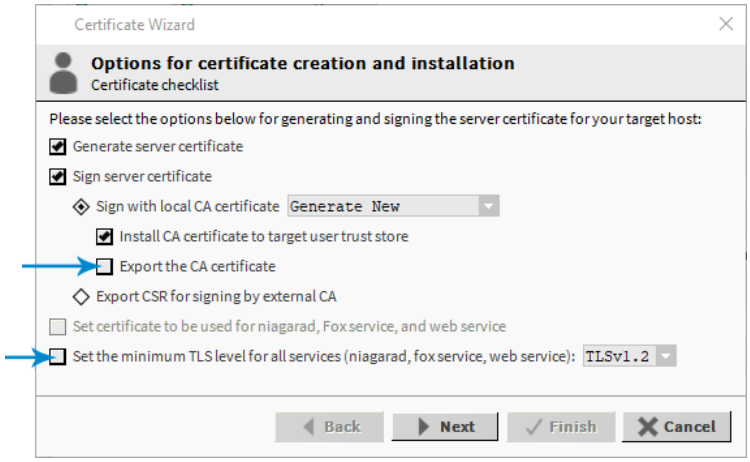
Prerequisites:

You have the required authority to create certificates. You are working in Workbench on a computer that is dedicated to certificate management, is not on the Internet or the company's LAN and is physically secure in a vault or other secure location. You have a thumb drive ready to which to copy the root CA certificate for safe keeping.

Step 1 In Workbench, open a localhost platform connection and in the **Application Director** view click **Stop** to stop any station that is running.

Step 2 In the Nav tree, right-click on the platform and click **Certificate Wizard**.

The **Certificate Wizard** window opens displaying options for certificate creation and installation.



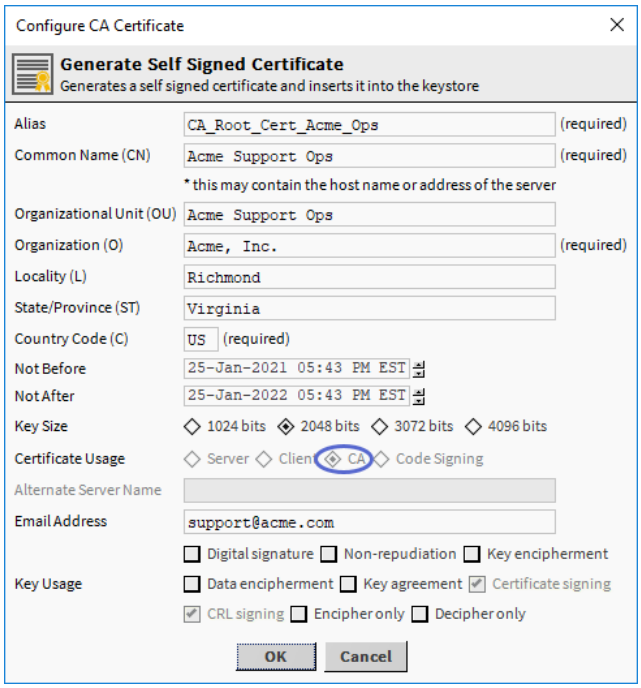
Step 3 In addition to the default selections, configure two optional properties.

- To export the root CA certificate with its private key, click on **Export the CA certificate**. It is a good idea to back up this certificate for archival storage in a secure location.
- To configure the TLS version, **Set minimum TLS level for all services→TLSv1.2**.

  NOTE:

  As of Niagara 4.13, TLSv1.0 and TLSv1.1 are still supported for backwards compatibility, but it is recommended to use TLSv1.2 and higher.

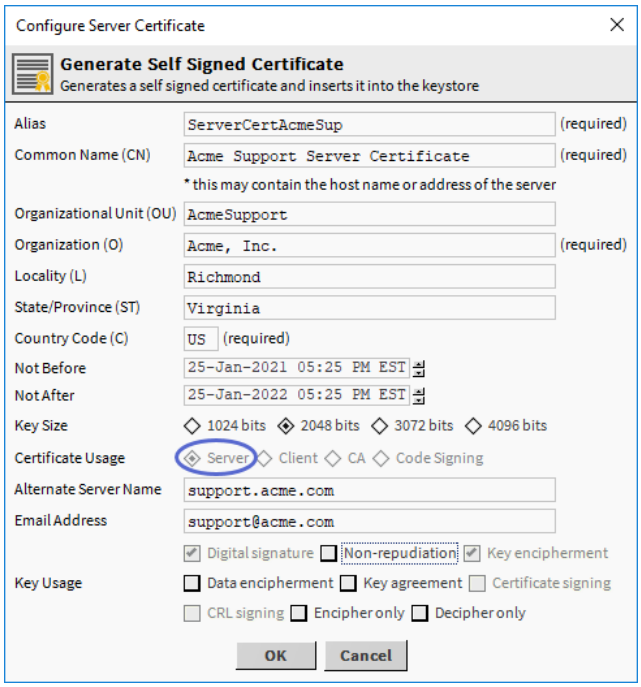
The **Configure CA Certificate** window opens for you to enter the root CA certificate information.



Step 4 In the **Configure CA Certificate** window, fill in the form, and click **OK**.

Step 5 When prompted for a **Private Key Password**, enter and confirm a strong password (minimum 10 characters, include at least one of each: a number, lowercase, and uppercase character), and click **OK**. For example, `Private123%`.

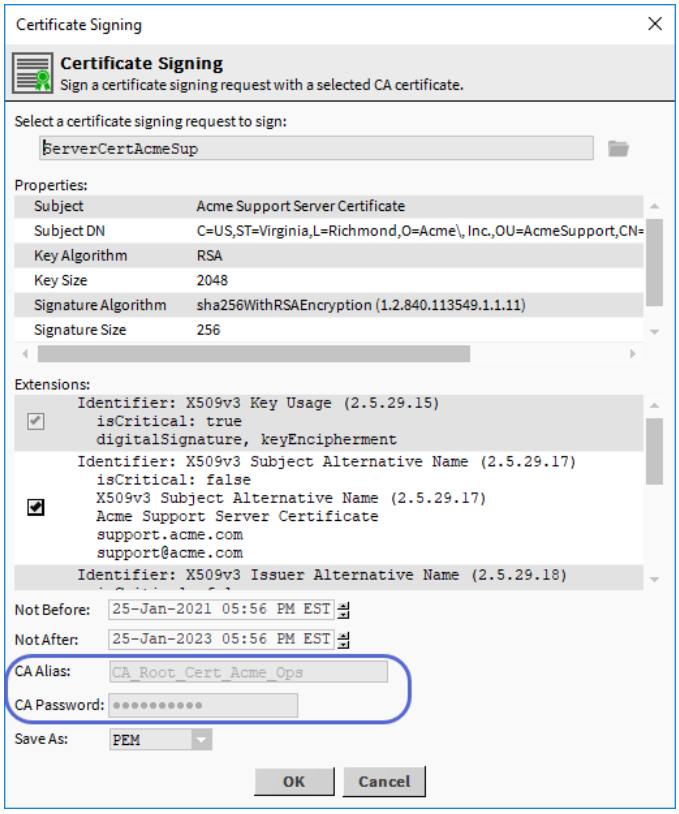
The software creates the new root CA certificate in the background. When complete, the wizard opens another **Configure CA Certificate** window. This one is for the server certificate.



Step 6 In the **Configure Server Certificate** window, fill in the form, and click **OK**.

This process generates a server certificate that is ready to be signed. The platform will never be a client, but the station will routinely function as a one, and, since the platform and the station share the same trust store, only one server certificate is required. You will need to run the wizard again when this certificate expires.

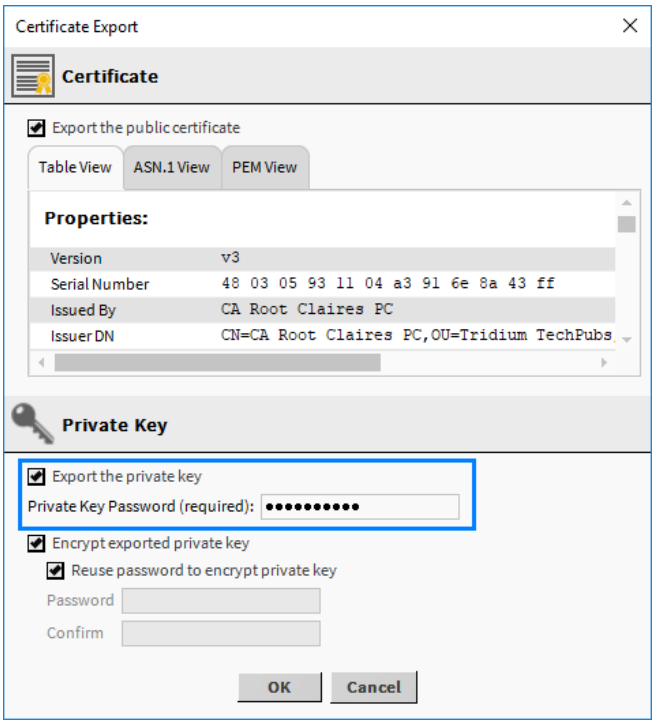
Server certificate generation occurs in the background. When complete the wizard opens the **Certificate Signing** window.



NOTE: The server certificate that is about to be signed is already selected. You cannot change the selection. Also, the root CA certificate and the CA password are already identified. There is no need to make other selections or entries.

Step 7 In the **Certificate Signing** window, review the details (similar to the example shown) and click **OK** to continue.

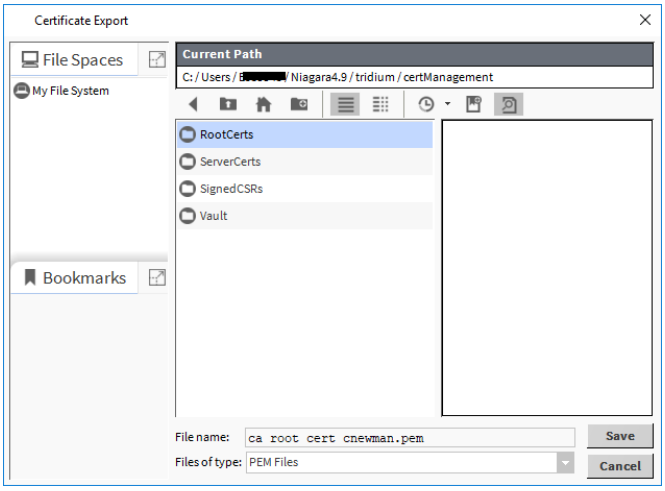
Since we did not choose to export the CSR, the wizard does not display it but proceeds directly to import the signed CSR into the Supervisor station's **User Key Store** and the new root CA certificate into its **User Trust Store**. When complete the wizard opens the **Certificate Export** window.



Step 8 In the **Certificate Export** window, in addition to the default selection, click the optional check box: **Export the private key**, enter the private key password, and click **OK**.

By default, the wizard exports the root CA certificate with only its public key. This is appropriate for distributing the root CA certificate, which must be imported to the **User Trust Store** of every platform/station throughout the enterprise, any PC that hosts an instance of the Workbench, and any browser used to monitor and control the system. You export a root CA certificate with its private key only for the purpose of backing it up to a secure location.

The wizard opens the **Certificate Export** window.



Step 9 Use the folder icon to locate the storage location for the exported root CA certificate in the local-host file system, such as an added subfolder in your `certManagement` folder (as shown) or a thumb drive, and click **Save**.

Within the `certManagement` folder, you can create subfolders for storing certificates and certificate signing requests (CSRs). In the above example, the RootCerts folder is a suitable location for

the root CA certificate with its public key, while the Vault folder simulates a secure storage location for the root CA certificate with its private key, which should be kept under lock and key.

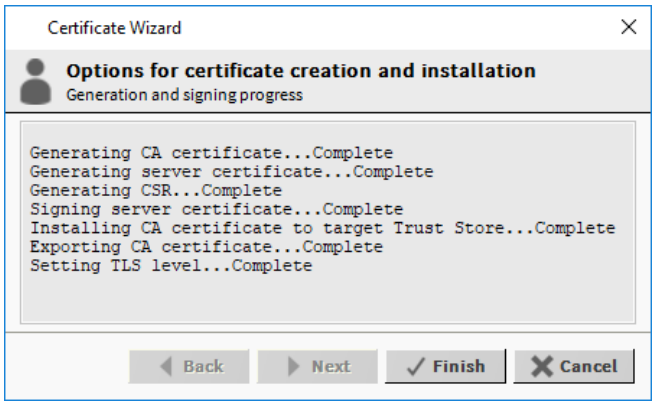
On completion, the wizard acknowledges that the export was successful.

Step 10 To continue, click **OK**.

The **Select Station** window opens.

Step 11 In the **Select Station** window, click the drop-down list of all stations in the Platform Daemon Home, and select the station to Set the TLS levels on, and click **OK**.

The wizard displays a progress summary as you complete the various steps.



Step 12 When prompted with the message, "All operations are complete", click **OK** and **Finish**.

The wizard modifies the station's .bog file in the Platform Daemon Home.

The **Certificate Wizard** successfully generated the new server certificate for the Supervisor PC, and the new root CA certificate for use in signing other server certificates. Those certificates are exported to the certManagment folder in the local file system for subsequent use. Additionally, the wizard set the TLS levels on the selected station to the selected value: TLSv1.2.

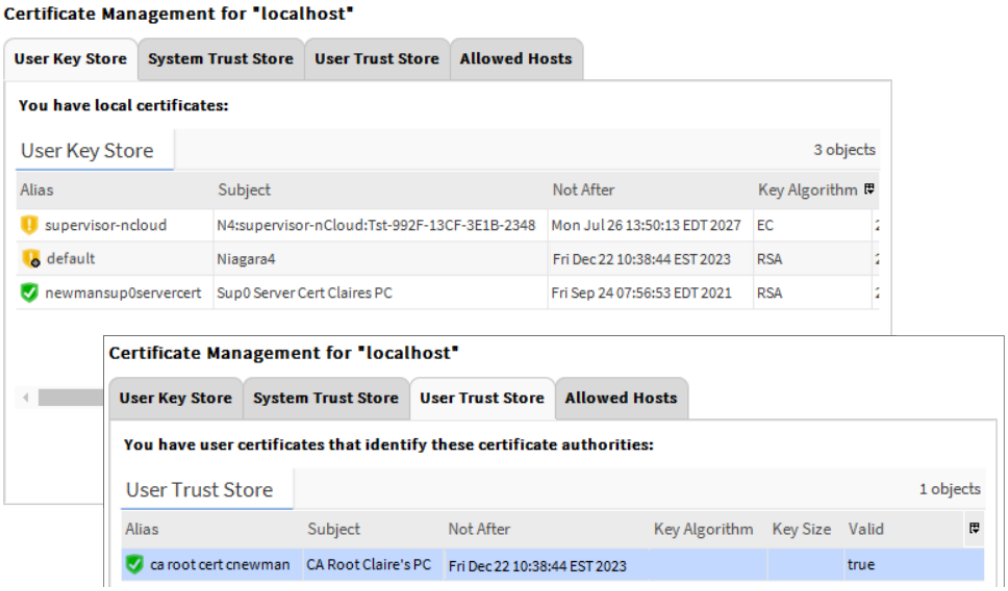
Recommended verifications

The platform and the station share the same certificate management stores, while the Workbench application has its own stores. Once you have set up certificates, confirm that the stores contain the certificates you expect.

Verify that the new certificates are installed into Certificate Management.

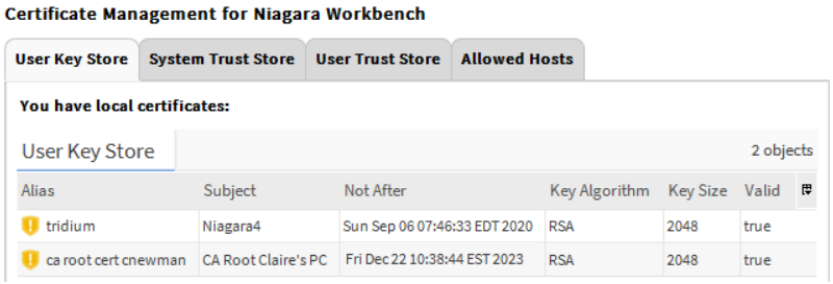
From the remote controller platform, double-click on **Certificate Management** and verify that the certificates were installed:

- The new server certificate appears in the **User Key Store**.
- The new root CA certificate appears in the **User Trust Store**. This is a copy of the root CA certificate exported with its public key.



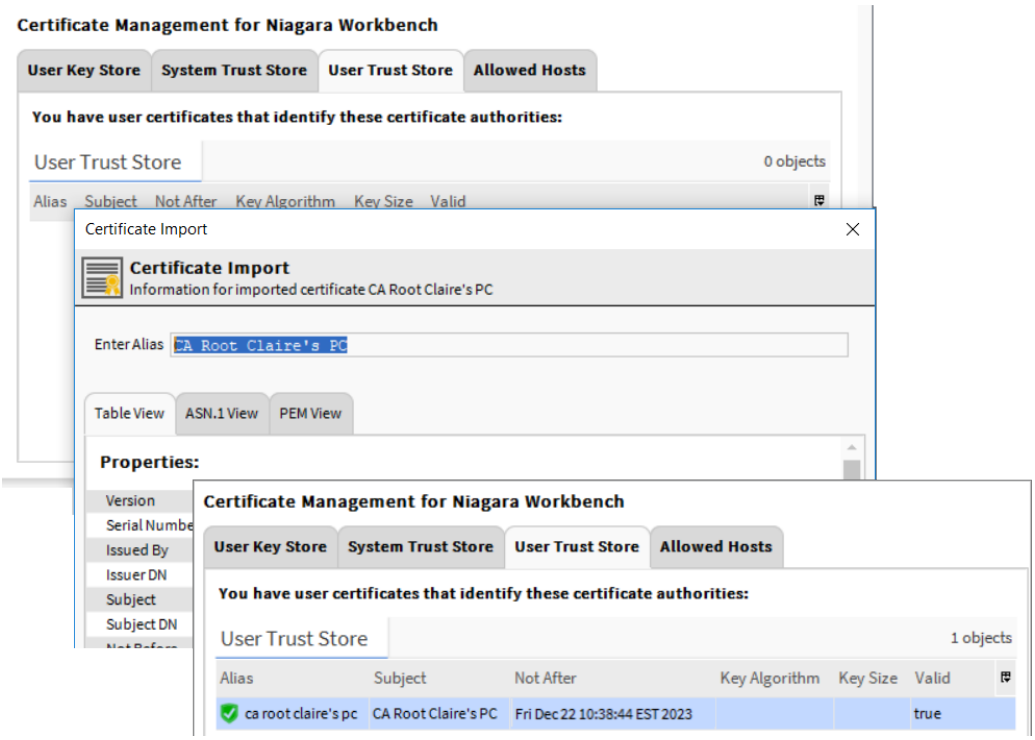
From the Workbench, click **Tools→Certificate Management**.

- Confirm that the new root CA certificate created by this instance of the Workbench (for use by the wizard) is found in the **User Key Store**. of the Workbench, alongside the self-signed **`default`** server certificate. The Workbench is a client when connecting to platforms and stations, so it is worth pointing out that this is not a Server Certificate. The root CA certificate is available to the Workbench for use in signing server certificates.



- Notice that the root CA certificate was not imported into the **User Trust Store** of the Workbench by the **Certificate Wizard**. You need to import the root CA certificate into this location to ensure that this instance of the Workbench can validate server certificates while handshaking with platforms and stations on the network.

  To do this within **Certificate Management**: Click **Import**, locate and select the new root CA certificate in `~certManagement`, and click **OK**.



Verify that the TLS level for each of the following is set to TLSv1.2:

NOTE:

As of Niagara 4.13, TLSv1.0 and TLSv1.1 are still supported for backwards compatibility, but it is recommended to use TLSv1.2 and higher.

- Platform TLS Settings — Using **Platform Administration**, view the Change TLS Settings option and verify the Protocol value.
- Station Web Service — In a station connection open a Property Sheet view on the Web Service and verify the Https Min Protocol property value.
- Station Fox Service — Open a Property Sheet view on the Fox Service and verify the Foxs Min Protocol property value.

Additional recommendations

If you are currently reading the *Niagara Platform Guide*, typically, you need to select the appropriate Server Certificate for use in secure platform and station connections.

- Set the new server certificate to be used for secure platform (niagarad) communications.
- Set the new Server Certificate to be used for secure station communications via Fox and Web Services.

For details, refer to the *Niagara Station Security Guide*.

Signing multiple certificates

Niagara 4.10 and later supports bulk certificate signing. This procedure describes how to sign multiple certificates in one operation.

Prerequisites:

- You are working in Workbench on a physically and electronically secure PC that is never connected to the Internet, and is used exclusively to sign certificates.
- The root CA or intermediate certificate that will do the signing is in the Workbench **Certificate Management User Key Store**.

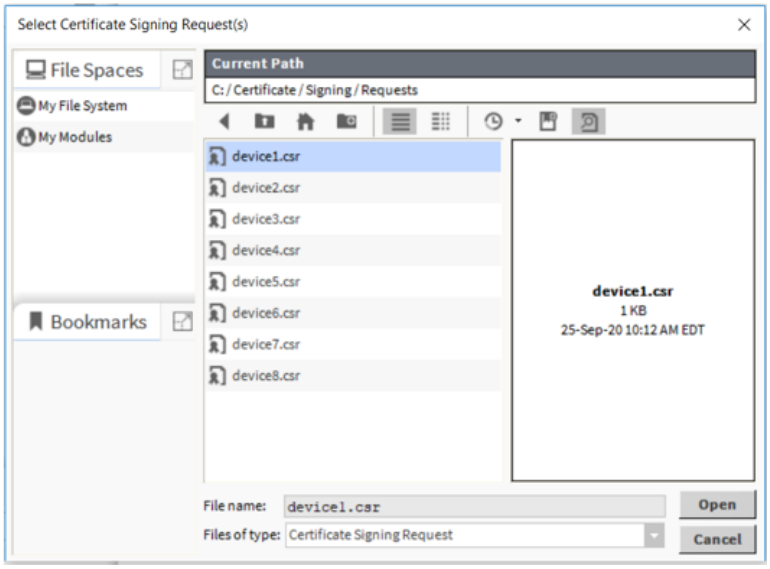
- You know the password of the CA signing certificate (root or intermediate) that will sign the certificate(s).
- You have one or more server certificate CSR files (signing requests) ready to sign.
- The "`bulkCertSigner`" license feature is enabled for your platform.

Signing certificates is the job of a CA (Certificate Authority). A variety of certificate-signing software tools are available. You are not required to use Niagara and Workbench to sign certificates. For companies who serve as their own CA, for example in a large installation, you may use your root CA certificate to sign any intermediate certificates and the intermediate certificates to sign your server and code-signing certificates. While in a small installation, you may use your root CA certificate to sign all certificates.

NOTE: To ensure network security, always sign certificates using Workbench on a computer that is disconnected from the Internet and from the company LAN. Maintain this computer in a physically secure location.

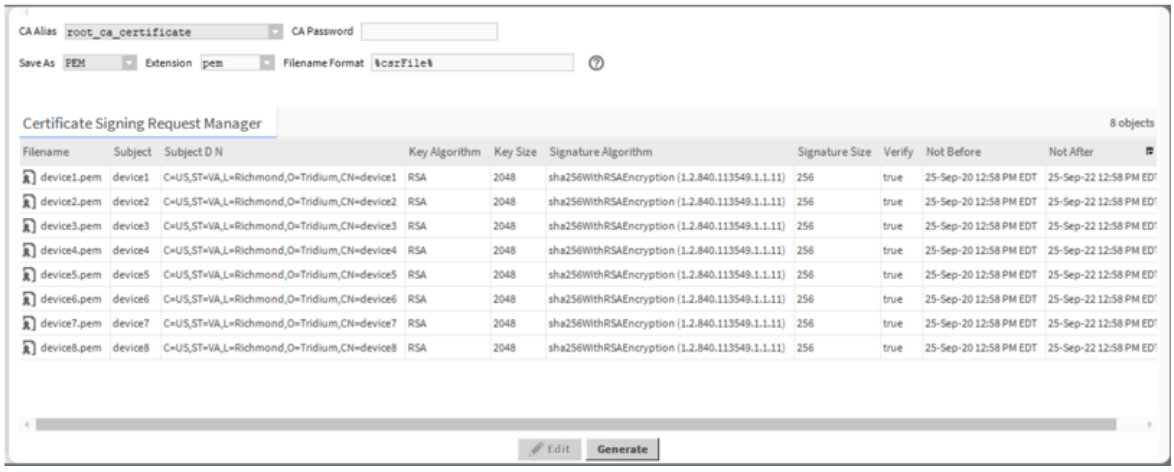
Step 1 In Workbench click **Tools→Certificate Signer Multiple Selection Tool**.

The tool opens a window to select one or more certificate signing requests (CSRs), as shown.



Step 2 In the local file system navigate to the folder containing the CSRs, select one or more, and click **Open**.

The tool window displays common properties above the table listing the selected CSRs.



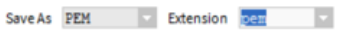
The common properties are:

- **`CA Alias`**: CA certificate in the Workbench **User Key Store** to use for signing

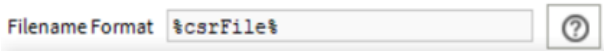
- **CA Password**: certificate password
- **Save As**: certificate type to generate from the CSR
- **Extension**: file extension to use when saving generated certificates
- **Filename Format**: filename to use pattern for generated certificates

a. In the **CA Password** field, enter the CA certificate's password.

b. In the **Save As** field, specify the type of certificate(s) to generate and the file extension to use, either by selecting from the drop-down list or by entering it.



c. In the **File Format** field, you can define an alternate BFormat string to use when generating the signed certificate filenames. By default the BFformat string for the original CSR filename is used.



Click Help button **(?)** for details on how to define alternate filenames.

Step 3 The**Certificate Signing Request Manager** is a standard Workbench manager view. If you wish, select and edit any listed CSRs as needed.

Properties that you can edit are:

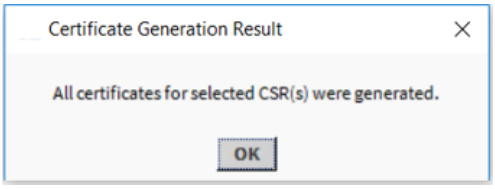
- **Filename**
- **Subject**
- **Not Before**: specified date/time
- **Not After**: specified date/time

Step 4 Once all edits are completed, click **Generate**.

Step 5 In the **Chooser** window navigate to select the directory location for saving the generated signed certificates and click **Choose**.

Step 6 In the **Confirm Generating Certificates** window, click **OK** to continue.

Step 7 In the **Certificate Generation Result** window, click **OK** to close the window.



The procedure is complete. You successfully generated one or more signed certificates which are saved to the indicated directory.

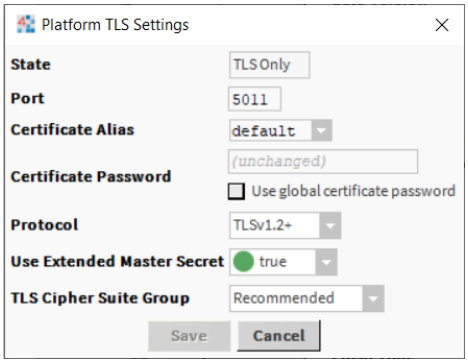
Configuring secure platform communication

Platform and station security are independent of one another. The system defaults to enabling secure communication for both platform and station. Configuring a platform (Niagarad) for secure communication (platformtls) involves confirming the port, selecting the signed server certificate to use, and, if required, restricting the TLS protocol version.

A station's window into the platform-resident secure communication features is just like any other **Platform Service** under the station's **Platform Administration** node in the Nav tree. This means that anything configured for a platform is independent of whatever station is running. Follow this procedure for the Supervisor and all remote controller platforms.

Step 1 Double-click **Platform**, double-click **Platform Administration** and click **Change TLS Settings**.

The **Platform TLS Settings** window opens.



Default settings are:

- **`State`**: `TLS only`. This can be changed on the controller to `Enable` or `Disable`.
- Daemon HTTPS **`Port`**: `5011`

  `Certificate`: `default`. If you are using a separate certificate for verifying niagarad communication, this is where you select the certificate which is already imported into the **Certificate Management User Trust Store**.
- **`Protocol`**: `TLSv1.2+`. This can be set to another version via the drop-down list or set during the certificate generation process.

  NOTE:

  As of Niagara 4.13, TLSv1.0 and TLSv1.1 are still supported for backwards compatibility, but it is recommended to use TLSv1.2 and higher.

Step 2 Configure the properties as needed and click **Save**.

Configuring secure station communication

This topic explains how to set up secure Foxs and Https communication for Supervisor and controller stations.

Follow this procedure for both Foxs and Webs.

Step 1 Make a secure connection to the station.

Step 2 Right-click **`FoxService`** or **`WebService`** under **Config→Services** in the Nav tree.

The **Property Sheet** opens and click **Views→Property Sheet**.

Step 3 Confirm that the `true` check box is selected for **`Foxs Enabled`** or **`Https Enabled`**.

Step 4 From the **`Foxs Cert`** or **`Https Cert`** list, select the appropriate certificate.

Each platform/station should have its own unique, signed server certificate. Do not use the same server certificate for more than one platform/station. If you choose to use a different certificate for your **`FoxService`** from that used for your **`WebService`**, this is where you specify it.

Enabling clients and configuring them for the correct port

While not directly related to secure communication, setting up each platform/station as a client and server is important for setting up basic communication relationships.

Step 1 If it is not already open, double-click the **NiagaraNetwork** node in the Nav tree of both the Supervisor and the controller stations.

The **Station Manager** view opens.

Step 2 Double-click the client station under the client in the **Database** pane.

For the Supervisor station, this is the controller station as client; and for the controller station, this is the Supervisor station as client.

Step 3 For each client, confirm that the **`Fox Port`** is set to `4911`, and that **`Use Foxs`** is set to `true`.

Installing a station copy on another platform

At times you may need to install a copy of a station which was created on another platform. In such cases, the station bog file passphrase most likely differ from that of the target platform, and many times the passphrase and user credentials are unknown. In that case, before you can run the station you must edit the bog file offline. This procedure describes the steps to edit the bog file offline, set a new passphrase, change the admin user password, and install the copied station on the target platform. This procedure works for a Supervisor or controller station, and assumes the copied station is provided as a ZIP file.

Prerequisites:

- The source and target controllers are running the latest Niagara version.
- You are using Workbench running on a PC.
- You have a **NiagaraStation** copy that was created on another platform and the BOG file passphrase is not known.

  NOTE: The copied station might be a single *.bog file but typically it is provided as a ZIP file containing a collection of files that may include any combination of the following: components (*.bog), templates (*.ntpl), Px files (*.px), graphic images, and etc.

Step 1 Using Windows File Explorer, extract the downloaded ZIP file and copy the station folder to your User Home stations folder for your installation, (`C:\Users\UserName\Niagara4.x\brand\stations`).

Step 2 In the Workbench **NavTree**, expand **My Host→My File System→User Home** and navigate to the station folder just copied to that location.

Step 3 Expand the station folder and double-click `config.bog` to open the file.

Step 4 On the Workbench tool bar, click (Bog File Protection tool).

The **Bog File Protection** tool allows you to enter, change, or add a bog file passphrase.

CAUTION: Any password values that are encoded with the current passphrase will be cleared. So you will need to enter/re-enter the passwords for platform users and station users.

Step 5 In the **Bog File Passphrase** window, click **Force the file to start using a different passphrase that you specify**, enter the new Passphase and Passphrase Confirm values and click **Update** and on the subsequent view, click **Close**.

Step 6 In the **NavTree**, expand **config.bog→Services**, and double-click UserService.

Step 7 In the **AX User Manager**, double-click the admin user and under Authenticator, change the Password and Password Confirm values.

Repeat this step as needed to change passwords for other users.

Step 8 Right-click the `config.bog` file and click **Save**.

Step 9 Open the **Station Copier** and copy the station from the your User Home to the target platform.

On completion, you have successfully edited the station `config.bog` offline, set a new passphrase, and changed the admin user password.

Typically, the station will autostart after copying. If not, us **Application Director** to start it. At this point you can connect to the station.

Securing email

Niagara supports secure outgoing and incoming email using TLS (Transport Layer Security).

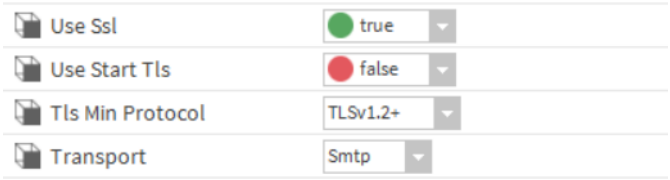
Prerequisites: The **`EmailService`** is in your **Services** container with both **`IncomingAccount`** and **`OutgoingAccount`** components. If not, add the **`EmailService`** component from the **email** palette before you begin. You may have multiple incoming and outgoing accounts, which allow you to set up connections to servers that support secure communication and others that may not.

Follow this procedure for both your incoming and outgoing accounts.

Step 1 In the station's Nav tree, right-click the **IncomingAccount** or **OutgoingAccount** node under the **EmailService** container and click **Views→Property Sheet**. The account **Property Sheet** opens.

Step 2 As of Niagara 4.13, for **`Email Authenticator`**, select the preferred email authentication type (for example, by Microsoft 365/Exchange and Gmail). For more information, see "email-IncomingAccount" and "email-OutgoingAccount" components in the Niagara Alarms Guide.

Step 3 The system provides two secure communication options:



- The default, **`Use Ssl`**, encrypts the connection before it is ever opened. To do the encryption, it automatically uses either SSL v3 or TLS (depending on email server requirements). This provides the most secure data transmission since the connection is encrypted from the start.
- **`Use Start Tls`** makes it possible to connect to an unprotected email server. The handshake occurs without encryption, then switches to encrypt the message itself.

  `Use Ssl` and **`Use Start Tls`** are mutually exclusive. Both may be `false`.
- For **`Tls Min Protocol`**, select the minimum acceptable TLS version to use.

Step 4 To provide secure email, set one property to `true`, and the other `false`.

The example shows the configuration when **`Transport`** is set to `Smtp`.

Incoming and outgoing messages use different ports for secure communication as follows:

Table 2 Email ports based on transport type

	Outgoing (SMTP)	Incoming (IMAP)	Incoming (POP3)
Not encrypted	25	143	110
Use Start Tls	587	143	110
Use Ssl	465	993	995

Not all servers follow these rules. You may need to check with your ISP (Internet Service Provider).

NOTE: Do not enable or disable the **`Use Ssl`** or **`Use Start Tls`** properties without configuring the **`Port`**.

Step 5 Change the **`Port`** to the appropriate port number (defaults are: 25 for outgoing and 110 for incoming email).

The system also provides server identity verification. For most email servers, the root certificate is already in the **System Trust Store**.

Step 6 If no root CA certificate for the email server is in the station's **System Trust Store** (third-party signed certificate) or in the **User Trust Store** (your own certificate if you provide your own secure email server), either:

- Import your own or a third-party signed root CA certificate into the station's **User Trust Store**.
- Or, if you do not have a signed certificate yet, accept the system-generated, self-signed certificate when challenged. This creates an exemption in the **Allowed Hosts** list. Later, import the root CA certificate and delete this temporary exemption.

Secure communication troubleshooting

This topic suggests solutions for common connection security problems.

When I attempt to import the signed server certificate back into the host User Key Store, I get errors.

This may happen if you deleted the original certificate created on the host from which you created the CSR. If you backed up this certificate, import it back into the **User Key Store** and import the CSR again. Generating a new certificate with the same name does not generate the same key pair and will result in errors when you attempt to import a signed certificate whose keys do not match.

For months I have been able to log in without being prompted to accept a certificate. All of a sudden the software is asking me to accept the certificate again.

One or more of the following may be occurring:

- The client may no longer contain the host's root CA certificate in the **User Trust Store** (for whatever reason). Check the certificate and import a matching root CA certificate into the **User Trust Store**.
- The root CA certificate may have expired or changed and you need to import new certificates. Check the server certificate carefully to make sure it is trusted and temporarily approve it, creating an exemption in the **Allowed Hosts** list. Create or acquire a new root CA certificate and create new server certificates. Get the new server certificates signed by the new root CA certificate. Finally, import the certificates into the appropriate stores, deleting any expired certificates and any temporary exemptions you approved in the **Allowed Hosts** list.
- There may be a problem with the Fox port. Check the **`FoxService`** on the client **NiagaraNetwork** to ensure the correct Fox port: 4911 for Foxs; 1911 for Fox.
- You may be subject to a man-in-the-middle attack and no trusted root CA certificate exists for the attacker in the platform/station **Trust Stores**. Check the certificate's `Issued By` and `Issuer DN` (Distinguished Name) carefully. Do not manually approve a certificate for an issuer you do not recognize.

The Session Info window (right-click Station→Session Info) shows a red shield with a white x in the section that reports host identity authentication.

There may be a number of reasons for this:

- If you are serving as your own Certificate Authority, a platform and station requires your root CA certificate in its **User Trust Store**. When you start the platform or station for the first time the **User Trust Store** is empty. This causes the system to generate a self-signed certificate and display it for you to approve before it establishes the connection. Compare the `Issued By` and `Subject` properties. They are the same for a self-signed certificate. If you recognize the name, you can manually approve the certificate and rest assured that communication is secure. If you do not recognize the name, do not manually approve the certificate. If you are configuring the host for the first time, import your root CA certificate to the host's **User Trust Store** as soon as possible and delete the default, self-signed certificate.
- If, in a hurry, you allowed a certificate without checking its `Issued By` and `Issuer DN` (Distinguished Name), and you are worried about what you approved, open the platform/station stores (**Config→Services→Platform Services→CertManagerService**); click the **Allowed Hosts** tab; locate the certificate and, if you do not recognize it, click **Unapprove**.

When running in a browser, the Https in the address is crossed through with a red X next to it.

This indicates that you are using a self-signed certificate for which no client certificate exists in the browser's trust store. Using Google Chrome, the browser caches nothing. You can still access the platform and station, but system performance is less than desirable. To speed performance, set up and import your own root CA certificate into the browser's trust store, or purchase and install a signed client certificate from a CA (Certificate Authority).

I enabled SSL and logged in using a secure connection, but the platform icon does not include the lock symbol. Why did the platform boot with a connection that is not secure?

Most likely there is something wrong with the certificate. If a certificate fails, or for any reason secure communication cannot start, rather than lock you out of the platform, the system enables a connection without security. Restart the platform.

NOTE: If you have to replace a platform certificate, assuming you exported the keys, you can import them to configure the new platform for secure communication.

I enabled SSL and logged in using a secure connection. The platform icon shows the lock symbol, but no communication is occurring.

A firewall or secure router may be blocking or ignoring a port. Consult your firewall or router documentation for a list of blocked ports, then either unblock the port in the firewall or router, or change the port using Workbench.

I'm using a signed server certificate, but the message "Unable to verify host identity" still appears when connecting to the platform.

The system cannot find a root CA certificate in a **Trust Store** that matches the server certificate. Import the root CA certificate used to sign the server certificate into the **User Trust Store**.

My platform or Supervisor private key has been compromised, what should I do?

Get on site as quickly as possible. Take the entire network off the Internet. Configure security again for each compromised platform creating and signing all new certificates.

When importing a root CR certificate into a client User Trust Store I get the message, "The 'Import' command encountered an error" or the certificate simply did not import.

Click the **Details** button to view the Workbench console. Investigate these possibilities:

- You may be attempting to import a private key into the **User Trust Store**. This cannot be done.

  Export the root CA certificate from the Workbench **User Key Store** without its private key and try to import it again into the client **User Trust Store**.
- The `Issuer` of the certificate you are importing must be the same as the `Subject` of the certificate that is below it in the certificate tree (the certificate used to sign the one that is causing the error). This may

be an intermediate certificate or the root CA certificate. Beginning at the bottom of the tree, the issuer-subject relationship is something like this:

Issuer, SubjectC, DB, CA, B

Where "A" is the root CA (Certificate Authority) at the root of the certificate tree. "D" is the subject of the final server certificate in the tree. The rest are intermediate certificates.

If necessary, delete the certificate, create a new certificate, sign it using the certificate below it in the trusted certificate tree, and attempt to import again.

I'm trying to get two stations to connect and it is not working.

If this is the first time you are making this connection, check the **Allowed Hosts** list. The station serving as the client may not have a certificate in its **User Trust Store** for the station that is serving as the server. In the **Allowed Hosts** list, analyze the exemption, then select the certificate to make sure that you recognize its `Issued By` and `Issuer DN` (Distinguished Name) and click **Approve**. Check the certificate for the correct name and port number in the Host column.

If you have been connecting successfully but suddenly you are unable to connect, try to figure out what changed. Check the daemon logs for an error message.

If you are using root and intermediate certificates, check the `Issuer` name on your signed intermediate and server certificates. It should be the same as the `Subject` name on the root CA certificate. When it is unable to validate the certificate tree, the software prevents communication.

We use self-signed certificates. All hosts are approved in the Allowed Hosts list, and we've been able to connect to our platforms without getting the message that our hosts are not trusted. All of a sudden we're getting that message again. What happened?

If the IP address of the platforms changed, the entry in the **Allowed Hosts** list is no longer valid.

I get the message, "Cannot connect. Ensure server is running on specified port." when I attempt to log in to a secure station:

This is a general message. A number of things could be wrong:

- There may be a problem with the controller. Ensure that the controller is connected to power and the power is on.
- You entered invalid credentials or, for some other reason, you are having difficulty logging on (the station may have stopped running). Confirm your credentials, start the platform and use the Platform Application Director to start the station, then connect again.
- Your secure **`WebService`** (Https) or **`FoxService`** (Foxs) may not be enabled (set to `true`). Both must be enabled to make a secure station connection. Make a regular station connection by clicking **File→Open→Open Station**, select `Station Connection`, provide credentials, then, on the **`FoxService` Property Sheet**, confirm that **`Foxs`** is enabled (set to `true`), close the station and connect to it again. Make sure you select a `Station TLS Connection` for **`Type`** in the **Connect** window.

Default TCP/IP ports

The various system protocols (fox, foxs, etc.) manage communication across specific ports.

This table summarizes the default TCP/IP port numbers following commissioning. You may change these ports as needed. If a firewall or router blocks a port, communication fails. Be aware of this potential and make appropriate exemption rules where necessary.

Protocol	Default port	Type of communication	Security	To change this port...
fox	1911	station	not secure	expand **Config→Services**, and double-click**FoxServices**.
foxs	4911	station	secure	
platform daemon	3011	niagarad (platform)	not secure	use the **Change HTTP Port** button under **PlatformPlatform Administration**.

Protocol	Default port	Type of communication	Security	To change this port...
platformtls (Platform Port)	5011	niagarad (platform)	secure	use the **Change TLS Settings** button under **PlatformPlatform Administration**.
http	80	browser	not secure	expand **Config**→, and double-click**FoxServices**.
https	443	browser	secure	
email, incoming account	110	receive	not secure	expand **Config**→, and double-click**EmailServices**. **NOTE:** Do not enable Use Ssl or Use Start Tls without configuring the port as indicated in this table. Refer to the *Secure Communication* chapter for more information about these ports and properties.
email, outgoing account	25	send	not secure	
email, incoming account, **`Use Ssl`**	993 (IMAP), 995 (POP3)	receive	secure	
email outgoing account, **`Use Ssl`**	465	send	secure	
email, incoming account, **`Use Start Tls`**	143 (IMAP), 110 (POP3)	receive	secure	
email, incoming account, **`Use Start Tls`**	587	send	secure	

After changing a port for an individual station, disconnect from the station, and restart the station using **`Application Director`**. If you change the fox or foxs port, when you reconnect, use **File→Open→Open Station**. This gives you the opportunity to change the default communication port: 1911 for fox and 4911 for foxs. Otherwise, you will be unable to connect.

Certificate management when replacing a controller

When replacing JACE controller in the field, you may reuse backups of the **User Key Store** and **User Trust Store** from the old controller. If no station backup is available, you must generate a new server certificate and sign it or get it signed.

Prerequisites: You are on site. Remotely importing a security backup into JACE controller is not recommended because you should not restore the **User Key Store** and **User Trust Store** while the station is connected to the Internet.

Step 1 Make sure that the JACE controller is not on the Internet.

Step 2 Reboot the controller and restore the station.

Step 3 Either restore from the station backup, or import the stores from a previously exported file.

Accepting a self-signed certificate after a change

Your system is less secure if, instead of implementing signed server certificates, you accept self-signed certificates. If, after acceptance, the self-signed certificate's public key changes, the system negates the certificate, changes the green shield icon on the **Allowed Hosts** tab to a yellow icon with an exclamation mark (), and denies access, causing an error.

Prerequisites: You are working in Workbench and are connected to the appropriate station.

If you trust the new key, follow this procedure to accept the changed certificate. If you suspect something is wrong, investigate further. Do not accept a self-signed certificate with a new public key unless you are confident that it is secure. Better yet, stop using self-signed certificates and implement signed certificates, which provide server authentication as well as encryption.

Step 1 To access an **Allowed Hosts** tab do one of the following:

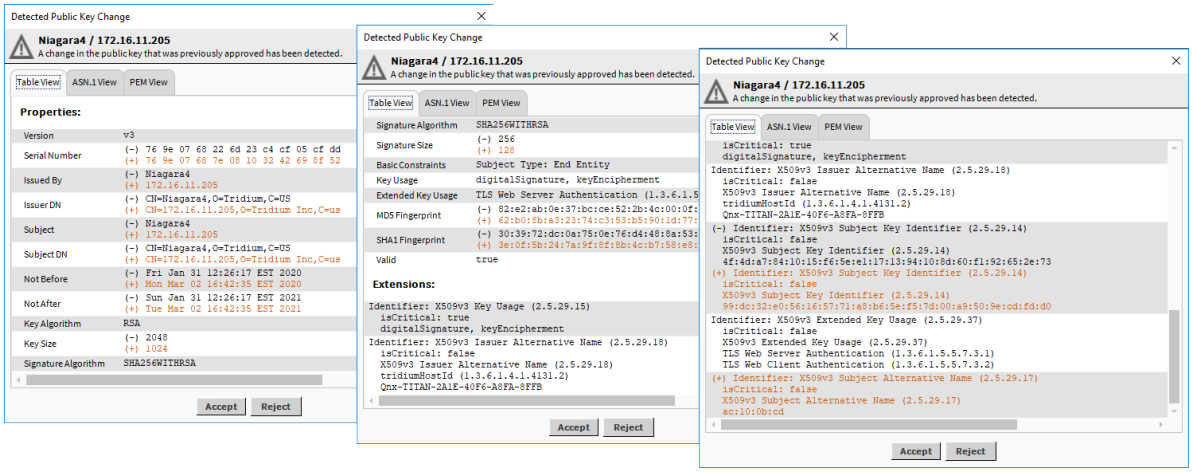
- To access the Workbench **Allowed Hosts** list, click **Tools→Certificate Management**, and click the **Allowed Hosts** tab.

- To access the platform/station **Allowed Hosts** list, expand **Platform** and double-click **Certificate Management** in the Nav tree. Then, click the **Allowed Hosts** tab.
- You may also access the platform/station stores by expanding **Station→Config→Services→-PlatformServices** and double-clicking **CertManagerService** in the Nav tree.

The tab opens.

Step 2 To confirm that the public key changed, select the certificate row in the table and click **View**.

The certificate opens in the **Detected Public Key Change** window.



The screen captures show an example certificate after scrolling down to the mid-scroll and end-scroll regions.

Step 3 Confirm at least the **`Issued By`** and **`Subject`** properties.

The two names should be names you recognize as belonging to your company.

Step 4 Accept the self-signed certificate with the new public key, click **Accept**.

The certificate icon changes to a green shield with a check mark.

Chapter 3 User authentication

Topics covered in this chapter

♦ User authentication checklist
♦ Authentication schemes
♦ Client certificate authentication setup
♦ Logging in via a browser using client certificate authentication
♦ Logging in via Workbench using client certificate authentication
♦ Enabling a kiosk-like mode using ClientCertAuth
♦ Setting up Google authentication
♦ Single Sign On
♦ Users
♦ Assigning authentication schemes to users
♦ Password management
♦ Logging on to a station
♦ Station Auto Logoff
♦ Changing your password
♦ User authentication troubleshooting

User authentication validates the identity of a subject, which can be a human user, a system, or an application. The **AuthenticationService** is designed to be extensible by supporting a variety of authentication schemes. In addition, the **gauth** palette (Google Authenticator app) provides a two-factor mechanism that requires a user to enter their password as well as a single-use token to authenticate.

All stations must have an **AuthenticationService**, with the **Authenticator** property for each user set to one of the supported schemes.

When a station attempts a connection, it checks the user's login credentials: user name, password, and token (if using the Google Authenticator app) against the users under the station's **UserService**. This process is called *user authentication*. The actual process depends on the authentication scheme and on the type of connection:

- Workbench-to-station (**FoxService**)

  When a user opens a station (**File→Open→Open Station**), Workbench prompts for user name and password (and token if using the Google Authenticator app). When using Niagara 4, this type of authentication defaults to the **DigestScheme**. Connections to older software versions (NiagaraAX) default to the **AXDigestScheme**.

- HTTPs browser-to-station (**WebService**)

  When a user opens a station from a browser, the system prompts for user name and password (and token if using the Google Authenticator app). The authentication mechanism used depends on the scheme selected in the **AuthenticationService**.

- Station-to-station (**FoxService**)

  As for Workbench-to-station connection, a station-to-station connection requires an assigned authentication scheme and a pre-configured user name and password. The role assigned to a station user (machine-to-machine communication) should grant only the permissions needed by the accessing station.

User authentication checklist

As the system administrator, use this checklist to verify that you completed all required tasks to set up user authentication.

☐ Connections are secure (https rather than http; foxs rather than fox).

☐ The authentication scheme has been selected for each user.

☐ User roles have been identified. You need to determine what each user can do with each component in the system. Objects to protect are components, files, and histories. Each of these is assigned a category.

☐ Roles have been created and assigned to each user. This assignment grants permission for the user to access each category of object. The user's role defines exactly what each user can do with each object in the system.

☐ If you are using client user authentication, each user has created a client certificate and submitted their certificate with its public key.

☐ If you are using the Google Authenticator, the app has been installed on the user's mobile device.

☐ Each user has been created.

☐ The audit log has been set up for later analysis.

Authentication schemes

An authentication scheme verifies that a user is authorized to access a station. All authentication requests are routed through the system's **AuthenticationService**.

These default authentication schemes are provided as standard components of the **AuthenticationService**:

- `DigestScheme` never sends a user password directly to the station. Instead, it sends proof that the user knows the password. This scheme connects a Niagara 4 Supervisor to a Niagara 4 station.
- `AXDigestScheme` passes several messages back and forth to prove that the client knows the password. It never actually transmits the client's password, which helps protect the system if another layer of security, such as secure TLS communication, fails.

  This scheme provides compatibility with stations running previous software versions. Stations running NiagaraAX must have been upgraded with the following security updates: 3.8, 3.7u1, 3.6u4, or 3.5u4. This scheme allows a Niagara 4 supervisor to connect to a NiagaraAX station.
- `HTTPBasicScheme` performs HTTP-Basic authentication using standard HTTP headers. It only works via the web, and is intended for clients that cannot use cookies. This authentication scheme sends the user name and password over the connection. This component is located in the **`baja`** palette.

Both the **DigestScheme** and **AXDigestScheme** use SCRAM-SHA (Salted Challenge Response Authentication Mechanism) to secure the transmission of clear-text passwords over a channel protected by TLS (Transport Layer Security). This authentication mechanism conforms to the RFC 5802 standard as defined by the IETF (Internet Engineering Task Force). It is the same mechanism for both schemes. The main difference between the two schemes has to do with the order of operations that is required to support the differences between NiagaraAX and Niagara 4.

Additional schemes

A station can support more than one authentication scheme. Schemes may be added to or removed from the **AuthenticationSchemes** container in the **AuthenticationService** under the **Services** container.

Each user account is associated with a specific scheme. This allows some user accounts to use one scheme, while other accounts use different schemes. For example, a digest scheme is appropriate for human users, whereas a Certificate or HTTP-Basic scheme is more appropriate for devices. The system supports only schemes that have been added to the **AuthenticationService**.

CAUTION: Deleting a scheme may leave your users with an invalid reference to a non-existent scheme.

Additional schemes are in the **`baja`**, **`ldap`**, and **`saml`**, and **`clientCertAuth`** palettes. Other schemes may be found in other palettes, and developers may create new authentication schemes. Third-party schemes may also be available.

The following schemes (which require the use of an LDAP server and additional properties, which must be configured) are in the **`ldap`** palette:

- LdapScheme

- KerberosScheme

Client Certificate Authentication Scheme

In Niagara, the **clientCertAuth** palette contains the **ClientCertAuthScheme**, which provides authentication using a client certificate. Each user's client certificate with its public key is directly bound to the user object. Each certificate's public key matches its private key. During a login attempt, the system prompts the user to upload his or her certificate with its private key. To authenticate the user, the system verifies that the private key matches the public key stored on the user object.

Google Authentication Scheme

The **gauth** palette contains the **GoogleAuthenticationScheme** (Google Auth Authenticator), which provides two-factor authentication using a password and single-use token sent to the user's mobile phone. The authenticator app is time based and automatically updates the tokens every 30 seconds.

In addition to adding the **GoogleAuthenticationScheme** to the standard **AuthenticationService**, this scheme requires that the Google Authenticator app be installed on the user's phone.

SAML Authentication Scheme

The **saml** palette contains the **SAMLAuthenticationScheme**, which can be added to the **AuthenticationSchemes** container to configure the station for SAML SSO (Single Sign On). Authentication schemes that support SSO allow supported users to bypass entering a username in the pre-login step. Instead, the users are redirected to an alternate login page.

Client certificate authentication setup

Setting up client certificate authentication is a multi-step process where some procedures must be performed by the station administrator, and other procedures by the user.

The station administrator's part in the process is to obtain several pieces of information from the local IT network administrator and then configure the user for client certificate authentication.

The user must complete several tasks as well as provide the station administrator with the client certificate and its public key.

Refer to the following administrator and user workflows for the list of procedures for each.

Admin workflow for client certificate authentication

The station administrator performs this workflow to configure the station for client certificate authentication.

Client authentication is a method for users to securely access a remote station (via a browser) by exchanging a client certificate with the remote station. The certificate effectively represents a user identity and handles logging-in and authenticating to the station.

Only the user (client) should have access to the certificate with the private key for client certificate authentication. However, the certificate with the public key can be emailed and shared. For this workflow, the station administrator first gets the user's client certificate with its public key, creates the station user account, and assigns the certificate with the public key in the server authenticator. The following procedure details this configuration method.

Configuring a user for client certificate authentication

A station admin user performs this procedure to configure a new user account to use the ClientCertAuthScheme, and assign the user's certificate with its public key to the user's ClientCertAuthenticator.

Prerequisites:

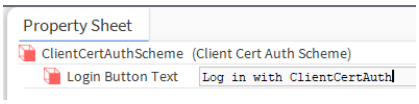
- You are working in a properly licensed Workbench installation.
- You have already acquired the certificate with the public key created by the user.

This chapter provides a separate workflow for the user. It describes how to create a client certificate, export it with its public, then its private key, and install the certificate in a browser.

Step 1 In the Workbench Nav tree, expand the station's **Services→AuthenticationService→AuthenticationSchemes** node.

Step 2 Open the **clientCertAuth** palette and drag the **ClientCertAuthScheme** to the **AuthenticationSchemes** folder.

Step 3 Expand the **AuthenticationSchemes** and double-click the **ClientCertAuthScheme** to open the **Property Sheet** view, and edit the default **Login Button Text** as needed.



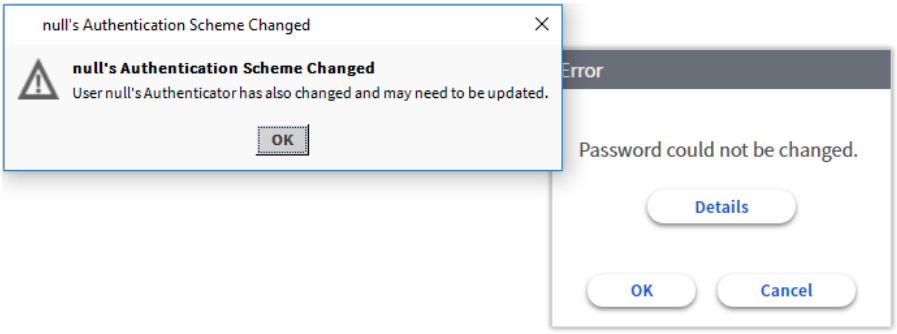
This login button is added to the login window for a browser station connection (in addition to any SSO login buttons for other configured SSO schemes).

Step 4 To create a new user, double-click **UserService**, and in the **User Manager** click **New** .

Step 5 To accept default entries for **Type to add** and **Number to add**, click **OK** in the configuration pop-up window.

Step 6 In the second configuration window. enter user details (include a password otherwise you will be prompted to enter one), click the **Authentication Scheme Name** drop-down list, select the **ClientCertAuthScheme**, and click **OK**.

At this point, you may see the following messages. If so, disregard the messages, click **OK** to close each popup window, and continue with the next step.

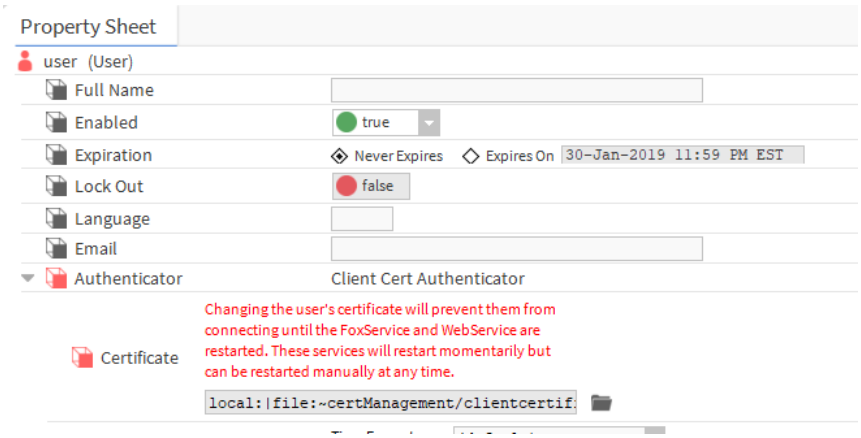


The new user is added in the **User Manager** view.

Step 7 Double-click the new user to open a **Property Sheet** view, and click to expand **Authenticator**.

Step 8 Under **Certificate**, click **Choose File** to open a File Chooser window, browse to locate and select the user-provided public certificate (`*.pem`) file and click **OK**.

A notice appears alerting you that the user's certificate change will prevent them from connecting until the **FoxService** and **WebService** are restarted.



Step 9 Click **Save**.

The **Save** action triggers a timer to restart the Fox and Web services in two-minutes. You can also restart the services manually. The restart is necessary for your changes to take effect.

After this configuration is successfully completed, when the user attempts to log in to the station via a browser, the browser first prompts the user to select the private certificate to use to authenticate to the station. Next, the browser displays the station pre-login window where the user clicks the **Login With ClientCertAuth** button and immediately authenticates to the station. There is no need to enter username and password credentials. For more details, refer to the procedure "Logging in via browser using client certificate authentication".

User workflow for client certificate authentication

A user performs this workflow as a preliminary step prior to the station administrator setting up for client certificate authentication.

The user generates a new client certificate and exports it with its public key, and again with its encrypted private key.

Afterwards, the user gives the certificate with public key to the station administrator who uses it to set up Client Certificate Authentication on the station.

The user saves the exported certificate with its encrypted private key in a safe location on our PC file system for later login use. The private key is sensitive information and should be kept well protected. As a user, do not store your certificate with its private key somewhere where it might be accessible to others.

In a subsequent procedure, the user installs the certificate with its private key in the web browser.

Creating a client certificate

This procedure uses the Workbench **Certificate Management** tool rather than the station's certificate stores to generate a client certificate for client certificate authentication. By generating the certificate this way, the certificate resides in the Workbench stores.

Prerequisites:

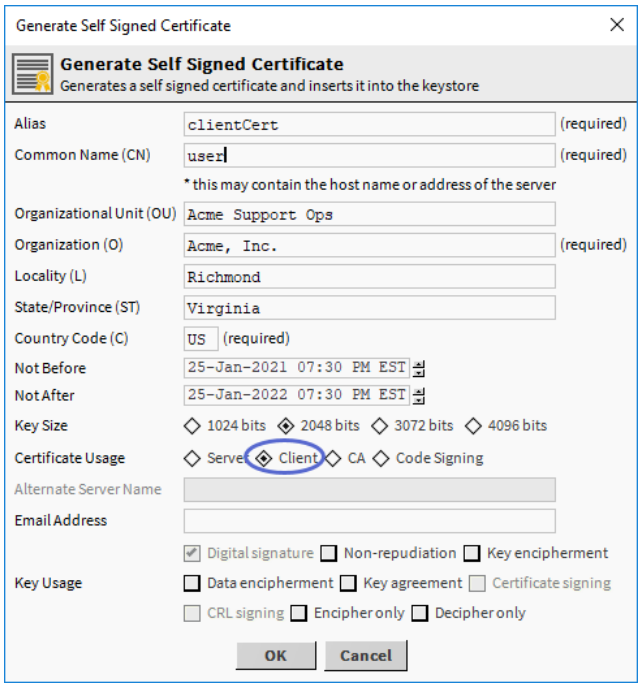
- You have the required authority to create certificates.
- You are running Workbench on your PC.

NOTE: Those end users who do not have Workbench will need to use some other tool (for example, OpenSSL) to generate a client certificate.

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

Step 2 Click the **New** button at the bottom of the view.

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



Step 3 Give the certificate at least the required information**Alias**, **Common Name (CN)** , **Organization**, and **Country Code**.

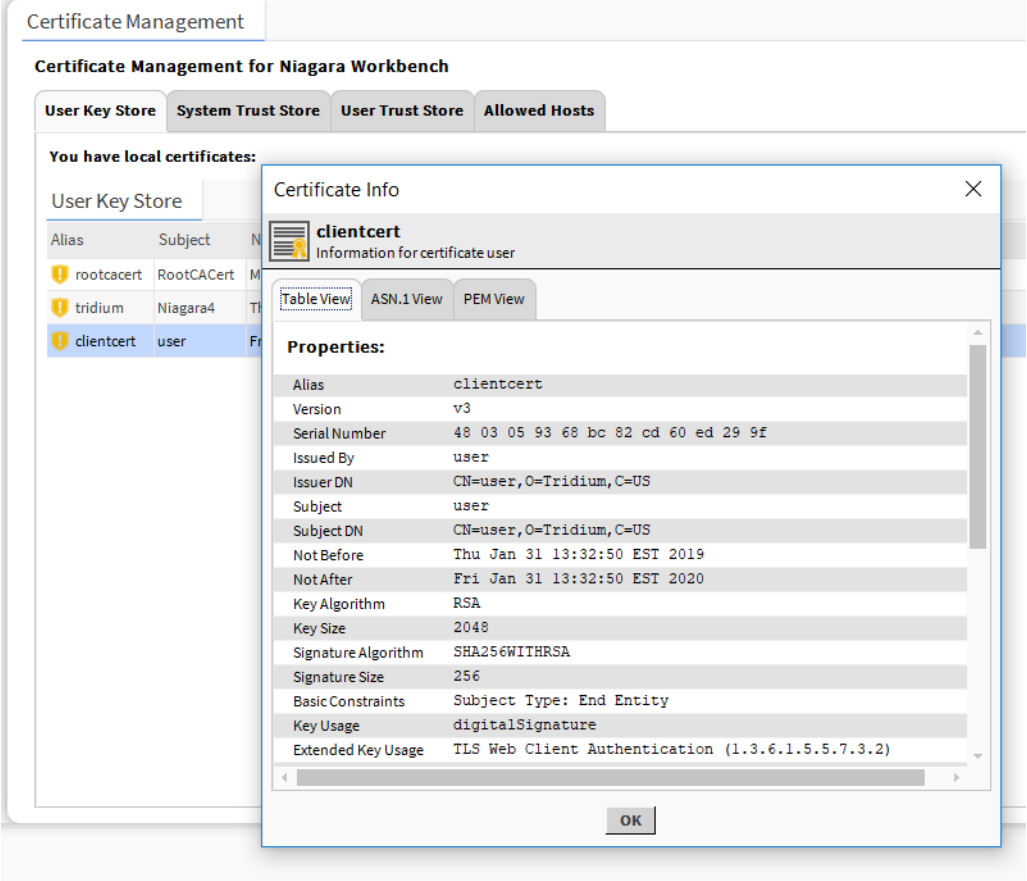
- Use **Alias** to identify this as a client certificate.
- Entering your station username in the **Common Name** property facilitates later authentication on the station.
- The two-character **Country Code** is required and must be a known value, such as: US, IN, CA, FR, DE, ES, etc. (refer to the ISO CODE column at countrycode.org).

Step 4 For **Certificate Usage**, select **Client**.

Step 5 When you have filled in the required fields, click **OK**.

The system submits the certificate for processing in the background. A pop-up window in the lower right portion of your screen advises you regarding the time it may take to generate the certificate. The length of time it takes depends on the key size and the platform's processing capability.

When created, the certificate appears as a new row in the **User Key Store** table.



The next part of the workflow is to export this client certificate in two different formats: public and private.

Exporting a client certificate

This procedure describes the steps to export your client certificate in two formats: public key and private key. The certificate with **Public** key is not considered protected data, you can share it as needed. By contrast, the certificate with an encrypted **Private** key is protected data, for your use only. It is part of your digital identity, and should be kept in a safe location, not accessible by anyone else.

Prerequisites:

- You are running Workbench on your PC.
- You are logged in to the station.
- You have already generated a client certificate, which places it in your certificate **User Key Store**.

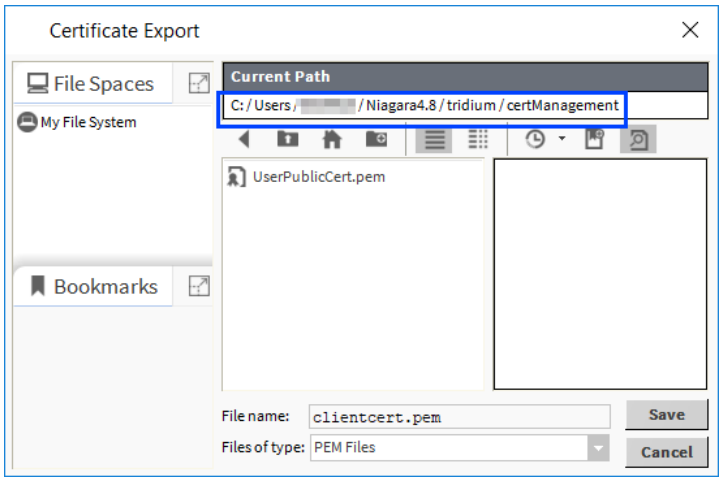
Step 1 In Workbench, open the **Certificate Management** view.

Step 2 On the **User Key Store** tab, select your client certificate and click **Export**.

The system opens the **Certificate Export** window.

Step 3 To export the **Public** certificate, just click **OK** (do not select `Export the private key`).

A second **Certificate Export** window opens.



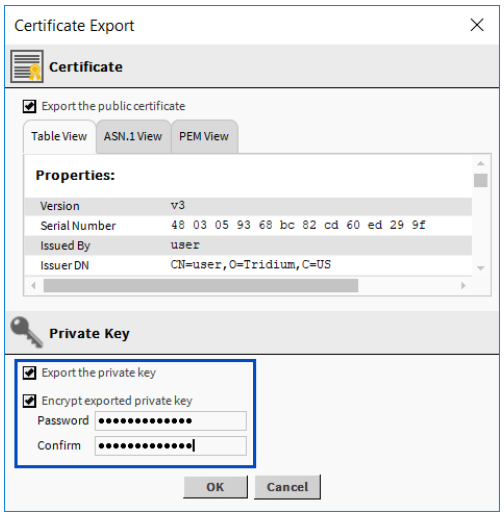
Step 4 Use the default location on your PC's file system (or navigate to another location) and click **Save**.

The system confirms that the certificate export was successful.

Step 5 To close the confirmation window, click **OK** and proceed with the remaining steps to export the certificate with its **Private** key.

Step 6 On the **User Key Store** tab, where your client certificate is still selected and click **Export** a second time.

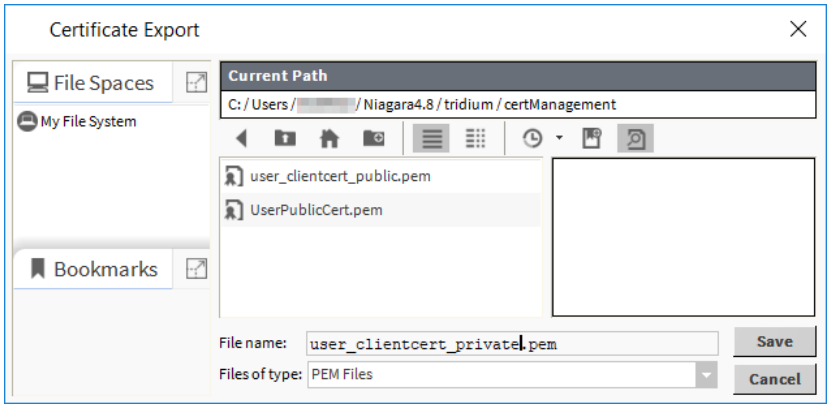
The **Certificate Export** window opens.



Step 7 This time in the **Certificate Export** window, select `Export the private key`, under `Encrypt exported private key`, create a strong password and click **OK**.

NOTE: Be sure to make note of this password, and keep it in a secure place. Later, when authenticating to a station using the client certificate, you will be prompted to enter this private key password.

The second **Certificate Export** window opens.



Step 8 Use the default location on your PC's file system (or navigate to another safer location) and click **Save**.

The software saves your public and private client certificates as `*.pem` files to the `~certManagement` folder in your User Home, or in the location you selected during the export. Make sure this location is safe, and not accessible by anyone else.

The system confirms that the certificate export was successful. To close the confirmation window, click **OK**.

Step 9 Give the public certificate file to the Station Admin who will use it in setting up Client Certificate Authentication on the station.

In a separate procedure, you will install the private certificate file in your browser trust store for use when logging in to the station.

NOTE: Not all browsers will accept private certificate files in `*.pem` file format. Instead, they require other formats (`*.pfx`, `*.p12`, etc.). If your browser requires other than `*.pem` files, conversion tools (e.g. OpenSSL, etc.) are readily available which you can use to convert your private certificate file to the required format.

Installing the private client certificate in your browser trust store

This procedure describes the steps to install your client certificate with its **Private** key in your browser's certificate trust store. When a station a station is configured for client certificate authentication, its login references this certificate via the browser.

Prerequisites:

- You have previously generated a client certificate and exported it with its encrypted private key.
- If needed, you have used a third-party conversion tool to convert your private certificate `*.pem` file to the certificate file format required by your browser (for example, `*.pfx` or `*.p12` /`*.pkcs12`).

This procedure describes installing the private certificate in the Chrome web browser certificate stores using the Certificate import tool available there. The procedure varies somewhat depending on which browser you use, but it should be similar to what is described here.

Step 1 In the Chrome browser's **Settings** view, click on the **Main Menu** icon (upper left) and click **Advanced→Privacy and security**.

Step 2 On the **Privacy and security** view, scroll down and click **Manage certificates**.

Step 3 In the **Certificates** window on the **Personal** tab, click **Import** and follow the prompts in the **Certificate Import Wizard** to import your converted private certificate file to the browser trust store.

Use the default file location indicated by the import tool. For example, by default the import tool in Chrome indicates the Personal trust store location.

Your private certificate is successfully installed in your browser's certificate trust store.

At this point, you have completed the user workflow for client certificate authentication. If the station is properly configured, you should be able to log in via browser (or Workbench) using client certificate authentication.

Logging in via a browser using client certificate authentication

This procedure describes the steps to login via your browser to a station configured for client certificate authentication.

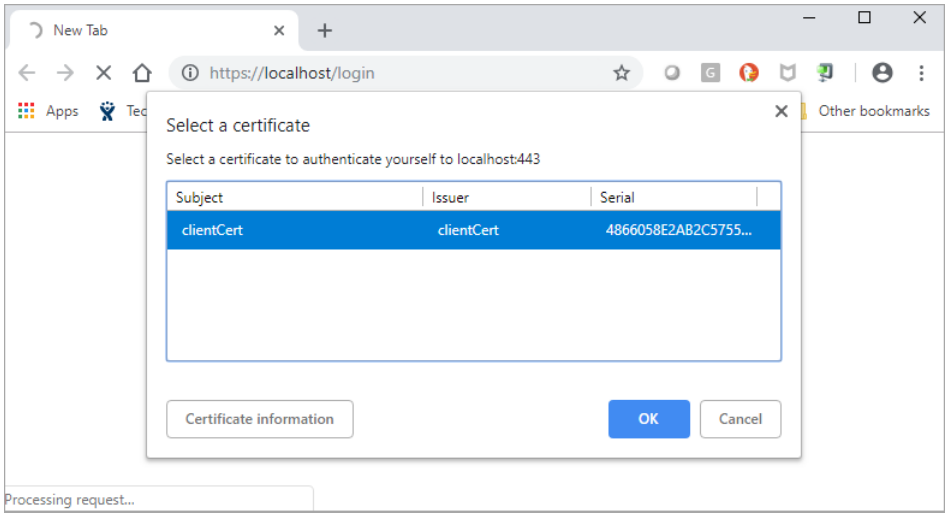
Prerequisites:

- You are running Workbench installation
- You have previously generated a client certificate (your **public** certificate) and given this *.pem file to the station admin.
- The station admin has already configured your user account for ClientCertAuthentication.
- You have also previously generated a client certificate with private key (your **private** certificate) and saved it to your PC file system.
- You have already installed your private certificate in the browser's certificate trust stores.

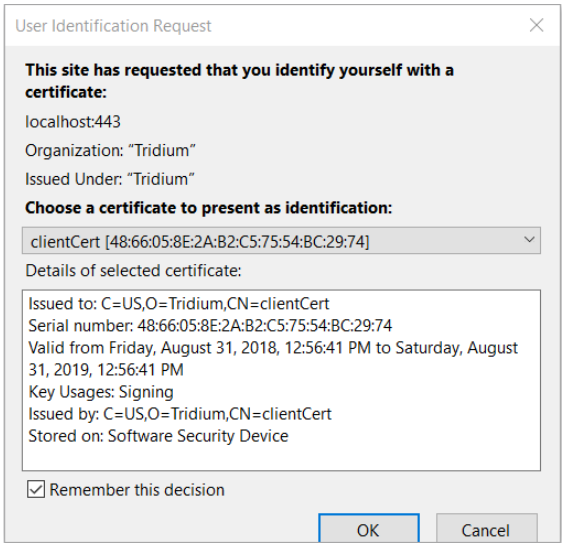
Step 1 In the browser, enter the station address and press Enter.

The browser opens a window, prompting you to select a certificate to authenticate yourself to the station.

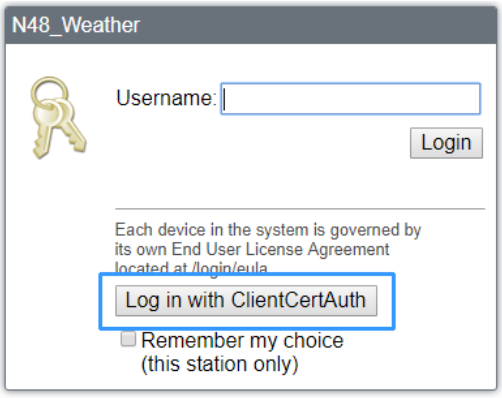
Step 2 In the **Select a certificate** window, click to select your private certificate and click **OK**.



NOTE: Different browsers will present different dialogs, but all browsers will allow you to select a certificate to identify yourself. For example, the **Select a certificate** dialog (above) is seen when using Chrome, while the **User Identification Request** dialog (below) is seen when using Firefox.

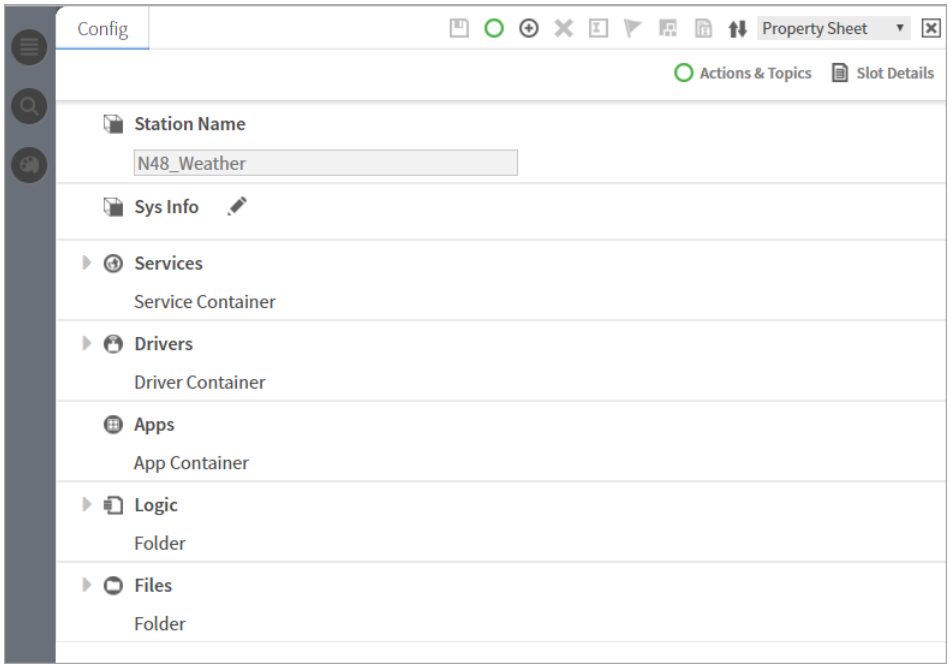


Step 3 In the station **Pre-Login** window, simply click the **Login With ClientCertAuth** button.



NOTE: There is no need to enter username/password credentials.

Upon clicking the button, you immediately authenticate to the station.



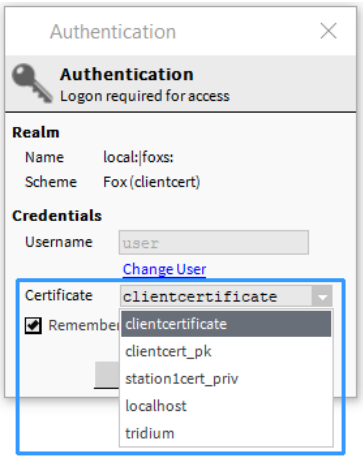
Logging in via Workbench using client certificate authentication

This procedure describes the steps to login via your Workbench to a station configured for client certificate authentication.

Prerequisites:

Step 1 In Workbench, initiate the station connection.

Step 2 When prompted, select your private certificate from the dropdown list.



NOTE: There is no need to enter username/password credentials.

Step 3 Click **OK** to continue.

You are immediately authenticated to the station.

Enabling a kiosk-like mode using ClientCertAuth

In Niagara, you can use the Client Certificate Authentication feature to facilitate a "kiosk-like" application. This would be useful for the purpose of providing an information display in a lobby, or an operator terminal in a mechanical room, where the browser automatically connects and authenticates to the station without user interaction. This procedure is performed by the Station Admin.

Prerequisites:

- A Niagara station running with the **AuthenticationService** already configured for Single Sign On.
- Admin privileges adequate for certificate management and creating/configuring station users.

Step 1 In a **Property Sheet** view of the station's **Services** container, click to expand **AuthenticationService→SSO Configuration** and confirm that the **`Auto Attempt Single Sign On`** property is set to **false**.

This allows authentication to bypass the automatic SSO logon prompt when a user access the station.

Step 2 Follow the workflows provided to "Set up client certificate authentication" (described in the Station Security Guide, User Authentication chapter).

NOTE: You will need to complete the procedures for both the Admin and User workflows for client certificate authentication. You will be creating a client certificate for a new user for the kiosk-like mode on this station, and you will also configuring this user for client certificate authentication.

Step 3 In the NavTree, double-click on the UserService to open the **User Manager** view, and click **New** to create a new station user (e.g., "kioskUser"), and configure the new user as follows:

a. For **`Auto Logoff Enabled`**, click the checkbox to deselect (disable) it.

b. For the **`Authentication Scheme Name`** click the dropdown list and click to select ClientCertAuthScheme

c. For **`Password`**, enter the required Private Key Password for the user's client certificate.

Step 4 In **User Manager** view, click the **Views** dropdown list and click on **Permissions Browser**.

Step 5 In the **Permissions Browser** expand folders and confirm that this new user has a limited permissions set, appropriate for this kiosk-like mode.

Setting up Google authentication

The Google Authentication Scheme is a two-factor authentication mechanism that requires the user to enter their password as well as a single-use token when logging in to a station. This protects a user's account even if their password is compromised. This authentication scheme relies on TOTP (Time-based One Time Password) and the Google Authenticator app on the user's mobile device to generate and verify single-use authentication tokens. Google authentication is time based, so there is no dependency on network communication between the user's mobile device, the station, or external servers. Since the authenticator is time based, the time in the station and time in the phone must stay relatively in sync. The app provides a buffer of plus or minus 1.5 minutes to account for clock skew.

Prerequisites: The user's mobile phone requires the Google Authentication app. You are working in Workbench. The user exists in the station database.

Step 1 Open the **`gauth`** palette and add the **GoogleAuthenticationScheme** to the **Services→AuthenticationService→AuthenticationSchemes** node in the Nav tree.

Step 2 Right-click **UserService**, and double-click the user in the table.

The **Edit** view for the user opens.

Step 3 Configure the **`Authentication Scheme Name`** as needed and click **Save**.

Step 4 Click the button next to **`Secret Key`** under the user's authenticator and follow the prompts.

Step 5 To complete the configuration, click **Save**.

Depending the view you are using, you may have to open the user again or refresh after saving.

Single Sign On

How SAML SSO works

In Niagara, SAML SSO is the main supported SSO scheme. The **`SAMLAuthenticationScheme`**, found in the **`saml`** palette, configures a station for SAML Single Sign-On (SSO). This scheme stores configuration properties required for communications with a SAML IdP (Identity Provider). With SSO, when a user tries to access the station and has not been authenticated, the station delegates authentication to the configured IdP.

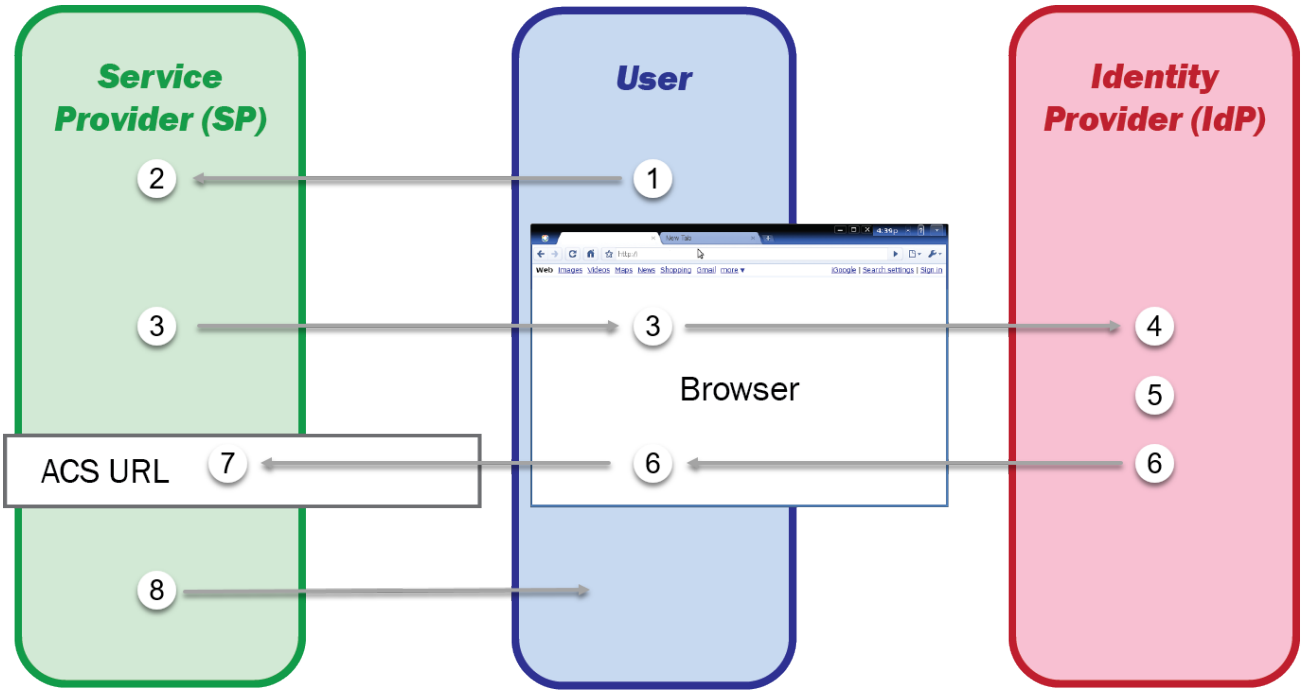
The framework's SAML scheme uses SAML 2.0 (Security Assertion Markup Language v.2.0). This is an open standard for exchanging authentication and authorization data in the form of encrypted messages passed between security domains. Specific protocols process the SAML request-response message exchanges over a secure connection. SSO via a web browser uses the SAML 2.0 Web Browser SSO profile to define how to use SAML messages and bindings between browser-connected Supervisor and remote controller stations.

With SSO, the process of SAML request-response message exchanges occurs between two system entities:

- SP (Service Provider), which is a station typically running on a remote controller.
- IdP (Identity Provider), which stores and maintains the authentication and authorization information.

SAML SSO uses an approach similar to LDAP authentication, which also stores authentication credentials external to the remote station.

Figure 7 SAML transaction steps



Beginning with step number one:

1. A user directs their web browser to a URL hosted by a Niagara station with the `SAML Authentication Scheme` available.

2. The SP generates a SAML request. Since the URL being requested requires authentication, the station opens a login window with various options. The user chooses to use SAML authentication. The station, acting as the SP, generates a SAML request that is unique to the user and to this particular session (visit).
3. The SP redirects the browser to the SAML IdP, Single Sign On (SSO) URL. In Niagara, the SAML IdP can be another Niagara station, such as the Supervisor station or it can be a third-party SAML IdP server.
4. The IdP receives the redirect, decodes it and prompts the user to log in to the IdP using their credentials.
5. After the user successfully logs in, the IdP generates a SAML response to be sent back to the SP.
6. The IdP encodes the SAML response and redirects the user's web browser back to the SP's Assertion Consumer Service (ACS) URL. The redirected URL contains the encoded SAML response and all the user data, which the SP is authorized to receive.
7. The SP (ACS URL) verifies the SAML response.
8. The SP grants access to the application and redirects the user's web browser back to the original URL they attempted to access. This time, based on the IdP authentication, the application's web page opens.

To configure a station for SAML SSO, in addition to the default authentication schemes (Digest and AxDigest), the station's **AuthenticationService** must contain a properly configured SAML authentication scheme. The **AuthenticationService** also contains SSO configuration properties with which to tailor the authentication workflow.

NOTE: InNiagara, an added baja-UserPrototype component supports the **UserService**. SAML authentication requires this UserPrototype.

NOTE: In Niagara 4.10 and later, the WebService **`Same Site`** property value "Strict" is not supported for SAML authentication. If the property is set to "Strict" it will cause SAML authentication to fail.

The Components chapter of this guide provides more information about the WebService component, and user-authentication components: baja-UserService, saml-SAMLAuthenticationScheme, saml-SAMLAttributeMapper, and saml-SamlXmlDecryptor.

Creating a User Prototype for SAML Authentication

SAML Authentication requires a user prototype of the type "baja:UserPrototype". This procedure describes how to create this new prototype and configure the **`Alternate Default Prototype`** for the **UserService**.

Prerequisites:

- You have connected to an existing station.
- You have the **`baja`** palette open.
- You have already obtained the necessary IdP configuration metadata that the IdP requires for authentication. Specifically, you need to know the value of the SAML attribute: `prototypeName`.

Step 1 Open a **Property Sheet** view of the station's **UserService**.

Step 2 Drag the **`UserPrototype`** component from the **`baja`** palette to the **`User Prototypes`** folder under the **UserService**.

Step 3 In the **Name** window, enter a name for this prototype that exactly matches the value of the `prototypeName` attribute being used by your SAML IdP and click **OK**.

If the SAML IdP sends the attribute `prototypeName=SAMLPrototype`, the prototype that you create must be named, "`SAMLPrototype`".

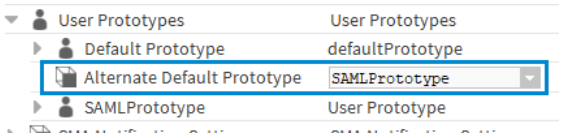
Step 4 In the Nav tree, right-click the station and click **Save Station**.

The system adds the new UserPrototype to the drop-down list for **`Alternate User Prototypes`**.

Step 5 Expand the **`Alternate Default Prototype`** drop-down list, select your new prototype, and click **Save**.

NOTE: If the SAML `prototypeName` attribute value does not match your prototype name, the **UserService** defaults the name to `Default Prototype`.

You have created a new prototype of the type "baja:UserPrototype", and configured the **UserService** **`Alternate Default Prototype`** to be this new prototype, as shown.



Configuring the SAML Authentication Scheme

SAML SSO is enabled by adding a SAML Authentication Scheme to the station. The scheme must be configured for a particular IdP (Identity Provider). You will need to obtain several configuration metadata from your IdP and use them in configuring the scheme. You will also need to provide the IdP with your station's SP metadata. These SAML metadata are used to share configuration information between the IdP and the SP (for more details refer to the Prerequisites section in this topic.). XML files define the metadata. Once the SAML authentication scheme is properly configured, the station is able to exchange SAML authentication messages with the IdP.

Prerequisites:

- You have the **`saml`** palette open.
- You have already obtained the necessary IdP configuration metadata that the IdP requires for authentication. Typically, the IdP SAML Server administrator provides these values. The configuration metadata, which may be provided in an XML file, are as follows:
  - HTTP-Redirect URL (corresponds to IdP Host URL, IdP Host Port, and IdP Login Path properties)
  - IdP Cert

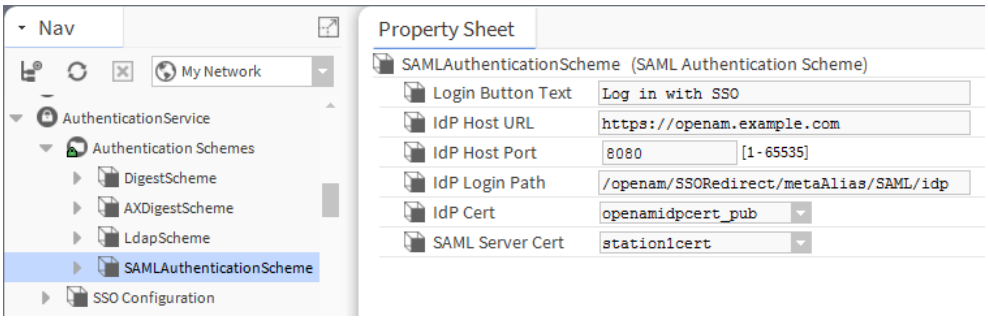
  Since SAML is an open standard, a number of third-party SAML Servers are available (i.e. OpenAM, Salesforce, etc.).
- You have provided the IdP SAML server administrator with an XML file containing your station's SP metadata and SAML public certificate. The SP metadata typically include the SP "Entity ID" and the "Assertion Consumer Service". The IdP needs these metadata to uniquely identify the SP and validate the messages sent by the station.

  The Entity ID is a unique name that you choose as an SP, usually a URL. For example, the Entity ID typically is something like this: `https://controller.domain.com:portNumber/saml`, where you would use your controller's hostname. A port number is required. The "Assertion Consumer Service" would be another URL, for example: `https://controller.domain.com:portNumber/saml/assertionConsumerService`, again using your controller's hostname. Once you have generated your SP metadata, save it in XML format and share the file with the IdP SAML server administrator.
- You have already created an Alternate Default Prototype for SAML authentication using the UserPrototype component in the **`baja`** palette. This UserPrototype is required for SAML authentication.

Step 1 In the Nav tree, expand the station's **Config→Services→AuthenticationService** node and drag the **SAMLAuthenticationScheme** component from the **`saml`** palette onto the **AuthenticationSchemes** folder.

Step 2 In the **Name** window, enter a name (or use the default text) and click **OK**.

Step 3 Expand **AuthenticationSchemes** and double-click on the **SAMLAuthenticationScheme** to open a **Property Sheet** view.



Shown here is an example of the SAML Authentication Scheme configured for the third-party OpenAM Idp.

Step 4 Enter values for the following properties:

Step 5 On completion, click **Save**.

a. For **`Login Button Text`** enter the preferred text label for the SSO login button that appears on the**Login** window.

b. For **`IdP Host URL`** enter the host of your Identity Provider (obtained from IdP admin).

c. For **`IdP Host Port`** enter the port number of your Identity Provider (obtained from IdP admin).

d. For **`IdP Login Path`** enter the location on the Identity Provider to which you must navigate to trigger the SAML authentication (obtained from IdP admin).

e. For **`IdP Cert`** enter the certificate used to encrypt messages sent to the IdP, and to validate messages signed by the IdP (obtained from IdP admin).

f. For **`SAML Server Cert`** enter the certificate used by the station to sign the messages being sent back to the IdP, and decrypt messages sent by the IdP.

NOTE: For the IdP to read and validate the messages sent by the station, the certificate with its public key must be provided to the IdP SAML server administrator as well.

Customizing SAML attribute mapping

This optional procedure describes how to configure the station to map arbitrarily named SAML attributes to User properties. Useful when the default mappings are not suitable, you may customize the property and attribute mappings as described here.

Prerequisites:

- You have already configured the SAMLAuthenticationScheme for the station.
- You have identified which SAML attributes are coming in from the IdP.
- You have the **`saml`** palette open.

Refer to the IdP-provided documentation to determine which SAML attributes are coming in from the IdP. As an alternative, you can install a SAML add-on to your web browser, which lets you view the attributes coming in from the IdP. For example, there is the SAML DevTools extension for Chrome, which you can use.

Step 1 In the station, navigate to the **SAMLAuthenticationScheme**.

Step 2 From the **`saml`** palette, drag the **SAMLAttributeMapper** to the **SAMLAuthenticationScheme**.

Step 3 Click the plus (+) to expand the **SAMLAttributeMapper** field editor.

An editor for a new mapping opens.

Step 4 In the editor, replace `attributeName` with the name of the attribute sent by the SAML IdP. For example, `employeeGroup`.

Step 5 Expand the drop-down list to select a property in the user prototype. For example, **`PrototypeName`**.

This maps the SAML attribute `employeeGroup` to the `PrototypeName` slot in the UserPrototype.

Certain properties may require additional information to map an attribute. In this case, an extra field editor or check box appears. For example, `Expiration` requires additional information - the format in which the expiration time is sent so that the date and time can be appropriately parsed. Similarly, `PrototypeName` provides a check box to be selected in cases where an IdP returns a Distinguished Name (DN) for the `prototypeName` attribute. For more details, refer to "saml-AttributeMapper" in the Components section of this document.

Step 6 Repeat these steps as needed to map additional attributes and click **Save** when finished.

Logging in with SAML SSO

SAML SSO works via a browser connection to a station. With SSO, you log in to one station and you are automatically allowed access to all other networked stations that are also configured for SSO. You will not be prompted for credentials when logging in to the other networked stations.

Prerequisites:

- Your station is already configured for SSO.
- You have already provided your IdP admin with any required data.
- You are using a web browser.

When entering the URL for the station in the browser, communications are bound by the domain specified by the Identity Provider (such as `station1.domain.com`). This means that you cannot make a local connection using `https:\\localhost`. Instead you would use `https:\\station1.domain.com`. This actually depends on the IdP requirements. Different IdPs may require different information and in a different format. For example, for the Salesforce IdP a field specifies the host name that you will use; and for the OpenAM IdP, you need to provide a specially-formatted XML file that supplies the host name and other data. You will need to ask the IdP administrator what information to provide.

Step 1 In the web browser, open a station connection.

Step 2 In the **Login** window, enter your username and click **Log In with SSO** (the actual button text may differ depending on the SSO scheme configuration).

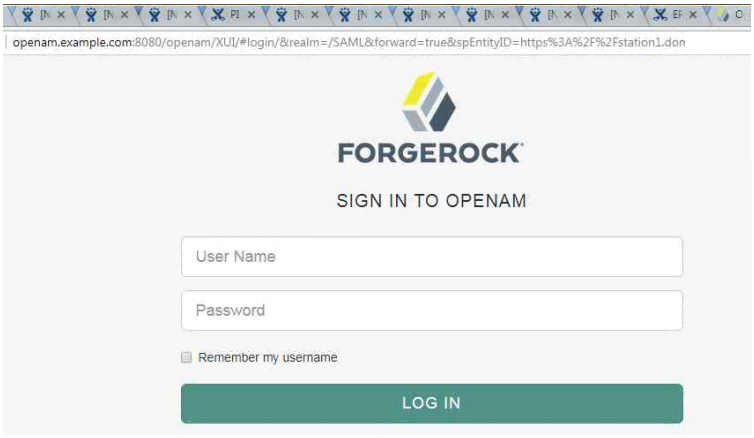
The **Remember my choice** option is most useful when there are multiple SAML authentication schemes in the station. In that situation, a separate SSO Login button displays for each SSO scheme. When checked, the login function remembers the chosen SSO Login button automatically uses it on subsequent attempts to access the station. This setting can also apply when there is just one SSO scheme. If the station is not set for auto-SSO, clicking this check box simulates auto-SSO by attempting to log in with the saved scheme.

If you have already logged in with SSO, the station connects immediately.

If this is the first time you are logging in with SSO your browser redirects to the Identity Provider's site.

Step 3 In the Identity Provider's login window, enter your station credentials (username and password) and click **Log In**.

The example shown here shows the OpenAM IdP SSO Login window.



On successful authentication completion, you are logged in to the station and the browser is immediately redirected there. Also, you immediately gain access to this station and to all other networked stations. Additionally, you have an active session with the IdP, which allows you to bypass entering credentials the next time you log in to a station. Actually, you still are redirected to the IdP but it knows you have already logged in and redirects you right back to the station.

About the SAML IdP Service

In the latest versions of Niagara, there is added support for the **`SAML IdP Service`**. Setting up this service on your Supervisor station allows you to take advantage of SAML functionality without having to set up an external Identity Provider (IdP). Installing and configuring the service allows you to set up an internal IdP that works with the **`SAML Authentication Scheme`**.

The `samlDP` feature license is required to run the **SAMLIdPService**.

Within the service is the **`Circle of Trust`** (COT) component. Once configuration is complete, the COT lists the collection of subordinate stations and the collection of users that are allowed to log in to those stations. For example, if a Supervisor that is connected to 100+ stations is using SAML authentication for just a few of them, you can group those few stations together within a **`Circle of Trust`**.

You can configure the service with multiple **`Circle of Trust`** components.

Additionally, you can add a remote station that is not in the **NiagaraNetwork** to any Circle of Trust, or specify other authentication schemes or other user prototypes that may be used when logging in.

Users not included in a **`Circle of Trust`** cannot log in to the station(s) specified in that COT. Such attempts to log in are rejected by the IdP.

CAUTION: Any user that has admin access to the SAMLIdPService can see the following sensitive information for all stations in the **NiagaraNetwork**:

- A list of all users (usernames only) in the stations
- A list of all stations (station names and hostname only)
- A list of all authentication schemes (scheme name only)

This is true **regardless of the permissions that user has on those components**. For security purposes, when adding the SAMLIdPService to a station take special care to place it in appropriate categories.

Additionally, any user with admin write permissions on the SAMLIdPService can effectively control who logs in to **all stations** in the **NiagaraNetwork**, and with what permissions (to the extent allowed by the remote station for remote users). The recommended security best practice is that you treat this as a **very sensitive role**, assigning it only to highly trusted users who manage authentication.

A provisoning job that is run on the Supervisor simplifies configuring subordinate stations to use SAML IdP as well.

Preliminary steps

Complete these steps in preparation for setting up the SAML IdP Service.

The following steps can be completed in any order.

- Generate a server certificate in the Supervisor's **Certificate Management**. The certificate does not need to be signed since SAML IdP uses certificate pinning. It just needs to be present in the **User Key Store**.
- Determine how many Circles of Trust you intend to create. For each one, identify which stations to include and which users to include as user prototypes.
- Add all of the subordinate stations to the Supervisor's **NiagaraNetwork** and make sure that the stations ping. This connection is necessary because later you will run a provisioning job on the subordinate stations.
- Create user prototypes in each remote station. You need all of the user prototypes for all of the users in any Circles of Trust that a station is a part of. The user prototypes do not need to exist in the Supervisor station. When an attempt to log in to a subordinate station as a SAML user is made, the user is created in the subordinate station according to a user prototype. For this reason, the prototypes must exist in the remote station beforehand.

  NOTE: The SAML Authentication Scheme only supports the baja-UserPrototype. While LDAP and Kerberos support this user prototype as well as the default user prototype.
- In the Supervisor's UserService, create all of the users that you need for any Circle of Trust or subordinate station.

Basic workflow for setting up the Service

Use this basic workflow for setting up the SAML IdP Service. Most of the workflow occurs on the Supervisor station. However, there are a few steps that must be completed on the remote station(s).

On the Supervisor station

The platform must be licensed for `samlDP`.

1. Install the SAML IdP Service.
2. Configure properties for the SAML IdP Service.
3. Configure the Circle Of Trust.
   a. Add subordinate stations.
   b. Add users.
   c. Specify any additional authentication schemes that may be used when logging in.

      NOTE: This is necessary only to add users that do not yet exist in the Supervisor, such as LDAP users.
   d. Specify the names of the user prototypes for all of the users that are included in this COT. These are the names of the prototypes that already exist in the subordinate station(s).
4. Configure the subordinate station(s) for SAMLIdPService by running a provisioning job with this step: **`Configure Niagara IdP & Saml Scheme`**.

On each Subordinate Station

1. Set up User Prototypes. These will be used as templates to create the users upon login to the station.
2. A signing Certificate (Server cert with both public and private keys) is required. The provisioning job runs on the Supervisor generates the necessary cert.
3. The SAML Authentication Scheme must be installed and configured. The provisioning job runs on the Supervisor creates the necessary scheme.

Setting up the SAML IdP Service

Setting up the SAML IdP Service on your Supervisor station makes it easy to configure an internal Niagara IdP that works with your SAML Authentication Scheme.

Prerequisites:

- The platform is licensed for `samlDP` (with the **`On/Off`** attribute set to `true`).
- A Server Certificate has already been generated in the User Key Store for use with SAML IdP.
- The Supervisor station's WebService is running.
- The **`saml`** palette is open.

NOTE: Install the SAMLIdPService only on a Supervisor station.

Step 1 From the **`saml`** palette drag the **`SAMLIdPService`** component to the Supervisor station's **`Services`** node, and when prompted, enter a name for the service. For example, Niagara IdP.

Step 2 Open a **Property Sheet** view of the service and configure the following properties:

a. **`IdP Signing Cert`** — click the drop-down list and select the existing Server Certificate from the station's **User Key Store**.

b. **`EntityID`** — enter the station's IP address (or hostname) plus the port the WebService is running on.

NOTE: The port number is required (e.g, 443 for https). It must be included in the EntityID otherwise the service will fail. Also required are the characters "`/saml/`" which must be appended to the EntityID value. For example, if you entered "`https://192.68.19.20:443`" for the **`EntityID`**, you then need to append it with "`/saml/`", so that it reads: "`https://192.68.19.20:443/saml/`".

c. **`Time Skew`** — set the number of minutes to extend the validity period of the SAML request from the subordinate station. This is intended to allow SAML message to be accepted when the supervisor and subordinate stations cannot synchronize their time. Use positive values. Default value is 3 minutes.

d. **`Apply Skew to Response`** — click the dropdown list and select "`true`" to apply the specified Time Skew setting to the response. For cases where a time difference exists between the Supervisor and a subordinate station, this will apply the Time Skew to the response(s).

Step 3 Double-click **`Circle of Trust`** to open the **Circle Of Trust Editor** and configure the following properties.

a. **`Description`** — (optional) enter a description for this Circle of Trust.

b. **`Http Redirect Endpoint`** — A read-only value that shows the URL for this Circle of Trust.

Step 4 The lower half of the view lists subordinate stations in the **NiagaraNetwork**. Click a checkbox for any station(s) to include it in this Circle of Trust.

NOTE: You are not limited to stations in your **NiagaraNetwork**. Click **Add Station** to select a non-**NiagaraNetwork** station. You will be prompted to enter a Name, Server Certificate public key, and Issuer URL for the station. Any stations not already in the **NiagaraNetwork** cannot be configured via the Configure Niagara IdP and SAML Scheme provisioning job and the SAMLAuthenticationSchemes on the remote station must be added manually.

Step 5 Click **Users** to select the user(s) from your UserService to include in this Circle of Trust. This allows those users to log into the specified stations in the COT.

Step 6 Click **Auth Schemes** to specify which other authentication schemes may be used when logging in.

NOTE: This is to accommodate users who may not yet exist in the Supervisor station. For example, you might specify the LdapScheme so that LDAP users can login. However, for typical (SAML Authentication Scheme) usage you can skip this step.

Step 7 Click **Prototypes** to specify one or more user prototypes that may be used when logging in.

NOTE: The specified prototypes should correspond to User Prototypes on the subordinate station (s). The User Prototype is used to create the user on the subordinate station on login and needs to exist on the station at that time.

At this point you have completed configuring the SAMLIdPService and adding user prototypes to the Circle of Trust.

Step 8 In the Supervisor's UserService, assign a SAML Prototype to each Circle of Trust user. Do this for each Circle of Trust that the user is in.

NOTE: You can choose from any of the prototype names that have been added to that Circle of Trust.

Completing the Circle of Trust configuration is the last item in the workflow for configuring the SAML IdP Service on the Supervisor. Additional configuration must be done on the subordinate station(s). For those details see "Configuring the subordinate station for SAMLIdPService".

Configuring subordinate stations for SAML IdP Service and Scheme

A provisioning job can configure each remote station's SAML (Security Assertion Markup Language) authentication scheme and local SAML IdP (Identity Provider) service. This procedure covers running a provisioning job on the Supervisor station to configure one or more remote stations with a server certificate (private and public keys) and the SAML authentication scheme configured for the internal IdP.

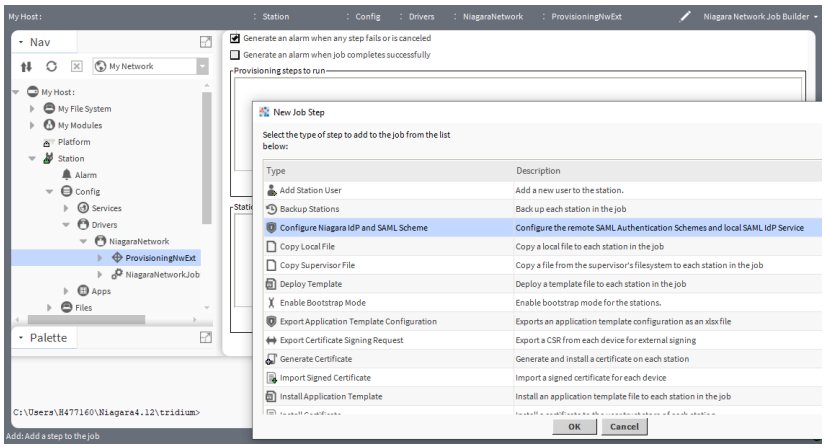
Prerequisites: You are working in a Supervisor station. The **SAMLIdService** is installed and configured on the Supervisor station.

The **NiagaraNetwork** on the Supervisor contains one or more remote stations.

Step 1 Expand **Config→Drivers→NiagaraNetwork**.

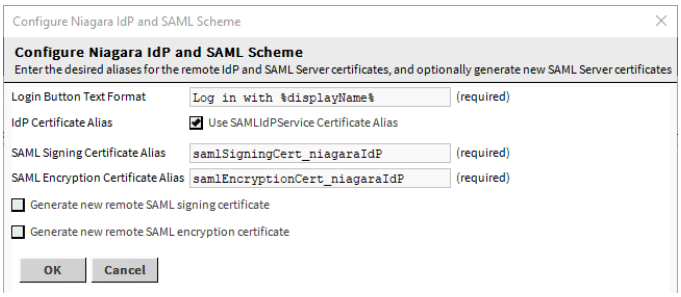
Step 2 To open the **Niagara Network Job Builder** view, double-click on **ProvisioningNwExt**

The **Niagara Network Job Builder** view opens.



Step 3 In the top pane, **Provisioning steps to run**, click add ⊕, click **Configure Niagara IdP and SAML Scheme**, and click **OK**.

The **Configure Niagara IdP and SAML Scheme** window opens.



Step 4 Fill in the following required properties and click **OK**.

- **`Login Button Text Format`** is the preferred text to display on the login button to access the subordinate station. For example, `Log in to Floor1_%displayName%`, where the system substitutes the name of the Circle of Trust for the BFormat script %displayName%. If multiple Circles of Trust include the subordinate station, the system creates multiple login buttons.
- **`IdP Certificate Alias`** is the public key of the certificate configured as the **`IdP Signing Cert`** in the **SAMLIdPService** that the system imports to the subordinate station's **User Trust Store**. This property configures the alias of the resulting public key as it appears in the subordinate station, for example: `niagaraIdP`. Entering a value in this property activates the **OK** button in the window.
- **`SAML Signing Certificate Alias`** defines the alias of the certificate to use as the subordinate station's SAML Server Certificate. If not generating a new server certificate (see option below), the certificate should already exist in the subordinate station's **User Key Store**.
- Optionally, you can generate a new server certificate and/or a new remote SAML encryption certificate to use for this purpose. Click the check box to **`Generate new remote SAML signing certificate`** and/or to **`Generate new remote SAML encryption certificate`**, fill in the required data in the additional properties, and click **OK**.

The provisioning application adds the step to the job builder.

Step 5 In the bottom pane, **Stations to include in the job**, click (Add).

The **Add Device** window opens

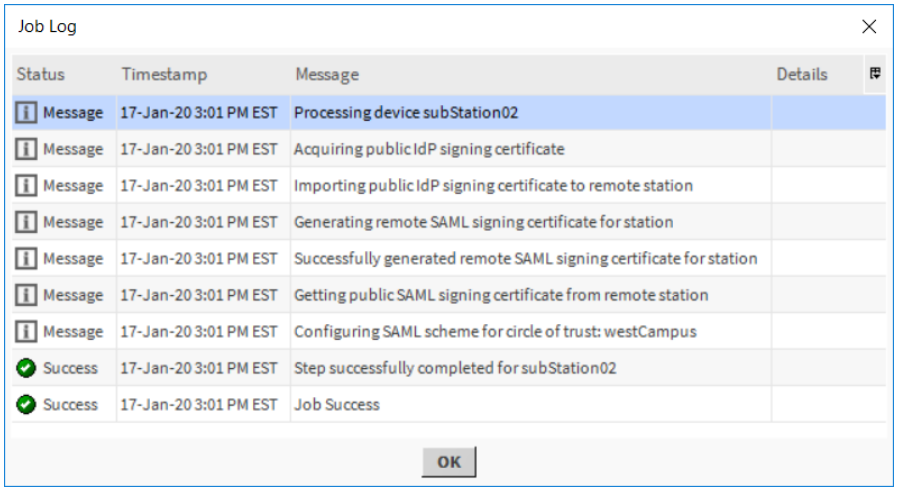
Step 6 Select the devices (that is, stations in the **NiagaraNetwork** that are included in a Circle of Trust) to be added to the job, and click **OK**.

The provisioning application adds the job to the step builder.

Step 7 To start the provisioning job, click **Run Now**.

Step 8 To view the job progress open the Job Service **Job Log** view.

The **Job Log** opens.



This provisioning job exports the public key of the Supervisor's IdP Signing Certificate to the **User Trust Store** of each subordinate station in the job. For each station, it generates a unique SAML Signing Certificate in the station's **User Key Store** (or selects from server certificates already existing in the **User Key Store**). It then assigns a copy of this certificate's public key to the Station Service Provider under the Circle of Trust in the Supervisor's **SAMLIdPService** using certificate pinning.

NOTE: The signing certificates mentioned in the previous paragraph are actually Server certificates. They should not be confused with Code Signing Certificates which have a different purpose.

For each Circle of Trust that a subordinate station is a part of, the provisioning job creates a SAML Authentication Scheme in the subordinate station's **AuthenticationService**.

For more information, see "About the SAML IdP Service" section of the *Niagara Station Security Guide*.

Users

A user is a function or machine, for example, admin, operator, etc., that represents a type of system access rather than an individual person.

Users are authorized to log in to Supervisor and remote controller stations for the purpose of configuring and managing components and data. Each station may have the same set of users. And each station may have its own unique local user(s).

In a station database, a component represents each user under the station's **UserService** (`baja` module). Using the **User Manager** view of this service, you add, modify, and delete user components.

The default prototype configures the properties associated with each new user.

In latest versions of Niagara, there is added support for the , any active session associated with a user terminates when you make these changes in **UserService Property Sheet**.

- If you remove the user from the **UserService Property Sheet**.
- If the **`Enabled`** property is set to `false`.
- If the **`Expiration`** property is changed to a date that has already expired.
- If the **`Authentication Scheme Name`** is changed.
- If the **`Allow Concurrent Sessions`** is set to `false`.

User prototypes

A prototype groups users who share the same permissions and other characteristics. This grouping facilitates setting up system users under the **UserService**. Customizing the default prototype and creating custom user prototypes simplify user management.

The **User Prototypes** container is a frozen slot under a station's **UserService**. It contains a frozen **Default Prototype** to support centralized users in the station's **NiagaraNetwork** and allows for additional user prototypes that support user synchronization and remote authentication schemes, such as LDAP, Kerberos, and SAML.

Default prototype

This set of user properties is a child component (**baja-user**) of the **User Prototypes** container under the **UserService**. Using the default prototype simplifies user management because its default values populate each new user. For example the default prototype can configure a set of minimum permissions that apply to all users and a typical Nav file to apply to all.

The only property value that does not serve as a default when creating a new user is **Password**.

LDAP and Kerberos support the default user prototype, however, SAML does not.

User prototypes

These prototypes (also **baja-user** components) enable network user synchronization. To create this type of prototype you duplicate the default prototype, giving each duplicate a new name and then modify its properties. You should name duplicate prototypes using descriptive text that can be logically associated with groups of station users, such as AdminHvac, GenOperations, LtgAndAlarms, and so on.

User prototypes appear under the **UserService** in the Nav tree and on its **Property Sheet**, but the **User Manager** view does not list them. Instead, when you add a user, the property **Prototype Name** provides a list of available prototypes from which to choose.

NOTE: When you manually add a new user in the **User Manager** view, the user's property values default to those of the default prototype regardless of the **Prototype Name** you choose. The default prototype serves as a template to populate a new user's properties (all except **Password**). The **Prototype Name** applies only when synchronizing users.

Authentication prototypes

This prototype (**baja-UserPrototype**) supports remote authentication schemes, including LDAP, Kerberos and SAML.

NOTE: The SAML authentication scheme only supports the **baja-UserPrototype**. While LDAP and Kerberos support this user prototype as well the default user prototype.

The properties provided by the **baja-UserPrototype** are similar to those of the default prototype with some exceptions. An **Overridable** property prevents a value from being overwritten with a default value at next login.

Station-to-station users

A station-to-station user is a machine user as opposed to a human user type or function.

By convention, a station-to-station user should be named something memorable (perhaps a name that is unique to your company or even to a job site).

As with all user, human and machine, you should carefully guard user passwords. Although frequently a station-to-station user is assigned a role with many `admin`-level Write permissions, every user, human and machine should be assigned roles that permit them (it) to access only the components required to do their job. To improve system security, do not make a station-to-station user a super user.

When adding this user, properties, such as **Facets**, **Nav File**, and **Web Profile**, which apply to browser access are inconsequential.

NOTE: Do not use a station-to-station user to log in as a human user to a station! Instead, you reference this user in another station, when adding a device under a **NiagaraNetwork**.

Adding or editing a user

A network user defines the type of person or machine that is allowed access to a station. For example, admin, operator, machine-to-machine (M2M). Under the station's **Services** container, the **UserService** provides a default **User Manager** view for you to add, delete, and edit network users.

Prerequisites: Your system has at least one Supervisor station and one or more subordinate controller stations. Each station in your network provides access to a set of users who share configuration properties. These may include station-to-station users as well as an admin, operator, and other typical access functions. You are working in Workbench and are connected to a local Supervisor station.

Step 1 To configure access permissions, expand **Config→Services→RoleService** and add (click **New**) a role for each type of user.

Station-to-station users should have very limited roles. Other types of users, such as operators, may be allowed to view data but not make changes.

Step 2 Create the same roles in each subordinate controller station.

Synchronization does not replicate roles in target, subordinate stations.

Step 3 Configure the default prototype by expanding **Config→Services→UserService**, and double-clicking **Default Prototype**.

The **UserService** uses this prototype to set up each system user. It should contain all the standard values for user properties.

Step 4 Make any changes to the default prototype required to support your company and click **Save**.

Step 5 To make a custom user prototype, which the system will use to synchronize users across your network, right-click the **Default Prototype**, click `Duplicate` and give the prototype a descriptive name based on its uniquely-configured properties.

Step 6 Double-click the **UserService** node in the station Nav tree.

The **User Manager** view opens.

Step 7 To create a new user, click the **New** button, otherwise, to edit an existing user select the user and click the **Edit** button.

The **New** or **Edit** window opens.

NOTE: If the **`Secure Only Password Set`** property is set to `true`, and the connection to the station is not secure (using Fox or HTTP instead of Foxs or HTTPs), the **New** button is disabled.

Step 8 Create or modify user properties and click **Save**.

Properties to configure include **`Prototype Name`**, **`Network User`**, **`Authentication Scheme Name`**, and **`Roles`**.

In a multi-station system, you could create this same user in each station or use synchronization to replicate the user changes in one station to all other stations in the network.

Adding roles and permissions

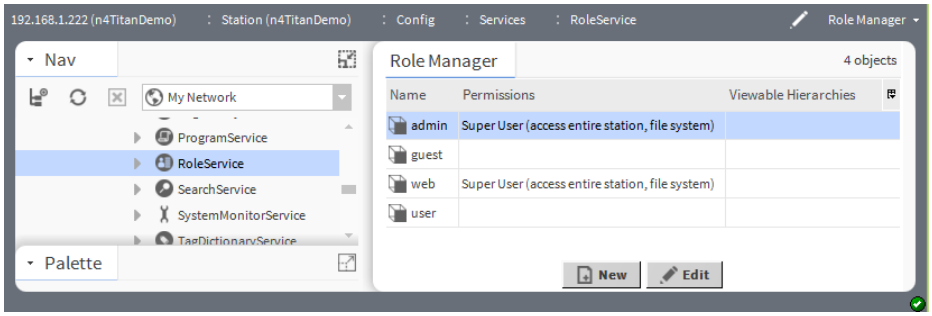
You add roles using the station's **Role Manager** view (**`RoleService`**).

Prerequisites: You enabled the `Operator` config flag enabled for any restricted components. Categories have been created and any basic categories assigned to components.

Most companies require, as a minimum, an administrator (super user) role, a manager role, and a regular user or operator role.

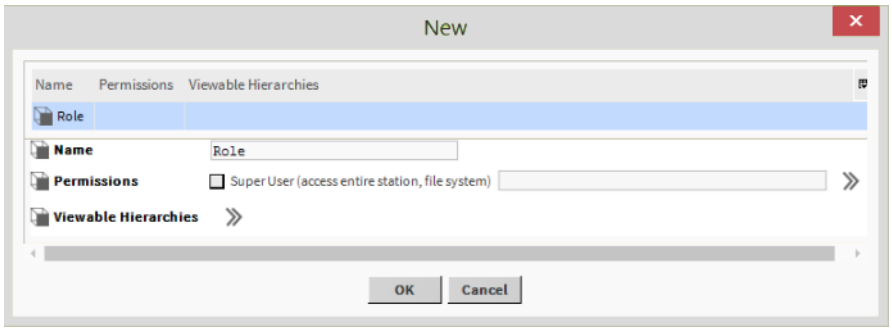
Step 1 Right-click **RoleService** in the Nav tree, click **Views→Role Manager**.

The **Role Manager** view opens.



Step 2 Click the **New** button, enter the number of roles to create in the pop-up window and click **OK**.

The system displays the **New** window with a row for each role you are creating.



Step 3 Name the role.

Step 4 To configure a role as a super user, click the `Permissions` check box for `Super User`.

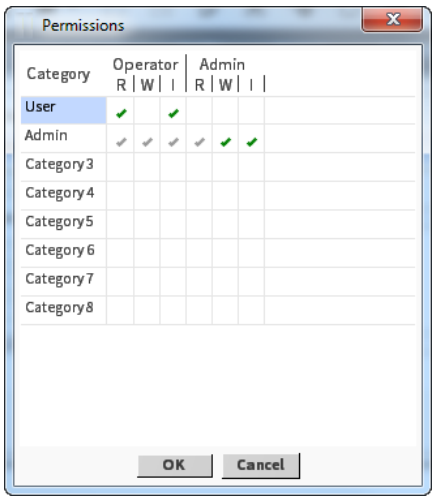
The built-in `Admin` role grants all possible rights for every category (super user). Only when logged in as the `Admin` user, or another super user, can you assign super user rights using the `Super User` check box.

In general, assigning super user rights should be strictly limited and based on special needs. For example, a Supervisor station may need super user rights to connect with other station clients (machine login vs. login by a person) in scenarios where Program objects are exported from stations using ExportTags. Human users may need super user rights to add and edit Program or Robot components.

CAUTION: Do not make it a common practice to give station-to-station users admin privileges. If your network is breached, a station-to-station user could cause significant damage without drawing attention to what is happening.

Step 5 To set up individual permissions, click the chevron at the end of the **`Permissions`** property.

The **Permissions** map opens.



The first column, Category, lists the groups to which you may grant permission. The Operator and Admin columns relate to the permissions level configured on each component. Below these headings are the cells to use for assigning one of three permissions to each category:

- R = Read allows the user to view the object.
- W = Write allows the user to change the object.
- I = Invoke allows the user to initiate an action related to the object.

Depending on how the permission level is set on the slot, six permissions are derived:

- To allow a user to view `operator`-level information, check the `Operator` config flag on the slot and select the Operator R column on the permission map.
- To allow a user to modify `operator`-level information (if it is not read-only), check the `Operator` config flag on the slot and select the Operator W column on the permission map.
- To allow the user to view and invoke `operator`-level operations (actions), check the `Operator` config flag on the slot and select the Operator I column on the permission map
- To allow the user to view `admin`-level information, leave the `Operator` config flag unchecked on the slot and select the Admin R column on the permission map.
- To allow the user to modify `admin`-level information (if it is not read-only), leave the `Operator` config flag unchecked on the slot and select the Admin W column on the permission map.
- To allow the user to view and invoke `admin`-level operations (actions), leave the `Operator` config flag unchecked on the slot and select the Admin I column on the permission map.

When you assign permissions, higher-level permissions (green check marks) automatically include the lower-level ones (gray check marks). For example, if you enable `admin`-level write (W), the system automatically enables `admin`-level read (R), as well as `operator`-level read and write (RW).

Step 6 Click the cell to assign a permission and click **OK**.

The e **`Permissions`** property displays the permissions.

Step 7 To finalize permissions, click **OK**.

Step 8 In a multi-station system, perform these same steps in each station so that each station has the same set of roles.

NOTE: During the network user synchronization process the framework sends the user's role assignment to the receiving station, however, it does not create the actual role(s) on the receiving station. You must set up matching roles on each receiving station before synchronizing network users.

Assigning roles to users

This task associates the permissions defined by a specific role with each user.

Prerequisites: Roles and users have already been created. In a multi-station system, each station has the same set of roles.

Step 1 Right-click **UserService** in the Nav tree, click **Views→User Manager**.

Step 2 Double-click the user's row in the **User Manager**.

The user's **Property Sheet** appears.

Step 3 For the **Roles** property, select one or more role names by clicking in the check box and click **OK**.

Assigning authentication schemes to users

Each user is assigned their own AuthenticationScheme. This allows different users to use different schemes appropriate to the user type. Some schemes apply to both Fox and Web. Other schemes, such as HTTP-Basic, apply only to certain web logins (for example, Obix clients). These schemes do not work over Fox or even via the form login.

Prerequisites: The authentication scheme to use has been added to the **AuthenticationService**.

By default, each new station comes with the **DigestScheme** and **AXDigestScheme** already installed. The **DigestScheme** is assigned to all users, so that in simple cases no additional setup is required.

Step 1 Right-click **UserService** in the Nav tree and click **Views→User Manager**.

The **User Manager** view opens.

Step 2 Select the user and click **Edit**.

The **UserService Property Sheet** opens.

Step 3 Scroll down to **Authentication Scheme Name** and expand the **Authenticator** section.

Step 4 To assign an authentication scheme, select the scheme from the **Authentication Scheme Name** drop-down list.

Once these setup steps are complete, the station is ready for authentication.

Password management

Managing passwords involves configuring the strength of the passwords to be used authentication scheme, establishing the period of time after which the password expires, setting the warning period, and setting up the password for each user.

The system supports three password features designed to strengthen access security:

- Password strength that may be configured for each authentication scheme.
- An expiration interval for a password
- Password history

Setting up password strength

Strong passwords are recommended. Along with the other password features, password strength will frustrate any attempt to breach your system.

Prerequisites: Authentication scheme has been added to the **AuthenticationService**.

Password strength is associated with the selected authentication scheme, for example, Digest or Baisc, but not LDAP, for which password strength is managed by the LDAP server. You can create different strengths

for different schemes and apply those schemes to different classes of user. For example, an administrator could have stricter password strength requirements.

Once the New Station wizard completes, you can adjust the scheme's password strength properties as needed. If changed for a scheme, any future password change for any station user (including the `admin` user) requires the minimum values specified in the **`Password Strength`** properties.

NOTE: Although you may reduce password strength by entering zeros for its property values, it is strongly recommended that you retain a level of password strength similar to the default level, if not greater. For example, you may wish to require at least one special and at least two upper case characters.

You configure password strength for each authentication scheme.

Step 1 Right-click the **`AuthenticationService`** in the Nav tree and click **Views→Property Sheet**.

The **AuthenticationService Property Sheet** window appears.

Step 2 Expand the scheme and the **Global Password Configuration→Password Strength** container for the scheme.

Step 3 Configure the minimum character requirements, **`Expiration Interval`**, **`Warning Period`**, and **`Password History Length`** (5 or 10 characters).

Step 4 Do the same for any other scheme you plan to use and click **Save**.

Setting up password options

In most cases, users create their own passwords. You may create a temporary password for each user in the **`UserService`** and require them to change the password at their next login. You may also configure the password expiration date.

Prerequisites: The authentication scheme you need is available in the **`AuthenticationService`**.

Step 1 Right-click **`UserService`** and click **Views→Property Sheet** in the Nav tree.

Step 2 Open the user's **Property Sheet**.

Step 3 Expand the user whose password you want to set.

Step 4 Scroll down and expand the **Authenticator→Password Config** container under the user name.

`Force Reset At Next Login` defaults to `true`.

Step 5 To allow the user to continue using the same password, set **`Force Reset At Next Login`** to `false`.

Step 6 Set the password expiration date, scroll down and click **OK**.

Setting up a user's password

You configure user passwords through the UserService. If you are accessing the UserService from a browser, your connection must be secure (https) or you will be unable to set the password.

Step 1 Double-click **UserServices** in the Nav tree and double-click the user record.

Step 2 To view the password properties, expand the **Authenticator**.

Step 3 Enter and confirm the password, then click **OK**.

Logging on to a station

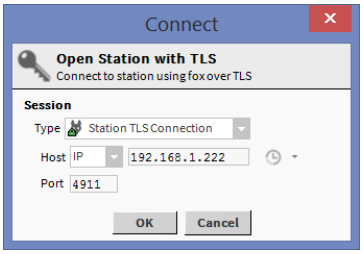
Using TLS, a secure communication session is established before the system asks for your user credentials. When you log on using the station Authentication window, the system confirms your identity, which determines your Nav tree configuration and the components you have permission to access. The system is designed to require minimum interaction while providing a secure connection and ensuring authorized access.

Prerequisites: An authentication scheme has been assigned to each user, and a user name and password created.

This procedure demonstrates user authentication using the default DigestScheme.

Step 1 Open the station.

The system opens a station **Connect** window.



This window initiates the process of verifying the server.

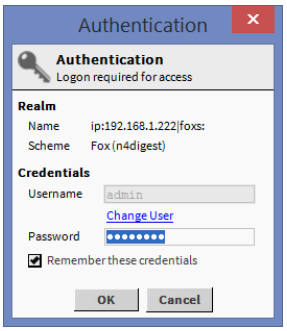
Step 2 Enter the IP address or confirm the default address and click **OK**.

If no matching root CA certificate can be found in the client's System or User Trust Stores, the system presents a default certificate for your approval.

Step 3 If you are presented with a certificate, make sure you recognize the certificate's **`Issued By`** and **`Subject`** properties.

CAUTION: Do not approve a certificate if you do not recognize these properties. The weakest link in the security chain is the user who simply clicks OK without thinking.

The system displays the station **Authentication** window.



Step 4 If you are logging on for the first time, enter your user name.

Stations can have many authentication schemes. The first time you log on to a new station the system allows you to enter the **`Username`**. It uses this information to determine what authentication scheme to use. After that initial logon, you cannot change the user because another user may use a different scheme with different credential requirements. The `Change User` link provides a way for a different user with a different authentication scheme to log on.

Step 5 To change to a different user, click the `Change User` link and enter a different name.

Step 6 Enter your station password, select **`Remember these credentials`** and click **OK**.

When you select the `Remember these credentials` check box, the system saves the last user name and password you entered and defaults to them the next time you log on.

This procedure establishes a secure TLS connection to the station using the Foxs protocol over port 4911 (this is the default port).

The default logon threshold is five attempts. If you make five unsuccessful attempts to log in during a 30-second period the system locks you out for 10 seconds. You may change the logon threshold in the **`UserService`**.

To log off, close Workbench or the browser.

Each authentication scheme supports its own audit log, including saving date, user, and event. This information is written to the AuditHistory in the following location: **Station→History→<station name>→AuditHistory**. Permission must be assigned to this file in the **`RoleService`** to grant a user access to view it.

Station Auto Logoff

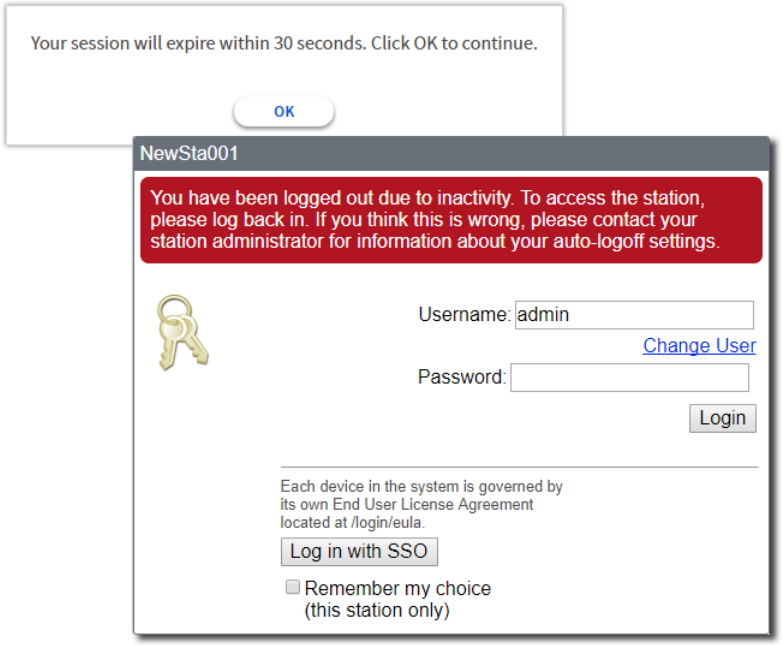
In Niagara, there is added support for the and later there is added support for the station auto logoff capability. Meaning that any station connection, in a web browser or in Workbench, can log the user off due to inactivity. This is important for security reasons, helping to prevent the opportunity for unauthorized access.

NOTE: The Workbench has separate auto logoff settings (under **Tools→Options**) that apply to only to the Workbench session.

Enabled by default, the station Auto LogOff feature can be configured using the **`Default Auto Logoff Period`** property in the UserService and additional auto logoff properties in the individual User accounts.

When the station does not detect any user activity for a configurable period of time, it first displays a warning popup. Clicking **OK** in the warning allows you to continue working in the station connection, otherwise the station automatically logs you off. Once auto logoff occurs, you are presented with an auto logoff notice in the **Login** window in the browser, as shown. If your station connection is via Workbench, a similar warning and logoff notice displays in the window.

Figure 8 Auto Logoff warning popup and alert notice in browser login



The intended purpose of the station auto logoff properties and separate Workbench auto logoff options is to allow for setting more restrictive (shorter) or less restrictive (longer) auto logoff period times to accommodate different use cases. A security best practice is using these properties to set reasonable limits for periods of inactivity for both the Workbench and station users.

For details on the station auto logoff properties, refer to “baja-UserService.”

Changing your password

Based on the password configuration, the system warns you when your password is about to expire. You can only change your password when required to do so by the system. Follow the login prompts.

User authentication troubleshooting

Once authentication is configured and passwords assigned, very little can go wrong.

I am attempting to set up new users using a browser, and the New button is not available.

This is a security control. Most likely the **Secure Only Password Set** is on (set to `true`), and you have made a regular connection (http) to the platform/station. When a secure connection is required, you must make a secure connection (https) to the platform/station to set the password.

My credentials were working, but they no longer allow me to log in.

- Your password may have expired.
- You may not have permission to access this station. Check with your manager.
- The credentials cache may have become corrupted causing the saved credentials to no longer work. Clear the `Remember these credentials` check box, re-type the password or clear your computer's cache.
- There is a problem with the configuration of the authentication scheme. For example, if you are using the LDAP scheme, the port is blocked by a firewall or the wrong LDAP host name or port has been configured. Refer to the documentation for the LDAP scheme.

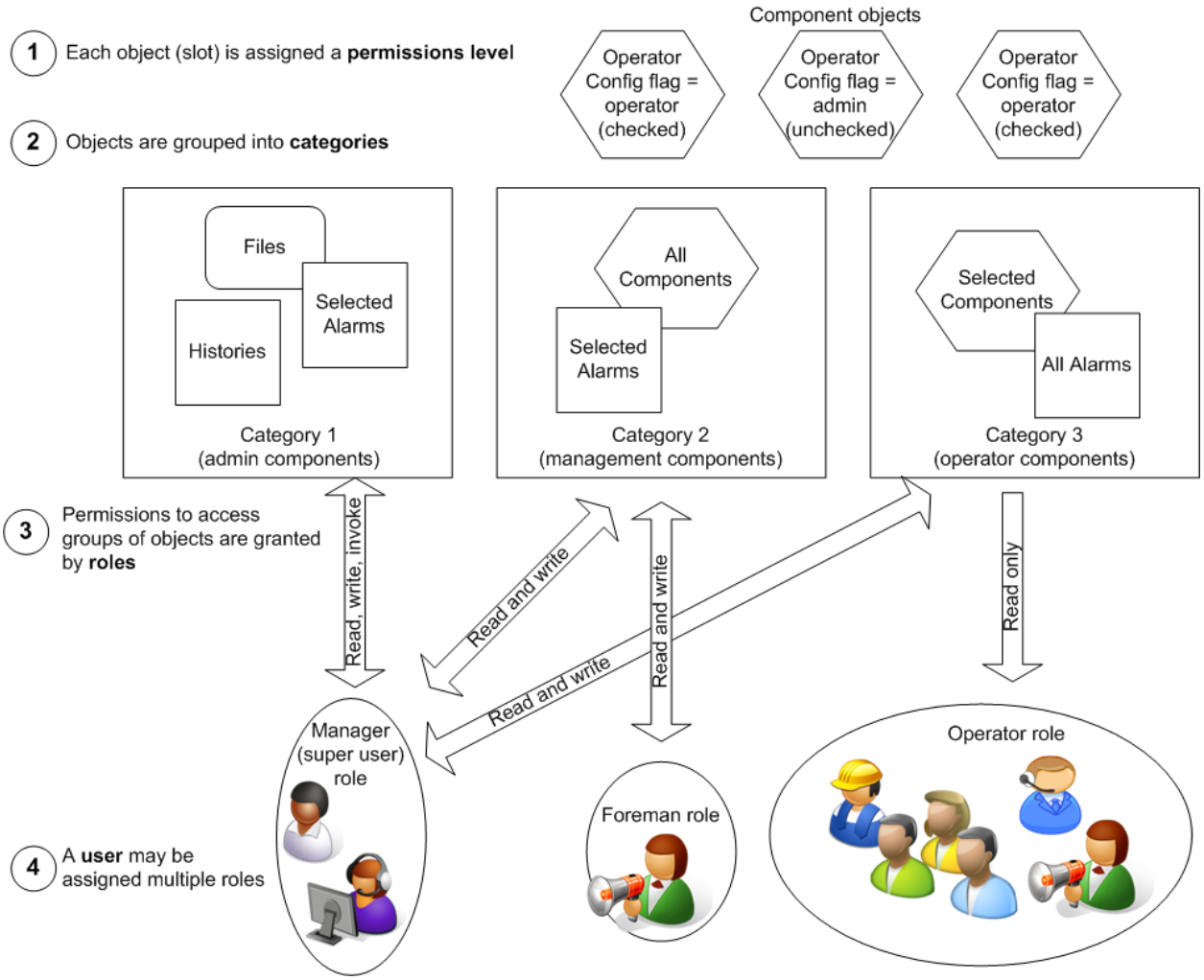
Chapter 4 Authorization management

Topics covered in this chapter

- ♦ Component permissions checklist
- ♦ Component permission level
- ♦ Categories
- ♦ Roles and permissions
- ♦ Component permissions troubleshooting

Once a human or remote station user is authenticated, authorization to access station components is based upon the permission level assigned to each slot, the category(ies) into which components are grouped, and the role assigned to each user. All configuration is stored in the system database, using services, components, and component views.

Figure 9 Station security configuration includes categories, roles and users



1. Beginning at the top of the diagram, the *permission level* may be configured on each component as needed. You change the default permission level for a component by turning the `Operator` config flag for the slot on or off.

2. *Categories* organize components, files and histories into groups. You set up categories using the **Category Manager** view (**CategoryService**).
3. *Roles* associate permissions to read, write, or invoke an action on a category of system components with a generic name, such as *Manager*, *Foreman* or *Maintenance crew*. You set up roles and permissions using the **Role Manager** view (**RoleService**). The **New Station** wizard installs the `Admin` role. This special super user cannot be modified or configured, and does not appear in the **Role Manager**.
4. Human and machine *users* are assigned to roles for the purpose of granting users the right to read, write and invoke actions on components. You assign roles to individual users using the **User Manager** view (**UserService**).

Component permissions checklist

Use this checklist to verify that you completed all required tasks to set up roles and permissions.

- Categories have been set up and basic categories assigned to components.
- System components that require access control have been identified.
- The permission level config flag has been set on component slots.
- Basic categories have been set up.
- Basic categories have been assigned to components.
- Roles have been created and permissions granted.
- Roles have been assigned to users.
- Station security has been tested.

Component permission level

The component permission level is a config flag associated with the slot. The configuration of this flag begins the process of granting permission to access individual component slots.

- If the component slot's `Operator` config flag is cleared (unchecked), the slot is configured for the `admin` *permission level*. A user must be assigned a role with at least the minimum permission set in the **Role Manager** view to `admin`-level Read (R).
- If the slot's `Operator` config flag is set (checked), the slot is configured for the `operator` permission level, and can be accessed by a user who has been assigned a role with the minimum permission set in the **Role Manager** view to `operator`-level read (r). In other words, any user assigned this role may access the slot at least to read it.

With the `admin` permission level, users can see and change slot flags from the slot sheet of any component. By default, most slots are configured for the `admin` permission level (the out slot is typically set to the `operator` permission level).

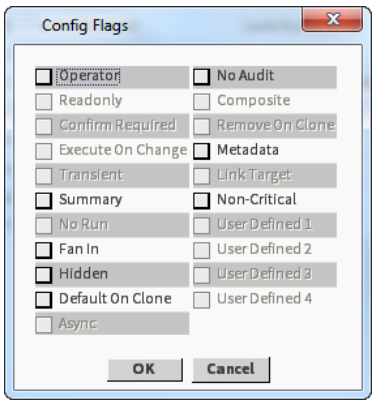
Changing a component Config flag

Config flags set up system features at the component level.

Step 1 Display the component slot sheet.

Step 2 Right-click the slot and click **Config Flags**.

The Config Flags editor appears.



Step 3 Click in the `Operator` check box to set the config flag and click **OK**.

UserService permission levels

By design, the **`UserService`** component enjoys a special permission level scheme—one that varies from the scheme described for other component access.

By default, these user properties appear as slots in the **`UserService`**:

- Email
- Password
- Cell Phone Number
- Facets (time format and unit conversion)

The `Operator` config flag for these slots may be enabled (checked) and disabled (unchecked) just as you would configure the permission level on any other slot. The special scheme that applies only to the **`UserService`** component yields the following results:

- If the `operator` permission level is enabled (`Operator` checked) on the slot, and the role assigned to the user grants read permission (r), the user is allowed read-only access to the user properties (email, password, etc.) on their own user account (all other users are hidden).
- If the `operator` permission level is enabled (`Operator` checked) on the slot, and the role assigned to the user grants write permissions (rw), the user is allowed both read and write access to the user properties on their own user account (all other users are hidden). This is the configuration required to allow a user to change their own password.
- If the `admin` permission level is enabled (`Operator` unchecked) on the slot, and the role assigned to a user grants read permission (rR), the user is allowed read-only access to all user properties for all available users.
- If the `admin` permission level is enabled (`Operator` unchecked) on the slot, and the role assigned to a user grants write permissions (rwRW), the user is allowed both read and write access to all properties for all available non-super users. Moreover, they have access to the **User Manager**, and can add new users and delete selected users. In addition, the **Permissions Browser** view of the **`UserService`** is available to them.

NOTE: A user cannot assign permissions to other users that they do not have themselves. For example, a non-super user cannot make another user a super user.

To allow each user to change their own password, but not have access to other users' passwords, you would set the config flag for the **`Authenticator`** slot to the **`operator`** permission level (checked; this is the default for this slot), and assign a role to the user that grants `operator`-level write (rw) permissions.

All non-super user roles should be configured for `operator`-level write (rw) permissions applied to the category that contains the **`UserService`**. (By default, the **New Station Wizard** assigns the **`UserService`** to the `Admin` named category (category 2), along with the **`CategoryService`**.)

NOTE: Any user granted super user permissions has access to all components, and can add more super users.

Categories

Categories are groups to which each station component may be assigned for the purpose of managing who has permission to access the component.

The system provides two types of categories:

- Basic (or explicit) categories are groups (collections) to which you assign each component.

  Each component maintains a bitmap for basic category membership. The default eight categories consume one byte and each increment of eight categories you add consumes an additional byte (1–8, 9–16, 17–24, and so on) in the component record.

- Inherited categories are passed down from component parent to component child.

All components must be assigned to at least one basic category, either an explicit assignment, or an inherited assignment from a parent component. Beyond this assignment, which type of category to use depends on your needs. You may use explicit and inherited categories at the same time.

Basic categories

The system maintains basic categories as indexes in an array. You individually assign components in the station to each category. Subsequently, you set up roles to grant permission to access components based on the category you associated with each component. Finally, you assign a role to each user.

A new station (created using the New Station wizard) comes with two default basic categories:

- `User` (Category 1)
- `Admin` (Category 2)

As the names imply, regular users may view and modify some station objects. Only administrators should have permission to access other objects. All objects default to the `User` category except for these, which default to the `Admin` category:

- The configuration services: **`UserService`**, **`CategoryService`**, and **`ProgramService`**
- All files (the entire file space)

You may add basic categories as needed. For example, you could group components by equipment type, such as `Lighting`, `Door access`, and `HVAC`. An alternate scheme might group components by geography: `Floor 1`, `Floor 2`, and `Floor 3`. How you group components depends on your overall building model, and specifically on how you plan to set up roles, permissions and users.

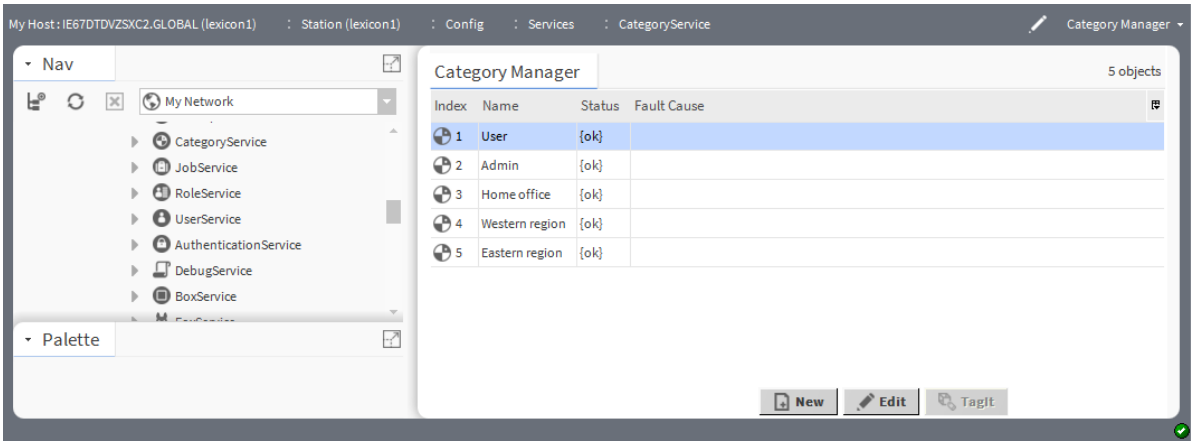
NOTE: Basic categories use station memory. To improve system performance, minimize the number of categories, and keep category indexes contiguous.

Adding and editing a basic category

You add and edit basic categories using the **Category Manager**.

Step 1 Right-click the **CategoryService** and click **Views→Category Manager**.

The **Category Manager** view opens.



The rows in the **Category Manager** view represent existing categories.

Step 2 Do one of the following:

- To edit the name or index of an existing category, double-click the category row in the table.
- To create a new category, click the **New** button.

Step 3 In the **New** window, choose `Category` for **`Type`**, select the number of categories to add, and click **OK**.

Selecting `Category` from the **`Type`** list allows you to assign a name to one or more basic categories.

Step 4 Enter a name and an index for each category and click **OK**.

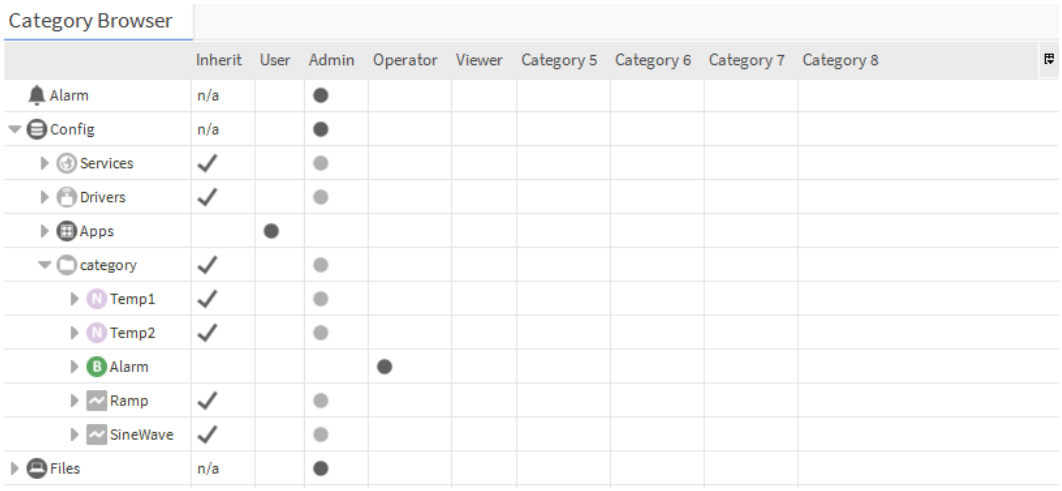
The name(s) replace the default “Category 1,” “Category 2”, etc. names.

Assigning a component to a basic category

You use the **Category Browser** to assign individual components to basic categories. Components may belong to more than one category at the same time.

Step 1 Right-click Expand **CategoryService** in the Nav tree and click **Views→Category Browser** in the Nav tree.

The **Category Browser** view appears. Use this view to assign components to categoriesategory



By default this view shows the component types collapsed into rows in a tree structure: Alarm, Config (components), Files, and History. The columns represent the categories in the station. The column titles identify the categories.

Step 2 To view all components that have category assignments, click the binocular icon) on the toolbar.

All components appear in the table.

Step 3 Use the expandable tree to navigate to components of interest in the table. To return to the previous collapsed view, select the `Category Browser` from the drop-down list of views.

Step 4 As needed, in any component row, click either:

- In the category column to assign a component to a category or click again (toggle) to remove the component assignment from the category.
- In the Inherit column to assign a component to any of the categories its parent is assigned to.

With the exception of the root components, Alarms, Config, Files, and History, each object must belong to at least one basic category or inherit its parent's category assignments. The root objects cannot inherit. They must belong to one or more categories.

NOTE: When an admin user (user with Admin write privilege on the **CategoryService** and on a particular station component being adjusted) makes a category mask adjustment, the user must also have at least the Operator write privilege on the category being adjusted in the category mask for the station object. This includes changes to check marks in the "Inherit" column—the user must have at least Operator write access to any altered categories applied from the Inherit change.

Using the **Category Browser** to assign a component to a basic category updates the component's category bitmap. Copying the component to another station or saving it in a bog for reuse includes the category bitmap.

Deleting a category name

You delete a category name using the **Category Manager**.

Prerequisites: The category has been added. You are viewing the Category Manager.

Step 1 To delete a category name, click the row in the **Category Manager** view and press **Delete** or right-click the row and click **Delete**.

Deleting a category name changes the name back to the generic index name of Category 1, 2, etc. It does not remove the category index from the object(s) with which it is associated.

Roles and permissions

Once a user has been authenticated, the user is granted or denied the right to access each protected object in the system using a permissions map (a category-based matrix), which is defined by the role assigned to the user. Permissions define the rights a user has within each category of station objects.

Roles ease the management of permissions for a large number of users. The permissions for a group of users who are assigned to the same role can be updated by changing the role. This saves having to update each user's permissions individually.

For example, if 40 operators need access to a new component in the station, you may need to update only their shared operator role, and then only if a category has been added or permissions need to be changed. The initial configuration of a station's security, which involves object permission levels, object categories, roles (permissions) and users may take time to design and set up in some configurations, but the trade off is worth the future time saved when updating the permissions of more than one user at a time.

CAUTION: There are risks involved in giving any user broad permissions on the Role Service. For example, giving a user **admin write** permissions on the Role Service allows that user to create, edit, rename or delete any role. Best practices recommend that such permissions on the Role Service be limited only to appropriately authorized users.

Adding roles and permissions

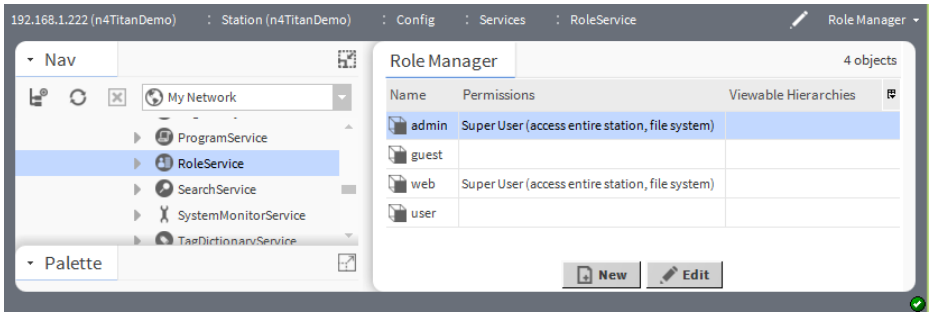
You add roles using the station's **Role Manager** view (**`RoleService`**).

Prerequisites: You enabled the `Operator` config flag enabled for any restricted components. Categories have been created and any basic categories assigned to components.

Most companies require, as a minimum, an administrator (super user) role, a manager role, and a regular user or operator role.

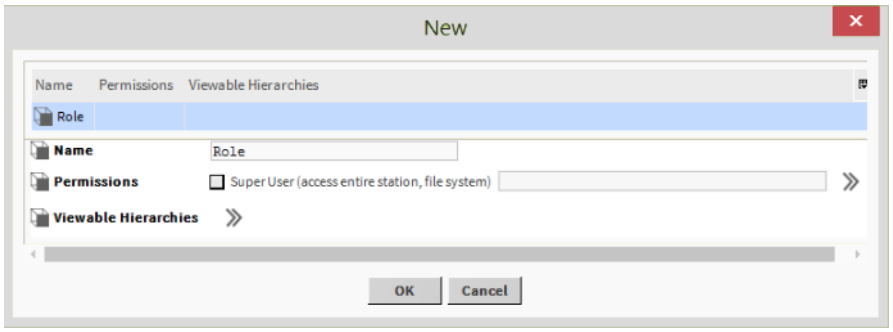
Step 1 Right-click **RoleService** in the Nav tree, click **Views→Role Manager**.

The **Role Manager** view opens.



Step 2 Click the **New** button, enter the number of roles to create in the pop-up window and click **OK**.

The system displays the **New** window with a row for each role you are creating.



Step 3 Name the role.

Step 4 To configure a role as a super user, click the `Permissions` check box for `Super User`.

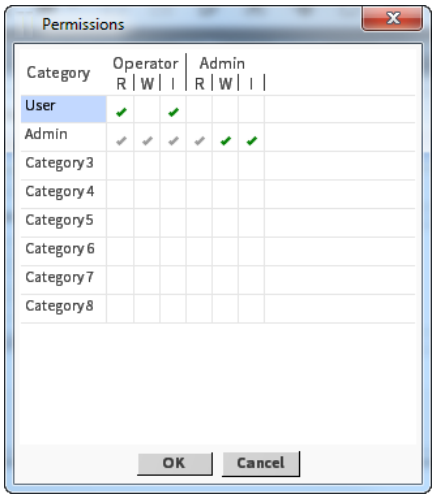
The built-in `Admin` role grants all possible rights for every category (super user). Only when logged in as the `Admin` user, or another super user, can you assign super user rights using the `Super User` check box.

In general, assigning super user rights should be strictly limited and based on special needs. For example, a Supervisor station may need super user rights to connect with other station clients (machine login vs. login by a person) in scenarios where Program objects are exported from stations using ExportTags. Human users may need super user rights to add and edit Program or Robot components.

CAUTION: Do not make it a common practice to give station-to-station users admin privileges. If your network is breached, a station-to-station user could cause significant damage without drawing attention to what is happening.

Step 5 To set up individual permissions, click the chevron at the end of the **`Permissions`** property.

The **Permissions** map opens.



The first column, Category, lists the groups to which you may grant permission. The Operator and Admin columns relate to the permissions level configured on each component. Below these headings are the cells to use for assigning one of three permissions to each category:

- R = Read allows the user to view the object.
- W = Write allows the user to change the object.
- I = Invoke allows the user to initiate an action related to the object.

Depending on how the permission level is set on the slot, six permissions are derived:

- To allow a user to view `operator`-level information, check the `Operator` config flag on the slot and select the Operator R column on the permission map.
- To allow a user to modify `operator`-level information (if it is not read-only), check the `Operator` config flag on the slot and select the Operator W column on the permission map.
- To allow the user to view and invoke `operator`-level operations (actions), check the `Operator` config flag on the slot and select the Operator I column on the permission map
- To allow the user to view `admin`-level information, leave the `Operator` config flag unchecked on the slot and select the Admin R column on the permission map.
- To allow the user to modify `admin`-level information (if it is not read-only), leave the `Operator` config flag unchecked on the slot and select the Admin W column on the permission map.

- To allow the user to view and invoke `admin`-level operations (actions), leave the `Operator` config flag unchecked on the slot and select the Admin I column on the permission map.

When you assign permissions, higher-level permissions (green check marks) automatically include the lower-level ones (gray check marks). For example, if you enable `admin`-level write (W), the system automatically enables `admin`-level read (R), as well as `operator`-level read and write (RW).

Step 6 Click the cell to assign a permission and click **OK**.

The e **`Permissions`** property displays the permissions.

Step 7 To finalize permissions, click **OK**.

Step 8 In a multi-station system, perform these same steps in each station so that each station has the same set of roles.

NOTE: During the network user synchronization process the framework sends the user's role assignment to the receiving station, however, it does not create the actual role(s) on the receiving station. You must set up matching roles on each receiving station before synchronizing network users.

Adding a component

Adding a component to your building model involves dragging the component from a palette, possibly setting the **Config Flag** on the component slot, and configuring the component to assign it to a category. A component may be a new network, device or service.

Prerequisites:

- If required, you have a license to add the component to your model.
- Any categories, roles (permissions) to assign to the component have been set up.
- The users who will access the component exist in the system.

Step 1 Open the palette that contains the component module.

Step 2 Expand the Nav tree to view the **Services** or **Drivers** container.

Step 3 Do one of the following:

- Drag the component from the palette to the **Property Sheet** or **Driver Manager**.
- Drag the component to the appropriate **Services** or **Driver** container in the Nav tree

The **Name** window opens.

Step 4 Change the name of the component, or use the default name and click **OK**.

Step 5 If you need to configure the permission level for the new component slot, right-click the component in the Nav tree and click **Slot Sheet**, then right-click the slot and click **Config Flags**.

- Leave the `Operator` config flag unchecked for the `admin` permission level (this level allows read and write access).
- Toggle this flag (checked) for the `operator` permission level (this level restricts user access to a minimum of read-only permission).

Step 6 To assign the component to a category, do one of the following:

- Add the component to the category using the **Category Browser**.
- Add the component to the category using the component's **Category Sheet**.

Step 7 If you created a new basic category, update the role assigned to users of this component to include permissions for the new component, otherwise confirm that the role includes the category.

Step 8 Confirm that component **Status** is `{ok}`.

You are ready to configure the component.

Editing roles and permissions

The primary reason for editing a role is to update permissions.

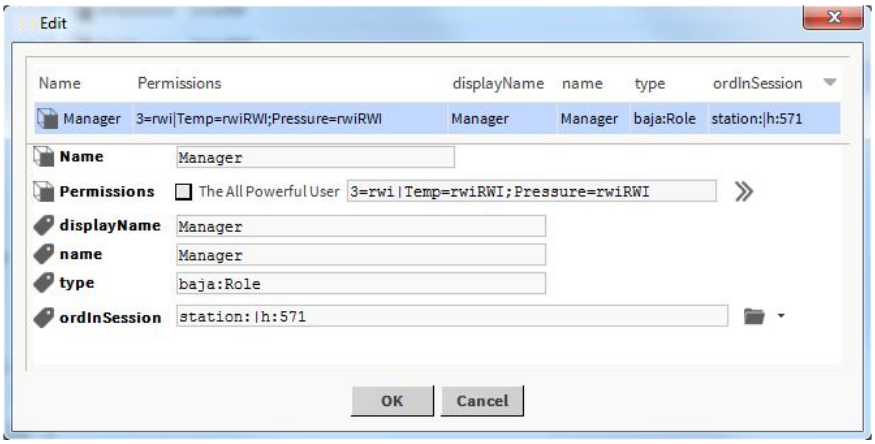
Prerequisites: The permission level is set appropriately on all components. Categories have been created. Roles have been created.

Step 1 Right-click Expand **RoleService** in the Nav tree and click **Views→Roll Manager** .

The **`Role Manager`** view opens.

Step 2 Double-click the row for the role to edit or select the row and click the **Edit** button.

The **Edit** window opens.



Step 3 To change permissions, click the chevron to the right of the **`Permissions`** property.

The **Permissions** map appears.

Step 4 Update the permissions as needed and click **OK**.

Step 5 To finalize the changes, click **OK**.

Assigning roles to users

This task associates the permissions defined by a specific role with each user.

Prerequisites: Roles and users have already been created. In a multi-station system, each station has the same set of roles.

Step 1 Right-click **UserService** in the Nav tree, click **Views→User Manager**.

Step 2 Double-click the user's row in the **User Manager**.

The user's **Property Sheet** appears.

Step 3 For the **`Roles`** property, select one or more role names by clicking in the check box and click **OK**.

Confirming access security

You should test each user's access rights before allowing users to use the system.

Step 1 Log out of the station.

Step 2 Log back in as a user that represents each role.

Step 3 Confirm that the Nav tree shows only those components to which the user has read, write or invoke action rights.

Reviewing permissions

The **Permissions Browser** view displays the rights granted to each role/user.

Prerequisites: Roles exist with permissions granted.

NOTE: In Niagara, there is added support for the **UserService** in the **Permissions Browser** view. Use the **Show Permissions for** drop-down list to switch between permissions for Users, and for Roles. When viewing permissions for **Users**, the view displays a separate column for each user as well as any prototype.

Step 1 Log in to the station as a super user or as the `Admin` user.

Step 2 Right-click RoleService (or UserService) in the Nav tree and click **Views→Permissions Browser** .

Step 3 Expand the Nav tree to view permissions for each role (or User).

For more details on the improvements to the Permissions Browser view for Niagara, see “wbutil-PermissionsBrowser” in the Components, views and windows section of this document.

NOTE: Although you may double-click any object row in the **Permissions Browser**, view the permissions map and update permissions for an individual user, this method of updating permissions does not change the permissions as configured in the user’s role.

Ancestor permissions

Often, you may wish to grant a user the right to access components using categories that are not included in the component’s parent, such that, permissions to a component's ancestor tree are not explicitly granted. In this case, the system automatically grants `operator` permission level read-only access to all ancestor components in the component tree. Otherwise, a user would be unable to navigate to target component in the Nav tree.

This automatic ancestor permission level assignment is done by the station periodically, but you can force it at any time with a right-click **Update** action on the **CategoryService** node in the Nav tree.

File permissions

By default, the New Station Wizard assigns the entire station’s File space to category 2 (`Admin`). A station's config.bog file and config.bog.backup files are not accessible (even by super users) in the station's file space. If needed, other station files and folders may be hidden from a remote station.

Users typically require that the role assigned to them have `operator`-level read permission on station file folders, such as ^nav, ^px, ^images, ^html, and so on. However, permissions higher than an `operator`-level read on the `Admin` category should only be assigned to selected users on an as-needed basis. In most situations, creating a new category containing *only* the components a user needs to access is a more appropriate solution.

Largely, rights granted to access categories that are used by files and folders are `operator`-level permissions as follows:

- Files require `operator`-level read (r) access to view, and `operator`-level write (rw) access to edit a file (if applicable). For instance, a user with `operator`-level write and write (rw) access to an .html file can modify it using the Text File Editor in Workbench.
- Folders (directories) require `operator`-level read (r) access to list and to copy child files, and `operator`-level write (rw) access to create new (or delete existing) child files.

A few views of files require `admin`-level Write (rwRW) permissions to access, such as the **`Nav File Editor`** for a Nav file. There are also these special case file permissions:

- The system automatically restricts any system module files to `operator`-level read (r) access.
- If a user is not a super user, the system denies all access outside of the station's home directory.
- Users need `admin`-level Read (rwRW) access to see a Supervisor station's `^provisioningNiagara` folder (written to by the Supervisor's provisioning mechanism).

- If you have a developer license, your system includes an additional category called NModuleDevFilePermission. This category grants rwRW permission to access all system modules.

History permissions

Histories require that the `operator` permission level be set and `operator`-level read (r) permission granted by the role to access all available views.

History views include History Chart, Collection Table, and History Table). This includes the ability to rename a history.

Component permissions troubleshooting

To begin troubleshooting component permissions, go back and review your original design and double-check all configuration properties.

I can successfully log in to a station, but I get the message, "User `username` does not have access to the station. Check permissions."

The user name you used to log in with is an authentic user name, but either no role has been associated with the user or the permissions contained in the role configuration do not allow the user to access the station. Log in with a user that has permission to access the station and update the configuration. A role needs permissions on at least one component for a user to be able to access the station.

I set up categories and roles, but my users can still access some components they should not be able to access and cannot access others that they should be able to access.

Confirm that you configured the component permission level correctly for each component.

Open the **Category Browser** and review the categories the user has permission to access. Verify that there are no unexpected components in the categories.

My users cannot change their own passwords.

You need to assign a role to each user that grants write access (rw) to the **`Password`** slot on the **`UserService`**.

Chapter 5 Components, views and windows

Topics covered in this chapter

♦ Components
♦ Plugins (views)
♦ Windows

The user interface includes components, views and windows, which provide the means for communicating with the system.

The Help topics include context sensitive information about each component and view, as well as information about individual windows.

Components

Components include services, folders and other model building blocks associated with a module. You may drag them to a property or wire sheet from a palette.

Descriptions included in the following topics appear as context-sensitive help topics when accessed by:

- Right-clicking on the object and selecting **Views→Guide Help**
- Clicking **Help→Guide On Target**

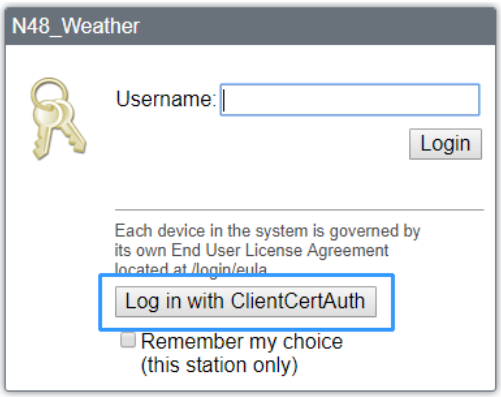
Client Cert Auth Scheme (clientCertAuth-ClientCertAuthScheme)

This component provides one method of verifying that a user is authorized to log in to a station. Provided by the **clientCertAuth** palette, the **ClientCertAuthScheme** is an authentication mechanism requiring that the user enter his or her password as well as a certificate.

With this scheme, each user object has an authenticator containing a public certificate, which matches the user certificate's private key. Additionally, each certificate must be added to the server socket's **TrustAnchor** list. During a login attempt, the user is prompted to upload the certificate. The server verifies that the certificate matches the certificate stored on the User object.

Adding this component to the station **AuthenticationSchemes** node adds a button to the **Login** window. The text label on the button is configurable via the **Login Button Text** property. By default, the button text label is "Sign in with SSO" but you should change this to your preferred text.

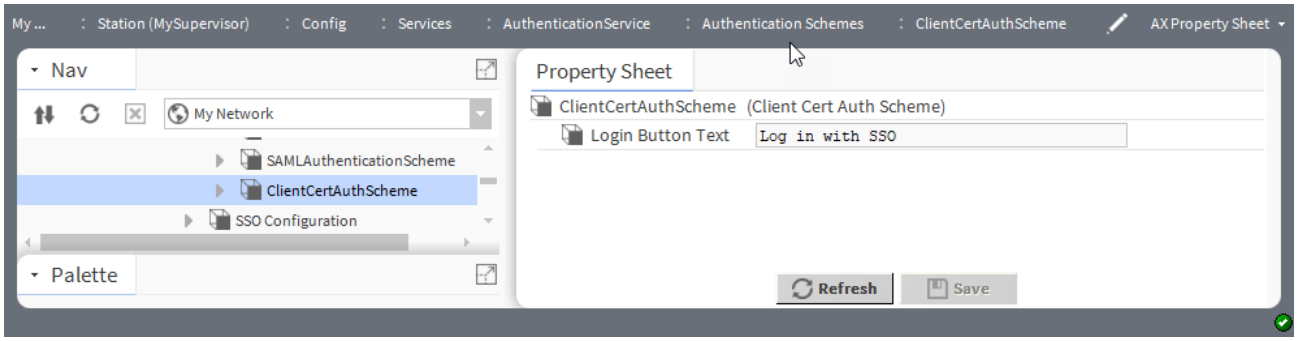
Figure 10 Example configured button visible in login window



NOTE: In the example shown, the user object is configured to use clientCertAuthScheme authentication, and to reset his or her authentication scheme on login (via the User **`Force Scheme Reset To`** property).

For additional information, refer to Admin/User workflow for client certificate authentication in the "User Authentication" chapter.

Figure 11 Certificate authorization property



To access this property, expand **Config→Services→AuthenticationService→Authentication Schemes** and double-click **ClientCertAuthScheme**.

Property	Value	Description
Login Button Text	text	Configures the text that appears on the login button.

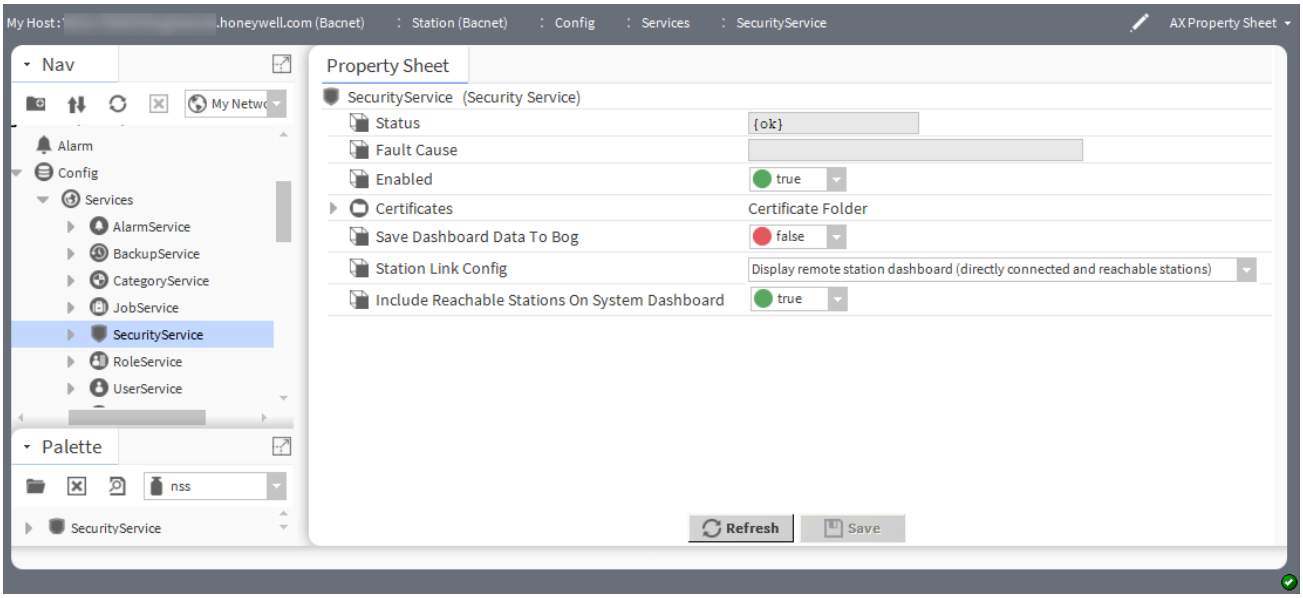
SecurityService (nss-SecurityService)

This component monitors certificates and generates alarms for those that are about to expire. Other station services with configurable security settings report to the **SecurityService**. The component is available in the **`nss`** palette

NOTE: Manually installing the nss modules (nss-rt, -ux, -wb) requires a station restart. Otherwise, when you click on the added **SecurityService** component, a "Not Found" error message displays in Workbench. In cases where you restart the station after installing the modules you do not see this error. Also, uninstalling the nss modules causes an auto restart of the station.

The **Security Dashboard** is the default view for the Security Service. For complete details on the view, refer to " nss-SecurityDashboardView, page 126" in the "Plugins" section of this guide.

Figure 12 SecurityService properties



To access these properties, expand **Config→Services**, right-click **SecurityService** and click **Views→AX Property Sheet**.

In addition to the standard properties (Status, Fault Cause, and Enabled), the following configuration properties are present.

Name	Value	Description
Certificates	additional properties	Contains the certificates in use in the station. For each certificate listed the following read-only data is shown.
Save Dashboard Data to Bog	true, false (default)	In a supervisor (or other station licensed for the system security dashboard), if set to `true`, the dashboard information is stored in the station's `.bog` file. This makes the information available immediately on station restart, instead of having to fetch it from the remote stations. If set to false (the default), the data will not be saved to the `.bog` file. On station restart, the system dashboard data will not be available until a dashboard refresh is triggered (via the action on the service).

Name	Value	Description
Station Link Config	drop-down menu	Determines the links on the System Security Dashboard. `Display remote station dashboard (directly connected stations only)` (default): On the System Security Dashboard, the link navigates you to the remote station's Security Dashboard for directly connected stations, but for reachable stations it links you to the Security Dashboard Extension on the Niagara station on the Supervisor. `Display local view of the remote station dashboard`: For all stations, the link navigates you to the Security Dashboard Extension on the Niagara station on the Supervisor. `Display remote station dashboard (directly connected and reachable stations)`: For all stations, on the System Security Dashboard the link navigates you to the remote station's Security Dashboard.
Include Reachable Stations On System Dashboard	`true` or `false` (default)	Enables the multi-tier system security dashboard. After enabling, the next time you view the Security Dashboard, the reachable stations are automatically added.

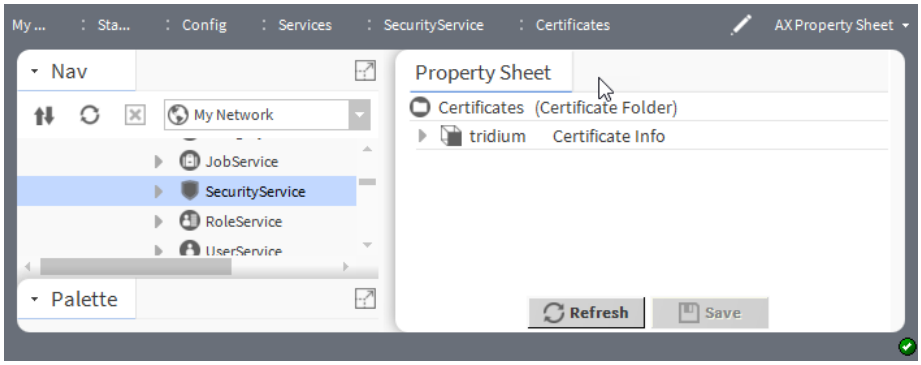
Actions

Refresh System Dashboard Data: This action fetches the Security Dashboard data from all remote stations (or all stations matching the provided filter).

Certificate Folder (nss-CertificateFolder)

This component serves as a container for certificates

Figure 13 Certificate Folder



This folder contains no unique properties of its own.

Certificate Info (nss-CertificateInfo)

This component provides information about specific certificates.

The SecurityService automatically populates the Certificates folder with a CertificateInfo component for any existing server certificate installed on the platform that is used by one of the Services (e.g. WebService, FoxService). You add this component from the **SecurityServiceElements** folder in the **nss** palette.

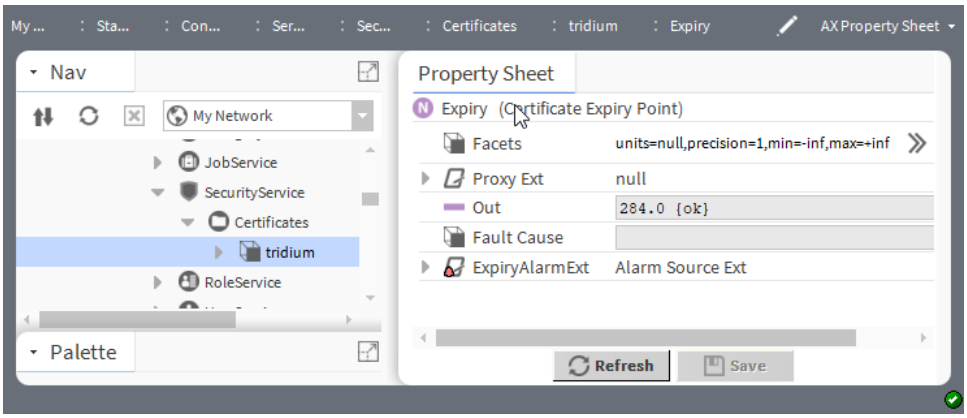
Figure 14 Certificate Info properties

Name	Value	Description
certificate name	text	Displays the certificate name/Alias
Expiry	number	Displays the number of days until/since the certificate expiry date, and the certificate status.
Used In	namd and ORD slot path	Displays the name and ORD/slot path of the Service using the certificate (e.g. WebService, FoxService).

Certificate Expiry Point (nss-CertificateExpiryPoint)

This component configures a numeric certificate expiration point.

Figure 15 Certificate Expiry Point properties



To access these properties, expand **Config→Services→SecurityService→Certificates→CertificateInfo** and click **Expiry**.

Property	Value	Description
Facets	Config Facets	Defines the unit of measure for a numeric value: `units` defines the unit of measure, for example from `acceleration` to `volumetric flow`. `precision` defines the number of decimal places.
Proxy Ext	additional propertiy	Provides access to the null control point.
Out	read-only	Reports the current value of the point.

Property	Value	Description
Fault Cause	read-only	Provides a brief explanation of the fault cause, if the point is in fault.
ExpiryAlarmExt	additional properties	Configures an alarm source extension for the point. The topic "alarm-AlarmSourceEx" in the *Alarms Guide* documents these properties.

Google authentication scheme (gauth-GoogleAuthenticationScheme)

The Google Authentication Scheme is a two-factor authentication mechanism that requires the user to enter their password as well as a single-use token when logging in to a station. This protects a user's account even if their password is compromised.

The component is available in the **gauth** palette.

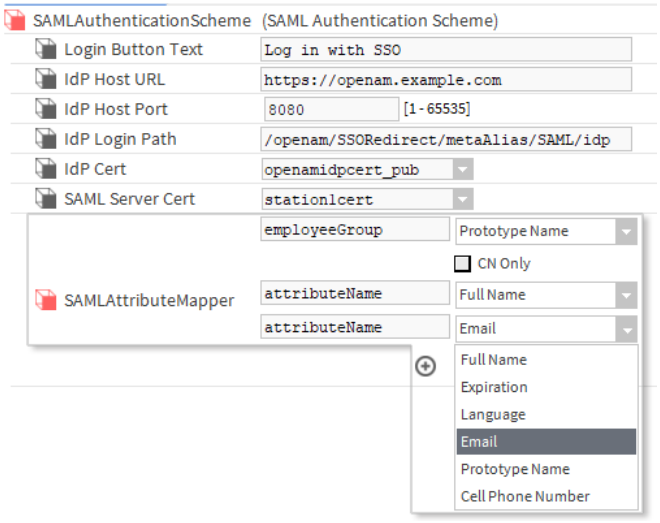
S A M L Attribute Mapper (saml-SAMLAttributeMapper)

This component configures the SAML Authentication Scheme to map specific SAML attributes to properties in the User Prototype.

During SAML Single Sign On, the SAML Identity Provider (IdP) may send the Service Provider (SP) various attributes. These may contain information about the user, and can be used by the station to build the user object. Many SAML IdPs can be configured to return the attributes with a customized name. However, other IdPs may not be configurable, or IT restrictions may prevent configuring an IdP that supports this feature. It is when the IdP is not configurable that you can use this component to configure the user prototype.

To use the SAMLAttributeMapper, drag it from the **saml** palette to the **SAMLAuthenticationScheme** component in the Nav tree.

Figure 16 Opening the SAML Attribute Mapper



The IdP-provided documentation indicates which SAML attributes are coming in from the IdP. As an alternative, you can install a SAML add-on to your web browser, which lets you view the attributes coming in from the IdP. For example, there is the SAML DevTools extension for Chrome, which you can use.

In some cases, an IdP sends back multiple values for the **prototypeName** attribute. If the IdP sends back multiple **prototypeNames** after you install the following patches, the **SAMLAuthenticationScheme** considers all returned values and extracts the one that appears highest on the list of UserPrototypes. This is similar to how LDAP works.

- For Niagara:
  - **saml-rt-4.x.xx.xx.x**
  - **saml-wb-4.x.xx.xx.x**

User properties that can be mapped from SAML attributes

- Full Name
- Expiration
- Language
- Email
- Prototype Name
- Cell PhoneNumber

The UserPrototype that is associated with the user supplies all other properties.

Default mappings

If no mappings are specified on the **SAMLAuthenticationScheme**, the following mappings are used.

SAML Attribute Name	User Property	Extra Information
Full Name	fullName	Not applicable.
Expiration	expiration	Format: D-MMM-YY h:mm:ss zz
Language	language	Not applicable.
Email	email	Not applicable.
Prototype Name	prototypeName	Select the **CN Only** checkbox if the IdP returns multiple values for user prototype.
Cell Phone Number	cellPhoneNumber	Not applicable.

How attribute mappings are processed

Attribute mappings are processed as follows when a user logs in to the system.

1. Customized mappings are considered first. If there are multiple mappings to the same property, the first successful mapping is used. For example, if there were two mappings to the "expiration" property, and the first mapping failed to parse properly, the second mapping would be attempted. If the first mapping parsed correctly, the second would be ignored.
2. Once all customized mappings are processed, the default mappings will be attempted for any User property not yet mapped.
3. Any property not mapped from a SAML attribute will be pulled from the UserPrototype, if possible.

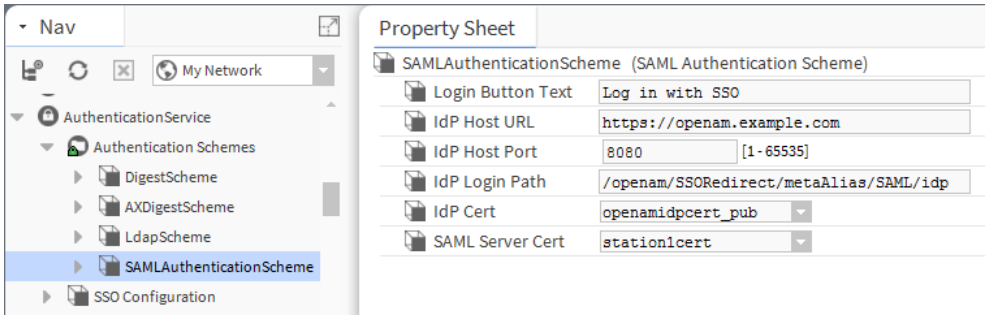
SAML Authentication Scheme (saml-SAMLAuthenticationScheme)

This component extends the SSO authentication scheme. A **SAMLAuthenticationScheme** component enables SAML SSO in the station. The scheme must be configured with a number of IdP configuration values. Typically these are obtained from the IdP SAML Server administrator.

Most SAML IdPs require you to provide an XML file with metadata about the service provider to add it to the SAML network. In Niagara, if a station is configured with a **SAMLAuthenticationScheme**, you can visit the following URL to automatically generate the station's SAML metadata XML: `https://host.domain.com/saml/samlrp/metadata?scheme=<schemeName>` (where you replace `<schemeName>` with the name of the station's **SAMLAuthenticationScheme**).

Since SAML is an open standard, a number of third-party SAML servers are available (for example, OpenAM, Salesforce, etc.). This example configures the authentication scheme for the OpenAm Identity Provider.

Figure 17 SAML Authentication Scheme properties



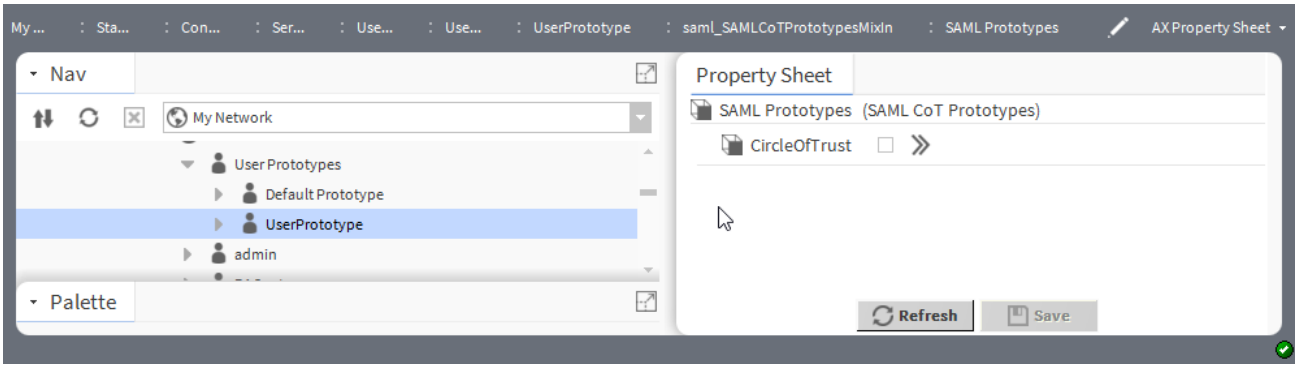
To access these properties, expand **Config→Services→AuthenticationService→Authentication Schemes** and double-click **SAMLAuthenticationScheme**.

Property	Value	Description
Login Button Text	text string, "Log in with SSO" (default)	Defines the preferred text label for the SSO login button that appears on the **Login** window. This button always displays if the corresponding scheme is in the authentication schemes folder.
IdP Host URL	text string, https://idp.domain.com (default)	Configures the URL for the host of your Identity Provider that provides the IdP data.
IdP Host Port	443	Configures the port number of your Identity Provider that provides IdP data.
IdP Host Login Path	/path/to/login	Configures the location of the Identity Provider that you must navigate to trigger SAML authentication for the IdP provided data.
IdP Cert	drop-down list	Identifies the certificate required to encrypt messages sent to the IdP, and validate messages sent from the IdP for the IdP provided data.
SAML Server Cert	drop-down list	Identifies the certificate used by the station to sign messages that are sent back to the IdP. This certificate is also provided to the IdP SAML Server admin so that the IdP can read and validate the messages. It also decrypts messages sent from the IdP to the station.

SAML CoT Prototypes (saml-SAMLCoTPrototypesMixIn)

This component serves as a folder for prototypes.

Figure 18 SAML CoT Prototypes property



To access these properties, expand **Config→Services→UserService→User Prototypes**, double-click **User-Prototype**, expand **saml_SAMLCotPrototypesMixIn** and double-click **SAML Prototypes**.

Property	Value	Description
CircleOfTrust	chevron	Opens the Edit Selected Prototypes window.

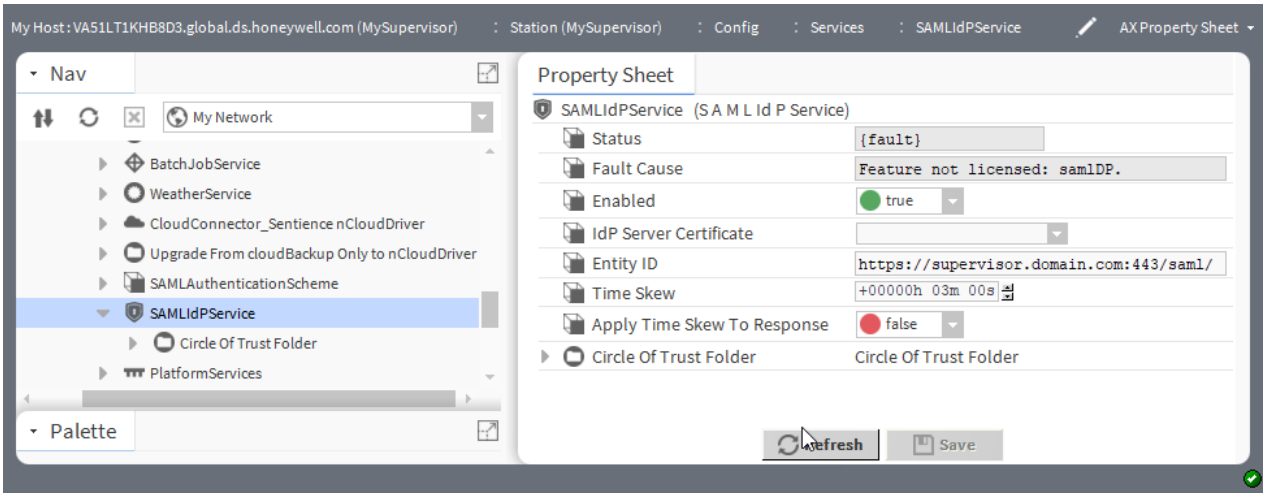
S A M L Id P Service (saml-SAMLIdPService)

This component takes advantage of SAML functionality without you having to set up an external IdP. Stations using Niagara that have access to this component.

The **`samlDP`** feature license is required to run this service.

You add this component to your station by expanding **SAML Identity Provider** in the **`saml`** palette and dragging the **SAMLIdPService** component to the **Services** folder in the Nav tree.

Figure 19 SAML IdP Service properties



To access these properties, expand **Config→Services** and double-click **SAMLIdPService**.

In addition to the common properties (status, and fault cause), this component has the following configuration properties.

Name	Value	Description
Enabled	`true` (default) and `false`	Enables (`true`) and disables (`false`) the service.
IdP Signing Cert	string	Identifies the server certificate as selected from the Supervisor station's User Key Store.
EntityID	string	Defines the Supervisor station's IP address (or hostname) plus the port number the **WebService** is running on (80 for http/ 443 for https). Always append the characters "/saml/" to the **`EntityID`** value. For example, if you entered "`https://192.68.19.20:443`" as the **`EntityID`**, you need to append to it "`/saml/`", so that it reads: "`https://192.68.19.20:443/saml/`
Time Skew	0000h 03m 00s (default)	Sets the number of minutes to extend the validity period of the SAML request from the subordinate station. This allows the SAML message to be accepted when the Supervisor and subordinate stations cannot synchronize their time values. Use positive values.
Apply Time Skew	`true` and `false` (default)	Applies (`true`) and ignores (`false`) the specified **`Time Skew`** setting to the response. For cases where a time difference exists between the Supervisor and a remote station, this applies the time skew to the response(s).
xmlEncrypter		

Circle of Trust Editor (saml-CircleOfTrust)

Located in the **SAMLIdPService**, this component specifies a collection of users that can log in to a collection of stations. It is useful for managing a select set of stations and the users logging in to those stations.

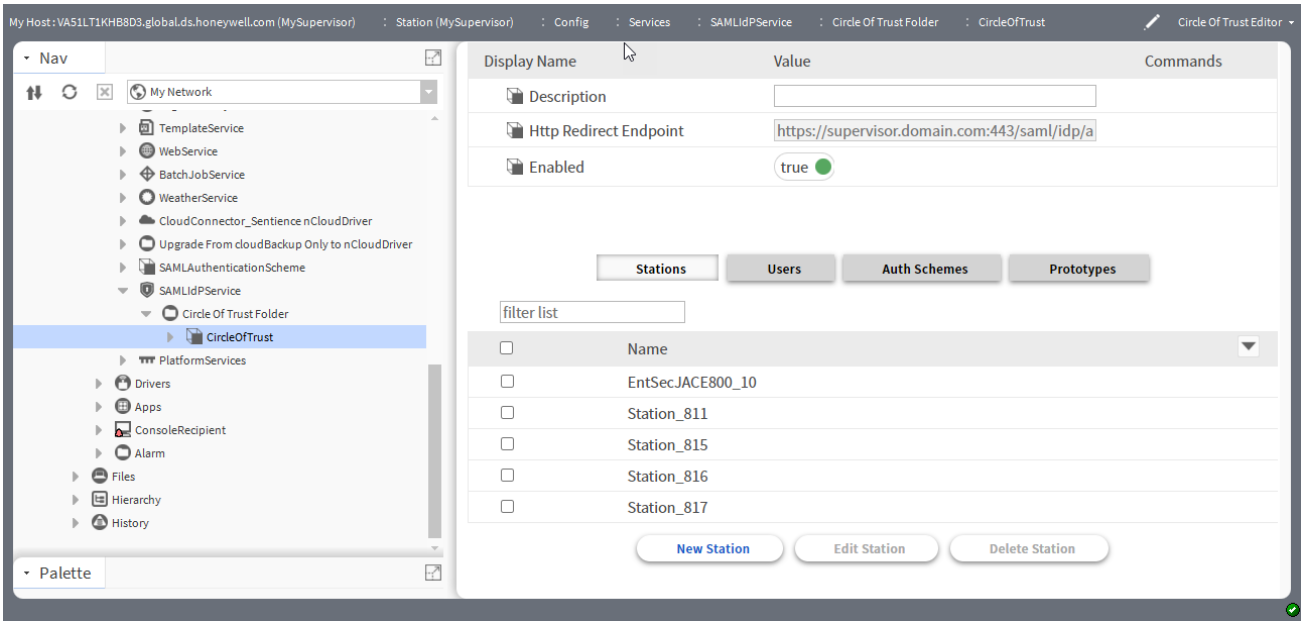
You configure this component's properties using the **Circle Of Trust Editor**, where you can name the circle, provide a description, etc.

Each circle must have a Supervisor that acts as the Identity Provider (IdP). The IdP manages only one circle. Only a Niagara station may serve as an IdP.

Hardware support depends on the release version of this feature.

This component is found in the **`saml`** palette.

Figure 20 Circle of Trust Editor properties



To access these properties, expand **Config→Services→SAMLIdPService→Circle Of Trust Folder** and double-click **CircleOfTrust** .

In addition to the standard property, Enabled, these properties configure this component.

Property	Value	Description
Description	string	Provides a name for this circle of trust.
Http Redirect Endpoint	read-only	Shows the URL for this circle of trust. This value (as well as the IdP Host URL+ IdP Host Port) configures the `IdP Login Path` in the remote station's SAML authentication scheme. Typically covered by the configure Niagara IdP and **SAMLAuthenticationScheme**'s provisioning job, but for stations not in the **NiagaraNetwork** the you must add the scheme manually.

Circles

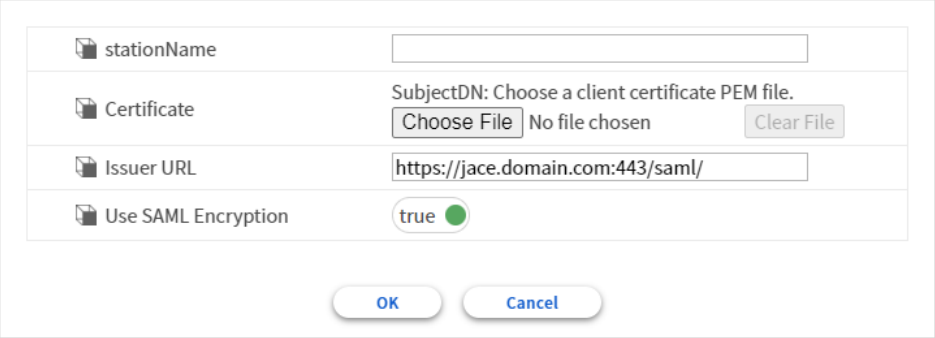
This editor applies to four groups of stations.

- Stations selects the station(s) to include in this circle of trust. You are not limited to stations in the **NiagaraNetwork**.
- Users selects user(s) from your **UserService** to include in this circle of trust. The selected users may log in to the stations in the circle.
- Auth Schemes specifies which authentication schemes may be used when logging in. This accommodates users who may not yet exist in your station. For example, you might specify the LdapScheme so that LDAP users can log in.
- Prototypes selects one or more user prototypes in the Supervisor's **UserService** that may be used when logging in to the remote station.

Stations buttons

Clicking the **New Station** button opens a window for setting up a new station. The **Edit Station** button opens the same window for an existing station. The **Delete Station** button removes the select station from the circle of trust.

Figure 21 Circle of trust New Station properties



Property	Value	Description
stationName	text string	Identifies the name of the station to add to the circle of trust.
Certificate	Choose File and Clear File buttons	Selects and clears the station's SAML certificate.
Issuer URL	URL	Identifies
Use SAML Encryption	`true` (default) or `false`	Enables (`true`) and disables (`false`) use of SAML encryption.

Buttons for the Prototypes circle

- **Add** opens an **Add Prototype** window with a single property used to identify the prototype.
- **Edit** opens a window for editing the name of the prototype.
- **Delete** removes the selected prototype from the circle of trust.

Prototype Folder (saml-SAMLUserPrototypeFolder)

This component serves as a container for the circle of trust user prototypes in the remote station's **UserService** and **UserPrototypes**, under the **CircleOfTrust**.

This component has no unique properties of its own.

Station Service Provider Folder (saml-StationServiceProviderFolder)

Located under the **CircleOfTrust** component, this is a container for the **StationServiceProvider** component.

This component is found in the **`saml`** palette. It has no unique properties of its own.

To find this folder, expand **Config→Services→SAMLIdPService→Circle Of Trust Folder→CircleOfTrust**, right-click **Service Providers** and click **Views→AX Property Sheet**.

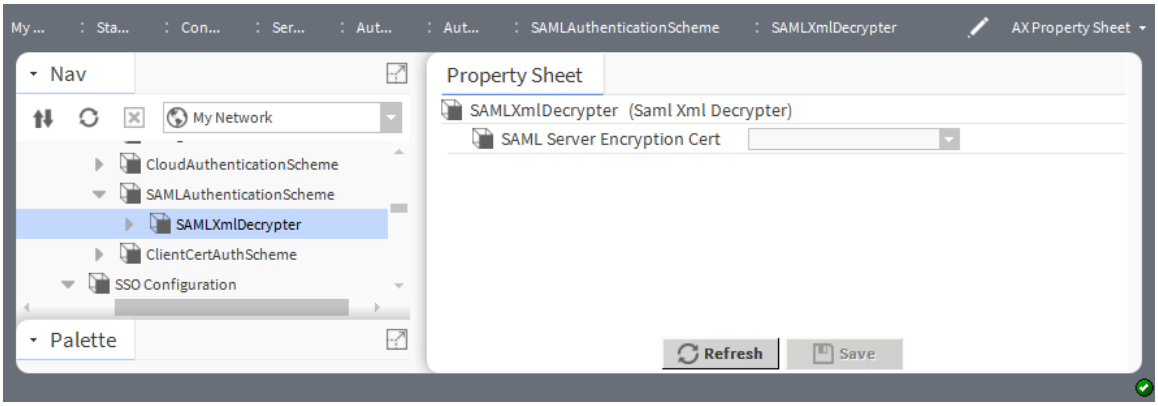
Saml Xml Decrypter (samlEncryption-SamlXmlDecrypter)

This component supports SAML EncryptedAssertions. If an IdP requires encryption, you can add a **SamlXmlDecrypter** to the **SAMLAuthenticationScheme**, and configure it with the encryption certificate from the **User Key Store**.

This component is available in the **samlEncryption** palette.

After adding the **SamlXmlDecrypter** to the **SAMLAuthenticationScheme**, you configure the decrypter's **SAML Server Encryption Cert** property with the appropriate encryption certificate. In some cases, you may be using the same certificate as the SAML server (signing) certificate.

Figure 22 Saml Xml Decrypter property



To access this property, expand **Config→Services→Authentication Service→Authentication Schemes→-SAMLAuthenticationScheme** and double-click **SAMLXmlDecrypter**.

Property	Value	Description
SAML Server Encryption Cert	drop-down list	Selects the certificate required by the SAML Server for encryption.

Saml Xml Encrypter (samlEncryption-SamlXmlEncrypter)

This component provides and XML encrypter.

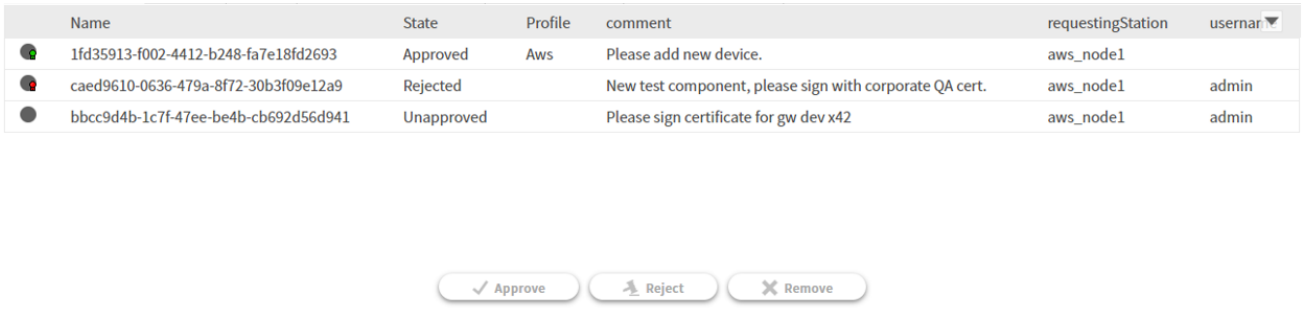
This component is available in the **samlEncryption** palette. You can drag it to the SAMLIdPService under **Config→Services**. It has no configurable properties.

Plugins (views)

Plugins provide views of components and can be accessed in many ways. For example, double-click a component in the Nav tree to see its default view. In addition, you can right-click on a component and select from its **Views** menu.

Session Token Manager View (signingService-SessionTokenUxManager)

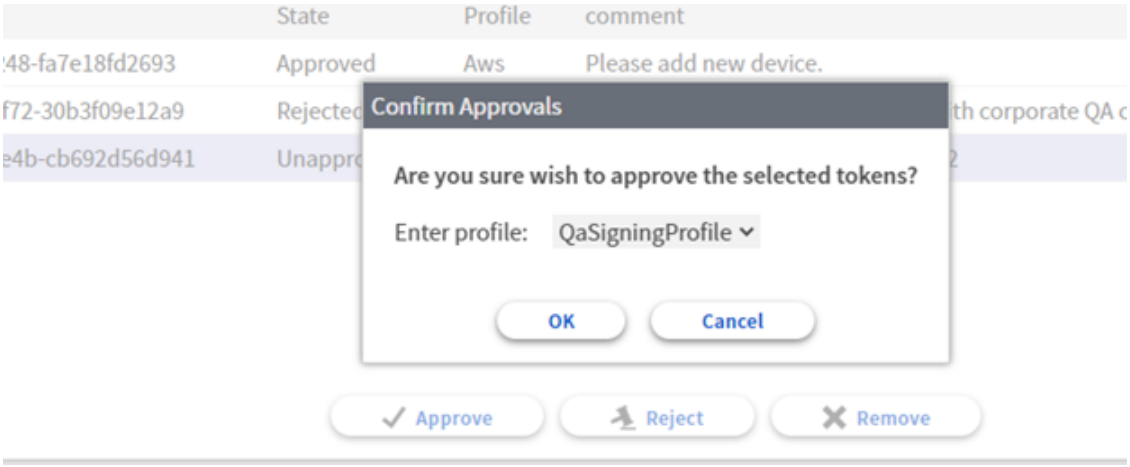
You can approve or reject tokens using actions on individual session tokens. Alternatively, you can use the **Session Token Manager** view, which is the default view for the store.



To access it, expand **Config→Services→SigningService→Transports→foxTransport** and double-click **Session Token Store**. The view displays each session token as a row that includes the status of the token plus all the metadata values submitted by the requesting component, which should help you on your decision to approve or reject the token.

NOTE: Only admin users are permitted to approve or reject tokens. They may batch approve, reject or delete session tokens by selecting multiple rows.

When approving a token, you must select the **Signing Profile** to which this request should be associated. The CSR will be signed by the CA certificate defined in that profile.



You can approve a rejected token at a later date provided the token has not automatically been removed before.

Security Dashboard View (nss-SecurityDashboardView)

The **Security Dashboard** is the main view for the Security Service. The view provides for administrators and other authorized users a snapshot of the security configuration of your station.

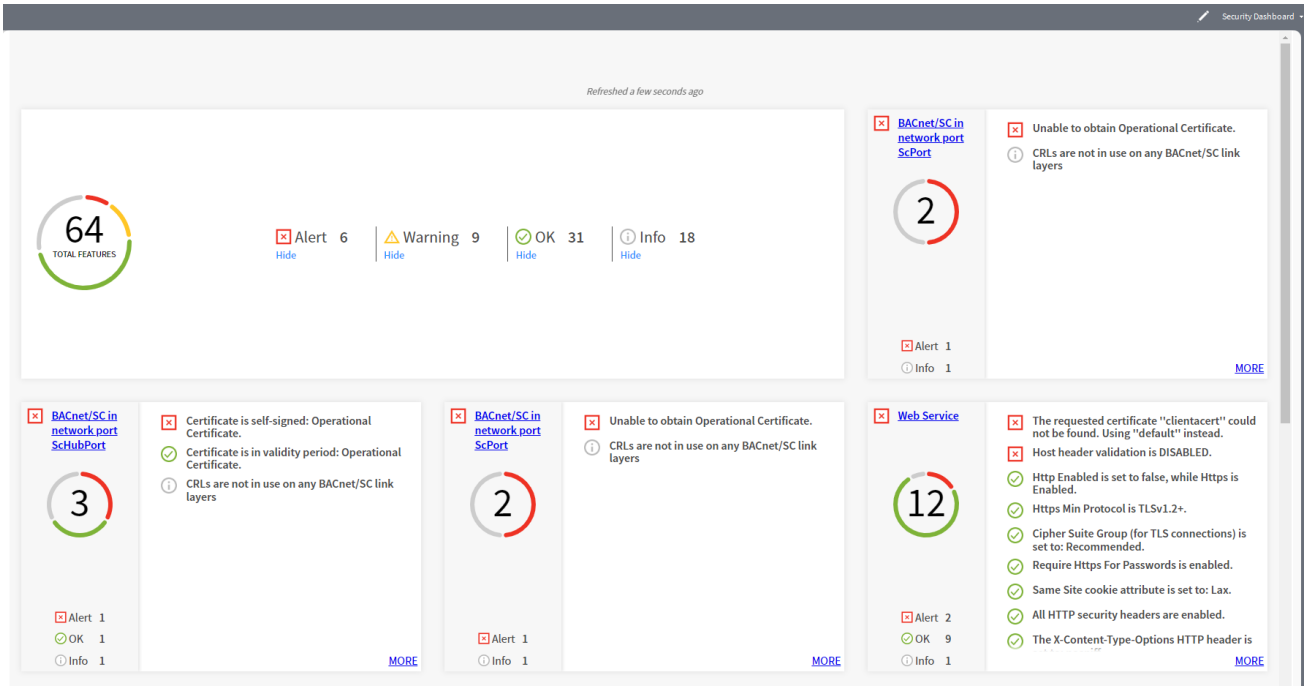
A license feature controls whether you can see the **System View** in the Security Dashboard. **System View** provides security details for each subordinate station in the **NiagaraNetwork**. To enable the **System View** feature, you need the "securityDashboard" license feature with the "system" attribute set to `true`. Without this setting, you see only the station dashboard (**Station View**) for the local station.

The Security Dashboard view is available in the following locations:

- **Services→Security Service**
- **Drivers→Niagara Network→{station name}→Security Dashboard Device Ext**

NOTE: The **Security Dashboard** transmits sensitive information. To minimize security risks, use the Foxs (secure Fox) protocol to manage platform connections. Also, the HTTPS protocol is enforced for secure communication over the network. The **Security Dashboard View** is not accessible over HTTP.

Figure 23 Example Security Dashboard View



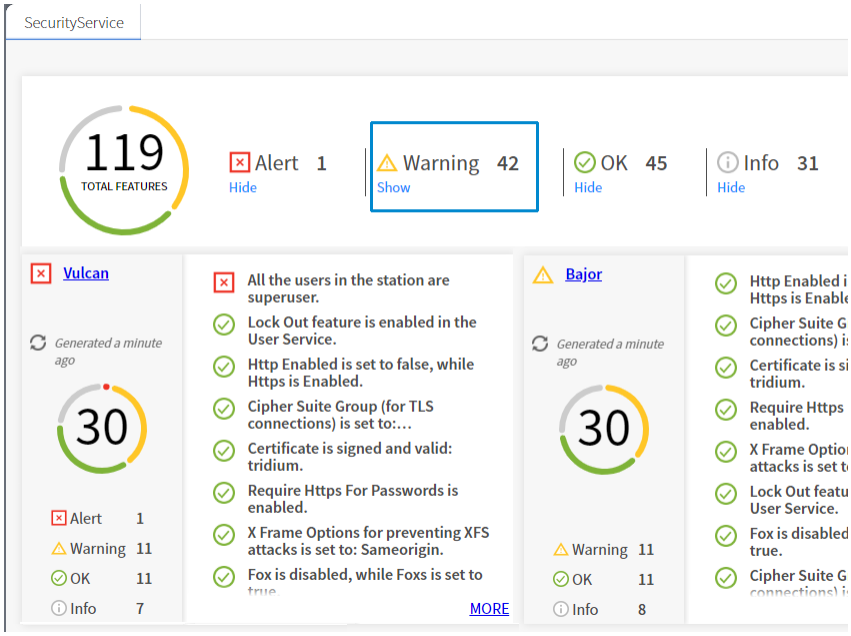
CAUTION: The **Security Dashboard View** may not display every possible security setting, and should not be considered as a guarantee that everything is configured securely. In particular, third party modules may have security settings that do not register to the dashboard.
For each "card" included in the view, a number of security-related items (for example. security settings on the FoxService shown in the FoxService card) are listed. Each card displays a status color which reflects the lowest status of any of its items.

- Gray Info icon ( ) indicates secondary information. For example, there is an info level that states how many users are in the station. You don't need to take a particular action. It is just presented for consideration.
- Green OK icon ( ) indicates the item's security status is good.
- Yellow Warning icon ( ) indicates a warning status on the item which means that the setting should be examined and possibly changed.
- Red Alert icon ( ) indicates an alert status on the item. The setting raises a security concern and should probably be changed.

Each card displays several of the most urgent items. If there are more items than fit on a card, a **More** button at the bottom of the card will pop up the full list of items for that service. Typically, a card provides a hyperlink to that particular service (or to a component) so that you can easily change the configuration. In cases where there is no component to link to, the pane provides no hyperlink. By default, the links on the individual cards in the Security Dashboard view link directly to the remote station. However, you can configure them using the **`Station Link Config`** property on the SecurityService component. For details, see SecurityService (nss-SecurityService), page 114.

The Summary card, which is located in the upper left corner, summarizes the number of security status messages for all services on the station. The Summary card features **Hide** / **Show** options, which allow you to hide, or show, all messages for one or more security status levels. For example, if you click the **Hide** option under Warning (as shown below) all of the Warning status messages for each card are hidden from view.

Figure 24 Example Summary card set to Hide all Warning status messages



Services reporting to the Security Dashboard include the following:

- Fox Service (for example, TLS status)
- Web Service (for example, TLS status)
- Authentication Service (for example, weak password strength)
- Debug Service (for example, FINE logs enabled)
- Module Permissions (for example, SEVERE permissions requested)
- Module Signatures (for example, modules unsigned)
- Program Objects (for example, unsigned program objects)
- Platform Settings (for example, TLS status)
- File System (for example, users with write access)
- User Service (for example, super user status)
- Syslog Settings (for example, Transport protocol status)

Other services and components may also be reporting to the Security Dashboard.

Additionally, the Dashboard is "pluggable" so that third parties can add their own security warnings for drivers.

Security Dashboard Refresh

In addition to the action available on the SecurityService, there are several ways that you can trigger a data refresh for this view:

- Attempting to retrieve the Dashboard data, for example, by viewing the Dashboard when there are no data available yet (possibly because the station has just restarted) triggers a refresh.
- An **Execute** action on the **NiagaraNetwork→Station→SecurityDashboardDeviceExt→Data Importer** refreshes the data for that station.
- A time trigger on the **NiagaraNetwork→Station→SecurityDashboardDeviceExt→Data Importer** that allows you to schedule a refresh. The default is to refresh daily.

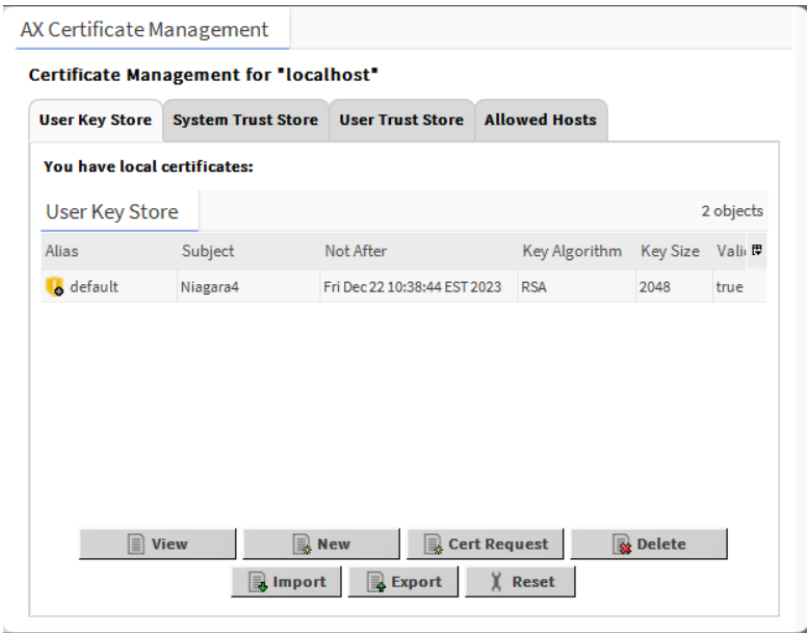
- The **Refresh System Dashboard Data** action on the SecurityService takes a String argument. It will refresh any station that matches that String. For example, the string, "`Richmond*`", will match any station that starts with Richmond; or "`*`"will match all stations).
- On the **System Dashboard View**, the card for each station has a **Refresh** icon ( ) next to the "Generated x time ago" text. Click the icon to trigger a refresh for the station.

Workbench Certificate Management (platCrypto-CertManagerTool)

This view accesses the Workbench key stores. You use it to create digital certificates and Certificate Signing Requests (CSRs), and to import and export keys and certificates to and from the Workbench stores.

You use this view to manage PKI (Public Key Infrastructure) and self-signed digital certificates to secure communication within the **NiagaraNetwork**. Certificates secure TLS connections to this host.

Figure 25 Certificate Management view



To access this view, click **Tools→Certificate Management**. It defaults to the User Key Store

This view has four tabs:

- **User Key Store** contains the root, server, client and intermediate certificates you create.
- **System Trust Store** contains the trusted, third-party, client certificates that commonly secure Internet servers.
- **User Trust Store** contains the trusted client certificates your company created to serve as its own Certificate Authority.
- **Allowed Hosts** contains approved self-signed certificates. These are certificates that you or someone else in your company knows to be secure certificates that can be used to encrypt data. These certificates cannot be used to authenticate a server because no root CA certificate in the **System Trust Store** or **User Trust Store** has signed them.

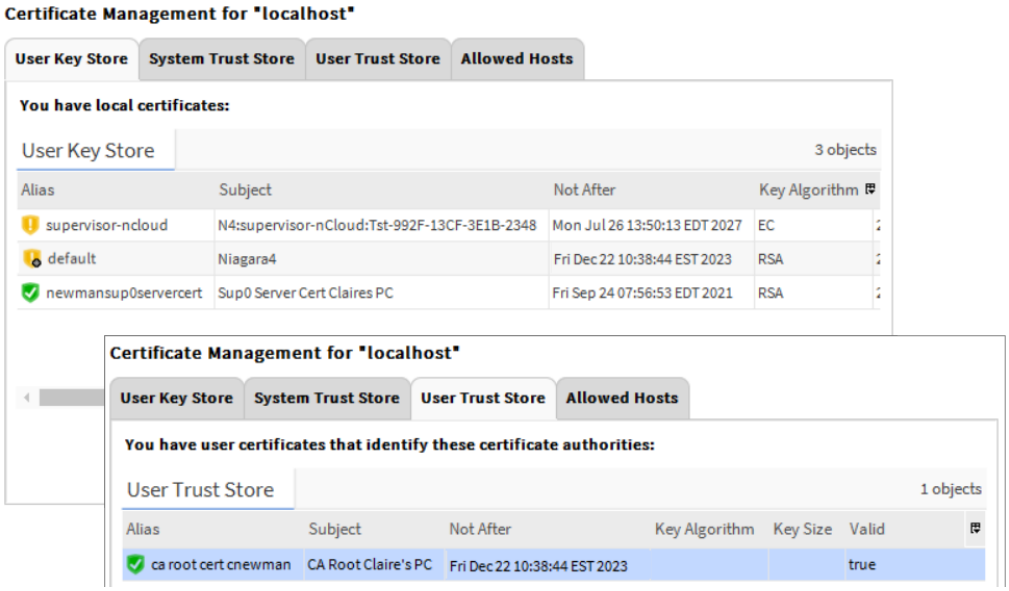
A separate topic documents each tab.

Platform Certificate Management (platCrypto-CertManagerService)

This view is the **Certificate Management** platform view on any Niagara host and the default view of the **CertManagerService** under a station's **PlatformServices**. Using this view you can create digital certificates

and certificate signing requests (CSRs), and to import and export keys and certificates to and from the Workbench, platform and station stores.

Figure 26 Certificate Management view for "localhost"



To access this view for the localhost stores, connect to the platform, expand **Platform** and double-click **Certificate Management** or, in the station, expand **Config→Services→PlatformServices** and double-click **CertManagerService**.

If you are viewing this topic from a guide other than the *Niagara Station Security Guide*, refer to the *Niagara Station Security Guide* for more information.

User Key Store tab

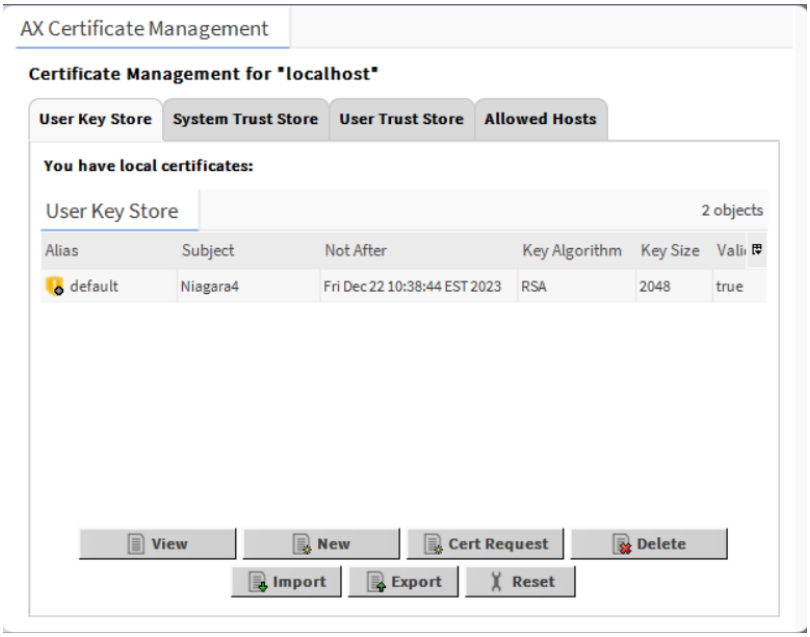
The **User Key Stores** contain server certificates and self-signed certificates with their matching keys. Each certificate has a pair of unique private and public encryption keys for each platform. A **User Key Store** supports the server side of the relationship by sending one of its signed server certificates in response to a client (Workbench, platform or station) request to connect.

If there are no certificates in a **User Key Store** when the server starts, such as when booting a new platform or station, the platform or station creates a default, self-signed certificate. This certificate must be approved as an allowed host. This is why you often see the certificate popup when opening a platform or station.

Default self-signed certificates have the same name in each **User Key Store** (`default`), however, each certificate is unique for each instance and is mainly used for recovery purposes. You cannot delete the default certificate.

Clicking the **New** and **Import** buttons also adds certificates to the **User Key Store**.

Figure 27 Example of a Key Store



Column	Description
Alias	Provides a short name used to distinguish certificates from one another in the **Key Store**. This property is required. It may identify the type of certificate (root, intermediate, server), location or function. This name does not have to match when comparing the server certificate with the CA certificate in the client's Trust Store.
Issued By	Identifies the entity that signed the certificate.
Subject	Specifies the Distinguished Name, the name of the company that owns the certificate.
Not Before	Specifies the date before which the certificate is not valid. This date on a server certificate should not be earlier than the **Not Before** date on the CA certificate used to sign it.
Not After	Specifies the expiration date for the certificate. This date on a server certificate should not be later than the **Not After** date on the CA certificate used to sign it. A period no longer than a year ensures regular certificate changes making it more likely that the certificate contains the latest cryptographic standards, and reducing the number of old, neglected certificates that can be stolen and re-used for phishing and drive-by malware attacks. Changing certificates more frequently is even better.
Key Algorithm	Refers to the cryptographic formula used to calculate the certificate keys.
Key Size	Specifies the size of the keys in bits. Four key sizes are allowed: 1024 bits, 2048 bits (this is the default), 3072 bits, and 4096 bits. Larger keys take longer to generate but offer greater security.
Signature Algorithm	Specifies the cryptographic formula used to sign the certificate.
Signature Size	Specifies the size of the signature.
Valid	Specifies certificate dates.
Self Signed	Indicates that the certificate was signed with its own private key.

User Key Store buttons

- **View** displays details for the selected item.
- **New** creates a new device record in the database.

- **Cert Request** opens a **Certificate Request** window, which is used to create a Certificate Signing Request (CSR).
- **Delete** removes the selected record from the database.
- **Import** adds an imported item to the database.
- **Export** saves a copy of the selected record to the hard disk.

  For certificates, the file extension is .pem.
- **Reset** deletes all certificates in the **User Key Store** and creates a new default certificate. It does not matter which certificate is selected when you click **Reset**

  CAUTION:

  Do not reset without considering the consequences. The **Reset** button facilitates creating a new key pair (private and public keys) for the entity, but may disable connections if valid certificates are already in use. Export all certificates before you reset.

Trust Store tabs

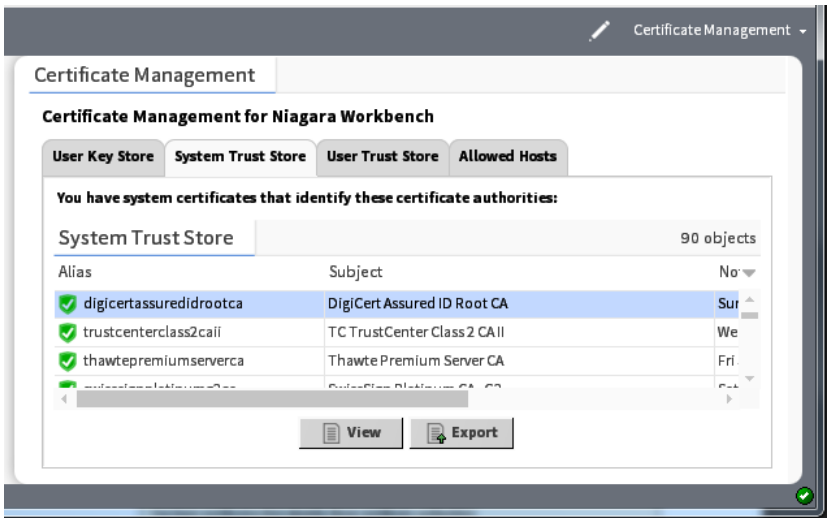
The **Trust Stores** contain signed and trusted root certificates with their public keys. These stores contain no private keys. A **Trust Store** supports the client side of the relationship by using its root CA certificates to verify the signatures of the certificates it receives from each server. If a client cannot validate a server certificate's signature, an error message allows you to approve or reject a security exemption (on the **Allowed Hosts** tab).

The **System Trust Stores** contain installed signed certificates by trusted entities (CA authorities) recognized by the Java Runtime Engine (JRE) of the currently opened platform. A **User Trust Store** contains installed signed certificates by trusted entities that you have imported (your own certificates).

Only certificates with public keys are stored in the **Trust Stores**. The majority of certificates in the **System Trust Store** come from the JRE. You add your own certificates to a **User Trust Store** by importing them.

Feel free to pass out such root certificates to your team; share them with your customers; make sure that any client that needs to connect to one of your servers has the server's root certificate in its client **Trust Store**.

Figure 28 System Trust StoreExample of a Trust Store



Trust Store columns

Column	Description
Alias	Provides a short name used to distinguish certificates from one another in the **Key Store**. This property is required. It may identify the type of certificate (root, intermediate, server), location or function. This name does not have to match when comparing the server certificate with the CA certificate in the client's Trust Store.
Issued By	Identifies the entity that signed the certificate.
Subject	Specifies the Distinguished Name, the name of the company that owns the certificate.
Not Before	Specifies the date before which the certificate is not valid. This date on a server certificate should not be earlier than the **Not Before** date on the CA certificate used to sign it.
Not After	Specifies the expiration date for the certificate. This date on a server certificate should not be later than the **Not After** date on the CA certificate used to sign it. A period no longer than a year ensures regular certificate changes making it more likely that the certificate contains the latest cryptographic standards, and reducing the number of old, neglected certificates that can be stolen and re-used for phishing and drive-by malware attacks. Changing certificates more frequently is even better.
Key Algorithm	Refers to the cryptographic formula used to calculate the certificate keys.
Key Size	Specifies the size of the keys in bits. Four key sizes are allowed: 1024 bits, 2048 bits (this is the default), 3072 bits, and 4096 bits. Larger keys take longer to generate but offer greater security.
Signature Algorithm	Specifies the cryptographic formula used to sign the certificate.
Signature Size	Specifies the size of the signature.
Valid	Specifies certificate dates.
Self Signed	Indicates that the certificate was signed with its own private key.

Trust Store buttons

The **Delete** and **Import** buttons are available only in a **User Trust Store**.

User Key Store buttons

- **View** displays details for the selected item.
- **Delete** removes the selected record from the database.
- **Import** adds an imported item to the database.
- **Export** saves a copy of the selected record to the hard disk.

  For certificates, the file extension is .pem.

Allowed Hosts tab

The **Allowed Hosts** tab contains security exemptions for the currently open platform. These are the certificates (signed or self-signed) received by a client from a server (host) that could not be validated against a root CA certificate in a client's **Trust Store**. Whether you approve or reject the certificate, the system lists it in the **Allowed Hosts** list.

To be authentic, a root CA certificate in the client's **System** or **User Trust Store** must be able to validate the server certificate's signature, and the `Subject` of the root CA certificate must be the same as the `Issuer` of the server certificate.

Allowing exemptions makes it possible for a human operator to override the lack of trust between a server and client when the human user knows the server can be trusted.

If this is a Workbench-to–station connection, the system prompts you to approve the host exemption. Workbench challenges server identity at connection time for unapproved hosts and, unless specific permission is

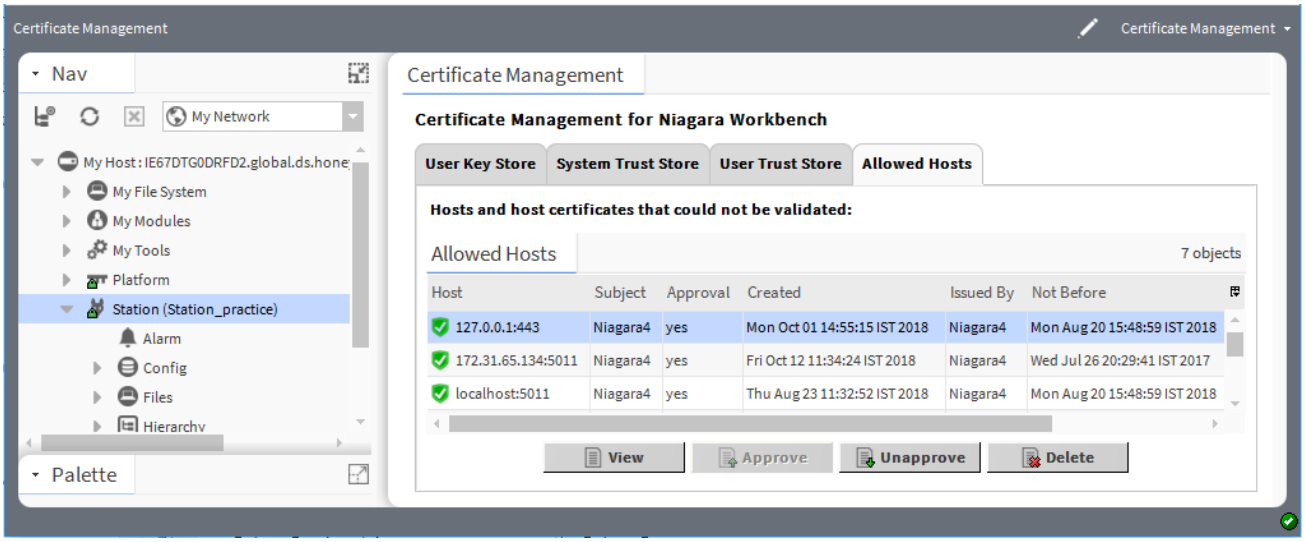
granted, prohibits communication. Once permission is granted, future communication occurs automatically (you still have to log in). Both approved and unapproved hosts remain in this list until deleted.

If this is a station–to–station connection, and there is a problem with the certificates, the connection fails silently. There is no prompt to approve the host exemption. However, the last failure cause in the station reports the problem (expand the station **ClientConnection** under **NiagaraNetwork**).

The approved host exemption in the **Allowed Hosts** list is only valid when a client connects to the server using the IP address or domain name that was used when the system originally created the exemption. If you use a different IP address or domain name to connect to the server, you must approve an updated exemption. The same is true if a new self-signed certificate is generated on the host.

If you continue to use an approved self-signed certificate (rather than implement signed certificates, which are more secure), and the self-signed certificate's public key changes, the system negates the certificate, the green shield icon changes to a yellow shield icon with an exclamation mark (), and the system returns an error. To approve this change, view the exemption (right-click the certificate row on the **Allowed Hosts** tab and click **View**) then approve the certificate by clicking **Accept**.

Figure 29 Example of an Allowed Hosts list



To open this view using Workbench, click **Tools→Certificate Management** and click the **Allowed Hosts** tab.

Allowed Hosts columns

Column	Description
Host	Reports the server, usually an IP address.
Subject	Specifies the Distinguished Name, the name of the company that owns the certificate.
Approval	Reports the servers within the network to which the a client may connect. If approval is `no`, the system does not allow the client to connect.
Created	Identifies the date the record was created.
Issued By	Identifies the entity that signed the certificate.
Not Before	Specifies the date before which the certificate is not valid. This date on a server certificate should not be earlier than the **Not Before** date on the CA certificate used to sign it.
Not After	Specifies the expiration date for the certificate. This date on a server certificate should not be later than the **Not After** date on the CA certificate used to sign it. A period no longer than a year ensures regular certificate changes making it more likely that the certificate contains the latest cryptographic standards, and reducing the number

Column	Description
	of old, neglected certificates that can be stolen and re-used for phishing and drive-by malware attacks. Changing certificates more frequently is even better.
Key Algorithm	Refers to the cryptographic formula used to calculate the certificate keys.
Key Size	Specifies the size of the keys in bits. Four key sizes are allowed: 1024 bits, 2048 bits (this is the default), 3072 bits, and 4096 bits. Larger keys take longer to generate but offer greater security.
Signature Algorithm	Specifies the cryptographic formula used to sign the certificate.
Signature Size	Specifies the size of the signature.
Valid	Specifies certificate dates.

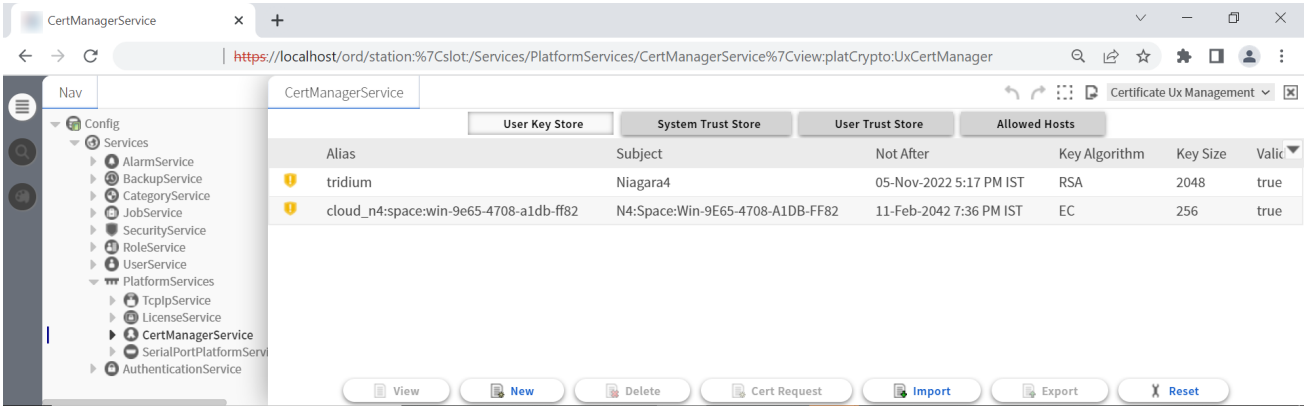
Allowed Hosts buttons

- **View** displays details for the selected item.
- **Approve** designates the server as an allowed host.
- **Unapprove** prohibits aconnection to this server host. The system terminates any attempted communication.

HTML–5 Certificate Ux Mangement View

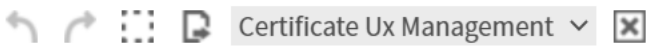
In Niagara 4.13 and later, there is added browser support for the Niagara Network Device Manager View. The HTML 5 version of this view is a web-browser-based implementation and it provides the same functions as the Workbench view.

Figure 30 Certificate Ux Mangement view



You can access it from a browser of your desktop or mobile device. To access this view from the browser, expand **Config→Services→PlatformServices** and double-click **CertManagerService** or right-click **CertManagerService** and click **Views→Certificate Ux Mangement**.

Figure 31 Toolbar of Certificate Ux Mangement



This toolbar offers familiar Workbench. The following tools, however, are unique to the HTML5 view.

Tools	Description
Undo	Reverses the previous command.
Redo	Restores a command action after the Undo command has removed it.

Tools	Description
Enable selection Mode	Enables you to individually select multiple points without holding down the ctrl key.
Export	Exports the current view or object.

Buttons

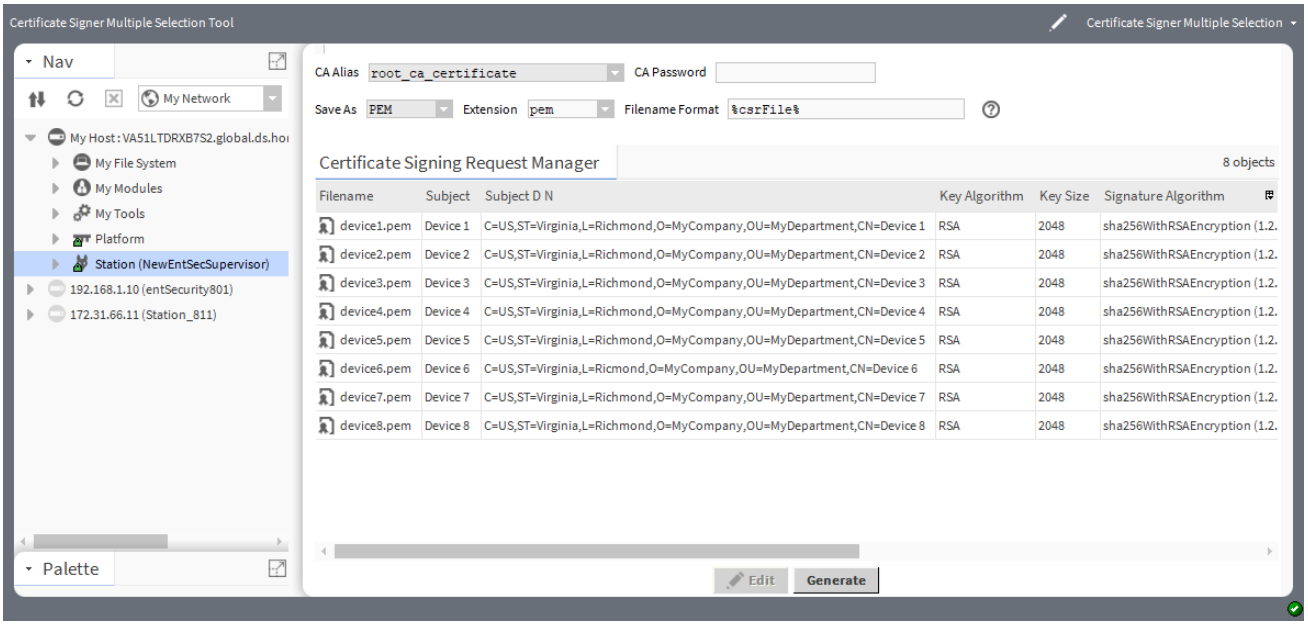
Buttons	Description
View	Allows you to view the information of the selected certificate.
New	Creates a new self-signed certificate.
Delete	Deletes the selected certificate from the Keystore.
Cert Request	Generates a certificate request and to exports it.
Import	Adds a new certificate in the keystore.
Export	Exports a selected certificate to a new directory.
Reset	Resets the Keystore and generates a new self-signed certificate.

Certificate Signer Multiple Selection (platCrypto-CertificateSignerMultipleSelectionTool)

This tool significantly reduces the time spent signing multiple certificate signing requests. The tool also includes a converter to generate browser-compatible certificate formats.

Using this tool you can simultaneously bulk-sign certificates with the same root CA certificate. This tool's **Save As** property generates a signed certificate in PKCS7 format. This format is essential for ensuring secure browser access.

Figure 32 Certificate signer multiple selection tool properties



To open this view, click **Tools→Certificate Signer Multiple Selection Tool**, in the **Select Certificate Signing Request(s)** window, select the certificates to sign and click **Open**.

Column	Description
CA Alias	Identifies the root CA certificate with which to sign the selected certificates.
CA Password	Reports the certificates private key password.
Save As	Reports one of these formats: `PEM` (Privacy-Enhanced Mail) `PKCS7` is a standard syntax for storing signed and/or encrypted data. PKCS stands for Public-Key Cryptography Standards.
Extension	Reports the extension for the certificate file. Each format has a set of extensions. For PEM the possible extensions are: `cer`, `crt`, `pem` or `key`. For PKCS7 the possible extensions are `p7b` or `p7c`.
Filename Format	Reports the format string used to create the certificate filename(s) generated from certificate signing requests. This normal text has embedded scripts surrounded by % character pairs. Saving the files appends an appropriate filename extension (for example, *.pem).

File format scripts

Valid scripts for filename formatting are these:

- `%csrFile%`, filename of the original csr selected.
- `%subject%`, subject attribute of the csr.
- `%altSubject%`, the first subject alternative name of the csr.

Other options include:

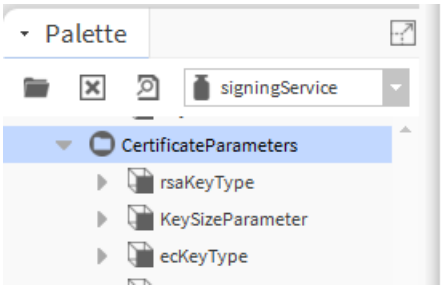
- `%csrFile%_%subject%`
- `%csrFile%_%altSubject%`
- `%csrFile%.pem`

Additional information can be found in the class documentation for BFormat.

Certificate Parameter (platCrypto-CertificateParameter)

You can add **Certificate Parameters** to a signing profile to define default values for any fields that may be unspecified in the incoming CSR. A validation error will be thrown if the incoming CSR defined a different value, causing the signing request to be rejected.

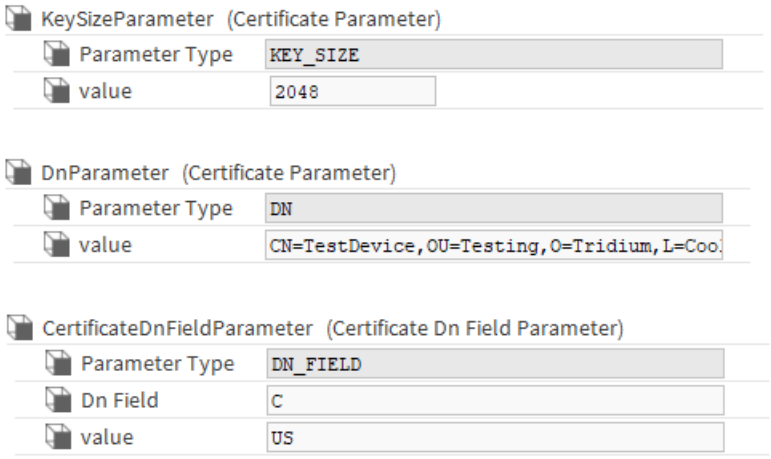
To use the **Certificate Parameter** components, drag them from those available in the **`signingService`** palette beneath a **Signing Profile** underneath the **Signing Service** in a running station.



Example parameters available in the **`signingService`** palette include:

- Key Type – RSA/EC

- Key Size – specify an exact required key size
- Minimum Key Size – specify a minimum required key size
- EC Curve – specify the curve name for an ecliptic curve key
- DN – define the full/partial distinguished name attribute values
- DN field parameter – define a single distinguished name attribute value

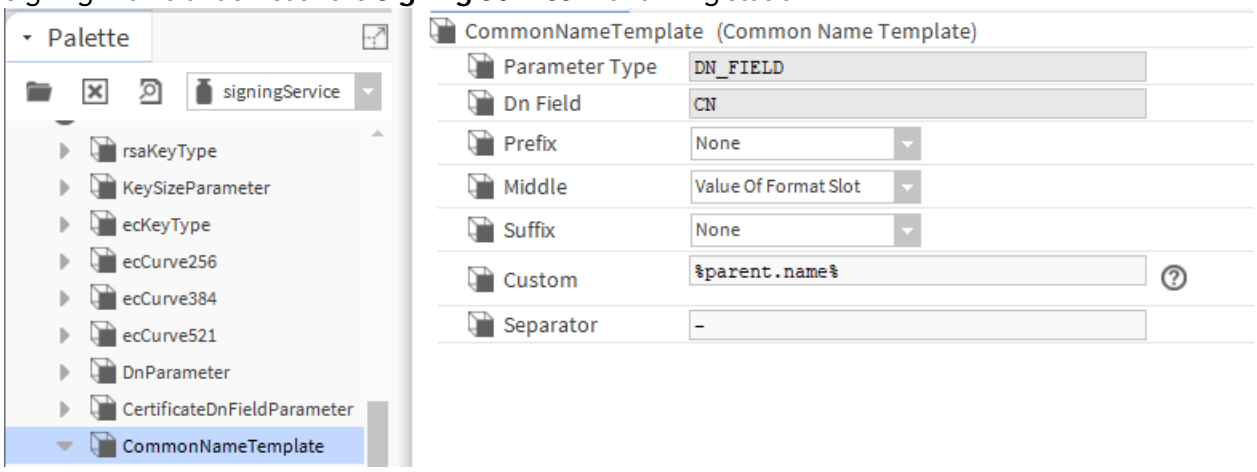


Property	Value	Description
Parameter Type	read-only	Displays the name of the certificate parameter.
Value	string	Defines the value of the certificate parameter.
Dn Field	string	Defines the name of the DN field to populate. Specific to CertificateDnField type parameter.

Common Name Template (platCrypto-CommonNameTemplate)

The **Common Name Template** component is a type of Certificate Parameter that will generate a dynamic common name for the signed certificate based upon configurable settings. The main benefit of this is the ability to generate a potentially unique common name for each component.

To use the **Common Name Template** component, drag it from the **signingService** palette beneath a Signing Profile underneath the **Signing Service** in a running station.



The generated common name will by default resolve the Custom BFormat against the requesting component resulting in a common name matching the component's parent name.

This can be customized by modifying the **Custom** value, and by selecting values to prepend or append, or both in the respective **Prefix** and **Suffix** drop-downs. The **Separator** value is used to join the different parts of the common name. Available values for the three different parts of the name are:

- None — no value and no separator used
- Value of Format Slot — the value of the BFormat resolved against the requesting component
- Station Name
- Device Name — or an empty string if the component is not beneath a device
- Network Name — or an empty string if the component is not beneath a network
- Host Name
- Uid — a 16–character unique string

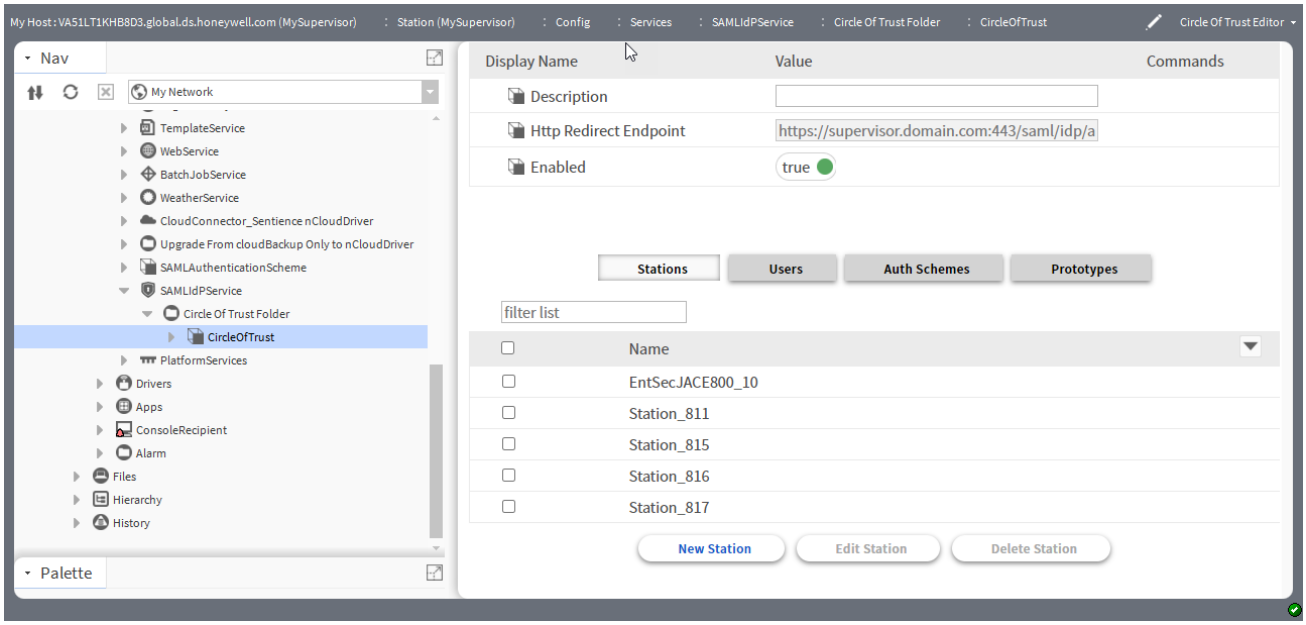
Property	Value	Description
Parameter Type	read-only	Displays the name of the certificate parameter.
Dn Field	read-only	CN
Prefix	drop-down	Selects the type of value to prepend to the common name.
Middle	drop-down	Selects the type of value forming the middle of the common name.
Suffix	drop-down	Selects the type of value to append to the common name.
Custom	BFormat string	Format used to resolve against the requesting component.
Separator	string	Specifies the value to join the different parts of the common name string.

Circle Of Trust Editor (saml-CircleOfTrustEditor)

This is the main view for the CircleOfTrust component. It is used to configure one or more Circles of Trust in the SAMLIdPService.

Before you can access this editor you (or someone else) must have opened the **saml** palette, and copied or dragged a **SAMLIdPService** to the **Services** folder, and a **CircleOfTrust** to the **Circle Of Trust Folder**.

Figure 33 Circle Of Trust Editor view



To access this view expand**Config→Services→SAMLIdService→Circle Of Trust Folder** and double-click **CircleOfTrust**.

Property	Value	Description
Description	text	
Http Redirect Endpoint	domain and port	
Enabled	`true` (default) or `false`	Enables (`true`) and disables (`false`) this editor.

Buttons

The following buttons are located in the upper half of the view:

- **Stations** selects subordinate stations from those found in the Supervisor's **NiagaraNetwork**, to include in the circle of trust.
- **Users** selects the users in the in the station.
- **Auth Schemes** selects authentication schemes that may be used when logging in to the remote station.
- **Prototypes** selects user prototypes in the Supervisor's **UserService** that may be used when logging in to the remote station.

The following buttons are located in the lower half of the Stations view:

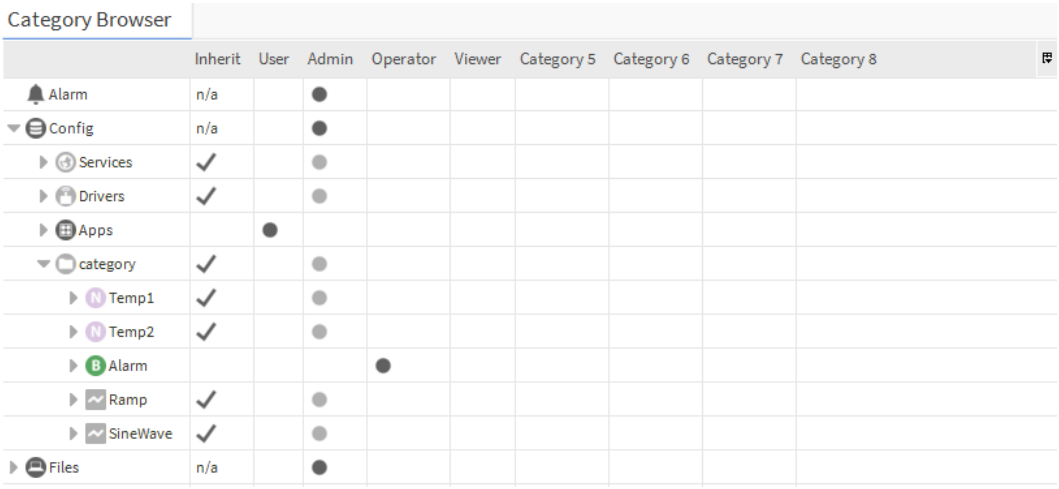
- **Add Station** adds non-**NiagaraNetwork** stations to the circle of trust
- **Edit Station** edits a station in the circle of trust
- **Delete Station** removes a station from the circle of trust

Category Browser (wbutil-CategoryBrowser)

This view is the default view of the station's **`CategoryService`**, and typically where you spend most of your time assigning categories to components after initially creating the categories.

NOTE: When an admin user (user with Admin write privilege on the **CategoryService** and on a particular station component being adjusted) makes a category mask adjustment, the user must also have at least the Operator write privilege on the category being adjusted in the category mask for the station object. This includes changes to check marks in the "Inherit" column—the user must have at least Operator write access to any altered categories applied from the Inherit change.

Figure 34 Category Browser view



To access this view, expand **Config→Services** and double-click **CategoryService**.

Columns

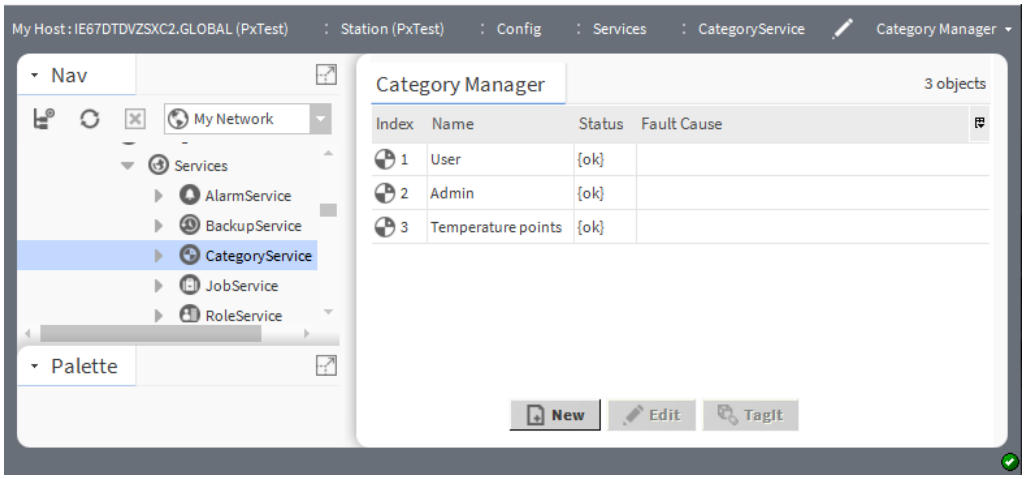
Column	Description
Inherit	A check mark indicates that the object inherits the category from its parent in the table.
User	Indicates the category. All system objects except for those listed as assigned to `Admin` are assigned to this category.
Admin	These objects default to the `Admin` category: • The configuration services: **`UserService`**, **`CategoryService`**, and **`ProgramService`** • All files (the entire file space)
Categories 3–8	A bold bullet indicates that the object is assigned to the category. A grayed out bullet indicates inheritance. Blank indicates that the category has not been assigned.

Category Manager (wbutil-CategoryManager)

This view of the CategoryService allows you to create, enable and delete the groups that the security model uses to control access to the objects in a station. Once you create categories, you use the **Category Browser** view to centrally assign system objects to categories. Or, at the individual component level, you use a component's **Category Sheet** view to assign the component to one or more categories.

You can assign an object to many categories at the same time. Each object stores its own categories.

Figure 35 Category Manager, Temperature Points as Category



To access this view, expand **Config→Services** and right-click **CategoryService** and click **Views→Category Manager**.

Columns

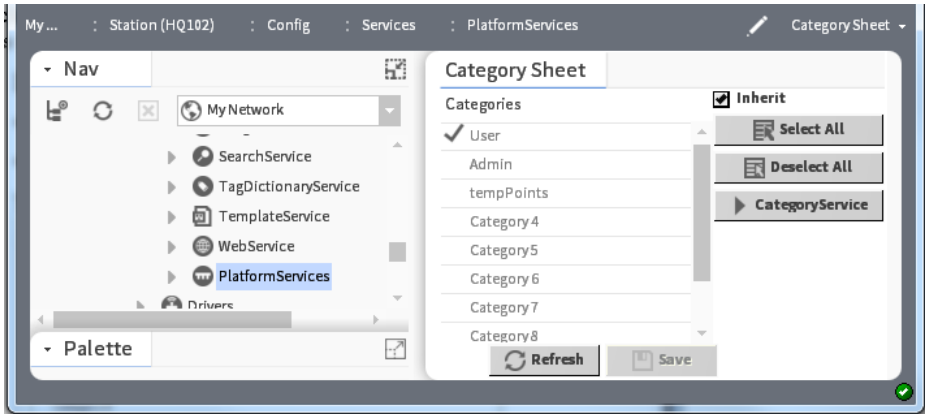
Column	Description
Index	Reports a unique number for the category as it is known to the station.
Name	Displays descriptive text that reflects the purpose of the entity or logical grouping.
Status	Reports the current condition of the entity as of the last refresh: {alarm}, {disabled}, {down}, {fault}, {ok}, {stale}, {unackedAlarm}
Fault Cause	Indicates the reason for a fault.

wbutil-CategorySheet

This view assigns a component to one or more categories (or configures it to inherit categories from its parent. Every component has a **Category Sheet** view.

NOTE: When an admin user (user with Admin write privilege on the **CategoryService** and on a particular station component being adjusted) makes a category mask adjustment, the user must also have at least the Operator write privilege on the category being adjusted in the category mask for the station object. This includes changes to check marks in the "Inherit" column—the user must have at least Operator write access to any altered categories applied from the Inherit change.

Figure 36 Category Sheet



Column/option	Description
Categories	Provides one table row for each category name.
Inherit	A check mark indicates that the component belongs to the same categories as its parent component. No check mark allows you to make explicit category assignments for this component.

Buttons

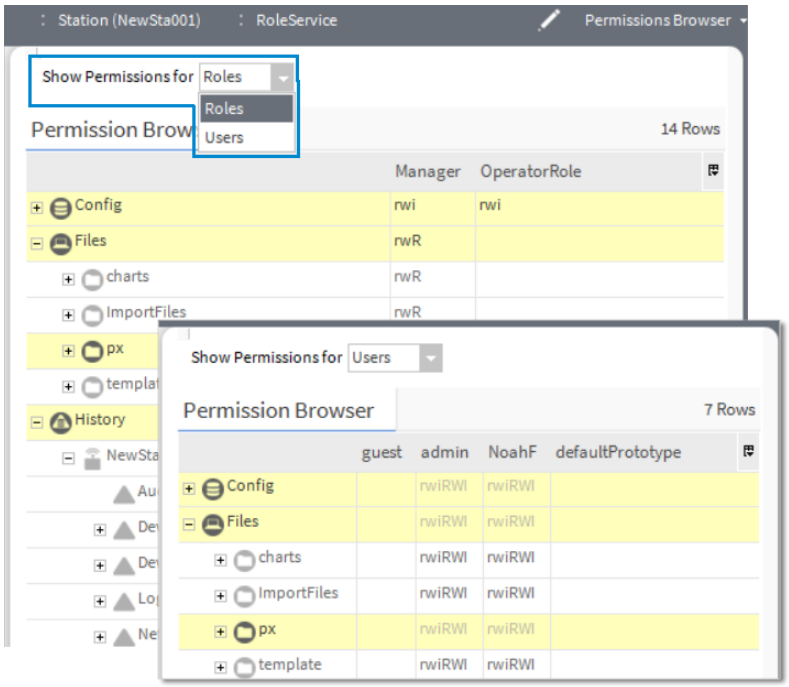
- **Select All** is effective if `Inherit` is cleared. Clicking this button assigns this component to all categories in this station.
- **Deselect All** is effective if `Inherit` is cleared. Clicking this button removes this component from all categories.
- **Category Service** opens the **Category Browser**.
- **Refresh** re-displays the **Category Sheet**.
- **Save** records the changes made.

wbutil-PermissionsBrowser

This view allows you to quickly review the objects that someone, who has been assigned a given role may access. You access this view by right-clicking **`RoleService`** in the Nav tree and clicking **Views→Permissions Browser**.

NOTE: In Niagara, there is added support for the **UserService** in the **Permissions Browser** view. Use the **Show Permissions for** drop-down list to switch between permissions for Users, and for Roles. When viewing permissions for **Users**, the view displays a separate column for each user as well as any prototype.

Figure 37 Permissions Browser view showing permissions for roles and users



Columns represent roles or users, and rows identify the objects in the station, with each table cell showing user permissions.

- Yellow rows are objects explicitly assigned with permissions.

- Dimmed rows represent objects that inherit their permissions from their parent object.

Double-click a cell to bring up the permissions window for that role or user depending on which option is selected in the **Show Permissions for** drop-down list. This allows you to globally change permission levels for any category in the station.

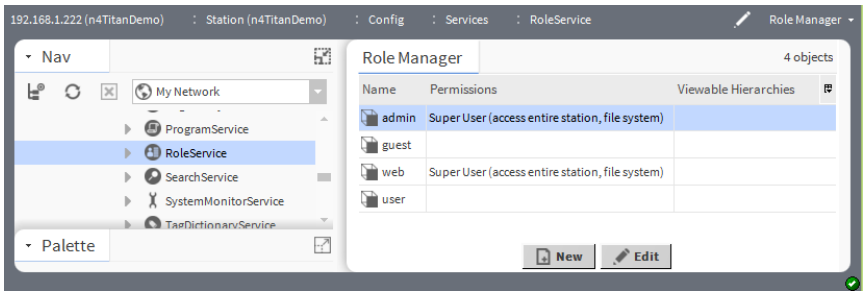
Column	Description
First column	Each Nav tree node occupies a row in the table. This expandable tree lets you navigate to objects of interest to review current permissions.
Admin	Reports the rights assigned to the `admin` role. As this is a super user, `admin` has rights to read, write and invoke an action for all objects. permissionsR = readW = writeI = invoke action`admin` level permissions appear in upper case.
User	Reports the rights assigned to the `user` role. The default is no rights assigned. permissionsr = readw = writei = invoke action`operator` permissions appear in lower case.

Role Manager (wbutil-RoleManager)

This manager creates, edits and deletes roles. It is the default view of the **RoleService** and is located in the station's **Services** container.

The system creates the `admin` role by default and grants it super user permissions. The `admin` role does not appear in the **Role Manager** view and you may not delete it.

Figure 38 Role Manager view



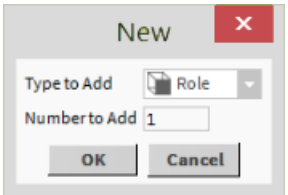
To access this view, expand **Config→Services** and double-click **RoleService**.

Columns

Column	Description
Name	Identifies the role to be assigned to one or more users. Role names are case sensitive.
Permissions	Associates a name with a specific set of permissions.
Viewable Hierarchies	Identifies the hierarchies this user may view.
Type	Identifies the type of entity being created.
Number to add	Allows you to create many rows at once in the **Role Manager** view's table.

New role window

Figure 39 New role window

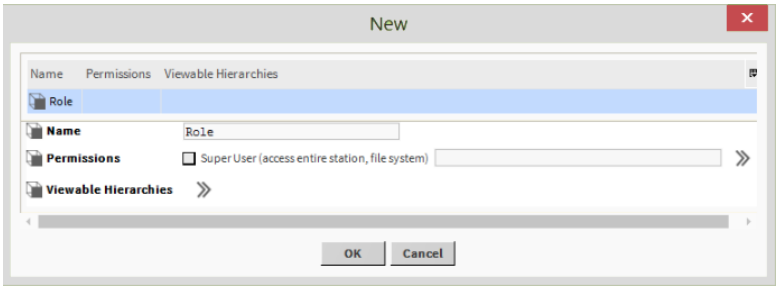


Property	Value	Description
Type	`Role` (default)	Identifies the type of entity being created.
Number to add	number	Allows you to create many rows at once in the **Role Manager** view's table.

New role properties

CAUTION: There are risks involved in giving any user broad permissions on the Role Service. For example, giving a user **admin write** permissions on the Role Service allows that user to create, edit, rename or delete any role. Best practices recommend that such permissions on the Role Service be limited only to appropriately authorized users.

Figure 40 New role properties



Property	Value	Description
Name	text	Provides a name for the role.
Permissions	check box and text field	Grants permissions to roles.

Windows

Windows create and edit database records or collect information when accessing a component. You access them by dragging a component from a palette into a station or by clicking a button.

Windows do not support **On View (F1)** and **Guide on Target** help. To learn about the information each contains, search the help system for key words.

Certificate Signing window

This window identifies the root CA certificate for signing other certificates and type of certificate to create.

Figure 41 Certificate signing properties

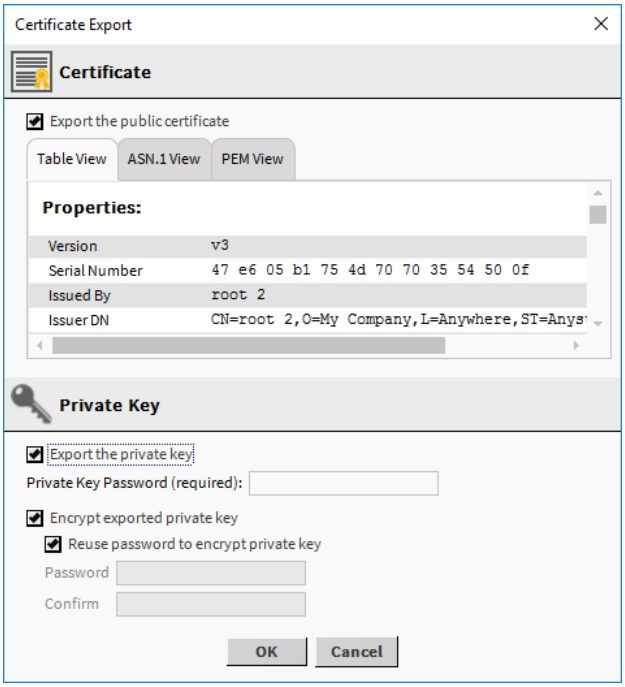
To access this window, click **Tools Certificate Signer Tool**.

Property	Value	Description
Select a certificte signing request to sign:	chooser	Locates the certificate on the local hosts system.
Not Before:	date and time	Sets the date and time when the certificate takes effect.
Not After:	date and time	Sets the date and time when the certificate expires.
CA Alias:	drop-down list	Identifies the root CA certificate with which to sign the selected certificates.
CA Password:	text	Applies the certificates private key password.
Save As	drop-down list	Selects among these formats: `PEM` (Privacy-Enhanced Mail) `PKCS7` is a standard syntax for storing signed and/or encrypted data. PKCS stands for Public-Key Cryptography Standards.

Certificate Export windows

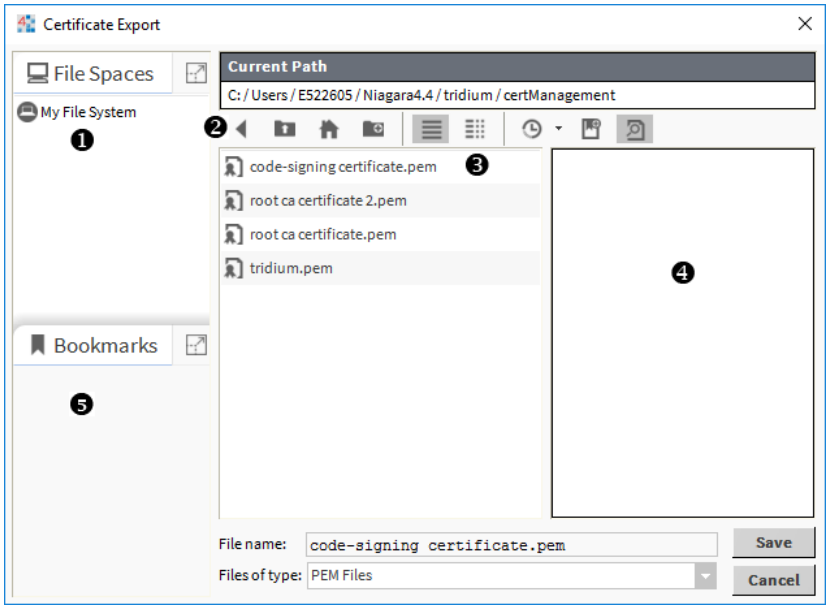
These windows save a copy of the selected certificate to a folder on your hard drive or thumb drive.

Certificate Export chooser



Property	Value	Description
Export the public certificate	check box, defaults to selected	Indicates that the system will export the certificate with only its public key.
Table, ASN.1 View and PEM View	tabs	Shows the contents of the certificate in terms of properties, ASN (Abstract Syntax Notation), and PEM (Privacy Enhanced Mail).
Export the private key	check box, defaults to de-selected	Instructs the system to export the private key with the public key. You would select this option only if you are backing up a root CA or intermediate certificate to a second secure location.
Private Key Password	text	Supplies the password created when the certificate was created.
Encrypt exported private key	check box, defaults to selected	Causes the system to encode the private key for enhanced security.
Reuse password to encrypt private key	check box, defaults to selected	Indicates that no additional password is required to encrypt the private key. Deselecting this option allows you to assign a second password to protect the private key.
Password and Confirm	text	Create and confirm the second password.

Certificate Export, file chooser

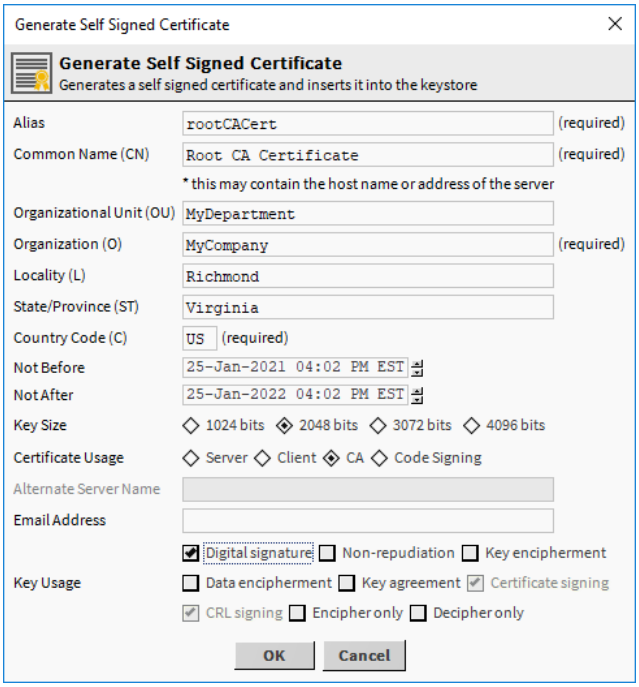


❶	Displays the available file spaces. The contraction ( ) and expansion ( ) icons resize the selected pane.
❷	The control icons determine the contents of the file view pane: returns to a previous folder. displays files in the folder above the current folder. displays files in the root of the user home. creates a new folder. configures the chooser to display the file name only. configures the chooser to display the details for each file. provides a drop-down list of available file paths. creates an entry in the Bookmarks pane. turns the display of the details pane on and off.
❸	The file view pane.
❹	This pane displays the details for the selected file.
❺	Bookmarks pane.

Generate Self-Signed Certificate window

This window defines the important information required to create a certificate. You use this window to create your own certificates along with a key pair (public and private).

Figure 42 Default view of the Generate Self-Signed Certificate window



This window opens when you click **New** at the bottom of the **User Key Store** tab.

A self-signed certificate provides data encryption only. Since it is not signed by a CA (Certificate Authority) it cannot verify server identify. Generating a self-signed certificate should be a temporary measure until a signed certificate is installed in the browser's and station's trust stores. After installing the signed certificate you should delete any self-signed certificates.

There is a limit of 64 characters for each of the following properties. Although blank properties are permitted, it is recommended to correctly fill in all properties, as not doing so may generate errors, or cause third-party CAs to reject your certificate. Spaces and periods are allowed. Enter full legal names.

Name	Value	Description
Alias	text	Provides a short name used to distinguish certificates from one another in the **Key Store**. This property is required. It may identify the type of certificate (root, intermediate, server), location or function. This name does not have to match when comparing the server certificate with the CA certificate in the client's Trust Store.
Common Name (CN)	text, required, alphanumeric; do not use "*" or "?" as part of the name	Also known as the Distinguished Name, this field should be the host name. It appears as the `Subject` in the **User Key Store**.
Organizational Unit (OU)	text	The name of a department within the organization or a Doing-Business-As (DBA entry). Frequently, this entry is listed as "IT", "Web Security," "Secure Services Department" or left blank.
Organization (O)	text	The legally registered name of your company or organization. Do not abbreviate this name. This property is required.
Locality (L)	text	The city in which the organization for which you are creating the certificate is located. This is required only for organizations registered at the local level. If you use it, do not abbreviate.

Name	Value	Description
State/Province (ST)	text	The complete name of the state or province in which your organization is located. This property is optional.
Country Code (C)	two-character ISO-format country code.	If you do not know your country's two-character code, check www.countrycode.org. This property is required.
Not Before	date	Specifies the date before which the certificate is not valid. This date on a server certificate should not be earlier than the **Not Before** date on the CA certificate used to sign it.
Not After	date (defaults to one year from the **Not Before** date)	Specifies the expiration date for the certificate. This date on a server certificate should not be later than the **Not After** date on the CA certificate used to sign it. A period no longer than a year ensures regular certificate changes making it more likely that the certificate contains the latest cryptographic standards, and reducing the number of old, neglected certificates that can be stolen and re-used for phishing and drive-by malware attacks. Changing certificates more frequently is even better.
Key Size	number	Specifies the size of the keys in bits. Four key sizes are allowed: 1024 bits, 2048 bits (this is the default), 3072 bits, and 4096 bits. Larger keys take longer to generate but offer greater security.
Certificate Usage:	text	Specifies the purpose of the certificate: server, client or CA certificate. Other certificate management software utilities may allow other usages.
Alternative Server Name	text	This property provides a name other than the Subject (`Common Name`) that the system can use to connect to the server. Like the `Common Name`, the system uses the `Alternative Server Name` to validate the server certificate making it possible to specify both an IP (Internet Protocol) and FQDN (Fully Qualified Domain Name).
Email Address	email address	The contact address for this certificate. It may also be the address to which your signed certificate (.pem file) will be sent.
Key Usage	radio buttons (defaults to `Digital signature`)	Indicates the business scenario that requires authentication, encryption, and digital signing. The public and private keys associated with each certificate may be used to provide these secure features. For a description of each, refer to Key Usage options, page 150

Key Usage options

Select this **Key Usage** option...	When...
`Digital signature`	the key is to be used to validate the authenticity of the server.
`Non-repudiation`	you need to ensure that a transferred message was sent and received by the parties claiming to have sent and received the message. This guarantees that the sender of a message cannot later deny having sent the message and that the recipient cannot deny having received the message.

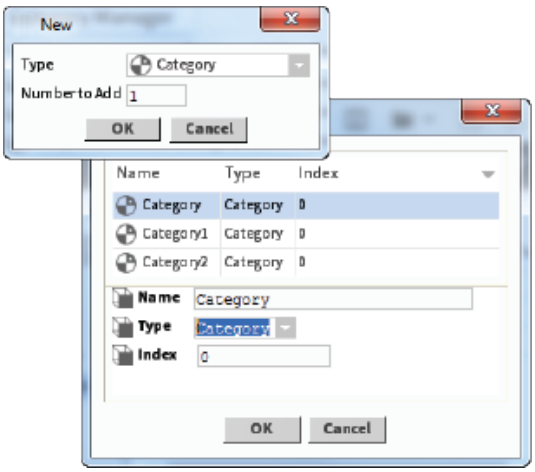
Select this Key Usage option...	When...
Key encipherment	you need a protocol that encrypts keys. An example is S/MIME that encrypts a fast (symmetric) key using the public key from the certificate. The SSL protocol also performs key encipherment.
Data encipherment	your application calls for using the public key to encrypt user data as well as the cryptographic keys.
Key agreement	the sender and receiver of the public key need to derive the key without using encryption. The application can then use the public key to encrypt messages between the sender and receiver. Diffie-Hellman ciphers usually ensure key agreement.
Certificate signing	the subject public key is used to verify a signature on certificates. Only CA certificates may use this extension.
CRL signing	the subject public key is used to verify a signature on revocation information, such as a CRL (Certificate Revocation List-a list of digital certificates that have been rejected by a certificate authority).
Encipher only	key agreement is also enabled. This enables the public key to be used only for enciphering data while performing key agreement.
Decipher only	key agreement is also enabled. This enables the public key to be used only for deciphering data while performing key agreement.

New category windows

This window sets up new categories.

Basic category windows

Figure 43 Category properties



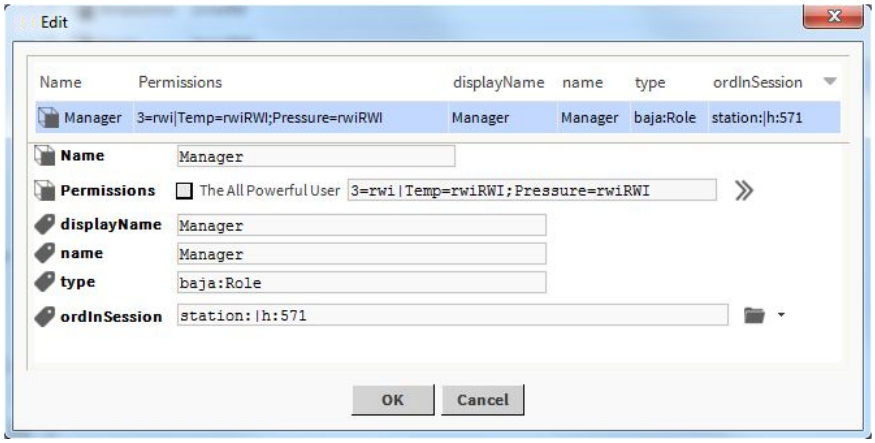
To open these windows, expand **Config→Services** right-click **CategoryService** and click **Views→Category Manager**

Property	Value	Description
Name	text	A name for the given category. This can be any name that describes how you are using the category.
Type	drop-down list	Selects the type of record.
Index	number	Creates a unique number for the category as it is known to the station.

New/Edit roles window

This window creates and edits roles and permissions. You access it from the **Role Manager** view (**`RoleService`**).

Figure 44 Edit role window

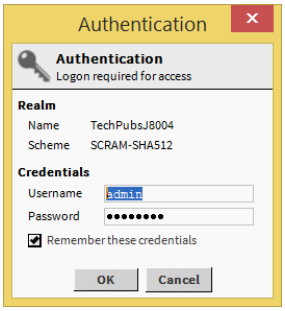


Property	Value	Description
Name	text	Provides descriptive text that reflects the identity of the entity or logical grouping.
Permissions	check box and chooser	Configures the type of access. `All Powerful User` sets up a super user with access to the entire station and file system. To configure individual permissions for a role, click the chevron. The box between the All Powerful User and the chevron displays all categories. Basic categories display by index number, for example: 3=rwi.
tags	text	Provide additional, searchable information about the object. Any tags associated with the object appear at the end of the **Property Sheet**. The tag icon identifies them.

Platform Authentication window

This window opens after a secure platform connection is made. Its purpose is to authenticate the platform user.

Figure 45 Platform authentication window

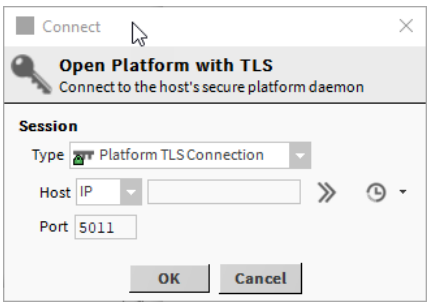


Name	Value	Description
Name	text	The name may include your company network name (when connecting to a Windows-based computer), the IP address of the controller, or host name of the controller.
Scheme	text	This information identifies the authentication scheme used to log on to a platform: HTTP-Basic, HTTP-Digest, or SCRAM-SHA512.
Credentials—Username	text	This is where you enter the platform user name created when the controller was commissioned. Access to this name is through **Platform Administration→User Accounts**.
Credentials—Password	text	This is the password created when the controller was commissioned. Access to this password is through **Platform Administration→User Accounts**.
Remember These Credentials	check box	Select this check box to have the system automatically fill in the user name and password when you log on the next time.

Platform Connect

This window opens when you open a platform (supervisor PC or controller).

Figure 46 Platform connect window



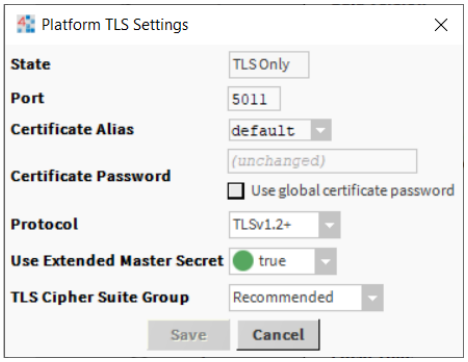
Name	Value	Description
Type	drop-down list (defaults to `Platform TLS Connection`.)	Selects the type of connection, standard or TLS secure.
Host (type)	type drop-down box and chooser (defaults to `IP`	Identifies the platform.

Name	Value	Description
IP address	IP address	Identifies the IP address or URL of the host platform.
Port	number (defaults to 5011)	Identifies the port for secure platform communication.

Platform TLS settings

This window sets up the platformtls (niagarad) properties that provide server authentication and encryption. To access it, right-click **Platform→Views→Platform Administration** and double-click **Change TLS Settings**.

Figure 47 Platform TLS Settings window



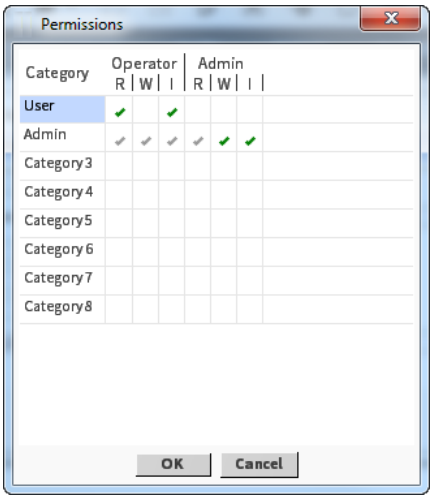
Name	Value	Description
State	TLS	Defaults to TLS only.
Port	number	The port for secure communication. Defaults to 5011
Certificate Alias	text (defaults to the `default` self-signed certificate)	Provides a list of available certificate aliases to choose from. As of Niagara 4.13, the `default` certificate is the self-signed certificate automatically created when you first accessed the platform. **NOTE:** If the `tridium` certificate is already used on the station or the platform runs a pre-Niagara 4.13 version, the `tridium` certificate is used, but it will not serve as a recovery certificate. It cannot be deleted and should be used for recovery purposes. The default certificate is protected by the global certificate password. If other certificates are in the host platform's key store, you can select them from the drop-down list.
Certificate Password	text and check box	As of Niagara 4.13, the certificate is password-protected by a unique password or the global certificate password. Prompts the user to provide the user-defined password or the global certificate password associated with the certificate.
Protocol	TLSv1.0+ — Includes TLS versions 1.0, 1.1, and 1.2, providing the most flexibility; TLSv1.1 + — Only TLS versions 1.1 or 1.2 are	Defines the minimum TLS (Transport Layer Security) protocol version that the platform daemon's secure server accepts to negotiate with a client for a secure platform connection. During the handshake, the server and client agree on which protocol to use.

Name	Value	Description
	accepted; TLSv1.2 + — (default) Only TLS versions 1.2 or 1.3 are accepted; TLSv1.3 — Only TLS version 1.3 is accepted.	
Use Extended Master Secret	`true` (default) or `false`	Turns on and off the "Extended Master Secret" on a server. When turned off (set to `false`) and the platform restarts, the CPU usage does not change significantly when connecting to the **Platform Administration** view from a FIPS-mode Workbench.
TLS Cipher Suite Group	drop-down list, `recommended` (default) or `supported`	Controls which cipher suites can be used during TLS negotiation. The default is more secure than the other option (`Supported`) and should be used unless it causes compatibility issues with the client.

Permissions map

This window associates permissions with categories and permission levels. To access it, add or edit a role and click the chevron next to the **`Permissions`** property.

Figure 48 Permissions window

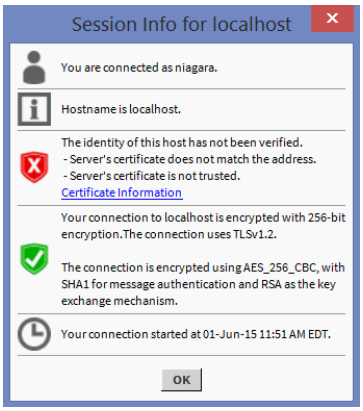


Column	Description
Category	`User` and `Admin` are default categories created by the **New Station** wizard. Each category occupies a single row in the Permissions map.
Operator	Provides a way to set access rights for components that are configured with the `operator` permission level. Permission level is set by the `Operator` config flag on the component's **Property Sheet**.
Admin	Indicates that the object may be read, written or an action invoked by only system users that have been granted `admin` rights. The Admin permission level is set by turning off the `Operator` config flag on the component's **Property Sheet**.

Session Info window

The **Session Info** window reports the security status of the current communication session. You view this window by right-clicking the Session Info icon ( ).

Figure 49 Session info when using a self-signed certificate



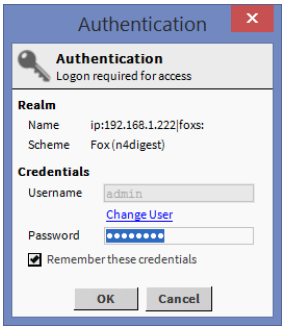
Section	Description
	Identifies the user account that is logged in to the station.
	Identifies the host name of the server.
 or 	Reports on the attempt to verify the authenticity of the server. A green shield indicates that a root CA certificate in the client's Trust Store verified the authenticity of the server certificate. A red shield indicates that the client system found no matching root CA certificate in a **Trust Store** with which to verify the server certificate. Clicking `Certificate Information` displays the certificate.

Section	Description
 or 	Describes the Foxs session encryption strength. A green shield indicates that the transmission is encrypted. A red shield indicates that the transmission is not encrypted.
	Logs when the Foxs session began.

Station Authentication window

This window opens after a secure station connection is made. Its purpose is to authenticate the station user.

Figure 50 Station Authentication window

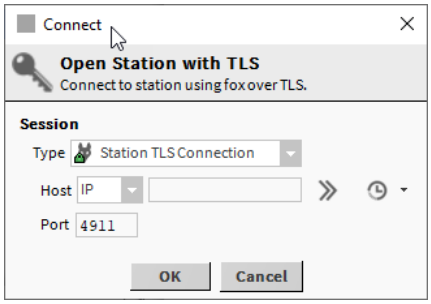


Name	Value	Description
Name	text	The station name, which, for a controller, is usually its IP address.
Scheme	text	This information identifies the authentication scheme used to log on to this station. The scheme used depends upon the user. Configuration is through the UserService.
Credentials—Username	text	This is where you enter your station user name.
Credentials—Password	text	This is your platform password, which is the same as your personal station password.
Remember These Credentials	check box	Select this check box to have the system automatically fill in the user name and password when you log on the next time.

Station Connect window

This window opens when you open a station.

Figure 51 Station Connect window



Name	Value	Description
Type	drop-down list	Defaults to `Station TLS Connection`.
Host (type)	drop-down list	Defaults to `IP`.
IP address	text	This is where you enter the IP address or URL of the host platform.
Port	number	The port for secure station (foxs) communication. Defaults to 4911.

Glossary

Certificate	A PKI (Public Key Certificate) or digital certificate is an electronic document used to prove ownership of a public key. The certificate includes information about the key, the identity of its owner, and the digital signature of an entity that verified the validity of the certificate's contents. If the signature is valid, and the client can trust the signer, the client can be confident that it can use the public key contained in the certificate to communicate with the server.
Certificate Authority (CA)	An entity that issues the digital certificates used to certify the ownership of a public key by the named subject of the certificate. This allows system users to rely upon signatures or assertions made by the private key that corresponds to the certified public key. In this relationship model, the party that relies on the certificate trusts that the subject (owner) of the certificate is authentic because of the relationship of both parties to the CA.
chain of trust	Also called a web of trust, certification path, or trusted certificate tree is an approach to server verification that uses a self-signed certificate owned by a CA (Certificate Authority) to begin the authorized relationships. The private key of this root CA certificate signs a company's server certificate(s). Intermediate certificates may be used to further isolate relationships, such as by geographic location or corporate division.
Distinguished Name	A Distinguished Name (DN) is a string that uniquely identifies an entry in the LDAP directory. It's comparable to a path in a file system. The CN portion of the DN is comparable to a file name. As it applies to SAML attribute mapping, an Identity Provider may return a DN (e.g. CN=userGroup, OU=Users, DC=domain, DC=net) for the prototypeName attribute. More details on SAML authentication and attribute mapping are available in the "Single Sign On" section of the *Niagara Station Security Guide*.
key	A digital key is a very large, difficult-to-predict number surrounded by a certificate. Keys serve these purposes: 1) The public key of a root CA certificate in a client's **System** or **User Trust Store** verifies the authenticity of each server. 2) The private key of a trusted root CA certificate may sign other certificates. 3) After server authentication, matching public and private keys encrypt and decrypt data transmission.
NEQL	Niagara Entity Query Language provides a simple mechanism for querying objects with tags. Whereas BQL supports the tree semantics and pathing of Workbench component space (for example parent.parent) and BFormat operations, NEQL queries only for tags using the Niagara 4 tagable and entity APIs.
object	An object is the base class required for all system entities that conform to the baja model. Objects group information used to construct a model that includes building devices, virtual devices, individual points, users, system features and services. Objects appear in the Nav tree as files, modules, installers, administrators, copiers, drivers and apps. Metadata associated with objects, including categories, roles (permissions), and hierarchies, provide access control and configuration options to manage automated buildings efficiently.
pem file	A Privacy Enhanced Mail Certificate (PEM) is a certificate file for authenticating a secure server. It may contain: a private key and various certificates, such as a certificate authority (CA) certificate, primary certificate, and intermediate certificate. These certificates make up the chain of trust.

permission	The right to access a component slot, folder, file or history. Three rights may be granted: the right to read information provided by the object, the right to write (change) the object, and the right to invoke an action on the object. Rights are granted using the **Role Manager**'s permissions map. Permissions are applied to users by assigning each user a role. A super user is automatically granted every permission in every category for every object.
permission level	A slot config flag that indicates who is allowed to access the slot. When unchecked (the default) at least `admin`-Read (R) permission is required (as defined in the **Role Manager**'s permissions map). When checked a user with a minimum of `operator`-read permission (r) may access the slot.
role	A logical grouping that is assigned as metadata to system users (human and machine) for security purposes. For example, roles may be used to group users as administrators (`admin`), regular users (user), and operators (`operator`). Roles speed the management of permissions for a large number of users. The permissions of a group of users that share the same role can be updated by changing the role's permissions instead of updating each user's permissions individually. You manage roles using the **`RoleService`**.
signature	A digital signature combines a unique hash that is created using a cryptographic algorithm (such as SHA-512) with a public key. This is done by using a matching private key to encrypt the hash. The resulting signature is unique to both the certificate and the user. Finally, the signing process embeds the digital signature in the certificate.
user permission	See permission.

Index

A

access rights
  reviewing ... 111
allowed hosts ... 45
Allowed Hosts ... 22
Allowed Hosts tab ... 133
ancestor permission level ... 111
audit log ... 96
authentication ... 79
  audit log ... 96
  of users ... 67
  scheme, assigning to users ... 95
  schemes ... 68
authentication scheme ... 119
  password management ... 95
authorization checklist ... 102
authorization management ... 11
auto logoff
  station ... 98
AX authentication ... 68
AXDigestScheme ... 68

B

backup, *See* certificate, exporting
basic categories ... 104
best security practices ... 12
browser trust store
  installing client certificate in ... 75
bulk certificate signing ... 56

C

CA certificate ... 28
categories ... 104
category
  adding and editing ... 104
  basic ... 104
  deleting ... 106
  for assigning components ... 105
  New window ... 151
Category
  Browser ... 140
  Crowser ... 105
Category Sheet ... 142
certificate ... 21
  backing up, *See* certificate, exporting
  change to the private key ... 65
  create ... 148
  creating a root CA certificate ... 28
  creating a server cert ... 35
  deleting ... 48
  expired ... 46
  exporting ... 42
  importing into a User Trust Store ... 43
  importing into the User Key Store ... 41
  installing in a remote platform/station ... 43
  naming convention ... 22
  self-signed ... 19
  Session Info ... 44
  signing ... 39
  stores locations ... 22–23
  TLS
    session info ... 44
  types ... 18
  wizard ... 48
Certificate Authority ... 37
Certificate Export windows ... 146
certificate management ... 130
Certificate Management plugin ... 129
Certificate Signing Request
  folder structure ... 24
Certificate Signing window ... 145
certificate wizard
  generate CA Root certificate ... 49
certificates
  verify imported ... 54
certManagement folder ... 24
CertManagerService ... 130
checklist
  certificate creation and signing ... 25
  platform/station for server verification ... 26
  Supervisor station for server identity ... 26
  user authentication ... 67
client certificate ... 18, 71
  exporting ... 73
  installing in browser ... 75
  user workflow ... 71
client certificate authentication ... 69
  admin workflow ... 69
Client Certificate Authentication
  configure ... 69
client/server relationships ... 17, 28
ClientCertAuth
  users workflow
    browser login ... 76
    Workbench login ... 78
clientCertAuth-ClientCertAuthScheme ... 113
ClientCertAuthScheme ... 68
code-signing certificate ... 37
component
  adding a component ... 109
  assigning to a basic category ... 105
  permission level ... 102
  permissions ... 11, 101
  permissions, troubleshooting ... 112
component permissions checklist ... 102
components ... 113, 137–138

config flag, setting 102
configuring client port 60
controller,
replacing securely 65
cryptography 17
CSR, *See* Certificate Signing Request
folder structure 24
customize SAML attribute mapping 83

D

DigestScheme 67–68
document change log 7

E

edit bog file passphrase offline 60
Edit roles window 152
email
securing using TLS 61
encryption 20–21
engineering platform/station
certificate configuration 26

F

file permissions 111
Foxs properties 59

G

gauth palette 67
gauth-GoogleAuthenticationScheme 118
Generate Self-Signed Certificate window 148
Google Authentication 79
Google Authenticator app 67
GoogleAuthenticationScheme 68

H

history permissions 112
https
required to set password in browser 96
Https properties 59

I

identity verification 17, 21
inherited categories 104
install a copied station 60
installing
client certificate 75
IT scanners 14

K

keys
public and private 20
kiosk 79
kiosk-like mode
enabling 79

L

Login with SSO 84

N

naming convention
for certificates 22
network user
prototypes 90
New category window 151
niagara_user_home 24
nss-CertificateExpiryPoint 117
nss-CertificateFolder 116
nss-SecurityDashboardView 126
nss-SecurityService 114

O

Operator config flag 102

P

password
expiration 95
management 95
renewing an expired password 98
setting up 96
setting up for a user 96
strength 32
strength, setting up 95
troubleshooting 99
warning period 95
permission level 102
permissions 107
editing 110
to access files 111
to view history files 112
Permissions
map 155
permissions checklist 102
PKI certificate
creating 35
platCrypto-CertificateParameter 137
platCrypto-
CertificateSignerMultipleSelectionTool 136
platCrypto-CommonNameTemplate 138
platform
authentication window 152

connect window 153
opening a secure connection 26
stores 22
TLS properties 154
platform/station
checklist for server verification 26
plugins 113, 125
Certificate Management view 129
precautions 11
private key 20
prototype
for SAML authentication 81
network users 90
public key 20

R

Role Manager 144
roles
about their configuration 107
add and edit 152
adding 92, 107
and permission level 102
assigning to users 95, 110
editing 110
reviewing access rights 111
testing 110
root CA certificate
creating 28
installing in remote platform/stations 43
root certificate
importing into Windows trust store 41

S

SAML authentication scheme 88
SAML Authentication Scheme
configure 82
SAML Idp Service
configure subordinate station(s) 88
saml-CircleOfTrust 122
saml-CircleOfTrustEditor 139
saml-SAMLCoTPrototypesMixIn 120
saml-SAMLIdPService 121
saml-SAMLUser PrototypeFolder 124
saml-SamlXmlDecrypter 124
saml-StationServiceProviderFolder 124
SAMLAttributeMapper 118
SAMLAuthenticationScheme 68, 119
samlEncryption-SamlXmlEncrypter 125
SAMLIdpService
preliminary steps 86
workflow 86
SAMLIdPService
about 85
setting up 87
scanners 14
secure communication 11, 60
configuring station security properties 59
configuring Supervisor and platforms 59
enabling clients 60
encryption 21
Foxs properties 59
Https properties 59
Niagara check list 25
NiagaraNetwork enabling 28
securing outgoing email 61
Security Dashboard
overview 13
security precautions 11
self-signed certificate 65
self-signed client key 33
server authentication
fixing error conditions 62
TCP ports 64
server certificate 18
creating 35
server identity
Supervisor setup check list 26
server identity verification
setting up 25
server verification
station setup check list 26
Session Info 44
window 156
Set up alarming
on certificate expiration 47
signing multiple certificates 56
signingService-SessionTokenUxManager 125
SSO 80, 118–119
login 84
station
authentication window 157
connect window 157
health 44
logging off 96
logging on 96
security model 101
set up server verification 25
stores 22
station-to-station user 91
stations
configuring security properties 59
stores
accessing 22
file names 24
locations on the Supervisor platform 23
System Trust Store 22

T

TCP ports 64
TLS, *See* Transport Layer Security protocol
keys 20
platform properties 154

TLS level
verify 54
Transport Layer Security protocol 17
troubleshooting
component permissions 112
server authentication 62
trust store
Windows 41
Trust Store tab 132
two-factor authentication 67, 79

U

user
adding and editing 92
authentication 68
password 95
prototypes 90
user authentication 11
user authentication checklist 67
User Key Store 22
tab 130
user prototype
for SAML authentication 81
User Trust Store 22
importing a certificate 43
users
and the UserService 90
UserService
component permission level 103
setting up a password 96

V

views ..... 113, 125, *See* plugins
vulnerability management tools 14

W

windows 113, 145
Windows trust store 41