

# Technical Document

## **Niagara Station Security Guide**

**July 25, 2019**



# Niagara Station Security Guide

## **Tridium, Inc.**

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

## **Confidentiality**

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

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

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

## **Trademark notice**

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

## **Copyright and patent notice**

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

Copyright © 2019 Tridium, Inc. All rights reserved.

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

# Contents

<b>About this guide .....</b>	<b>7</b>
Document change log .....	7
Related documents .....	9
<b>Chapter 1 About station security .....</b>	<b>11</b>
Security precautions .....	11
Security best practices.....	12
Security Dashboard overview.....	13
<b>Chapter 2 Secure communication .....</b>	<b>15</b>
Client/server relationships .....	15
Certificates .....	16
Self-signed certificates .....	17
Keys .....	18
How certificates verify identity.....	19
Encryption .....	19
Naming convention .....	19
Certificate stores.....	20
Accessing the stores.....	20
Stores folder structure.....	21
Stores file names .....	22
CSR folder structure .....	22
Creating a CSR folder structure.....	22
Certificate set up.....	23
Root and intermediate certificate checklist .....	23
Supervisor/engineering PC checklist .....	24
Platform and station checklist .....	24
Opening a secure platform connection (niagrad) .....	25
Enabling the NiagaraNetwork.....	26
Confirming client/server relationships .....	27
Creating a root CA certificate .....	27
Creating a server certificate .....	30
Creating a code-signing certificate.....	31
Creating a CSR.....	33
Signing a certificate .....	33
Importing the signed certificate back into the User Key Store .....	35
Installing a root certificate in the Windows trust store .....	35
Exporting a certificate .....	36
Manually importing a certificate into a User Trust Store .....	37
Installing a certificate.....	37
Station health confirmation.....	38
Viewing session information.....	39
Allowed hosts management.....	39
When a certificate expires.....	40

Deleting a certificate .....	40
Configuring secure platform communication .....	40
Configuring secure station communication .....	41
Enabling clients and configuring them for the correct port .....	41
Securing email.....	41
Secure communication troubleshooting .....	42
Default TCP/IP ports .....	45
Certificate management when replacing a controller .....	45
<b>Chapter 3 User authentication .....</b>	<b>47</b>
User authentication checklist .....	47
Authentication schemes .....	48
Set up client certificate authentication.....	49
Admin workflow for client certificate authentication .....	49
User workflow for client certificate authentication .....	51
Logging in via browser using client certificate authentication .....	56
Logging in via Workbench using client certificate authentication .....	58
Enabling a kiosk-like mode using ClientCertAuth .....	58
Setting up Google authentication .....	59
Single Sign On .....	59
Creating a User Prototype for SAML Authentication .....	61
Configuring the SAML Authentication Scheme .....	61
Customizing SAML attribute mapping.....	63
Logging in with SAML SSO .....	63
Network users.....	65
Challenge .....	65
User Service changes.....	65
NiagaraNetwork changes .....	66
Network user related properties .....	66
User prototypes .....	67
Station-to-station users .....	70
Adding or editing a user .....	70
Assigning authentication schemes to users .....	71
Password management .....	71
Setting up password strength .....	71
Setting up password options .....	72
Setting up a user's password .....	72
Logging on to a station.....	72
Station Auto Logoff.....	74
Changing your password .....	74
User authentication troubleshooting .....	75
<b>Chapter 4 Authorization management.....</b>	<b>77</b>
Component permissions checklist .....	78
Component permission level.....	78
Changing a component Config flag.....	78
UserService permission levels .....	79

Categories .....	80
Basic categories .....	80
Adding and editing a basic category .....	80
Assigning a component to a basic category .....	81
Deleting a category name .....	82
Roles and permissions .....	83
Adding roles and permissions .....	83
Adding a component .....	85
Editing roles and permissions .....	85
Assigning roles to users .....	86
Confirming access security .....	86
Reviewing permissions .....	86
Ancestor permissions .....	87
File permissions .....	87
History permissions .....	87
Component permissions troubleshooting .....	88
<b>Chapter 5 Components, views and windows .....</b>	<b>89</b>
Components .....	89
baja-AuthenticationService .....	89
baja-AuthenticationSchemeFolder .....	89
baja-SSOConfiguration .....	90
baja-DigestScheme .....	91
baja-GlobalPasswordConfiguration .....	91
baja-AXDigestScheme .....	92
baja-HTTPBasicAuthenticationScheme .....	92
baja-CategoryService .....	93
RoleService .....	94
baja-UserPrototype .....	94
baja-UserService .....	95
clientCertAuth-ClientCertAuthScheme .....	101
FoxService .....	102
nss-SecurityService .....	105
saml-SAMLAttributeMapper .....	106
saml-SAMLAuthenticationScheme .....	108
samlEncryption-SamlXmlDecrypter .....	109
web-WebService .....	109
Plugins (views) .....	114
nss-SecurityDashboardView .....	114
platCrypto-CertManagerView .....	117
wbutil-CategoryBrowser .....	123
wbutil-CategoryManager .....	124
wbutil-CategorySheet .....	125
wbutil-PermissionsBrowser .....	126
wbutil-RoleManager .....	128
Windows .....	129
Certificate Export windows .....	129

Edit history export window ..... 131

Generate Self-Signed Certificate window ..... 132

New category windows ..... 134

New/Edit roles window ..... 134

Platform Authentication window ..... 135

Platform Connect window ..... 136

Platform TLS settings ..... 136

Permissions map ..... 137

Session Info window ..... 138

Station Authentication window ..... 139

Station Connect window..... 139

**Glossary .....141**

**Index.....143**

## 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. In order to make the most of the information in this book, readers should have some training or previous experience with Niagara 4 or NiagaraAX 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.

### 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 SAMLAuthenticationScheme 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 [www.tridium.com/resources/library](http://www.tridium.com/resources/library))



# Chapter 1 About station security

## Topics covered in this chapter

- ◆ Security precautions
- ◆ Security best practices
- ◆ Security Dashboard overview

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 enterprise 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 two-factor authentication.



Do not share accounts (log on using someone else's credentials). Always log on 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.

## 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.

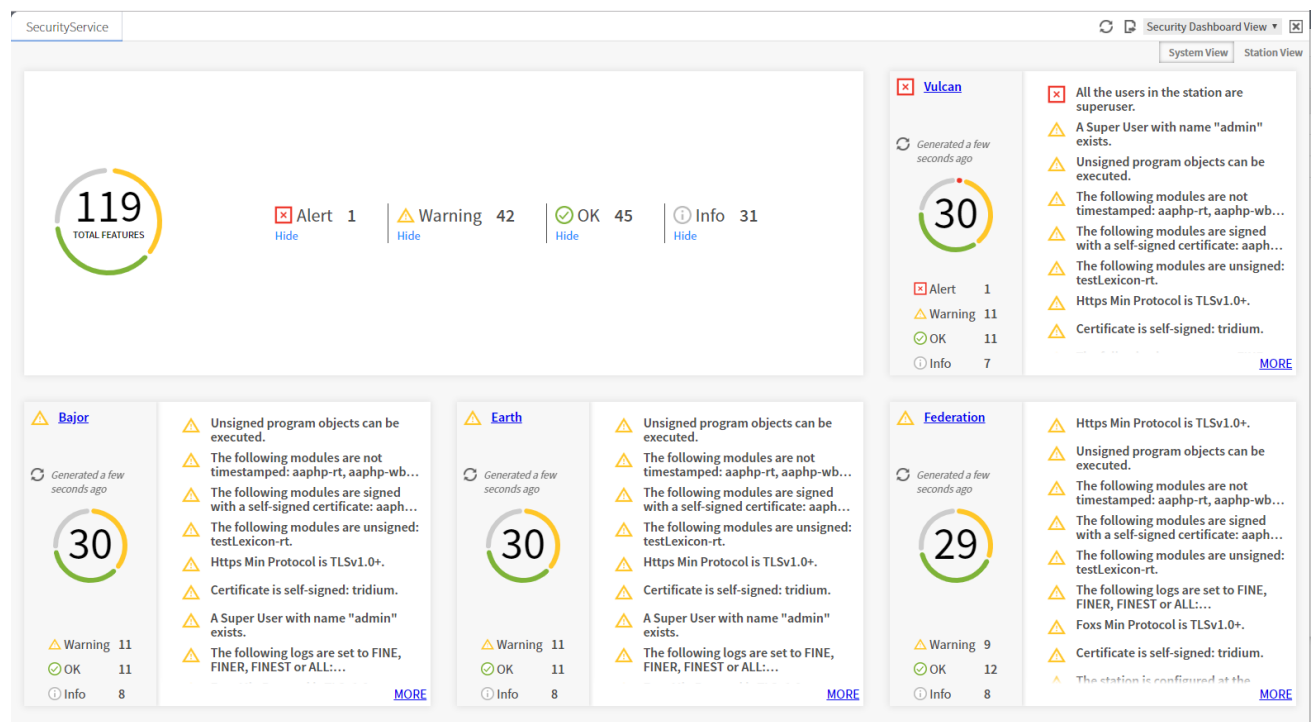
## Security Dashboard overview

In Niagara 4.8 and later, 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 main view on the station's SecurityService. The view alerts you to security weaknesses such as poor password strength settings; expired, self-signed or invalid certificates; unencrypted transport protocols, etc., indicating areas where the configuration should be more secure. Other reported data includes: system health, number of active accounts, inactive accounts, number of accounts with super-user permissions, etc.

Figure 1 Example Security Dashboard View



The Summary card in the upper left corner of the view summarizes the number and type of security status messages that are generated for all Services on the station. Each “card” (or pane) in the view provides security configuration data for a particular service. Typically, each card provides a hyperlink to that service (or to a component) so that you can easily change a configuration. In cases where there is no component to link to, no hyperlink is provided on the card.

For complete details, see “ [nss-SecurityDashboardView, page 114](#)” in the “Components and views” chapter of this guide.

# Chapter 2 Secure communication

## Topics covered in this chapter

- ◆ Client/server relationships
- ◆ Certificates
- ◆ Certificate stores
- ◆ CSR folder structure
- ◆ Certificate set up
- ◆ Configuring secure platform communication
- ◆ Configuring secure station communication
- ◆ Enabling clients and configuring them for the correct port
- ◆ 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 4 supports the TLS (Transport Layer Security) protocol, versions 1.0, 1.1 and 1.2. 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 4.0 supports and uses secure communication and signed code by default. You do not need to purchase an additional license.

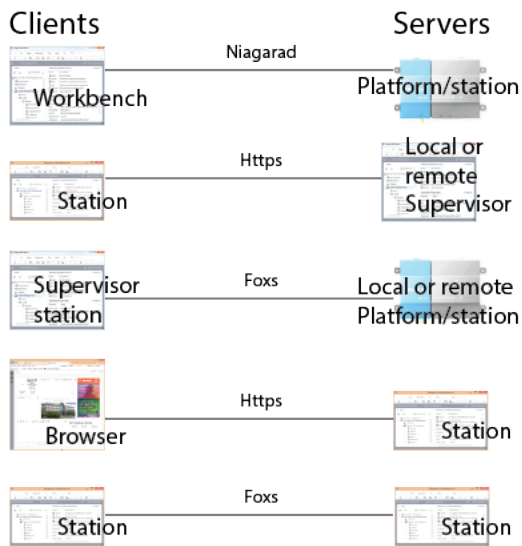
Security is an ongoing concern. While you will find much valuable information in the secure communication topics, expect future updates and changes.

## Client/server relationships

Client/server relationships identify the connections that require protection. Niagara 4.0 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 Communication 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 Niagaraad. 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 the Java Applet profile. 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:

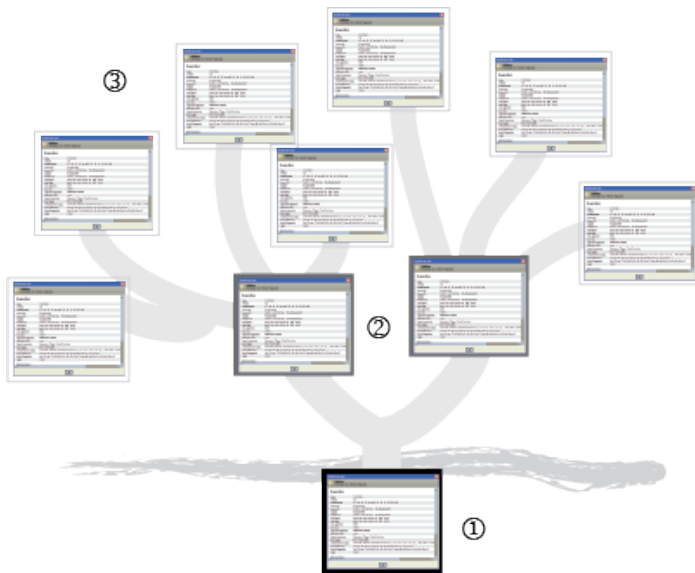
- A **CA (Certificate Authority) certificate** is a self-signed certificate that belongs to a CA. This could be a third party or a company serving as its own CA.
- A **root CA certificate** is a self-signed CA certificate whose private key is used to sign other certificates creating a trusted certificate 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.



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 Niagara (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.

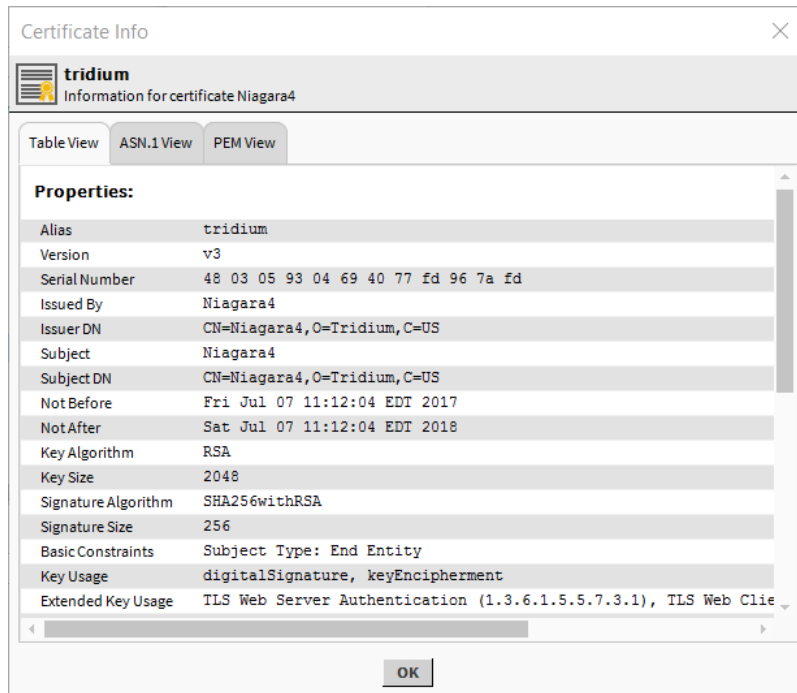
The system supports two types of self-signed certificates:

- 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 3 Example of a 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 Foxes 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.

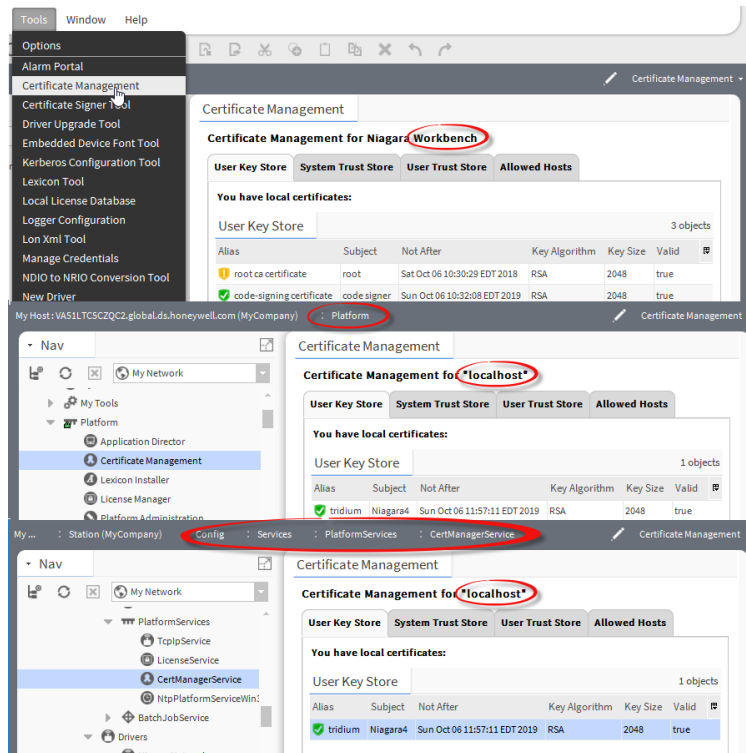
**Step 2** Open a supervisor or remote platform/station.

**Step 3** Do one of the following:

- To access the Workbench stores, 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.  
  
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**.
- 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 <sup>1</sup>	<b>My Host→My File System→User Home→security</b>	C:\Users\username\Niagara4.0\security
Supervisor or engineering workstation <sup>2</sup>	<b>Platform→RemoteFileSystem→User Home (Read Only)→security</b>	ProgramData\Niagara4.0\security
remote controller <sup>3</sup>	<b>Platform→RemoteFileSystem→User Home (Read Only)→security</b>	\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\Niagara4.0\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\niagara4.0\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 Stores**; and importing the root CA certificate (or intermediate certificate) into client **User Trust Stores**.

**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 in Workbench 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 your 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. See [Chapter 2 Secure communication, page 15](#).
- ☐ 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. See [Certificates, page 16](#).
- ☐ Logical certificate naming convention established (a naming convention is not required, but it will help you differentiate among your certificates). See [Naming convention, page 19](#).
- ☐ CSR folder structure under the **certManagement** folder in the `niagara_user_home` created. See [Creating a CSR folder structure, page 22](#).
- ☐ Root CA certificate and any intermediate certificates created. See [Creating a root CA certificate, page 27](#).
- ☐ CSR for each intermediate and code-signing certificate created. See [Creating a CSR, page 33](#).
- ☐ Any intermediate and code-signing certificates signed using the root CA certificate. See [Signing a certificate, page 33](#).
- ☐ Any signed intermediate certificates imported back into the Workbench User Key Store where they were originally created. See [Importing the signed certificate back into the User Key Store, page 35](#).

- ☐ Backup of the root CA certificate and the signed intermediate certificates created. See [Exporting a certificate, page 36](#).
- ☐ Root CA certificate with only its public key exported in preparation to import it into the platform/station Trust Stores. See [Exporting a certificate, page 36](#)

## 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. See [Enabling the NiagaraNetwork, page 26](#).
- ☐ Remote controller(s) set up as Supervisor/engineering PC client(s). See [Confirming client/server relationships, page 27](#).
- ☐ Server certificate for the Supervisor/engineering PC platform/station created. See [Creating a server certificate, page 30](#).
- ☐ CSR for the server certificate generated. See [Creating a CSR, page 33](#).
- ☐ Server certificate CSR signed by the root CA or intermediate certificate's private key. See [Signing a certificate, page 33](#).
- ☐ Signed server certificate imported back into the platform/station's **User Key Store**. See [Importing the signed certificate back into the User Key Store, page 35](#).
- ☐ 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**. See [Manually importing a certificate into a User Trust Store, page 37](#).
- ☐ Certificate .pem file deleted.
- ☐ Confirmed platform (Niagarad) enabled. See [Configuring secure platform communication, page 40](#).
- ☐ Secure **FoxService** confirmed (the default) and **Foxs Cert** selected. See [Configuring secure station communication, page 41](#)
- ☐ Secure **WebService** confirmed (the default) and **Https Cert** selected. See [Configuring secure station communication, page 41](#)

## 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. See [Enabling the NiagaraNetwork, page 26](#).
- ☐ Supervisor set up as controller client. See [Confirming client/server relationships, page 27](#).
- ☐ Server certificate(s) created on each platform/station. If needed, a separate server certificate created for each communication protocol: foxs, https, and platformtls created. See [Creating a server certificate, page 30](#).
- ☐ CSR(s) generated on each platform/station. See [Creating a CSR, page 33](#).
- ☐ CSR(s) signed by the root CA or intermediate certificate's private key. See [Signing a certificate, page 33](#).
- ☐ Signed server certificate(s) imported back into the platform/station **User Key Store**. See [Importing the signed certificate back into the User Key Store, page 35](#).
- ☐ 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.

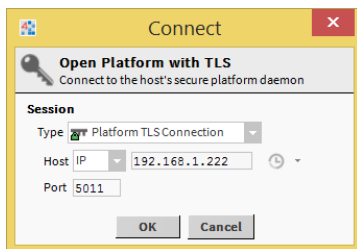
- ☐ If serving as the CA, company's root CA certificate imported into the platform/station's **User Trust Store** either one at a time, or using a provisioning job. See [Manually importing a certificate into a User Trust Store, page 37](#) and [Installing a certificate, page 37](#).
- ☐ Confirmed platform (Niagarad) enabled and correct certificate assigned. See [Configuring secure platform communication, page 40](#).
- ☐ Secure **FoxService** confirmed (the default) and **Foxs Cert** selected. For the specific procedure, see [Configuring secure station communication, page 41](#).
- ☐ Secure **WebService** confirmed (the default) and **Https Cert** selected. For the specific procedure, see [Configuring secure station communication, page 41](#).

## 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 **tridium** 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.

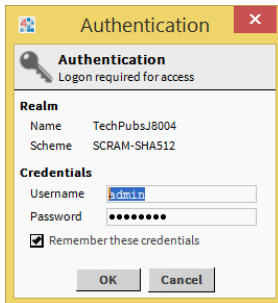
**CAUTION:** Do not approve a certificate if you do not recognize these properties. The weakest link in the security chain is the user who simply clicks OK without thinking.

**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 (**tridium**) 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 **tridium** certificate for each controller in your network.

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

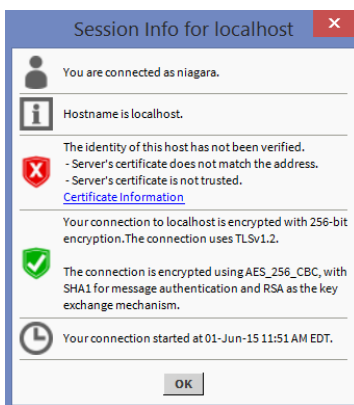


**Step 5** Enter your platform credentials and click **OK**.

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 root CA certificate is used to sign intermediate, server and code-signing certificates. It resides a **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 client'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 stores by clicking **Tools→Certificate Management**.  
The **Certificate Management** view opens to the **User Key Store**.
- Step 2 Click the **New** button at the bottom of the view.  
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 Give the certificate at least an **Alias**, **Common Name (CN)**, **Organization**, **Locality**, **State/Province**, and **Country Code**.
  - 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**.
  - Although **Locality** and **State/Province** are not required and are arbitrary, leaving them blank generates a warning message.
  - The two-digit **Country Code** is required and must be a known value, such as: US, IN, CA, FR, DE, ES, etc. (see [countrycode.org](http://countrycode.org) for a list).
  - **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.
- Step 4 For **Certificate Usage**, select **CA**, and click **OK**.  
The **Private Key Password** window opens.
- Step 5 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.
- Step 6 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, and code-signing). 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.

**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.

**Step 7** Select the new root CA certificate and click the **Export** button at the bottom of the view.

The **Certificate Export** window opens.

The **Certificate Export** window is shown with the following details:

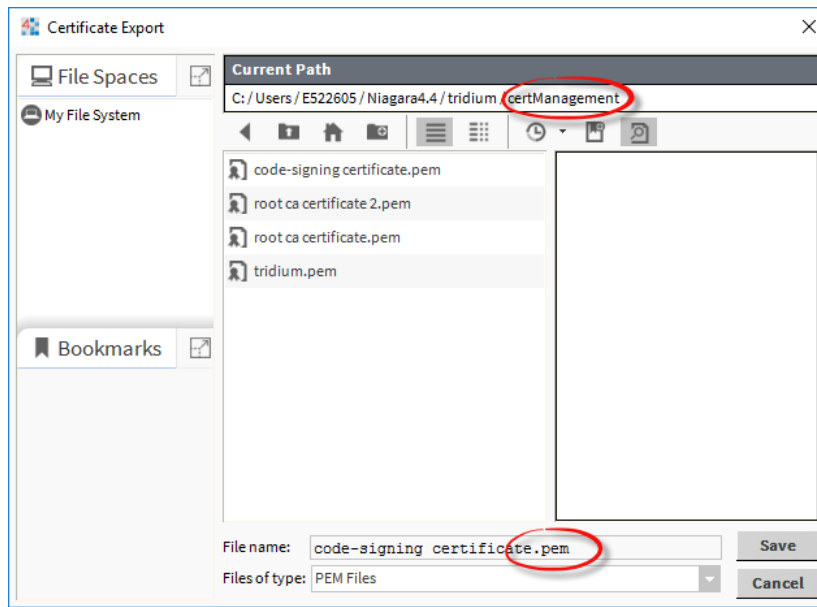
- Certificate Section:**
  - ☒ **Export the public certificate**
  - Buttons: **Table View** (selected), **ASN.1 View**, **PEM View**
  - Properties:**

Version	v3
Serial Number	47 e6 05 b1 75 4d 70 70 35 54 50 0f
Issued By	root 2
Issuer DN	CN=root 2,O=My Company,L=Anywhere,ST=Any...
- Private Key Section:**
  - ☐ **Export the private key**
  - Private Key Password (required):
  - ☒ **Encrypt exported private key**
    - ☒ **Reuse password to encrypt private key**
  - Password:
  - Confirm:
  - Buttons: **OK**, **Cancel**

**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 8** 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 9** Navigate to a location on a thumb drive, and click **Save**.

The system reports that it exported the certificate successfully.

**Step 10** 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 and 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 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:

- a. Expand **Platform** and double-click **Certificate Management**.
- b. 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 Names** 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-digit **Country Code** is required and must be a known value, such as: US, IN, CA, FR, DE, ES, etc. (See [countrycode.org](http://countrycode.org) for a list.).
- **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**.

**Step 5** When you have filled in all information, click **OK**.

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 6** 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 7** 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-digit code (must be US or us, not USA).

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 Niagara 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:**

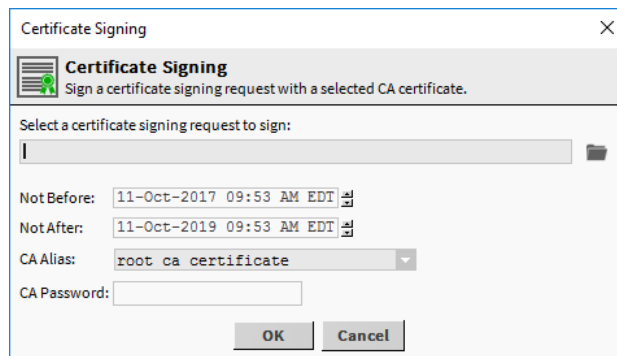
- You are working in Workbench on a physically and electronically secure PC that is never connected to the Internet, and is used exclusively to sign certificates.

- 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.

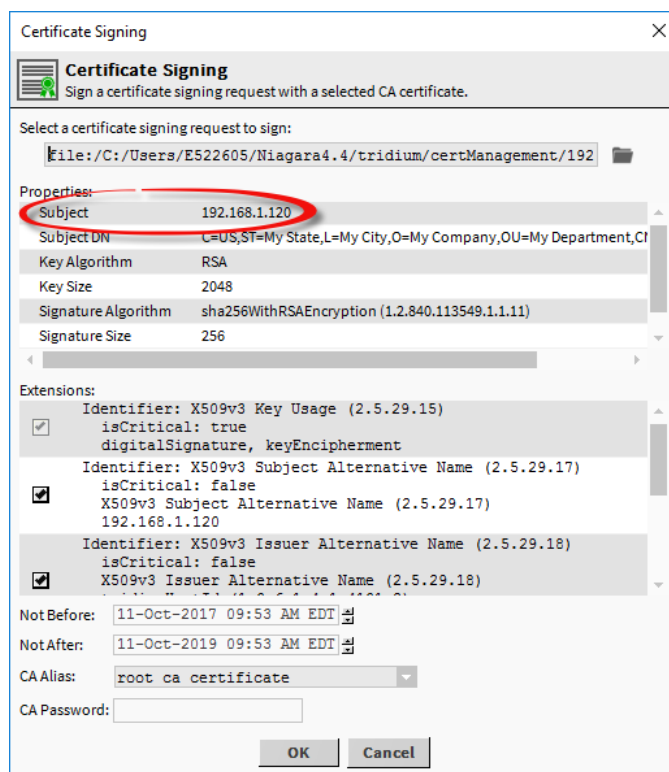
**Step 1** In Workbench on your physically and electronically secure (and never connected to the Internet) PC that is used exclusively to sign certificates, click **Tools→Certificate Signer Tool**.

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.

Repeat this procedure for each CSR.

## 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).

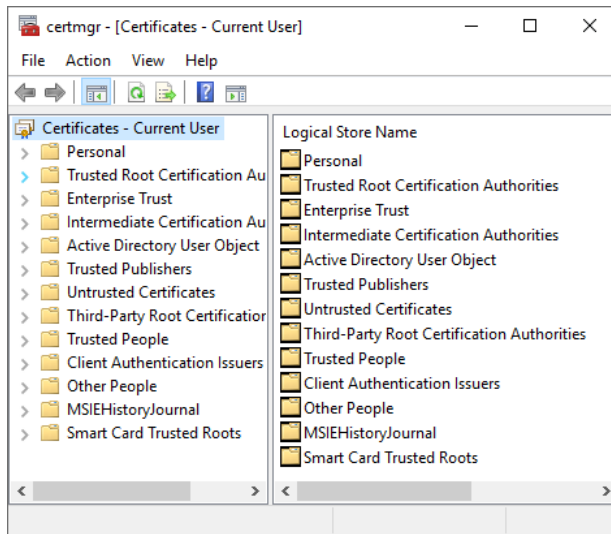
- Step 1 Click **Import**.
- Step 2 Locate and select the signed certificate's `.pem` file (the output of the certificate signer or the `.pem` file you received from a third-party CA) and click **Open**.  
The **Certificate Import** window opens.
- Step 3 Confirm that you are importing the correct certificate 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 secureWorkbench 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 Stores**, 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 your **System Trust Store** already contains the root CA certificate of 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** of all platform/stations that serve as system clients. To do this, use an Install Certificate provisioning job. This can be useful before running a signed provisioning robot on several stations.

**Prerequisites:**

- The BatchJobService and ProvisioningNwExt components are available under your NiagaraNetwork.
- The root CA or signed intermediate certificate is available on a thumb drive.
- The **provisioningNiagara** palette is open.

Step 1 Open the platform/station stores on a Supervisor (or engineering) computer and click the **User Trust Store** tab.

**NOTE:** Make sure you are in the platform/station stores. You cannot complete this procedure if you import the certificate into the Workbench **User Trust Store** of your Supervisor or engineering computer.

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

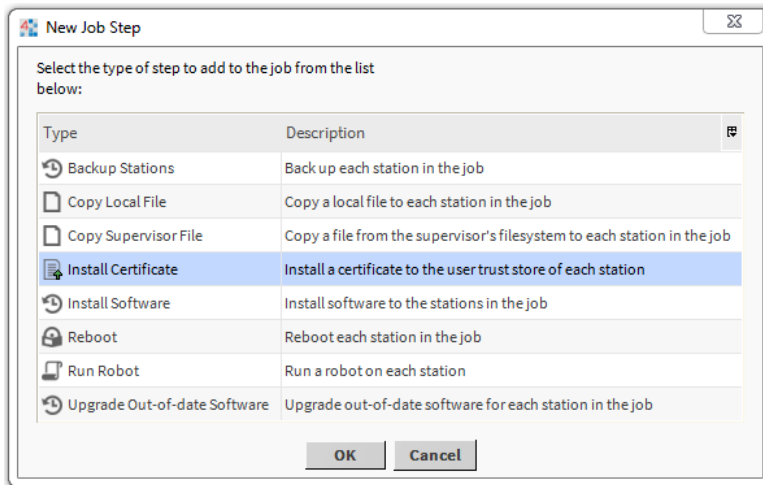
Step 3 Confirm that the **Subject** of the certificate identifies it as the root CA certificate and click **OK**.  
The system imports the certificate in preparation for the provisioning job.

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

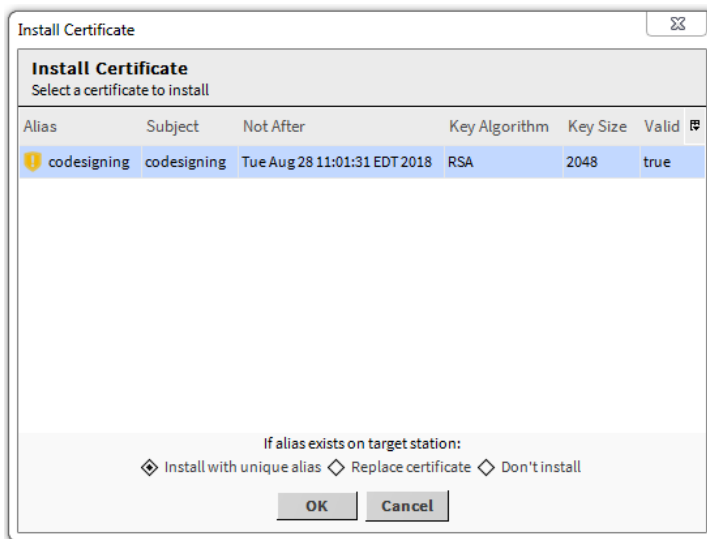
Step 5 Drag a NiagaraNetworkJobPrototype component to this location and name the component something like, "Root CA certificate provisioning."

Step 6 Double-click the new component.  
The **Niagara Network Prototype View** opens.

- Step 7** In the **Steps to run for each station** pane, click the plus icon.  
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

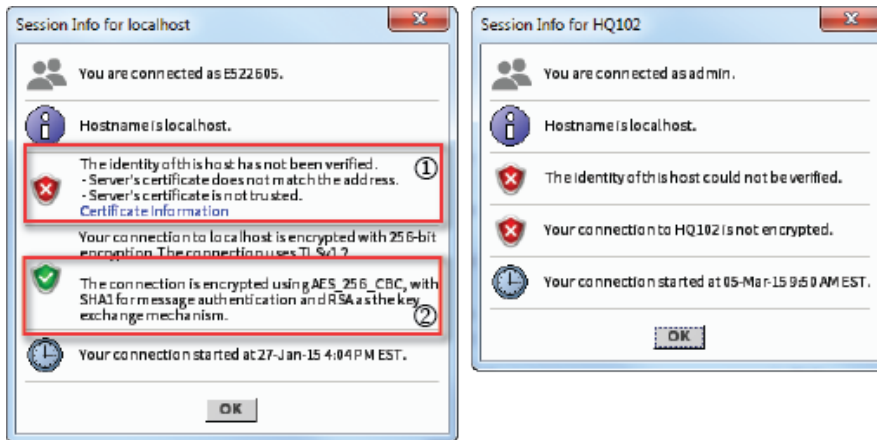
The **Session Info** window provides useful information and a graphical representation of certificate status (green and red icons).

**Prerequisites:** Workbench is running.

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

- Click the Session Info icon (i) 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

If you used self-signed certificates to get started, 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, and platform/station).

- 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.

## 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.

## 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**).

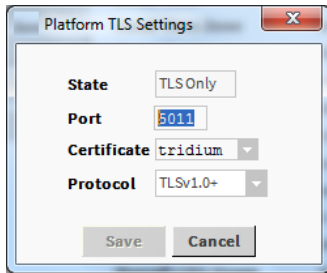
## Configuring secure platform communication

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**→**Platform Administration** and double-click **Change TLS Settings**.

The **Platform TLS Settings** window opens.



The default **State** is **TLS only** and the Daemon HTTP **Port** indicates 3011 (disabled in TLS settings) in the **Platform Administration** view. However, **State** can be changed on the controller to **Enable** or **Disable**. If you are using a separate certificate for verifying niagarad communication, this is where you select the **Certificate**.

Step 2 Configure the properties 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**.

## 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.

Use Ssl	<input type="radio"/> false
Use Start Tls	<input type="radio"/> false
Transport	Smtpt

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`.

- Step 2** To provide secure email, set one property to `true`, and the other `false`.

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

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 3** 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 4** 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 <b>Config→Services</b> , and double-click <b>FoxServices</b> .
foxs	4911	station	secure	
platform daemon	3011	niagarad (platform)	not secure	use the <b>Change HTTP Port</b> button under <b>PlatformPlatform Administration</b> .
platformtls (Platform Port)	5011	niagarad (platform)	secure	use the <b>Change TLS Settings</b> button under <b>PlatformPlatform Administration</b> .
http	80	browser	not secure	expand <b>Config→</b> , and double-click <b>FoxServices</b> .
https	443	browser	secure	
email, incoming account	110	receive	not secure	expand <b>Config→</b> , and double-click <b>EmailServices</b> .  <b>NOTE:</b> Do not enable Use Ssl or Use Start Tls without configuring the port as indicated in this table. Refer to the <i>Secure Communication</i> chapter for more information about these ports and properties.
email, outgoing account	25	send	not secure	
email, incoming account, <b>Use Ssl</b>	993 (IMAP), 995 (POP3)	receive	secure	
email outgoing account, <b>Use Ssl</b>	465	send	secure	
email, incoming account, <b>Use Start Tls</b>	143 (IMAP), 110 (POP3)	receive	secure	
email, incoming account, <b>Use Start Tls</b>	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.

# Chapter 3 User authentication

## Topics covered in this chapter

- ◆ User authentication checklist
- ◆ Authentication schemes
- ◆ Set up client certificate authentication
- ◆ Logging in via 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
- ◆ Network users
- ◆ Station-to-station users
- ◆ Adding or editing a user
- ◆ 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

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).

- Each user has been created.
- The authentication scheme has been selected for each user.
- If you are using the Google Authenticator, the app has been installed on the user's mobile device.
- Credentials (user name and password) have been set up for each user.
- User roles has 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.
- 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**: With this scheme, a user password is never directly sent to the station. Instead, proof is sent that the user knows the password. This scheme connects Niagara 4 supervisor to Niagara 4 station.
- **AXDigestScheme**: With this scheme, several messages are passed back and forth to prove that the client knows the password. The client's password is never actually transmitted, 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 Niagara 4 supervisor to connect to NiagaraAX station.

**NOTE:** 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 must be configured) are in the **ldap** palette:

- **LdapScheme**
- **KerberosScheme**

## Client Certificate Authentication Scheme

In Niagara 4.8 and later, the `clientCertAuth` palette contains the **ClientCertAuthScheme** which provides authentication using a user's certificate. Each user's certificate is directly bound to the user by storing the user's public certificate on the User object. Each user's public certificate matches the user certificate's private key. During a login attempt, the user is prompted to upload his or her certificate, the certificate is verified against the certificate stored on the User object. For more details, see [Admin workflow for client certificate authentication, page 49](#) and [User workflow for client certificate authentication, page 51](#) in the "User Authentication" chapter; and [clientCertAuth-ClientCertAuthScheme, page 101](#) in the "Components, views, and windows" chapter.

## 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 **Authentication-Schemes** container to configure the station for SAML Single Sign On. Authentication schemes that support Single Sign-On allow supported users to bypass entering a username in the pre-login step. Instead, the users are redirected to an alternate login page. For more details, see [saml-SAMLAuthenticationScheme, page 108](#).

## Set up client certificate authentication

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

The station Admin'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.

While the User must complete several tasks as well as provide the station Admin with the client certificate (public key).

Refer to the following Admin and User workflows for the list of procedures for each.

### Admin workflow for client certificate authentication

This workflow is performed by the station admin in order to configure the station for client certificate authentication.

Client authentication is a method for users to securely access a remote station (via 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.

Typically, only the user (client) has access to the private key of their certificate for client certificate authentication. However, the public key of the certificate is not considered private data, and can be shared. For this workflow the station admin first needs to acquire the user's public client certificate, then create the station user account, and then set that certificate in the server authenticator. The following procedure details the configuration method.

### Configuring a user for client certificate authentication

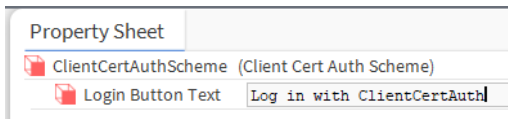
This procedure is performed by a station admin user. It describes the steps to configure a new user account to use the `ClientCertAuthScheme`, and assigns the user's public certificate to the user's `ClientCertAuthenticator`.

#### Prerequisites:

- You are working in a properly licensed Niagara 4.8 Workbench installation.
- You have already acquired the public certificate created by the user.

**NOTE:** A separate workflow for the user is provided in this chapter that describes how to create a client certificate, export it with private and public keys, and install the certificate in a browser. For details, see [User workflow for client certificate authentication, page 51](#).

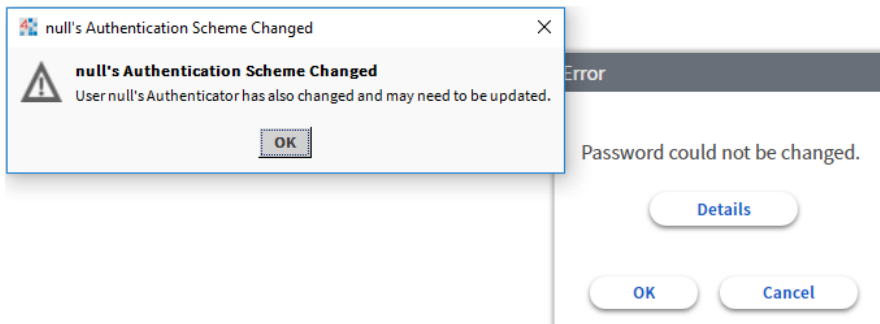
- Step 1 In the Workbench Nav tree, expand the station's **Services→AuthenticationService→AuthenticationSchemes** node.
- Step 2 Open the **clientCertAuth** palette and drag the **ClientCertAuthScheme** to the AuthenticationSchemes folder.
- Step 3 Expand the AuthenticationSchemes and double-click the 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 Double-click **UserService**, and in the **User Manager** click **New** to create a new user.
- Step 5 In the configuration popup window click **OK** to accept default entries for **Type to add** and **Number to add**.
- 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** dropdown list, select the **ClientCertAuthScheme**, and click **OK**.

**NOTE:** 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 and 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.

Property Sheet

user (User)

Full Name

Enabled ☒ true

Expiration ☒ Never Expires ☐ Expires On 30-Jan-2019 11:59 PM EST

Lock Out ☒ false

Language

Email

Authenticator Client Cert Authenticator

Certificate

Changing the user's certificate will prevent them from connecting until the FoxService and WebService are restarted. These services will restart momentarily but can be restarted manually at any time.

local:|file:~certManagement/clientcertif

Step 9 Click **Save**.

**NOTE:** The **Save** action triggers a timer to restart the Fox and Web services in 2-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 login to the station via 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 prelogin window where the user simply clicks the **Login With ClientCertificateAuth** button and immediately authenticates to the station. There is no need to enter username/password credentials. For more details, see the procedure "Logging in via browser using client certificate authentication".

## User workflow for client certificate authentication

This workflow is performed by a user as a preliminary step. This is done prior to the station admin setting up for client certificate authentication.

First, you will generate a new client certificate, and then export it in two different formats. Export the client certificate first with a public key which simply means that it is not considered protected data, you can share it as needed. Export the client certificate again but this time with an encrypted private key.

Afterwards, give the certificate with public key to the Station Admin who will use it in setting up Client Certificate Authentication on the station.

The exported certificate with encrypted private key should be saved to a safe location on our PC file system for later use. The private key is sensitive data and should be kept well protected. Do not store it somewhere where it might be accessible to others.

In a subsequent procedure, you will install your certificate with private key on your web browser.

### Creating a client certificate

This procedure is done by the User. It describes the steps to generate a new client certificate that will be used for client certificate authentication. This method uses the Workbench **Certificate Management** tool to generate the certificate, rather than the station's certificate stores. By generating it this way, the private certificate is installed on Workbench which you may need.

#### Prerequisites:

- You have the required authority to create certificates.
- You are running a Niagara 4.8 Workbench on your PC.

**NOTE:** Those end users who do not have Workbench will need to use some other tool (e.g. 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**.

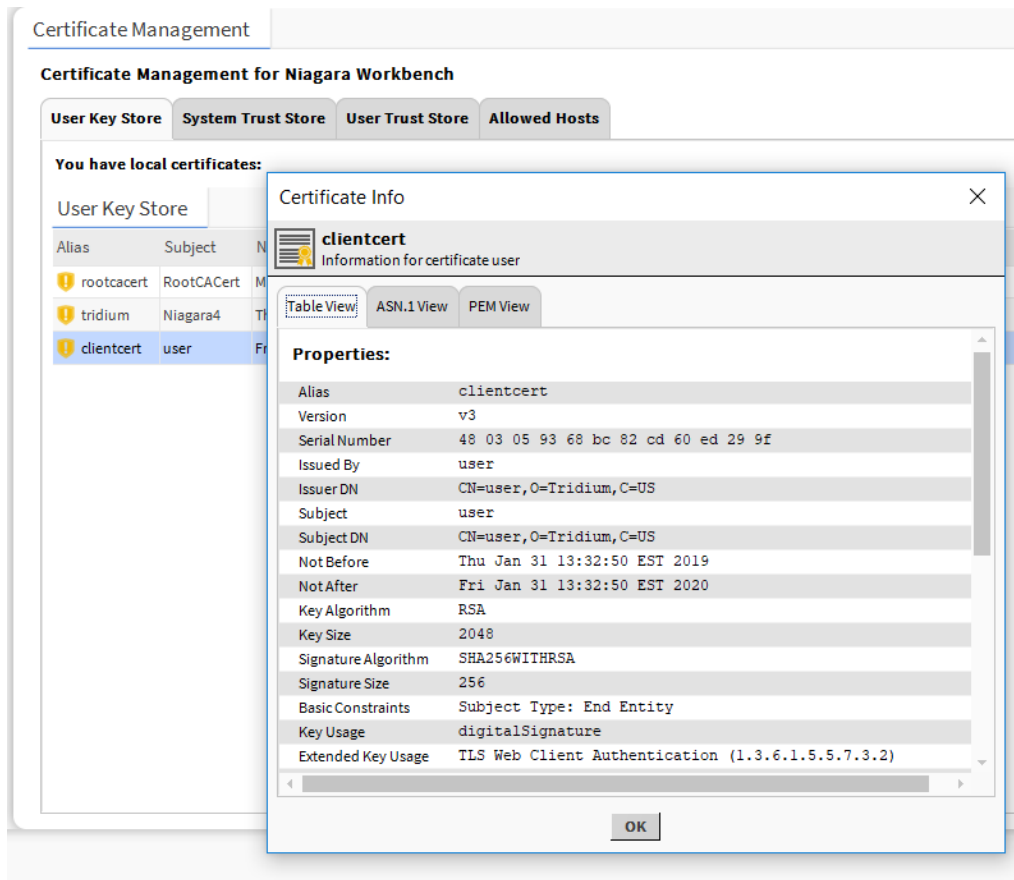
- Use **Alias** to identify this as a client certificate.
- Entering your station username in the **Common Name** field facilitates later authentication on the station.
- The two-digit **Country Code** is required and must be a known value, such as: US, IN, CA, FR, DE, ES, etc. (See [countrycode.org](http://countrycode.org) for a list.).

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

- Step 5** When you have filled in the required fields and selected “Client” for certificate usage, 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 a Niagara 4.8 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**

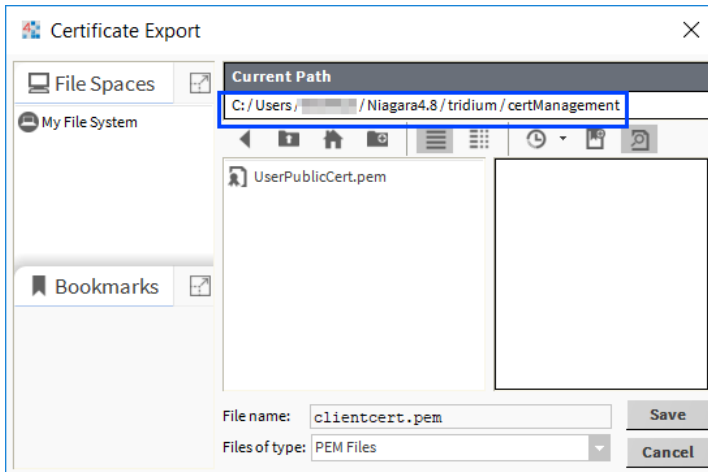
**Step 1** In Workbench, open the **Certificate Management** view.

**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**).

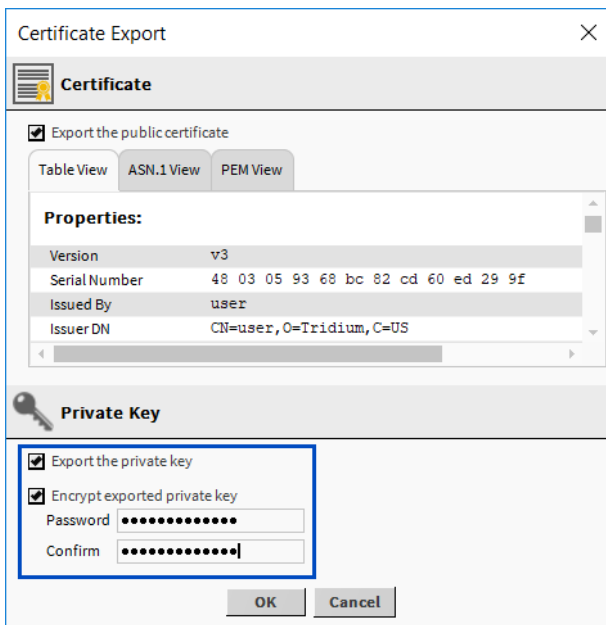
**Step 4** Use the default location on your PC's file system (or navigate to another location) and click **Save**.



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

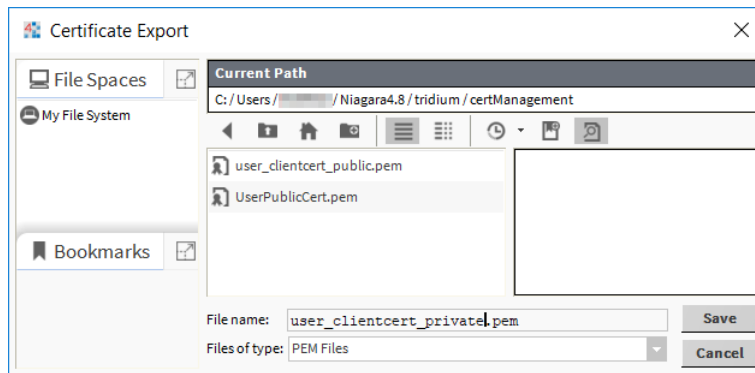
Proceed with the remaining steps ( 5–7) to export the **Private** certificate.

- Step 5** On the **User Key Store** tab, where your client certificate is still selected and click **Export** a second time.
- Step 6** This time in the **Certificate Export** window, click **Export** the private key and under **Encrypt** exported private key supply the additional 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.

- Step 7** Use the default location on your PC's file system (or navigate to another location) and click **Save**. 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**.

Your public and private client certificates are saved as \*.pem files to the ~certManagement folder in your User Home, or in the location you selected during the export.

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.

### ***Installing the private client certificate in your browser trust store***

This procedure describes the steps to install your client certificate with **Private** key in your browser's certificate trust store. This private certificate will be referenced on station login via browser, when the station is configured for client certificate authentication.

#### **Prerequisites:**

- You have previously generated a client certificate and exported it with 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 (e.g. \*.pfx or \*.p12 / \*.pkcs12).

**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.

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 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 browser using client certificate authentication

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

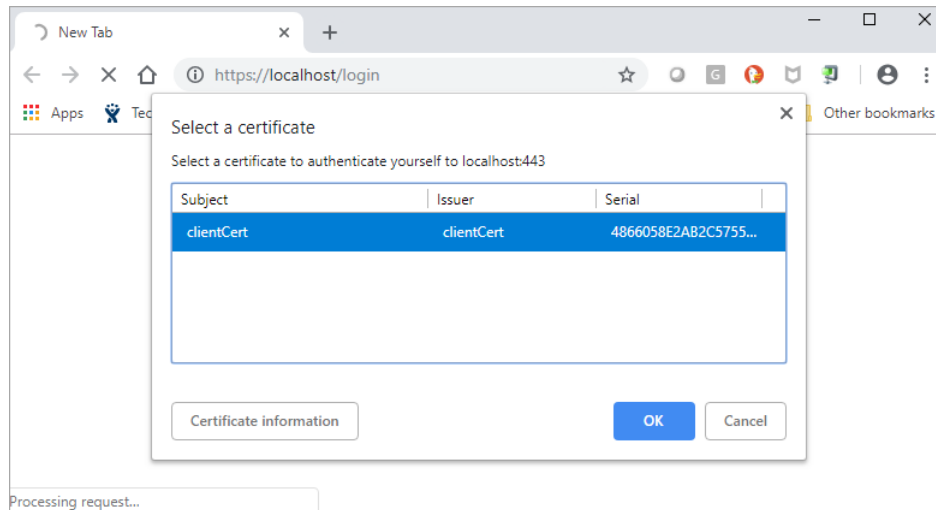
### Prerequisites:

- You are running a Niagara 4.8 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.

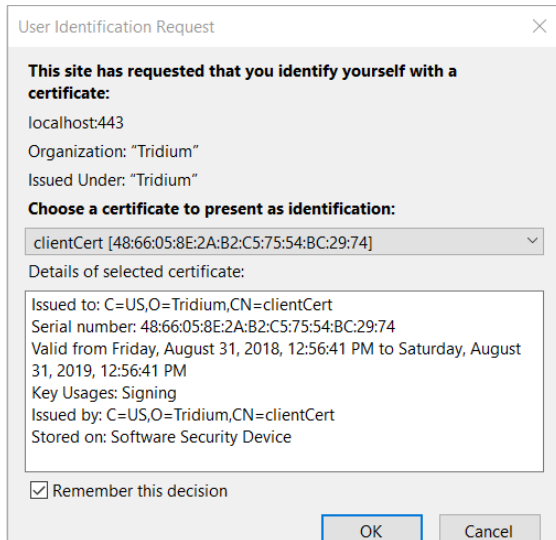
**Step 1** In the browser, enter the station address and press Enter.

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

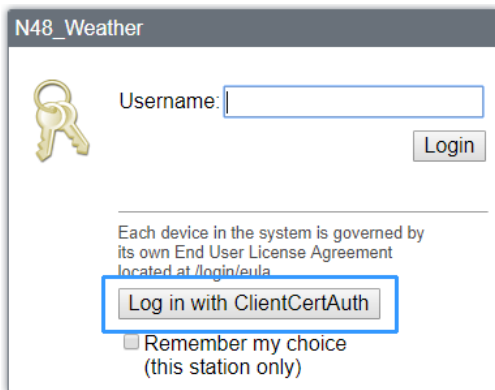
**Step 2** In the **Select a certificate** window, click to select your private certificate and click **OK**.



**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.

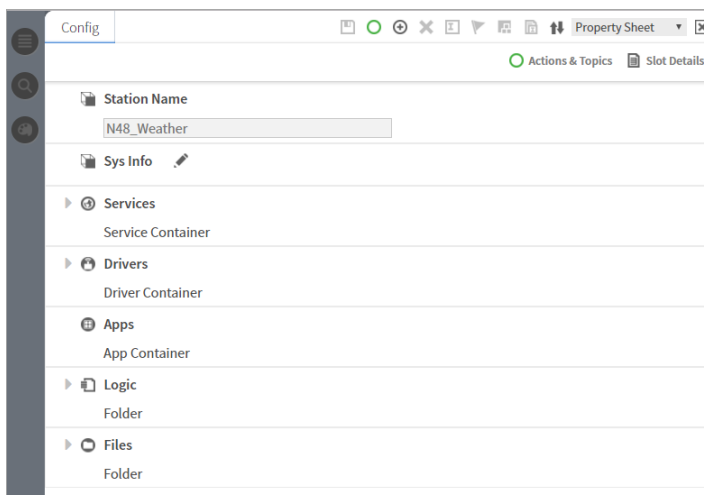


Step 3 In the station **Pre-Login** window, simply click the **Login With ClientCertAuth** button.



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

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

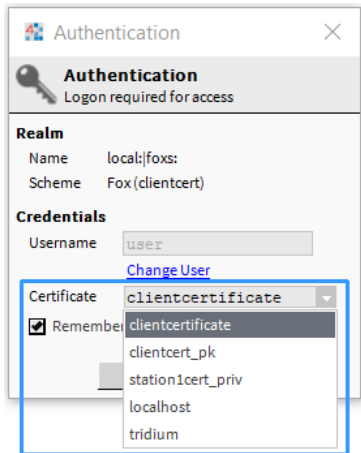


## Logging in via Workbench using client certificate authentication

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 4.8 and later, 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 station running Niagara 4.8 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

Niagara has an extensible Single Sign On (SSO) framework that can support many types of SSO. For example, the Kerberos scheme is an SSO scheme. And in Niagara 4.4 and later, there is added support for SAML SSO, which is the main supported SSO scheme and the focus of this topic.

### About SSO

SSO is an access control method that allows for automatic logging in to multiple related, but independent software systems. In the current implementation, SSO works via a browser connection to a station. When accessing multiple stations configured for Single Sign On, the user is only required to enter credentials once to access all stations. SSO also makes it possible to log in to individual stations without being prompted for user name or password each time.

**Figure 4** Login dialog for a station configured for SAML SSO

The advantages of this are evident for customers with more than one JACE controller:

- Users can log into one controller, and will not be prompted to log into other controllers which improves usability.
- There is a centralized management of credentials, meaning that users no longer need to maintain multiple copies of the same identity/role information, eliminating the errors inherent in duplication and being out of sync.
- One controlled authentication point, making authentication less complicated and ultimately, more secure.

A result of using SSO is that all credentials (identity information, authorization information via roles) are stored and managed centrally, and authentication is controlled centrally as well.

**NOTE:** Role names are managed centrally, but what the roles map to still needs to be managed by each individual station. For example, an Identity Provider might tell me that my role is "Party Planner", but the station needs to have a role with that name that maps to categories, etc., on that station.

## About SAML SSO

Starting in Niagara 4.4, SAML SSO is the main supported SSO scheme. The **SAMLAAuthenticationScheme**, found in the **saml** palette, is added to a station to configure it for SAML Single Sign-On. This scheme stores configuration properties required for communications with a SAML Identity Provider (IdP). With SSO, when a user tries to access the station and has not been authenticated, the station delegates authentication to the configured IdP.

The Niagara SSO 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 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 JACE stations.

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

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

For Niagara SAML SSO the IdP must be external, located on a third-party IdP server. The SSO SAML authentication process uses an approach similar to LDAP authentication in that it also stores authentication credentials external to the station.

To configure a station for SAML SSO, in addition to the default authentication schemes (Digest and AxDigest), the station's Authentication Service must contain a properly configured SAML authentication scheme. The Authentication Service also contains SSO Configuration properties which allow you to tailor the authentication workflow as needed.

**NOTE:** In Niagara 4.4 and later, there is an added `baja-UserPrototype` component that is implemented with the User Service. This `UserPrototype` is required for SAML authentication.

More details are available in the “Components” section of this guide, see `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 is required for authentication by the IdP. Specifically, you’ll 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**.

**NOTE:** If the SAML IdP is sending the attribute `prototypeName=SAMLPrototype`, then the prototype that you create must be named, “`SAMLPrototype`”.

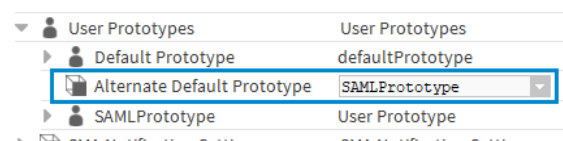
**Step 4** In the Nav tree, right-click the station and click **Save Station**.

The new `UserPrototype` is added to the dropdown list for Alternate User Prototypes.

**Step 5** Click the **Alternate Default Prototype** dropdown list, click to select your new prototype, and click **Save**.

**NOTE:** If there is a mismatch on the SAML `prototypeName` attribute value and your prototype name, the `UserService` will default to the 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 Identity Provider (IdP). You will need to obtain several configuration metadata from your IdP and use it in configuring the scheme. You will also need to provide the IdP with your station’s SP metadata. This SAML metadata is used to share configuration information between the IdP and the SP (for more details see the Prerequisites section in this topic.). The metadata is defined in XML files. 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 is required for authentication by the IdP. Typically, these values are provided by the IdP SAML Server administrator. The configuration metadata, which may be provided in an XML file, is as follows:

- HTTP-Redirect URL (corresponds to IdP Host URL, IdP Host Port, and IdP Login Path properties)
- IdP Cert

**NOTE:** 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 includes the SP "Entity ID" and the "Assertion Consumer Service". The IdP needs the metadata, which uniquely identifies the SP, and the certificate which is used to validate messages sent by the station.

**NOTE:** The Entity ID is simply a unique name that you choose as an SP, usually a URL. For example, the Entity ID typically is something like this: `https://jace.domain.com:portNumber/saml`, where you would use your JACE's hostname. Note that a port number is required. The "Assertion Consumer Service" metadata would be another URL, for example: `https://jace.domain.com:portNumber/saml/assertionConsumerService`, again using your JACE'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.

**NOTE:** This UserPrototype is required for SAML authentication.

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

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

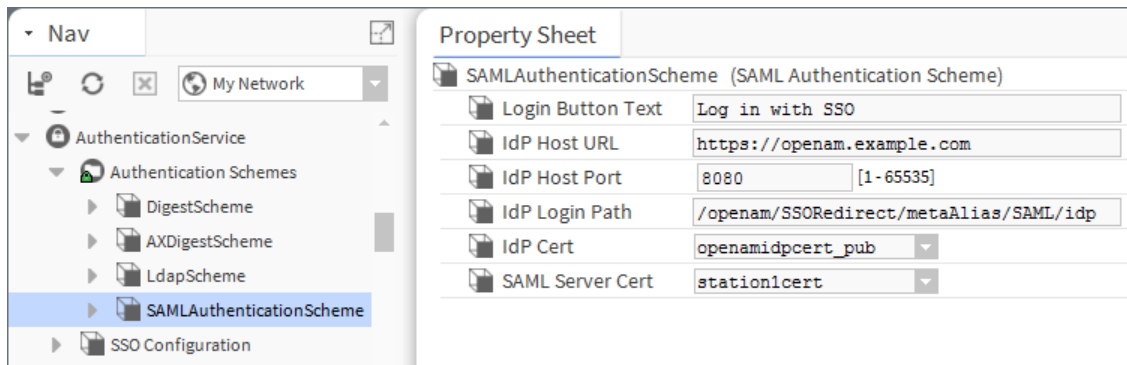
**Step 3** Expand Authentication Schemes and double-click on the SAMLAAuthenticationScheme to open a property sheet view, and enter values for the following properties:

- Login Button Text:** enter the preferred text label for the SSO login button that appears on the **Login** dialog.
- IdP Host URL:** enter the host of your Identity Provider (obtained from IdP admin).
- IdP Host Port:** enter the port number of your Identity Provider (obtained from IdP admin).
- IdP Login Path:** enter the location on the Identity Provider that you must navigate to that triggers the SAML authentication (obtained from IdP admin).
- 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).
- SAML Server Cert:** enter the certificate used by the station to sign the messages being sent back to the IdP, and is used to decrypt messages sent by the IdP.

**NOTE:** In order for the IdP to read and validate the messages sent by the station, the public certificate must be provided to the IdP SAML server administrator as well.

**Step 4** On completion click **Save**.

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



## 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, the property/attribute mappings may be customized 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.

**NOTE:** 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 "+" to expand the SAMLAttributeMapper field editor.  
An editor for a new mapping appears.
- Step 4** In the editor, replace "attributeName" with the name of the attribute sent by the SAML IdP. For example, `employeeGroup`.
- Step 5** Click on the dropdown 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.

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

- Step 6** Repeat steps 3–5 as needed to map additional attributes and click **Save** when finished.

## Logging in with SAML SSO

In Niagara 4.4 and later, SAML SSO works via a browser connection to a station. With SSO, you log into 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 into the other networked stations.

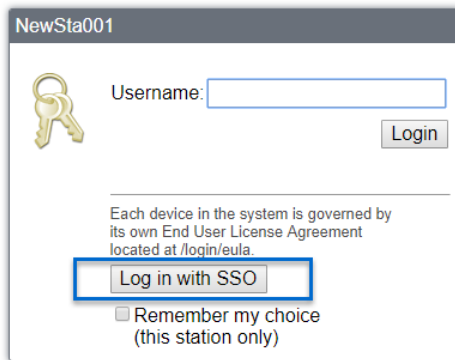
### Prerequisites:

- Your station is already configured for SSO
- You have already provided your IdP admin with any required data
- Web browser

**NOTE:** When entering the URL for the station in the browser, communications are bound by the domain specified to 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`. Note that this actually depends on the IdP requirements. Different IdPs may require different information and in a different format. For example, for the Salesforce IdP there is a field to specify the host name that you will use; and for the OpenAM IdP, you need to provide a specially formatted XML file that supplies the hostname and other data. You will need to ask the IdP admin how to provide the information.

**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** (actual button text may differ depending on the SSO scheme configuration).

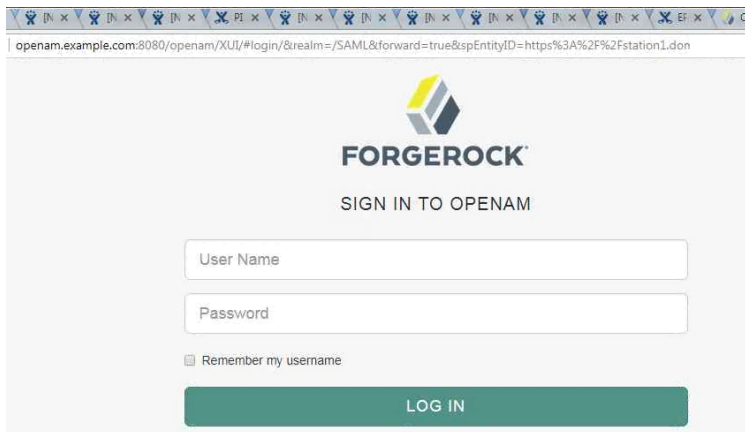


Note that the **Remember my choice** option is most useful when there are multiple SAML authentication schemes in the station. In that situation, a separate SSO Login button displays for each SSO scheme. When checked, the chosen SSO Login button is remembered and automatically used on subsequent attempts 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 checkbox simulates auto-SSO by attempting to log in with the saved scheme.

If you have already logged in with SSO, you will access this station immediately.

If this is the first time you are logging in with SSO your browser is redirected 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 into 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 try to 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.

## Network users

The `UserService` (`baja` module) and `NiagaraNetwork` (`niagaraDriver` module) permit “centralized management” of users in a multi-station system. This section provides an overview, summarizing the Challenge, User Service changes, and `NiagaraNetwork` changes.

**NOTE:** The “network user” feature is available between stations that all use the standard `UserService`. This feature is not supported between any stations using an LDAP authentication scheme (from `ldap` module). In that scenario, centralized user management depends on the LDAP or Active Directory server. For related details, refer to the *Niagara LDAP Guide*.

## Challenge

In any Niagara station database, station users are represented as individual User components, located under the station's user service. Typically, you use the **User Manager** view of this service to add, modify, and delete users. Until recent station security changes, you were able to manually copy user components from one station to other stations. However, this method no longer works, due to more secure password storage. It also never provided change coordination. If an edit is needed for such a user, the same change had to be made to that user in each (separate) station. This was an inefficient process.

**Solution:** Stations are configurable to allow users to be added, modified, or deleted in one station, and then have those changes automatically replicated (or “synchronized”) in other stations. The term “network user” applies to these users. Related are configurable user “prototypes.” When adding users, prototypes can be used in network user “strategies” between station. Note these station user changes are standard, but optional—accomplished with additional slots on existing components.

## User Service changes

Every User component (user) in the station has 2 related configuration properties: **Network User** (boolean) and **Prototype Name** (string). Currently, prototype name matters only if the user is network user, as this can be used in a “sync strategy” for distributing changes to network users.

Related to this, the `UserService` has a frozen child container slot **User Prototypes**, with a frozen child Default Prototype user component. If establishing network users, you can duplicate and edit additional user prototypes. User prototypes currently have the same properties as users, and are seen in the Nav tree and in the property sheet of the `UserService`—but are not listed in the **User Manager** view. Instead, when you add a User, the new property **Prototype Name** provides a selection list of available prototypes.

**NOTE:** In the **User Manager** view of the user service in any station, whenever you manually add a new user, property values in the **Default Prototype** are always used as defaults (regardless of whatever **Prototype Name** you may select in the **Add** dialog). In this way, the Default Prototype serves as a “template” to populate a new user’s properties (all except password).

This can simplify user management even in a “non network user” scenario, to specify typical user property settings in the UserService’s Default Prototype. For more details see “Default Prototype”.

For more details on UserService items related to network users, see:

- “Network user related properties”
- “About User prototypes”

## NiagaraNetwork changes

Each NiagaraStation (device component) under the station's NiagaraNetwork has a Users device extension, in addition to other standard device extensions like Points, Histories, Schedules, and Alarms. The Users extension contains properties that enable/configure network user synchronization “in” and “out” of this station, in relation to the station with its NiagaraNetwork. There is no special view (apart from property sheet) on this Users device extension, nor is it a container for other components.

Associated with this device extension is a view on the parent NiagaraNetwork: the **User Sync Manager**. This tabular view provides an aggregate look at all Users device extensions (one for each NiagaraStation). Each row represents a station, and columns lets you see every station's Users properties for sync configuration, sync status, sync strategy, and so on. You can select any or all rows for an edit dialog to make configuration changes, or issue manual sync commands.

**NOTE:** User synchronization requires stations (Supervisor, JACEs) to be using compatible password storage mechanisms. Otherwise, a user sync will fail. The corresponding NiagaraStation’s Users device extension will also be in fault.

**NOTE:** In Niagara 4, Sync In is supported between N4 stations, and Sync Out is supported between N4 stations (N4-to-N4). However, when syncing between N4 and AX stations, you can only Sync Out from an N4 station to an AX station (N4-to-AX). Additionally, roles are not created during the synchronization process. For any role assigned to a network user on the sending station, you must setup a matching role on the receiving station.

For more details on the “NiagaraNetwork side” of network users, see these sections in the *Niagara Drivers Guide*:

- “About the Users extension”
- “About the User Sync Manager”

## Network user related properties

Among User component properties are two that apply to the “network users” function.

Figure 5 User properties related to network user

Name	Full Name	Enabled	Expiration	Roles	Allow Concurrent Sessions	Network User	Prototype Name	Language	Authentication Scheme Name	Password	Confirm	Authenticator	Force Reset At Next Login	Email	Cell Phone Number	Time Format	Unit Conversion
NoahF	Noah Fence	true	Never	admin	true	true	HvacMgr		DigestScheme	.....	.....		false	nfence@newmetropolis.net	8005551234	(default)	None

**NOTE:** These properties are unused for any users in a station with an LDAP user service (e.g. LdapV3UserService).

**NOTE:** In Niagara 4, each user is assigned one or more roles which control that user's access to station objects, hierarchies, views, etc. During the user synchronization process a user's role assignment is sent to the receiving station however the actual role(s) is not created on the receiving station. You must setup matching roles on each receiving station.

These properties are described as follows:

Type	Value	Description
<b>Network User</b>	true, false (default)	A boolean that specifies whether this user can be made available in other stations. When set to true, this user can be synchronized with other stations. When using the <b>User Manager</b> to add new users, this typically defaults to "false" (unless the " <b>User Prototypes→Default Prototype</b> " component has been edited from defaults, where it has been set to "true". Leave the default or set to false whenever you wish this user to be local to this particular station only. For an overview of this feature, see "Network users".
<b>Prototype Name</b>	text string	Pick from a selection list showing available local User Prototypes. Blank or no selection is effectively the same as the frozen Default Prototype. Currently, this property setting applies only if the <b>Network User</b> property is "true".

## User prototypes

The importance of user prototypes are described in the following topics.

## Properties of User Prototypes

User prototypes under a station's **UserService** have the same properties as **User** components. The importance of user prototypes can be divided between the frozen "Default Prototype", and any Additional (non-default) User Prototypes you add, for example by duplicating and renaming. Currently, non-default user prototypes are only used when synchronizing network users between different stations. In this case, (identically-named) prototypes in both the source station and receiving station can be used in a "sync strategy" of "Prototype Required."

**NOTE:** In Niagara 4, each user is assigned one or more roles which control that user's access to station objects, hierarchies, views, etc. During the user synchronization process a user's role assignment is sent to the receiving station however the actual role(s) is not created on the receiving station. You must setup matching roles on each receiving station.

Note that alternative authentication schemes, such as the **Ldap Authentication Scheme**, leverage user prototypes as well. When using an alternative authentication scheme (in place of the standard **UserService**), you also create user prototypes. However, operation differs from the "network user" usage under the **UserService**. For details refer to "Setting up user prototypes" in the *Niagara LDAP Guide*.

## Default Prototype

Among User Prototypes, the "Default Prototype" is important in any station with any user service, as it is always the default source of User property values whenever you use the service's **User Manager** view to add a New user to the station.

Figure 6 Default Prototype of the **UserService**'s Default Prototypes is source of default user properties

The screenshot shows the 'Default Prototype (User)' configuration window. The breadcrumb trail at the top is: Config > Services > UserService > UserPrototypes > DefaultPrototype. The 'Property Sheet' tab is active. The configuration is organized into sections:
 

- Default Prototype (User)**: Includes 'Full Name' (text field), 'Enabled' (radio button, currently 'true'), 'Expiration' (radio buttons for 'Never Expires' and 'Expires On' with date '31-Aug-2017 11:59 PM'), 'Lock Out' (radio button, currently 'false'), 'Language' (text field), and 'Email' (text field).
- Authenticator**: Includes 'Password' and 'Confirm' (text fields), 'Force Reset At Next Login' (radio button, currently 'false'), and another 'Expiration' section with 'Never Expires' and 'Expires On' (31-Aug-2017 11:59 PM).
- Facets**: Includes 'Time Format' (dropdown, currently '(default)') and 'Unit Conversion' (dropdown, currently 'None').
- Nav File**: Includes 'Nav File' (text field, currently 'null').
- Prototype Name**: Includes 'Prototype Name' (text field).
- Network User**: Includes 'Network User' (radio button, currently 'false').
- Cell Phone Number**: Includes 'Cell Phone Number' (text field).

In this regard, all of the **Default Prototype** properties can be considered important—with one exception: password—which is not copied up to a new user created in the **User Manager**.

However, all other property values (except password) in the **Default Prototype** are used as "defaults" when you create any new user in the **User Manager**. This can be useful, for example, if you have some (minimum) collection of permissions for all users, or a typical Nav file, and so on.

**NOTE:** User Prototypes have a child “Password Configuration” container, just like User components. Inside are two properties as follows:

- **Force Reset At Next Login** - The default is “false” for any new station. If you find yourself typically changing this each time you create a new User, change it to desired value in the **Default Prototype**.
- **Expiration** - Default value is “Never expires”. Typically you leave this at default. However, it is possible you might change this to some “far future” date in the **Default Prototype**.

For more details on the operation and configuration of these two properties, see “About password expiration and reset”.

### ***Default Prototype importance in Network users scenario***

Additionally, in a multi-station job where you are using “Network users”, the user “sync strategy” of “Use Default Prototype,” the default prototype (in each station receiving network users) specifies the “local properties” that override the received properties of any network user. Note that by default only 2 properties are “local override” types: Permissions, and Nav File.

However, by going to the slot sheet of the Default Prototype (in each station receiving network users), and setting the “user defined 1” config flag, you can specify additional properties as “local override” types. You can also use this same technique with other (non-default) User Prototypes in stations that receive network users. For details, see “Specifying additional “local override” properties”.

### ***Additional (non-default) User Prototypes***

The importance of properties in an additional (non-default) User Prototypes vary among stations that are either sending or receiving network users:

- In a user-sending station (e.g. Supervisor) non-default user prototypes are important only in “name”, where you can simply duplicate the “Default Prototype” and rename each duplicate uniquely. Property values in these replicated prototypes are not used in any users—whether a user is local only to the Supervisor, or specified as a network user with this “Prototype Name.”
- In a user-receiving station (e.g. JACE) non-default user prototypes are important both in “name”, where matching prototype names in “user sending” stations provide sync strategy options, and also (by default) in two “local override” properties, described below.
  - Permissions - (either a permissions matrix of local categories and rights, or a “Super User”)
  - Nav File - (referencing a specific nav file under the local station's file structure)

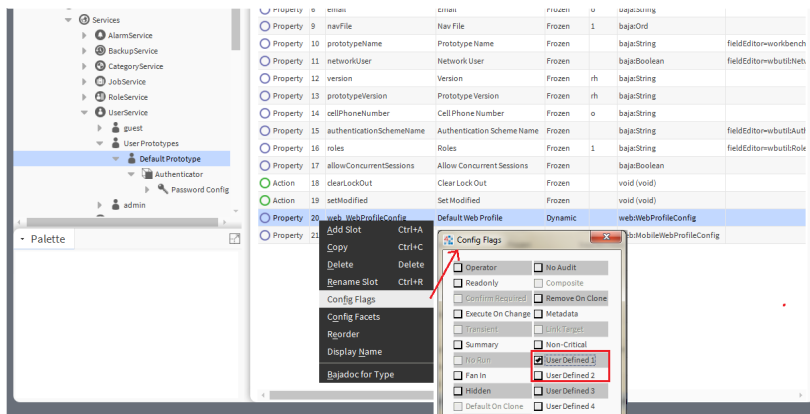
When a network user is added or modified in a “user sending” station, the two properties above are used in the “user receiving” station, instead of those same properties in the source network user. Note prototypes are configurable for other local overrides—see “Specifying additional “local override” properties”.

### ***Specifying additional “local override” properties***

In a “network user receiving” station (e.g. JACE) you can specify other properties of User Prototypes to act as “local overrides” for network users created in its station. This applies both to the single **Default Prototype**, as well as any additional (non-default) User Prototypes.

Do this from the slot sheet of the User Prototype: right-click the property, and select **Config Flags**, as shown being done for the `web_WebProfileConfig` slot (Default Web Profile) .

Figure 7 Example config flag being set in the Default Prototype of a “user receiving station”



In the Config Flags dialog, click the “User Defined 1” flag and click **OK**. Notice that in the slot sheet view, the “Flags” column now includes a “1” for that property, similar to the permissions and navFile slots. When a network user sync occurs for a user referencing this prototype, all properties with this flag use the local values as overrides.

### Naming User Prototypes

If creating additional User Prototypes (apart from the “Default Prototype”), it is recommended that you name them using descriptive text that can be logically associated with groups of station users, such as AdminHvac, GenOperations, LtgAndAlarms, and so on. You pick from these names when adding a new user and selecting a “Prototype Name.”

Keep in mind that in a multi-station job, if you choose a network “sync strategy” based upon the “Prototype Required” scheme, network users in the “user sending” station are replicated/sync’ed in the “user receiving” stations only if the UserService in each remote station has an identically named user prototype. Note there is also an alternative “Use Default Prototype” sync strategy you can use instead.

For related details, refer to the “About Users sync strategy” section in the *Drivers Guide*.

### Station-to-station users

A station-to-station user requires a machine user as opposed to a human user.

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).

**NOTE:** A station-to-station user should have only the permissions it requires. To improve system security, do not make a station-to-station user a super user.

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.

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

Users define possible connections to the station. Under the station's **Services** container, the **UserService** provides a default **User Manager** view for you to add, delete, and edit users.

**Prerequisites:** A local or remote station is open.

- Step 1 Double-click the **UserService** node in the station Nav tree.  
The **User Manager** view opens.
- Step 2 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 3 Create or modify user properties and click **Save**.

## 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

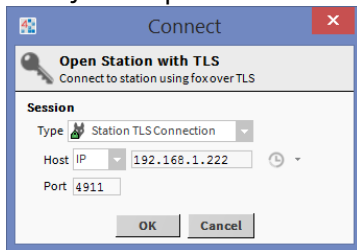
Using TLS, a secure communication session is established before the system asks for your user credentials. When you log on using the station Authentication window, the system confirms your identity, which determines your Nav tree configuration and the components you have permission to access. The system 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.

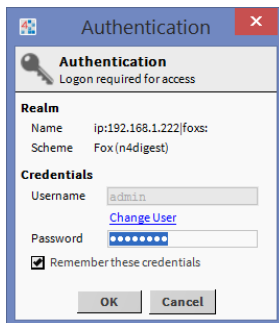
**Step 2** Enter the IP address or confirm the default address and click **OK**.

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

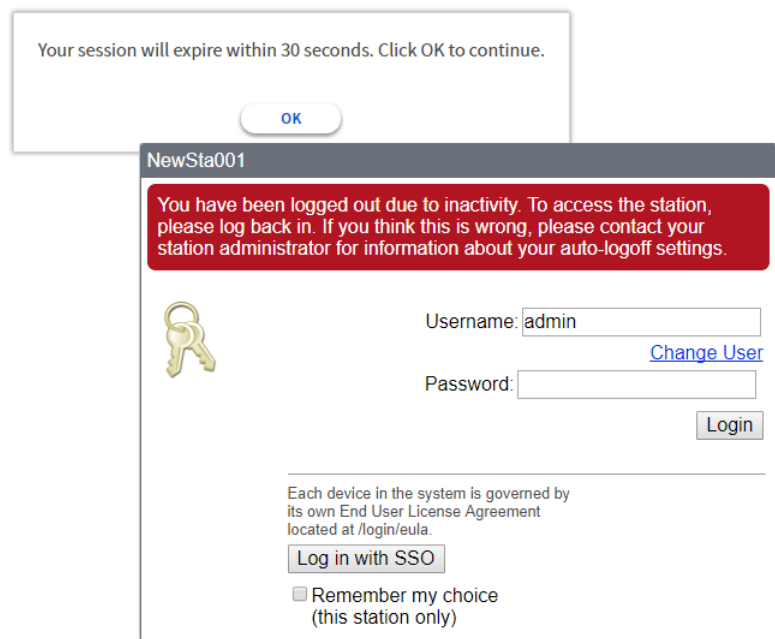
In Niagara 4.4 and later there is added support for the station auto logoff capability. Meaning that any station connection, in a web browser or in Workbench, can log the user off due to inactivity. This is important for security reasons, helping to prevent the opportunity for unauthorized access.

**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, see [baja-UserService, page 100](#).

## 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.



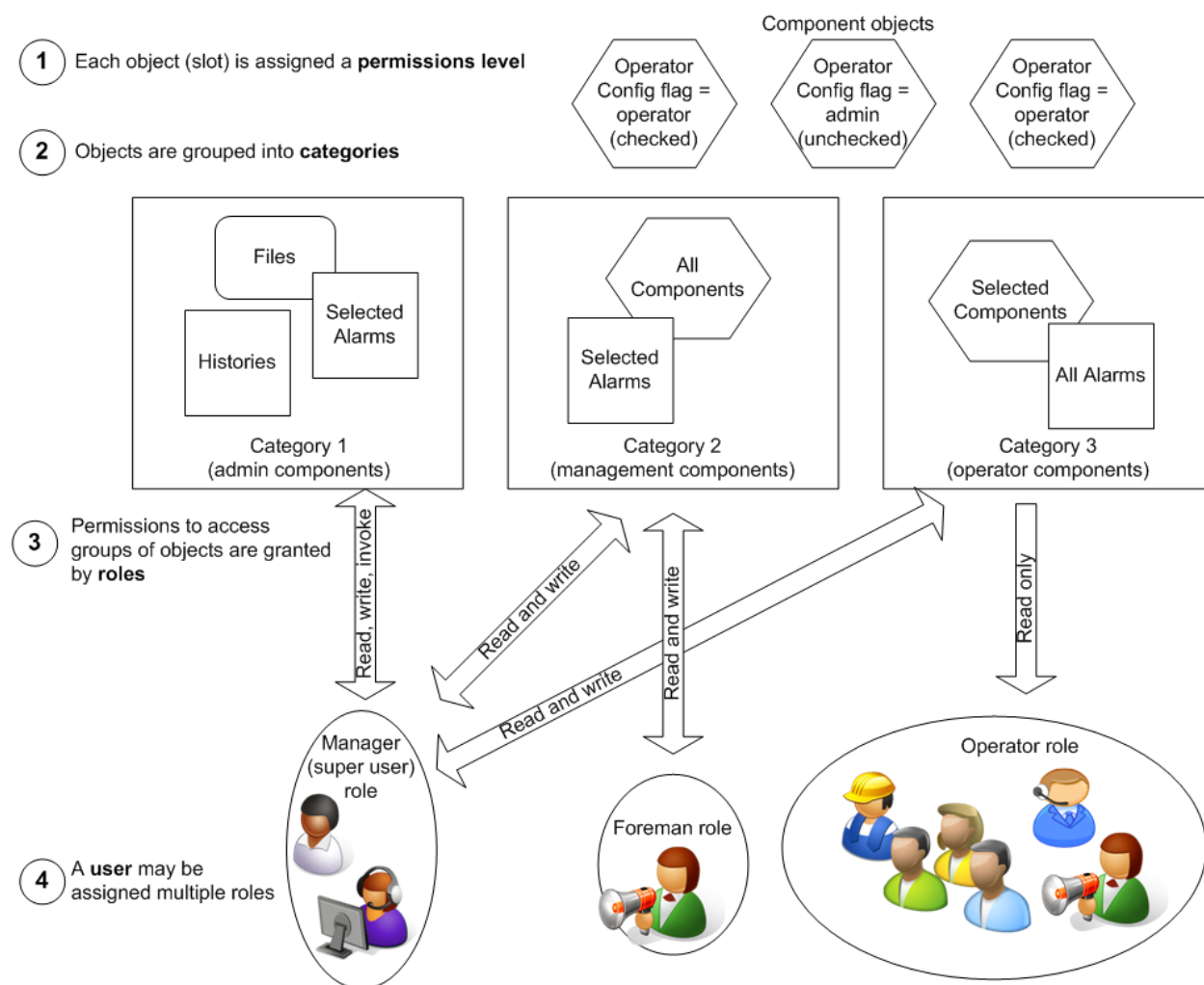
# Chapter 4 Authorization management

## 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



- 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).

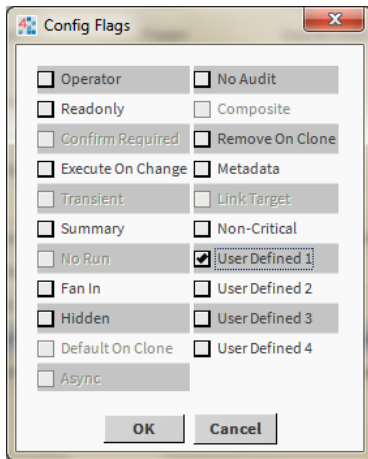
## Changing a component Config flag

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.

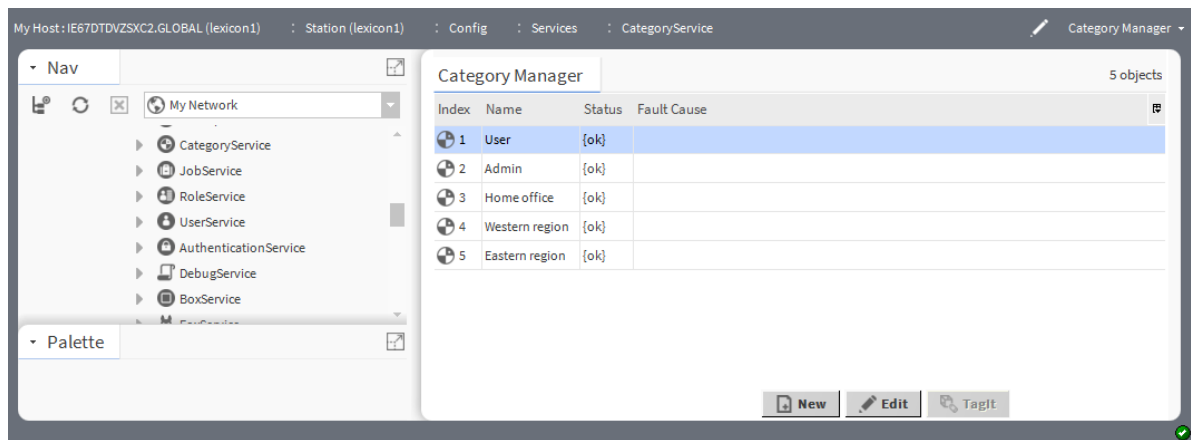
**NOTE:** Basic categories use station memory. To improve system performance, minimize the number of categories, and keep category indexes contiguous.

### 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**.

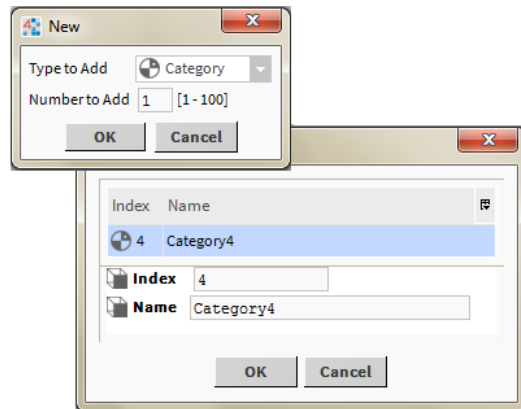
The **Category Manager** view opens.



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 categories.

Category Browser									
	Inherit	User	Admin	Operator	Viewer	Category 5	Category 6	Category 7	Category 8
Alarm	n/a		●						
Config	n/a		●						
Services	✓		●						
Drivers	✓		●						
Apps		●							
category	✓		●						
Temp1	✓		●						
Temp2	✓		●						
Alarm				●					
Ramp	✓		●						
SineWave	✓		●						
Files	n/a		●						

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.

**NOTE:** 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) is making a category mask adjustment, the user must also have at least 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:** It is important to understand the risk 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

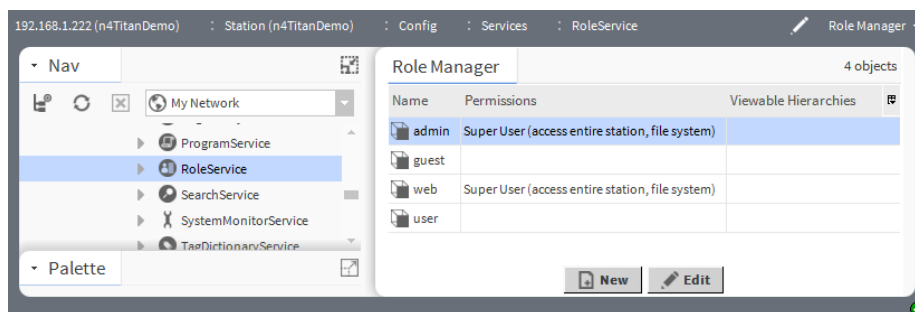
You add roles using the station's **Role Manager** view (**RoleService**).

**Prerequisites:** `Operator` config flag enabled for any restricted components. Categories 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.

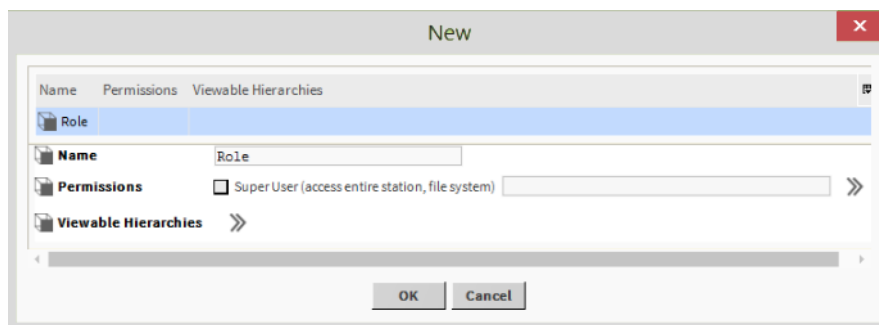
**Step 1** Right-click **RoleService** in the Nav tree, click **Views→Roll Manager**.

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 each role.

**Step 4** To configure a role as a super user, click the **Permissions** check box for **Super User**.

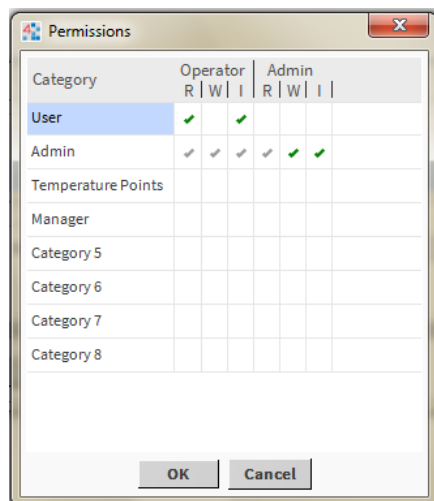
The built-in **Admin** role grants all possible rights for every category (super user). Only when logged in as the **Admin** user, or another super user, can you assign super user rights using the **Super User** check box.

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 appears.



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` 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 permissions appear in the **Permissions** property.

**Step 7** To finalize permissions, click **OK**.

## 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.

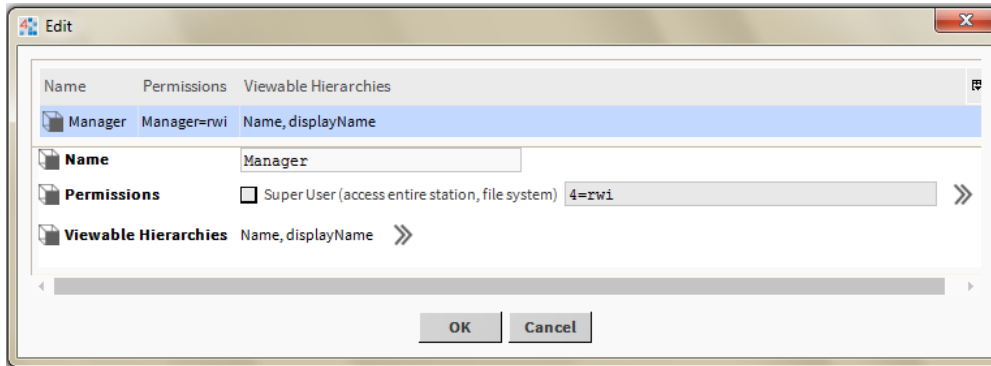
## Editing roles and permissions

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 **Ro1e 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.

Figure 10 Example of an Edit role window



- 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.

- 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 4.8 and later, there is added support for the **UserService** in the **Permissions Browser** view. Use the **Show Permissions for** dropdown 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 4.8 and later, see “wbu-til-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**

### baja-AuthenticationService

This component manages how users verify their identity to the station, using authentication schemes. Some schemes require password configuration, others do not. The **AuthenticationService** node is located in the **Services** container.

The **New Station** wizard installs two default authentication schemes:

- **DigestScheme** provides SCRAM-SHA256 (Salted Challenge Response Authentication Mechanism) technology for connecting Niagara 4 entities. Several messages are passed back and forth to prove the client knows the password.
- **AXDigestScheme** provides compatibility with stations running a previous software version.

Schemes available in the **ldap** palette include:

- **LdapScheme**
- **KerberosScheme**

Additional schemes may reside in other palettes. Developers may also create authentication schemes for special circumstances. You pick the one or two schemes you wish to use, drag them from the palette and drop them directly under the **AuthenticationService** in the Nav tree.

### baja-AuthenticationSchemeFolder

This component is a special container designed to store authentication schemes used in the station User-Service. Additional authentication schemes and authentication scheme folders may be added. This component is located in the **baja** palette.

The **AuthenticationSchemes** authentication scheme folder is a frozen slot on the **AuthenticationService** which contains the following default authentication scheme folders and schemes.

- **FoxAndWebSchemes** contains the **DigestScheme** and **AXDigestScheme**

- `WebServicesSchemes` contains the `HTTPBasicScheme`

## baja-SSOConfiguration

This component is a frozen slot on the **AuthenticationService**, used to configure Single Sign On (SSO) properties for the station. These properties allow you to enable different aspects of SSO functionality such as whether or not to automatically attempt single sign on when users log on to the station. This component is located in the **baja** palette.

Single Sign On is a method of controlling access to multiple related, but independent software systems. With SSO, a user logs in once and gains access to all networked systems without being prompted to log in again at each of them. Centrally managed credentials eliminate the opportunity for errors and using one point of authentication makes authentication less complicated and more secure.

SSO Configuration (SSO Configuration)	
Auto Attempt Single Sign On	<span style="color: red;">●</span> false
Ignore Auto SSO If User Cookie Present	<span style="color: green;">●</span> true
Display SSO Schemes On Login Page	<span style="color: green;">●</span> true
Remember My Choice Domain	

## SSOConfiguration properties

Name	Value	Description
Auto Attempt Single Sign On	true, false (default)	<p>When set to <code>true</code>, SSO is automatically attempted when logging you into the station. That is unless the user specifically visits the login or prelogin pages. Typically, when there is just one SSO scheme available you would set auto-SSO to <code>true</code>. In order to set this to <code>true</code>, there must be exactly one SSO scheme available.</p> <p>When multiple SSO schemes are present in the station this setting is automatically <code>false</code> and read only.</p>
Ignore Auto SSO If User Cookie Present	true (default), false	<p>When set to <code>true</code>, the presence of the <code>niagara_userid</code> cookie causes the user to always be redirected to the login screen, instead of automatically attempting SSO. When set to <code>false</code>, this has no effect.</p> <p>This is useful if you have certain users who need to login as station users rather than SSO users, such as admin users.</p>

Name	Value	Description
Display SSO Schemes On Login Page	true (default), false	When set to <code>true</code> , a separate login button for each SSO authentication scheme in the station displays on the login page as well as on the prelogin page. Users logging in select a scheme by clicking one of those buttons.  When using multiple SSO schemes, it is a good idea to configure the <b>Login Button Text</b> for each with a meaningful label. For example, OpenAM SSO Login.
Remember My Choice Domain	text string, null (default)	If no value in this field, logging in with SSO sets a cookie for that domain (i.e. jace1.myDomain.com) on that station only.  If a domain name is entered in the field the effect is that a user only has to login to one station to set a cookie for that domain on all networked stations.  For example, if stations all follow the pattern jace1.myDomain.com, jace2.myDomain.com, etc..., entering myDomain.com will cause a cookie for this domain to be set on all of the stations.  This is especially useful in an environment where <b>Auto Attempt Single Sign On</b> is set to <code>false</code> .

## baja-DigestScheme

This is an authentication scheme that uses SCRAM-SHA256 (Salted Challenge Response Authentication mechanism). One of the default schemes, this component is located in the **baja** palette.

When using the DigestScheme, the password is never sent across the wire. Instead, the client sends proof that they know the password.

## baja-GlobalPasswordConfiguration

These properties configure password requirements for a particular authentication scheme. You access them by expanding **Station→Config→Services→Authentication→Authentication Schemes** and double-clicking one of the schemes.

Figure 11 Global Password Configuration properties

Global Password Configuration

Global Password Configuration

Password Strength

Minimum Length: 10

Minimum Lower Case: 1

Minimum Upper Case: 1

Minimum Digits: 1

Minimum Special: 0

Expiration Interval: + 365 d 0 h 0 m 0 s

Warning Period: + 30 d 0 h 0 m 0 s

Password History Length: 0

## Scheme properties

Property	Value	Description
Password Strength	several sub-properties	See "Password strength properties," in the <i>Niagara Station Security Guide</i> .
Expiration Interval	number of days, hours, minutes and seconds	Defines the length of time from when the password is created until it is no longer valid. When this period of time expires, the system denies access.
Warning Period	number of days, hours, minutes and seconds	Defines how many days of warning a user receives prior to the expiration of the password.
Password History Length	number	Defines how many previously used passwords the system remembers.

## Password strength properties

Property	Value	Description
Minimum Length	number	Indicates the total number of characters required.
Minimum Lower Case	number	Indicates the minimum number of lower case letters required.
Minimum Upper Case	number	Indicates minimum number of upper case letters required.
Minimum Digits	number	Indicates the minimum number of digits (1, 2, 3 etc.)
Minimum Special	number	Indicates the number of special characters required. For example: ! @ # \$ % ^ , . ; etc.

## baja-AXDigestScheme

This default authentication scheme provides backward compatibility with stations running a previous software version. This component is located in the **baja** palette.

This authentication scheme provides compatibility with these : NiagaraAX versions:

- 3.5u4
- 3.6u4 and up
- 3.7u1 and up
- any 3.8 version

Earlier versions of NiagaraAX do not support the AXDigestScheme.

## baja-HTTPBasicAuthenticationScheme

This authentication scheme performs HTTP-Basic authentication using standard HTTP headers. It only works via the web, and is intended for clients that cannot use cookies. In this authentication scheme, the user name and password are sent over the connection. This component is located in the **baja** palette.

This component is located in the **baja** palette.

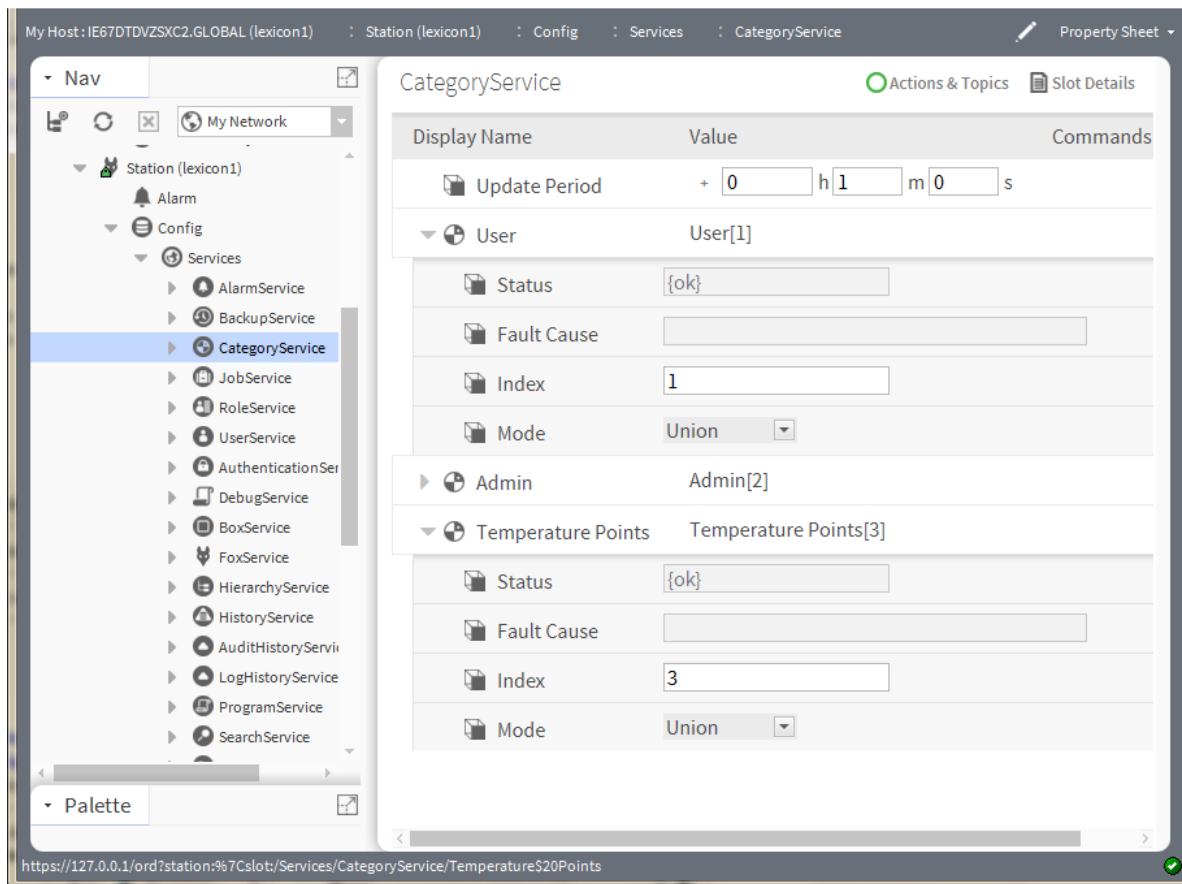
## baja-CategoryService

This service is the station container for all categories, which represent logical groupings to which you can assign components, files, and histories. It is located in a station's **Services** container.

The default view of this service, the **Category Browser**, lets you centrally assign different objects to categories, using an expandable tree view of the station. The CategoryService also provides a **Category Manager** view, for you to create, edit and delete categories. Categories play an integral role in station security, where you can give users permissions for some (or all) categories. By default, the CategoryService is included when you create a new station using the **New Station** wizard.

### Primary properties

Figure 12 CategoryService property sheet



In addition to being the container for child categories, the **CategoryService** has only one slot: **Update Period**.

Property	Value	Description
Update Period	hours minutes seconds	Sets the interval at which the system automatically assigns ancestor permissions. The default value is one (1) minute. If you assign a zero value, the system disables this feature.

## User, Admin and additional basic category properties

Property	Value	Description
Status	text	Read-only field. Indicates the condition of the component at last polling. <ul style="list-style-type: none"> <li><code>{ok}</code> indicates that the component is polling successfully.</li> <li><code>{down}</code> indicates that polling is unsuccessful, perhaps because of an incorrect property.</li> <li><code>{disabled}</code> indicates that the <b>Enable</b> property is set to <code>false</code>.</li> <li><code>fault</code> indicates another problem.</li> </ul>
Mode	drop-down list	There are two modes: Union and Intersection.
Fault Cause	text	Read-only field. Indicates why the network, component, or extension is in fault.
Index	integer	Sequential number that identifies the property in the station.

## RoleService

This service manages the Role Manager view, which is used to set up user roles in the system.

**CAUTION:** It is important to understand the risk 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.

## baja-UserPrototype

The baja-UserPrototype (an alternative to the legacy Default Prototype) was created to provide better control over how users are created and where their properties come from. This component is used only for remote authentication schemes (e.g. LDAP, Kerberos, SAML, and etc.). It is implemented with the User Service.

**NOTE:** The SAML Authentication Scheme only supports this UserPrototype. While LDAP and Kerberos support this user prototype as well the default user prototype.

Although the properties are similar to those of the Default Prototype, UserPrototype has only the relevant properties on it. Also, there is an **Overridable** user property that, when selected to `true`, prevents a value from being overwritten with a default value at next login.

Figure 13 UserPrototype properties

**UserPrototype (User Prototype)**

Full Name	User Prototype Property
Overridable	<input type="radio"/> false
value	
Enabled	User Prototype Property
Overridable	<input type="radio"/> false
value	<input checked="" type="radio"/> true
Expiration	User Prototype Property
Language	User Prototype Property
Email	User Prototype Property
Facets	User Prototype Property
Nav File	User Prototype Property
Cell Phone Number	User Prototype Property
Roles	User Prototype Property
Allow Concurrent Sessions	User Prototype Property
Auto Logoff Settings	User Prototype Property

Each of these properties have two sub-properties, as shown:

**Property Sheet**

**Engineer (User Prototype)**

Full Name	User Prototype Property
Overridable	<input type="radio"/> false
value	
Enabled	User Prototype Property
Expiration	User Prototype Property

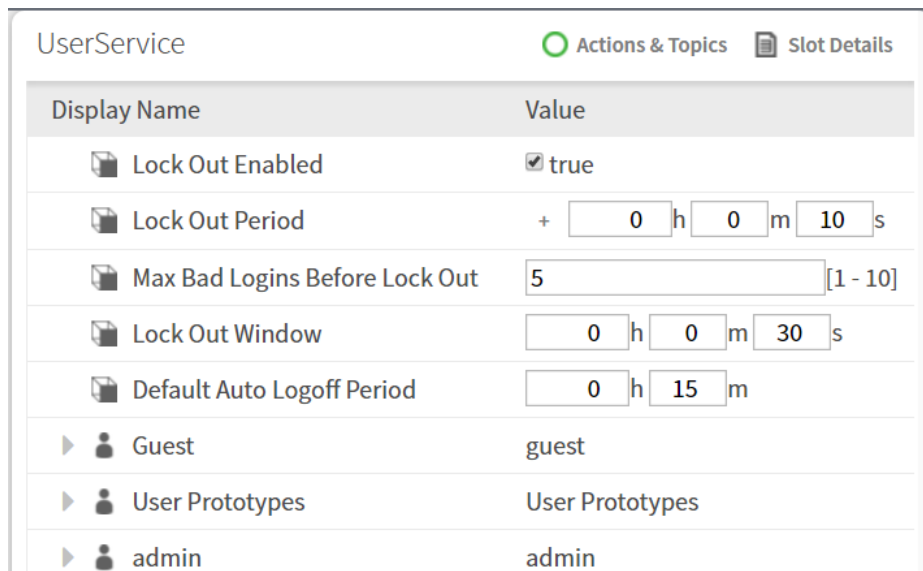
Name	Value	Description
Overridable	true, false (default)	Determines whether or not the property can be manually overriden on a user that was created from this prototype.
Value	string	Determines what the matching property on the user will be set to when creating or updating from a prototype.

## baja-UserService

This service manages all system users: human and machine. You access it by right-clicking **UserService** and clicking **Views→Property Sheet**.

The **User Manager** is the primary view of this service. By default, the UserService is included when you create a new station using the **New Station** wizard. The UserService component is available in the baja module.

Figure 14 User Service property sheet view



Display Name	Value
Lock Out Enabled	<input checked="" type="checkbox"/> true
Lock Out Period	+ <input type="text" value="0"/> h <input type="text" value="0"/> m <input type="text" value="10"/> s
Max Bad Logins Before Lock Out	<input type="text" value="5"/> [1 - 10]
Lock Out Window	<input type="text" value="0"/> h <input type="text" value="0"/> m <input type="text" value="30"/> s
Default Auto Logoff Period	<input type="text" value="0"/> h <input type="text" value="15"/> m
▶  Guest	guest
▶  User Prototypes	User Prototypes
▶  admin	admin

### Effect of property changes on user session

Starting in Niagara 4.3, any active session associated with a user will be terminated if the following changes are made 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 which has already expired.
- If the **Authentication Scheme Name** is changed.
- If the **Allow Concurrent Sessions** is set to `false`.

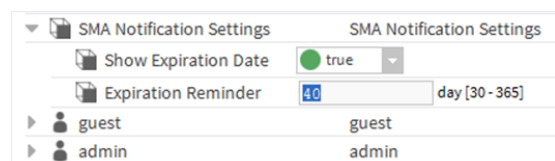
Property	Value	Description
Lock Out Enabled	true or false	If enabled ( <code>true</code> ), a number of consecutive authentication failures temporarily disables login access to the user account for the duration of the lock out period (next property). Using lock out makes it difficult to automate the guessing of passwords. <b>NOTE:</b> Each user has a Clear Lock Out action.
Lock Out Period	hours minutes seconds (default is 10 seconds)	If lock out is enabled, this property defines the period of time a user account is locked out before being reset. While locked out, any login attempt (even a valid one) is unsuccessful. <b>NOTE:</b> The default Lock Out value guards against an automated, brute-force password attack, where a computer application issues hundreds of login attempts a second. The 10 second latency thwarts such an attack, as the attacker must wait 10 seconds after each five unsuccessful login attempts. If deemed necessary, you can adjust this value to guard against human attack.
Max Bad Logins Before Lock Out	number from 1 — 10 (default is 5)	If lock out is enabled in conjunction with the <b>Lock Out Window</b> , this property specifies the number of consecutive failed user login attempts that trigger a lock out after a window of time.

Property	Value	Description
Lock Out Window	hours minutes seconds (default is 30 seconds)	<p>If lock out is enabled, and a user fails to log in successfully before the <b>Max Bad Logins Before Lock Out</b> window (period) expires, the user is locked out for the <b>Lock Out Period</b> duration.</p> <p>The system enforces changes to lock out properties the next time the user logs in. For example, if <b>Max Bad Logins Before Lock Out</b> is set to 5, user ScottF fails to log in four times within the <b>Lock Out Window</b>, and an admin-level user changes <b>Max Bad Logins Before Lock Out</b> to 3, the change does not lock ScottF out. User ScottF still has one more chance to log in before getting locked out.</p> <p>If ScottF's fifth attempt to log in fails, the system locks him out the next time he attempts to log in because five failed attempts is greater than or equal to the <b>Max Bad Logins Before Lock Out</b> of 3.</p>
Default Auto Log-off Period	0000h 15m (default)	Specifies the amount of time that a period of inactivity may last before a station connection is automatically disconnected. The acceptable range of values is two minutes to four hours. This limit is observed only when the User's <b>Use Default Auto Logoff Period</b> property is set to <code>true</code> .
SMA Notification Settings	multiple properties	See the next section
User Prototypes	multiple properties	See the next section.

## SMA Notification Settings

These are the properties to configure the Software Maintenance Agreement (SMA) expiration reminder. For details, see the *Niagara Platform Guide*.

Figure 15 SMA expiration reminder properties



Type	Value	Description
Show Expiration Date	true (default), false	<p>When set to <code>true</code>, the initial SMA expiration reminder displays in the browser-connected station <b>Login</b> window at 45 days prior to expiry. When set to <code>false</code>, the initial SMA expiration reminder is hidden, it does not display.</p> <p><b>NOTE:</b> Once the SMA expires, the SMA expiration reminder cannot be hidden.</p>
Expiration Reminder	45 day (default) [30–365 days]	By default, this is set to 45 days before expiration.

## User Prototypes

UserPrototypes is a frozen slot on a station's UserService. It contains a frozen **Default Prototype** (properties shown in the following figure) to support centralized users in the station's NiagaraNetwork, as well as a

frozen Alternate Default Prototype for any additional user prototypes needed to support remote authentication schemes such as LDAP, Kerberos, and SAML.

## Default Prototype

The User Service and Niagara Network still use the existing default user prototype functionality and there are no current plans to migrate them to the new UserPrototype. Also, LDAP and Kerberos will continue to support the default user prototypes.

## Alternate Default Prototype

This property allows you to specify an alternate user prototype to start with when creating a new user (instead of only using defaultPrototype). For example, you would use this to select a prototype (other than default prototype) specifically created to support a remote authentication scheme, such as SAML.

Figure 16 Default Prototype user properties

Property	Value	Description
Full Name	text	The user's name.
Enabled	true (default) or false	Unchecked (false) disables this user. Disabled users cannot access the system.
Expiration	radio buttons	<ul style="list-style-type: none"> <li>Never Expires permits this user to always log in.</li> <li>Expires On [date and time] allows this user to log in until the expiration date and time.</li> </ul>
Lock Out	true or false (default)	Checked enables a user to log in. Unchecked (true) prohibits this user from logging in.

Property	Value	Description
Language	two lower-case letters	Defines the ISO 639 language code. For a list of codes, see the following: <a href="http://www.loc.gov/standards/iso639-2/langcodes.html">http://www.loc.gov/standards/iso639-2/langcodes.html</a> .
Email	email address	Defines the user's email address.
Authenticator	additional properties	Manages user password.
Facets	timeFormat and unitConversion	Configures the time format and units to use when this user logs in to the system.
Nav File	file:^nav/Nav-File.nav	Identifies the file to use for displaying a customized navigation tree.
Prototype Name	text	Identifies the name of the prototype used to create this user.
Network User	true or false (default)	When true, this user account can be synchronized to other stations on the network.
Cell Phone Number	number	Defines the user's mobile phone number.
Authentication Scheme Name	drop-down list	Identifies the authentication scheme used to verify this user.
Roles	radio buttons	Identifies the user's role.
Allow Concurrent Sessions	true (default) or false	When checked, allows multiple sessions. When unchecked, a new session invalidates the old session.
Auto Logoff Settings	additional properties	Refer to the below section in this document.
Default Web Profile	additional properties	Refer to the below section in this document.
Mobile Web Profile	additional properties	Refer to the below section in this document.

## Auto Logoff Settings

Property	Value	Description
Auto Logoff Enabled	true (default) or false	<p>When <code>true</code>, a station connection (via Workbench or web browser) automatically disconnects if a period of inactivity exceeds the amount of time specified for the <b>Default Auto Logoff Period</b> (in the <code>UserService</code>).</p> <p>When <code>false</code>, the station does not automatically terminate a user's session due to inactivity.</p> <p><b>NOTE:</b> Separate auto logoff options exist for Workbench which functions independently of these station settings. For details, see the <i>Getting Started with Niagara</i>.</p>
Use Default Auto Logoff Period	true (default) or false	<p>If the property is set to <code>true</code>, the <b>Default Auto Logoff Period</b> (configured in the <code>UserService</code>) displays the specified time.</p> <p>When <code>false</code>, the specified <b>Default Auto Logoff Period</b> is not observed. Instead, use the <b>Auto Logoff Period</b> property to set a different auto logoff time period.</p>
Auto Logoff Period	0h 15m (default) (2-4 hours)	<p>When <b>Use Default Auto Logoff Period</b> property is set to <code>false</code>, use this property to specify a different time period . The range is 2 minutes to 4 hours.</p> <p>Otherwise, this property is read only, showing the value specified in the <code>UserService</code>'s <b>Default Auto Logoff Period</b>.</p>

## Authenticator—Password Config

Property	Value	Description
Force Reset at Next Login	true or false (default)	Requests that the user create a new password the next time they log in.
Expiration	radio button	Configures a password change for a specific date and time.

## Default Web Profile

These properties configure the information available to a specific user who accesses the system through a browser. By default all information is visible.

Property	Value	Description
Type Spec (type of profile)	drop-down list	<p>Identifies the type of profile:</p> <ul style="list-style-type: none"> <li>• hx (default)</li> <li>• axvelocity</li> <li>• mobile</li> <li>• Workbench</li> </ul>
Type Spec (type of profile)	drop-down list	<p>Identifies the type of profile:</p> <ul style="list-style-type: none"> <li>• BasicHxProfile</li> <li>• DefaultHxProfile</li> <li>• HTML5HxProfile (default)</li> <li>• HandHeldHxProfile</li> </ul>

Property	Value	Description
Hx Theme	drop-down list	Selects the look of the user interface: <ul style="list-style-type: none"> <li>• Zebra</li> <li>• Lucid</li> </ul>
Enable Hx Workbench Views	yes (default)	Removing the check mark disables the views.
Enable Nav Tree Side Bar	yes (default)	Removing the check mark disables the Nav tree.
Enable Search Side Bar	yes (default)	Removing the check mark disables the use of the search side bar.
Enable Palette Side Bar	yes (default)	Removing the check mark disables the use of the palette side bar.
Enable Nav File Tree	yes (default)	Removing the check mark disables the Nav tree.
Enable Config Tree	yes (default)	Removing the check mark disables the Config tree.
Enable Files Tree	yes (default)	Removing the check mark disables the Files tree.
Enable Histories Tree	yes (default)	Removing the check mark disables the Histories tree.
Enable Hierarchies Tree	yes (default)	Removing the check mark disables the Hierarchies tree.

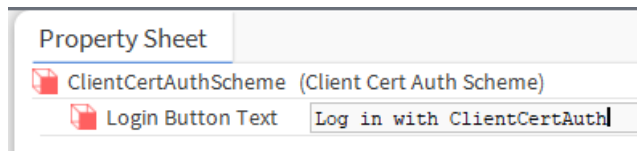
### Mobile Web Profile

Property	Value	Description
Type Spec (type of profile)	drop-down list	Identifies the type of profile: <ul style="list-style-type: none"> <li>• Hx</li> <li>• Mobile</li> </ul>
Type Spec (type of profile)	drop-down list	Identifies the type of profile: <ul style="list-style-type: none"> <li>• BasicHxProfile</li> <li>• DefaultHxProfile</li> <li>• HTML5HxProfile (default)</li> <li>• HandHeldHxProfile</li> </ul>
Mobile Nav File	ord	Identifies the location of the file that defines the mobile nav tree for this user.
Hx Theme	drop-down list	Selects the look of the user interface: <ul style="list-style-type: none"> <li>• Zebra</li> <li>• Lucid</li> </ul>

### clientCertAuth-ClientCertAuthScheme

In Niagara 4.8 and later, Client Certificate Authentication is 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.

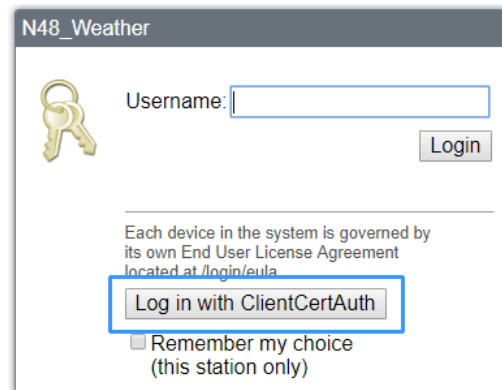
With this scheme, each User object has an authenticator containing a public certificate (which matches the user certificate's private key). Additionally, each certificate must be added to the server socket's **TrustAnchor** list. During a login attempt, the user is prompted to upload the certificate. 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.

**NOTE:** In the example shown, the User object is configured to use clientCertAuthScheme authentication, and to reset his/her authentication scheme on login (via the User **Force Scheme Reset To** property).

Figure 17 Example configured button visible in login window

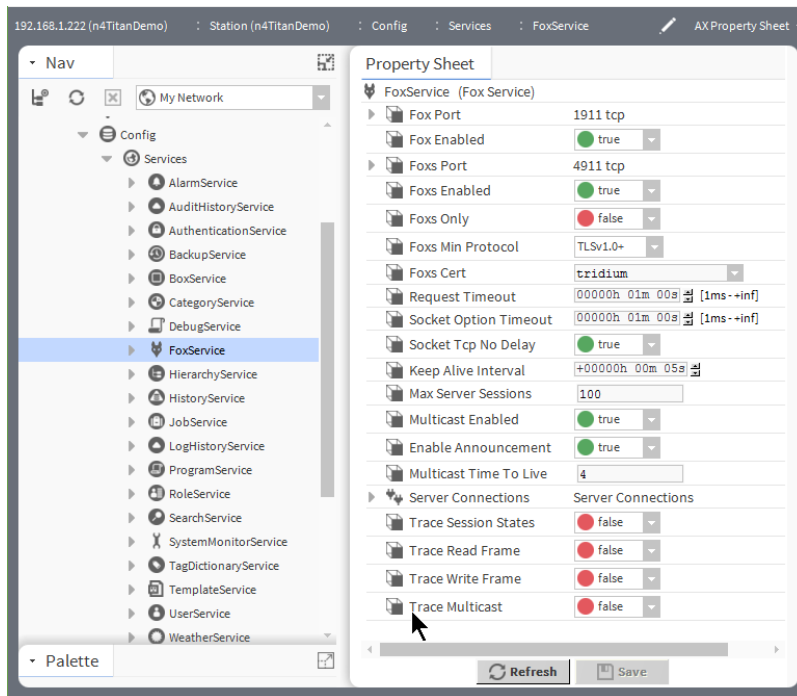


For additional information, see Admin/User workflow for client certificate authentication in the "User Authentication" chapter.

## FoxService

To view these properties expand the **Services** folder in the Nav tree, right-click **FoxService**→**Views**→**Property Sheet**.

Figure 18 Example of FoxService properties



Type	Value	Description
Fox Port	1911 (default) Tcp (default)	Public Server Port specifies the port number for standard Fox communication.  Ip Protocol specifies the protocol.
Fox Enabled	true (default) or false	When set to <code>true</code> , turns on a standard Fox connection (no communication encryption) using port 1911. When enabled, <b>Http Enabled</b> in the <b>WebService</b> must also be set to <code>true</code> (for wbapplet use).  When set to <code>false</code> , turns off the standard Fox connection causing the system to ignore attempts to connect using Fox port 1911. If <b>Foxs Only</b> is enabled, this setting ( <code>false</code> for <b>Fox Enabled</b> ) is irrelevant.
Foxs Port	4911 (default) Tcp (default)	Public Server Port specifies the port number for standard Fox communication.  Ip Protocol specifies the protocol.
Foxs Enabled	true (default) or false	When set to <code>true</code> , turns on secure Fox communication using port 4911. When enabled, <b>https Enabled</b> in the <b>WebService</b> must also be set to <code>true</code> (for webapplet use).  When set to <code>false</code> , turns off the secure Fox connection causing the system to ignore attempts to connect using Foxs port 4911.
Foxs Only	true or false (default)	When set to <code>true</code> redirects any attempt to connect using a connection that is not secure (Fox alone) to Foxs.  When set to <code>false</code> , does not redirect attempts to connect using Fox alone.

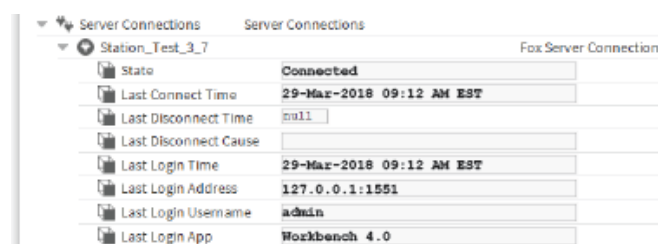
Type	Value	Description
Foxs Min Protocol	drop-down list TLSv1.0+ (default) TLSv1.1+ TLSv1.2	The minimum level of the TLS (Transport Layer Security) protocol to which the server will accept negotiation. The default includes versions 1.0, 1.1 and 1.2. It works with most clients, providing greater flexibility than an individual version.  During the handshake, the server and client agree on which protocol to use.  Change <b>Protocol</b> from the default if your network requires a specific version or if a future vulnerability is found in one of the versions.
Foxs Cert	drop-down list of server certificates; defaults to tridium	Specifies the alias of the host platform's server certificate, which the client uses to validate server authenticity. The default identifies a self-signed certificate that is automatically created when you initially log in to the server. If other certificates are in the host platform's key store, you can select them from the drop-down list.
Request Timeout	hours, minutes, seconds (default: 1 minute)	Specifies the time to wait for a response before assuming a connection is dead.
Socket Option Timeout	hours, minutes, seconds (default: 1 minute)	Specifies the time to wait on a socket read before assuming the connection is dead.
Socket Tcp No Delay	true (default) or false	Used to disable Nagle's algorithm, which may cause issues with delayed acknowledgements that occur in TCP socket communications between Fox clients and servers. The default disables Nagle's algorithm.  In Workbench, you can enter this line in the <code>system.properties</code> file to adjust this setting: <code>niagara.fox.tcpNoDelay=true</code> .
Keep Alive Interval	hours, minutes, seconds (default: 5 seconds)	Sets the amount of time between messages, which indicate that the connection is alive. This value should be well below the request timeout and socket option timeout.
Max Server Sessions	number (default: 100)	Sets the maximum number of Fox/Foxs server connections before additional client connections error with busy.
Multicast Enabled	true (default) or false	Allows the station to initiate UDP (User Datagram Protocol) multicasting. This feature is necessary for a discovery from this station. This type of multicasting differs from Workbench UDP multicast support, which can be optionally disabled via an entry in the Workbench host's <code>system.properties</code> file.
Enable Announcement	true (default) or false	Turns on support for UDP multicast announcement messages received by the station in support of learn/discover.
Multicast Time to Live	number (default: 4)	Specifies the number of hops to make before a multicast message expires.
Server Connections		See <a href="#">Server Connections, page 105</a>
Trace Session States	true or false (default)	Debug usage for tracing session state changes.

Type	Value	Description
Trace Read Frame	true or false (default)	Debug usage for dumping frames being read from the wire.
Trace Write Frame	true or false (default)	Debug usage for dumping frames being written to the wire.
Trace Multicast	true or false (default)	Debug usage for tracing multicast messaging.

## Server Connections

These properties provide status information about current Workbench client connections to the local station. They do not reflect station-to-station Fox connections. Each connection provides the same set of properties.

Figure 19 Example of Server Connections properties



Property	Value	Description
State	Not connected, Connected	Reports the current status of the connection.
Last Connect Time	hours, minutes, seconds	Reports the last time the client successfully connected to the server.
Last Disconnect Time	hours, minutes, seconds	Reports when the client last disconnected from the server.
Last Disconnect Cause	text	Provides a brief explanation.
Last Login Time	hours, minutes, seconds	Reports the last time a client logged in to the server.
Last Login Address	IP address	Identifies the platform.
Last Login Username	text	Identifies the user who made the connection.
Last Login App	text	Identifies the version of Workbench.

## nss-SecurityService

The SecurityService, available in the **nss** palette in Niagara 4.6 and later, provides the ability to monitor certificates and generate alarms for those that are about to expire. Other station services with configurable security settings report to the SecurityService.

**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 note that uninstalling the nss modules causes an auto restart of the station.

The **Security Dashboard** is the main view for the Security Service. For additional details on the view, see “[nss-SecurityDashboardView, page 114](#)” in the “Components and views” chapter of this guide.

Figure 20 SecurityService properties

## SecurityService properties

In addition to the standard properties (Status, Fault Cause, and Enabled), the following configuration properties are present.

Name	Value	Description
Certificates	additional properties	Container for certificates in use in the station. For each certificate listed the following read-only data is shown.
Certificate Info		Displays the certificate name/Alias
Expiry		Displays the number of days until/since the certificate expiry date.
Used In		Displays the name and ORD of the Service using the certificate.
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 <code>.bog</code> 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 <code>.bog</code> file. On station restart, the system dashboard data will not be available until a dashboard refresh is triggered (via the action on the service).
Station Link Config	Display remote station dashboard (default), Display local view of the remote station dashboard view	This property determines if the links on the SYSTEM dashboard link directly to the remote station, or if they link to the NiagaraNetwork station in the supervisor. In most cases, it is a better experience to link directly to the station, unless you know that the user will have access to those stations from the browser, or if you are concerned about the user having to log in to multiple stations. The SSO feature can improve that experience.

## Actions

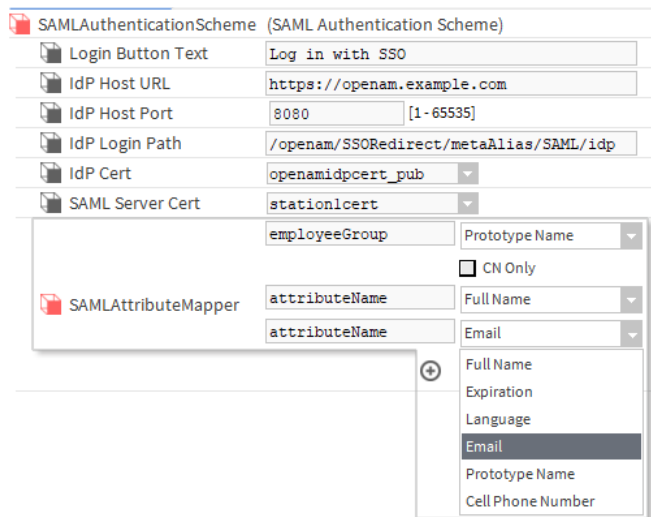
**Refresh System Dashboard Data** — this action fetches the Security dDashboard data from all remote stations (or all stations matching the provided filter).

## saml-SAMLAttributeMapper

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 customized name. However, other IdPs may not be configurable, or IT restrictions may prevent configuring an IdP that supports this feature. In this case, you can configure the SAML Authentication Scheme to map specific SAML attributes to properties in the User Prototype.

This is done by dragging the SAMLAttributeMapper from the **saml** palette to the SAMLAuthenticationScheme.



**NOTE:** 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.

**NOTE:** In some cases an IdP sends back multiple values for the prototypeName attribute. After the following patches if the IdP sends back multiple prototypeNames, the SAMLAuthenticationScheme considers all returned values and extracts the one that appears highest on the list of UserPrototypes (similar to how LDAP works).

- For Niagara 4.4U1:
  - **saml-rt-4.4.93.40.1**
  - **saml-wb-4.4.93.40.1**
- For Niagara 4.6:
  - **saml-rt-4.6.96.28.2**
  - **saml-wb-4.6.96.28.1**

### User properties that can be mapped from SAML attributes

The following User properties can be mapped:

- Full Name
- Expiration
- Language
- Email
- Prototype Name
- Cell PhoneNumber

All other properties will be acquired from the UserPrototype associated with the user.

## Default mappings

If no mappings are specified on the SAMLAuthenticationScheme, the following mappings will be 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 <b>CN Only</b> 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.

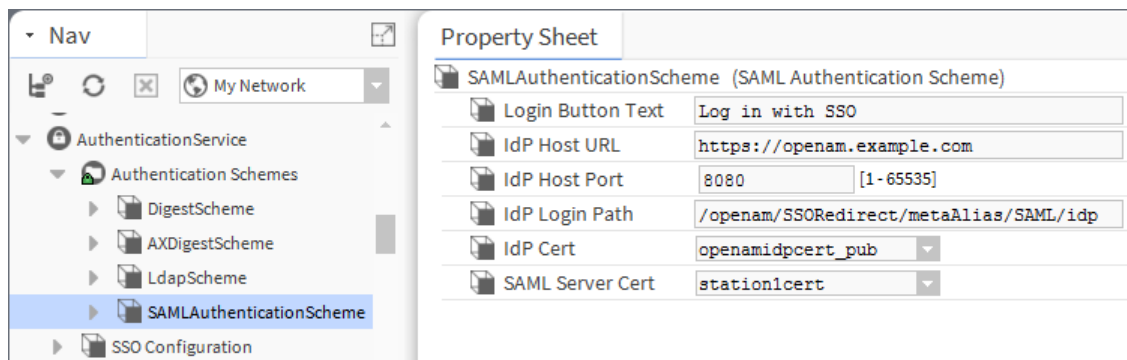
## saml-SAMLAuthenticationScheme

The SAML Authentication Scheme extends the SSO authentication scheme. SAML SSO is enabled by adding a SAML Authentication Scheme to the station. The scheme must be configured with a number of IdP configuration values, typically, these are obtained from the IdP SAML Server administrator.

Additionally, 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 4.8 and later, 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 (i.e. OpenAM, Salesforce, etc.). In the example shown here, the authentication scheme is configured for the OpenAM Identity Provider.



## SAML Authentication Scheme properties

Type	Value	Description
Login Button Text	text string, "Log in with SSO" (default)	The preferred text label for the SSO login button that appears on the <b>Login</b> 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)	IdP provided data, this is the URL for the host of your Identity Provider.
IdP Host Port	443	IdP provided data, this is the port number of your Identity Provider
IdP Host Login Path	/path/to/login	IdP provided data, this is the location on the Identity Provider that you must navigate to trigger SAML authentication.
IdP Cert	drop-down list	IdP provided data, this is the certificate required to encrypt messages sent to the IdP, and validate messages sent from the IdP.
SAML Server Cert	drop-down list	This is the certificate used by the station to sign messages that are sent back to the IdP. Note that this certificate is also provided to the IdP SAML Server admin so that the IdP can read and validate the messages. It is also used to decrypt messages sent from the IdP to the station.

## samlEncryption-SamlXmlDecrypter

In Niagara 4.4U1 and later, there is added support for 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**.

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) Cert.

This component is available in the samlEncryption-rt module. You will need the latest saml-rt and samlEncryption-rt patches for Niagara 4.4U1, Niagara 4.4U2, Niagara 4.6, and Niagara 4.7.

## web-WebService

This service encapsulates access to the HTTP server as well as the servlet infrastructure used to expose custom applications over HTTP. The WebService is available in the **web** palette. It is also one of the default services in a station created by using the **New Station** tool. Only one WebService is supported in a station.

Figure 21 Example of Webservice properties

The screenshot displays the 'WebService (Web Service)' configuration window. The left sidebar shows a tree view of services under 'Config' and a palette of email-related components. The main area, titled 'Property Sheet', lists the following properties and their current values:

- Status: {ok}
- Fault Cause: (empty text field)
- Enabled: true
- Http Port: 80 tcp
- Http Enabled: false
- Https Port: 443 tcp
- Https Enabled: true
- Https Only: true
- Https Min Protocol: TLSv1.0+
- Https Cert: tridium
- Require Https For Passwords: true
- X Frame Options: Sameorigin
- Remember User Id Cookie: true
- Login Template: null
- Log File Directory: file:^^webLogs
- Client Environments: Client Environments
- Show Stack Trace: false
- Load JxBrowser from Cloud (Beta): Never
- Applet Module Caching Type: Host
- Web Start Config: Web Start Config
- Cache Config: Cache Config
- JettyWebServer: Jetty Web Server (started)
- User Data Storage: User Data Config

At the bottom right of the property sheet, there are 'Refresh' and 'Save' buttons.

Name	Value	Description
Status	text	Read-only field. Indicates the condition of the component at last polling. <ul style="list-style-type: none"> <li>{ok} indicates that the component is polling successfully.</li> <li>{down} indicates that polling is unsuccessful, perhaps because of an incorrect property.</li> <li>{disabled} indicates that the <b>Enable</b> property is set to false.</li> <li>fault indicates another problem.</li> </ul>
Fault Cause	text	Read-only field. Indicates why the network, component, or extension is in fault.
Enabled	true or false	Activates and deactivates use of the function.
Http Port	80 (default)	Specifies the TCP port the service listens on for HTTP client connections, where port 80 is the default.
Http Enabled	true (default) or false	Determines if HTTP client requests are processed. When set to true, turns on a standard Http connection (no communication security) using port 80. When enabled, <b>Fox Enabled</b> in the <b>FoxService</b> must also be set to true (for wbapplet use).

Name	Value	Description
		When set to <code>false</code> , turns off the standard Http connection causing the system to ignore any attempts to connect using Http port 80. If <b>Https Only</b> is enabled, this setting ( <code>false</code> for <b>Http Enabled</b> ) is irrelevant.
Https Port	443 (default)	Specifies the TCP port the service listens on for HTTPS (secure) client connections, where port 443 is the default.
Https Enabled	<code>true</code> (default) or <code>false</code>	Determines if HTTPS client requests are processed. When set to <code>true</code> , turns on secure Http communication using port 443. When enabled, <b>Foxs Enabled</b> in the <b>FoxService</b> must also be set to <code>true</code> (for webapplet use).  When set to <code>false</code> , turns off the secure Https connection causing the system to ignore any attempts to connect using Https port 443.
Https only	<code>true</code> (default) or <code>false</code>	When set to <code>true</code> redirects any attempt to connect using a connection that is not secure (Http alone) to Https.  When set to <code>false</code> , does not redirect attempts to connect using Http alone.
Https Min Protocol	drop-down list TLSv1.0+ (default) TLSv1.1+ TLSv1.2	The minimum level of the TLS (Transport Layer Security) protocol to which the server will accept negotiation. The default includes versions 1.0, 1.1 and 1.2. It works with most clients, providing greater flexibility than an individual version.  During the handshake, the server and client agree on which protocol to use.  Change <b>Protocol</b> from the default if your network requires a specific version or if a future vulnerability is found in one of the versions.
Https Cert	drop-down list of server certificates; defaults to <code>tridium</code>	Specifies the alias of the host platform's server certificate, which the client uses to validate server authenticity. The default identifies a self-signed certificate that is automatically created when you initially log in to the server. If other certificates are in the host platform's key store, you can select them from the drop-down list.
Require Https For Passwords	<code>true</code> (default) or <code>false</code>	When set to <code>true</code> , the <b>HTTPSs Enabled</b> property also must be set to <code>true</code> , or the system disables the <b>New</b> button (for creating a new user in the <b>UserService</b> ). This prevents the creation of a password for a new user across a connection that is not secure.  When set to <code>false</code> , the <b>New</b> button (for creating a new user in the <b>UserService</b> ) remains enabled even if <b>HTTPSs Enabled</b> is <code>false</code> . This combination of settings creates a security vulnerability when creating passwords for new users and is not recommended.
X Frame Options	<code>Sameorigin</code> (default) <code>Deny</code> or <code>Any</code> (least secure)	The X-Frame-Options HTTP response header can be used to indicate whether or not a browser should be allowed to render a page in a <code>&lt;frame&gt;</code> or <code>&lt;iframe&gt;</code> . You can use this to avoid clickjacking attacks, by ensuring that content is not embedded into other sites.

Name	Value	Description
		<p>If you specify <code>Sameorigin</code>, the page will load in a frame as long as the site including it in a frame is the same as the one serving the page (same server). If a page specifies <code>Sameorigin</code>, browsers will forbid framing only if the top-level origin FQDN (fully-qualified-domain-name) does not exactly match FQDN of the subframe page that demanded the <code>Sameorigin</code> restriction. This is considered a safe practice.</p> <p>If you specify the <code>Deny</code>, attempts to load the page in a frame will always fail.</p> <p><b>NOTE:</b> The <code>Deny</code> option inhibits display of some typical Shell Hx Profile views.</p> <p><b>CAUTION:</b></p> <p>If you specify the <code>Any</code> option then Cross-Frame Scripting (SFS) and Cross-Site Scriptin (XSS) are allowed.</p>
Login Template	<code>null</code>	When <code>Any</code> is selected, no custom login template is used. When <code>Any</code> is not selected, the option list shows available custom login templates that you can select for a station login page.
Log File directory	<code>filepath</code>	Default is <code>file:^^webLogs</code> . The folder in the station's file space in which log files are stored. Log file names use a <code>YYMMDD.log</code> (date) convention, such as <code>230501.log</code> for a file created May 1, 2023.
Client Environments	<code>additional properties</code>	This property is a container for Mobile Client Environment (mobile) entries, available if the station's host is licensed with the mobile feature. It is used in detection of a user's browser type (e.g. desktop or mobile) and the selection of the appropriate <code>webProfile</code> for that user. See details below. Also, refer to the <i>Niagara Mobile Guide</i> .
Show Stack Trace	<code>true or false</code> (default)	Shows the stack trace, if set to <code>true</code> .
Load JxBrowser from Cloud	<code>drop-down list</code>	Loads the JxBrowser from the cloud.
Applet Module Caching Type	<code>Host (default) or User</code>	<p>The default option, <code>Host</code>, results in a folder and the downloading of installation modules to the <code>module</code> folder (<code>n4applet</code> for N4, and <code>wbapplet</code> for AX). This results in the creation of multiple folders of downloaded modules, which negatively affects platform memory usage.</p> <p>The <code>User</code> option results in the creation of a <code>.sharedModuleCache</code> folder. The system then downloads to a sub-folder at this location (<code>n4applet</code> for N4, and <code>wbapplet</code> for AX). This option minimizes the memory required when running in JACE.</p>
Web Start Config	<code>additional properties</code>	Container for several subproperties used to configure aspects of Niagara Java Web Start which provides an applet-like Workbench environment that runs completely outside of a web browser. For more details see, <i>Niagara Java-based Web Clients Guide</i> .

Name	Value	Description
Cache Config	additional properties	In Niagara 4.4 and later, contains subproperties used to configure Cache Config, which caches all station home image files in the web browser. See more details below.
JettyWebServer	read-only	Jetty Web Server Started. See more details below.
User Data Storage	true (default), or false	Enabled by default, this allows web apps to store user-specific data (i.e. user options for the HTML5 Alarm Console) in the <code>userdata</code> folder on the station file system. Typically left enabled, a user with admin privileges can set this to disabled to clear User Data Storage.

### Client Environments—Mobile

The Client Environments container slot allows the station to automatically detect the user agent of an incoming client and use the appropriate Web Profile for the user:

- Default Web Profile if using a Java-enabled device, such as a PC
- Mobile Web Profile if using a mobile device, such as a cell phone or tablet

Name	Value	Description
Enabled	true or false	Activates and deactivates use of the function.
Status	text	Read-only field. Indicates the condition of the component at last polling. <ul style="list-style-type: none"> <li>• <code>{ok}</code> indicates that the component is polling successfully.</li> <li>• <code>{down}</code> indicates that polling is unsuccessful, perhaps because of an incorrect property.</li> <li>• <code>{disabled}</code> indicates that the <code>Enabled</code> property is set to false.</li> <li>• <code>fault</code> indicates another problem.</li> </ul>
Fault Cause	text	Read-only field. Indicates why the network, component, or extension is in fault.
User Agent Pattern	text separated by the pipe symbol ( )	A list of one or more user agents separated by the pipe symbol that identify the target display types.

### Cache Config

In Niagara 4.4 and later the Cache Config property, enabled by default, caches all station home image files in the web browser.

**NOTE:** When upgrading from Niagara 4.3 to Niagara 4.4, you need to clear the browser cache once manually to take advantage of this change. Additionally, when changes are made to a station home image file, clear the browser cache manually to update the cache. One exception to this is if the changed image file type is not one that is included in the Cached File Extensions property setting.

To revert back to the Niagara 4.3 behavior, set the `Enabled` property to `false`.

Type	Value	Description
Enabled	true or false	Activates and deactivates use of the function.
Cached File extensions	png, jpg, gif, svg (default) or *	Set the desired file type(s) to cache or set to "*" to cache all file types without re-validation.

## JettyWebServer

Name	Value	Description
Server State	read-only	Displays the state of the server.
Min Threads	4 - 30 (defaults to 4)	Specifies the minimum number, concurrent connections (threads) that the station makes with the server.
Max Threads	10– max (defaults to 30)	Specifies the maximum number of multiple, concurrent connections (threads) that the station makes with the server.
N C S A Log	NCSA Request Log	Refer the below section..

## N C S A Log

This is a common format of a standardized text file that web servers use to keep track of processed requests.

Name	Value	Description
Enabled	true or false	Activates and deactivates use of the function.
Retain Days	7 (default)	Limits the size of the log by defining how many days to save log information.
Extended Format	true (default) or false	Extends the format of a standardized text file.
Log Cookies	true or false (default)	Logs the cookies of processed requests.
Log Time Zone	list of time zones	Identifies the time zone to use for time stamps.

## 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.

For summary documentation on any view, select **Help→On View (F1)** from the menu or press **F1** while the view is open.

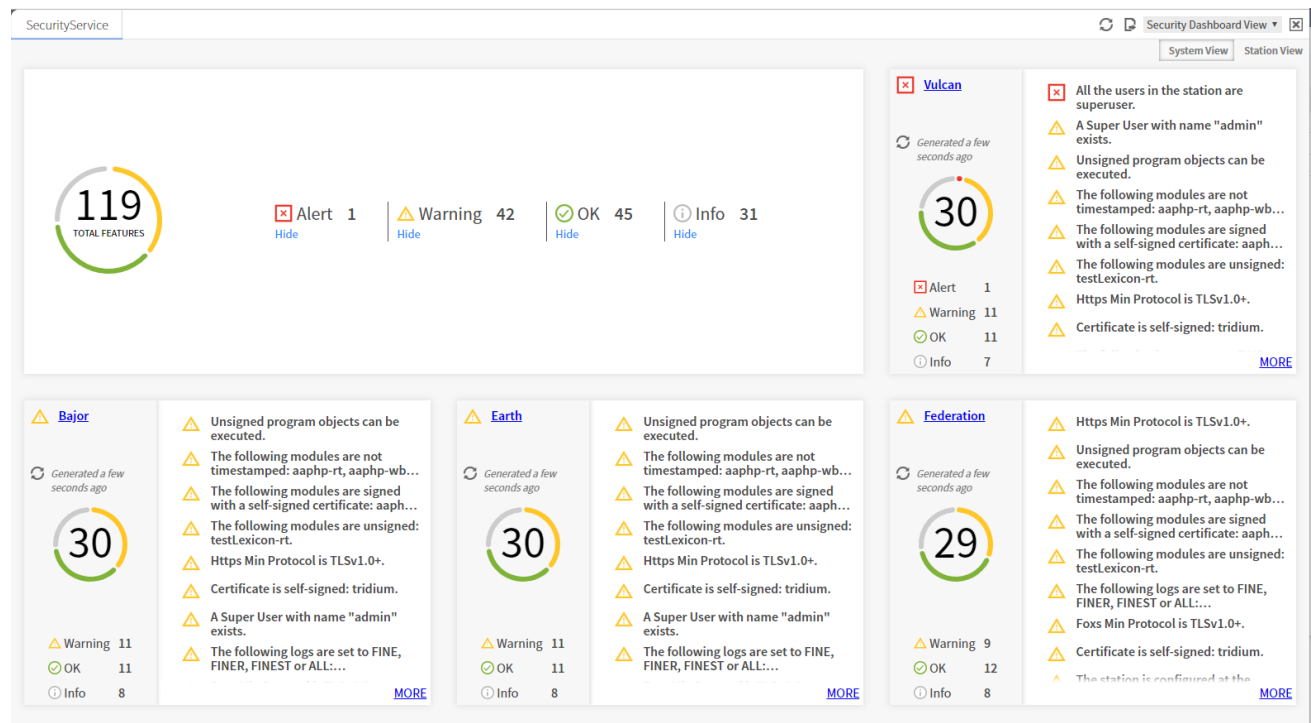
## nss-SecurityDashboardView

In Niagara 4.8 and later, there is added support for the Security Dashboard feature. The **Security Dashboard** is the main view for the Security Service. The view provides (for admin and other authorized users) a snapshot of the security configuration of your station.

**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.

**CAUTION:** The **Security Dashboard View** presents data of a sensitive nature. Users with access should be made aware of this and take necessary precautions to safe-guard the information. For example, a user should not walk away from the PC while the view is open for others to see. A security best practice recommendation would be that any user who has access to the dashboard should be configured for auto-logout.

Figure 22 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 (e.g. 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. That is, if any item is red (alert), the card's status color is red. Similarly, each item listed in a card has a status displayed as a color flag (highest-to-lowest): "Info", "OK", "Warning", or "Alert" as gray, green, yellow, or red icons.

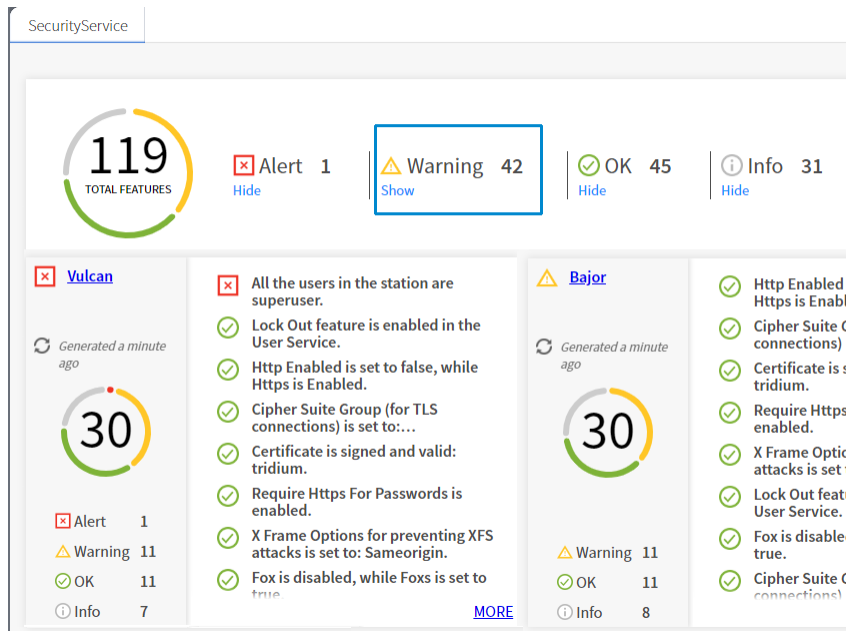
- Gray Info icon (i) indicates secondary information. For example, there is an info level that states how many users are in the station. There is no particular action to take regarding this, it is just presented for consideration.
- Green OK icon (✓) indicates the item's security status is good.
- Yellow 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 which means that the setting is 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 popup 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, no hyperlink is provided on the card. By default, the links on the individual cards in the Security Dashboard view link directly to the remote station. However, this is configurable via the **Station Link Config** property on the SecurityService component. For details, see [nss-SecurityService, page 105](#).

The Summary card, 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 23** Example Summary card set to Hide all Warning status messages



Services reporting to the Security Dashboard include the following:

- Fox Service (e.g. TLS status)
- Web Service (e.g. TLS status)
- Authentication Service (e.g. weak password strength)
- Debug Service (e.g. FINE logs enabled)
- Module Permissions (e.g. SEVERE permissions requested)
- Module Signatures (e.g. modules unsigned)
- Program Objects (e.g. unsigned program objects)
- Platform Settings (e.g. TLS status)
- File System (e.g. users with write access)
- User Service (e.g. super user 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 (e.g. by viewing the Dashboard) when there is 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.

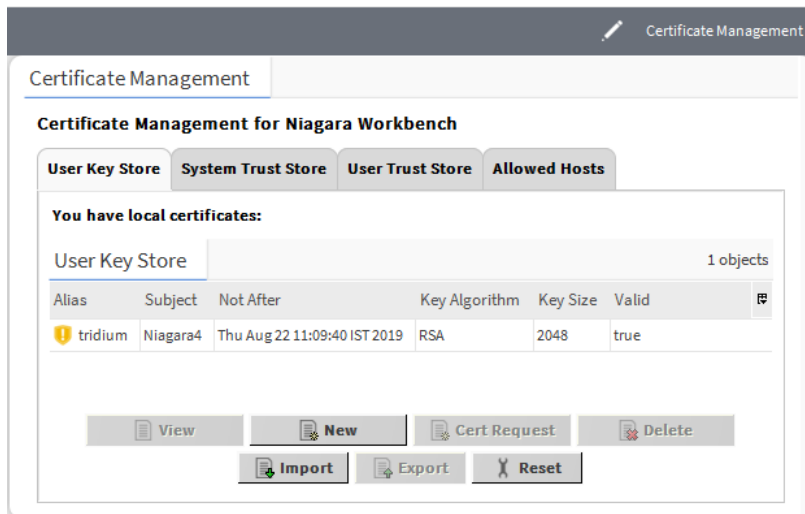
- A time trigger on the **NiagaraNetwork→Station→SecurityDashboardDeviceExt→Data Importer** that allows you to schedule a refresh. The default is to refresh daily.
- The **Refresh System Dashboard Data** action on the **SecurityService** takes a String argument. It will refresh any station that matches 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 on the icon to trigger a refresh for the station.

## platCrypto-CertManagerView

This view is a platform view on any Niagara host. It is also the default view of the **CertManagerService** under a station's **PlatformServices**. The **Certificate Management** view allows you to create digital certificates and Certificate Signing Requests (CSRs). You use this view to import and export keys and certificates to and from the Workbench, platform and station stores. You access this view, via **Tools→Certificate Management**. Also included is a related **Tools Certificate Signer Tool** view.

You use this view to manage PKI (Public Key Infrastructure) and self-signed digital certificates to secure communication within Niagara network. Certificates secure TLS connections to this host.

Figure 24 Example of a Key Store



The Certificate Management view has four tabs:

- User Key Store
- System Trust Store
- User Trust Store
- Allowed Hosts

### User Key Store tab

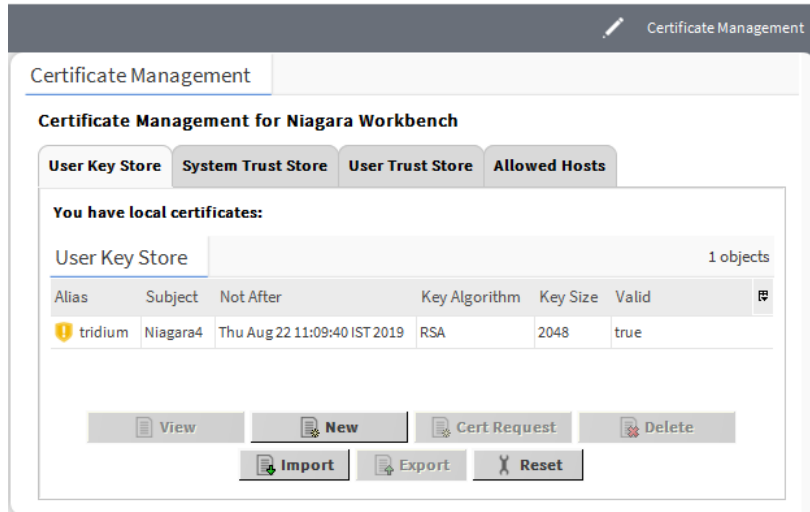
The **User Key Stores** contain server certificates and self-signed certificates with their matching keys. Each certificate has a pair of unique private and public encryption keys for each platform. A **User Key Store** supports the server side of the relationship by sending one of its signed server 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** (`tridium`), however, each certificate is unique for each instance.

Clicking the **New** and **Import** buttons also adds certificates to the **User Key Store**.

Figure 25 Example of a Key Store



Name	Value	Description
Alias	text	A short name used to distinguish certificates from one another in the <b>Key Store</b> . 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	text	Identifies the entity that signed the certificate.
Subject	text	Specifies the Distinguished Name, the name of the company that owns the certificate.
Not Before	date	Specifies the date before which the certificate is not valid. This date on a server certificate should not exceed the <b>Not Before</b> date on the root CA certificate used to sign it.
Not After	date	Specifies the expiration date for the certificate. This date on a server certificate should not exceed the <b>Not After</b> date on the root CA certificate used to sign it.
Key Algorithm	text	Refers to the cryptographic formula used to calculate the certificate keys.
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.
Signature Algorithm	formula text	Specifies the cryptographic formula used to sign the certificate.
Signature Size	KB	Specifies the size of the signature.

Name	Value	Description
Valid		Specifies certificate dates.
Self Signed	text	Read-only. Indicates that the certificate was signed with its own private key.

### User Key Store buttons

Name	Value	Description
View	button	Displays details for the selected item
New	button	Opens the window used to create the entity you are working on.
Cert Request	button	Opens a <b>Certificate Request</b> window, which is used to create a Certificate Signing Request (CSR).
Delete	button	Removes the selected record from the database.
Import	button	Adds an imported item to the database.
Export	button	Saves a copy of the selected record to the hard disk. For certificates, the file extension is .pem.
Reset	button	<p>Deletes all certificates in the <b>User Key Store</b> and creates a new default certificate. It does not matter which certificate is selected when you click <b>Reset</b>.</p> <p><b>CAUTION:</b></p> <p>Do not reset without considering the consequences. The <b>Reset</b> 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.</p>

### Trust Store tabs

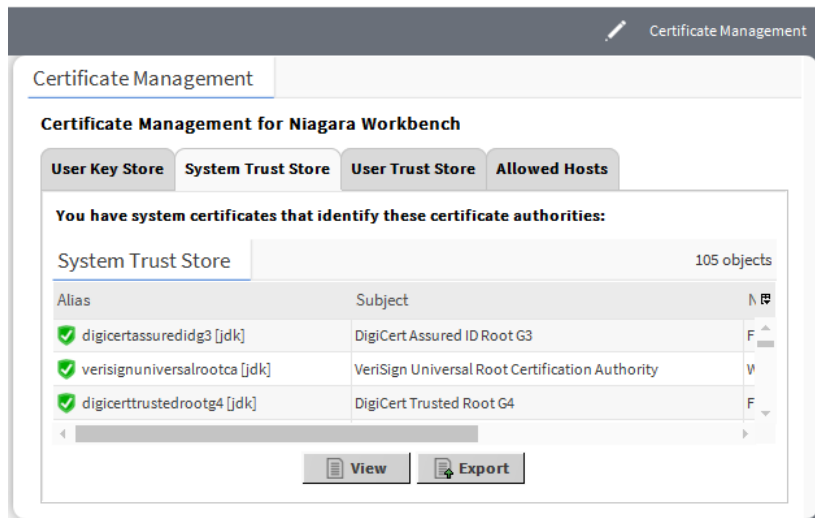
The **Trust Stores** contain signed and trusted root certificates with their public keys. These stores contain no private keys. A **Trust Store** supports the client side of the relationship by using its root CA certificates to verify the signatures of the certificates it receives from each server. If 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 26 Example of a System Trust Store



### Trust Store columns

Name	Value	Description
Alias	text	A short name used to distinguish certificates from one another in the <b>Key Store</b> . 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	text	Identifies the entity that signed the certificate.
Subject	text	Specifies the Distinguished Name, the name of the company that owns the certificate.
Not Before	date	Specifies the date before which the certificate is not valid. This date on a server certificate should not exceed the <b>Not Before</b> date on the root CA certificate used to sign it.
Not After	date	Specifies the expiration date for the certificate. This date on a server certificate should not exceed the <b>Not After</b> date on the root CA certificate used to sign it.
Key Algorithm	text	Refers to the cryptographic formula used to calculate the certificate keys.
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.
Signature Algorithm	formula text	Specifies the cryptographic formula used to sign the certificate.
Signature Size	KB	Specifies the size of the signature.
Valid		Specifies certificate dates.
Self Signed	text	Read-only. 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**.

Name	Value	Description
View	button	Displays details for the selected item
Delete	button	Removes the selected record from the database.
Import	button	Adds an imported item to the database.
Export	button	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 **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.

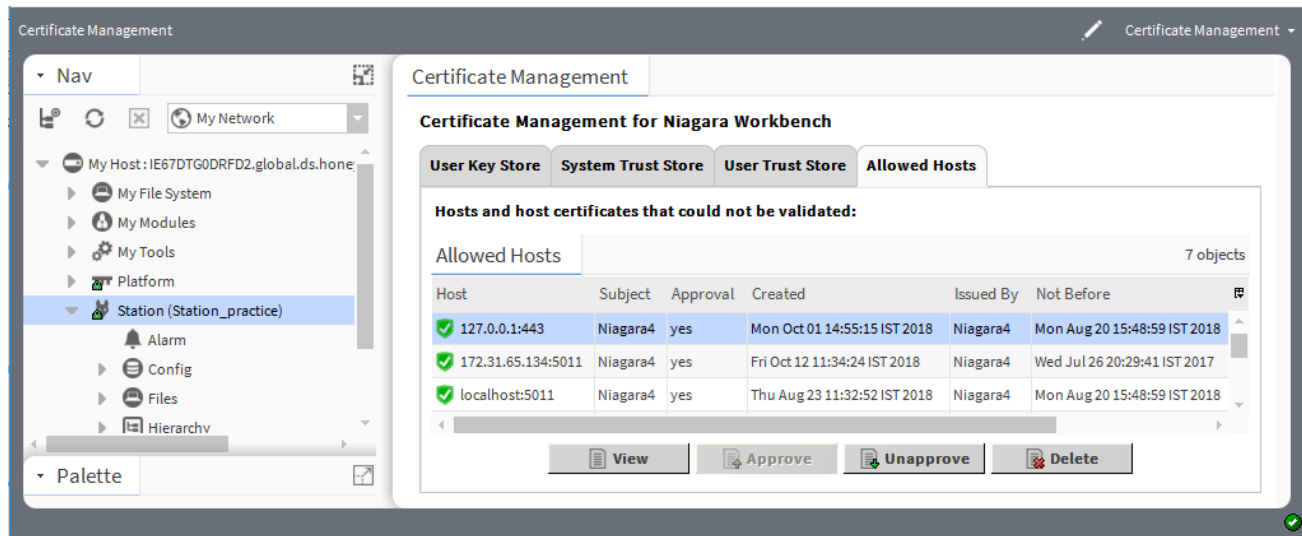
If this is Workbench to station connection, the system prompts you to approve the host exemption. Workbench challenges server identity at connection for unapproved hosts and, unless specific permission is granted, prohibits communication. Once permission is granted, future communication occurs automatically (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 (expand the station **ClientConnection** under **NiagaraNetwork**) reports the problem.

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 will need to approve an updated exemption. The same is true if a new self-signed certificate is generated on the host.

## Allowed Hosts columns

Figure 27 Example of an Allowed Hosts list



Name	Value	Description
Host	text	Specifies the server, usually an IP address.
Subject	text	Specifies the Distinguished Name, the name of the company that owns the certificate.
Approval	Yes or No	Specifies the servers within the network to which the a client may connect. If approval is set to <b>no</b> , the system does not allow the client to connect.
Created	date	Identifies the date the record was created.
Issued By	text	Identifies the entity that signed the certificate.
Not Before	date	Specifies the date before which the certificate is not valid. This date on a server certificate should not exceed the <b>Not Before</b> date on the root CA certificate used to sign it.
Not After	date	Specifies the expiration date for the certificate. This date on a server certificate should not exceed the <b>Not After</b> date on the root CA certificate used to sign it.
Key Algorithm	text	Refers to the cryptographic formula used to calculate the certificate keys.
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.
Signature Algorithm	formula text	Specifies the cryptographic formula used to sign the certificate.
Signature Size	KB	Specifies the size of the signature.
Valid		Specifies certificate dates.

## Allowed Hosts buttons

Name	Value	Description
View	button	Displays details for the selected item
Approve	Yes or No	Designates the server as an allowed host.
Unapprove	Yes or No	Does not allow connection to this server host. The system terminates any attempted communication.

## 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) is making a category mask adjustment, the user must also have at least 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 28 Category Browser

Category Browser									
	Inherit	User	Admin	Operator	Viewer	Category 5	Category 6	Category 7	Category 8
Alarm	n/a		●						
▼ Config	n/a		●						
▶ Services	✓		●						
▶ Drivers	✓		●						
▶ Apps		●							
▼ category	✓		●						
▶ Temp1	✓		●						
▶ Temp2	✓		●						
▶ Alarm				●					
▶ Ramp	✓		●						
▶ SineWave	✓		●						
▶ Files	n/a		●						

## Columns

Column	Value	Description
Inherit	check mark or blank	A check mark indicates that the object inherits the category from its parent in the table.
User	Category 1	All system objects except for those listed as assigned to Admin are assigned to this category.

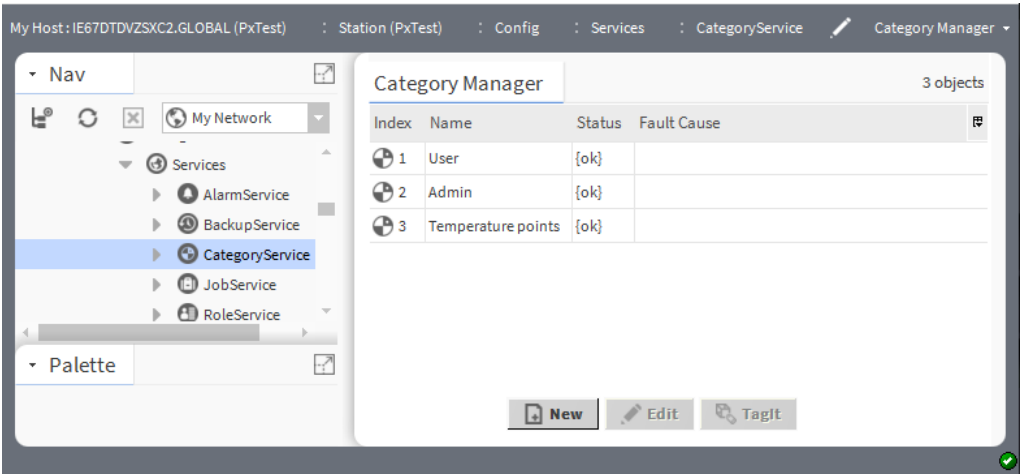
Column	Value	Description
Admin	Category 2	These objects default to the Admin category: <ul style="list-style-type: none"><li>The configuration services: <b>UserService</b>, <b>CategoryService</b>, and <b>ProgramService</b></li><li>All files (the entire file space)</li></ul>
Categories 3–8	bold bullet, grayed out bullet, or blank	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.

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 29 Category Manager, Temperature Points as Category 3



Column	Value	Description
Index	number	A unique number for the category, as it is known to the station.
Name	text string	Descriptive text that reflects the purpose of the entity or logical grouping.

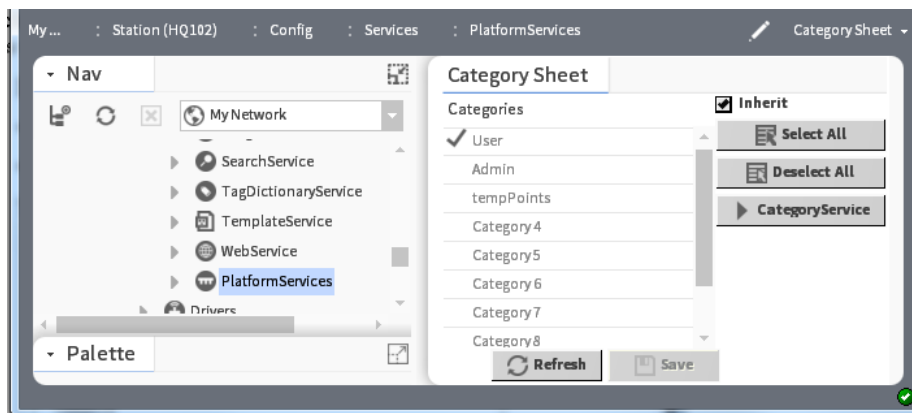
Column	Value	Description
Status	text	Read-only field. Indicates the condition of the component at last polling. <ul style="list-style-type: none"> <li><code>{ok}</code> indicates that the component is polling successfully.</li> <li><code>{down}</code> indicates that polling is unsuccessful, perhaps because of an incorrect property.</li> <li><code>{disabled}</code> indicates that the <b>Enable</b> property is set to false.</li> <li><code>fault</code> indicates another problem.</li> </ul>
Fault Cause	text	Read-only field. Indicates why the network, component, or extension is in 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) is making a category mask adjustment, the user must also have at least 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 30 Category Sheet



Option/button	Value	Description
Categories	text	Provides one table row for each category name.
Inherit	toggle	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.
Select All	button	Effective if <code>Inherit</code> is cleared, clicking this button assigns this component to all categories in this station.
Deselect All	button	Effective if <code>Inherit</code> is cleared, clicking this button removes this component from all categories.
CategoryService	button	Opens the <b>Category Browser</b> .

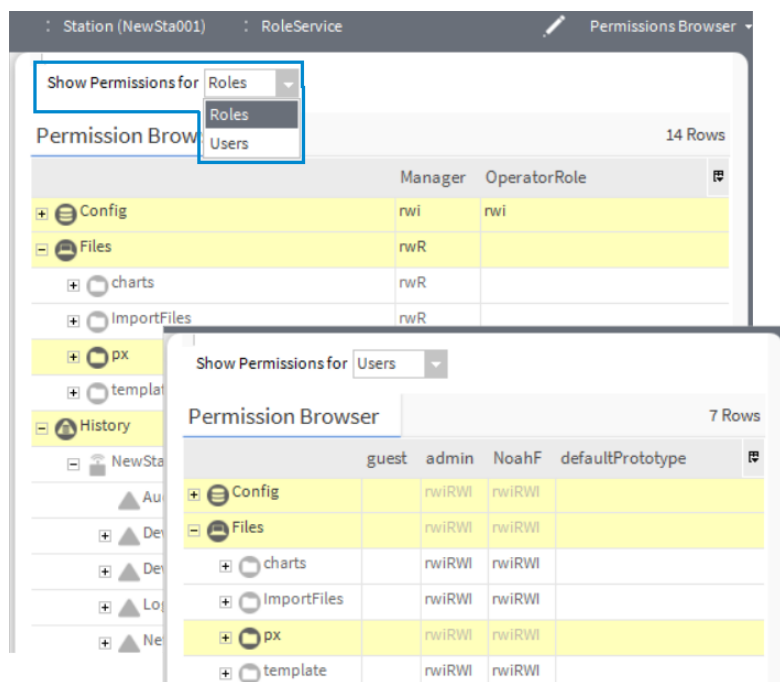
Option/button	Value	Description
Refresh	button	Re-displays the <b>Category Sheet</b> .
Save	button	Records the changes made.

## 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 4.8 and later, there is added support for the UserService in the **Permissions Browser** view. Use the **Show Permissions for** dropdown 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 31 Permissions Browser view shows permissions for Roles and Users




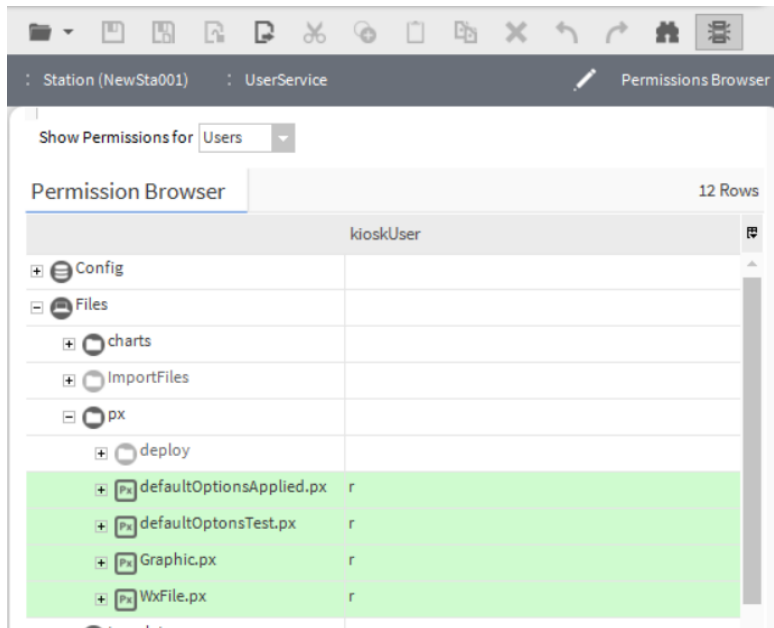
Columns represent roles or users, and rows identify the objects in the station, with each table cell showing user permissions.


- Yellow rows are objects explicitly assigned with permissions.
- Dimmed rows represent objects that inherit their permissions from their parent object.

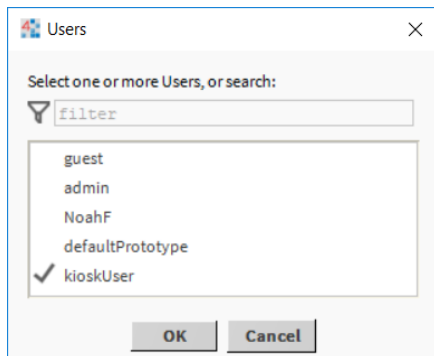
Double-click a cell to bring up the permissions window for that role or user depending on which option is selected in the **Show Permissions for** dropdown list. This allows you to globally change a permission levels for any category in the station.

Additional enhancements to the **Permissions Browser** view include the following:

- **Highlight Accessible** – applies green shading to entries that are accessible by at least one displayed column (user or role). Do this by clicking the **Highlight Accessible** icon (  ) in the Workbench toolbar.

**Figure 32** Highlight Accessible tool highlights entries accessible by this user

- **Filtering** – filters results in the table for selected users/roles. Do this by clicking the table **Options** icon (  ) at the far right side of the column heading row, to display the options menu, and click on **Show Users...** (or **Show Roles...**) to open a window that lists all users (or roles) which you can filter by search or selection and click **OK**.

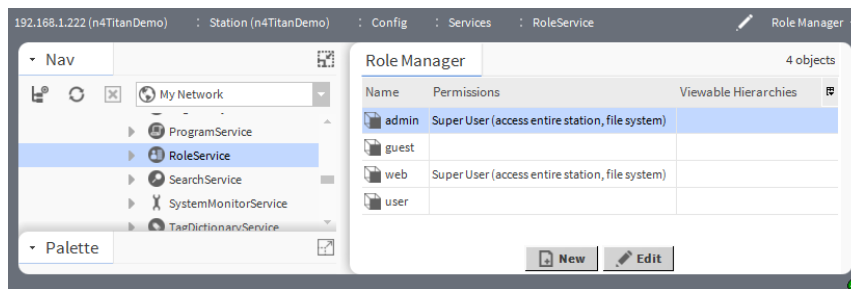
**Figure 33** Filtering shows permissions for selected users/roles

Column	Value	Description
First column	Nav tree for station Config, Files and History	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	permissionsR = readW = writel = invoke actionadmin level permissions appear in upper case.	Reports the rights assigned to the <code>admin</code> role. As this is a super user, <code>admin</code> has rights to read, write and invoke an action for all objects.
user	permissionsr = readw = writei = invoke actionoperator permissions appear in lower case.	Reports the rights assigned to the <code>user</code> role. The default is no rights assigned.

## wbutil-RoleManager

This manager allows you to create, edit and delete roles. It is the default view of the RoleService and is located in the station's **Services** container.

Figure 34 Role Manager view

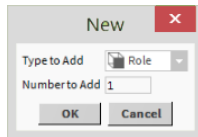


The system creates the `admin` role by default and grants it super user permissions. The `admin` role does not appear in the **Role Manager** view and you may not delete it.

Column or field	Value	Description
Name	text	Identifies the role to be assigned to one or more users. Role names are case sensitive.
Permissions	text	Associates a name with a specific set of permissions.
Viewable Hierarchies	text	Identifies the hierarchies this user may view.
Type	Role (default)	Identifies the type of entity being created.
Number to add	number	Allows you to create many rows at once in the <b>Role Manager</b> view's table.

## New role window

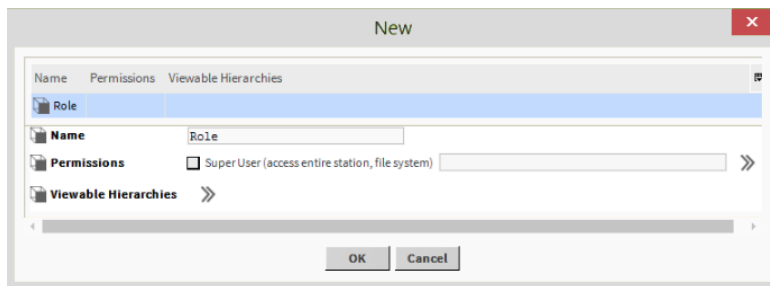
Figure 35 New Role window



Column or field	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 <b>Role Manager</b> view's table.

## New role properties

Figure 36 New role properties



**CAUTION:** It is important to understand the risk 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.

## Windows

Windows create and edit database records or collect information when accessing a component. You access them by dragging a component from a palette to a nav tree node 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 Export windows

These windows save a copy of the selected certificate to a folder on your hard drive or thumb drive.

## Certificate Export chooser

**Certificate Export**

**Certificate**

☒ Export the public certificate

Table View ASN.1 View PEM View

**Properties:**

Version	v3
Serial Number	47 e6 05 b1 75 4d 70 70 35 54 50 0f
Issued By	root 2
Issuer DN	CN=root 2, O=My Company, L=Anywhere, ST=Any...

**Private Key**

☒ Export the private key

Private Key Password (required):

☒ Encrypt exported private key

☒ Reuse password to encrypt private key

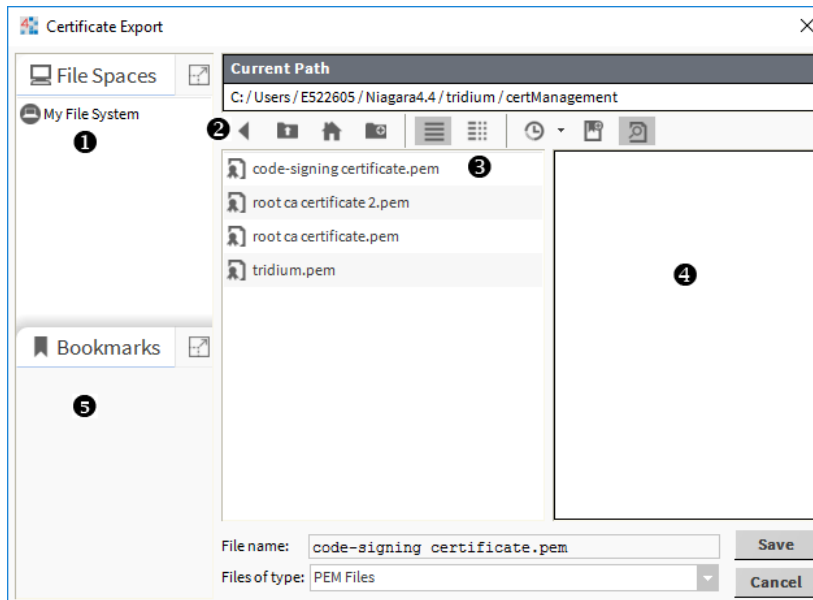
Password

Confirm

OK Cancel

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



①	Displays the available file spaces. The contraction (🔍) and expansion (🗲) icons resize the selected pane.
②	<p>The control icons determine the contents of the file view pane:</p> <ul style="list-style-type: none"> <li>◀ returns to a previous folder.</li> <li>📁 displays files in the folder above the current folder.</li> <li>🏠 displays files in the root of the user home.</li> <li>📁 creates a new folder.</li> <li>☰ configures the chooser to display the file name only.</li> <li>☷ configures the chooser to display the details for each file.</li> <li>🕒 provides a drop-down list of available file paths.</li> <li>🔖 creates an entry in the Bookmarks pane.</li> <li>🔍 turns the display of the details pane on and off.</li> </ul>
③	The file view pane.
④	This pane displays the details for the selected file.
⑤	Bookmarks pane.

## Edit history export window

The **Edit** window shows the configuration properties of the history export descriptor, plus **Name**, which is equivalent to the right-click **Rename** command on the descriptor. To access all properties, including all status properties, view the HistoryService property sheet.

The **NiagaraHistoryExport** line above the properties summarizes the properties.

Property	Value	Description
Name	Text string followed by numbers	For a history originating in the local host station, the name begins with <code>Local_</code> . If Discovered for import, typically left at default. For a system history export, originating in the remote station, the name begins with <code>NiagaraSystemHistoryExport</code> .
Execution Time — Manual	N/A	Requires human intervention to initiate a history export or import.
HistoryId	Text in two parts: <code>/stationname/historyname</code>	Specifies the history name in the local station's History space, using two parts: <code>"/&lt;stationName&gt;"</code> and <code>"/&lt;historyName&gt;"</code> . If Discovered, station name is <code>"^"</code> (a character representing the device name of the parent container) and history name reflects the source history name. Typically, you leave both fields at default values, or edit the second ( <code>&lt;historyName&gt;</code> ) field only.
Execution Time — Daily (default)	Time Of Day hours:minutes:seconds AM/PM timezone Randomization Days Of Week	Defines when the daily export or import automatically takes place. The hours follow a 24-hour clock.
Execution Time — Interval	Interval hours:minutes:seconds Time Of Day Days Of Week	Defines the amount of time between automatic exports or imports. Hours may number in the thousands.
Enabled	true or false	Activates and deactivates use of the function.

## 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 37 Default view of the Generate Self-Signed Certificate window

Generate Self Signed Certificate

Generates a self signed certificate and inserts it into the keystore

Alias  (required)

Common Name (CN)  (required)

\* this may contain the host name or address of the server

Organizational Unit (OU)

Organization (O)  (required)

Locality (L)

State/Province (ST)

Country Code (C)  (required)

Not Before  24-Jun-2014 09:09 AM EDT

Not After  24-Jun-2015 09:09 AM EDT

Key Size ☒ 1024 bits ☐ 2048 bits ☐ 3072 bits ☐ 4096 bits

Certificate Usage ☒ Server Certificate ☐ Client Certificate ☐ CA Certificate

Alternate Server Name

Email Address

OK Cancel

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 identity. 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. See the *Niagara Station Security Guide* for more information about using TLS (Transfer Layer Security) to secure communication among security system components.

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	A short name used to distinguish certificates from one another in the <b>Key Store</b> . 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 <b>Subject</b> in the <b>User Key Store</b> .
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.
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 <a href="http://www.countrycode.org">www.countrycode.org</a> . 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 exceed the <b>Not Before</b> date on the root CA certificate used to sign it.
Not After	date	Specifies the expiration date for the certificate. This date on a server certificate should not exceed the <b>Not After</b> date on the root CA certificate used to sign it.
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.

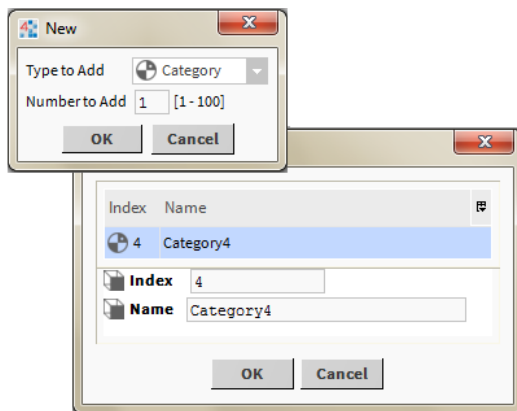
Name	Value	Description
Alternative Server Name	text	This property provides a name other than the Subject (Common Name) that the system can use to connect to the server. Like the Common Name, the system uses the Alternative Server Name to validate the server certificate making it possible to specify both 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.

## New category windows

The **New** window appears when you click the **New** button at the bottom of the **Category Manager** view. When you click **OK** a second **New** window displays, allowing you to assign name and index details.

### Basic category window

Figure 38 Creating basic categories

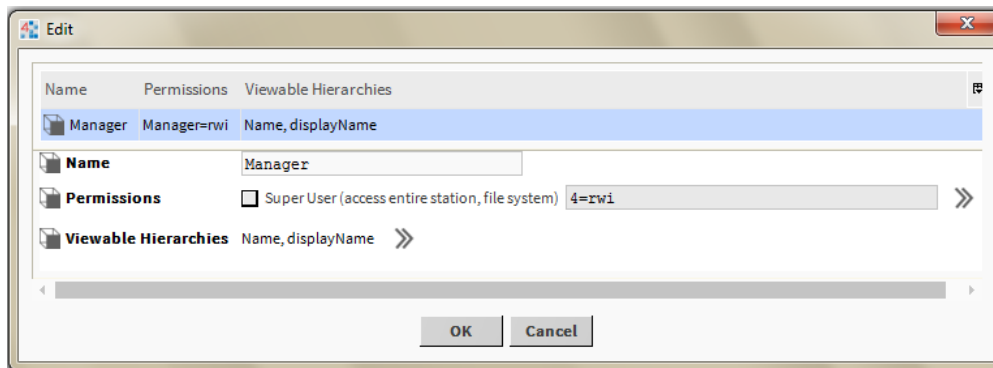


Property	Value	Description
Name	text	A name for the given category. This can be any name that describes how you are using the category.
Index	number	A unique number for the category, as it is known to the station.

## New/Edit roles window

This window creates and edits roles and permissions. You access it from the **Role Manager** view (**RoleService**).

Figure 39 Example of an Edit role dialog box

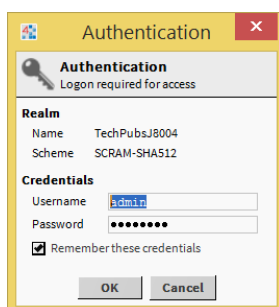


Property	Value	Description
Name	text string	Descriptive text that reflects the purpose of the entity or logical grouping.
Permissions	check box and chevron	<p>Checking the <b>All Powerful User</b> (super user) check box sets up a role that allows the user to access the entire station and file system. To set individual permissions for a role, click the chevron.</p> <p>The box between The All Powerful User and the chevron displays all categories. Basic categories display by index number (for example: 3=rwi).</p>
tags	various	Any tags associated with the object appear at the end of the property sheet. The tag icon identifies them. The systems integrator sets up tags to provide additional, searchable information about the object.

## Platform Authentication window

This window opens after a secure platform connection is made. Its purpose is to authenticate the platform user.

Figure 40 Platform authentication window

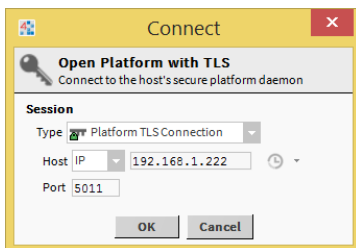


Name	Value	Description
Name	text	The name may include your company network name (when connecting to a Windows-based computer), the IP address of the controller, or host name of the controller.
Scheme	text	This information identifies the authentication scheme used to log on to a platform: HTTP-Basic, 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 <b>Platform Administration→User Accounts</b> .
Credentials—Password	text	This is the password created when the controller was commissioned. Access to this password is through <b>Platform Administration→User Accounts</b> .
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 window

This window opens when you open a platform (supervisor PC or controller).

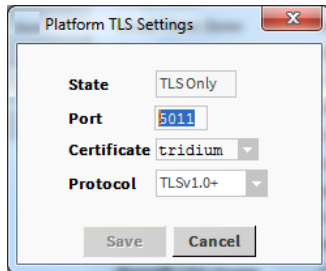
Figure 41 Platform connect window



Name	Value	Description
Type	drop-down list	Defaults to Platform TLS Connection.
Host (type)		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 platform communication. Defaults to 5011.

## 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**.

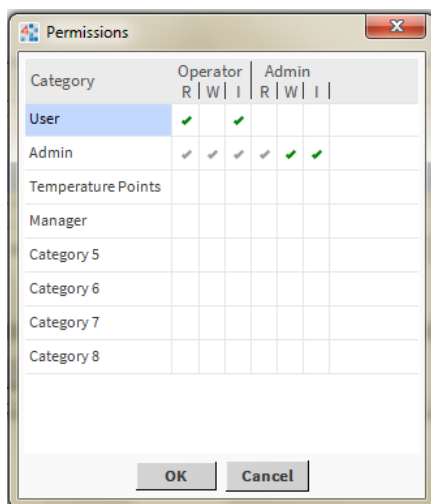


Name	Value	Description
State	TLS	Defaults to TLS only.
Port	number	The port for secure communication. Defaults to 5011
Certificate	drop-down list	Provides a list of available certificate aliases. The <code>tridium</code> certificate is the default, self-signed certificate created when you first accessed the platform.
Protocol	drop-down list TLSv1.0+ (default) TLSv1.1+ TLSv1.2	<p>The minimum level of the TLS (Transport Layer Security) protocol to which the server will accept negotiation. The default includes versions 1.0, 1.1 and 1.2. It works with most clients, providing greater flexibility than an individual version.</p> <p>During the handshake, the server and client agree on which protocol to use.</p> <p>Change <b>Protocol</b> from the default if your network requires a specific version or if a future vulnerability is found in one of the versions.</p>

## 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 42 Permissions map

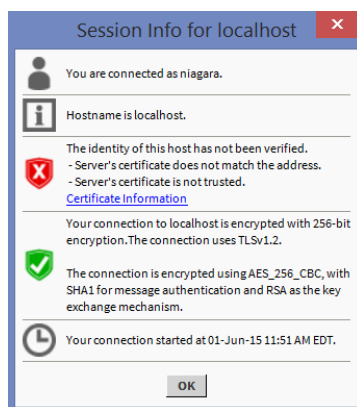


Column	Value	Description
Category	table row	User and Admin are default categories created by the <b>New Station</b> wizard. Each category occupies a single row in the Permissions map.
Operator	permission level	Provides a way to set access rights for components that are configured with the <code>operator</code> permission level.  Permission level is set by the <code>Operator</code> config flag on the component's <b>Property Sheet</b> .
Admin	permission level	Indicates that the object may be read, written or an action invoked by only system users that have been granted <code>admin</code> rights.  The Admin permission level is set by turning off the <code>Operator</code> config flag on the component's <b>Property Sheet</b> .




## 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 (i).

Figure 43 Example of Session Info when using a self-signed certificate



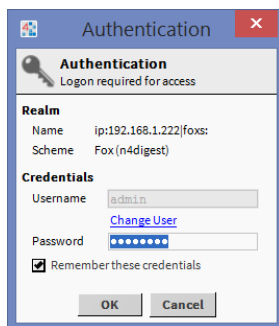
Screen element	Value	Description
	Connection	Identifies the user account that is logged in to the station.
	Hostname	Identifies the host name of the server.
or	Identity verification	Reports on the attempt to verify the authenticity of the server.  A green shield indicates that a root CA certificate in 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 <b>Trust Store</b> with which to verify the server certificate.  Clicking <code>Certificate Information</code> displays the certificate.

Screen element	Value	Description
 or 	Encryption	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.
	Time	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 44 Station Authentication window

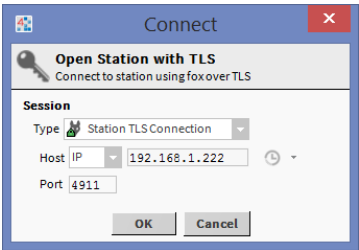


Name	Value	Description
Name	text	The station name, which, for a controller, is usually its IP address.
Scheme	text	This information identifies the authentication scheme used to log on to this station. The scheme used depends upon the user. Configuration is through the UserService.
Credentials—Username	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 45 Station Connect window



Name	Value	Description
Type	drop-down list	Defaults to Station TLS Connection.
Host (type)		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	<p>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.</p> <p>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 <i>Niagara Station Security Guide</i>.</p>
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 <b>System</b> or <b>User Trust Store</b> 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.
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 <b>Role Manager's</b> 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 <code>admin-Read (R)</code> permission is required (as defined in the <b>Role Manager's</b> permissions map). When checked a user with a minimum of <code>operator-read</code> permission ( <code>r</code> ) may access the slot.
role	<p>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 (<code>admin</code>), regular users (<code>user</code>), and operators (<code>operator</code>).</p> <p>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.</p> <p>You manage roles using the <code>RoleService</code>.</p>
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 .....	86
allowed hosts .....	39
Allowed Hosts .....	20
Allowed Hosts tab .....	121
ancestor permission level .....	87
audit log .....	72
authentication .....	59
audit log .....	72
AXDigestScheme .....	92
DigestScheme .....	91
HTTPBasicScheme .....	92
of users .....	47
scheme, assigning to users .....	71
schemes .....	48
authentication scheme .....	108
password management .....	71
AuthenticationService .....	89
authorization checklist .....	78
authorization management .....	11
auto logoff	
station .....	74
AX authentication .....	48
AXDigestScheme .....	48, 89, 92

## B

backup, <i>See</i> certificate, exporting	
baja-AuthenticationSchemeFolder .....	89
baja-AuthenticationSchemes .....	89
baja-SSOConfiguration .....	90
baja-UserPrototype .....	94
basic categories .....	80
best practices .....	12
browser trust store	
installing client certificate in .....	55

## C

categories .....	80
category	
adding and editing .....	80
basic .....	80
deleting .....	82
for assigning components .....	81
New window .....	134
Category	
Browser .....	123
Crowser .....	81
Category Sheet .....	125
certificate .....	19
backing up, <i>See</i> certificate, exporting	
create .....	132

creating a root CA cert .....	27
creating a server cert .....	30
deleting .....	40
expired .....	40
exporting .....	36
importing into a User Trust Store .....	37
importing into the User Key Store .....	35
installing in a remote platform/station .....	37
naming convention .....	20
self-signed .....	17
Session Info .....	39
signing .....	33
stores locations .....	20–21
TLS	
session info .....	39
types .....	16
Certificate Authority .....	31
Certificate Export windows .....	129
Certificate Management plugin .....	117
Certificate Signing Request	
folder structure .....	22
certManagement folder .....	22
checklist	
certificate creation and signing .....	23
platform/station for server verification .....	24
Supervisor station for server identity .....	24
user authentication .....	47
client certificate .....	16
exporting .....	53
installing in browser .....	55
public key .....	51
user workflow .....	51
client certificate authentication	
admin workflow .....	49
Client Certificate Authentication	
configure .....	49
client/server relationships .....	15, 27
ClientCertAuth	
users workflow	
browser login .....	56
Workbench login .....	58
clientCertAuth-ClientCertAuthScheme .....	101
ClientCertAuthScheme .....	48
code-signing certificate .....	31
component	
adding a component .....	85
assigning to a basic category .....	81
permission level .....	78
permissions .....	11, 77
permissions, troubleshooting .....	88
component permissions checklist .....	78
components .....	89
config flag, setting .....	78
configure SAML Authentication Scheme .....	61
configuring client port .....	41

controller,  
     replacing securely .....45  
 create user prototype for SAML authentication....61  
 cryptography.....15  
 CSR, See Certificate Signing Request  
     folder structure .....22  
 customize SAML attribute mapping.....63

## D

DigestScheme ..... 47–48, 89, 91  
 document change log .....7

## E

edit .....131  
 Edit roles window .....134  
 email  
     securing using TLS.....41  
 encryption..... 18–19  
 engineering platform/station  
     certificate configuration.....24  
 exporting  
     client certificate.....53

## F

file permissions.....87  
 Foxs properties .....41  
 FoxService .....102

## G

gauth palette.....47  
 Generate Self-Signed Certificate window.....132  
 global password configuration .....91  
 Google Authentication.....59  
 Google Authenticator app .....47  
 GoogleAuthenticationScheme.....48

## H

history export window .....131  
 history permissions .....87  
 HTTPBasicScheme ..... 89, 92  
 https  
     required to set password in browser .....72  
 Https properties .....41

## I

identity verification ..... 15, 19  
 inherited categories .....80  
 installing  
     client certificate.....55

## K

keys  
     public and private.....18  
 kiosk .....58  
 kiosk-like mode  
     enabling.....58

## L

Login with SSO .....63

## N

naming convention  
     for certificates.....20  
 network users.....65  
     challenge .....65  
 New category window .....134  
 niagara\_user\_home .....22  
 nss-SecurityDashboardView .....114  
 nss-SecurityService .....105

## O

Operator config flag .....78

## P

password  
     expiration .....71  
     global configuration .....91  
     management.....71  
     renewing an expired password.....74  
     setting up .....72  
     setting up for a user .....72  
     strength.....29  
     strength, setting up.....71  
     troubleshooting .....75  
     warning period.....71  
 permission level.....78  
 permissions .....83  
     editing .....85  
     to access files .....87  
     to view history files.....87  
 Permissions  
     map .....137  
 permissions checklist .....78  
 PKI certificate  
     creating .....30  
 platform  
     authentication window .....135  
     connect window .....136  
     opening a secure connection.....25  
     stores .....20  
     TLS properties.....136  
 platform/station

checklist for server verification .....	24
plugins .....	89, 114
Certificate Management view .....	117
precautions .....	11
private key .....	18
provisioning	
installing a certificate .....	37
public key .....	18

## R

Role Manager .....	128
roles	
about their configuration .....	83
add and edit .....	134
adding .....	83
and permission level .....	78
assigning to users .....	86
editing .....	85
reviewing access rights .....	86
testing .....	86
RoleService .....	94
root CA certificate	
creating .....	27
installing in remote platform/stations .....	37
root certificate	
importing into Windows trust store .....	35

## S

SAML authentication .....	59
SAML SSO	
about .....	59
saml-SamlXmlDecrypter .....	109
SAMLAttributeMapper .....	107
SAMLAuthenticationScheme .....	48, 108
secure communication .....	11, 41
configuring station security properties .....	41
configuring Supervisor and platforms .....	40
enabling clients .....	41
encryption .....	19
Foxs properties .....	41
FoxSrvce configuration properties .....	102
Https properties .....	41
Niagara check list .....	23
NiagaraNetwork enabling .....	26
securing outgoing email .....	41
Security Dashboard	
overview .....	13
security precautions .....	11
server authentication	
fixing error conditions .....	42
TCP ports .....	45
server certificate .....	16
creating .....	30
server identity	
Supervisor setup check list .....	24

server identity verification	
setting up .....	23
server verification	
station setup check list .....	24
Session Info .....	39
window .....	138
Single Sign On	
about .....	59
configuration properties .....	90
SSO .....	59, 90, 107–108
station	
authentication window .....	139
connect window .....	139
health .....	38
logging off .....	72
logging on .....	72
security model .....	77
set up server verification .....	23
stores .....	20
station-to-station user .....	70
stations	
configuring security properties .....	41
stores	
accessing .....	20
file names .....	22
locations on the Supervisor platform .....	21
System Trust Store .....	20

## T

TCP ports .....	45
TLS, See Transport Layer Security protocol	
keys .....	18
platform properties .....	136
Transport Layer Security protocol .....	15
troubleshooting	
component permissions .....	88
server authentication .....	42
trust store	
Windows .....	35
Trust Store tab .....	119
two-factor authentication .....	47, 59

## U

user	
adding and editing .....	70
authentication .....	48
password .....	71
user authentication .....	11
user authentication checklist .....	47
User Key Store .....	20
tab .....	117
User Trust Store .....	20
importing a certificate .....	37
UserService	
component permission level .....	79

setting up a password .....72

## **V**

views..... 89, 114, See plugins

## **W**

Web Service ..... 109

windows..... 89, 129

Windows trust store.....35