Connecting to Azure IoT Hub with Certificate Based Authentication

The certificate-based authentication for Azure IoT works like this:

- A root certificate used to sign device certificates will be created and added into Azure IoTHub
- For each IoT device we want to authenticate, we will create a separate client certificate, signed by the same certificate authority.
- At the time of the authentication, our IoT device will present the client certificate as the proof of identity.
- Azure IoT Hub will verify the identity based on the root certificate and the device name.

This document details how to create the certificate authority, a root certificate, and a device certificate along with adding the required certificates to both the Azure IoT Hub and Azure Injector configurations.

> You will also need the CA certificate that signed the SSL certificate being used on the Azure IoT Hub server to include in the Azure Injector configuration

> The command line tools openssl and keytool are used.
>
> Install the OpenSSL command line tool and add the OpenSSL PATH in the Windows environment variables if necessary.
>
> Keytool is part of the standard java distribution and is located in the bin sub-directory of your jdk installation directory. Add the keytool PATH in the Windows environment variables if necessary.
>
> You will need to restart any open command window to pick up this configuration change.

As a first step, we need to generate the certificate hierarchy.

Create the following folder structure on your local drive to hold the various certificates in the hierarchy that we will be generating:

```
iotcerts/
 ca/
 certs/
     device/
```

These are the steps that need to be completed for the certificate hierarchy:

- Generate Root CA
- Generate Device certificate signed with the Root CA’s private key

Generate Root Certificate

1. Generate a private key file (ca.key) for the Root CA using the command below. You may choose to enter a passphrase to be associated with the ca.key file as well.

   > Make note of this passphrase if you set one for the Root CA private key file (ca.key) as it will be used multiple times.

   ```
   openssl genrsa -des3 -out ca/ca.key 4096
   ```

2. Generate a self-signed certificate (ca.crt) for the Root CA using the command below. This command generates a new self-signed X.509 certificate named "ca.crt" valid for 3650 days (10 years) using the RSA private key "ca.key". You will be required to enter the pass phrase associated with the private key file "ca.key".

   ```
   openssl req -new -x509 -key ca/ca.key -days 3650 -out ca/ca.crt
   ```

There are a number of fields associated with the creation of the certificate. Fill them out with your relevant details.

Example CA Creation

```
$ openssl req -new -x509 -key ca/ca.key -days 3650 -out ca/ca.crt
Enter pass phrase for ca/ca.key:
You are about to be asked to enter information that will be incorporated
into your certificate request.
What you are about to enter is what is called a Distinguished Name or a DN.
There are quite a few fields but you can leave some blank
For some fields there will be a default value,
If you enter '.', the field will be left blank.
-----
Country Name (2 letter code) [AU]:US
State or Province Name (full name) [Some-State]:KS
Locality Name (eg, city) []:Stilwell
Organization Name (eg, company) [Internet Widgits Pty Ltd]:Cirrus Link Solutions
Organizational Unit Name (eg, section) []:Support
Common Name (e.g. server FQDN or YOUR name) []:CLS Example Root CA
Email Address []:
$
```

3. Azure will need the certificates in PEM format. Convert the CRT format to PEM format using the command below.

```
openssl x509 -in ca/ca.crt -out ca/ca.pem -outform PEM
```

You should have the following files created:

```
iotcerts/
 ca/
    ca.crt
    ca.pem
    ca.key
```

Depending on the version of openSSL that you are using, you may see additional .srl files created which contain the signed certificate's unique serial number. These files are not used directly by the modules and not included in the certificate hierarchy displayed above.

Generate Device Certificate

1. Generate private key in PSCK8 format (CertDevice.key) for the device using the command below.

```
openssl genrsa -out certs/device/CertDevice.key 4096
```

2. Generate a Certificate Signing Request (CSR) for the device using the command below. This command generates a new CSR named "CertDevice.csr' using the RSA private key "CertDevice.key".

```
openssl req -new -key certs/device/CertDevice.key -out certs/device/CertDevice.csr
```

There are a number of fields associated with the creation of the certificate. Fill them out with your relevant details.

Example Device CSR Creation

```
$ openssl req -new -key certs/device/CertDevice.key -out certs/device/CertDevice.csr
You are about to be asked to enter information that will be incorporated
into your certificate request.
What you are about to enter is what is called a Distinguished Name or a DN.
There are quite a few fields but you can leave some blank
For some fields there will be a default value,
If you enter '.', the field will be left blank.
-----
Country Name (2 letter code) [AU]:US
State or Province Name (full name) [Some-State]:KS
Locality Name (eg, city) []:Stilwell
Organization Name (eg, company) [Internet Widgits Pty Ltd]:Cirrus Link Solutions
Organizational Unit Name (eg, section) []:Support
Common Name (e.g. server FQDN or YOUR name) []:Device01
Email Address []:

Please enter the following 'extra' attributes
to be sent with your certificate request
A challenge password []:
An optional company name []: $
```

The Common Name or FQDN must match the name of the logical Device Id in IoT Hub. In our example the Device Id is Device01

3. Sign the Device CSR with the Device CA using the command below. This command will sign the CSR "CertDevice.csr" with the Root CA certificate ‘ca.crt’ and Root CA's RSA private key ‘ca.key’, creating a new X.509 certificate named ‘CertDevice.crt’ valid for 365 days (1 year). You will be required to enter the passphrase associated with the private key file "ca.key".

```
openssl x509 -req -in certs/device/CertDevice.csr -CA ca/ca.crt -CAkey ca/ca.key -CAcreateserial -out
certs/device/CertDevice.crt -days 365
```

4. Azure will need the certificate in PEM format. Convert the CRT format to PEM format using the command below:

```
openssl x509 -in certs/device/CertDevice.crt -out certs/device/CertDevice.pem -outform PEM
```

A unique client certificate will be required for each logical Device Id.

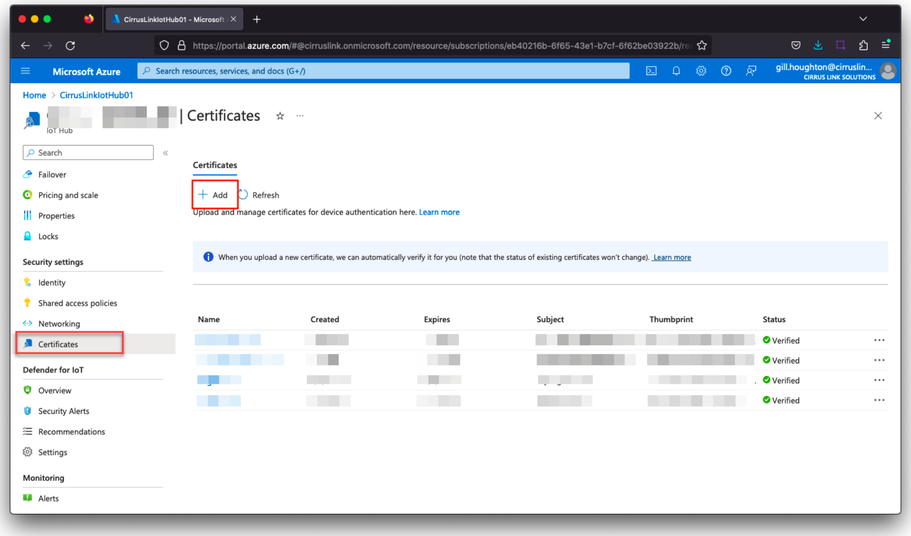
If you have multiple devices, you will need to expand your folder structure adding an appropriate folder for each device and run the Generate Client Certificate steps for each device amending the commands above appropriately.

You should now have the following files created:

```
iotcerts/
 ca/
    ca.crt
    ca.pem
    ca.key
 certs/
     device/
        CertDevice.crt
        CertDevice.pem
        CertDevice.csr
        CertDevice.key
```

Add Root Certificate to IoT Hub

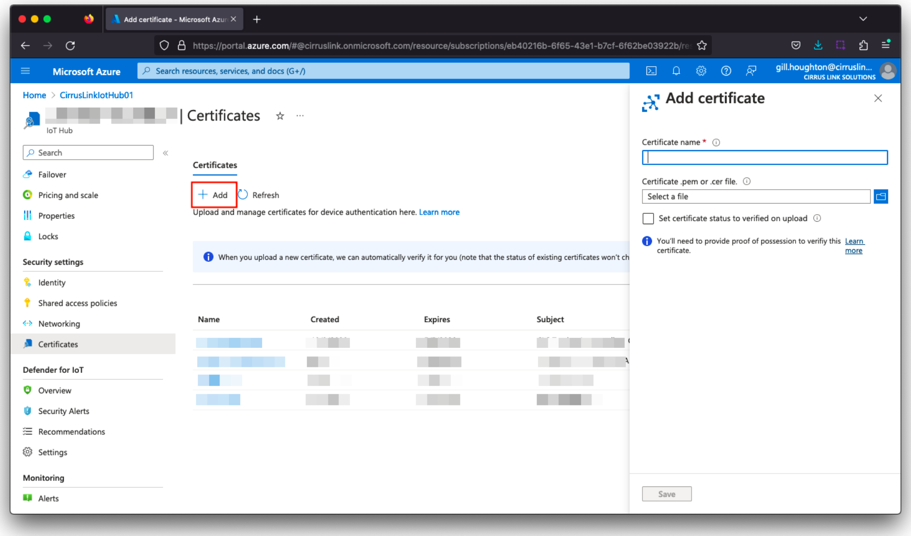
- On the IoT Hub resource Overview page, click “Certificates” menu on the left blade, and click the “Add” button.



Upload and manage certificates for device authentication here. Learn more

When you upload a new certificate, we can automatically verify it for you (note that the status of existing certificates won't change). Learn more

- Give a certificate name (eg. MyOrg RootCertificate) and import the *ca.pem* file from your *iotcerts/ca* folder. Check the “Set certificate status to verified on upload” checkbox and click Save.

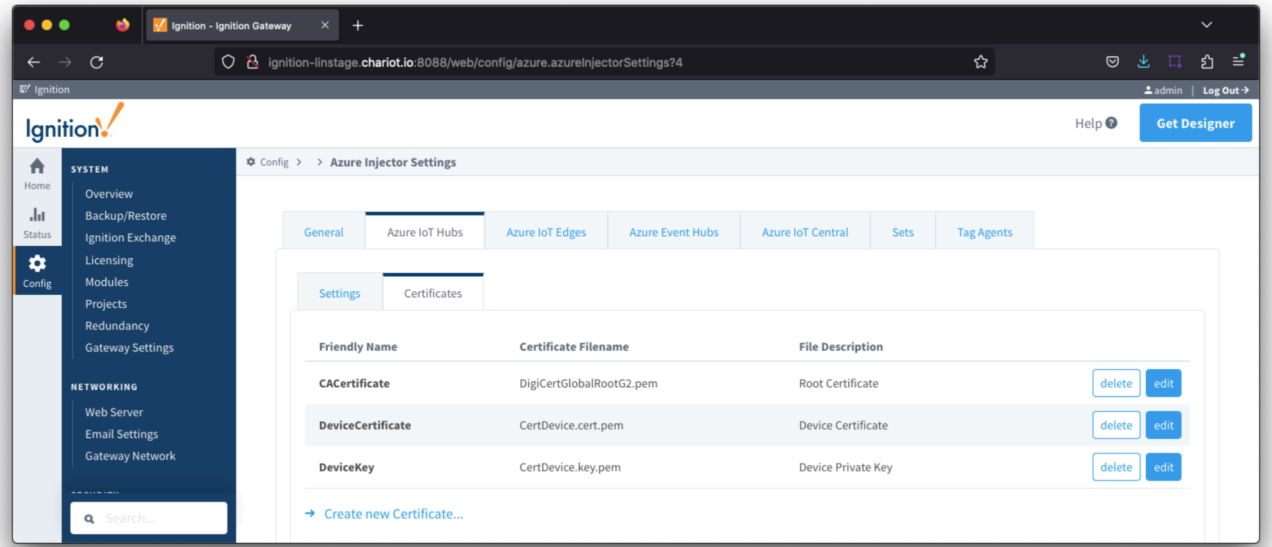


You'll need to provide proof of possession to verify this certificate. Learn more

Add device certificates to Azure Injector IoT Hub configuration

Navigate to the Azure Injector > Settings > Azure IoT Hubs > Certificates and add the certificates as shown below:

Friendly Name	Certificate Filename	File Description	File Location
CACertificate	The CA Certificate that signed the SSL cert being used on the IoTHub server		
DeviceCertificate	CertDevice.pem	Device Certificate	iotcerts/certs/device/CertDevice.pem
DeviceKey	CertDevice.key	Device Private Key	iotcerts/certs/device/CertDevice.key


ignition-linstage.chariot.io:8088/web/config/azure.azureInjectorSettings?4

Config > Azure Injector Settings

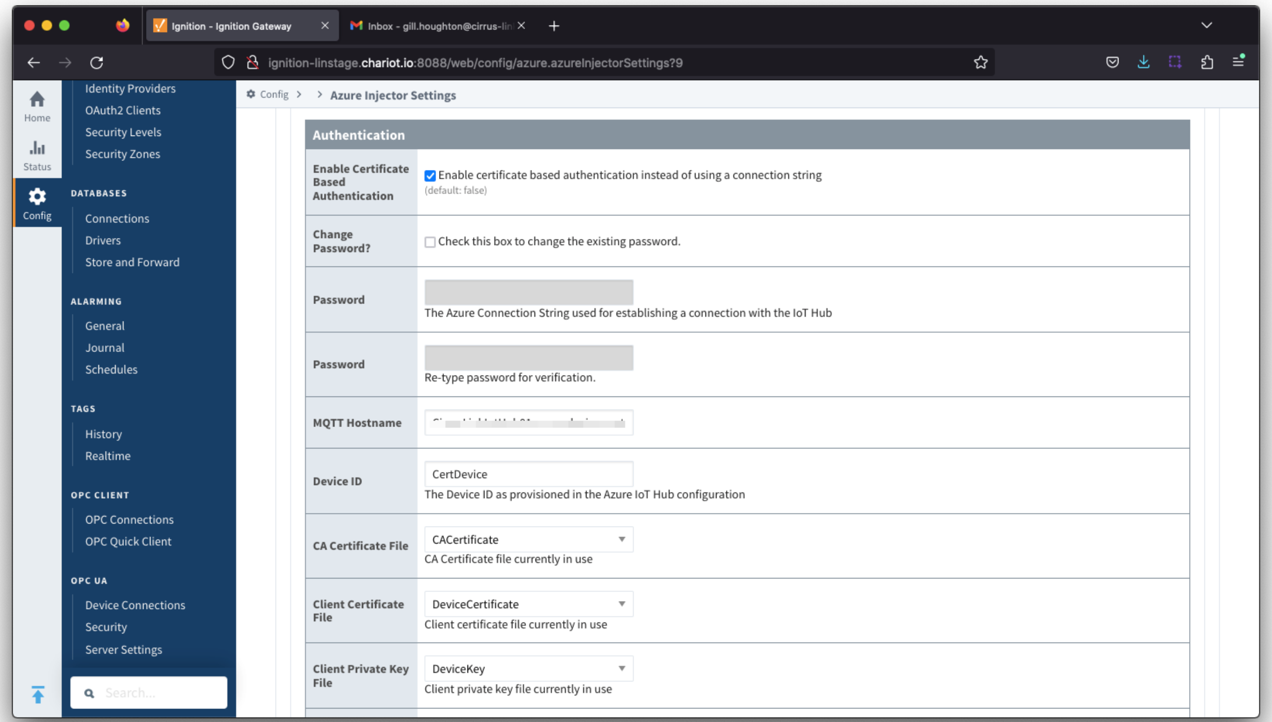
General | Azure IoT Hubs | Azure IoT Edges | Azure Event Hubs | Azure IoT Central | Sets | Tag Agents

Settings | Certificates

Friendly Name	Certificate Filename	File Description	
CACertificate	DigiCertGlobalRootG2.pem	Root Certificate	delete edit
DeviceCertificate	CertDevice.cert.pem	Device Certificate	delete edit
DeviceKey	CertDevice.key.pem	Device Private Key	delete edit

→ Create new Certificate...

Update the Azure Injector > Settings > Azure IoT Hubs > Settings configuration to use the certificates as shown below. Note : the certificates created do not use a Password

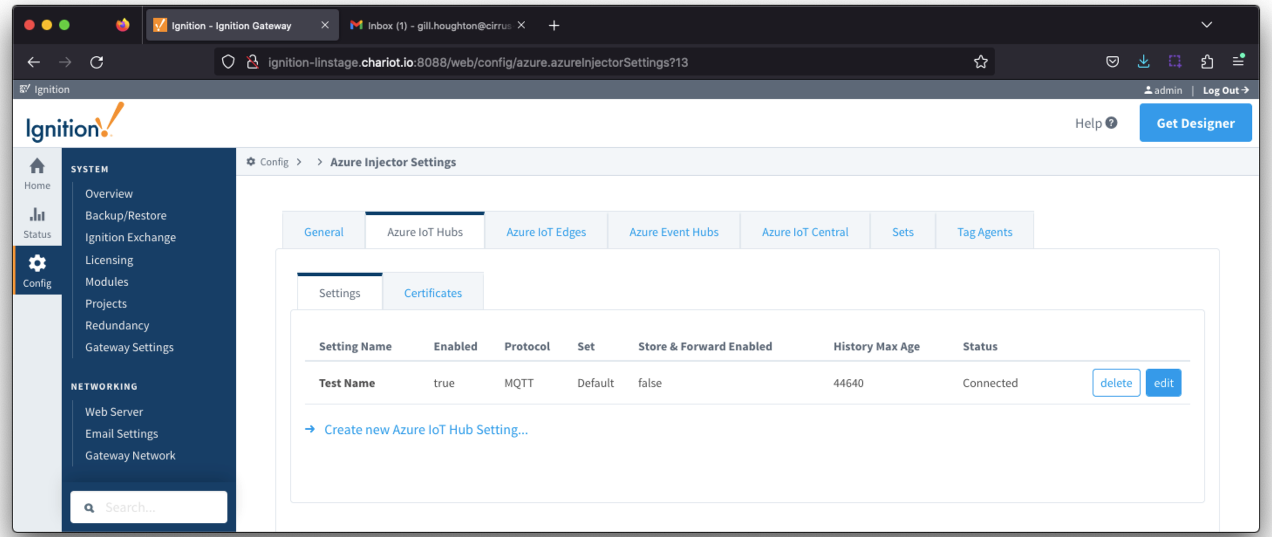

ignition-linstage.chariot.io:8088/web/config/azure.azureInjectorSettings?9

Config > Azure Injector Settings

Authentication

Enable Certificate Based Authentication	☑ Enable certificate based authentication instead of using a connection string (default: false)
Change Password?	☐ Check this box to change the existing password.
Password	The Azure Connection String used for establishing a connection with the IoT Hub
Password	Re-type password for verification.
MQTT Hostname	[redacted]
Device ID	CertDevice The Device ID as provisioned in the Azure IoT Hub configuration
CA Certificate File	CACertificate CA Certificate file currently in use
Client Certificate File	DeviceCertificate Client certificate file currently in use
Client Private Key File	DeviceKey Client private key file currently in use

Verify the connection is established as shown by the Status on the Azure Injector > Settings > Azure IoT Hubs view



Setting Name	Enabled	Protocol	Set	Store & Forward Enabled	History Max Age	Status
Test Name	true	MQTT	Default	false	44640	Connected