Preparing for Automic Automation v21

Creating and Using TLS/SSL Self-Signed Certificates

Version 1.3

Please note that this document is meant as a tutorial to explain TLS/SSL related concepts and how this impacts Automic Automation. This document should not be used as replacement for the product documentation.

Table of Contents

Chapter 1: Introduction

A **KeyStore** is a repository of security certificates – either authorization certificates or public key certificates – plus corresponding private keys used for TLS/SSL encryption.

With the advent of Automic Automation v21, TLS/SSL is the only communication between Automation Engines, Automic Web Interface (AWI), and TLS/SSL Agents (Windows, UNIX, and Java-based Agents). As such, the implementation of certificates for secure communications is mandatory.

This guide is relevant for customers who do not already use certificates signed by an internal CA or a commercial provider. It guides you through creating and deploying self-signed certificates for Automic Automation and details the steps required to set up self-signed certificates using the Open Source utility KeyStore Explorer. For more information, see https://keystore-explorer.org/.

For testing purposes or applications that are not Internet-facing, it is possible to use self-signed certificates to secure connections between the Automic Automation server and other components. There are several free tools to generate keystores and certificates, such as Java keytool, openssl or the KeyStore Explorer.

In this guide, KeyStore Explorer version 5.4.4 was used on a Windows 10 system to create the required files. Please be aware that based on your system and version, the User Interface of your tool might look slightly different than in this guide.

Chapter 2: Creating a New Keystore and Certificate with KeyStore Explorer

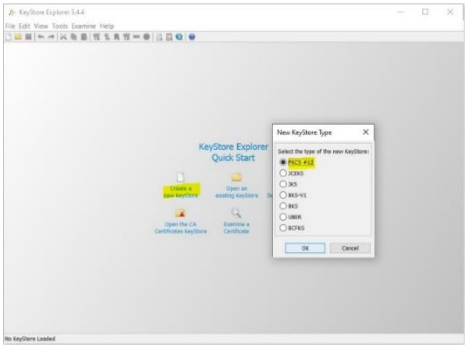
2.1 Creating the Keystore

The Automation Engine requires a KeyStore to store and process TLS/SSL requests from its components, such as the Automic Web Interface (AWI), TLS/SSL Agents, and so on.

Upon creating the KeyStore, you have to store the KeyStore file in a secure location and protect it with a password as it contains the private keys necessary for TLS/SSL authentication.

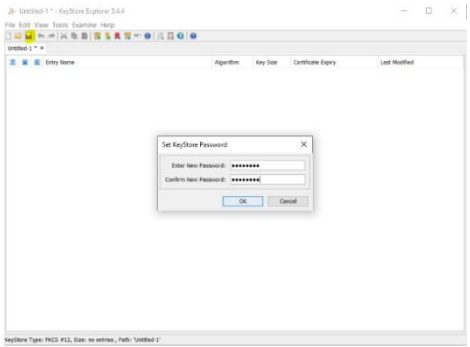
NOTE Automic Automation requires the Keystore type PKCS #12.

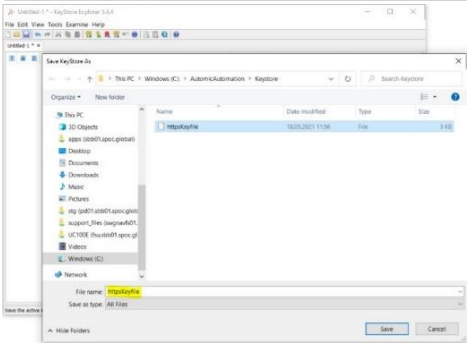
2.1.1 Defining a PKCS#12 Keystore



1. Open the KeyStore Explorer application.
2. Select **Create a new KeyStore**.
3. In the **New KeyStore Type** dialog, select **PKCS#12**.
4. Click **OK.**

2.1.2 Saving the Keystore File in a Secure Location





1. In the tool bar of the KeyStore Explorer, click the **Save** icon.
2. In the **Set a KeyStore Password** dialog, define a secure password. The default password is `changeit`. It is highly recommend using a stronger password.
3. Click **OK**.

The following example shows the default entries in the INI file of the Automation Engine (ucsrv.ini):

```
[TLS]
;
; keystore: Path and file where the keystore for TLS
certificate is stored
;
keystore=.\httpsKeyfile
;
; keystorePassword: Password of the keystore File
;
keystorePassword=changeit
```

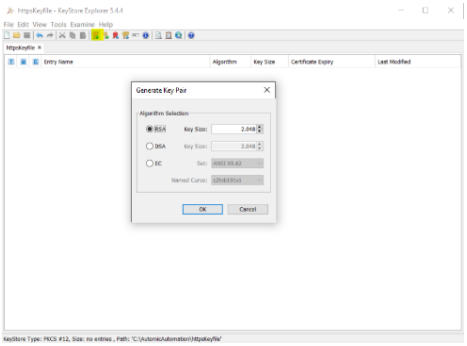
2.2 Generating a Key Pair and a Self-Signed Certificate in the Keystore

Once you have created the KeyStore and stored the KeyStore file you can generate a key pair within the KeyStore file.

You have to define at least the type, expiry and common names. However, if you operate a high availability environment or have JCPs running on different servers you also have to define subject alternative names.

2.2.1 Generating the Key Pair

To use authentication during the TLS/SSL connection with the client (agent) a private-public key-pair and a (public key) certificate are required to confirm the server's identity. The clients can use the self-signed certificate containing the server hostname and the public key to secure a connection to the intended Automic server.



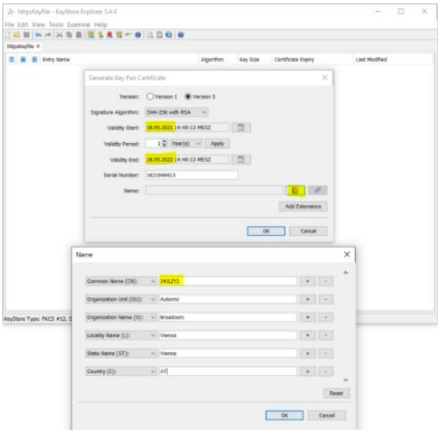
NOTE It is recommended to use RSA keys with a size of at least 2048 bits.

To generate the key pair follow these steps:

1. In the tool bar of the KeyStore Explorer, click the **Generate Key-Pair** icon.
2. In the **Generating Key Pair** dialog, select **RSA** and set the **Key Size** to 2048.
3. Click **OK**.

2.2.2 Defining Validity Periods and Certificate Details

Set the validity period for the certificate.

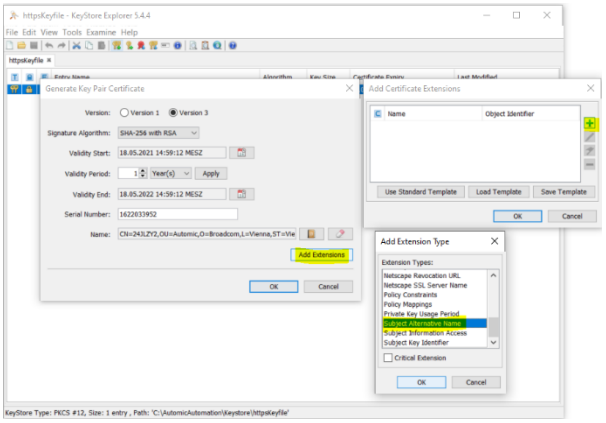


1. Once you have generated the key pair, the **Generate Key-Pair Certificate** dialog is displayed.
2. Define the certificate's validity (start and end).
3. Click the icon next to the **Name** field and define the name and other details.
4. Click **OK**.

NOTE The Common Name (CN) must match the hostname/domain/address that the agents will use to connect to the server.

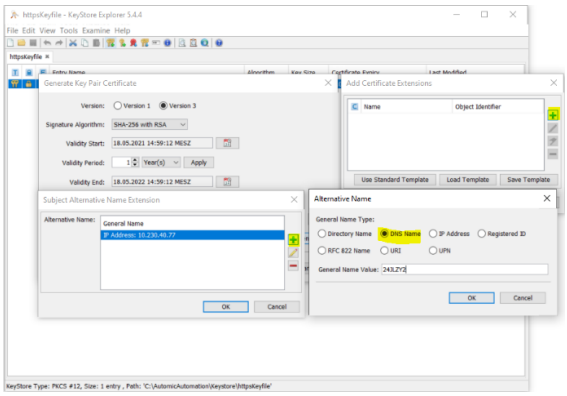
2.2.3 High Availability, Multiple Network Cards or JCPs on Multiple Servers

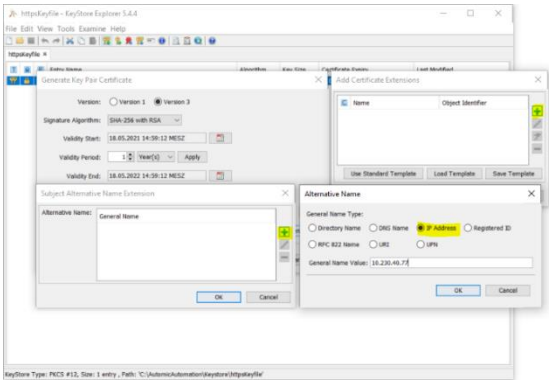
If you have multiple network cards on servers, different internal and external addresses, or numerous JCPs across different servers you have to define each IP Address or Server Name in the certificate. You can do so by adding Subject Alternative Names into the certificate.



NOTE If you define a Subject Alternative Name, you must repeat the Common Name in the Subject Alternative Names list because the Common Name is ignored when Subject Alternative Names have been defined.

1. In the **Generate Key-Pair Certificate** dialog, select **Add Extensions**.
2. In the **Add Certificate Extensions** dialog, select **Add**.
3. In the **Add Extension Type** dialog, select **Subject Alternative Name** and click **OK**.
4. In the **Subject Alternative Name Extension** dialog, click the **Add (+)** button.
5. In the **Alternative Name** dialog, select either the **DSN Name** or **IP Address** and define them accordingly.
6. Repeat the last step until you have added all IP Addresses or DSN names relevant for your environment.



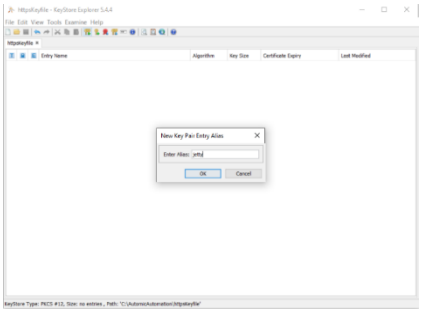


2.2.4 Setting Alias and Password for the Key Pair

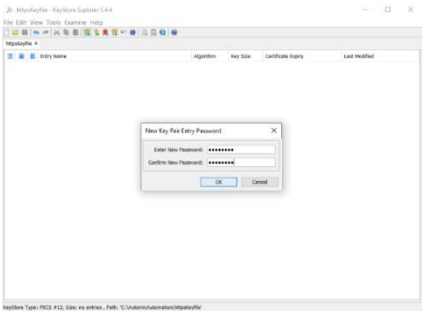
As a last step, you have to set an alias and password for the generated key pair. This definition has to match the definition in the INI file of the Automation Engine (ucsrv.ini).

The following example shows the default entries in the INI file of the Automation Engine (ucsrv.ini):

```
[TLS]
…
; keyPassword: Password for the Keys protection
;
keyPassword=changeit
;
; keyAlias: The name which the key is identified with.
;
keyAlias=jetty
```



1. Once you have defined the certificate extensions, the **New Key Pair Entry Alias** dialog is displayed.
2. Define the alias and click **OK**.
3. In the **New Key Pair Entry Password** dialog, set the password to the same value you are going to use in the Automic configuration (INI) files.
4. Click **OK**.



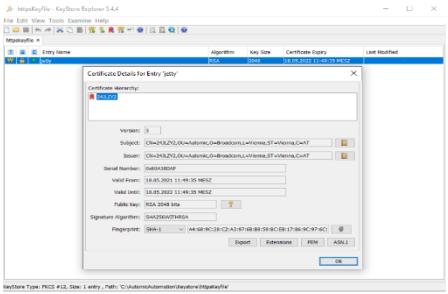
The key pair has been added to the KeyStore. You can now see the configured certificate details.

ATTENTION Do not forget to save the Keystore before closing the KeyStore Explorer.

2.2.5 Exporting the Certificate for Later Deployment

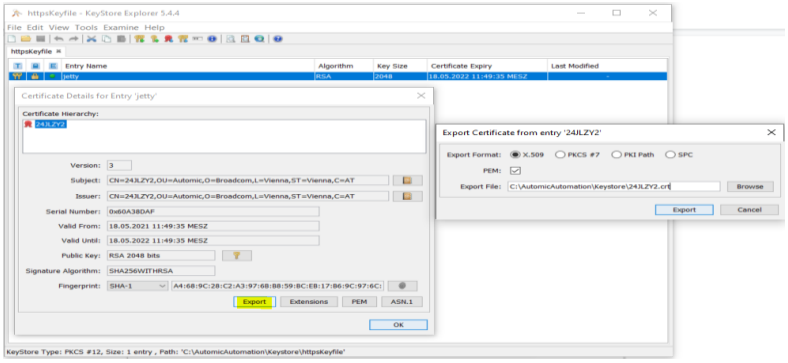
There are two mechanisms to export the self-signed certificate created with KeyStore Explorer. Use either one to get a copy of the certificate to deploy to the server hosting the Automic Web Interface (AWI).

2.2.5.1 Exporting the Certificate with the KeyStore Explorer



1. In the KeyStore Explorer, open the Automic server KeyStore. and the key pair entry.
2. Access the relevant key pair entry and select **Export**.
3. Export the self-signed certificate in PEM format and click **Export.**

The Agents and the Automic Web Interface (AWI) use this certificate to authenticate to the Automic Engine.



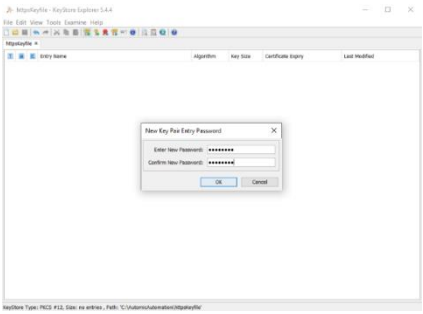
2.2.5.2 Downloading the Certificate from the Browser

If you prefer, you can download the certificate from a browser once you have deployed and tested the connection on the Automation Engine (see chapter 3.4.1).

2.3 Alternative: Creating the KeyStore with Java KeyTool Utility

Alternatively, you can use the Java KeyTool utility from the command-line to create the KeyStore and generate the key pair.

```
keytool -genkeypair -alias jetty -keyalg RSA -storetype PKCS12 -keypass changeit -
storepass changeit -keystore httpsKeyfile -validity 365 -
keysize 2048 -dname "cn=24JLZY2, ou=Automic, o=Broadcom,
l=Vienna, c=AT"
```



Chapter 3: Deploying the TLS/SSL KeyStore and Configuring the Automation Engine

3.1 Creating Encrypted Passwords for KeyStore Configuration Settings

As a system administrator, you define the password for KeyStore connections in the **[TLS]** section of the INI files of the Automation Engine.

For security reasons, make sure that the password is encoded. To do so, you can use UCYBCRYP.EXE, which is located in the **Tools > encrypt** folder of your installation directory.

NOTE UCYBCRYP.EXE requires the C++ 2010 Redistributable Package.

Use the following parameters to enter the program via the command line:

```
UCYBCRYP[.EXE] -p -n Password
```

NOTE The password length is limited to 32 characters. The PASSWORD.UCC file, which contains the encoded password, is created in the same directory. You can copy it to the relevant INI file.

Example

```
ucybcryp -p -n uc4
```

ATTENTION An encrypted password starts with two leading hyphens. If the content of the PASSWORD.UCC file is output with the command TYPE in Windows, two exclamation marks are displayed instead of the leading hyphens; therefore, make sure to copy the password from the file.

3.2 Configuring the AE INI File (ucrsv.ini) for the Automation Engine Server

If you used the default values for the KeyStore and key parameters, no changes in the AE INI file (ucsrv.ini) are required.

The final TLS/SSL configuration (using encrypted passwords) looks like this:

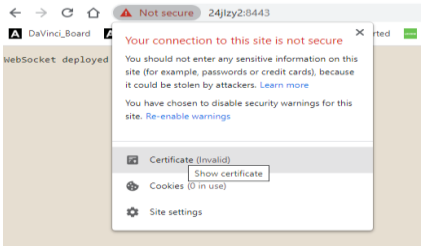
```
[TLS]
;
; keystore: Path and file where the keystore for TLS certificate is stored
;
keystore=.\httpsKeyfile
;
; keystorePassword: Password of the keystore File
;
keystorePassword=¬¬103B02A4E9656774333BD58DBFE5857D33
;
; keyPassword: Password for the Keys protection
;
keyPassword=¬¬103B02A4E965677433B1B5F761168102C7;
; keyAlias: The name which the key is identified with.
;
keyAlias=jetty
```

3.3 Deploying the KeyStore File to the Target Server

If you created the KeyStore on a different server you will have to copy the KeyStore file to the file identified in the AE INI file (ucsrv.ini).

3.4 Starting the Automation Engine and Testing Connections

Once all processes have started, test that the JCP endpoint is reachable.

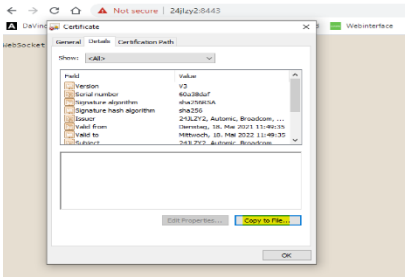


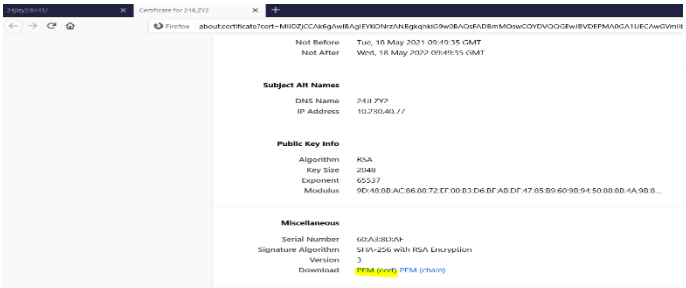
1. In a browser, enter the server name and port number defined in the AE INI file (ucsrv.ini).
2. Make sure you check all hostnames and IP addresses configured in the certificates.
   https://24JLZY2:8443/
   https://198.51.100.2:8443/
3. By default, the browser does not trust a self-signed certificate and displays a warning. You can add an exception to the browser.
4. Open the certificate from the top left corner and check that it matches the one you previously created.

3.4.1 Optional: Downloading the Certificate from the Browser

If you did not create the certificate from the KeyStore Explorer earlier, you can do it at this point.

1. Open the certificate from the top left corner.
2. Export it to a file in DER format from Chrome or download it as PEM from Firefox.
   The certificate could have the host's name to identify quickly which server it belongs to, for example, 24JLZY2.crt.





Chapter 4: Configuring the Automic Web Interface (AWI) for Self-Signed Certificates

The certificate has to be accessible to the AWI WebServer. There are two ways you can do this:

- Import the server certificate in Java certificates truststore
- Set the trustedCertFolder parameter in the AWI configuration file

4.1 Importing the Server Certificate in Java Certificates Truststore (cacerts)

Import the certificate into the Java certificates truststore using the Java supplied tool (KeyTool).

```
<path-to-java-keytool>\keytool -import -alias jetty
-keystore <path-to-java-cacerts>\cacerts
-file <path-to-server-certificate>\24JLZY2.crt -storepass changeit
```

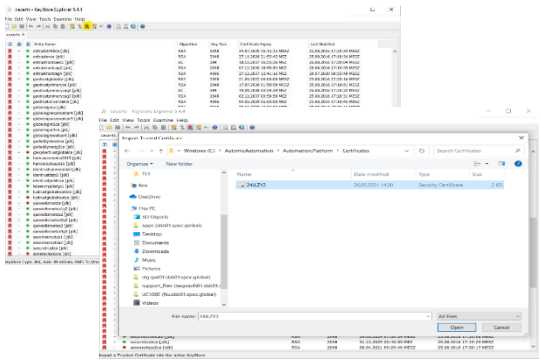
The following example shows how the path to the Java truststore (cacerts) could look like on Windows:

```
C:\Program Files (x86)\Java\jre1.8.0_281\lib\security
```

4.1.1 Multiple Java Installations

If you have multiple Java installations and do not know which one is used by the AWI WebServer, you can import the server certificate for all the Java truststores.

You can also use the KeyStore Explorer as an administrator and import the server certificate into the cacerts keystore.



4.2 Setting the trustedCertFolder Parameter in the AWI Configuration File (uc4config.xml)

Use the following example as a guide to set the trustedCertFolder parameter:

```
<?xml version="1.0" encoding="iso-8859-15"?>
<configuration>
   <logging count="10"/>
   <trace count="10" xml="0"/>
   <connections trustedCertFolder="C:\AutomicAutomation\Certs">
         <connection name="AUTOMIC" system="AUTOMIC">
               <cp ip="24JLZY2" port="8443"/>
         </connection>
   </connections>
</configuration>
```

4.3 Restarting the Automic Web Interface (AWI)

You can now restart the WebServer. Make sure that you can connect to the Automic Web Interface (AWI).

Chapter 5: Configuring the TLS/SSL Agents with Self-Signed Certificates

5.1 Configuring TLS/SSL Agents to use the Certificates

You can configure the path to the valid self-signed certificate in the INI file of every TLS/SSL Agent (Windows, UNIX, and Java-based Agents).

To do so, define the path in the **trustedCertFolder=** parameter of the **[AUTHORIZATION]** section of the relevant Agent INI file. For example:

```
[AUTHORIZATION]
trustedCertFolder = C:\AutomicAutomation\Certs
```

You can then deploy the exported certificates to the directory you define here.

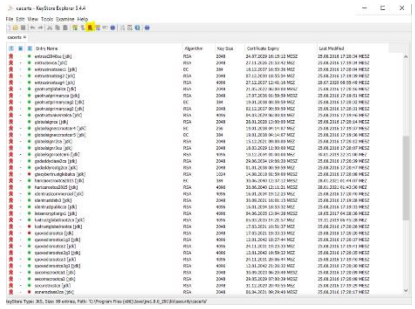
5.2 Using Java TrustStore / Windows Store for Certificates

Instead of maintaining certificates in the Automic directory structure, you can import the certificate into the Java truststore or Windows store. In this case, you can skip configuring the **trustedCertFolder** parameter as described above.

5.2.1 Importing Certificates for Java-Based Agents

Import the server certificate in the Java certificates truststore (cacerts)

```
<path-to-java-keytool>\keytool -import -alias jetty
-keystore <path-to-java-cacerts>\cacerts
-file <path-to-server-certificate>\24JLZY2.crt -storepass
changeit
```



The following example shows how the path to the Java truststore (cacerts) could look like on Windows

```
C:\Program Files (x86)\Java\jre1.8.0_281\lib\security
```

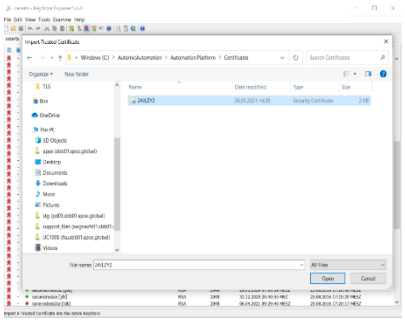
If you have multiple Java installations and do not know which one is used by the Agent, you can import the server certificate for all the Java truststores.

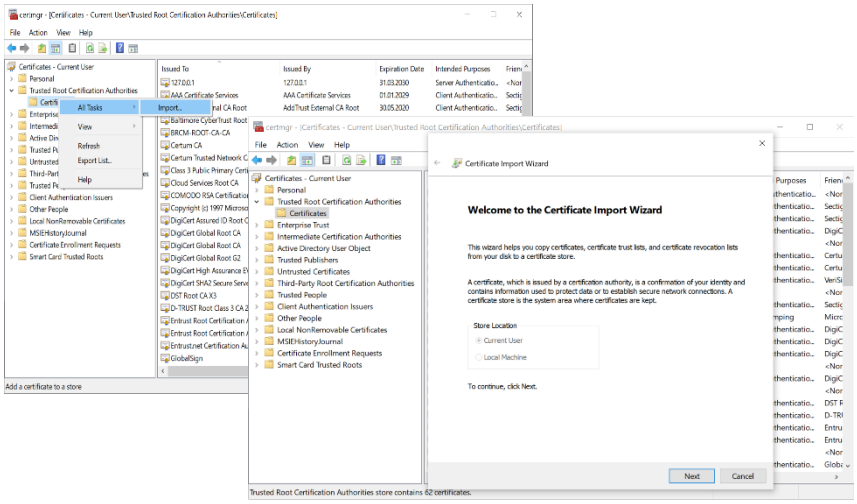
You can also use the KeyStore Explorer as an administrator and import the server certificate into the cacerts keystore.

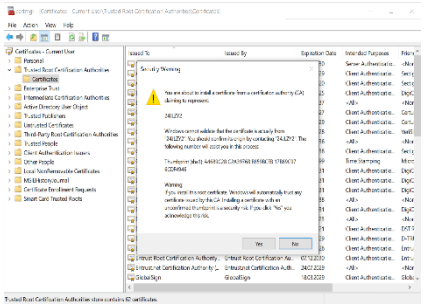
5.2.2 Importing Certificates for Windows Agents

To import the certificates for Windows Agents do the following:

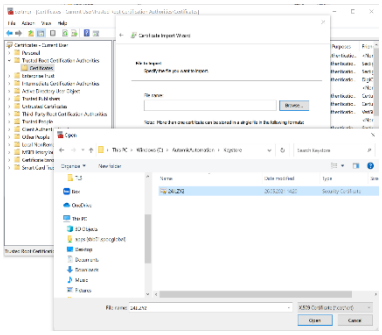
1. Open the Windows certificate truststore.
2. Use the Certificate Import Wizard to import the server certificate.



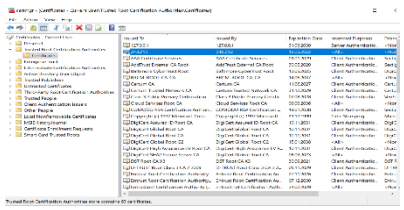




Because the certificate is self-signed and does not have the signature of a trusted CA, a warning is displayed.



3. Once you have accepted the risk, the certificate is added to the Windows trusted certificates and authorized by the Windows agents running.




BROADCOM®