

Technical Document

NiagaraAX SSL Connectivity Guide

November 11, 2013



NiagaraAX SSL Connectivity Guide

3951 Westerre Pkwy., Suite 350

Richmond

Virginia

23233

U.S.A.

Confidentiality Notice

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 is 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 © 2009-2013 Tridium, Inc. All rights reserved.

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

PREFACE

Preface

- [Document Change Log](#)

Document Change Log

Updates (changes or additions) to this document are listed as follows.

- Changes to the document made: November 8, 2013:
 - Modified [“Move your legacy certificate folder structure \(Niagara 3.8\)”](#) on page 2-3 to clarify that copying the legacy file structure forward applies only if upgrading to NiagaraAX 3.8 from a previous version.
 - Modified [“Create a folder structure”](#) on page 3-2 to accommodate both version 3.7 and 3.8 users.
- Changes based on NiagaraAX 3.8 made: November 5, 2013
 - Rewrote the answer to the question about the upgrade path to explain that the **security** folder is automatically copied forward when NiagaraAX is updated. See [“Frequently-asked questions”](#) on page 1-1.
 - Added a topic to suggest that legacy users should cut and paste their current certificate folder structure to the new **certManagement** folder. See [“Move your legacy certificate folder structure \(Niagara 3.8\)”](#) on page 2-3.
 - Modified [“Create a folder structure”](#) on page 3-2 to explain that opening Certificate Management for the first time automatically creates the **certManagement** folder.
 - Modified the certificate creation and signing graphics in [“Certificate creation”](#) on page 5-2 to illustrate the **certManagement** folder. Added introductory sentences to several of the items in the ordered lists that follow the certificate creation illustrations.
- Changes to 3.7u1 made: August 16, 2013
 - Made general editorial changes throughout the book, for example, removed “decrypt” and “decryption” based on the assumption that the word “encrypt/encryption” covers both directions.
 - Rewrote the answer to [“Does using multiple keys slow performance?”](#) in [“Frequently-asked questions”](#) on page 1-1.
 - Combined all best practices into one section of the introductory chapter. See [“NiagaraAX SSL best practices”](#) on page 1-3.
 - Rewrote the answer to the comment/question, **“Our company already has signed certificates. Can they be used on our JACEs?”** See [“NiagaraAX SSL best practices”](#) on page 1-3.
 - Rewrote the answer to the question that begins, **“Is there an upgrade path for SSL security...”**. This is the last paragraph before [“NiagaraAX SSL best practices”](#) on page 1-3.
 - Added two sentences to the fifth bullet, rewrote the fourth bullet, and rewrote the last bullet from the end of [“Security considerations”](#) on page 1-4.
 - Changed the note following the last bullet in [“Security considerations”](#) on page 1-4 to a caution and rewrote the final sentence.
 - Added a sentence that begins, “Before you begin...” in the introduction to [“Install SSL module”](#) on page 2-2.
 - Added a qualifying phrase at the beginning of the section [“Check for SSL license”](#) on page 2-3.
 - Rewrote the discussion following step 3 in [“Enable the Web Service connection”](#) on page 2-7.
 - Added a note to explain that when **Https Enabled** is set to **true**, **Foxs Enabled** must also be set to **true**. See [“Confirm the Fox Service connection”](#) on page 2-7
 - Added the paragraph beginning “Ideally, you want to set...” as part of the discussion following step 2 in [“Confirm the Fox Service connection”](#) on page 2-7
 - Created a new section, [“Temporarily approve the self-signed certificates”](#) on page 2-9.

- Reworked “Configure outgoing email” on page 2-9.
- Reworked “Upgrade from crypto.jar” on page 2-10.
- Resized all the drawings in “About SSL—Alice, Bob, Cathy and Bart” on page 4-1 and “About NiagaraAX SSL” on page 5-1 so that they will look better in the help system.
- Added an explanation of “man-in-the-middle attack” to “When things go wrong” on page 4-2.
- Reworked “Keys” on page 4-4.
- Reworked portions of “Signing a certificate with a private key” on page 4-4.
- Made significant changes to the Figure 5-1 in the section titled “NiagaraAX’s client/server architecture” on page 5-1, including adding numbers to the illustration so that it is easier to identify the relationships being called out.
- Reworked “Workbench certificate signing” on page 5-3, including changing an error in the illustration.
- Combined very similar information into “About the SSL Toolset” on page 5-7 and reworked the wording to contain all the unique information from the similar sections.
- Added a new question and answer pair to document a login error. See “Fix error conditions” on page 6-1.
- Reworked “Reset or replace a JACE securely” on page 6-2.
- Rewrote the definition of CA Certificate in “SSL Toolset terminology” on page 7-1.
- Rewrote the descriptions of these properties: **Fox Enabled**, **Foxxs Enabled**, **Foxxs only** and added a note describing how each are used. See “Fox Service configuration properties” on page 7-8. Did the same for **Http Enabled**, **Https Enabled**, and **Https only**. See “Web Service configuration properties” on page 7-9.
- 3.7 Update 1 (3.7u1): May 31, 2013.
 - Rewrote a number of procedures in “Set up Workbench and stations for SSL” on page 2-1 and “Create certificates” on page 3-1, including replacing screen captures.
 - Confirmed that **daemonCrypto** and **cryptoCore** have been removed as required modules leaving only a single module: **platCrypto**. See “Prerequisites” on page 2-1 and “Upgrade from crypto.jar” on page 2-11.
 - Removed any recommendation to delete the old **crypto.jar** from the modules folder. This file is deleted automatically. See “Upgrade from crypto.jar” on page 2-11.
 - Updated the section titled “Enable clients and configure them for the correct port” on page 2-9 with an additional step to temporarily approve the unsecured host until one or more certificates are configured. Added a paragraph explaining this use of the **Allowed Hosts** list to “About Allowed Hosts” on page 5-10.
 - Added the step to “Enable SSL” on page 2-4 to right-click and select the action to disconnect all Fox sessions so that the station can reconnect using Foxxs. Also added a question/answer pair about this issue to “Fix error conditions” on page 6-1.
 - Added to this Document Change Log a reference to “3.7 Update 1 (3.7u1)” to clarify that this is a modified version of a release that is already in the field.
 - Performed other document maintenance tasks.
- Updated: February 21, 2013

Noted in several areas that starting in the NiagaraAX 3.7 Update 1 release (3.7u1 or 3.7.104) or later, only the **platCrypto** module is required to be installed in a Hotspot JACE for SSL support. In the initial 3.7 release (3.7.44), modules **cryptoCore** and **daemonCore** were also required. Affected sections are “Prerequisites” on page 2-1, “Install SSL module” on page 2-2, and “Upgrade from crypto.jar” on page 2-10.

More specific details were also given about JACE platforms that *do not support* the “SSL Toolset” described in this document: the JACE-2 series and JACE-4/5 series controllers—which use the IBM J9 VM instead of the required Hotspot VM. Even if upgraded to NiagaraAX 3.7 or later, this group of JACE-2/4/5 controllers must continue to use the station-based CryptoService for SSL. Affected sections were “Prerequisites” on page 2-1 and “Upgrade from crypto.jar” on page 2-10.
- Initial publication: August 30, 2012.

CONTENTS

Document Change Log	v
SSL security for NiagaraAX	1-1
SSL Toolset features	1-1
Frequently-asked questions	1-1
NiagaraAX SSL best practices	1-3
How many keys and certificates do I need?	1-3
Who should sign my certificates?	1-3
Naming conventions	1-3
Security considerations	1-4
Set up Workbench and stations for SSL	2-1
Prerequisites	2-1
SSL setup checklists	2-1
Set up Supervisor checklist	2-1
Set up JACE checklist	2-2
Install SSL module	2-2
<i>Install SSL module</i>	2-3
Check for SSL license	2-3
Move your legacy certificate folder structure (Niagara 3.8)	2-3
Enable SSL for the Supervisor and JACE platforms	2-3
<i>Enable SSL</i>	2-4
<i>Open a secure platform connection (Niagara)</i>	2-5
Enable SSL for the Supervisor and JACE stations	2-7
<i>Enable the Web Service connection</i>	2-7
<i>Confirm the Fox Service connection</i>	2-7
<i>Enable NiagaraNetwork</i>	2-8
<i>Set up client/server relationships</i>	2-9
<i>Enable clients and configure them for the correct port</i>	2-9
<i>Temporarily approve the self-signed certificates</i>	2-9
Configure outgoing email	2-9
<i>Configure secure outgoing email</i>	2-10
Upgrade from crypto.jar	2-10
<i>Upgrade from crypto.jar</i>	2-11
Create certificates	3-1
Set up certificates checklist	3-1
Design the certificate chain of trust	3-2
Create a folder structure	3-2
Establish a naming convention	3-2
Create strong passwords	3-2
Set up the root and intermediate certificates	3-3

- Create the root and intermediate certificates 3-3
- Create a CSR for the intermediate certificate 3-5
- Sign the intermediate certificate using the root certificate's private key 3-6
- Import the intermediate certificate back into the Key Store 3-7
- Export the root and intermediate certificates 3-8
- Set up the JACE and Supervisor server certificates 3-9**
 - Create new JACE and Supervisor server certificates 3-9
 - Create a CSR for each server certificate 3-10
- Sign the server certificates 3-11**
 - Sign the server certificates using the intermediate certificates 3-11
- Import the signed server certificate and configure each station 3-12**
 - Import the server certificate into the Key Store 3-13
 - Enable platform SSL and select the platform server certificate 3-13
 - Select the Https and Foxs service certificates 3-13
- Set up the Trust Stores 3-14**
 - Set up the platform and station Trust Stores 3-14
 - Set up the Workbench Trust Store 3-15
- Install certificates in a client browser 3-15**
 - View and install certificates in Internet Explorer 10 3-15
 - Install a certificate in Firefox 21 3-15
 - Install a certificate in Google Chrome 24 3-15
- Updating a certificate 3-16**
 - Delete a certificate 3-16
- Back up the stores 3-16**
 - To back up the stores 3-16
- Managing allowed hosts 3-16**
- Test station health 3-16**

About SSL—Alice, Bob, Cathy and Bart 4-1

- What makes any system secure? 4-1**
- Friends and enemies 4-1**
 - When trust is secure 4-1
 - When things go wrong 4-2
 - Verifying authenticity 4-2
- Identity verification 4-3**
 - More about certificates 4-3
 - Keys 4-4
 - Signing a certificate with a private key 4-4
 - Creating a chain of trust 4-6
 - More about intermediate certificates 4-7
- Cryptography 4-8**
 - Encrypting the handshake 4-8
 - Data transmission after the handshake 4-8
 - More about key size 4-8

About NiagaraAX SSL 5-1

- NiagaraAX's client/server architecture 5-1**
- About the certificate creation and signing process 5-2**
 - Certificate creation 5-2
 - Workbench certificate signing 5-3
 - JACE certificate signing 5-3
 - Supervisor certificate signing 5-4
 - Setting up the client Trust Stores 5-4
- About NiagaraAX SSL certificates 5-5**

- Types of NiagaraAX certificates5-5
- About self-signed certificates5-5
- Workbench keys and certificates5-7
 - View the Workbench stores* 5-7
- About the SSL Toolset 5-7**
 - About the Key Stores5-8
 - About the Trust Stores5-9
 - About Allowed Hosts5-10
- Troubleshooting6-1**
 - Fix error conditions 6-1**
 - Reset or replace a JACE securely 6-2**
 - To reset or replace a JACE with security* 6-2
- Reference7-1**
 - SSL Toolset terminology 7-1**
 - About TCP ports 7-2**
 - About the Certificate Management view 7-2**
 - About the Key Store tab7-3
 - About the Private Key Password dialog7-5
 - About the Trust Store tab7-5
 - About the Allowed Hosts tab7-6
 - About the Certificate Signing dialog 7-7**
 - SSL configuration properties 7-7**
 - Platform SSL properties7-7
 - Fox Service configuration properties7-8
 - Web Service configuration properties7-9

CHAPTER 1

SSL security for NiagaraAX

The NiagaraAX implementation of the industry-standard Secure Socket Layer (SSLv3) and Transport Layer Security (TLSv1) protocols provides:

- Server authentication
- Encryption of data transmitted between client and server

Note: *NiagaraAX's implementation of SSL does not secure data stored on a storage device. Also, it is not available on any JACE platform using the IBM J9 JVM (JACE-2/4/5 series).*

This topic contains these sub-topics:

- [“SSL Toolset features”](#) on page 1-1
- [“Frequently-asked questions”](#) on page 1-1
- [“NiagaraAX SSL best practices”](#) on page 1-3

SSL Toolset features

- Secure Platform—Niagara over SSL for JACE network controllers that support the HotSpot JVM
- Secure Fox Service—Fox over SSL (Foxs, pronounced “fox-s”)
- Secure Web Service—Http over SSL (https, pronounced “h-t-t-p-s”)
- The creation of self-signed server certificates
- Certificate management for server and Certificate Authority (CA) certificates (both third-party and company-signed certificates) including:
 - Generation of Certificate Signing Requests (CSR)
 - Installation and management of certificates
 - Management of trusted CA certificates
- Allowed Hosts list for manually controlling the hosts (servers) to which a client can connect with a certificate that has not been validated (no matching certificate exists in the client’s **Trust Store**).
- Certificate signing tool
- Outgoing email security

Frequently-asked questions

Q: What is SSL? What is TLS?

A: Secure Sockets Layer (SSL) and Transport Layer Security (TLS) are cryptographic protocols for server authentication and secure encryption of data over the internet.

Q: What is the difference between SSL and TLS? How long have they been out? Is one better than the other?

A: Both standards have been out for a while and both offer the same level of security. The two standards do not compete.

SSL v. 3 is the currently-accepted SSL version. (Version 1 was actually never released. Version 2 had vulnerabilities and is no longer supported by browsers.)

TLS v. 1 is based on SSL v. 3, although the two are incompatible. TLS v. 1.11 and 1.12 provide additional minor feature enhancements.

Q: What organizations support SSL and TLS standards?

A: SSL originated with Netscape Corporation in the 1990s. TLS is developed and promoted by the Internet Engineering Task Force (IETF), a voluntary organization that cooperates closely with the World Wide Web consortium (W3C) and the International Organization for Standardization (ISO)/International Electrotechnical Commission (IEC).

Q: Is an SSL v. 3 key 128-bit based?

A: The number of bits in the key depends on the ciphers used. SSL and TLS allow you to choose the key size. SSL Toolset supports a maximum 4096-bit key.

Q: How does SSL Toolset compare with credit card security?

A: Any technology is as secure as the guidelines followed. For data transport, SSL Toolset provides the same type of security that is provided by the credit card and banking industries.

Q: Our company has multiple locations. Each location has a network of JACEs, none of which is on the internet. How should we ensure that no unauthorized person can intercept communication at any of our sites?

A: You can create your own corporate Certificate Authority (CA) certificate and use its private key to sign a certificate for each location. Then, use the location certificate's private key to sign the certificate for each JACE, and distribute the signed certificate with each JACE or, if the JACEs are already in the field, import the certificate to each JACE's Trust Store. When a JACE comes on line, the handshake validates the entire certificate chain of trust.

Q: Our company already has signed certificates. Can they be used on our JACEs?

A: Yes. When you boot each JACE, the auto-generated default certificate (with its public and private keys) can be used to encrypt communication while you import each platform's company-signed certificate into its Key Store, and the CA (root) certificate (public key only) to its Trust Store. To provide this minimal security, enable SSL using the default (tridium) certificate. To minimize the risk of a man-in-the-middle attack, make an off line, direct connection to each JACE.

Q: Do we have to use the SSL Toolset tools to generate and sign certificates?

A: No. NiagaraAX certificates, public and private keys conform to established standards. You can use any software tool to create them. The tools provided by NiagaraAX are designed to be intuitive and easy to use.

Q: Can the same certificate be used for Foxs, Https, Web Service, and Platform (Niagarad) security?

A: Yes, the same certificate for all three is usually adequate. But, you may have your own reasons for wanting separate security for each service. For example, if you have a lot of people using a station, and the station connection is compromised, a separate certificate for the Niagarad connection would stop the breach immediately.

Q: Does using multiple keys slow performance?

A: Not much. Communication security slows processing in two ways: 1) when generating a complex key on a JACE and 2) during the initial handshake to establish communication. Once communication is established, data are encrypted using a single key, which speeds processing. Actual throughput depends on what you are doing.

Q: Since it takes a long time to generate a key on a JACE, is it acceptable to generate the key on a PC and download it into the JACE?

A: You can generate a key in Workbench on a PC, export it from the PC, and import it into the JACE, but be aware that the transmission may be over a connection that is not secure.

Q: Is there an upgrade path for SSL security from one version of Workbench to the next version?

A: Yes. When you upgrade a Supervisor station, Workbench or JACE, the wizard automatically copies the **security** folder forward assuming that you enabled the option to copy settings from your previous Niagara installation. This folder contains the SSL Key and Trust Stores and the Allowed Host exemptions.

NiagaraAX SSL best practices

Consider these topics before you design your security system:

- [“How many keys and certificates do I need?”](#) on page 1-3
- [“Who should sign my certificates?”](#) on page 1-3
- [“Naming conventions”](#) on page 1-3
- [“Security considerations”](#) on page 1-4

How many keys and certificates do I need?

Each JACE within a Niagara network requires its own unique private key and signed server certificate, which is stored in its **Key Store**.

Each client within a Niagara network requires a copy of the CA (root) certificate used to sign the server's certificate for any servers to communicate securely within the network. These certificates reside in the client **Trust Store**.

One certificate may be used to secure Fox station communication, Web Service and Niagara or each service may have its own server certificate and pair of keys.

The number of certificates you need depends on the number of platforms in your network and the type of communication you need to secure.

For more on how many certificates you may need, see [“Design the certificate chain of trust”](#) on page 3-2.

Who should sign my certificates?

All certificates are signed in one of these ways:

- For a self-signed certificate, the certificate's private key is used to sign its own certificate. This type of signature is not recommended to secure a server because trust cannot be verified.

Note: A self-signed certificate provides only data encryption.

- If your network is self-contained, you can serve as your own Certificate Authority (CA). When a company serves as its own CA, a certificate signed by the private key associated with one of its CA certificates (the root certificate) must be installed in the client browser, and imported into the client platform/station's **Trust Store**.
- If your network is exposed to the internet, a third-party CA provides the most secure communication, however at a cost.

Third-party companies that provide certificate signing services include VeriSign® and Thawte. Certificates that contain only a public key and are signed by the third party are distributed with a user's browser. NiagaraAX Workbench comes with a number of these certificates. No separate step is required to install them, but you must trust that the browser installation was secure. See [“Install certificates in a client browser”](#) on page 3-15).

You have your server certificates signed by creating a Certificate Signing Request (CSR) that contains your subject (Distinguished Name) and your public key. The process of generating a CSR also creates the private key, which must be kept secure. You send the CSR to the Certificate Authority.



Warning Do not send your private key to the CA and do not distribute it via email.

When the CA receives the CSR file, it extracts certain information from the CSR, verifies your identity, creates a new certificate with itself as the Issuer and signs the certificate with its root or an intermediate private key. The CA returns to you the signed client certificate(s) (root and intermediate).

Naming conventions

To keep certificate usage straight, plan your station and certificate names carefully. For station names, include the words “supervisor” and “JACE.” For certificates and Certificate Signing Requests (CSRs), include the station name to which the certificate belongs.

Consider adding the words “Server” or “CA” to differentiate certificate types and “Public” or “Private” to identify certificates that contain the keys. Certificates exported with their private keys should be heavily encrypted and password protected when stored on a company server.

Security considerations

- Computer equipment should be secured in a locked room. Wiring should be protected to prevent an unauthorized person from plugging in to it.
- Software passwords need to be strong. They should also be stored and used securely. Access to the file system needs to be controlled.
- The transmission of data within a network, which may occur over wires or a wireless connection, needs to be secure. This is where SSL Toolset applies. It is concerned with the transport, not the storage of data.
- Third-party certificates, such as those from Certificate Authorities (CAs) VeriSign and Thawte, are installed with the browser. This requires trust in the browser installation program. If your company is acting as a CA, your signed client certificate(s) need to be separately installed in the user's browser.
- A certificate chain of trust allows you to create a CA (root) certificate, have it signed by a CA (or become your own CA), and then use that certificate to sign the certificate generated in each JACE.
- Generating a key and certificate on a JACE is the most secure way to create the certificate for the JACE even though the generation process takes time.
- Although it can be done, it is not recommended to generate a certificate (and key pair) on a local computer, and then download the certificate into the JACE. This is because the download may occur over a connection that is not secure. If you are going to implement SSL on each JACE using a certificate created elsewhere, enable SSL using the default certificate before you import the signed certificate into the station's Key Store. The transmission will at least be encrypted.
- You can export all certificates and keys from Workbench and one or more station(s). While the Key and Trust Stores are backed up with the station, they are not part of the station copy. To import the stores back into Workbench or a station, take the computer running Workbench or take the station off the network, import the stores, and put the computer or station back on the network.
- To access the corporate JACE network from a remote location via the internet, use a VPN solution that incorporates RSA two-factor authentication.
- For high traffic stations (especially stations that provide public access to a JACE network), secure Niagara using a separate certificate from that used for Fox and Web Service.
- Do not mix secure platforms with platforms that are not secure on the same network. Make sure that all JACEs and the Supervisor station(s) are secure.



Caution

To ensure security, do not use self-signed certificates; do not use guest accounts; and do not use the default password. If you connect host stations directly to the public internet, make sure you are using CA signed certificates.

CHAPTER 2

Set up Workbench and stations for SSL

This section explains how to install, enable and configure the SSL Toolset or upgrade to it from NiagaraAX CryptoService. Specifically, it contains these topics:

- “Prerequisites” on page 2-1
- “SSL setup checklists” on page 2-1
- “Install SSL module” on page 2-2
- “Check for SSL license” on page 2-3
- “Enable SSL for the Supervisor and JACE platforms” on page 2-3
- “Enable SSL for the Supervisor and JACE stations” on page 2-7
- “Configure outgoing email” on page 2-9
- “Upgrade from crypto.jar” on page 2-10

Note: *The platform on which SSL Toolset is to be installed or upgraded should be off line or located on a private, closed network. If an existing platform is on the internet or any company network, remove it before you get started.*

Prerequisites

- NiagaraAX 3.7 Update 1 (3.7u1 or 3.7.104) or later.
- A NiagaraAX platform that uses the HotSpot VM (virtual machine) from Sun Microsystems. This includes any of the QNX-based JACE-3/6/7 series controllers, as well as any Windows-based host.
Note: *The QNX-based JACE-2/4/5 series controllers use the IBM J9 VM, and require CryptoService (crypto.jar). For details, refer to the engineering notes document NiagaraAX CryptoService (SSL).*
- This module installed: **platCrypto** (3.7u1 or 3.7.104 or later, only module required)
Note: *In the initial 3.7 release (3.7.44), modules **cryptoCore** and **daemonCrypto** were also required, in addition to **platCrypto**. However, starting in 3.7u1 those two modules are included in a core .dist file for each JACE, and automatically installed in a different location than the modules folder.*
- SSL Toolset license for the Supervisor and each JACE. This is the same license required by NiagaraAX version 3.6 and earlier to use CryptoService.

SSL setup checklists

Note: *If you are upgrading from CryptoService, see “Upgrade from crypto.jar” on page 2-10 before you continue with the checklists.*

- “Set up Supervisor checklist” on page 2-1
- “Set up JACE checklist” on page 2-2

As a best practice, configure SSL for a Supervisor station or a JACE while the units are off line. Use a “crossover” cable from your laptop to the JACE. If you must install SSL while a JACE is online, connect the crossover cable to a secondary port.

Set up Supervisor checklist

1. Supervisor PC off the company LAN and global internet.
2. Supervisor licensed for SSL.
See “Check for SSL license” on page 2-3.
3. Platform (Niagarad) enabled.
See “Enable SSL for the Supervisor and JACE platforms” on page 2-3.

4. SSL modules installed.
See [“Install SSL module”](#) on page 2-2.
5. **NiagaraNetwork** enabled.
See [“Enable NiagaraNetwork”](#) on page 2-8.
6. **Web Service** enabled.
See [“Enable the Web Service connection”](#) on page 2-7.
7. **Fox Service** enabled and port assigned (defaults to 4911).
See [“Confirm the Fox Service connection”](#) on page 2-7.
8. JACE station set up as Supervisor client.
See [“Set up client/server relationships”](#) on page 2-9.
9. SSL enabled for JACE client connection, and JACE client connection port set correctly.
See [“Enable clients and configure them for the correct port”](#) on page 2-9.
10. Certificates created, signed and imported.
See [Chapter 3, “Create certificates”](#).
11. Certificate selected for each service: Foxs, Https and Niagarad.
See [“Import the signed server certificate and configure each station”](#) on page 3-12.

SSL is now set up for the Supervisor. Configuration continues with setting up the JACE platforms and stations ([“Set up JACE checklist”](#) on page 2-2), and certificates ([Chapter 3, “Create certificates”](#)).

Set up JACE checklist

1. JACE off the network, global internet, and connected directly to the computer using a crossover cable.
2. JACE licensed for SSL.
See [“Check for SSL license”](#) on page 2-3.
3. Platform (Niagarad) enabled.
See [“Enable SSL for the Supervisor and JACE platforms”](#) on page 2-3.
4. **NiagaraNetwork** enabled.
See [“Enable NiagaraNetwork”](#) on page 2-8.
5. **Web Service** enabled.
See [“Enable the Web Service connection”](#) on page 2-7.
6. **Fox Service** enabled and port assigned (defaults to 4911).
See [“Confirm the Fox Service connection”](#) on page 2-7.
7. Supervisor set up as JACE client.
See [“Set up client/server relationships”](#) on page 2-9.
8. SSL enabled for Supervisor client connection, and Supervisor client connection port set correctly.
See [“Enable clients and configure them for the correct port”](#) on page 2-9.
9. Certificates created, signed and imported.
See [Chapter 3, “Create certificates”](#).
10. Certificate selected for each service: Foxs, Https and Niagarad.
See [“Import the signed server certificate and configure each station”](#) on page 3-12.

SSL is now set up for the JACE.

Install SSL module

If commissioning a new platform, install this module as part of commissioning:

- **platCrypto**

Before you begin, check the modules folder for **platCrypto**. If the module exists, continue with [“Check for SSL license”](#) on page 2-3, otherwise, see [“Install SSL module”](#) on page 2-3.

Note: In the initial 3.7 release (3.7.44), modules **cryptoCore** and **daemonCrypto** are also required, in addition to **platCrypto**. However, starting in 3.7u1 (3.7.104 and later) those two modules are included in a *core.dist* file for each JACE, and automatically installed in a different location than the modules folder.

Install SSL module

- Step 1 Double-click the **Platform** node in the Nav tree and double-click the **Software Manager**.
- Step 2 Use the wizard to install the module.

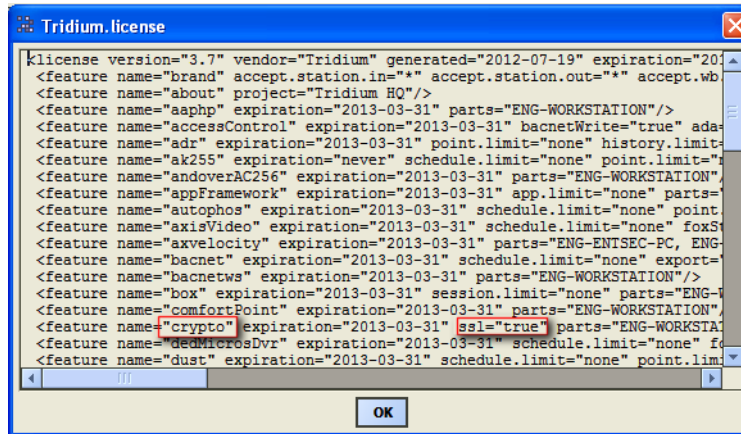
You may need to import the module if it is not in your Workbench software database.

For more information about Software Manager and how to install modules, see “Software Manager” in the NiagaraAX *Platform Guide*.

Check for SSL license

For those upgrading from earlier versions of Workbench, the license to use SSL is the same as the license to use CryptoService. In the station’s **Services > PlatformServices > LicenseService** view, double-click the license to view it.

Figure 2-1 License view



If the **crypto** feature name is included with **ssl="true"**, the platform is licensed to use SSL. For more licensing information, see various topics in the *NiagaraAX Platform Guide*, such as “Workbench License Manager”, “About the local license database”, and “License Manager.”

Move your legacy certificate folder structure (Niagara 3.8)

When you first open Certificate Management on a platform or station (JACE or Supervisor), NiagaraAX 3.8 automatically creates the **certManagement** folder under the **niagara** folder (at the same level as the **licenses** and **modules** folders). You use this location to temporarily store certificate requests (CSRs), as well as intermediate and server certificate .pem files.

Note: *Versions of NiagaraAX prior to 3.8 do not automatically create the certManagement folder. Earlier versions leave it up to you to create a folder structure to hold your CSR and .pem files.*

Do not confuse **certManagement** with the **certificates** folder, which is part of confirming your NiagaraAX license.

If you are upgrading to version 3.8 from a previous version of NiagaraAX, you may cut your previous folder structure and paste it into the automatically-created **certManagement** folder.

Note: *Certificates created under previous versions of NiagaraAX may or may not be usable in the new version. Delete any certificate request (.csr) files once you receive the signed certificates.*



Caution *Do not store certificate files that contain private keys in this or any other folder structure on a computer that is located in an unprotected area.*

Enable SSL for the Supervisor and JACE platforms

Platform and station security are independent of one another. You can configure SSL for only your stations or for both your platforms (Niagard) and stations (Fox).

A station’s “window” into the platform-resident SSL features is just like any other platform service under the station’s **PlatformService** node in the Nav tree. This means that anything configured in **PlatformServices** is independent of whatever station is running.

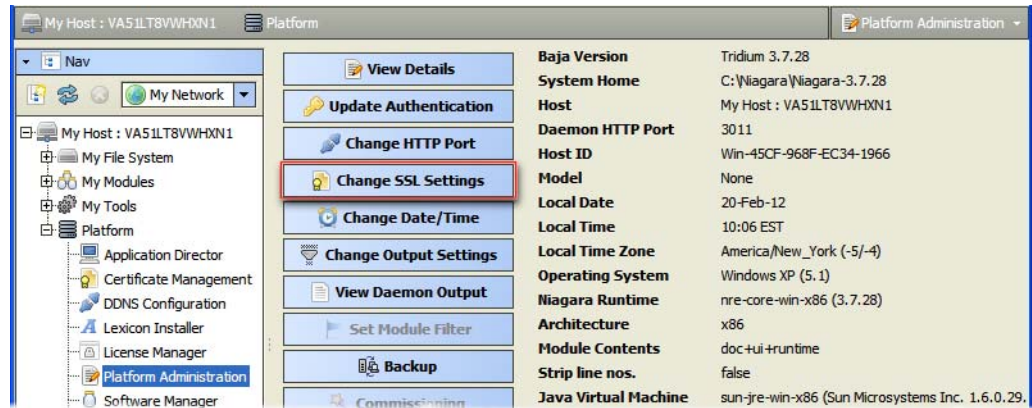
This topic explains how to:

- “Enable SSL” on page 2-4
 - “Open a secure platform connection (Niagarad)” on page 2-5
- To configure SSL for your stations, see “Enable SSL for the Supervisor and JACE stations” on page 2-7.

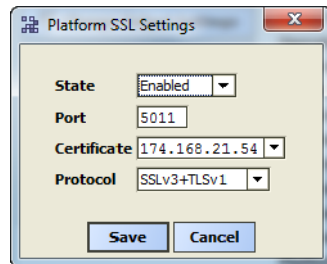
Enable SSL

This procedure involves making an unencrypted platform connection, enabling SSL, disconnecting from the unencrypted platform, and re-connecting using a secure connection.

- Step 1 Make an unencrypted connection to the platform.
- Step 2 Under **Platform** in the Nav tree, double-click **Platform Administration**.
The **Platform Administration** view appears.



- Step 3 Click **Change SSL Settings**.
The **Platform SSL Settings** dialog appears.



The default **Port** for platform connections over SSL is 5011.

Certificate provides a drop-down list of available certificates. Assuming this is a new platform, the only certificate in the list is **tridium**, the auto-generated self-signed certificate.

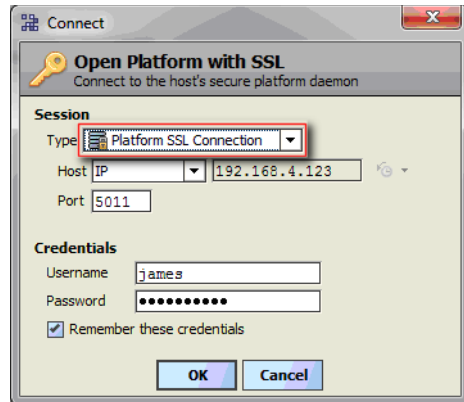
The **Protocol** list allows you to choose one protocol over the other (SSL or TLS). SSL Toolset supports both, which is the default way browsers work. There is no performance reason to choose one over the other. This property is provided in case your situation (contract or agreement) requires you to use one or the other.

- Step 4 Change **State** to **Enabled** and click **Save**.
The system enables the SSL port and restarts the unencrypted connection, most likely using the TLS protocol. This restart occurs for reasons other than security.
- Step 5 Disconnect the Fox connection (right-click the **station** node in the Nav tree and select the action to disconnect all Fox sessions).
- Step 6 Disconnect from the platform session (right-click the platform in the Nav tree and click **Close**).

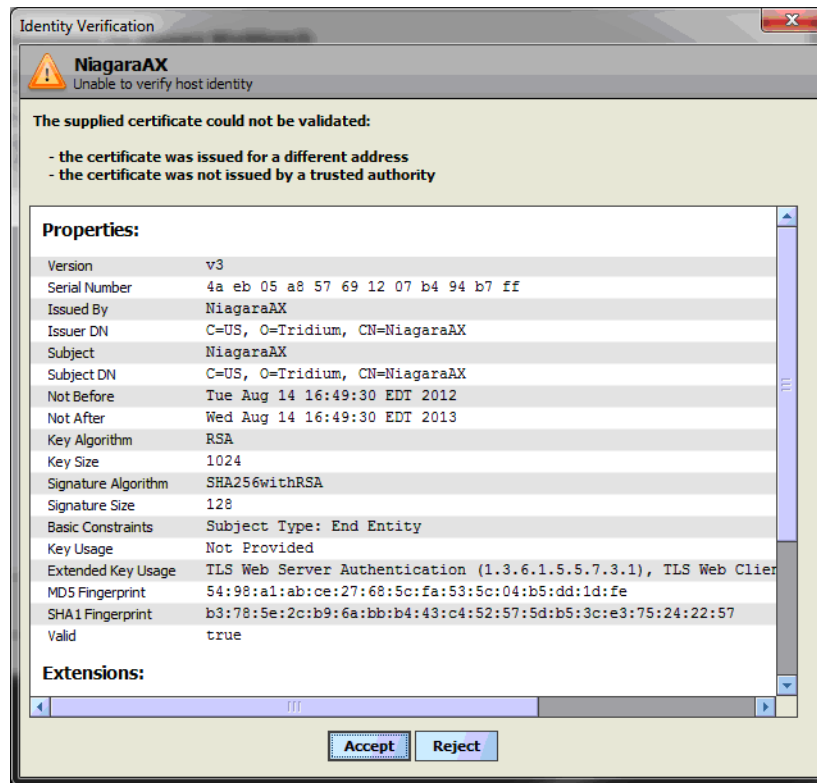
Open a secure platform connection (Niagara)

Now that SSL is enabled, you can open the platform securely.

- Step 1 Click **File > Open > Open Platform**.
The **Open Platform** dialog appears.



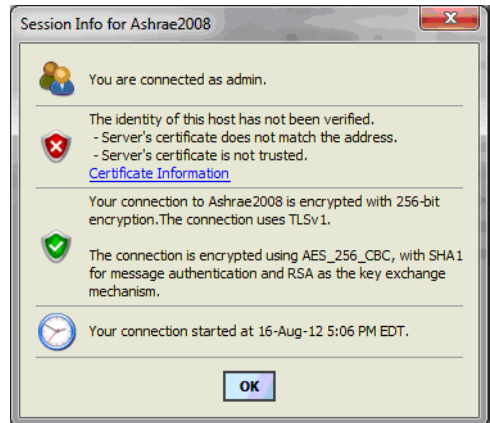
- Step 2 Select **Platform SSL Connection** from the **Session Type** drop-down list.
- Step 3 Define the host IP, enter your **Credentials** and click **OK**.
The system displays the identity verification warning.



This error message is expected for two reasons:

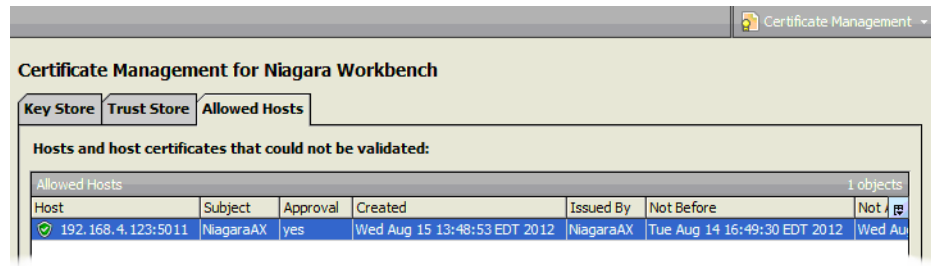
- The certificate's **subject**, or Common Name (CN) is NiagaraAX. This name does not match the host's name, which is usually its IP address or domain name.
- The certificate signature does not match the signature on any certificate in the client **Trust Store**. The fact that the **Issued By** and **Subject** are the same would indicate that the certificate has been self-signed.

- Step 4 Since this is the default **tridium** certificate, which can be trusted, click **Accept**.
Accepting the certificate creates an approved host exception in the **Allowed Hosts** list. If you did not select **Remember these credentials** when you logged in, the system asks you to confirm your platform credentials again.
- Step 5 Enter your credentials and click **OK**.
The platform is now connected over a secure connection. All data transmitted is encrypted, but the server’s identity was not validated.
- Step 6 To confirm this state, right-click **Platform** and click **Session Info**.
The system displays session information.



- The red shield with the X indicates that the software was unable to verify the authenticity of the server certificate. It is a self-signed certificate and no matching CA (root) certificate exists in the platform Trust Store. To view the certificate, click the link.
- The green shield with the check mark indicates that encryption is enabled (this is a secure connection). In this example, the secure connection is using TLSv1 as the protocol and data is encrypted using “AES_128_CBC with SHA1.”

- Step 7 Click **OK**.
The tiny lock on the platform icon in the Nav tree indicates a secure connection.
- Step 8 To view this allowed host, click **Tools > Certificate Management** and check the **Allowed Hosts** tab.



The green shield indicates that the exception is approved.

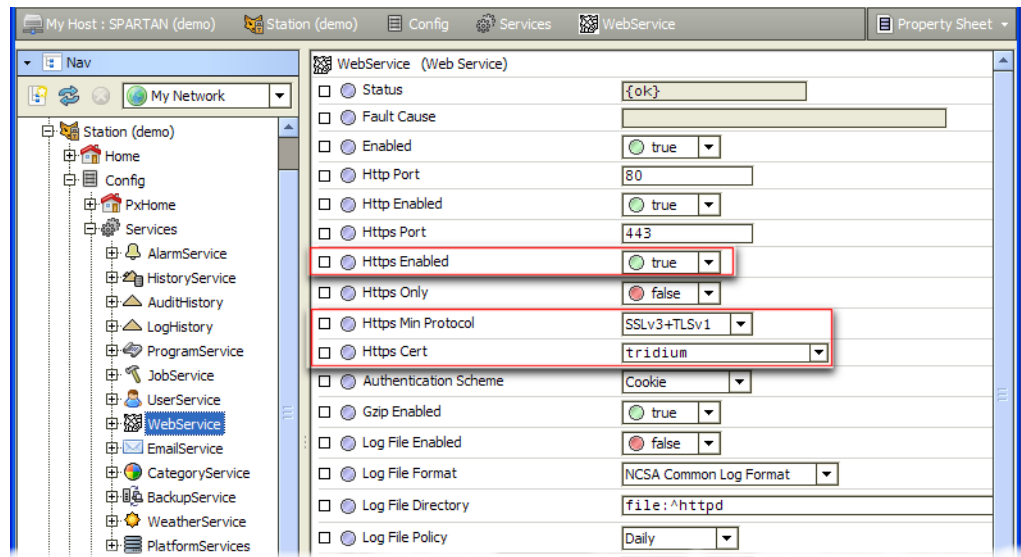
Enable SSL for the Supervisor and JACE stations

This topic explains how to enable SSL and how to turn on security for NiagaraAX Web Service (Https) and Fox Service (Foxs).

- “Enable the Web Service connection” on page 2-7
- “Confirm the Fox Service connection” on page 2-7
- “Enable NiagaraNetwork” on page 2-8
- “Set up client/server relationships” on page 2-9
- “Enable clients and configure them for the correct port” on page 2-9
- “Temporarily approve the self-signed certificates” on page 2-9

Enable the Web Service connection

- Step 1 Connect to the station.
- Step 2 Expand the station’s **Config > Services** node in the Nav tree and double-click **Web Service**. The Web Service properties appear.



- Step 3 Set **Https Enabled** to **true**.

Ideally, you want to set **Http Enabled** to **false** (that is, you want to turn off Http completely) and **Https Enabled** to **true**, but sometimes this configuration is not practical. If you are implementing SSL security in an existing system, you may have many pointers to the old Http port number. Between the highlighted properties above is **Https Only**. If this property is set to **true**, and an attempt is made to connect using Http, NiagaraAX redirects the connection to the secure SSL connection (Https). This saves having to manually change each occurrence of the Http port. For more information, see “[Web Service configuration properties](#)” on page 7-9.

Https Min Protocol is already set to SSL and TLS.

Leave **Https Cert** configured to use the default **tridium** certificate until you have a signed certificate to use here.

- Step 4 Click **Save**.

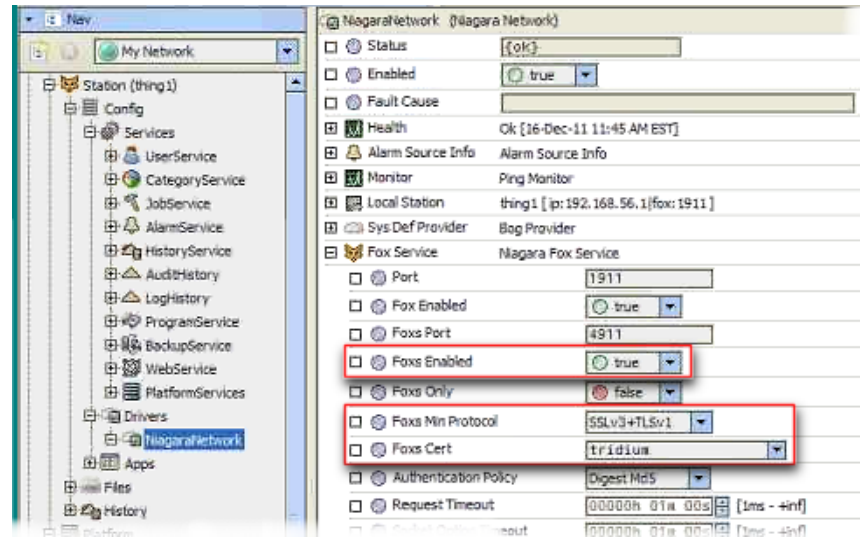
Confirm the Fox Service connection

The Fox Service’s **Foxs Enabled** property is automatically set to **true** when enabling the Web Service Https feature. This procedure verifies that the Foxs Service is enabled.

Note: If a station is configured for Https it must also be configured for Foxs. If either setting is disabled while the other is enabled, an error message appears at login.

- Step 1 Under the station, expand the **Config > Drivers** node in the Nav tree, right-click **Niagara Network**, click **Views > Property Sheet**, and expand the **Fox Service** properties.

The NiagaraNetwork properties contain the Fox Service properties.



Step 2 Notice that **Foxs Enabled** is already set to **true**.
If you enabled **Https** in the **Web Service**, the system automatically enabled **Foxs** in the **Fox Service**; there's nothing to change here.

Ideally, you want to set **Fox Enabled** to **false** (that is, you want to turn off Fox completely) and **Foxs Enabled** to **true**, but sometimes this configuration is not practical. If you are implementing SSL security in an existing system, you may have many pointers to the old Fox port number. Between the highlighted properties above is **Foxs Only**. If this property is set to **true**, and an attempt is made to connect using Fox, NiagaraAX redirects the connection to the secure SSL connection (Foxs). This saves having to manually change each occurrence of the Fox port. For more information, see [“Fox Service configuration properties”](#) on page 7-8.

Step 3 Notice that the same default **tridium** certificate (**Foxs Cert** field), which was used for the platform is selected here.

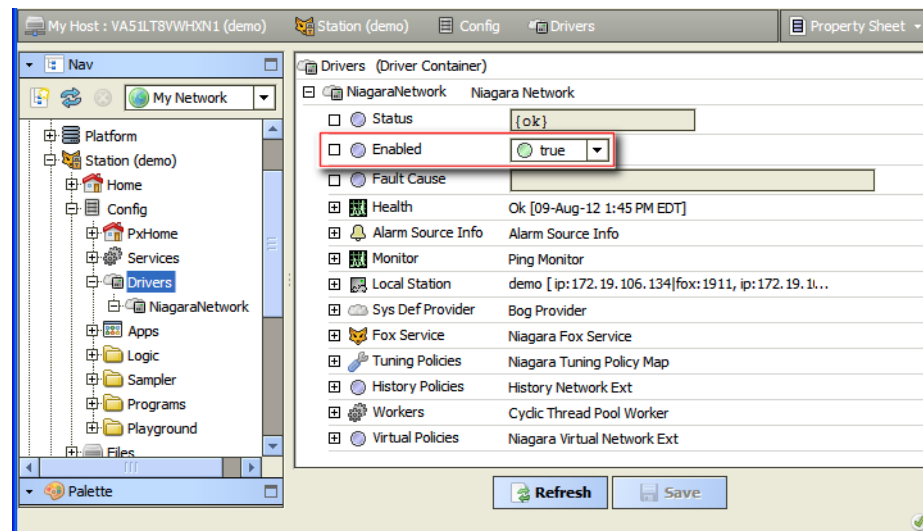
Even though this default name is the same for each JACE, this server certificate is unique to this platform (and station).

If you choose to use a different certificate for your **Fox Service** from that used with your **Web Service**, this is where you would specify it.

Enable NiagaraNetwork

Step 1 Right-click the **Drivers** node in the **Config** folder and click **Views > Property Sheet**.

Step 2 Expand the **NiagaraNetwork** property.

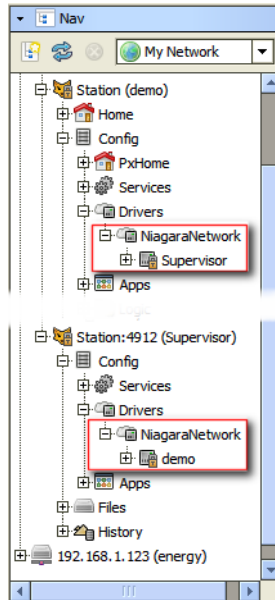


- Step 3 Confirm that **Enabled** is set to **true**.

Set up client/server relationships

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

- Step 1 Expand the **Drivers > NiagaraNetwork** node in the Supervisor Nav tree. It should contain a node for the JACE station.



- Step 2 Expand the **Drivers > NiagaraNetwork** node in the JACE Nav tree. It should contain a node for the Supervisor station.

Enable clients and configure them for the correct port

- Step 1 If it is not already open, double-click the **NiagaraNetwork** node in the Nav tree of both the Supervisor and the JACE 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 JACE station as client; and for the JACE 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**.

Temporarily approve the self-signed certificates

- Step 1 For each client, click **Tools > Certificate Management**.
- Step 2 Click the **Allowed Hosts** tab.
- Step 3 Select the default self-signed certificate and click **Approve**
When certificate configuration is complete, you should delete this approval from the **Allowed Hosts** list for each station. Approving an unrecognized host limits the effectiveness of SSL security and should not be done unless you know for sure that the host is trustworthy.

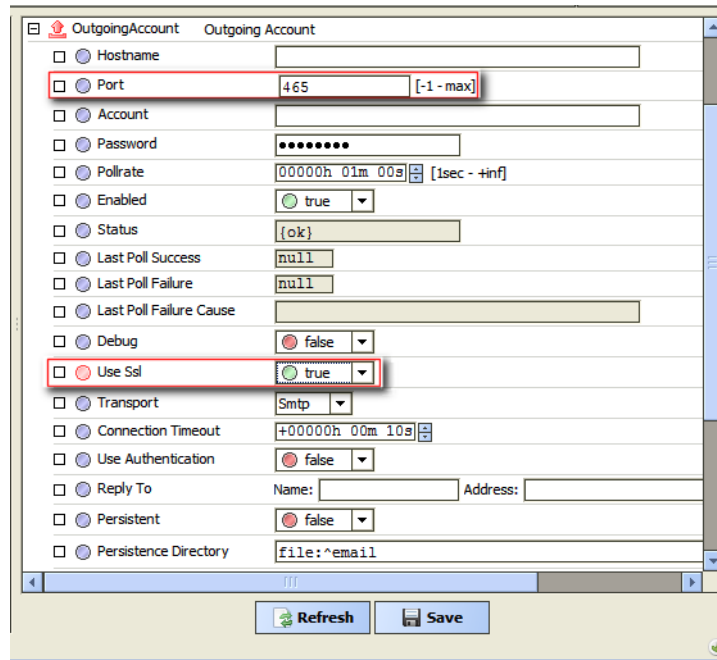
Configure outgoing email

The station acts as a client to a mail server. Email uses the platform Key Store to provide secure outgoing messaging.

- To set up secure outgoing email, see [“Configure secure outgoing email”](#) on page 2-10.

Configure secure outgoing email

- Step 1 Right-click the **EmailService** node under **Services** in the station's Nav tree, then click **Views > Property Sheet** and expand the **OutgoingAccount** properties. The Edit email account dialog appears.



The example shows the configuration when using SMTP.

Incoming and outgoing messages use different ports for SSL encryption as follows:

Table 2-1 SMTP email ports

	Incoming	Outgoing
non-SSL	110	25
SSL	587	465

- Step 2 Set **Use Ssl** to **true**, change the **Port** to the appropriate SMTP SSL port (default 465), and click **Save**.
Note: Do not change the **Use SSL** setting from **true** to **false** or vice versa without changing the **Port**.
- Step 3 If the email server's certificate is signed by a CA whose root certificate is not located in the station's **Trust Store**, you may:
- Import the CA (root) certificate into the station's **Trust Store**. For more information see [“Set up the platform and station Trust Stores”](#) on page 3-14.
 - Or, when challenged, accept the exception creating a record in the **Allowed Hosts** list. For more information see [“Managing allowed hosts”](#) on page 3-16 and [“About the Allowed Hosts tab”](#) on page 7-6. You can import a CA (root) certificate later and delete this temporary exception.

Upgrade from crypto.jar

The CryptoService feature provided by versions of NiagaraAX prior to version 3.7 was station-based. All SSL configuration properties were stored in the station database. Prior to version 3.7, CryptoService supported only SSLv3 Http communication. The Fox Service and Niagarad protocols were not protected.

The SSL Toolset is platform-based and can be configured without a running station. In addition to protecting Http, the SSL Toolset:

- Protects Fox Service and Niagarad.
- Provides for the creation and signing of certificates for each NiagaraAX service (Fox, Http and Niagarad).
- Makes it possible to deny stations that are not secure.
- Offers a choice of cryptographic protocols to use (SSL or TLS).
- Allows any port to be changed.

If you have been using the CryptoService with a version of NiagaraAX prior to version 3.7 on a platform that supports the HotSpot VM (for example a JACE-6, JACE-7, JACE-6E, or any Windows-based host), follow this general procedure to upgrade to the SSL Toolset.

- “Upgrade from crypto.jar” on page 2-11

Upgrade from crypto.jar

Note: *You cannot upgrade a JACE to the SSL Toolset if it supports only the IBM J9 VM, for example, the software is running on a JACE-2 or JACE-4/5 series controller. A station running on a JACE-2/4/5 must continue to use the CryptoService for web SSL (Https using crypto.jar), even if it has been upgraded to NiagaraAX 3.7 Update 1 (3.7u1 or 3.7.104) or later.*

- Step 1 Back up and save the station.
- Step 2 Put the platform on a safe, secure network, such as a private, closed network.
- Step 3 Right-click **CryptoService** in the station **Services** node of the Nav tree and click **Delete**.
- Step 4 Save the station.
- Step 5 Stop the station.
- Step 6 Remove **crypto.jar** from the platform using the platform **Software Manager**.
- Step 7 Run the platform **Commissioning Wizard** to upgrade the platform.

When selecting software modules, make sure that **platCrypto** is selected for installation.

*In the initial 3.7 release (3.7.44), modules **cryptoCore** and **daemonCrypto** are also required, in addition to **platCrypto**. However, starting in 3.7u1 (3.7.104 or later) those two modules are included in a core .dist file for each JACE, and automatically installed in a different location than the modules folder.*

When commissioning is complete, the platform reboots and the station starts. Continue with [Chapter 3, “Create certificates”](#).

CHAPTER 3

Create certificates

The SSL Toolset provides the mechanism to create two types of certificates:

- A server certificate for each JACE and Supervisor platform.
- A Certificate Authority (CA) certificate that you can use to sign other certificates in a certificate chain of trust. Ideally, CA certificates are generated and kept in a vault where there is no network connectivity.

Note: *SSL Toolset provides an easy-to-use interface for creating these certificates, but you may use other tools to create your certificates.*

Third-party CAs, such as VeriSign, are unlikely to sign your CA certificates. This section explains how to become your own Certificate Authority, and includes:

- [“Set up certificates checklist”](#) on page 3-1
- [“Design the certificate chain of trust”](#) on page 3-2
- [“Set up the root and intermediate certificates”](#) on page 3-3
- [“Set up the JACE and Supervisor server certificates”](#) on page 3-9
- [“Sign the server certificates”](#) on page 3-11
- [“Import the signed server certificate and configure each station”](#) on page 3-12
- [“Set up the Trust Stores”](#) on page 3-14
- [“Install certificates in a client browser”](#) on page 3-15
- [“Updating a certificate”](#) on page 3-16
- [“Back up the stores”](#) on page 3-16
- [“Managing allowed hosts”](#) on page 3-16
- [“Test station health”](#) on page 3-16

Set up certificates checklist

This checklist assumes that within a given platform, a single server certificate will secure Niagara, Foxs and Https. If your implementation involves a separate certificate for each service, repeat the checklist as needed for each service.

1. Folder structure under the **certManagement** folder created and naming convention designed.
See [“Create a folder structure”](#) on page 3-2.
See [“Establish a naming convention”](#) on page 3-2.
2. Root and intermediate certificates created.
See [“Create the root and intermediate certificates”](#) on page 3-3.
See [“Create a CSR for the intermediate certificate”](#) on page 3-5.
See [“Sign the intermediate certificate using the root certificate’s private key”](#) on page 3-6.
See [“Import the intermediate certificate back into the Key Store”](#) on page 3-7.
See [“Export the root and intermediate certificates”](#) on page 3-8.
3. Server certificate created for each JACE.
See [“Create new JACE and Supervisor server certificates”](#) on page 3-9.
See [“Create a CSR for each server certificate”](#) on page 3-10.
See [“Sign the server certificates using the intermediate certificates”](#) on page 3-11.
See [“Import the server certificate into the Key Store”](#) on page 3-13.
4. Certificates imported into client Trust Stores.
See [“Set up the platform and station Trust Stores”](#) on page 3-14.
See [“Set up the Workbench Trust Store”](#) on page 3-15.

5. Certificate to use for each service selected.
See [“Import the signed server certificate and configure each station”](#) on page 3-12.

Design the certificate chain of trust

To be your own CA, you will need a root certificate and possibly one or more intermediate certificates as well as a server certificate for every Supervisor and JACE in the network. The private key of your root certificate will be used to sign any intermediate certificates, which, in turn, will be used to sign your server certificates. Before you begin, consider these questions.

- How many intermediate certificates do you need? You might break them down by geography or department. Using intermediate certificates improves security. If one key is compromised, only the compromised chain is at risk. The rest of your network remains secure.
- How many Supervisor stations do you have?
- How many JACEs do you have?
- Can the Supervisor and JACE platform/stations use the default server certificate with the 1024-bit key pair or do you need to create a more secure key pair for each?

Creating a certificate chain of trust also involves setting up the Workbench, Supervisor and JACE Key and Trust Stores. What to import into each entity depends on the function of the entity. As noted in [“NiagaraAX’s client/server architecture”](#) on page 5-1, a given entity may serve as a client or a server. For an illustration of what goes in each store, see [“About the certificate creation and signing process”](#) on page 5-2.

This section includes these topics:

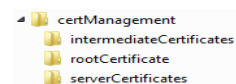
- [“Create a folder structure”](#) on page 3-2
- [“Establish a naming convention”](#) on page 3-2
- [“Create strong passwords”](#) on page 3-2

Create a folder structure

If you are using a version of NiagaraAX earlier than version 3.8, use Windows to set up a folder structure on your computer’s hard disk to hold your Certificate Signing Requests (CSRs) and signed certificates (.pem files). You can use the same structure illustrated in [Figure 3-1](#) or name your own structure.

If you are using NiagaraAX 3.8, the first time you access the Certificate Management view, NiagaraAX creates an empty **certManagement** folder in the **niagara** folder. Within this folder, use Windows to create a structure similar to the following:

Figure 3-1 Example of a folder structure for certificates



Do not confuse the **certManagement** folder with the **certificates** folder that stores one or more certificates used to validate the authenticity of NiagaraAX licensing files. The **certificates** folder has nothing to do with SSL security.

Establish a naming convention

The **Key** and **Trust Stores** form the heart of the SSL Toolset. The **Alias** field in the **Key Store** is a name you can use to differentiate your certificates and keys. Since certificates look a lot alike, use different aliases to differentiate the purpose of each certificate. Consider including the words “root” and “intermediate” in the alias name.

Create strong passwords

A password is required to protect each certificate’s private key. When backing up certificates, an encryption password may also be used. For security’s sake your passwords need to be strong.

A strong password:

- Has eight or more characters.
- Includes letters, punctuation, symbols, and numbers.
- Is different for each password. Don’t use any password more than once.
- Avoids common password pitfalls, such as dictionary words in any language, words spelled backwards or that use common misspellings and abbreviations, sequences or repeated characters, personal information such as your birthday, driver’s license, passport number, etc.

This password information was adapted from information at microsoft.com, which provides a secure password checker you can use to test the strength of your passwords.

Set up the root and intermediate certificates

The procedures in this topic explain how to create a CA (root) certificate and a single intermediate certificate. The Workbench steps to create root and intermediate certificates are functionally the same. The content of each certificate is what differentiates them from one another.

The root certificate is a special case because it may be self-signed or signed by a third-party CA. If you are serving as your own CA, your root certificate is always self-signed. You use it to sign your intermediate certificates (or your JACE server certificates if you are not using intermediate certificates) and export it with only its public key for importing into each client **Trust Store**.

The topic includes:

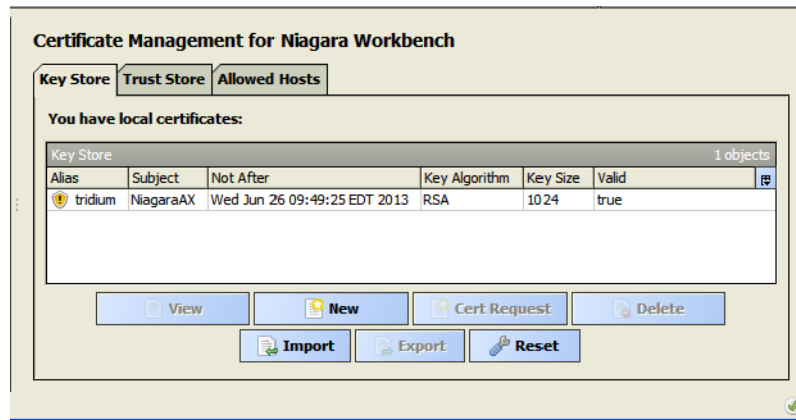
- “Create the root and intermediate certificates” on page 3-3
- “Create a CSR for the intermediate certificate” on page 3-5
- “Sign the intermediate certificate using the root certificate’s private key” on page 3-6
- “Import the intermediate certificate back into the Key Store” on page 3-7
- “Export the root and intermediate certificates” on page 3-8



Caution To ensure the security of your network, always perform these tasks using a computer that is disconnected from the internet and company network. It is recommended to maintain this computer in a secure physical location.

Create the root and intermediate certificates

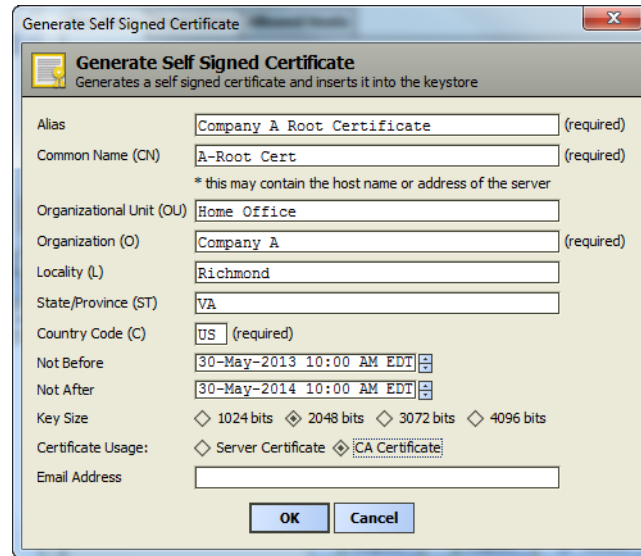
- Step 1 If it is not already displaying, click **Tools > Certificate Management**.
The Workbench **Certificate Management** view appears with the focus on the **Key Store** tab.



- Step 2 Check the title at the top of the **Certificate Management** view to ensure that you are viewing the Workbench **Key Store** and not a JACE **Key Store**.
The Workbench and platform/station stores are separate.
The **tridium** certificate shown in was automatically generated when you started Workbench. This is a default self-signed server certificate.

- Step 3 Click

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



Step 4 Fill in the fields. **Alias** should identify root and intermediate certificates by company, and geography or department respectively.

Common Name (CN) is the same as Distinguished Name and can be the same as the **Alias**.

The two-digit **Country Code** is required.

For more information about each field, see [“About the Generate Self-Signed Certificate dialog” on page 7-4](#).

Step 5 Select the **CA Certificate** property for **Certificate Usage**.


Step 6 Click **OK**.

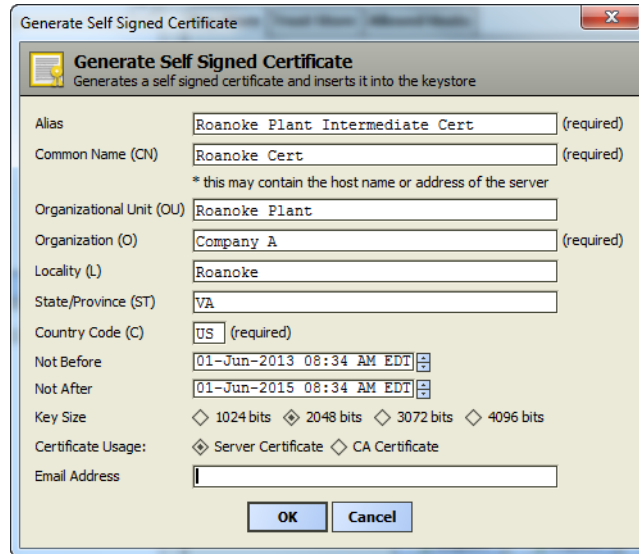
The system prompts you to create a password for the certificate’s private key.



This password protects the private key and is required when using the certificate to sign other certificates. Create strong passwords.

Step 7 Type and confirm the private key password, and click **OK**.

Step 8 To view the certificate, double-click it or select it and click .



is an example of an intermediate certificate. Notice the word “intermediate” is included in the **Alias**.

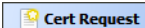
Step 9 Confirm that the information is correct.

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.

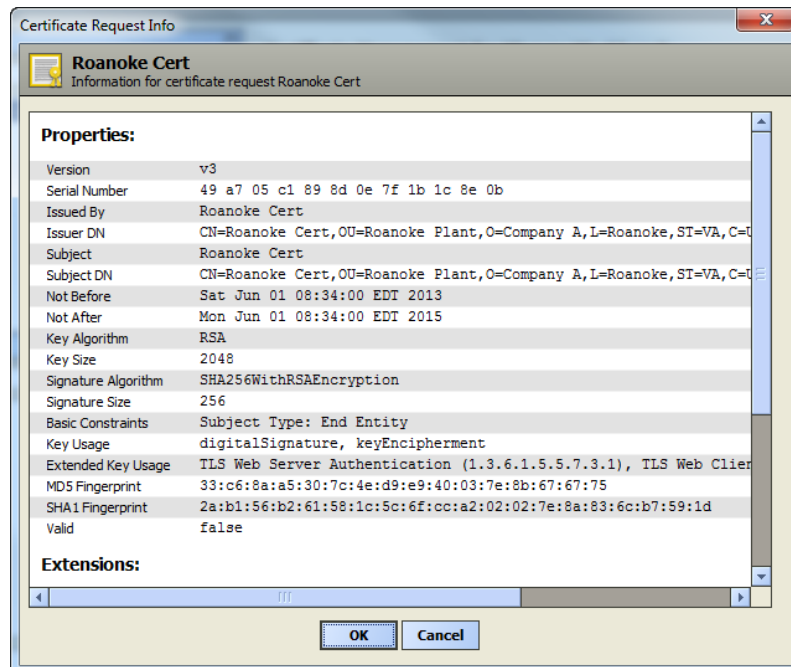
Step 10 When you have created your root certificate, repeat this procedure to create any intermediate certificates.

Create a CSR for the intermediate certificate

A Certificate Signing Request (CSR) prepares the intermediate certificate to be signed by the root certificate. You don’t need to create a CSR for the root certificate unless it will be signed by a third-party Certificate Authority.

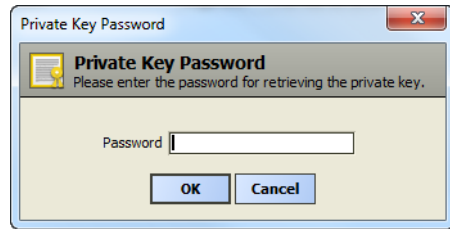
Step 1 Select the intermediate certificate you just created, and click .

Step 2 The **Certificate Request Info** view appears.



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

Certificate Management prompts you for the private key password.



Step 4 Type the password you defined when you created the certificate and click **OK**.

Step 5 Select the folder for intermediate certificates you created in the planning step and click **OK**.

The **Alias** for the certificate is used as the file name with the extension: .csr and the Certificate Manager prompts you to complete the CSR by clicking **OK**.



Step 6 To confirm completion, click **OK**.

Step 7 Repeat this procedure for each intermediate certificate.

Note: *If an external Certificate Authority, such as VeriSign or Thawte, will sign your root certificate, follow the CSR submission procedure as required by the CA. They will verify that you are who you claim to be, that this certificate is for a server that your organization actually maintains, and other important information. They will then return the signed certificate to you.*

Sign the intermediate certificate using the root certificate's private key

This procedure uses the Workbench SSL tools and the root certificate you created to sign your intermediate certificates.

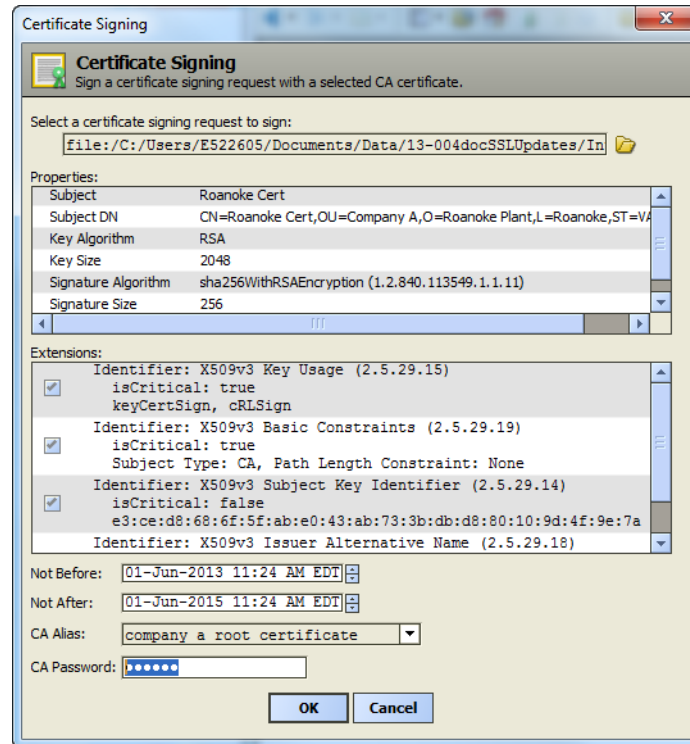
Step 1 In Workbench, select **Tools > Certificate Signer Tool**.

The **Certificate Signing** dialog appears.



Step 2 Click the browser icon, locate, and open a CSR for an intermediate certificate you created.

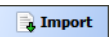

The **Certificate Signing** dialog expands to show the certificate details.



- Step 3 Confirm that this is the intermediate certificate you created.
- Step 4 Select the date on which the certificate becomes effective (**Not Before**) and the date after which it expires (**Not After**).
- Step 5 Select the root certificate for **CA Alias**, type the root certificate’s private key password for **CA Password**, and click **OK**.
Signing is done by the private key of the root certificate, which is why the password you created for the root certificate’s private key is required.
Repeat this procedure for each intermediate CA certificate.

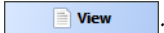
Import the intermediate certificate back into the Key Store


The next step is to import the newly-signed certificate back into the Key Store to complete the process changing the shield icon from yellow (with an exclamation mark) to green (with a check mark).

- Step 1 To view the Workbench **Key Store** click **Tools > Certificate Management**.
- Step 2 Click , locate the intermediate certificate’s .pem file and click .
The Certificate Manager asks you to supply and confirm the certificate’s password.



- Step 3 Enter the intermediate certificate’s private key password, confirm the password, 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.


Note: The certificate you import back into the Workbench **Key Store** must match the original **Alias**. To view the contents of a certificate, select the certificate in the Key Store and click .

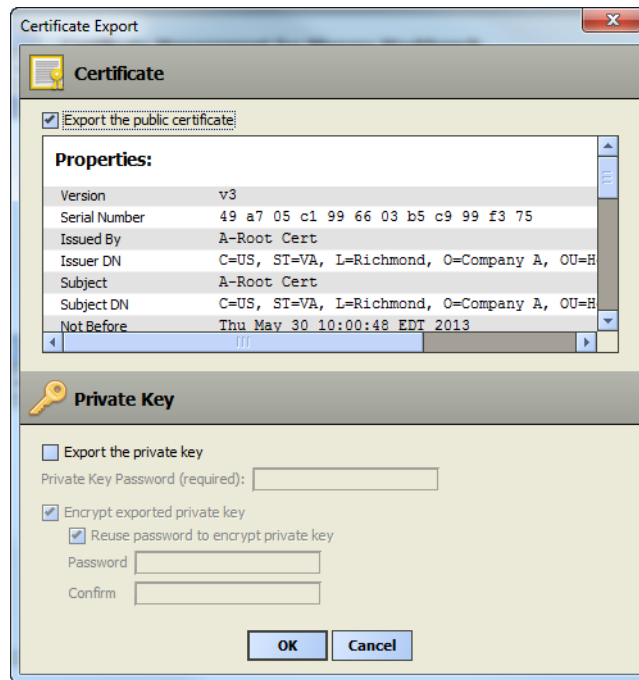
- Step 4 If needed enter the **Alias**.
The **Certificate Import** dialog appears.
 - Step 5 Confirm that this is the certificate you expect and click **OK**.
The green shield icon  appears next to the certificate **Alias** in the **Key Store**.
- Note:** *If your root certificate was signed by an external CA, you will need to follow this same procedure to import it back into the Workbench **Key Store**.*
- Repeat this procedure for each intermediate certificate.

Export the root and intermediate certificates

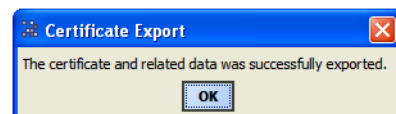
There are two reasons to export certificates:

- To import the root certificate into the **Trust Store** of each client and browser.
- To back up your root and intermediate certificates with their private keys.

- Step 1 On the **Key Store** tab, select the certificate and click  .
The system displays the Certificate Export dialog.



- Step 2 To back up a certificate with its private key, click **Export the private key** box and supply the private key password.
In addition to the private key password, an encryption password can be used to provide double-password protection. The default encryption password is the same as the private key password.
- Step 3 To use the additional protection, deselect **Reuse password to encrypt private key** under **Encrypt exported private key** and supply the additional encryption password.
- Step 4 To export the certificate, click **OK**, locate the root or intermediate **certManagement** folder and click **Save**.
The system reports that the export was successful.



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

Set up the JACE and Supervisor server certificates

For each JACE, follow these procedures using Workbench running on a computer that is connected by a crossover cable to the JACE.

For each Supervisor station, disconnect the Supervisor station from the internet and company network before following these procedures.

- “Create new JACE and Supervisor server certificates” on page 3-9
- “Create a CSR for each server certificate” on page 3-10

Create new JACE and Supervisor server certificates

You need a server certificate for each JACE and Supervisor in the network. All signed server certificates reside in the specific server’s Key Store.

There are multiple ways to create a server certificate.

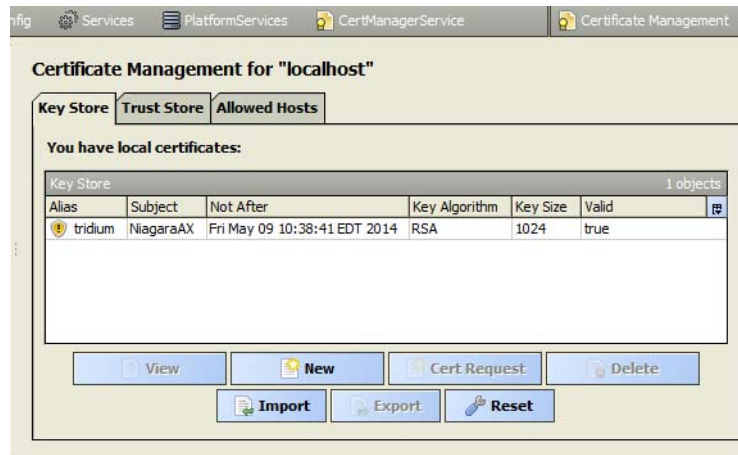
- For each JACE, you can use the default server certificate that is automatically generated when you boot the JACE for the first time.
- If you are connected to a JACE using a crossover cable, you can use **PlatformServices** to create a new server certificate.
- Using Workbench on a secure computer, you can create a server certificate. This procedure creates a new 2048-bit server certificate on a JACE.

Step 1 Launch Workbench and connect to the JACE station using Foxs.

Note: *Workbench issues warnings if managing certificates via an unencrypted Fox connection.*

Step 2 Locate the JACE station in the Nav tree and double-click **CertManagerService** under **Config > Services > PlatformServices**.

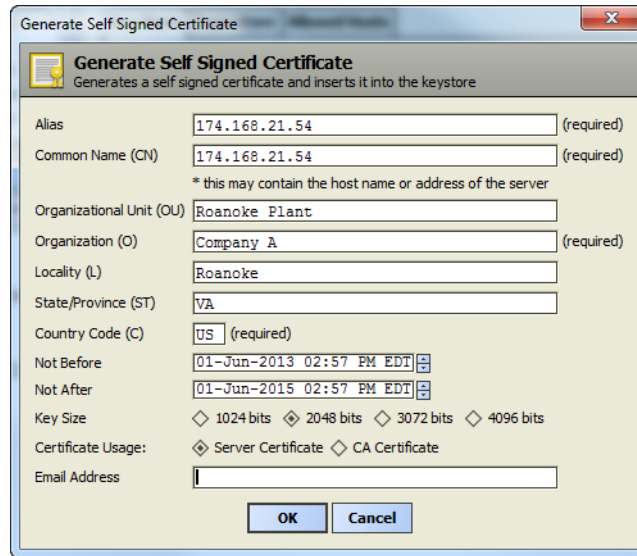
The **Certificate Management** view appears with the focus on the **Key Store** tab.



Step 3 Check the title at the top of the **Certificate Management** view to ensure that you are viewing the JACE’s **Key Store** and not the Workbench **Key Store** (in this case “localhost”), then click



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



Step 4 Fill in the fields and click **OK**.

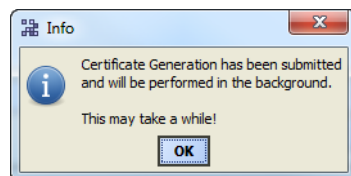
Common Name (CN) is the same as Distinguished Name and can be the same as the **Alias**. Follow these recommendations:

- Create a name that identifies the JACE. You might use the JACE’s IP address or a location code. Do not use the same **Common Name** that you also used for the root or intermediate certificates.
- The **Common Name** should match host name, which is how the server identifies itself. The IP address of the JACE or domain name may be an appropriate **Alias** and **Common Name** for a JACE. The **Common Name** becomes the **Subject** in the certificate.

For more information, see “[About the Generate Self-Signed Certificate dialog](#)” on page 7-4.

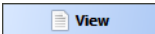
Note: **Certificate Usage** defaults to **Server Certificate** and **Key Size** defaults to **2048**. A larger key takes longer to generate, but improves security. If a third-party will sign the certificate, consult with your CA to determine the acceptable key size. Some CAs support a limited number of key sizes.

A pop-up in the lower right corner indicates certificate creation success or failure, Workbench displays an information message, and adds the certificate to the **Key Store**.



Step 5 Click **OK** to close the **Info** message.

The length of time it takes to generate the certificate depends on the key size and the platform. When finished, you will have a certificate and key pair (public and private keys).

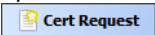
Step 6 To view the certificate, double-click it or select it and click .

Step 7 Confirm that the information is correct.

To change a certificate, you must delete it and create a new certificate.

Create a CSR for each server certificate

For each server certificate to be signed by an intermediate certificate (or the root certificate if your installation is small and does not require intermediate certificates), a Certificate Signing Request (CSR) is required. This procedure is the same if you are using the default server certificate or a server certificate that you created.

Step 1 While you are securely connected to the JACE and are viewing the station’s **Key Store**, select the certificate and click .

The **Certificate Request Info** view appears.

Step 2 Confirm that the certificate properties are correct.

- Step 3 To save the CSR, click **OK**, select the folder for server certificates on your computer, and click **OK**.
The system uses the **Alias** as the certificate file name and the extension of: .csr. This file does not contain the certificate's unique private key.
- Step 4 Copy the CSR to a flash drive or store it on the laptop Workbench computer for transport to the secure, standalone Workbench computer that contains the root and intermediate certificates for signing.
- Step 5 Copy the files into the **certManagement** folder.

Sign the server certificates

Each CSR on the flash drive or laptop Workbench computer needs to be signed by the private key of the root or an intermediate certificate. Although each server certificate CSR does not contain its private key, transportation to the Workbench computer on which the root and intermediate certificates are stored should be secure.



Caution *To ensure the security of your network, always sign certificates using a computer that is disconnected from the internet and company network. It is recommended to maintain this computer in a secure physical location.*

See “[Sign the server certificates using the intermediate certificates](#)” on page 3-11.

Sign the server certificates using the intermediate certificates

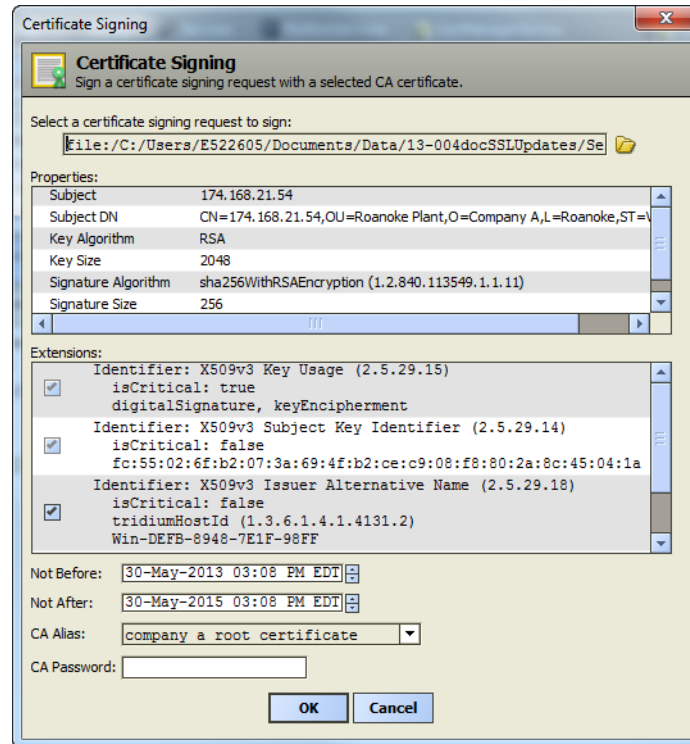
This procedure uses the Workbench tools and the intermediate certificates to sign server certificates.

- Step 1 Select **Tools > Certificate Signer Tool**.
The **Certificate Signing** dialog appears.



- Step 2 Click the browser icon, locate, and open the CSR for a server certificate. and click **Open**.

The **Certificate Signing** dialog expands to display the certificate details.



- Step 3 Confirm that this is the correct server certificate.
- Step 4 Select valid dates.
- Step 5 Select the intermediate or root certificate to use to sign the server certificate, type the root or intermediate certificate’s private key password for **CA Password** and click **OK**.
This generates a new certificate file with the extension of: .pem. This file contains only the public key associated with the certificate.
- Step 6 Save the signed certificate in the server **certManagement** folder.
- Step 7 Repeat this procedure for each server certificate.

Import the signed server certificate and configure each station

When signing is complete, the .pem files for the server certificates need to be transported (on a flash/ thumb drive or laptop computer) back to where they came from and stored in the folder you set up for server certificates. Even though these files do not contain their private keys, you should transport them securely between locations.

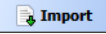


These tasks remain to set up SSL on a JACE or Supervisor:

- [“Import the server certificate into the Key Store”](#) on page 3-13
- [“Enable platform SSL and select the platform server certificate”](#) on page 3-13
- [“Select the Https and Foxs service certificates”](#) on page 3-13

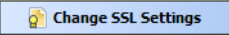
Import the server certificate into the Key Store

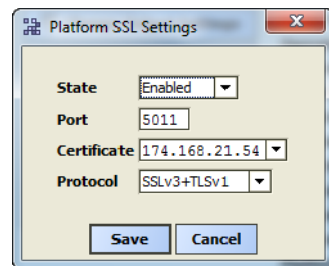
The JACE platform and station share the same Key and Trust stores. This procedure demonstrates how to use station **PlatformServices** to import a signed server certificate back into each platform **Key Store**.

Note: The certificate you import back into the platform must match the original **Alias**.

- Step 1 Connect to the station using a Foxs connection, otherwise Workbench displays a warning message.
- Step 2 In Workbench, click **Station > Config > Services > PlatformServices > CertManagerServices**.
Certificate Management opens with the focus on the **Key Store**.
- Step 3 Click  , locate the certificate .pem file and click  .
Certificate Import displays the certificate details.
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 Confirm that this is the certificate you expect and click **OK**.
The green shield icon  appears next to the certificate **Alias** in the **Key Store**.

Enable platform SSL and select the platform server certificate

- Step 1 Make a secure connection to the platform.
- Step 2 Double-click the platform node in the Nav tree and double-click **Platform**.
- Step 3 Double-click **Platform Administration**.
- Step 4 Click  .
The **Platform SSL Settings** dialog appears.



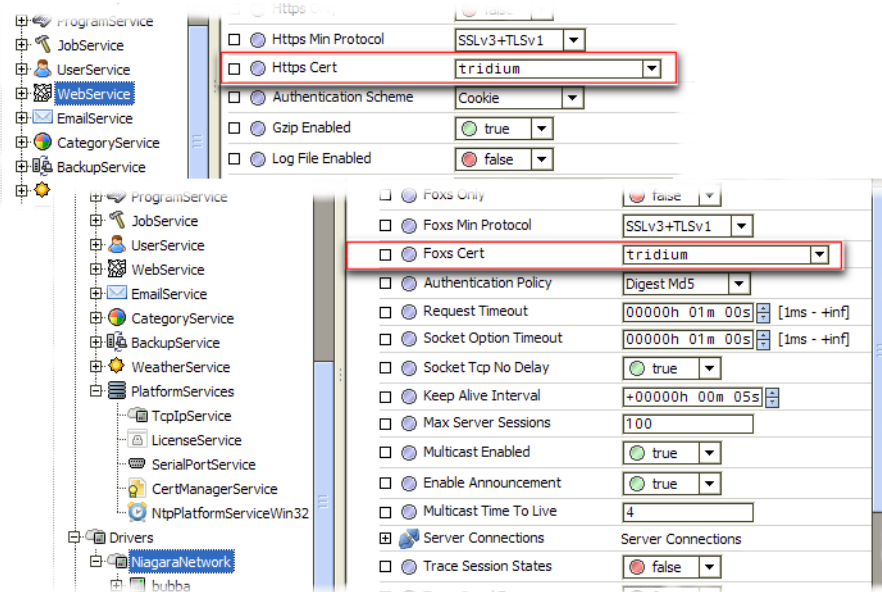
- Step 5 Change **State** to **Enabled**, change the **Certificate** to the certificate you created, and click **Save**.

Select the Https and Foxs service certificates

This procedure assumes you have enabled Https and Foxs. For more information about enabling the services, see [Chapter 2, “Set up Workbench and stations for SSL”](#).

- Step 1 Make a secure connection to the station.
- Step 2 Select the property sheet for the service in the Nav tree:
 - For Web Service select **Station > Config > Web Service**.
 - For Fox Service select **Station > Config > Drivers > Niagara Network**.

The property sheet appears.



- Step 3 Select the certificate to use for the **Https Cert** from the drop-down list.
- Step 4 Select the certificate to use for the **Foxxs Cert** from the drop-down list.

Set up the Trust Stores

Each JACE and Supervisor can also serve as a client. Workbench and the browser are always clients. This topic explains how to set up the **Trust Stores**.

- “Set up the platform and station Trust Stores” on page 3-14
- “Set up the Workbench Trust Store” on page 3-15

In addition to the above Trust Stores, you may also need to import certificates into the Java Trust Store (**Java Control Panel > Security > Certificates**). This is only necessary when connecting in a browser with a Workbench profile that uses a Java applet.



Caution


If your only recourse is to email a root certificate, use a heavily encrypted ZIP file and communicate the ZIP file password over the phone. Otherwise, the receiver of the ZIP file has no way to verify that the ZIP contains the expected certificate.

*You could also tell the recipient what the MD5 and SHA 1 fingerprints are so they can verify the values before using the certificate. To see a certificate's MD5 and SHA 1 fingerprints, select the certificate in the Key Store and click **View**.*

If you are installing a brand new network, certificate transfer can be done in the shop during initial commissioning of a group of JACEs that will be later installed on site (commission the JACEs first and then install the certificates).

Set up the platform and station Trust Stores

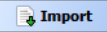

The only certificate to import into the Trust Store for each JACE is the root certificate. Server certificates that have been signed by an intermediate certificate carry the intermediate certificate's information and public key with them.

- Step 1 To view the JACE or Supervisor station's **Trust Store** using a Foxxs connection, click **PlatformServices > CertManagerService** in the Nav tree.
- Step 2 Click the **Trust Store** tab.
- Step 3 Click  and locate the root certificate .pem file and click . **Certificate Import** displays the certificate details.
- Step 4 Confirm that this is the certificate you expect and click **OK**.

The certificate appears in the **Trust Store**. All servers that have server certificates signed by the private key associated with this certificate will be trusted automatically.

Follow this procedure for each certificate to be trusted by a client platform/station.

Set up the Workbench Trust Store

- Step 1 To view the Workbench **Trust Store** click **Tools > Certificate Management** and click the **Trust Store** tab.
- Step 2 Click  and locate the certificate .pem file and click . **Certificate Import** displays the certificate details.
- Step 3 Confirm that this is the certificate you expect and click **OK**.
The certificate appears in the **Trust Store**. All servers that have server certificates signed by the private key associated with this certificate will be trusted automatically.
Follow this procedure for each certificate to be trusted by Workbench.

Install certificates in a client browser

Installing your signed client certificate in the browser ensures that security will be enabled automatically. Installing a certificate in a browser varies by browser, and by browser version. Use these procedures as examples. The steps you will follow may vary.

- “[View and install certificates in Internet Explorer 10](#)” on page 3-15
- “[Install a certificate in Firefox 21](#)” on page 3-15
- “[Install a certificate in Google Chrome 24](#)” on page 3-15

View and install certificates in Internet Explorer 10

Each browser maintains the list of approved certificates. How to view this list varies with the browser. To confirm that the NiagaraAX certificate for https is stored in Internet Explorer 10's approved list:

- Step 1 Launch Internet Explorer and click **Tools > Internet Options**.
- Step 2 On the **Content** tab click the **Certificates** button.
- Step 3 On the **Certificates** tab click the **Trusted Root Certification Authorities** tab and look for the name of your root CA certificate in the list.
- Step 4 Click the **Intermediate Certification Authorities** tab and look for the name of your intermediate CA certificate in the list.
- Step 5 Click **Import...** and follow the wizard to import a certificate.

Install a certificate in Firefox 21

- Step 1 Using the Firefox menu in the upper left corner of the page, click **Tools > Options > Advanced**.
- Step 2 Click the **Encryption** tab.
- Step 3 To view certificates, click **View Certificates** and scroll through the list.
- Step 4 To import a certificate, click **Import...** and follow the wizard.

Install a certificate in Google Chrome 24

- Step 1 Click the wrench icon in the upper right corner of the page.
- Step 2 Click **Settings > Show advanced settings**.
- Step 3 In the **HTTPS/SSL** section, click **Manage certificates...**
- Step 4 Click **Import** and follow the wizard.

Updating a certificate

As a general rule, third-party certificates are not changed. Some CAs will not allow any changes once the certificate is signed. If a change needs to be made, you should import the new certificate and configure it for use before you delete the old certificate.

Follow these procedures:

- “Delete a certificate” on page 3-16


Delete a certificate



Warning

Do not delete a certificate until its replacement is in place and configured. If you delete a certificate that is configured for use by the platform, Fox or Web Services could fail to restart. If you have the services configured for Https only, platformssl only, or Foxs only, a missing certificate could prohibit connectivity using SSL connections.

Workbench gives no warning if you delete a certificate that is currently being used by Workbench or the platform/station.

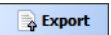
- Step 1 Open the platform.
- Step 2 Click **Tools > Certificate Management**.
- Step 3 Select the certificate in the **Key Store** and click  .

Back up the stores

To back up the Key and Trust Stores, select the certificates and export them to a secure location on your computer's hard disk or on a flash/thumb drive.

- “To back up the stores” on page 3-16

To back up the stores

- Step 1 Access both sets of stores:
 - To access the Workbench stores, click **Tools > Certificate Management**.
 - To access the platform stores through the Nav tree use either **Platform > Certificate Management**, or **Station > Config > Services > PlatformServices > CertManagerServices**.
- Step 2 Select each certificate and click  .
You export each certificate one at a time.
- Step 3 When exporting from a **Key Store**, export the private key along with the certificate, creating an encryption password for each private key.
- Step 4 Store the off-line storage medium (for example, a thumb drive) in a safe place.

Managing allowed hosts

If you used the self-signed certificates to provide initial encryption, more than one exception may be allowed in your **Allowed Hosts** list. Once you have set up signed certificates for all hosts and clients, delete the exceptions from the **Allowed Hosts** lists both for Workbench and for the platform/station.

Test station health

When you finish configuring a server, stop and restart a secure station and check station health. Existing connections using SSL are not revalidated against new certificates until they are reset. Existing connections using Http and Fox are not automatically changed to Https and Foxs even when **httpsOnly** and **foxsOnly** are enabled, until the connection is reset.

CHAPTER 4

About SSL—Alice, Bob, Cathy and Bart

This topic introduces basic security concepts:

- “What makes any system secure?” on page 4-1
- “Friends and enemies” on page 4-1
- “Identity verification” on page 4-3
- “Cryptography” on page 4-8

What makes any system secure?

A secure system requires:

- Physical security: Your JACE network should be located in a guarded location with appropriate locks, security systems, and physical access control.
- Protection against hacking: Your computers need firewalls, passwords and other software access controls.
- Verification of authenticity: *Certificates* verify that the contacted server is the expected server. This verification of authenticity thwarts imposters who attempt to steal trust for the purpose of disrupting network operations.
- Data transmission security: Data encryption defeats hackers who would use listening devices to capture transmitting data.

The third and fourth bullets above are the security requirements configured by the SSL Toolset that is part of NiagaraAX 3.7 and later.

Friends and enemies

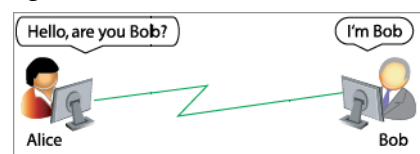
Alice, Bob and Cathy are remote friends. Bart is an imposter. This topic explains how internet security systems work.

- “When trust is secure” on page 4-1
- “When things go wrong” on page 4-2
- “Verifying authenticity” on page 4-2

When trust is secure

Alice connects to Bob remotely using the internet. For example, Bob may run a commercial website from which Alice wants to purchase goods or services.

Figure 4-1 Communication in a trusted world

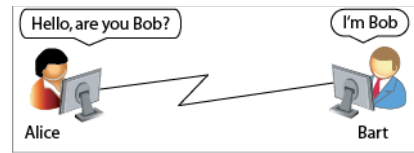


In [Figure 4-1](#), Alice and Bob are both authentic and legitimate. Communication between them is safe.

When things go wrong

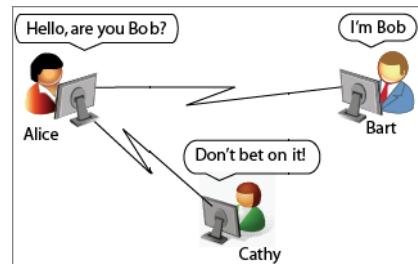
In [Figure 4-2](#), Bart is pretending to be Bob so he can steal Alice’s identity and confidence. This is referred to as a man-in-the-middle attack if the “attacker” (Bart) makes a connection with the victims (Alice and Bob) and relays messages between them while “eavesdropping” on their communication.

Figure 4-2 When things go wrong



How can Alice be confident that she is having a private “conversation” with Bob and not an imposter?

Figure 4-3 How the bad get caught

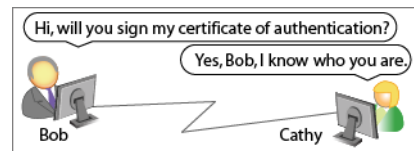


The answer is that she and Bob have a mutual friend, Cathy, who authenticates Bob’s credentials and warns Alice when the person to whom she is connected is not Bob.

Verifying authenticity

In reality, there is no person (or computer) on the internet whose job it is to monitor each client/server transaction. So, who or what is Cathy and how did Cathy know to warn Alice?

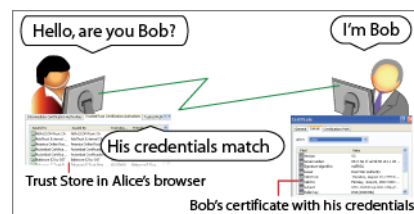
Figure 4-4 Cathy, a third-party Certificate Authority signs Bob's credentials



Cathy is a file known as a *certificate of authentication* owned and distributed by a Certificate Authority (CA). Bob also is a certificate of authentication. Ahead of time, Bob’s company sent its certificate to Cathy’s company, which verified Bob’s company identity and signed his certificate.

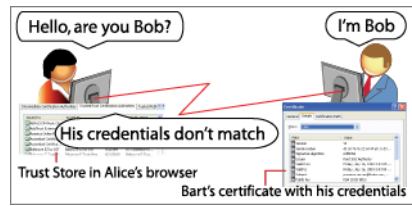
When Alice installed her browser, Cathy’s certificate was installed in Alice’s browser’s *Trust Store*. As the name implies, a Trust Store contains certificates from trusted entities.

Figure 4-5 Secure communication: the signatures match



As soon as Alice contacts Bob, he sends her his certificate. Alice’s browser checks the signature on Bob’s certificate against the signature on Cathy’s certificate in its Trust Store. The signatures match and Alice’s browser authorizes the beginning of a trusted connection between Alice and Bob.

Figure 4-6 Rejected communication: the signatures do not match



Alice's browser immediately rejects Bart's certificate because it was not signed by Cathy. Its signature does not match the signature on Cathy's certificate in Alice's browser's Trust Store.

Identity verification

Public Key Infrastructure (PKI) is the name of the technology that employs two security processes to ensure secure communication:

- Identity verification is described in more detail in these topics:
 - [“More about certificates”](#) on page 4-3
 - [“Keys”](#) on page 4-4
 - [“Signing a certificate with a private key”](#) on page 4-4
 - [“Creating a chain of trust”](#) on page 4-6
 - [“More about intermediate certificates”](#) on page 4-7
- [“Cryptography”](#) on page 4-8, which is the general term for encrypting (scrambling and unscrambling) data to ensure they are not intercepted during transmission.

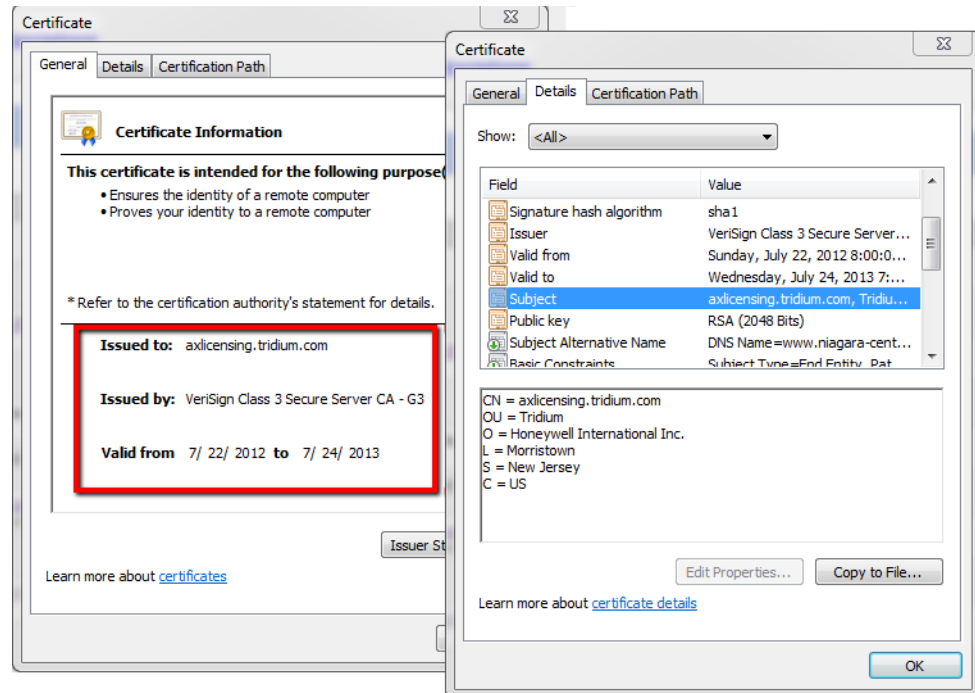
More about certificates

A certificate is an electronic document that uses a digital signature to bind a *public key* with a person or organization. Identity verification uses multiple certificates in a *chain of trust*. The example of Alice, Bob, Cathy and Bart involves at least three certificates:

- During the handshake with Alice, Bob presents his *server certificate*.
- The Trust Store in Alice's browser contains a copy of the *root certificate* that Cathy used to sign her own and Bob's server certificates. The successful matching of the signature on Bob's server certificate with the signature on Cathy's root certificate allows communication to begin.
- Bart also has a signed server certificate, but it was not signed by Cathy, therefore, his attempt to impersonate Bob is not trusted.

Each certificate contains metadata that identifies the certificate owner and the purpose of the certificate. [Figure 4-7](#) shows a certificate as it appears in Windows 7.

Figure 4-7 Certificate with metadata



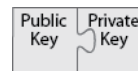
The **General** tab identifies to whom the certificate was issued (axlicensing.tridium.com), who the trusted Certificate Authority (CA) was that issued the certificate (VeriSign), and for how long the certificate is valid (until 7/24/2013). It is typical for certificates to be valid for a year or two. It is unusual for a certificate to be valid indefinitely.

The **Details** tab provides more information, including the Subject, which is also known as the Common Name (CN). In addition to signatures matching, Server and client CNs must match for secure communication to begin.

Keys

A pair of asymmetric *keys* (one public and the other private) makes SSL authenticity verification and encryption possible. The term “asymmetric” means that each key is unique but they match each other. 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 a JACE generates this pair of asymmetric keys.

Figure 4-8 Asymmetric keys



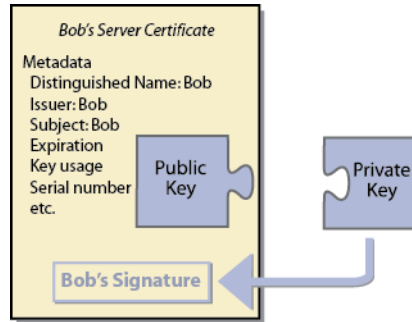
- A *public key* is a string of bytes wrapped by a *certificate*. This key resides in the server’s Trust Store and is used to identify the authenticity of the connecting client certificate.
- A *private key* is also a string of bytes that resides on the authentic server. The root certificate’s private key must be physically protected for a chain of trust 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).

Signing a certificate with a private key

Cathy’s company, a *Certificate Authority (CA)*, verified Bob’s identity and signed his *server certificate* with Cathy’s private key. Here’s how it happened:

1. Bob created a pair of asymmetric keys and a certificate that contained his credentials (his name, address, etc.).

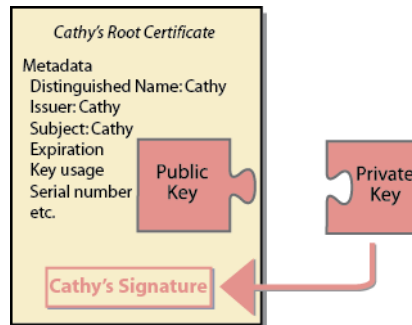
Figure 4-9 Bob's keys and self-signed server certificate



Bob's server certificate was not yet signed by a CA. At this stage it is *self-signed* using his own private key. Notice that the **Issuer** and **Subject** are the same.

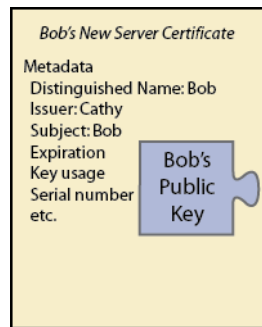
2. Bob sent this certificate to Cathy with a request that she verify his identity. (He probably also sent money with his request.) He did not send Cathy his private key.
3. As a CA, Cathy owns a pair of keys and a trusted root certificate.

Figure 4-10 Cathy's keys and root certificate



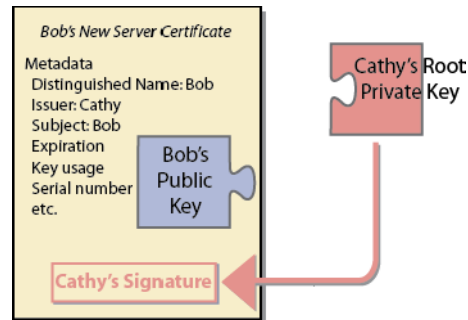
Cathy's root certificate is also a self-signed certificate. It serves as the top of the chain of trust. Cathy stores it on a computer that is not on the internet that is kept in a vault.

Figure 4-11 New server certificate for Bob



4. After thoroughly checking Bob's credentials, Cathy extracted Bob's public key and metadata from his self-signed certificate and created a new certificate with her name as the **Issuer**. Notice how the **Issuer** and **Subject** are different from the self-signed certificate that Bob sent Cathy.
5. Cathy then used the *private key* of her root certificate to sign this new certificate.

Figure 4-12 Signed server certificate for Bob



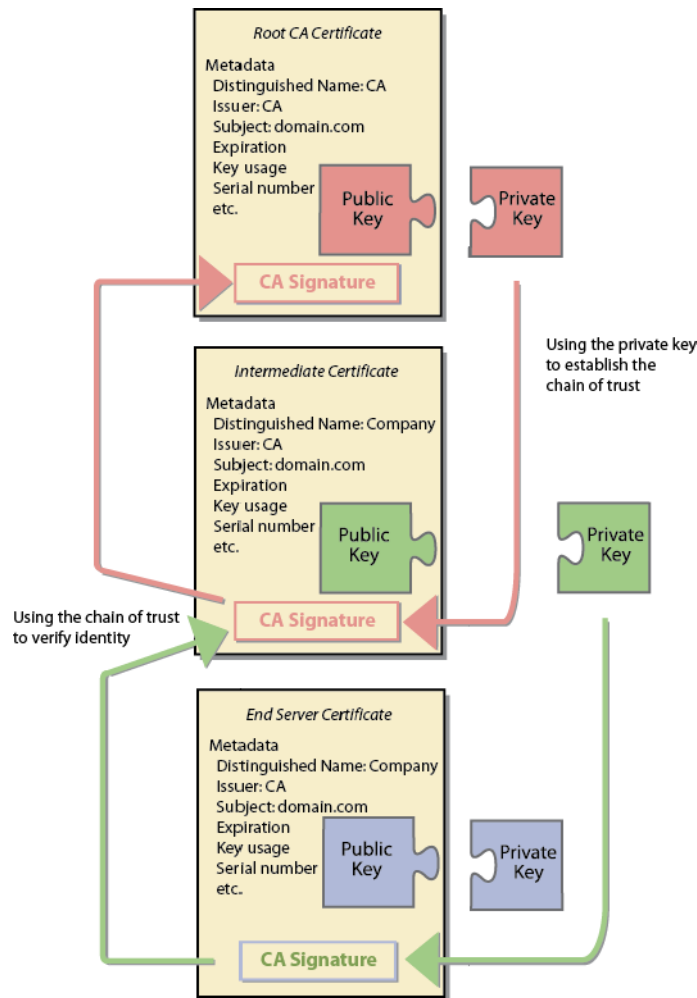
6. Cathy compressed both the new server certificate and a copy of her root certificate containing only her public key (Figure 4-10) with password protection, put both on a website and emailed the links to Bob. The public key part doesn't have to be protected and can be emailed.
7. Then she phoned Bob and gave him the password for the two compressed, password-protected files.
8. Bob expanded the files and imported his signed server certificate into the *Key Store* on his JACE. This action replaced his self-signed certificate. The imported certificate must match the certificate that created the CSR in the first place.
9. Finally, Cathy's root certificate needs to be installed in Alice's Trust Store. There are several ways for Alice to install Cathy's root certificate in her Trust Store:
 - If Cathy is a well-known CA, her root certificate may have been installed when Alice installed her browser.
 - Since Alice is a customer of Bob's, he may have created an installation program, which Alice can download and install in her browser.
 - If Alice is a JACE controller, Bob may install Cathy's root certificate in the office, when he sets up each JACE before taking them out to the field. Or he may put Cathy's root certificate on a thumb drive and take it to the JACE for installation.

The root certificate with its public key is a public certificate that can be emailed.

Creating a chain of trust

A *certificate chain* of trust, also known as the *certification path* is a structure with a root certificate at the top-most level. A root certificate is made trustworthy by securing its physical distribution.

Figure 4-13 A chain of trust with a Root Certificate Authority (CA) certificate at the top.



The arrows on the right show how the chain of trust was established using the private keys to sign the certificates at the next level down in the chain. All certificates immediately below the root certificate inherit the trustworthiness of the root certificate. Certificates further down the chain depend on the trustworthiness of the intermediate certificate(s).

The server certificate's private key is not part of establishing the chain of trust. It is used to encrypt and decrypt data after identity is established.

The arrows on the left side of the drawing show how identity is verified during the handshake. Working up the chain, the server sends the client its server and intermediate certificates. The client:

1. Verifies that the server certificate was signed by (matches) the intermediate certificate and that the **Distinguished Names** match.
2. Verifies that the intermediate certificate was signed by the root certificate, which it has in its Trust Store.

Assuming all signatures match, communication begins.

Note: Certificates have an expiration date. Every couple of years they must be renewed.

More about intermediate certificates

Any number of intermediate certificates may be set up in the chain of trust. A certain amount of overhead is carried to verify each certificate during the handshake. The benefit of multiple intermediate certificates is that they reduce the impact of a security compromise.

Cryptography

Once Bob’s identity is confirmed during the handshake with Alice, they no longer worry about confirming identities. Bob’s private key takes over the encryption task. Encryption prevents eaves dropping on their “conversation.”

Public Key Infrastructure (PKI) encrypts in two steps:

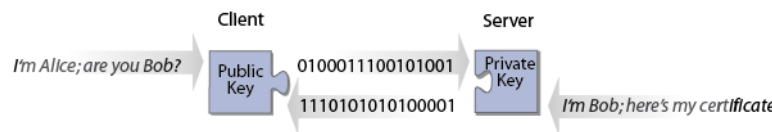
1. At the start of communication, the handshake receives extra protection. See “[Encrypting the handshake](#)” on page 4-8.
2. Once communication is established, encrypted data transmission begins. See “[Data transmission after the handshake](#)” on page 4-8.

Key size is directly related to security. See “[More about key size](#)” on page 4-8.

Encrypting the handshake

Encryption using both keys protects the exchange of the identity-verifying certificates.

Figure 4-14 The handshake uses asymmetric keys to encrypt the exchange of certificates



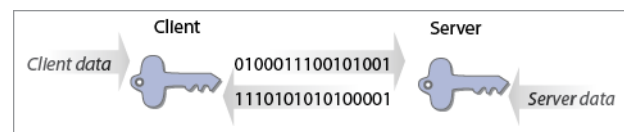
The private key on the server side encrypts the opening handshake, and the client’s matching public key decrypts it. This action (called asymmetric cryptography) protects the exchange of certificates used to establish identity.

- The advantage of using asymmetric keys, is that they can be larger (more secure) than symmetric keys (the same key used at both ends).
- The disadvantage of using asymmetric keys is that their size and complexity can make encryption slow. This is why they are used only to establish the connection.

Data transmission after the handshake

Once server identity has been established using asymmetric cryptography, the transmission continues with a negotiated symmetric key that both encrypts the data (symmetric cryptography).

Figure 4-15 A single key speeds symmetric encryption/decryption



- The advantage of symmetric cryptography is that it is simpler and faster than asymmetric cryptography.
- The disadvantage of using a single key for symmetric encryption is that there is no easy way to share the key (it should not be sent via email).

More about key size

The size of the key is directly related to the quality of security it offers. Larger keys are more secure. Your options for the key size property are:

- 1024, the size of the default key generated at the JACE, Workbench and Supervisor at start-up.
- 2048, the default if you generate a key manually. At the time of issuing this guide, this size is required by most Certificate Authorities.
- 3072
- 4096

To generate a 1024-bit key on a JACE takes about 15 seconds. To generate a 4096-bit key on a JACE can take a while. The reason is that key generation is math-intensive (it randomly generates prime numbers that meet certain criteria). Ideally, this type of security should be installed in person. If you are sitting next to each JACE with your laptop directly connected to the JACE, you can safely generate a complex key on your laptop and import it into the JACE.

CHAPTER 5

About NiagaraAX SSL

This topic describes how SSL security works in a NiagaraAX network.

- “NiagaraAX’s client/server architecture” on page 5-1
- “About the certificate creation and signing process” on page 5-2
- “About NiagaraAX SSL certificates” on page 5-5
- “About the SSL Toolset” on page 5-7

NiagaraAX’s client/server architecture

The typical internet client/server relationship with a third-party verifying the server becomes a little more complicated when considering the relationships between the various programs (processes) within the NiagaraAX Framework.

NiagaraAX’s SSL implementation is platform based. All SSL Toolset functions can be configured using only a platform connection and without any station running.

Three NiagaraAX programs (processes) require communication protection:

- Fox Service, a proprietary program that is used for all network communication between stations as well as between the Workbench and stations.
- Web Service for the station. This service is used to download the Workbench applet (wbapplet) and the modules that are required for running the station. It is also used to display hx pages, mobile pages, etc.
- Platform for Niagara, which provides Http connectivity to a platform.

For more information about how the NiagaraAX programs provide NiagaraAX functionality, see *Types of NiagaraAX Programs* in the *Networking and IT Guide*.

Figure 5-1 NiagaraAX client/server relationships

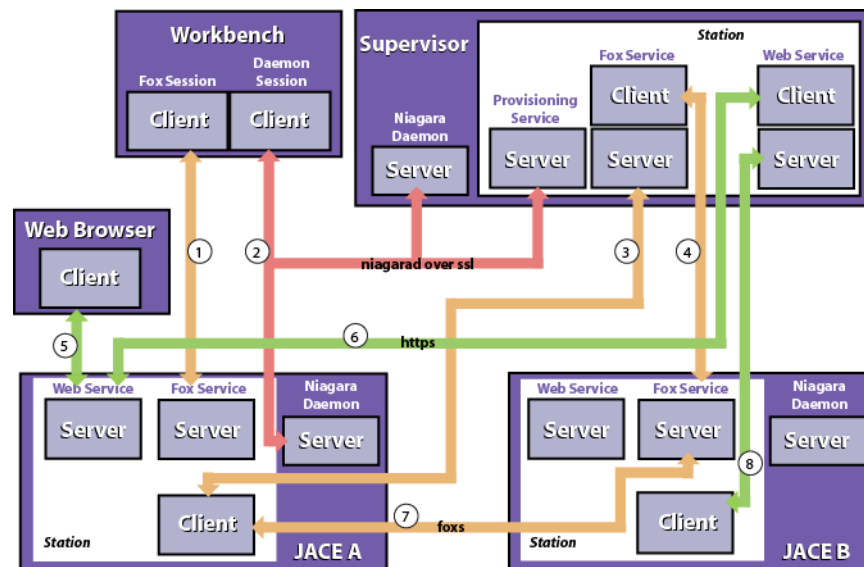


Figure 5-1 illustrates the possible client/server relationships within the NiagaraAX Framework:

1. Workbench station (client) connected to JACE station (server) using secure Fox Service.

2. Workbench (client) logged in to a JACE platform (server) using Niagarad (platform connection).
3. JACE station (client) connected to a Supervisor station (server) using secure Fox Service.
4. Supervisor station (client) connected to a JACE station (server) using secure Fox Service.
5. Web browser (client) connected to the station (server) using Https.
6. Station (client) connected to the remote Supervisor (server) (Http).
7. JACE station (client) logged in to another JACE station (server) using secure Fox Service.
8. Supervisor station (client) connected to a station (server) using Https.

Note: Each JACE in figure [Figure 5-1](#) has all the same connections to Workbench and the Supervisor as the other JACE. For simplicity the illustration uses each to demonstrate the nearest connections (nearest in the drawing).

About the certificate creation and signing process

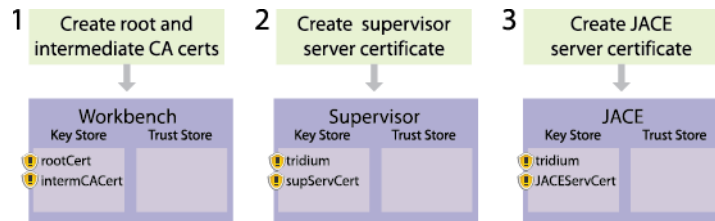
The illustrations in this section are intended to help you visualize what needs to be done. The actual steps you will take may vary. For example, to save time it makes sense to do everything in Workbench at one time rather than go back and forth between the Workbench and platform/station tools.

- [“Certificate creation”](#) on page 5-2
- [“Workbench certificate signing”](#) on page 5-3
- [“JACE certificate signing”](#) on page 5-3
- [“Supervisor certificate signing”](#) on page 5-4
- [“Setting up the client Trust Stores”](#) on page 5-4

Certificate creation

To set up a chain of trust you begin by creating a root CA certificate and a server certificate for each JACE and Supervisor. You may also require several intermediate CA certificates. For example, if your company has multiple locations, you may want an intermediate certificate for each location.

Figure 5-2 Summary of CA and server certificates to create



1. The first step is to create the root and intermediate certificates. If you use intermediate certificates, you probably have more than one. The illustrations show only one `intermCACert` for the sake of simplicity.

The Workbench stores are separate from each platform/station stores. You access the Workbench SSL tools by using a menu option. Click **Tools > Certificate Management**.

For the procedure, see [“Create the root and intermediate certificates”](#) on page 3-3.

2. The next step is to create one or more supervisor certificates. You access the platform/JACE stores by double-clicking `CertManagerService` under `PlatformServices` in the station Nav tree. For the procedure, see [“Create new JACE and Supervisor server certificates”](#) on page 3-9.

3. Finally, you create a certificate for each JACE. The `tridium` certificates you see in the Supervisor and JACE Key Stores are the default self-signed certificates that are created at initial station startup. Although the certificates are all named “tridium,” each is unique to the platform on which it was created. In their unsigned, default state these certificates do not provide server authentication, but they do provide encryption.

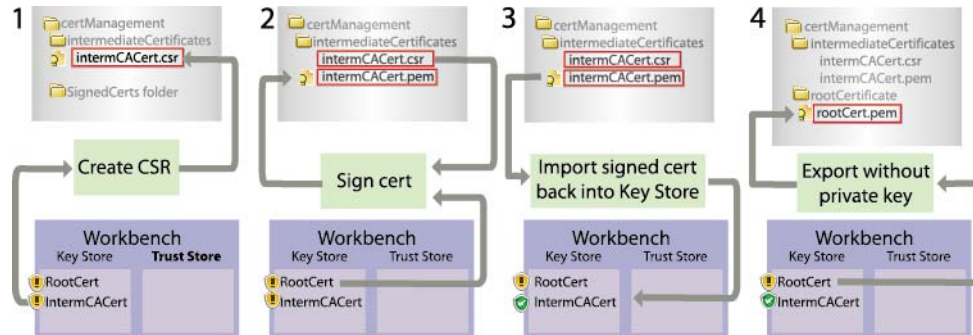
This example demonstrates creating server certificates for a Supervisor and JACE using 2048-bit keys, which are more secure than the 1024-bit keys of the default certificates.


To save space, the remainder of the illustrations do not show the `tridium` server certificate. After the new certificates are signed and imported, you will select them for each station. At that time you can delete the `tridium` certificates.

Workbench certificate signing

Using the SSL Toolset the intermediate certificates are signed by the private key of the root certificate.


Figure 5-3 Setting up the Workbench certificates



1. The first step is to create a Certificate Signing Request (CSR) for the intermediate certificate. For the procedure, see “[Create a CSR for the intermediate certificate](#)” on page 3-5.
2. In the next step, you use the certificate signing tool and the private key of the root certificate to sign the intermediate certificate. Once signed, the certificate should be stored in a separate folder. For the procedure, see “[Sign the intermediate certificate using the root certificate’s private key](#)” on page 3-6.
3. This step marries the signed certificate with its private key, which never left the Key Store. The green shield  indicates that the intermediate CA certificate has been signed.
4. In this step you export the root certificate without its private key. You will then import the root certificate into the Trust Store of each JACE.

The intermediate certificate (IntermCACert) does not need to be exported. It is used to sign server certificates and does not need to be imported into the any Trust Store. The only reason to export an intermediate certificate would be to back it up.

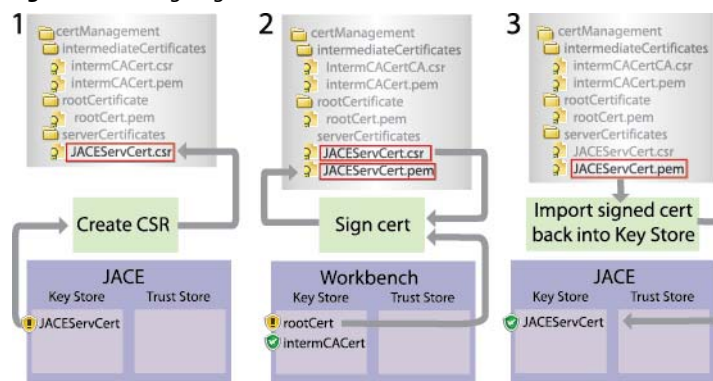
For the procedure, see “[Export the root and intermediate certificates](#)” on page 3-8.


Note: *In the Workbench Key Store, the root certificate always shows the caution shield . As the highest authority in the chain of trust, this certificate must be self-signed. For this reason, the root certificate must be physically protected for the security system to provide any protection to the network.*

JACE certificate signing

Each JACE has its own separate and unique platform/station stores.

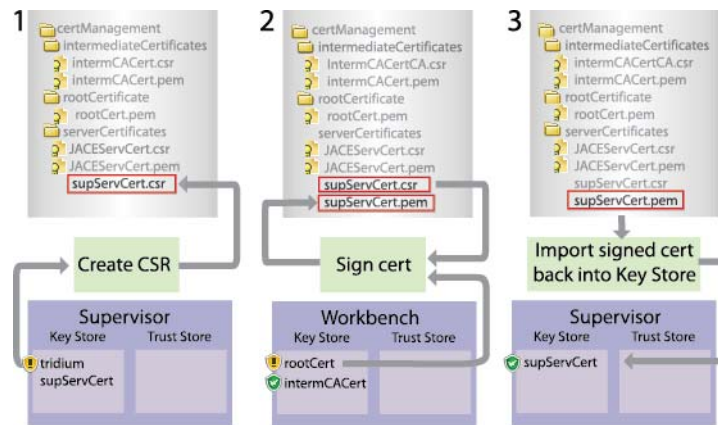
Figure 5-4 Signing the JACE server certificate



1. Creating the CSR for the JACE server certificate prepares it for signing without its private key. For the procedure, see “[Create a CSR for each server certificate](#)” on page 3-10.
2. Signing the JACE server certificate is done using the private key of the intermediate certificate. For the procedure, see “[Sign the server certificates using the intermediate certificates](#)” on page 3-11.
3. This step marries the signed JACE server certificate with its private key, which never left the Key Store. The green shield  indicates that the JACE server certificate has been signed.

Supervisor certificate signing

Figure 5-5 Signing the Supervisor certificate



The Supervisor procedure is the same as that for the JACE.

1. Creating the Supervisor certificate CSR prepares it for signing without its private key. For the procedure, see [“Create a CSR for each server certificate”](#) on page 3-10.
2. Signing the Supervisor server certificate is done using the private key of the root certificate.
3. Importing the signed Supervisor server certificate back into the Key Store changes the caution shield to the green shield.

Setting up the client Trust Stores

After creating and signing the certificates, the final step is to import the root certificate into the client Trust Store for each server (JACE) and browser.

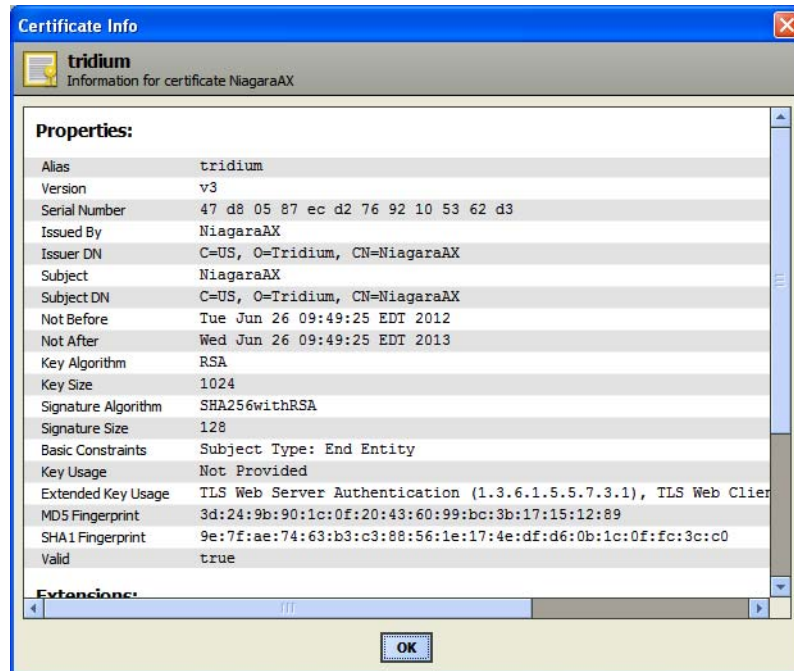
There is no need to import the server certificates nor the intermediate CA certificates to any Trust Stores. Each server certificate carries within it any intermediate certificate information. If you are acting as a local CA, you will need to import the root CA certificate into the Trust Stores. However, if the server's certificate has been signed by a well-known trusted CA whose root certificate is already in the Trust Stores, then there are no additional steps required.

For the procedure, see [“Set up the platform and station Trust Stores”](#) on page 3-14.

About NiagaraAX SSL certificates

NiagaraAX certificates are presented in an easy-to-read format.

Figure 5-6 An example of the default Tridium certificate



A certificate contains a public key, a signature and metadata including: subject, issuer, valid date ranges, the algorithm and key size, and the purpose of the certificate (key usage).

- “Types of NiagaraAX certificates” on page 5-5
- “About self-signed certificates” on page 5-5

Types of NiagaraAX certificates

Certificates serve a variety of purposes depending on how the certificates’s **Key Usage** field is configured. To reduce complexity, NiagaraAX 3.7 and later arbitrarily manages these types of certificates:

- A *server certificate* resides with its matching private key in the **Key Store** on the server (JACE or Supervisor). No password is required to *use* this certificate.
- A *CA certificate* is used to sign other certificates. The private key requires the creation of a password on export and the provision of a password on import.
A root CA certificate exported with only its public key serves as a *client certificate* in the Workbench and station **Trust Stores** of each client.

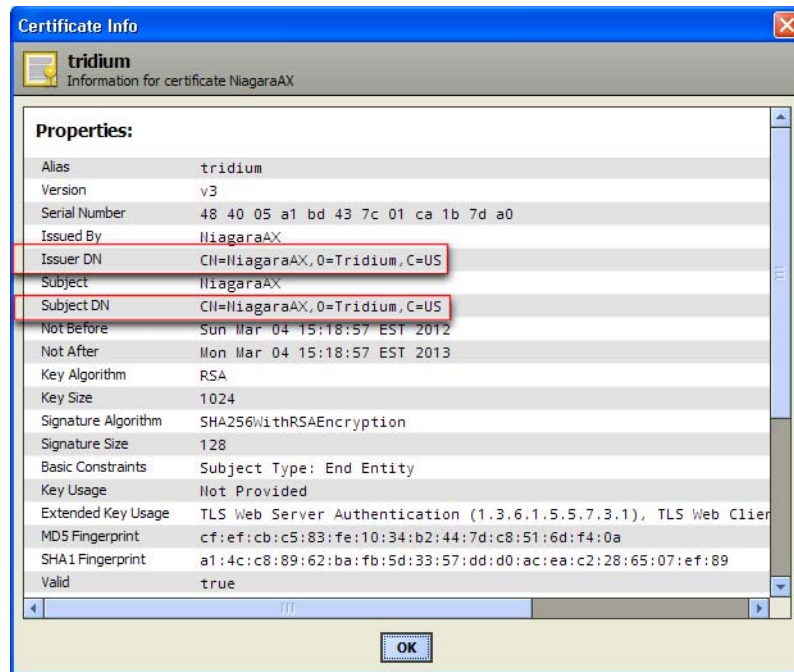
About self-signed certificates

A self-signed certificate is one that is signed by default using its own private key rather than by the private key that is owned by a CA. This type of certificate cannot be validated by a client and is not recommended for robust security when used as a JACE server certificate. There is no procedure for self-signing a certificate. Each is created self-signed.

Two self-signed certificates are used in a JACE network:

- A *default self-signed certificate for each JACE*: When a JACE starts up for the first time, it creates this unique, self-signed certificate, the primary purpose of which is to provide immediate encryption. This certificate uses a 1024-bit pair of keys.

Figure 5-7 The default self-signed certificate



Notice the **Issuer DN** and **Subject DN** properties. The **Issuer DN** (Distinguished Name) is logically the same as the **Subject DN**. This indicates that this certificate is signed with its own private key. Because it is self-signed, the client **Trust Store** does not contain a certificate with a public key that matches this certificate’s signature. The **Issuer DN** and **Subject DN** are different for a certificate signed by a Certificate Authority (CA).

The first time a client connects to the JACE, the software displays a message indicating that the default certificate is not trusted. Whether you approve or disapprove the certificate, the software lists it in the **Allowed Hosts** list. If approved, the host is identified as trusted and you will not have to approve the connection each time it is made.

Unless you replace it with a certificate that you create, this certificate can only be used to encrypt data. You should not copy this certificate (or a certificate you create for this server) from one platform to another.

Self-signed certificates are inherently less secure because they cannot authenticate the server. Validating the server’s identity helps protect against man-in-the-middle attacks. 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 not publicly accessible from the internet.



Caution

If you intend to use self-signed certificates, before you access the JACE from Workbench for the first time, make sure that your PC and the JACE are not on any corporate network or the internet. Once disconnected, connect the PC directly to the JACE, access the JACE from Workbench, and approve its self-signed certificate. Only then should you reconnect the JACE to the corporate network or internet.

The approved host exception in the **Allowed Hosts** list is only valid when connecting to the server using the IP address or domain name that was used when the exception was originally created. If you use a different IP address or domain name to connect to the server, you will need to approve an updated exception. The same is true if a new self-signed certificate is generated on the host.

In the identity verification warning dialog, the changed items are indicated using different colored text.

- **Root certificate:** This type of self-signed certificate is implicitly trusted because there is no higher authority than the entity that created this certificate. For this reason, Certificate Authorities, whose business it is to endorse other people’s certificates, closely guard their root certificate(s).

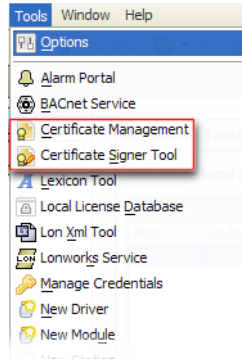
Workbench keys and certificates

Workbench functions purely as a client with its own stores and **Allowed Hosts** list. This topic contains these procedures:

- “View the Workbench stores” on page 5-7

View the Workbench stores

- Step 1 Launch Workbench.
- Step 2 Click the **Tools** menu.



Two menu items manage the Workbench keys and certificates.

- **Certificate Management** is used to create certificates, create Certificate Signing Requests (CSRs), and to import and export keys and certificates to the Workbench stores.
 - **Certificate Signer Tool** is used to sign certificates.
- Step 3 To view Workbench’s client stores, click **Certificate Management**.

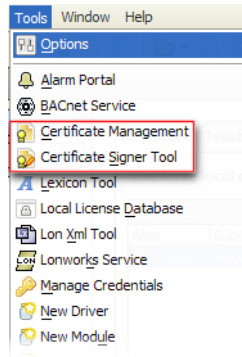
The **Certificate Management** view displays the three lists: **Key Store**, **Trust Store** and **Allowed Hosts**.

About the SSL Toolset

Two sets of certificates are maintained by the SSL Toolset.

- *Workbench certificates*: Workbench, which always functions as a client, has its own set of certificates and keys that are separate from the platform/station certificates and keys.

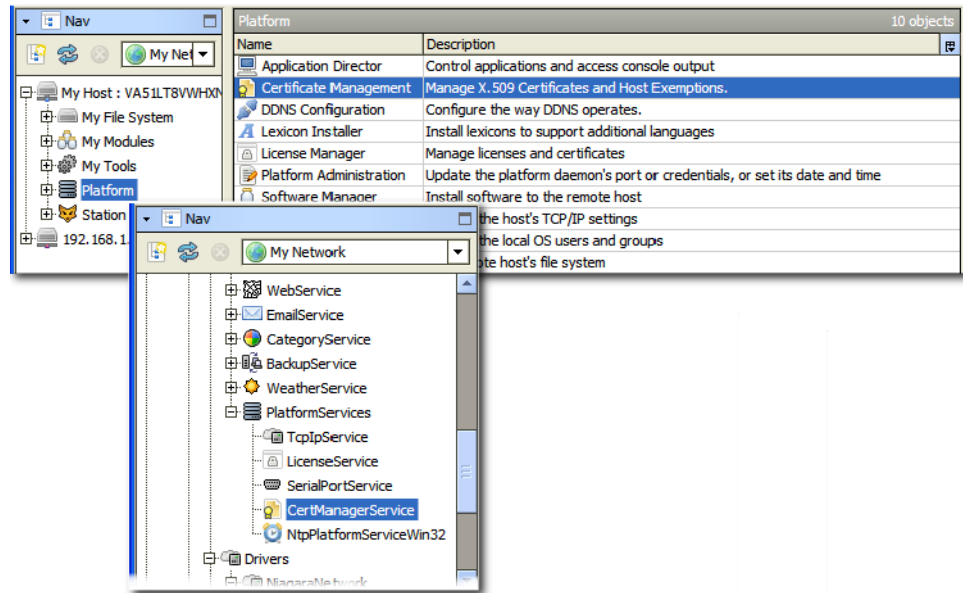
Figure 5-8 Workbench SSL tools



You manage Workbench certificates using Workbench’s **Tools > Certificate Management**, and **Tools > Certificate Signer Tool** menu options.

- *Server certificates*: You manage the set of certificates and keys for each supervisor and JACE through **PlatformServices**.

Figure 5-9 Two ways to access the platform/station SSL tools



To access the supervisor and station server certificates:

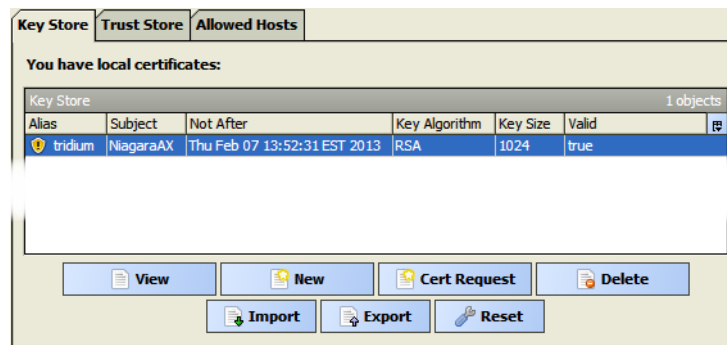
- Make a platform connection to the Niagara host, double-click **Platform** in the Nav tree, and double-click **Certificate Management** in the **Platform** pane, or
- Make a station (Foxs) connection to the Niagara host, expand the **Services > PlatformServices** node in the Nav tree, and double-click **CertManagerService**. This is the same view as that accessed from the platform connection.

Whether you are accessing the Workbench or platform certificates and keys, the interface is essentially the same. Certificates and keys appear as rows in one of three tables:

- The **Key Store** list (table), see “About the Key Stores” on page 5-8.
- The **Trust Store** list (table), see “About the Trust Stores” on page 5-9.
- The **Allowed Hosts** list (table), see “About Allowed Hosts” on page 5-10.

About the Key Stores

Figure 5-10 Example of a Key Store list



The **Key Store** in Workbench, and in the JACE and Supervisor stations contains one or more server certificates, each with its pair of private and public cryptographic keys. The default server certificate (**tridium**) has the same name in each key store, however, its keys are unique for each instance.

If there are no certificates in the Key Store when starting the Workbench application or a Niagara station (JACE or Supervisor), a default self-signed certificate is created. Clicking the **New** or **Import** buttons also adds certificates to the Key Store.

Double-clicking the certificate row in the table allows you to view certificate details.

Note: Once created, you cannot edit a certificate. To correct an error you must delete the certificate and start again. This is important for security.

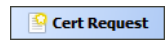
Key Store buttons



displays certificate details for the selected certificate.



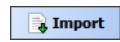
opens the **Generate Self Signed Certificate** dialog, which is used to create CA and server certificates.



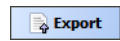
opens a **Certificate Request** dialog, which is used to create a Certificate Signing Request (CSR).



removes the certificate from the **Key Store**. If you delete a certificate that is currently configured and in use, you could lose a connection.



adds the certificate (.pem file) to the **Key Store** if the certificate is not already in the Key Store. Otherwise, importing updates the existing certificate. For example, importing is used to update an existing certificate with a signature.



saves a copy of the selected certificate to the hard disk. The file extension is .pem.



deletes all keys in the **Key Store** and creates a new default key pair and certificate. It does not matter which certificate is selected when you click **Reset**. Reset does not reboot the JACE.



Caution Do not reset without considering the consequences. The Reset button facilitates creating a new key pair (private and public keys) for the entity, but may disable connections if valid certificates are already in use. Export all certificates before you reset.

About the Trust Stores

Figure 5-11 Example of a Trust Store list

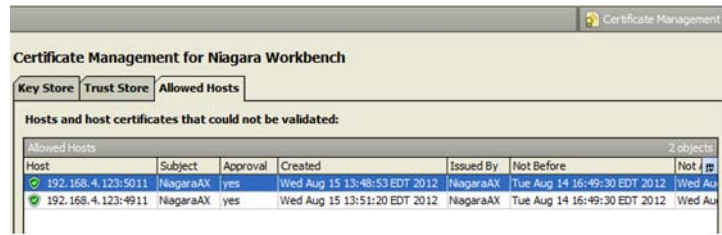
Trust Store	Subject	Not After
digicertassuredidrootca	DigiCert Assured ID Root CA	Sun Nov 0
trustcenterclass2caii	TC TrustCenter Class 2 CA II	Wed Dec 3
thawtepremiumserverca	Thawte Premium Server CA	Fri Jan 01
swissignplatinumg2ca	SwissSign Platinum CA - G2	Sat Oct 25
swissignsilverg2ca	SwissSign Silver CA - G2	Sat Oct 25
thawteserverca	Thawte Server CA	Fri Jan 01
equifaxsecurebusinessca1	Equifax Secure eBusiness CA-1	Sun Jun 2
utnuserfirstclientauthemailca	UTN-USERFirst-Client Authentication and Email	Tue Jul 09
thawtepersonalfreemailca	Thawte Personal Freemail CA	Fri Jan 01
entrustevca	Entrust Root Certification Authority	Fri Nov 27
utnuserfirsthardwareca	UTN-USERFirst-Hardware	Tue Jul 09
certumca	Certum CA	Fri Jun 11
addtrustclass1ca	AddTrust Class 1 CA Root	Sat May 3
entrustrootcag2	Entrust Root Certification Authority - G2	Sat Dec 07
equifaxsecureca	Equifax Secure Certificate Authority	Wed Aug 3
quovadisrootca3	QuoVadis Root CA 3	Mon Nov 2
quovadisrootca2	QuoVadis Root CA 2	Mon Nov 2

The **Trust Store** contains signed and trusted CA certificates with their public keys. the Trust Store contains no private keys. Trust Store CA certificates are used by the client Workbench or station/platform to validate the digital signature of the server certificate to which the client is connecting. If the certificate names and signatures match, communication begins. If they do not match, an error message allows you to approve a security exception.

You add certificates to Workbench's **Trust Store** or a station's **Trust Store** by importing them.

About Allowed Hosts

Figure 5-12 Example of Allowed Hosts list



The **Allowed Hosts** list contains security exceptions. These are hosts that submitted server certificates (during the handshake), which could not be validated (the private key used to sign the server certificate does not match the public key of a root or intermediate certificate in the **Trust Store**). Regardless of the response to the message (Allow or Reject), the host that sent the unmatched server certificate is listed in the **Allowed Hosts** list.

Exceptions are also created when the certificate was issued for a different host.

If you are using the default self-signed certificate or you have not imported the CA's root certificate into the applicable Trust Stores before initiating the Foxs connection, the connection fails and creates a host exception that needs to be approved.

If this is a station to station connection, the connection fails essentially silently. There is no prompt to approve the host exception. If this is a Workbench to station connection you are prompted with a dialog to approve the host exception.

Workbench challenges server identity at startup 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.

Note: *Host identity includes the IP address and port number. Port numbers are different for secure platform and station connections, thus you can have two certificates for the same IP host. If the IP address of an approved host changes, and no matching certificate exists in the **Trust Store**, Workbench again challenges server identity at startup and enters a "new" host in the **Allowed Hosts** list.*

*You should only approve exceptions if you are sure of the identity of the host and know why the matching certificate was not found in the **Trust Store**.*

CHAPTER 6

Troubleshooting

- [“Fix error conditions”](#) on page 6-1
- [“Reset or replace a JACE securely”](#) on page 6-2

Fix error conditions

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

A: Most likely there is something wrong with the certificate. If a certificate fails, or for any reason SSL does not enable, rather than lock you out of the platform, the system enables a connection without security.

Reset the JACE.

Note: If you have to replace or reset a JACE, assuming you exported the keys, you can import them to the reset JACE.

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

A: A port may be blocked or ignored by your firewall or secure router. See [“About TCP ports”](#) on page 7-2 for the list of default port numbers. 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.

Q: I’m using a signed server certificate, but the message “Unable to verify host identity” still appears when connecting to the platform.

A: The server certificate does not have a matching public key in the **Trust Store**. Import the root certificate into the **Trust Store**.

Q: My JACE or Supervisor private key has been compromised, what should I do?

A: Get on site as quickly as possible. Take the entire network off the internet. Reset each JACE and configure security again creating and signing all new certificates.

Q: 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.

A: The problem may be caused by one or more of the following:

- The Workbench client may no longer contain the host’s certificate in the Trust Store (for whatever reason). Check the certificate and import as needed.
- The certificate may have expired or changed and a new certificate needs to be imported into the Trust Store. To be on the safe side, clear the stores and have the platform generate a new certificate and key pair. Then, either get the new certificate signed and import it into the Trust Store or approve it in the Allowed Hosts list.
- There may be a problem with the Fox port. Check the Fox Service on the Client NiagaraNetwork to ensure the correct Fox port: 4911 for Foxes; 1911 for Fox.

Q: When importing a client certificate into a client Trust Store I get the message, “The ‘Import’ command encountered an error.”

A: Check the certificate and re-create it if necessary. Click the **Details** button to view the Workbench console. You may be attempting to import a private key into the **Trust Store**. This cannot be done.

Q: I’m trying to get two stations to connect and it’s not working.

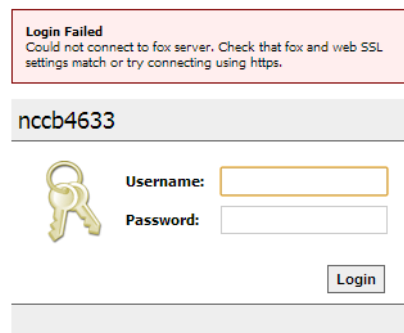
A: If this is the first time you are making this connection, check the **Allowed Hosts** list. The station serving as the client may not have a certificate in its **Trust Store** for the station that is serving as the server. In the **Allowed Hosts** list, select the certificate and click **Approve**. Make sure the certificate has the correct name and port number in the **Host** column.

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

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

Q: I configured SSL and supplied one or more certificates, but I cannot make a secure connection to the station.

A: When you begin the process you connect to the station using a standard Fox connection that is not secure. Once SSL is configured, you must right-click on the station and disconnect all Fox sessions, then reconnect to the station using Foxs.

Q: I get the following message when I attempt to log in to a secure station:**Figure 6-1** Login error

A: Either secure Web Service (Https) is enabled (set to **true**) and Foxs is not enabled, or Foxs is enabled and Https is not. For a secure connection, both **Https Enabled** and **Foxs Enabled** must be set to **true**. To check and change these properties, see [“Enable SSL for the Supervisor and JACE stations”](#) on page 2-7.

Reset or replace a JACE securely

To reset a JACE, you should be on site. Resetting a JACE remotely is not recommended because restoring the Key and Trust Stores should not be performed while the station is connected to the internet.

If you have to replace a JACE, you can reuse the Key and Trust Stores from the old JACE. If no station backup is available, you must generate a new certificate and sign it or get it signed again.

- [“To reset or replace a JACE with security”](#) on page 6-2

To reset or replace a JACE with security

-
- Step 1 Make sure that the JACE is not on the internet.
- Step 2 Reboot the JACE and restore the station.
- Step 3 Either restore from the station backup, or import the Key and Trust Stores from a previously exported file.

CHAPTER 7

Reference

This topic provides quick reference information for SSL Toolset tools and features.

- [“SSL Toolset terminology”](#) on page 7-1
- [“About TCP ports”](#) on page 7-2
- [“About the Certificate Management view”](#) on page 7-2
- [“About the Certificate Signing dialog”](#) on page 7-7
- [“SSL configuration properties”](#) on page 7-7

SSL Toolset terminology

Allowed host A host whose server certificate cannot be validated by a trusted certificate in the Trust Store.

Base certificate A certificate that will be used to create a Certificate Signing Request. A base certificate may be a default certificate created by Workbench or a JACE on system start-up, or a certificate you create using the SSL Toolset.

CA certificate A certificate whose private key is used to digitally sign other certificates. With only its public key, this certificate may be downloaded from the internet or sent via email. Do not download or send it with its private key unless it is heavily encrypted and in a ZIP file with a strong password.

Client A browser, Workbench, JACE, Supervisor or program (process) that seeks information from a server in a NiagaraAX network.

Client certificate The certificate with its public key (no private key) that resides on the client Trust Store. See also CA certificate, Intermediate certificate, and Root certificate.

Certificate Authority (CA) An entity (certification authority) that issues digital certificates to certify the ownership of a public key by the named subject of the certificate. This allows others to rely upon the signature presented by the subject and use its key pair (public and private) to encrypt data.

Certificate/certificate of authentication A general name for an electronic file that establishes a user's credentials when doing business or other transactions over the internet. A digital certificate contains the subject, expiration dates, copy of the certificate holder's public key (used to encrypt messages and digital signatures), and the purpose of the certificate (server, intermediate or root certificate).

The certificate may be signed by a Certificate Authority (CA), or it may be self-signed.

See also CA certificate, Self-signed certificate, Root certificate, and Server certificate.

Key Store A location for storing a certificate with both its public and private keys.

Encryption The process of using a pair of keys to scramble data at the sending end of a communication, and unscramble the data at the receiving end.

Handshake The initial exchange of certificates between a client and server that establishes a communication session.

Intermediate certificate A certificate between the root certificate and server certificate in a certificate chain of trust. An intermediate certificate is signed by the private key of the root certificate or another intermediate certificate. During identity verification, the signature of the server certificate is validated using the signature of the intermediate certificate, whose signature is, in turn, validated against the root certificate's signature.

The use of intermediate keys isolates servers. If one key is compromised the entire network is not threatened.

Private key A software entity based on prime numbers and used to encrypt data. For encryption to remain secure the private key must be physically protected.

Program (or process) One of these NiagaraAX entities that provides communication services within a NiagaraAX network: Niagarad, Fox Service, and Web Service.

Protocol A set of rules that facilitates information exchange within a computer system, between computers, and between a client and server.

Public key A software entity based on prime numbers and used to encrypt data. This key matches its private key and may be distributed freely.

Root certificate A self-signed certificate that is implicitly trusted and used to sign other certificates. See also CA certificate.

Secure Socket Layer (SSL) A commonly-used protocol for managing the security of message transmission over the internet. SSL uses encryption keys and includes a digital certificate.

See also Transport Layer Security (TLS)

Self-signed certificate A certificate that has not been signed by a CA. It is signed, but by its own private key. In NiagaraAX no separate step or procedure is required to self-sign a certificate. When first generated each certificate is self-signed by default.

Server A platform or program (process) that offers information to another platform, program browser or Workbench in a NiagaraAX network. See [“NiagaraAX’s client/server architecture”](#) on page 5-1 for a description of the various roles within a NiagaraAX network.

Server certificate A certificate that is used primarily by the JACE or Supervisor station/platform for encryption. In NiagaraAX, the private key of a server certificate is not used to sign other certificates. See also Certificate, CA Certificate.

Transport Layer Security (TLS) A commonly-used protocol for managing the security of message transmission over the internet. TLS uses encryption keys and includes a digital certificate.

See also Secure Socket Layer (SSL).

Trust Store A location for storing a trusted certificate with its public key. The Trust Store contains no private keys.

About TCP ports

Ports may be blocked or ignored by firewalls or secure routers. You must be aware of these blocked ports, and make appropriate exception rules where necessary. If a port is blocked by a firewall or router, communication will not succeed.

The default Niagara TCP port numbers are:

- fox: 1911
- foxs: 4911
- platform: 3011
- platformssl: 5011
- http: 80
- https: 443
- email: 25
- email ssl: 587, 465

About the Certificate Management view

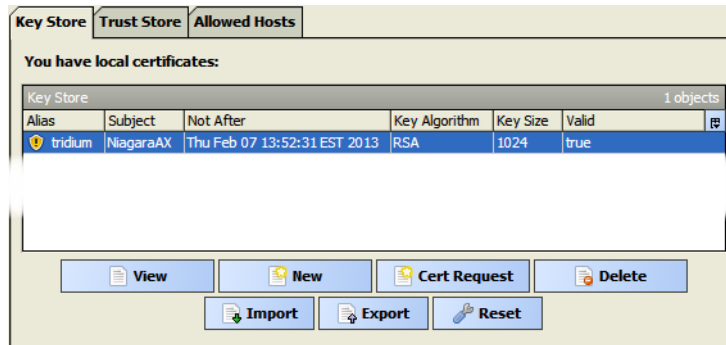
The Certificate Management view of the SSL Toolset allows you to create PKI (Public Key Infrastructure) digital certificates; to create Certificate Signing Requests (CSRs); and to import and export keys and certificates to and from the Workbench and JACE key stores.

- [“About the Key Store tab”](#) on page 7-3
- [“About the Private Key Password dialog”](#) on page 7-5
- [“About the Trust Store tab”](#) on page 7-5

- “About the Allowed Hosts tab” on page 7-6

About the Key Store tab

Figure 7-1 Key Store tab

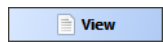


- “Key Store columns” on page 7-3
- “Key Store buttons” on page 7-3
- “About the Generate Self-Signed Certificate dialog” on page 7-4

Key Store columns

- **Alias** is a name used to distinguish certificates from one another in the **Key Store**. Use it to identify certificates by location or function.
- **Issued By** identifies the entity that created the certificate.
- **Subject** is the Distinguished Name, the name of the company that owns the certificate.
- **Not Before** displays the date before which the certificate is not valid.
- **Not After** displays the expiration date for the certificate.
- **Key Algorithm** refers to the mathematical formula used to calculate the certificate keys.
- **Key Size** shows 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. The bigger the key, the longer it takes to generate.
- **Signature Algorithm** refers to the mathematical formula used to sign the certificate.
- **Signature Size** shows the size of the signature.
- **Valid** shows certificate dates.
- **Self Signed** indicates that the certificate was signed with its own private key.

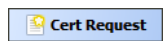
Key Store buttons



displays certificate details for the selected certificate.



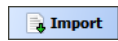
opens the **Generate Self Signed Certificate** dialog, which is used to create CA and server certificates.



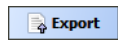
opens a **Certificate Request** dialog, which is used to create a Certificate Signing Request (CSR).



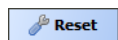
removes the certificate from the **Key Store**.



adds the certificate (.pem file) to the **Key Store**.



saves a copy of the selected certificate to the hard disk. The file extension is .pem.



deletes all certificates in the **Key Store** and creates a new default certificate. It does not matter which certificate is selected when you click **Reset**.

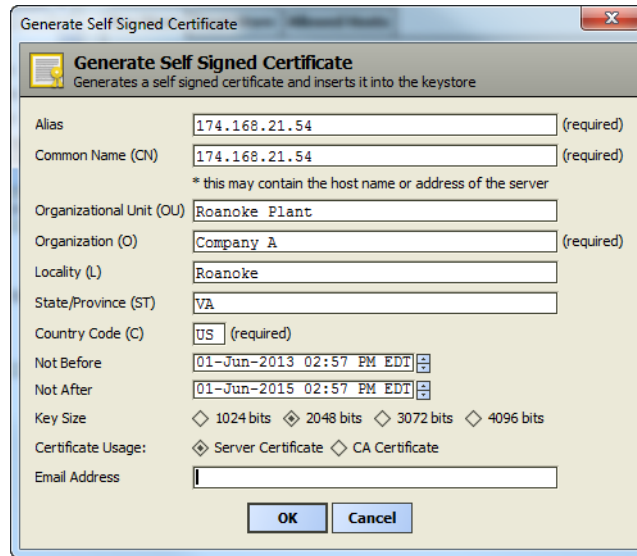


Caution The **Reset** button facilitates creating a new key pair (private and public keys) for the entity, but may have unintended consequences if you delete valid certificates. Export all certificates before you reset.

About the Generate Self-Signed Certificate dialog

This dialog appears when you click  at the bottom of the **Key Store** tab.

Figure 7-2 Distinguished Name dialog



You use this dialog to create your own certificates along with a key pair (public and private).

There is a limit of 64 characters for each of the following fields. Do not enter blank in any field. Spaces and periods are allowed. Enter full legal names.

Alias is a short name used to identify the certificate. This field is required. Use this field to indicate the type of certificate (root, intermediate, server) and where the certificate will be used.

Common Name (CN) is your Distinguished Name and can be the same as the Alias. Do not use the symbols “*” or “?” as part of this name. This name appears as the **Subject** in the **Key Store**. This field is required.

Organizational Unit (OU) is 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) is the legally registered name of your company or organization. Do not abbreviate this name. This field is required.

Locality (L) is 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) is the complete name of the state or province in which your organization is located. This field is optional.

Country Code (C) is a two-character ISO-format country code. If you do not know your country’s two-character code, check the internet. This field is required.

Not Before indicates the date on which the certificate becomes valid.

Not After indicates when the certificate expires.

Key Size establishes the size of the key in bits. The larger the key, the longer it takes to generate.

Certificate Usage: identifies the purpose of the certificate. In NiagaraAX, certificates are either server or CA certificates. Other open-source certificate management software utilities may allow other usages.

Email Address is the address to which your CSR will be sent.

About the Private Key Password dialog

Figure 7-3 Private Key Password dialog



Password is the credential that will be required for each action that involves the private key.

Confirm ensures you meant what you typed for Password.



creates a certificate (.pem file) with the private key and enables the password.

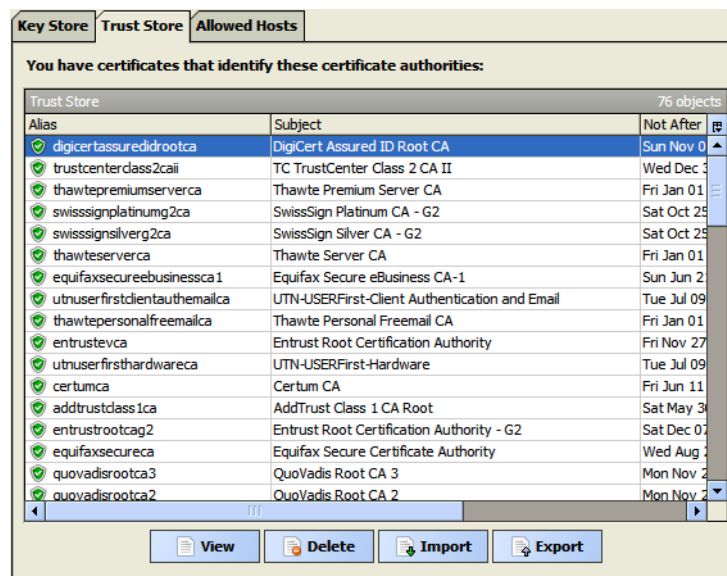


aborts the export process and displays the **Key Store** tab view.

About the Trust Store tab

The trust store contains a list of trusted certificates.

Figure 7-4 Trust Store tab



- “Trust Store columns” on page 7-5
- “Trust Store buttons” on page 7-5

Trust Store columns

Trust Store columns are similar to those for the Key Store. See “Key Store columns” on page 7-3.

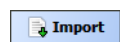
Trust Store buttons



displays certificate details.



removes the certificate from the **Trust Store**.



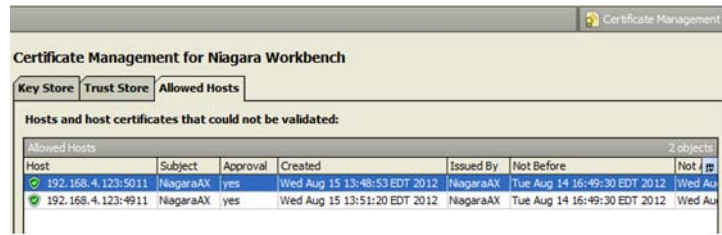
imports a certificate into the **Trust Store**. If the private key is part of the certificate, it is ignored. Private keys do not belong in the **Trust Store**.



saves a copy of the selected certificates to the local hard disk using the .pem extension.

About the Allowed Hosts tab

Figure 7-5 Allowed Hosts tab

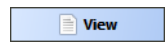


- “Allowed hosts columns” on page 7-6
- “Allowed hosts buttons” on page 7-6

Allowed hosts columns

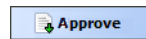
- **Host** identifies the server.
- **Subject** displays the Distinguished Name.
- **Approval** indicates the servers within the network to which the a client may connect. If approval is set to **no**, the system will not allow the client to connect.
- **Created** is the date the record was created.
- **Issued By** identifies the Certificate Authority.
- **Not Before** is the date before which the certificate is valid.
- **Not After** is the date after which the certificate becomes invalid.
- **Key Algorithm** identifies the algorithm used to generate the key.
- **Key Size** identifies the size of the key.
- **Signature Algorithm** identifies the algorithm used to digitally sign the certificate.
- **Signature Size** identifies the signature size.
- **Valid** indicates if this server is trusted.

Allowed hosts buttons



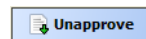
View

displays server details.



Approve

designates server as an allowed host.



Unapprove

designates server as an unallowed host. Any attempted communication will be terminated.



Delete

deletes the server from the list.

About the Certificate Signing dialog

Figure 7-6 Certificate signing properties



Signing properties:

Select a certificate signing request to sign: This is the CSR you created.

Not Before Date this certificate becomes valid.

Not After The expiration date for the certificate.

CA Alias is the short name you assigned to the CA certificate whose private key will be used to sign this certificate.

CA Password is the password that protects the private key of the CA certificate being used to sign this certificate.

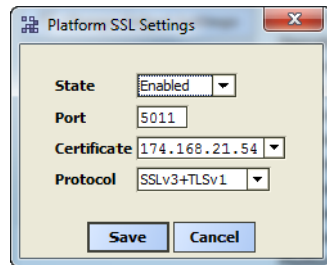
SSL configuration properties

- “[SSL configuration properties](#)” on page 7-7
- “[Fox Service configuration properties](#)” on page 7-8
- “[Web Service configuration properties](#)” on page 7-9

Platform SSL properties

To access this dialog, expand the **Platform** node in the Nav tree, double-click **Platform Administration** and click **Change SSL Settings**.

Figure 7-7 Platform SSL properties



State Enabled or Disabled.

Port The port for SSL.

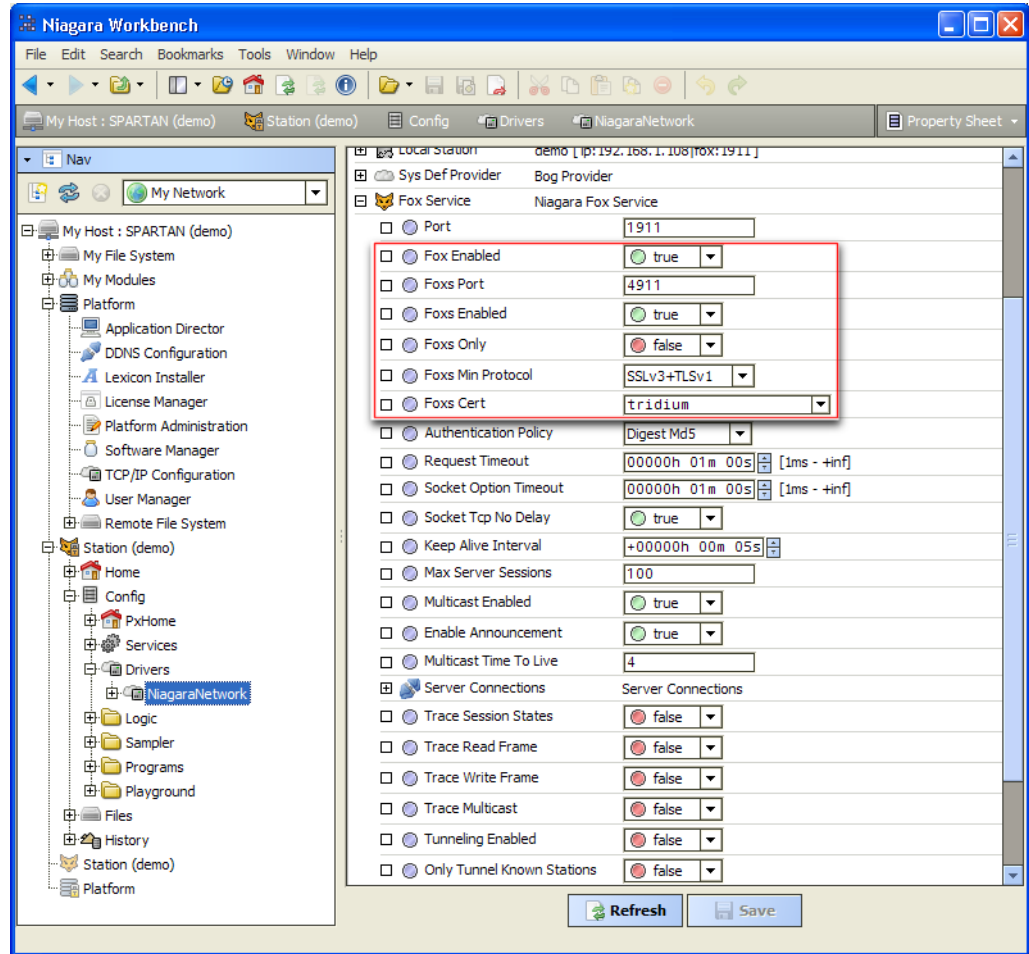
Certificate The **Alias** for the server certificate.

Protocol The security service to use. The default is to use both standard protocols (**SSLv3+TLSv1**). During the handshake, the server and client agree on which protocol to use. It may be one or the other.

Fox Service configuration properties

To view this screen expand the Station in the Nav tree to the **Station > Config > Drivers** level, right-click **Niagara Network** and click **Views > Property Sheet**.

Figure 7-8 Fox Service properties



Fox Service properties related to SSL:

Port indicates the port number for standard Fox communication. The default is 1911.

Fox Enabled, when set to **true**, turns on a standard Fox connection (no SSL security) using port 1911. When set to **false**, turns off the standard Fox connection causing attempts to connect using Fox port 1911 to be ignored.

Foxs Port indicates the port number for secure Fox communication. The default is 4911.

Foxs Enabled, when set to **true**, turns on secure Fox communication using port 4911. When set to **false**, turns off the secure Fox connection causing attempts to connect using Foxs port 4911 to be ignored.

Foxs only, when set to **true** redirects any attempt to connect using a connection that is not secure (Fox alone) to Foxs. When set to **false**, does not redirect attempts to connect using Fox alone.

Note: To ensure secure communication, set **Fox Enabled** to **false** and **Foxs Enabled** to **true**. In this configuration, how you set **Foxs Only** does not matter because no redirection is required. During the transition to a secure network, a station may need to continue to accommodate Fox Service communication that is not secure (it may take time to change all old port number pointers in the system). If you set all three properties: **Fox Enabled**, **Foxs Enabled**, and **Foxs Only** to true, the station will continue to respond on port 1911 (not secure) and redirect the communication to the secure connection. Later you can return and set **Fox Enabled** to false.

When you set **Fox Enabled** to false, you must also set **Http Enabled** to false.

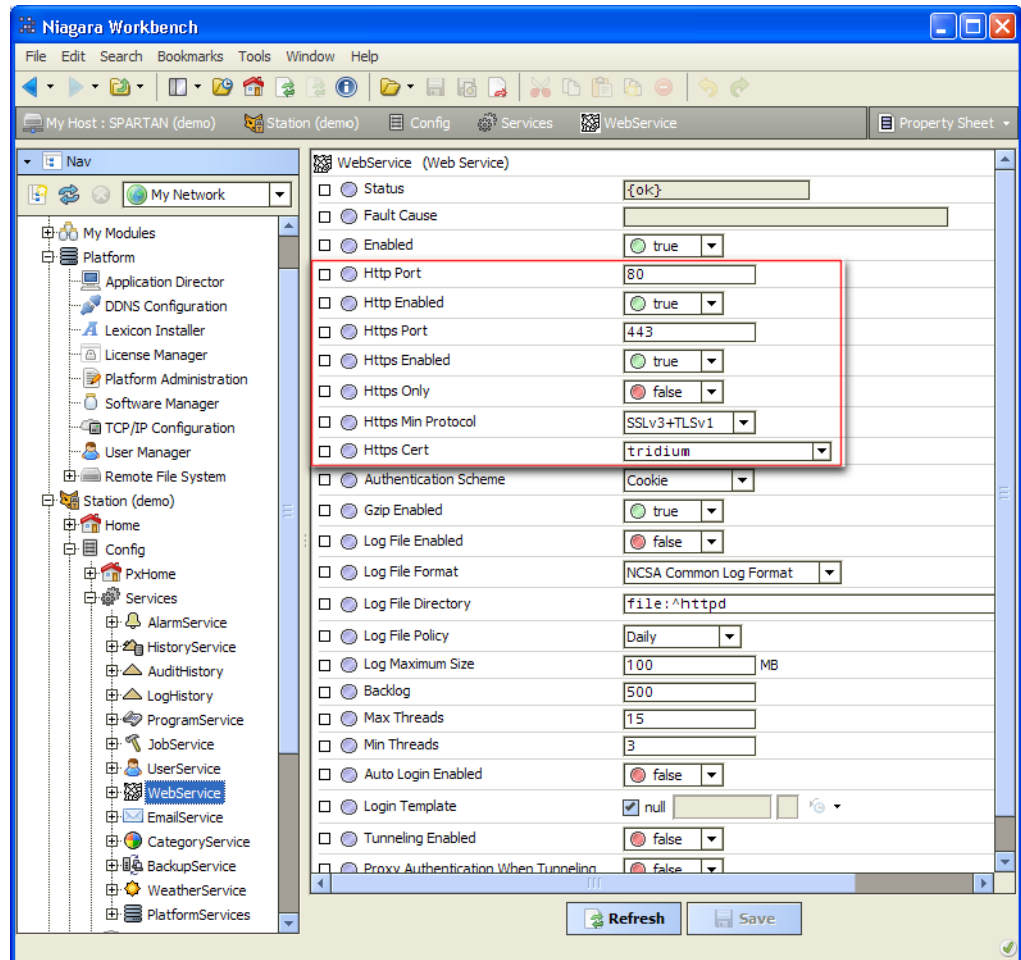
Foxxs Min Protocol selects the communication protocol. The default is to use both standard protocols (**SSLv3+TLSv1**). During the handshake, the server and client agree on which protocol to use. It may be one or the other.

Foxxs Cert allows you to select a unique certificate for Foxxs only.

Web Service configuration properties

To view this screen expand the Station in the Nav tree to the **Station > Config > Services** and double-click **Web Service**.

Figure 7-9 Web Service properties



Web Service properties related to SSL:

Http Port indicates the port number for standard Http communication. The default is 80.

Http Enabled, when set to **true**, turns on a standard Http connection (no SSL security) using port 80. When set to **false**, turns off the standard Http connection causing attempts to connect using Http port 80 to be ignored.

Https Port indicates the port number for secure Http communication. The default is 443.

Https Enabled, when set to **true**, turns on secure Http communication using port 443. When set to **false**, turns off the secure Https connection causing attempts to connect using Https port 443 to be ignored.

Https only when set to **true** redirects any attempt to connect using a connection that is not secure (Http alone) to Https. When set to **false**, does not redirect attempts to connect using Http alone.

Note: To ensure secure communication, set **Http Enabled** to **false** and **Https Enabled** to **true**. In this configuration, how you set **Https Only** does not matter because no redirection is required.

*During the transition to a secure network, a station may need to continue to accommodate Web Service communication that is not secure (it may take time to change all old port number pointers in the system). If you set all three properties: **Http Enabled**, **Https Enabled**, and **Https Only** to true, the station will continue to respond on port 80 (not secure) and redirect the communication to the secure connection. Later you can return and set **Http Enabled** to false.*

*If you set **Http Enabled** to false, you must also set **Fox Enabled** to false.*

Https Min Protocol selects the security protocol. The default is to use both standard protocols (**SSLv3+TLSv1**). During the handshake, the server and client agree on which protocol to use. It may be one or the other.

Https Cert allows you to select a unique certificate for Https only.