Preparing for Automic Automation v21

Creating and Using TLS/SSL Certificates Signed by an Internal CA

Version 1.4

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.

October 25, 2022

Contents

Chapter 1: Introduction

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.

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 intend to use certificates signed by an internal Certificate Authority (CA). It guides you through creating and deploying Internal CA-signed certificates for Automic Automation and details the steps required to set up Internal CA-signed certificates using the OpenSSL. For more information, see https://www.openssl.org/.

For testing purposes or applications that are not Internet-facing, you can use Internal CA-signed certificates to secure connections between the Automic Automation server and other components.

In this guide, we used OpenSSL version 1.1.1 on a Linux Ubuntu system to create the required files. Please be aware that based on your system and version, the commands of your tool might look slightly different than in this guide.

We mention an alternative way to view certificate within this guide using the Open Source utility KeyStore Explorer. For more information, see https://keystore-explorer.org/.

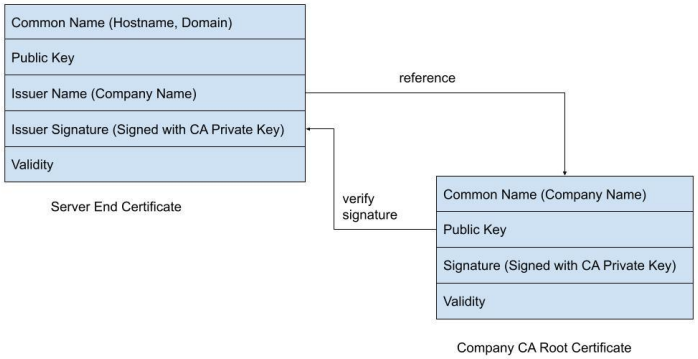
.

Chapter 2: Create a New Internal Certificate Authority (CA) and Certificate with OpenSSL

NOTE If your company already has an Internal CA used to create end certificates for other applications you can you can contact the responsible department and skip the rest of this chapter.

An Internal Certificate Authority can be used to sign certificates to enable TLS/SSL connections.

While multiple layers of CAs can be created (by adding intermediate certificates and forming a so-called certificate chain), in this guide, we directly use the Internal CA root certificate to sign the end certificate of the Automic server.



2.1 Create the Internal CA

A private key and self-signed root certificate for an Internal CA can be created with OpenSSL and include information relevant for the CA, like the company or department name. The private key needs to be stored in a secure location and should be password protected.

```
$ openssl genrsa -aes256 -out automicCA.key 2048
```

When creating the root certificate, you are asked for the private key password (passphrase) to continue. The validity period of the CA root certificate can be longer since the end certificate(s) used for the server can have a shorter lifespan.

```
$ openssl req -x509 -new -key automicCA.key -sha256 -days 700 -out automicCA.crt
```

You can view the information in the certificate with this command:

```
$ openssl x509 -noout -text -in automicCA.crt
```

2.2 Create and sign certificates using the Internal CA

To sign other certificates with this CA a Certificate Signing Request (CSR) is required. Create a private key for the Automic server. It is recommended to use RSA keys with a size of at least 2048 bits.

The default password used by Automic is `changeit`, but we recommend using a stronger one.

```
$ openssl genrsa -aes256 -out MyServer.key 2048
```

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

Every IP Address and server name used in Automic needs to exist in the certificate. So, multiple network cards, different internal and external addresses, or numerous JCPs across different servers all require Subject Alternative Names (SAN) to be defined.

Remember: If you define Subject Alternative Name, you must repeat the Common Name in the Subject Alternative Names list. This is because the Common Name will be ignored when Subject Alternative Names are present.

To create a CSR you need to specify the Common Name (CN) for the server certificate and this must match the hostname/domain/address that the agents will use to connect to the server.

You will be prompted for the password of the private key that you set in the previous step.

Create a config file containing the required details as in this example that also includes Subject Alternative Names:

```
automicssl.cnf:

[ req ]
default_bits           = 2048
default_md             = sha512
default_keyfile        = MyServer.key
prompt                 = no
distinguished_name     = req_distinguished_name
req_extensions         = v3_req

[ req_distinguished_name ]
countryName            = AT
stateOrProvinceName    = Vienna
localityName           = Vienna
organizationName       = Broadcom
organizationalUnitName = Automic
commonName             = MyServer

[ v3_req ]
basicConstraints       = CA:FALSE
keyUsage               = digitalSignature, keyEncipherment
subjectAltName         = @alt_names

[alt_names]
DNS.1                  = MyServer
IP.1                   = 198.51.100.2
```

In this instance, openssl requires the config file to be able to create the CSR:

```
$ openssl req -new -key MyServer.key -out MyServer.csr -config automicssl.cnf
```

You can view the content of the CSR request using this command:

```
$ openssl req -text -noout -verify -in MyServer.csr
```

Finally, you can create the server certificate based on the CSR and sign it with the internal CA. As SANs were defined previously, you need to explicitly add this extension to include them in the end certificate.

```
$ openssl x509 -req -CA automicCA.crt -CAkey automicCA.key \
-CAcreateserial -extensions v3_req -extfile automicssl.cnf \
-in MyServer.csr -out MyServer.crt -days 365
```

The end certificate that Automic Automation will later use contains the information required for the TLS/SSL connections. Please be aware that the CA certificate used to sign it is not included in the same file.

```
$ openssl x509 -noout -text -in MyServer.crt
```

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

3.1 Encrypting 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.

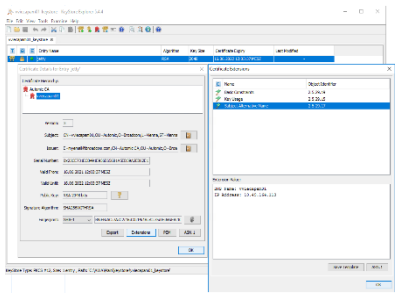
3.2 Configuring the AE INI File (ucrsv.ini) for the Automation Engine Server

The previously created server certificate needs to be converted into the PKCS#12 format for the Automic server (Automation Engine) to be able to use it. The alias of the entry in the keystore is set via the name parameter and the default used by the Automic Server is `jetty`.

You will need to enter the passphrase for the key when prompted (this was set in a previous step), but also set one for the keystore (for example `changeit`).

```
$ openssl pkcs12 -export -out MyServer_keystore -inkey
MyServer.key -in MyServer.crt -name jetty -certfile automicCA.crt
```

NOTE The keystore in PKCS#12 format can also be opened with the KeyStore Explorer and the information relevant for the server certificate, including the signing Internal CA and SANs, can be viewed via the UI.



NOTE You can also use the KeyStore Explorer to export the Automic CA root certificate, since it was included during the keystore generation. This was used to sign the Server end certificate and needs to be trusted by all Clients that will connect via TLS/SSL.

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
```

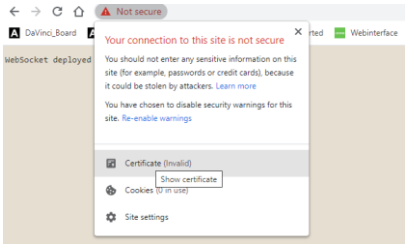
3.3 Deploying the KeyStore File to the Target Server

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.
3. https://MyServer:8443/
   https://198.51.100.2:8443/
4. The Internal CA root certificate is not yet trusted by the Browser, since it was not yet included in the Trusted Root CAs.
5. 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, MyServer.crt.

NOTE The Internal CA root certificate is not yet trusted by the Browser, since it was not yet included in the Trusted Root CAs

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

Unlike with self-signed certificates, where the server certificate is required, for Internal CA-signed, it is possible to establish TLS/SSL connections between AWI and the Automic Server by only ensuring that the signing Certificate Authority is trusted.

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

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

4.1 Importing the 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-ca-certificate>\automicCA.crt -storepass changeit
```

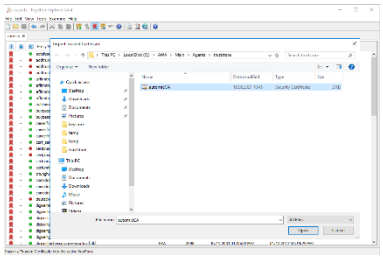
The following example shows how the path to the Java truststore (cacerts) could look like on Windows:

```
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 certificate for all the Java truststores.

You can also use the KeyStore Explorer as an administrator and import the 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="MyServer" 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 TLS/SSL Agents with Internal CA-Signed Certificates

There are 2 options to use Internal CA-signed certificates for secure connections between the Automic Server and the agents:

- Copy and configure all agents to use the CA root certificate
- Install the CA root certificate to use the already available Java or OS truststores

5.1 Configuring TLS/SSL Agents to use the Certificates

You can configure the path to the valid CA-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.

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 Using Java TrustStore / OS Stores for Certificates

5.2.1 Importing Certificates for Java-Based Agents

Import the 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>\automicCA.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
```

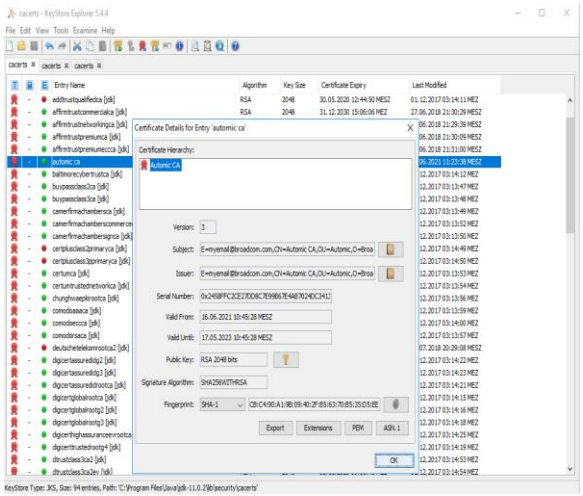
If you have multiple Java installations and do not know which one is used by the Agent, you can import the certificate for all the Java truststores.

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 certificate for all the Java truststores.

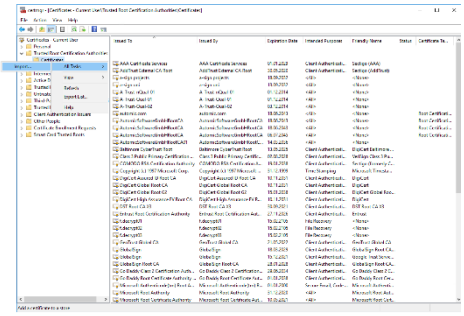
You can also use the KeyStore Explorer as an administrator and import the certificate into the cacerts keystore.

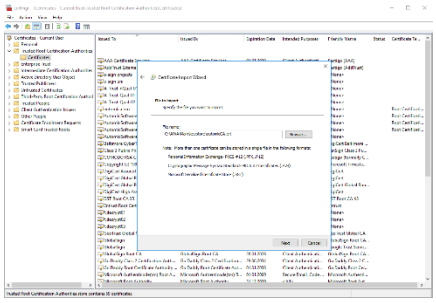


5.2.2 Importing Certificates for Windows Agents

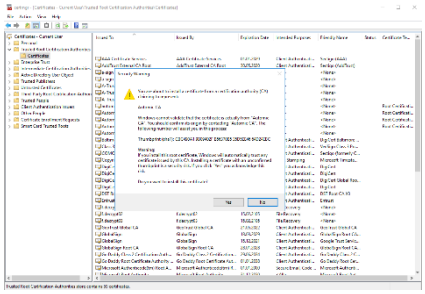
To import the certificates for Windows Agents do the following:

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

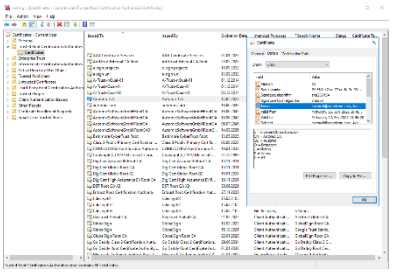




Because the certificate is self-signed and does not have the signature of a trusted CA, a warning is displayed.



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



NOTE If your agents were started via Service Manager, restarting is required for the imported certificates to be trusted by the agents.

5.2.3 Import Certificates for Unix Agents

To import the certificates for Unix Agents do the following:

1. Copy the Internal CA root certificate into the folder where the local user certificates are located

```
$ sudo cp automicCA.crt /usr/local/share/ca-certificates/
```

2. Update the SSL CA certificates truststore so that it adds a new .pem file containing the certificate

```
$ sudo update-ca-certificates

Updating certificates in /etc/ssl/certs...
1 added, 0 removed; done.
Running hooks in /etc/ca-certificates/update.d...

Adding debian:automicCA.pem
done.
```

You can then check that the Internal CA root certificate was imported into the OpenSSL truststore.

```
$ ls /etc/ssl/certs/ | grep automic
automicCA.pem
```

NOTE In the previous example, the location of the SSL truststore is the default one on Ubuntu 4.15.0-142-generic. Depending on your distribution and version, this might be different or the certificates might require another format.

When the agent is started, if the SSLCertDir were not set in the ini file, it will use the certificates from the default paths if they exist `/etc/ssl/certs`, `/etc/ssl/certs/ca-certificates.crt`, `/etc/pki/tls/certs`, `/etc/pki/tls/certs/ca-bundle.crt`.

If openssl is installed on the system, another option to determine the location of the trusted certificates folder is using the command below:

```
$ openssl version -d
```

This points to the default OPENSSLDIR where the CA certificates are installed.

But openssl is not necessarily installed on all hosts, since the Unix agent includes the libraries it needs for the TLS/SSL connections. In this case, you can search your filesystem for widely used commercial CAs like DigiCert or DST/ISRG.

```
$ sudo find /etc/ssl/ -name '*DigiCert*'
$ sudo grep -R "DigiCert" /etc/ssl/
$ sudo find /etc/ssl/ -name '*DST*'
$ sudo find /etc/ssl/ -name '*ISRG*'
```


BROADCOM®