

Information and/or specifications published here are current as of the date of publication of this document. Tridium, Inc. reserves the right to change or modify specifications without prior notice. The latest product specifications can be found by contacting our corporate headquarters, Richmond, Virginia. Products or features contained herein are covered by one or more U.S. or foreign patents. 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. Complete Confidentiality, Trademark, Copyright and Patent notifications can be found at: <http://www.tridium.com/galleries/SignUp/Confidentiality.pdf>. © 2013 Tridium, Inc.

JACE, Niagara Framework, Niagara AX Framework and the Sedona Framework are trademarks of Tridium, Inc.

## NiagaraAX CryptoService (SSL)

The NiagaraAX CryptoService provides Secure Socket Layer (SSL) services for all stations (Windows-based, Linux-based, QNX-based) running NiagaraAX 3.6 and earlier builds. In addition, it provides the same services for JACE 2, 4 and 5 model controllers (which use the IBM J9 VM) running NiagaraAX 3.7 and newer builds.

Other platforms running NiagaraAX 3.7 and newer builds use the SSL Toolset. For more information, see the 3.7 *SSL Connectivity Guide*.

This document includes the following topics.

- “Capabilities” on page 2
- “Prerequisites” on page 2
- “Install the CryptoService modules” on page 2
  - “Install the crypto.jar module” on page 2
  - “Install the TKS Provider” on page 3
- “Configure CryptoService” on page 3
  - “Configure the CryptoService” on page 4
  - “Enable the https protocol” on page 5
- “Install a signed certificate” on page 6
  - “Generate the key pair for the certificate request” on page 6
  - “Generate the certificate request” on page 6
  - “Import the signed certificate” on page 7
  - “Check the Key Store” on page 7
  - “Expired certificates” on page 8
  - “Add a certificate from a trusted entity” on page 8
  - “Keytool options” on page 8
- “Import a private key to a TKS Key Store” on page 9
  - “Set up OpenSSL” on page 9
  - “If necessary, create a .pem file” on page 9
  - “Prepare the .pem file” on page 10
  - “Export the .pem file to a PKCS12 file” on page 10
  - “Import, verify and edit the Key Store” on page 11
  - “Set up the browser” on page 12
- “Configure email for SSL encryption” on page 14
- “About NiagaraAX cryptographic service” on page 14
  - “How it works” on page 15
  - “CryptoService terminology” on page 15
  - “CryptoService components” on page 15
  - “Frequently Asked Questions” on page 16
- “Troubleshooting” on page 16
- “Document change log” on page 17

## Capabilities

CryptoService provides:

- Secure WebService connectivity using https.  
***Note:** CryptoService encrypts only WebService connections. Fox and other connections remain unencrypted.*
- The ability to use a unique, signed certificate for each server (JACE).
- Encrypted email transmission.

## Prerequisites

- One or more JACE platform(s) running NiagaraAX-3.6 and earlier (CryptoService was first introduced with NiagaraAX-3.3) or a JACE-2, JACE-4, JACE-5 platform running NiagaraAX-3.7
- A license for the CryptoService feature: Before you begin, install the CryptoService license, which is available in an email message, by downloading it from Niagara Central, or by using the License Manager view. For installation details, see the “License Manager” section of the *Platform Guide*. After installing the license, restart the station.
- The following Workbench modules:
  - **crypto.jar** provides SSL services.
  - **tridiumprovider.jar** provides a way to create a root certificate and use it to sign individual server certificates.
- Java Standard Edition (Java SE) for creating the Key Store and certificates. Java SE consists of:
  - Java Development Kit (JDK). The JDK includes the keytool.exe—a command line utility for creating the certificate. Because Oracle limits the distribution of individual developer kit tools, such as keytool.exe, you must install the JDK on your system. It is available for free at oracle.com.
  - Java Runtime Environment (JRE), which is required to run the JDK.***Note:** The JDK includes the JRE. Only one download and install are required.*
- A PC or laptop from which to perform the installation steps.

## Install the CryptoService modules

This section covers these procedures:

- [“Install the crypto.jar module”](#) on page 2
- [“Install the TKS Provider”](#) on page 3

### Install the crypto.jar module

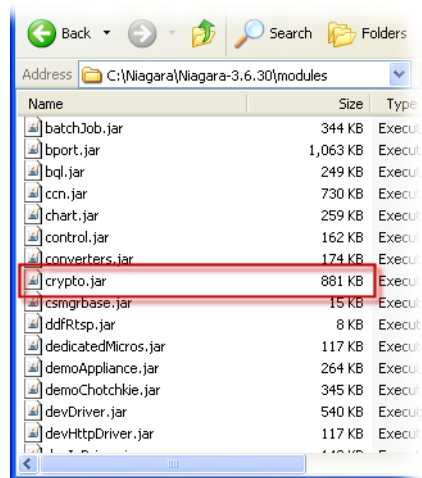
The latest `crypto.jar` may or may not be present in the Workbench **modules** folder. Even if `crypto.jar` is present, it may need to be replaced.

Users of NiagaraAX-3.5.25.1 and earlier versions have a *stub* `crypto.jar` file instead of the actual module. This was because, by law, the US Government required Tridium to verify that the transfer of encryption software was not being made to a “denied person.” This restriction has been lifted. NiagaraAX-3.5.25.2 and later versions include the normal `crypto.jar`.

If you have a stub file, you need the correct `crypto.jar`. The email, which you may have received with the license, may also include the correct `crypto.jar` file or you may download `crypto.jar` from Niagara Central.

- Step 1 To check the Workbench version, **Help > About**.
- Step 2 Check the **Niagara\version\modules** folder (where **version** is the version of NiagaraX you are using).

**Figure 4-1** Normal crypto.jar file



A crypto stub file is about 6KB. The normal `crypto.jar` file in the example above is 881 KB.

- Step 3 If you have a stub file or are in doubt, download the latest module .jar files from Niagara Central.
- Step 4 Save the modules in the **Niagara\version\modules** folder.

### Install the TKS Provider

If you previously installed **StandaloneTksProvider.jar** in your JRE, delete it from the **lib/ext** directory.

- Step 1 If you haven't already done so, download the Tridium Key Store (TKS) Provider jar file, **TridiumProvider.jar**, from Niagara Central.
- Step 2 Install the jar into the **lib/ext** directory of your chosen JRE.



**Caution** Do not install *TridiumProvider.jar* into the Niagara JRE.

- Step 3 Add the following line to the list in the **lib/security/java.security** file, which is located in your JRE:
- ```
security.provider.nn=com.tridium.crypto.TksProvider
```
- where *nn* is a sequential number.

**Note:** Make sure that the number following "security.provider" is sequential.

## Configure CryptoService

The CryptoService is available in the **crypto** palette.

These procedures should be carried out on a PC. They start from the Workbench main menu, and assume an established connection to the target platform.

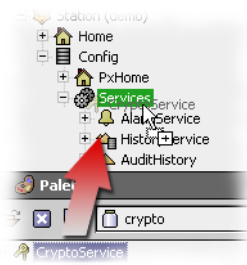
This section covers these procedures

- "Configure the CryptoService" on page 4
- "Enable the https protocol" on page 5

## Configure the CryptoService

- Step 1 Drag and drop CryptoService from the palette onto the Services node in the Nav tree.

**Figure 4-2** Dropping CryptoService on the Services node



- Step 2 To confirm the CryptoService is configured correctly, double-click the CryptoService component in the Nav tree.

The CryptoService property sheet view appears in the view pane.

**Figure 4-3** Example of a CryptoService property sheet



- Step 3 Expand the **Ssl** node and confirm the CryptoService properties.

Figure 4-3 shows the default properties.

- **Console Debug**  
true causes the system to display details of the SSL handshake in the station output view pane. This information is valuable for debugging communication problems, and requires SSL expertise to decipher.  
false is the default. It disables the console. Unless you are an experienced in debugging SSL, leave Console Debug set to false.
- **Key Store**  
Defines the path to the Tridium Key Store (TKS) file that contains keys and certificates. By default, this string value points to: file: !security/ssl.tks, which is located under the security folder of the NiagaraAX installation directory. You may change the path to use your own TKS. To create your own TKS, see [“Install a signed certificate”](#) on page 6.
- **Key Store Password**  
The string password protects the Key Store file and is set when the Key Store is generated. The password for the default Tridium Key Store is “tridium.” If you intend to use the default Key Store, do not change this password. (For obvious security reasons, you are encouraged to create your own TKS, certificates and strong password.)
- **Key Store Type**  
The TKS is the only supported type of store for the CryptoService feature.
- **Trust Store**  
While the Trust Store is not required to use the CryptoService, the value of this property must point to the same ssl.tks file defined for the Key Store.
- **Trust Store Password**  
While the Trust Store is not required to use the CryptoService feature, if you change the Key Store-password, the Trust Store password must also be changed to the same password you set up for the Key Store.
- **Trust Store Type**  
The TKS is the only supported type of store for the CryptoService feature.

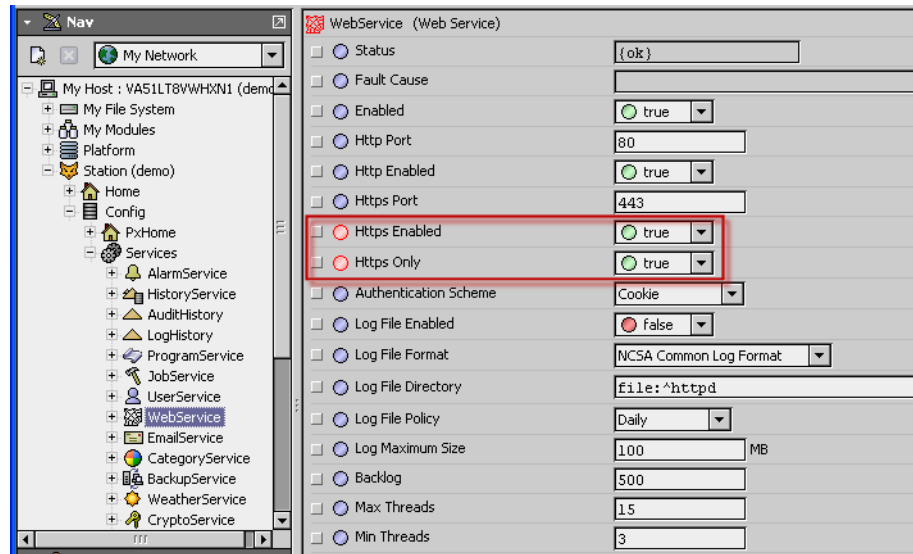
Step 4 Update the properties as needed.

**Note:** If you change the location or password of the Key Store, make sure you also change the location of the Trust Store and password. The Trust Store must point to the same file used for the Key Store and the password must be the same.

### Enable the https protocol

Step 1 To configure the station for https, double-click the station's WebService under Services in the Nav tree. The WebService property sheet appears in the view pane.

**Figure 4-4** WebService property sheet with https properties enabled



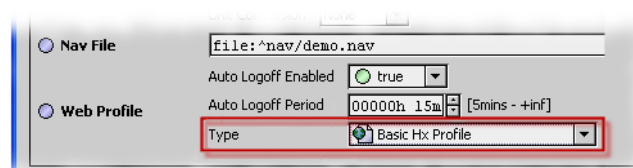
- **Https Enabled**  
Turns on SSL encryption.
- **Https Only**  
Restricts communications to the https protocol only. This property is not required for secure https communications, however, to strengthen the security of web-based access to the system, configure **Https Only** with a value of true, configure users for Hx profiles, and use only Hx views. Doing so will prevent un-encrypted http connections from being made.

Step 2 Set **Https Enabled** to true, and, to ensure security, **Https Only** also to true.

Step 3 To configure a user profile for the https protocol, double-click UserServices in the Nav tree and double-click the user record.

The UserManager property sheet appears.

**Figure 4-5** User configured for Basic Hx profile



When using a Workbench-type profile (uses a Java applet), communication between the browser client and the station uses both http and Fox connections. With CryptoService it is only possible to encrypt the http connection by using an https socket. Even though the https communication is encrypted, some communication still occurs over the unencrypted Fox connection.

The most robust security implementation uses Hx profiles instead of Workbench profiles. When using an HX profile with an https connection, all communication between the browser client and station occurs over an encrypted https connection.

Step 4 Select the profile from the drop-down list.

For more information about web profiles, see "About Web Profiles" in the *NiagaraAX User Guide*.

Once configured, if a client browser attempts an http connection, the station redirects the connection from http to https.

## Install a signed certificate

When a station that contains CryptoService boots, the software creates a self-signed certificate, which can be used to encrypt https data. To strengthen system security, it is better to use an actual signed certificate for each server (JACE). In addition to encryption, signed certificates provide identity verification.

Certificates may be signed by a third-party Certificate Authority (CA) or you may serve as your own CA by creating a self-signed root certificate and using it to sign your server certificates.

The procedures that follow use the Oracle® Keytool utility to create server certificates, generate Certificate Signing Requests (CSRs) and import signed certificates. To actually sign your own certificates, you need OpenSSL, the Niagara AX 3.7 SSL Toolkit, or some other third-party application.



### Caution

*Serving as your own CA will provide secure communications if and only if you physically guard the root certificate's private key. Consider storing the computer used to sign server certificates in a vault or other secure location. Never send the root certificate's private key as an email attachment and be careful to exclude the private key when exporting a root certificate to a thumb drive or other storage device.*

This section covers these procedures and sections:

- “Generate the key pair for the certificate request” on page 6
- “Generate the certificate request” on page 6
- “Import the signed certificate” on page 7
- “Check the Key Store” on page 7
- “Expired certificates” on page 8
- “Add a certificate from a trusted entity” on page 8
- “Keytool options” on page 8

For a summary of the most likely Keytool options, see “Keytool options” on page 8. Comprehensive help for using Keytool is available on Oracle’s web site (at current writing: <http://docs.oracle.com/javase/7/docs/technotes/tools/windows/keytool.html>). For more information about how CryptoService works, see “How it works” on page 15.

### Generate the key pair for the certificate request

A “key pair” refers to a set of private and public software keys used to encrypt data and to verify identity

- Step 1 Open a command prompt and make sure that **jre/bin** is in your PATH.
- Step 2 Change directories to the security directory for your Niagara installation.
- Step 3 As a backup, rename the existing **ssl.tks** file to **ssl.tks.orig**.
- Step 4 Run the keytool utility with the following command:
 

```
keytool -genkey -alias tridium -keystore ssl.tks -storepass tridium -storetype TKS -keyalg RSA -keysize 2048
```

It may be necessary to adjust the **-keyalg** and **-keysize** arguments based on the requirements of your Certificate Authority.
- Step 5 When prompted for your first and last name, enter the base domain name of the dns entry for your server, for example, tridium.com.
- Step 6 Answer the remaining questions as accurately as possible.
- Step 7 When prompted to enter a password for the key pair, press Enter to use the Key Store password.
- Step 8 As a backup, make a copy of the new **ssl.tks**, naming it **ssl.tks.new**.

### Generate the certificate request

- Step 1 Create the Certificate Signing Request (CSR) using this command:
 

```
keytool -certreq -alias tridium -keystore ssl.tks -storepass tridium -file certreq.cer -storetype TKS
```

This creates a new file called **certreq.cer**.
- Step 2 Submit this CSR file to your Certificate Authority (CA) along with any other information they require.

When the CA completes the signing process, you will receive an email or a file that looks something like this:

**Figure 4-6** Certificate chain

```
-----BEGIN CERTIFICATE-----
MIIFUTCCBmgAwIBAgIQdYL06pVxhgnBQNHptRI6NzANBgkqhkiG9w0BAQUFADCB
yZELMAkGA1UEBhMCVVMxZzAVBgNVBAAoTDI1ZmlTaWduLCBjbmuMTAwLgYDVQQL

etc...

CU+QUBAAZAN4sWbSVL4KULDC4AMSEHk86mB56NUL0SH23H51140NCKZDUM/HCA
myEslcmph/HcpdBAA7guhGvvqkCytC4Bry5IGedPgYgZStIudAlPdkeUtC5/mvry0
ctI785MRsEhtCsmryqIVrYrecYb8
-----END CERTIFICATE-----
-----BEGIN CERTIFICATE-----
MIIFDCCBGSgAwIBAgIQfju3hLvGVKvSuNZ37MOUqDANBgkqhkiG9w0BAQUFADCB
jDELMakGA1UEBhMCVVMxZzAVBgNVBAAoTDI1ZmlTaWduLCBjbmuMTAwLgYDVQQL

etc.

wDIO3oYL2bT/66tw46Kx3Q/Z02pp7YW+BRvKejBYXN9FJxsXEkPKpz4SRvSQL15Y
Bst7q03nK0LZQ4/LE+hufigvx08JdgGd4o4cOvZ6o4MQaMgX/0lwNjC+4PWuKfmrK
mr+RhpSkc72cEE09//XsYesTmetY3nOuqXAQqIH41z3KAzqNgohXXcF8W1F3ytTT
qlyWbMkJonRpXbE3U0rWI33ywiS7Lak8n4dR8tNoqqIrP6sDtvolx9/+qTPWGnYy
tV0P/hcJt5CbqE7008EnPQ==
-----END CERTIFICATE-----
-----BEGIN CERTIFICATE-----
MIIEVzCCAz+gAwIBAgIQFoFkpCjKt+rEvGfsbk1VDANBgkqhkiG9w0BAQUFADCB
jDELMakGA1UEBhMCVVMxZzAVBgNVBAAoTDI1ZmlTaWduLCBjbmuMTAwLgYDVQQL
BydGbn3IgVGvzdCBQdXJwb3NlcyBpbmxc5LiAgTm8gYXNedXJhbmNlcy4xMjAwBgNV

etc.

WOGp13vzWRvWggu6dm2WVKJfvPfmS1mAP0YmTcIwjdINX1U6sSsJEOnlTR9zCoo
4cK8wVcWZpbuPZb5geszhS7YeABUPIAAfF1YQCIMULtpa6HFzzm7edf72N3HfWE
aQNg95KnKGrDUI=
-----END CERTIFICATE-----
```

Step 3 Save this certificate chain to a file and give it a name, such as **certchain.cer**.

### Import the signed certificate

Step 1 Run the following command on the same Key Store that you used to generate the initial CSR.

```
keytool -importcert -trustcacerts -file certchain.cer -keystore ssl.tks -
storepass tridium -storetype TKS -alias tridium
```

Step 2 If a message indicates that this certificate is not trusted, "Install reply anyway?" click **Yes**.

### Check the Key Store

Step 1 Dump the contents of the Key Store with the following command:

```
keytool -list -alias tridium -keystore ssl.tks -storepass tridium -storetype
TKS -v
```

The first few lines should read something like:

```
Alias name: tridium
Creation date: July 31, 2014
Entry type: PrivateKeyEntry
```

Step 2 Verify that the third line reads, **PrivateKeyEntry**.

Step 3 Look for the **Owner** and **Issuer** lines in the first certificate. For example:

```
Certificate[1]:
Owner: CN=foo.com, OU=engineering, O=tridium, L=richmond, ST=virginia, C=us
Issuer: C=us, ST=virginia, L=richmond, O=tridium, OU=engineering, CN=interme-
diateca
```

Quick reference to the acronyms:

- CN = Common Name is the domain name for which the certificate will be used including the sub-domain. This is a required field.
- OU = Organizational Unit name is the full name of the organization or department.
- O = Organization name is the full name of the organization.
- L = Locality name is the geographical city or town where you or your organization resides.
- ST = The full name of the state or province for you or your organization.
- C = the capitalized, two-letter country code for you or your organization.

Step 4 Verify that the owner is the end certificate that you signed.

Step 5 Look through each subsequent certificate to make sure that the **Owner** is the same as the **Issuer** on the previous certificate. For example:

```
Certificate[2]:
Owner: C=us, ST=virginia, L=richmond, O=tridium, OU=engineering, CN=interme-
diateca
Issuer: c=us, ST=virginia, L=richmond, O=tridium, OU=engineering, CN=rootca
```



- Step 6 Verify that the last certificate is a self-signed certificate where the owner and issuer are the same. For example:

```
Certificate[3]:  
Owner: C=us, ST=virginia, L=richmond, O=tridium, OU=engineering, CN=rootca  
Issuer: C=us, ST=virginia, L=richmond, O=tridium, OU=engineering, CN=rootca
```

Your **ssl.tks** now contains a signed certificate.

**Note:** *Additional information:*

- The signed certificate you installed only validates correctly the domain for which you created it. Do not attempt to use this certificate on another domain.
- Your Certificate Authority may have other requirements and instructions, and should be able to assist you with any trouble.
- The certificate chain must be installed into the Key Store that contains the matching private key entry.

## Expired certificates

If a certificate expires, repeat this process for the same key (alias).

## Add a certificate from a trusted entity

If your installation requires a certificate from a system located outside the NiagaraAX environment, add the certificate to the Key Store using the import command as documented in the BSSLProvider.bajadoc, substituting an arbitrary alias for “peer” in the command.

When the peer certificate expires, delete it and import a new certificate.

## Keytool options

The utility is fully documented on Oracle’s web site (at publication release the Keytool documentation is located at: <http://docs.oracle.com/javase/7/docs/technotes/tools/windows/keytool.html>). This section explains common options.

- **alias**  
Specifies the name that should be used to refer to the certificate in the Key Store. When self-signing the certificate, this should be the same name as that used to create the certificate. Aliases are not case sensitive (for example “MyCertificate” and “mycertificate” would refer to the same Key Store entry.)
- **certreq**  
Generates a certificate request.
- **delete**  
Deletes a certificate
- **dname**  
Specifies the subject to use for the certificate. The subject typically contains at least a CN attribute. The attributes are as follows:
  - C specifies the Country in which the certificate will be used.
  - CN is the fully-qualified Common Name of the system on which the certificate will be installed.
  - O (capital letter O) specifies the name of the Organization (or company).
  - OU stands for Organizational Unit.
- **file**  
Specifies the path to the file containing the signed certificate. The file should be in either the DER-encoded binary format or the base64-encoded ASCII format.
- **genkey**  
Creates the certificate.
- **import**  
Imports a certificate.
- **keyalg RSA**  
Keyalg stands for “key algorithm” and specifies the algorithm that should be used to generate the private key. This should almost always be **rsa**.
- **keypass**  
Specifies the password that should be used to protect the private key in the Key Store. If you don’t provide a password, you will be prompted for it.
- **keystore**  
Specifies the path to the Key Store file. The file will be created if it does not already exist. The default Tridium Key Store (tks) path used by the directory server is **config/keystore**.
- **provider**  
Specifies the name of the cryptographic service provider’s master class file when the service provider is not listed in the security properties file.



- **storepass**  
Specifies the password that should be used to protect the contents of the Key Store. If you don't provide a password, you will be prompted for it. The directory server expects the password used for the `-keypass` and `-storepass` options to be the same.
- **storetype**  
Specifies the Key Store type that should be used. For the NiagaraAX Key Store, the value should always be **TKS**.
- **validity**  
Specifies the length of time in days that the certificate will be valid. The default validity is 90 days.

## Import a private key to a TKS Key Store

If you have a file that includes a signed certificate with a private key, which does not match the private key currently in the ssl.tks Key Store, this procedure explains how to import the file into the existing ssl.tks Key Store for the JACE.

To complete this procedure you will need:

- Administrator access to Windows.
- The file that contains the certificates, and the private and public keys. This file is most likely a PKCS12-type file. Its file extension may be .p12 or .pfx or .pem.
- The password to access this file should be provided with the file from the CA.
- A text editor.

This section covers these procedures:

- “Set up OpenSSL” on page 9
- “If necessary, create a .pem file” on page 9
- “Prepare the .pem file” on page 10
- “Export the .pem file to a PKCS12 file” on page 10
- “Import, verify and edit the Key Store” on page 11
- “Set up the browser” on page 12

### Set up OpenSSL

- Step 1 Download a copy of OpenSSL. When this Engineering Note was released, OpenSSL could be found at this internet location: <https://code.google.com/p/openssl-for-windows/>.
- Step 2 Extract the zip file to a location on the PC (for example, c:\ProgramFiles\openssl-0.9.8k\_x64).
- Note:** *There is no installation file to run.*
- Step 3 Update the Windows path variable to include the bin sub-folder of the extracted files from the previous step. (for example, c:\Program Files\openssl-0.9.8k\_x64\bin).

### If necessary, create a .pem file

The result of this procedure is a .pem file, which is required by the TKS Key Store. If you already have a .pem file, continue with “Prepare the .pem file” on page 10.

- Step 1 Save the Key Store file that contains the public and private keys and certificates to a working folder on your PC (for example, c:\NiagaraCerts\)
- Step 2 Open a Windows command prompt.
- Step 3 If the Key Store file is a PKCS12-type file, use OpenSSL to export the PKCS12 file to a .pem file using the following command:

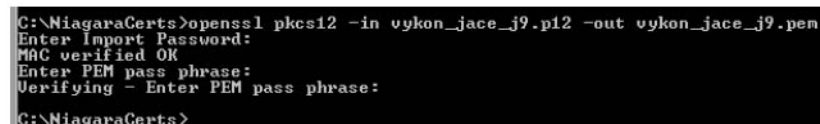
```
openssl pkcs12 -in PKCS12_file -out PEM_file
```

where *PKCS12\_file* is the name of the source PKCS12-type file, and *PEM\_file* is the resulting file.

OpenSSL prompts you to enter the password for the PKCS12 Key Store and to double-enter a new password for the new .pem file.

**Figure 4-7** Example—export PKCS12 file to a .pem file

```
openssl pkcs12 -in vykon_jace_j9.p12 -out vykon_jace_j9.pem
```



```
C:\NiagaraCerts>openssl pkcs12 -in vykon_jace_j9.p12 -out vykon_jace_j9.pem
Enter Import Password:
MAC verified OK
Enter PEM pass phrase:
Verifying - Enter PEM pass phrase:
C:\NiagaraCerts>
```

- |        |                                                                                                                                             |
|--------|---------------------------------------------------------------------------------------------------------------------------------------------|
| Step 1 | Open the .pem file using a text file editor.                                                                                                |
| Step 2 | Look for and delete any extra text outside of the <b>BEGIN</b> and <b>END</b> marker lines that demarcate the private key and certificates. |

**Figure 4-8** *Text to be deleted*

```
Bag Attributes
  local:cityID: 06 25 00 00 B0 3E D4 E4 8A 83 E3 E3 69 2A 6E 89 C8 32 6A BC
  subject: C=US,192.168.4.136/O=VYKON/O=VYKON,ST=Charlotte/ST=NC/C=US
  issuer: C=VYKON, C=Charlotte, C=VYKON/O=VYKON/L=Charlotte/ST=NC/C=US
-----BEGIN CERTIFICATE-----
MIIEhDCCA2ygAwIBAgIRMcUeFhnetVLTEj;5MA0GCSqGSIb3DQEBCwUAMG4xh3Ae
BgNVBAMMFVZ5a2UeIudvYyVybVNVkAwF02SBKOTEMwGA1UeFVnIzBk24xMDMj
BgNVBAMCMBVZ5a2UeIudvYyVybVNVkAwF02SBKOTEMwGA1UeFVnIzBk24xMDMj
-----END CERTIFICATE-----
```

In the example above, the first four lines related to the **Bag Attributes** need to be deleted.

**Figure 4-9** *Text to keep*

```
-----BEGIN RSA PRIVATE KEY-----
Proc-Type: 4,ENCRPTED
DEK-Info: DES-EDS-CBC, 46429E607A2CFF76

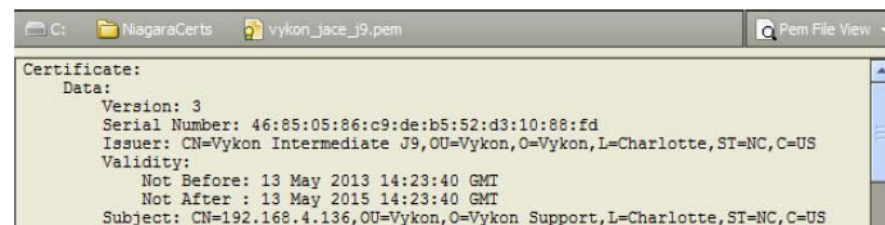
q0R1w3GJjven29nVYwtS180aHkSboTEgKbzm1gqpp3KsDVvnmnQDa6vFmXvDIIPkH
mfz1w5w9cSf68T1xzoGGP7rEtpJYUgYUj1IWBZ6e4pYd3jvQHSTDqkGMCAlnm3j+
nqKzED+kQb1Sp3Qz8r9uQoDcm1K5SjNtGrUTZvB+421upJyk1u5ZkPCFWD5bX
```

The highlighted text above belongs in the file and should not be deleted. This text indicates that the key is encrypted and identifies the type of encryption used.

If Niagara AX 3.7 is available, you can use it to verify that the .pem file contains no unsupported text.

- Step 3** To test the file using NiagaraAX 3.7 or a newer build, view the file using **PEM File View**. If the file contains only valid text, it displays information about the certificate.

**Figure 4-10** Text from a valid .pem file as displayed using NiagaraAX 3.7 PEM File View



If the file contains unsupported text, the above screen is blank. If you see certificate information, your file contains no unsupported text.

## Export the .pem file to a PKCS12 file

- Step 1 Using OpenSSL at a command prompt, enter this command:
- ```
openssl pkcs12 -export -PEM_file -out PKCS12_file
```
- where *PEM file* is the source file, and *PKCS12 file* is the name of the resulting PKCS12-type file.

**Figure 4-11** Example—exporting a .pem file to a PKCS12 file

```
openssl pkcs12 -export -in vykon_jace_j9.pem -out vykon_jace_j9_v1.p12
```

```
C:\NiagaraCerts>openssl pkcs12 -export -in vykon_jace_j9.pem -out vykon_jace_j9_v1.p12
Loading 'screen' into random state - done
Enter pass phrase for vykon_jace_j9.pem:
Enter Export Password:
Verifying - Enter Export Password:

C:\NiagaraCerts>
```

OpenSSL prompts you to enter the password you created for the .pem file in a previous step, or which was provided by the CA (if the .pem file was sent from the CA).

- |        |  |
|--------|--|
| Step 2 | Install the TKS Provider and configure the JRE. For the procedure, see <a href="#">“Install the TKS Provider”</a> on page 3.                                   |
| Step 3 | Update the Windows path variable to include the bin sub-folder of the JRE installation.  |
| Step 4 | Copy the ssl.tks file from the Workbench security sub-folder (file:!security/ssl.tks) to the working folder on your PC (for example, c:\NiagaraCerts\ssl.tks). |

## Import, verify and edit the Key Store

- Step 1 To import the source Key Store (PKCS12 file) to the TKS Key Store (ssl.tks file), execute this command at the command prompt:

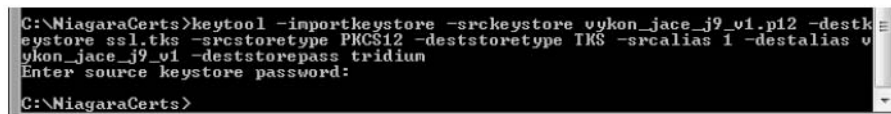
```
keytool -importkeystore -srckeystore PKCS_File -destkeystore TKS_file -
srcstoretype src_store_type -deststoretype dest_store_type -srcalias src_alias
-destalias dest_alias -deststorepass dest_store_password
```

where:

- *PKCS\_File* is the name of the PKCS12-type file.
- *TKS\_file* is the name of your TKS file, usually **ssl.tks**.
- *src\_store\_type* is **PKCS12**.
- *dest\_store\_type* is **TKS**.
- *src\_alias* is the digit **1** (one).
- *dest\_alias* is a name that represents the JACE you are configuring.
- *dest\_store\_password* is the password for the ssl.tks file. The default password is **tridium**. If the default password has changed, use the appropriate password.

**Figure 4-12** Example—importing the Key Store

```
keytool -importkeystore -srckeystore vykon_jace_j9_v1.p12 -destkeystore ssl.tks -srcstoretype
PKCS12 -deststoretype TKS -srcalias 1 -destalias vykon_jace_j9_v1 -deststorepass tridium
```



- Step 2 To verify that the key and certificate were imported, execute this command:

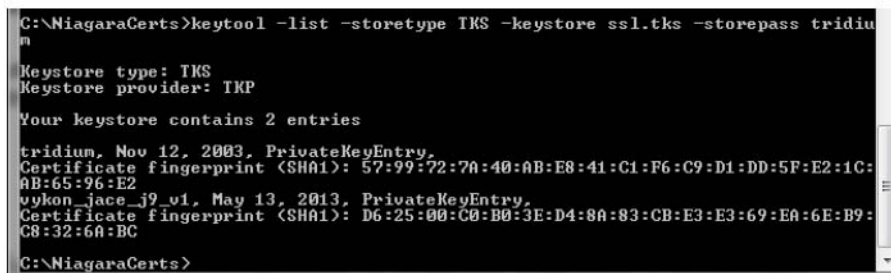
```
keytool -list -storetype store_type -keystore store_file -storepass
store_password
```

where

- *store\_type* is **TKS**.
- *store\_file* is the name of your TKS file, usually **ssl.tks**.
- *store\_password* is the password for the ssl.tks file. The default password is **tridium**. If the default password has changed, use the appropriate password.

**Figure 4-13** Example—imported Key Store

```
keytool -list -storetype TKS -keystore ssl.tks -storepass tridium
```



The example above contains two private key entries. The first entry is the default private key with the alias **tridium**. This entry is no longer needed.

The second entry is the private key that was imported in this step. Its alias is **vykon\_jace\_j9\_v1**.

- Step 3 To delete the private key with the **tridium** alias from the key store, execute this command:

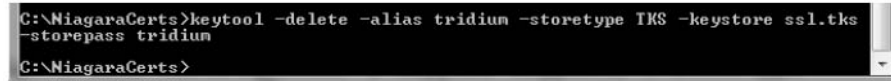
```
keytool -delete -alias key_alias -storetype store_type -keystore store_file -
storepass store_password
```

where:

- *key\_alias* is **tridium**.
- *store\_type* is **TKS**.
- *store\_file* is **ssl.tks**.
- *store\_password* is the password for the ssl.tks file. The default password is **tridium**. If the default password has changed, use the appropriate password..

**Figure 4-14** Example—deleting the extra private key

`keytool -delete -alias tridium -storetype TKS -keystore ssl.tks -storepass tridium`



- Step 4 To ensure security, change the password for the Key Store from **tridium**, to your own strong password by entering this command:

```
keytool -storepasswd -storetype store_type -keystore store_file -storepass store_password
```


where:

- `store_type` is **TKS**.
- `store_file` is **ssl.tks**.
- `store_password` is the password for the ssl.tks file. The default password is **tridium**. If the default password has changed, use the appropriate password.

The Keytool utility prompts you to double-enter a new password for the Key Store.

**Figure 4-15** Example—changing the password

`keytool -storepasswd -storetype TKS -keystore ssl.tks -storepass tridium`



The key password and store passwords must be the same or you will receive a padding error when trying to start the https web service.

- Step 5 To change the key password to match the store password, execute this command:

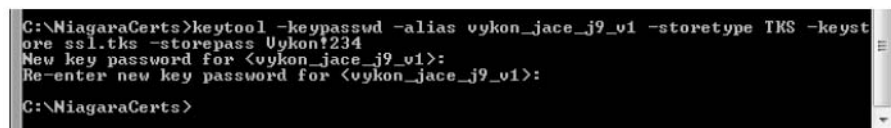
```
keytool -keypasswd -alias key_alias -storetype store_type -keystore store_file -storepass store_password
```

where:

- `key_alias` is the name of your Key Store.
- `store_type` is **TKS**.
- `store_file` is **ssl.tks**.
- `store_password` is the password for the ssl.tks file. The default password is **tridium**. If the default password has changed, use the appropriate password.

**Figure 4-16** Example—changing the key password

`keytool -keypasswd -alias vykon_jace_j9_v1 -storetype TKS -keystore ssl.tks -storepass Vykon!234`



- Step 6 Copy the ssl.tks file to the security sub-folder of the desired host (file!security/ssl.tks).

- Step 7 To update the CryptoService properties in the station, see [“Configure CryptoService”](#) on page 3.

Make sure that the **Key Store Password** and **Trust Store Password** properties are configured with the new password you assigned in a previous step. The Trust Store and Key Store passwords must be the same.

## Set up the browser

If the root certificate used to sign the server’s certificate is not from a well known CA whose root certificate is already in the browser’s trust store, you may need to import the root certificate into the browser Trust Store.

If the CA did not provide a copy of the root certificate, this procedure explains how to locate and prepare the certificate, and import it into the browser.

- Step 1 Open the .pem file you created in [“If necessary, create a .pem file”](#) on page 9 using a text editor.

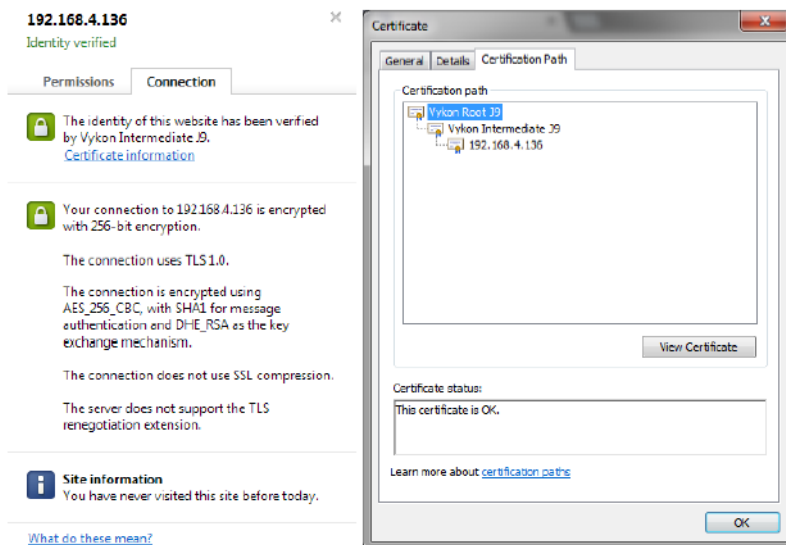
- Step 2**    Locate the root certificate. It should be the last certificate in the chain.

**Figure 4-17** Example of a root certificate

[illegible]

- Step 3 Copy the entire text of the root certificate including the **BEGIN** and **END** marker lines.
- Step 4 Create a new text file and paste the copied text into it.
- Step 5 Save the file using a .cer extension (for example, c:\NiagaraCerts\myLocalCaRoot.cer).
- Step 6 Use your browser tools to import this root certificate file into the client browser's trust store.
- Step 7 Using the browser, make a secure connection to the station (https) and verify that the expected certificate is presented to the browser.

**Figure 4-18** Certificate verified by the browser



- Step 8** If the station will be making SSL connections to other servers that are using certificates signed by the same CA's root certificate, import the root certificate to the `ssl.tks` trust store using this command:

```
keytool -importcert -file root_cert_file -storetype store_type -keystore
store file -storepass store password
```

where:

- *root\_cert\_file* is the name of the .cer file you received from the CA or created that contains the root certificate.
- *store\_type* is **TKS**.
- *store\_file* is **ssl.tks**.
- *store\_password* is the password you created for the Key Store.



**Figure 4-19** Example—root certificate imported into the TKS Key Store

```
keytool -importcert -file myLocalCaRoot.cer -storetype TKS -keystore ssl.tks -storepass
Vykon!234
```

```
C:\NiagaraCerts>keytool -importcert -file myLocalCaRoot.cer -storetype TKS -keystore ssl.tks -storepass Vykon!234
Certificate fingerprint:
MD5: 3E:2B:83:37:2C:C4:69:3B:E0:08:EC:5D:64:30:D5:76
SHA1: 2D:18:C9:3F:DE:2E:E5:F7:C7:E3:73:32:G0:2B:FE:5E:06:36:72:0F
SHA256: 58:6F:81:D0:2E:D9:7D:97:58:22:E7:05:65:C9:91:1D:3B:5D:AD:04:9F:
AF:E1:13:99:9C:D1:8C:F3:3E:5C:6D
Signature algorithm name: SHA256withRSA
Version: 3

Extensions:
#1: ObjectId: 2.5.29.19 Criticality=true
BasicConstraints: [
  CA: true
  PathLen: 2147483647
]
#2: ObjectId: 2.5.29.18 Criticality=false
IssuerAlternativeName [
  Other-Name: Unrecognized ObjectIdentifier: 1.3.6.1.4.1.4131.2
]
#3: ObjectId: 2.5.29.15 Criticality=true
KeyUsage [
  Key Cert Sign
  Crl Sign
]
#4: ObjectId: 2.5.29.14 Criticality=false
SubjectKeyIdentifier [
  KeyIdentifier [
    0000: 4C C7 CE F9 D9 91 18 90 96 29 D5 67 7F 01 52 2A L.....>.g..R=
    0010: DE 64 3C 4E .d<M
  ]
]
Trust this certificate? (no): yes
Certificate was added to keystore
C:\NiagaraCerts>
```

- Step 9 If you imported the CA's root certificate into the ssl.tks, copy the updated ssl.tks file to the host's security sub-folder (file:security/ssl.tks).

## Configure email for SSL encryption

### To configure the email client

- Step 1 Double-click the **EmailService** node under **Services** in the station's Nav tree, then double-click the IncomingAccount or OutgoingAccount in the **Email Account Manager** table. The Edit email account dialog box appears.

**Figure 4-20** Example of the outgoing EmailServices property sheet

Name	Enabled	Transport	Hostname	Port	Use Authentication	Account	Password	Pollrate	Use Ssl
OutgoingAccount	false	Smtp	hostname	25	false	account	--password--	10sec	true

☐ Name: OutgoingAccount  
☐ Enabled: false  
☐ Transport: Smtp  
☐ Hostname: hostname  
☐ Port: 25 [-1 - max]  
☐ Use Authentication: false  
☐ Account: account  
☐ Password: .....  
☐ Pollrate: 00000h 00m 10s [1sec - +inf]  
☒ Use Ssl: true

OK Cancel

- Step 2 Set **Use Ssl** to true.

## About NiagaraAX cryptographic service

CryptoService is an implementation of the industry standard Secure Socket Layer (SSL) technology for encrypting http communication between a browser or email client and JACE server or supervisor web server. It is required for https and secure (SSL) email.

**Note:** The CryptoService provides SSL encryption for the https protocol (Hx web profiles and Hx views). Secondary connections that use the Fox protocol and web Workbench connections remain un-encrypted.

This section covers these topics:

- “How it works” on page 15
- “CryptoService terminology” on page 15
- “CryptoService components” on page 15
- “Frequently Asked Questions” on page 16

## How it works

**Figure 4-21** How SSL works



SSL uses a cryptographic system with a pair of keys for each user (one pair for the server and one for the client). One key—a public key—is known to everyone; the second key—a private or secret key—is known only to the recipient of the message.

Communication between the client (browser or email client) and the server (station) begins with a handshake during which the sockets agree on the keys and other parameters. NiagaraAX uses the keys to encrypt and decrypt the shared information.

By convention, URLs that use an SSL connection start with “https:” instead of “http:”.

## CryptoService terminology

**Digital certificate** An electronic record that establishes a user’s credentials when doing business or other transactions over the internet. A digital certificate contains the user name, a serial number, expiration dates, copy of the certificate holder’s public key (used to encrypt messages and digital signatures), and the digital signature, which verifies to the recipient that the certificate is real.

The certificate may be signed by a Certification Authority (CA), or it may be self-signed as the default Tridium certificate is. A self-signed certificate allows for communication to be encrypted, but the server cannot be validated.

**Handshake** The initial exchange of records between a client and server that establishes the encryption keys to be used during the communication session.

**Hx profile** A customized Workbench user interface that uses Hx technology (HTML). For more information, see “About Px target media” in the *NiagaraAX User Guide*.

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

**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.

**Sockets** The end points in the connection between a the client and server.

## CryptoService components

- CryptoService provides a central location to process SSL-encrypted messaging.
- SslProvider provides SSL configuration.



## Frequently Asked Questions

**Q: What does the crypto.jar file do?**

A: The CryptoService module (crypto.jar) enables Secure Socket Layer (SSL) encryption between a NiagaraAX client (browser using the https protocol or email client) and server (remote station).

**Q: Does the CryptoService support Transport Layer Security (TLS) and/or Simple Authentication and Security Level (SASL)?**

A: It supports only SSL.

**Q: Is a license to use CryptoService required in the US?**

A: A license is required to encrypt WebService communications using ssl regardless of your geographical location.

**Q: Can I distribute the crypto.jar file to customers outside the US?**

A: You may not distribute the crypto.jar to any country on the US “banned countries” list. These include Cuba, Iran, North Korea, Sudan, and Syria. Others may be added by the US government at any time and it is up to you to adhere to all applicable US Export laws.

**Q: How do I set up SSL for email?**

A: When setting up incoming and outgoing email accounts, you enable SSL by setting the **Use Ssl** property to ‘true’. For more information, see “email-IncomingAccount” and “email-OutgoingAccount” in the *NiagaraAX User Guide*.

**Q: How do I configure a station WebService for encryption?**

A: Open the platform and station; drag and drop the CryptoService module from the palette onto the station’s Services folder in the Nav tree; then, enable https communications on the WebService property sheet. For more information, see “[Configure CryptoService](#)” on page 3.

**Q: Where can I find the crypto.jar?**

A: The crypto.jar is in the modules directory. If your version of NiagaraAX predates version 3.5.25.2, the crypto.jar may be a stub file, which will need to be overwritten with the crypto.jar file attached to the license email. Within Workbench, the CryptoService is located in the Palette.

**Q: When an applet is running in a secure browser are the applet communications secure?**

A: No. CryptoService supports secure https communications indicated by the “s” at the end of “http” and the visible lock symbol next to the URL. Secondary connections that use Wb profiles are not secure. To ensure secure communications, configure your application for Hx profiles such that the Workbench applet will not be used.

**Q: Is it possible to configure the CryptoService for EmailService only?**

A: To use for secure email only, do not make any changes to the WebService properties. You would use this configuration if you are more concerned to secure inbound than outbound communications.

**Q: Can I use a certificate with private and public keys that was created by third-party utility other than Oracle’s Keytool?**

A: Yes, You can create and sign your own certificate chain using any appropriate utility. See “[Import a private key to a TKS Key Store](#)” on page 9 for how to prepare and import the file to the Key Store.

**Q: Can I be my own Certificate Authority? Can I use CryptoService to sign certificates?**

A: No. CryptoService cannot be used to create or sign certificates. This Engineering Note explains how to use Oracle’s Keytool and OpenSSL to manage the TKS Key Store and create certificates. You may use the NiagaraAX 3.7 (or later) SSL Toolkit to create and sign certificates for use with NiagaraAX 3.6 and earlier versions.

## Troubleshooting

- **I get the message, “Cannot Connect” when attempting to access the station from the browser.**  
Make sure that:
  - You’re using the correct IP address.
  - The JACE license has been updated.
  - CryptoService is installed in the Services node of the station Nav tree.
  - WebService is correctly configured (**Https Enabled** is set to true)

- **My browser pops up the following certificate error.**

**Figure 4-22** Certificate error



The Tridium certificate, which is valid through 2054, is self-signed and the browser cannot validate its authenticity with a trusted root certificate. Some browsers will let you add the certificate so that you don't see this message again. Other browsers will not, and you will see it every time.

- **My station is unable to start the HTTPS service.**  
The Key and Trust Store properties may be different. For CryptoService, the same file (default name: ssl.tks) is used for both the Key and Trust Stores. This includes the password defined for the Trust Store, which must be the same as that defined for the Key Store. If you are creating your own server certificates, make sure the Trust Store properties match their analogs for the key store. See [“Configure the CryptoService”](#) on page 4.
- **I get the following message when I try to start the HTTPS web service.**

**Figure 4-23** Padding error message

```
Caused by: java.security.UnrecoverableKeyException: Given final block not properly padded
at com.tridium.crypto.Tks$KeyEntry.decodeKey(Tks.java:682)
at com.tridium.crypto.Tks.engineGetKey(Tks.java:203)
```

The private key password in the Key Store has a different password from the Key Store itself. These passwords must be identical. To set this password using the Keytool utility, see [“Import, verify and edit the Key Store”](#) on page 11.

## Document change log

Updates (changes/additions) to this *NiagaraAX CryptoService (SSL)—Engineering Notes* document are listed below.

- July 19, 2013; made these changes:
  - Moved the conceptual information to follow the procedures.
  - Rewrote the introductory paragraphs to the Engineering Note.
  - Rewrote [“Prerequisites”](#) on page 2.
  - Rewrote SSL property definitions following [Figure 4-3](#).
  - Cropped [Figure 4-4](#).
  - Explained the security difference between the Hx and Wb profiles. (Discussion follows [Figure 4-5](#)).
  - Moved the discussion about how to secure email to follow the explanation of how to install certificates.
  - Replaced “Create your own certificate” with [“Install a signed certificate”](#) on page 6.
  - Rewrote the introduction to [“Keytool options”](#) on page 8.
  - Added a new section, [“Import a private key to a TKS Key Store”](#) on page 9.
  - Added two FAQ entries about creating and signing certificates. See [“Frequently Asked Questions”](#) on page 16
  - Added two symptoms and solutions to [“Troubleshooting”](#) on page 16.
- Updated: May 24, 2013  
Added a Note at the beginning of the document that explains that starting in AX-3.7, a different (and improved) SSL/TLS configuration is available for most NiagaraAX host platforms.
- Draft: April 19, 2011  
Initial document.

