



SARA-R5 series

Configuration on Azure IoT Hub and DPS

Application note



Abstract

This document provides examples of how to use AT commands to connect the Azure IoT service with u-blox SARA-R5 series modules.

Document information

Title	SARA-R5 series	
Subtitle	Configuration on Azure IoT Hub and DPS	
Document type	Application note	
Document number	UBX-21002509	
Revision and date	R01	25-Oct-2021
Disclosure restriction	C1-Public	

This document applies to the following products:

Product name
SARA-R5 series

u-blox or third parties may hold intellectual property rights in the products, names, logos and designs included in this document. Copying, reproduction, modification or disclosure to third parties of this document or any part thereof is only permitted with the express written permission of u-blox.

The information contained herein is provided "as is" and u-blox assumes no liability for its use. No warranty, either express or implied, is given, including but not limited to, with respect to the accuracy, correctness, reliability and fitness for a particular purpose of the information. This document may be revised by u-blox at any time without notice. For the most recent documents, visit www.u-blox.com.

Copyright © u-blox AG.

Contents

Document information	2
Contents	3
1 Introduction	4
1.1 Prerequisites	4
2 Configure device for DPS	5
2.1 Create root CA certificate	5
2.2 Get verification code.....	6
2.3 Require PoP certificate	8
2.4 Certificate verification	8
2.5 Configure enrollment group.....	9
2.6 Enable ZTP on u-blox Thingstream portal.....	10
2.7 ZTP data provisioning	12
2.8 Retrieve client certificate from module via AT command	12
2.9 Configure IoT device connection to DPS	13
3 Example of Azure IoT Hub connection	15
3.1 Cellular module configuration	15
3.1.1 Store certificates in the module file system	15
3.1.2 Check CA, CC, and PK in the file system.....	16
3.1.3 Certificates manager configuration	16
3.2 Azure IoT Hub connection via MQTT protocol	17
3.2.1 Module setup MQTT session	17
3.2.2 MQTT example: connection, subscribe, publish and read	17
4 Integration with Azure IoT Explorer	19
4.1 IoT Hub connection	19
4.2 View Devices.....	20
4.3 Interact with a device	20
Appendix	22
A Glossary	22
Related documentation	23
Revision history	23
Contact	24

1 Introduction

This document describes how to connect a u-blox SARA-R5 series module with the Azure IoT services.

SARA-R5 series can easily transfer data from field devices (meters, sensors, etc.) to cloud applications. This getting started guide provides step-by-step instruction on getting the device provisioned to Azure IoT Hub using Device Provisioning Service (DPS) and using IoT Explorer to interact with the device's capabilities.

1.1 Prerequisites

The user shall have the following items ready before beginning the process:

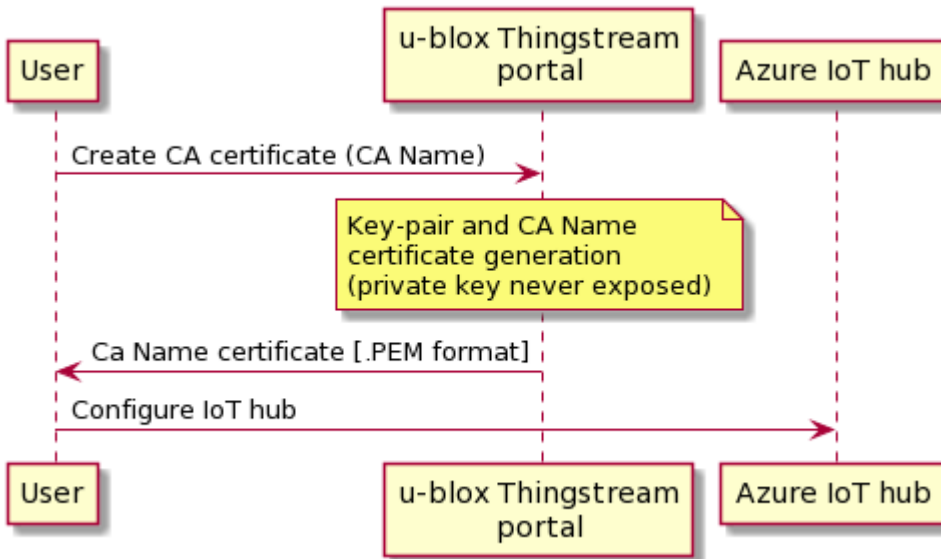
- SARA-R5 series cellular module (for more details, see the SARA-R5 series data sheet [\[1\]](#))
- u-blox Thingstream portal account
- Azure account
- Azure IoT Hub instance
- Azure IoT Hub device provisioning service

2 Configure device for DPS

To get started with the Azure IoT service, it is possible to use the IoT Hub device provisioning service (DPS) that enables zero-touch provisioning (ZTP) and configures the device connection to the cloud without requiring human intervention, allowing customers to configure multiple devices in a secure and scalable manner.

All the required steps are explained in detail in the following sections.

2.1 Create root CA certificate



The root certificate authority (CA) certificate in ".pem" format can be created and downloaded using the following POST request.

```
POST /ztp/rootca/create
```

The API call to create a root CA, which will be used by the server to generate device-specific certificates. Once generated, this certificate needs to be uploaded to the server platform of the client.

Every POST request to the u-blox portal is successful only if the CA name provided is globally unique and the URL header credentials are valid and authorized.

REQUEST BODY:

```
{
  "CAName": "azure_ztp_device_root_ca",
  "CACertificateValidity": 1825,
  "DeviceCertificateValidity": 1825
}
```

RESPONSE BODY:

```
{
  "CAName": "azure_ztp_device_root_ca",
  "CACertificateValidity": 1825,
  "DeviceCertificateValidity": 1825,
  "Certificate": "-----BEGIN CERTIFICATE
  << certificate data >>
  -----END CERTIFICATE-----",
  "CreatedDate": "2020-10-16T13:06:38Z"
}
```

In a different manner but with the same result, it is possible to collect the root CA directly from the u-blox Thingstream portal webpage, available at portal.thingstream.io.

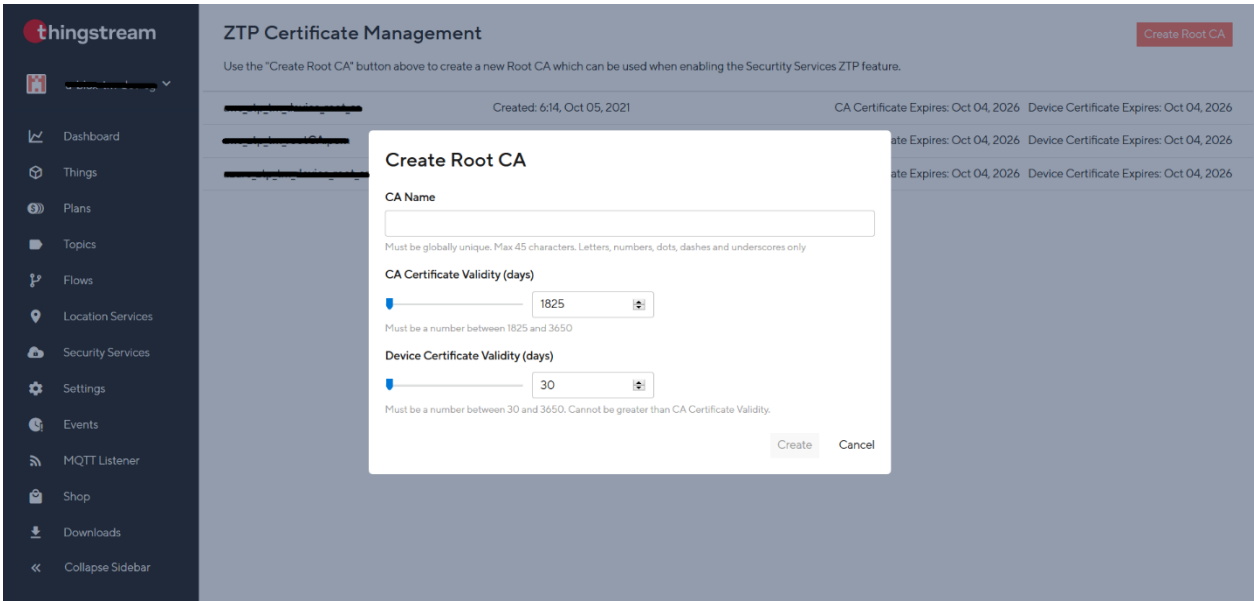


Figure 1: ZTP Certificate Management section in u-blox Thingstream portal

Clicking on root CA Name in the **ZTP Certificate Management** page, the CA certificate in ".pem" format can be created and downloaded.

Root CA ×

azure_ztp_device_root_ca 📄

Certificate	Download Certificate	Created
	Generate POP Certificate	6:16, Oct 05, 2021
CA Certificate Validity	1825 days	
	Expires Oct 04, 2026	
Device Certificate Validity	30 days	
	Expires Nov 04, 2021	

Figure 2: Root CA certificate management section in u-blox Thingstream portal

2.2 Get verification code

At this stage, the user gets a registration code from the Azure portal, which is sent to the u-blox portal and used as the common name (CN) of the certificate signing request (CSR) for the private key verification certificate.

The registration code can be retrieved from the Azure portal as shown below.

- From: Azure portal -> DPS -> **Certificates** (left-hand menu).
- Click **Add**.
- Enter a name for the certificate and select **Browse** to the ".pem" file.
- Click **Save** once the certificate is successfully uploaded.

Microsoft Azure Search resources, services, and docs (G+)

Home > All resources > DPS-SER >

Add Certificate

[Learn more about certificates.](#)

Certificate Name * ⓘ
azure-ztp-tm-root-ca ✓

Certificate .pem or .cer file. * ⓘ
"azure_ztp_tm_root_ca.pem" 📁

Save

Figure 3: Add CA certificate in Azure DPS

The certificate will be shown in the Azure Certificate Explorer list. Note that the **status** of this certificate is **Unverified** as shown in Figure 4.

+ Add Refresh

i You can use this tool to upload and manage your certificates.

Name	Status	Expiry
azure-ztp-tm-root-ca	Unverified	Wed Oct 15 2025 15:06:37 GMT+02

Figure 4: Example of unverified certificate

- By selecting and clicking on the last imported certificate, it is possible to view some details.
- In **Certificate Details**, the user needs to **Generate Verification Code**.
- The provisioning service creates a verification code required to validate the certificate proof-of-possession (PoP).

Verification Code ⓘ
[Copy]

Generate Verification Code

Verification Certificate .pem or .cer file. * ⓘ
Select a file 📁

Figure 5: Example of Generate Verification Code

2.3 Require PoP certificate

The validation code can be sent to the u-blox Thingstream Portal using the **Generate PoP Certificate** option (see [Figure 2](#)) or via the following POST request. The subsequent POST response will include the verification certificate in “.pem” format.

```
POST /ztp/rootca/popcertificate/generate
```

The API call to generate a Proof-of-Possession certificate. After uploading the Root CA to the intended server, a registration code is generated. User needs to use this registration code to generate a Proof-of-Possession certificate using this API, and then upload the generated certificate to prove ownership of the Root CA that was initially uploaded to the server.

Parameters used in the API call:

CAName [Mandatory] - Certificate name, which was generated earlier.

RegistrationCode [Mandatory] - Registration code generated by the server where Root CA was uploaded.

curl example:

```
curl -X POST "https://ssapi.services.u-blox.com/ztp/rootca/popcertificate/generate " -H "accept: application/json" -H "Authorization: [Authorization Parameter]" -H "Content-Type: application/json" -d "{ \"CAName\": \"[CAName Parameter]\" \"RegistrationCode\": \"[RegistrationCode Parameter]\"}"
```

When the validation code is sent through the u-blox Thingstream portal, the root CA certificate shall be downloaded again as shown in [Figure 2](#).

2.4 Certificate verification

The resulting signature as a verification certificate can be uploaded to the DPS in the Azure portal.

In **Certificate Details** on the portal, use the file explorer icon next to the **Verification Certificate** “.pem” or “.cer” file field to upload the signed verification certificate from the system.

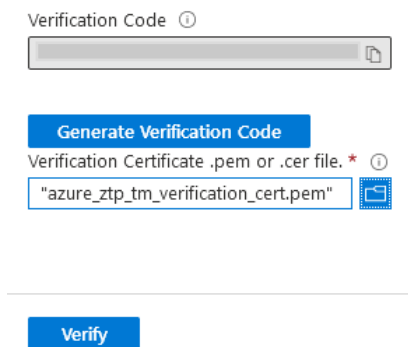


Figure 6: Upload of the signed verification certificate

Once the certificate is successfully uploaded, click **Verify**.

The **status** of the certificate changes to **Verified** in the Azure Certificate Explorer list (see the example shown in [Figure 7](#)).

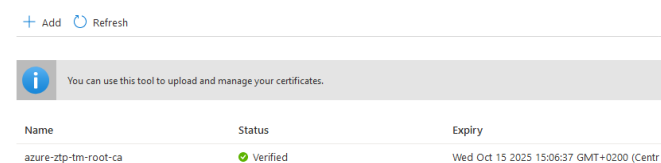
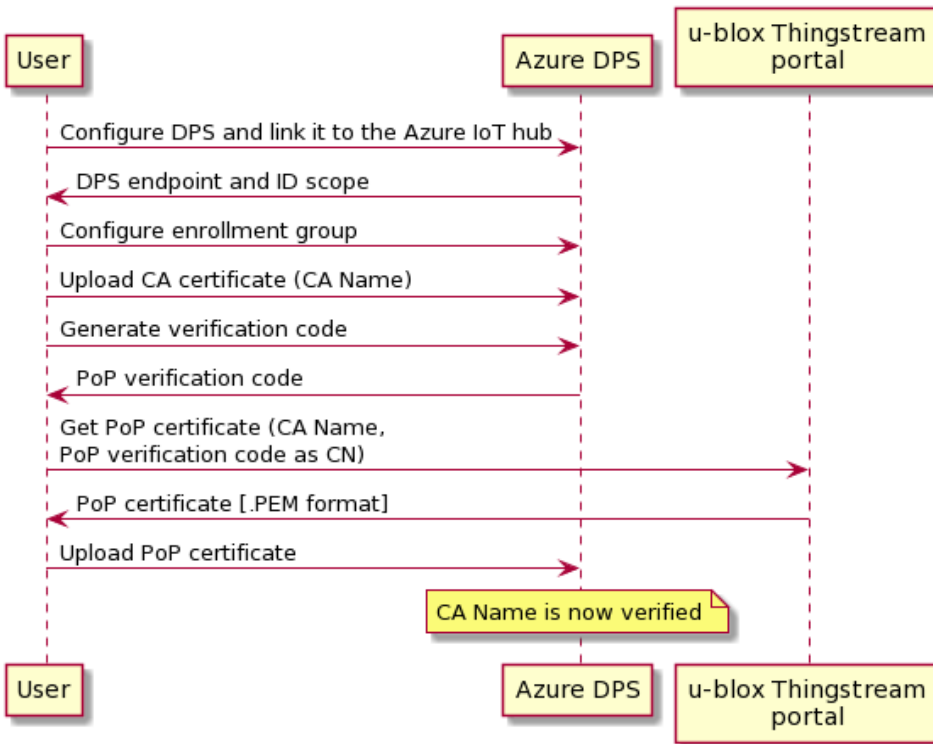


Figure 7: Example of verified certificate

The diagram below shows the overall procedure steps described in sections 2.2, 2.3, and 2.4.



2.5 Configure enrollment group

An enrollment group is an entry for a group of devices that share a common attestation mechanism of X.509 certificates, signed by the same signing certificate, which can be the root CA or the intermediate certificate, used to produce a device certificate on the physical device.

- Azure portal -> DPS -> **Manage enrollments** (left-hand menu).
- Click **Add enrollment group**.
- When the **"Add enrollment group"** panel appears, enter the information for
 - Group name
 - **Attestation Type** → "Certificate" option
 - **Certificate Type** → "CA Certificate" option
 - **Primary Certificate** → Select verified CA certificate from the list of existing certificates
 - Select policy
 - Select IoT hub
- Click **Save**.

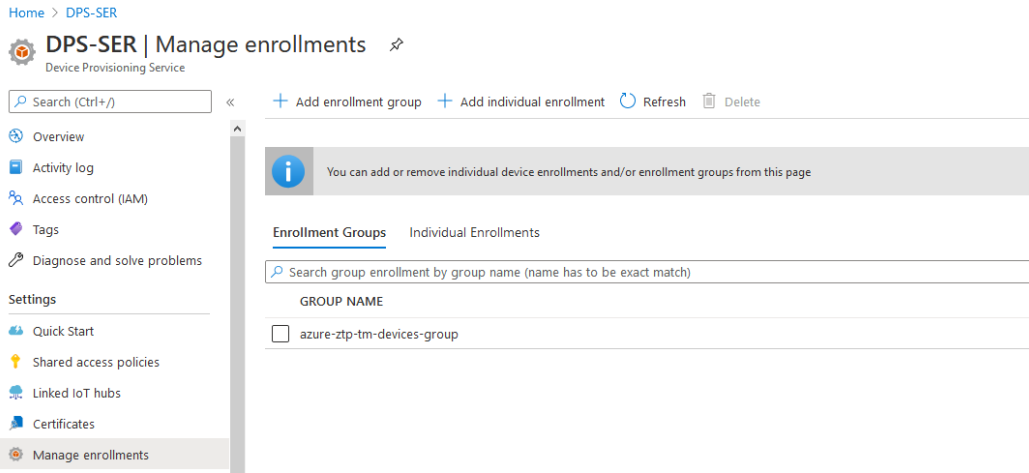


Figure 8: Overview of the DPS Manage enrollments page

2.6 Enable ZTP on u-blox Thingstream portal

Use the u-blox Thingstream portal and a specific set of API requests to set or update feature authorizations (ZTP, local encryption, C2C, C2CKeyPairing), and network parameters (security heartbeat) on new modules (i.e., devices that have not completed the bootstrap and claim of ownership procedures) and/or existing modules (i.e., they have already registered to security services and claimed by customer with his device profile UID). For more details, see u-blox Thingstream documentation page [5].

In the u-blox Thingstream portal, the user can define the list of authorized features for devices that have not yet completed a registration to u-blox security services (bootstrap and claim of ownership procedures) by logging into the **Security Services Device Profiles** section.

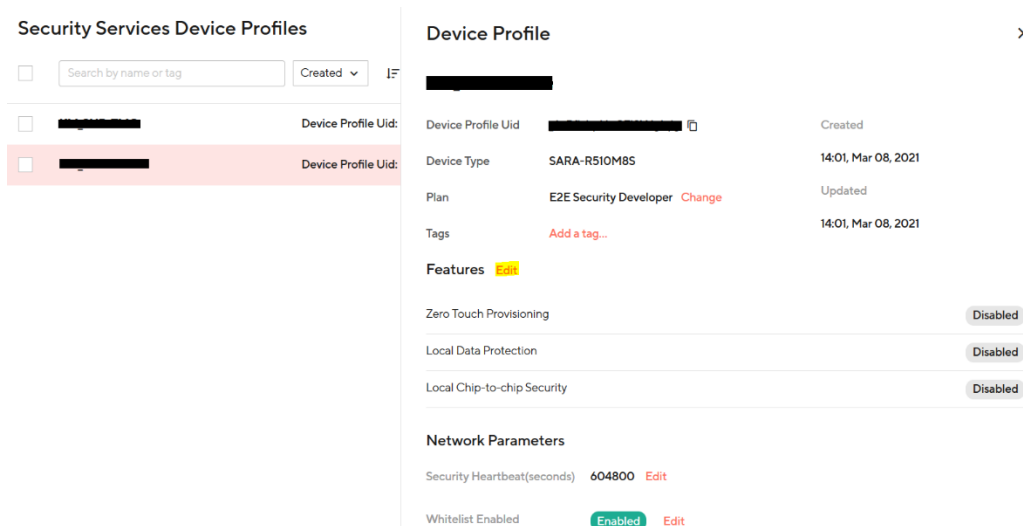


Figure 9: Device Profile management section in u-blox Thingstream portal

To manage “existing” devices, the user can access a list of registered devices (**Things**), select one of them, and edit the list of authorized features according to the user’s subscribed pooled plan(s).

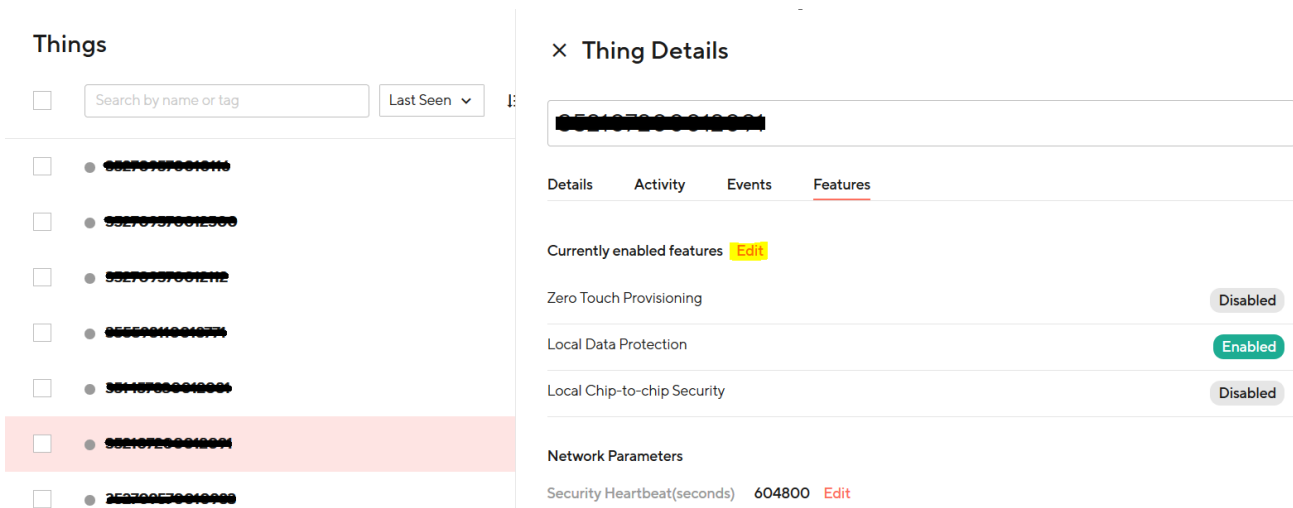


Figure 10: Device profile management section in u-blox Thingstream portal

As previously described, the security features with related parameters can be authorized and configured for one or more devices before registration to u-blox security services (not bootstrapped devices) or when they are already registered and claimed by the customer with the proper device profile UID (bootstrapped and claimed devices). Based on which set of devices the change will be applied, the following external customer REST APIs can be used:

- Provisioning “On bootstrap” (not bootstrapped devices)
 - POST /deviceprofile/bootstrapprovisioning/set
 - POST /deviceprofile/bootstrapprovisioning/get
 - POST /deviceprofile/bootstrapprovisioning/delete
- Provisioning “On existing” (bootstrapped and claimed devices)
 - POST /device/provisioning/set
 - POST /device/provisioning/get
 - POST /device/provisioning/delete

Using the following POST request of customer REST APIs, the user will be able to associate a CA Name to ZTP feature for a specific device profile UID.

```
POST /deviceprofile/bootstrapprovisioning/set
Set or update feature authorizations (ZTP, Local Encryption, C2C, C2CKeyPairing, ZTPV1), and network parameters (Security Heartbeat) on new SARA-R5 devices. Provisioning will happen when the device bootstraps. Enables or disables feature authorization and network parameters on the list of devices that belong to provided DeviceProfileUID in the request.
```

Parameters used in the API call:

- DeviceProfileUID [Mandatory] - DeviceProfileUID against which security features will be set.
- Provisioning [Mandatory] - The desired feature authorizations and/or network parameters to be provisioned on DeviceProfileUID.

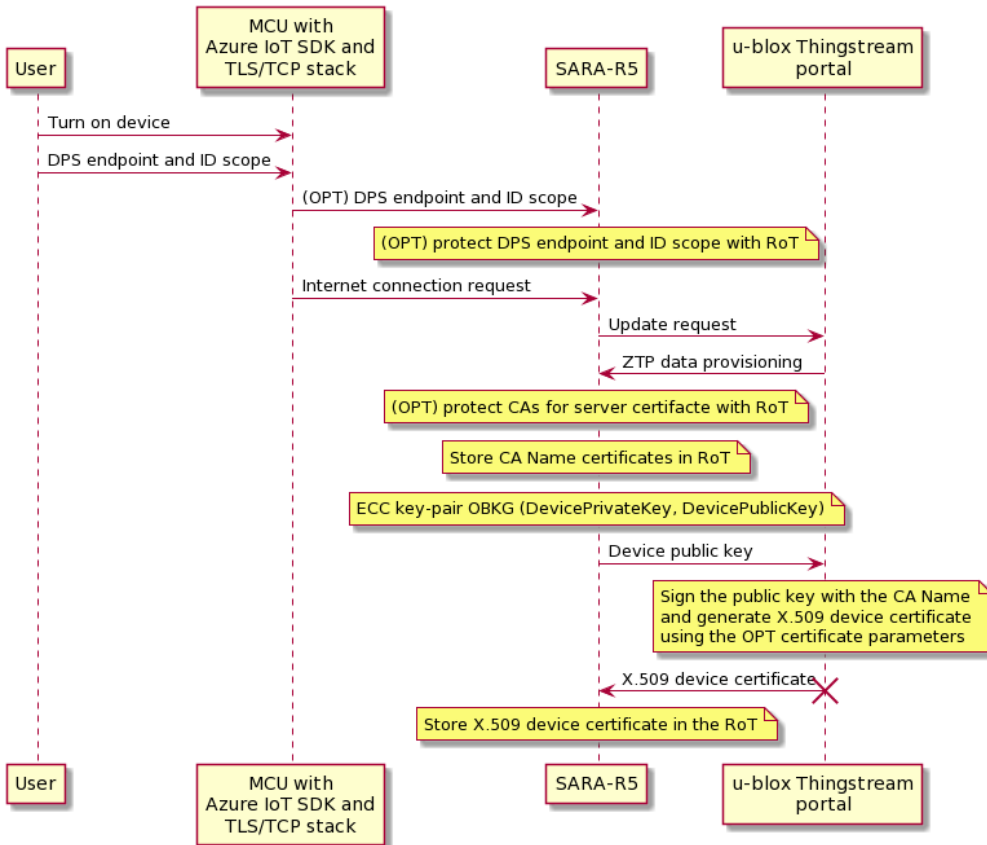
curl example:

```
curl -X POST "https://ssapi.services.u-blox.com/deviceprofile/bootstrapprovisioning/set" -H "accept: application/json" -H "Authorization: [Authorization Parameter]" -H "Content-Type: application/json" -d "{
  \"DeviceProfileUID\": \"[DeviceProfileUID Parameter]\"
  \"Provisioning\": {
    \"Authorization\": [ { \"[ Authorization Parameters]\" } ],
    \"NetworkParameters\": {
      \"[ Network Parameters]\" } } } }
```

For more details on the described steps and related topics, see the u-blox Thingstream documentation web page [5].

2.7 ZTP data provisioning

Once the previous steps are completed, the following zero-touch provisioning (ZTP) procedure will be straightforward. The user needs only to configure the proper device profile UID using the +USECDEVINFO AT command (if it is not already “claimed”), turn on the SARA-R5 series module, and make sure that it is able to access the internet to complete registration to u-blox security services (see u-blox Thingstream documentation [5]).



2.8 Retrieve client certificate from module via AT command

Users can retrieve the X.509 device certificate information needed to support a mutually authenticated SSL/TLS/DTLS session by using the u-blox IoT Dock and issuing the +USECDEVICERT AT command. This command retrieves the following certificates (in ".pem" format):

- device X.509 Private Key (PK);
- device X.509 Client Certificate (CC);
- X.509 root CA certificates.

Command	Response	Description
AT+USECDEVICERT?	+USECDEVICERT: 0,0,0 OK	Report that the device X.509 private key, certificate and root CA provisioning status (respectively in this order). 0 means certificates provisioned.
AT+USECDEVICERT=0	+USECDEVICERT: 0,"-----BEGIN EC PRIVATE KEY----- << certificate data >> -----END EC PRIVATE KEY-----" OK	Display and retrieve the client private key.

Command	Response	Description
AT+USECDEVCERT=1	+USECDEVCERT: 1, "-----BEGIN CERTIFICATE----- ----- << certificate data >> -----END CERTIFICATE -----" OK	Display and retrieve the client certificate.
AT+USECDEVCERT=2	+USECDEVCERT: 1, "-----BEGIN CERTIFICATE----- ----- << certificate data >> -----END CERTIFICATE -----" OK	Display and retrieve the CA certificate.

For more details on AT commands, see the SARA-R5 series AT commands manual [2].

2.9 Configure IoT device connection to DPS

The main parameters and information of **DPS** and **IoT Hub** are available on the Azure portal in the **“Overview”** section.

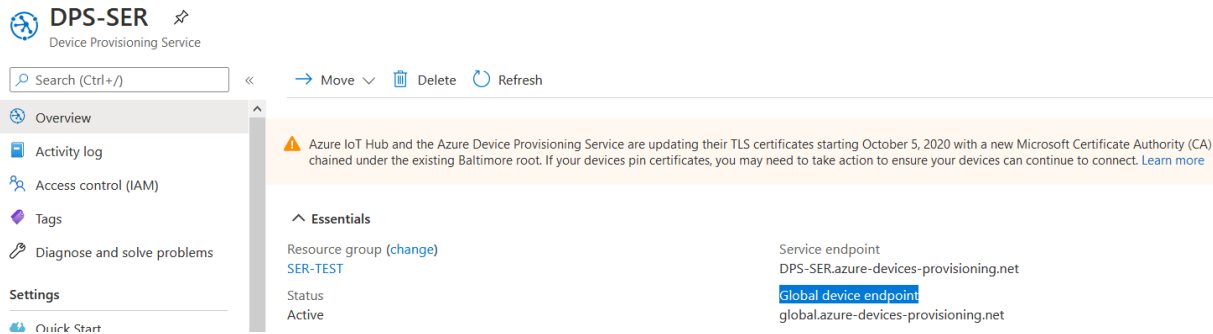


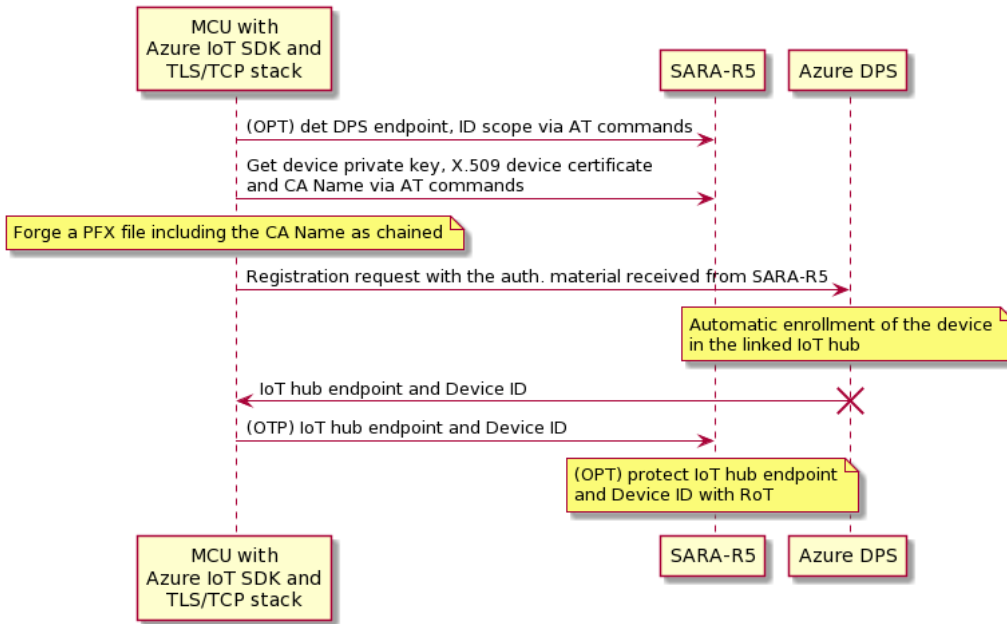
Figure 11: DPS Overview page

Use the **“azure-iot-device”** Python library to contact the DPS and trigger provisioning service. This SDK is provided by Azure at <https://github.com/Azure/azure-iot-sdk-python>.

As a final step, a Python script contacts the Azure Global device endpoint (that can be found in the DPS overview page, as shown in Figure 11) and provides the following information of the SARA-R5 series module:

- **Device ID:** device identity used for device authentication and access control (RoT UID)
- **ID Scope:** the ID assigned to a Device Provisioning Service when it is created by the user and is used to uniquely identify the specific provisioning service the device will register through
- X.509 certificates (paths on both the device certificate and private key)

The device will be registered with the related IoT Hub, and it will be displayed in the list of IoT devices.



The following code shows the fundamental parts of the Azure SDK python script involved in the connection process between the IoT device and DSP entity.

```

#-----
# X509 class
# A class with references to the certificate, key, and optional pass-phrase used to
# authenticate a TLS connection using x509 certificates
# X509(cert_file, key_file, pass_phrase=None)
#-----
x509 = X509(
    cert_file=device_cert_path,
    key_file=device_key_path,
    pass_phrase=None
)

#-----
# Create a client that can be used to run the registration of a device with
# provisioning service using X509 certificate authentication.
# create_from_x509_certificate(provisioning_host, registration_id, id_scope, x509,
# **kwargs)
# Returns: A ProvisioningDeviceClient which can register via Symmetric Key.
#-----
provisioning_device_client = ProvisioningDeviceClient.create_from_x509_certificate(
    provisioning_host=provisioning_host,
    registration_id=registration_id,
    id_scope=id_scope,
    x509=x509,
)

#-----
# Register the device with the provisioning service
# This is a synchronous call, meaning that this function will not return until the
# registration process has been completed successfully or the attempt has resulted in a
# failure.
# Returns: RegistrationResult indicating the result of the registration.
# (<xref:azure.iot.device.RegistrationResult>)
#-----
registration_result = await provisioning_device_client.register()
    
```

3 Example of Azure IoT Hub connection

3.1 Cellular module configuration

3.1.1 Store certificates in the module file system

After downloading the CA and CC certificates and PK from the module (as described in section 2.8), store them in the module file system via AT commands or using m-center.

3.1.1.1 AT commands procedure to store the file in the module

Use the +UDWNFILE AT command to store all the certificates and keys required for communication in the module file system.

Command	Response	Description
AT+UDWNFILE="BaltimoreCyberTrustRoot.pem", 1188	>-----BEGIN CERTIFICATE----- hDKXJioaldXgjUkK642M4UwtBV8ob2x... 4q5WTP468SQvvG5 -----END CERTIFICATE----- OK	After character ">" copy/paste the entire certificate. The file has been stored successfully.

Repeat the same procedure for the other certificates that may be necessary: e.g., for CC and PK.

3.1.1.2 m-center procedure to store the file in the module

Similarly, m-center evaluation software can be used to store the certificates file in the module. The software uses the +UDWNFILE AT command but it is masked by a simple GUI.

Any file can be stored in the module via the **File System Tab** (as shown in Figure 12), by clicking **Store file**. A window will open where the chosen file can be selected from Windows Explorer. Repeat the same procedure for the other certificates that may be necessary: e.g., for CC and PK.

By clicking **Dir**, the m-center window will display all the stored files.

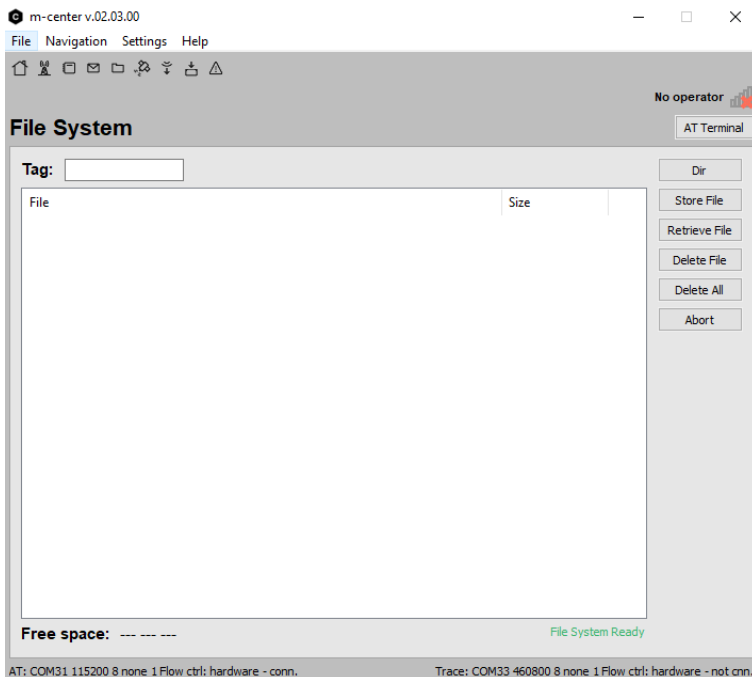


Figure 12: m-center File System tab

The u-blox m-center can be downloaded free of charge from the website, <http://www.u-blox.com>.

3.1.2 Check CA, CC, and PK in the file system


Command	Response	Description
AT+ULSTFILE=2,"BaltimoreCyberTrustRoot.pem"	+ULSTFILE: 1188 OK	CA availability in the module.
AT+ULSTFILE=2,"azure_ztp_device_cert.pem"	+ULSTFILE: 1224 OK	CC availability in the module.
AT+ULSTFILE=2,"azure_ztp_device_private_key.pem"	+ULSTFILE: 1679 OK	PK availability in the module

3.1.3 Certificates manager configuration

Command	Response	Description
AT+USECMNG=1,0,"azure_root_ca", "BaltimoreCyberTrustRoot.pem"	+USECMNG: 1,0,"azure_root_ca", "CB17E431673EE209FE455793F30AF A1C" OK	Import the CA in the certificates manager.
AT+USECMNG=1,1,"device_cert", "azure_ztp_device_cert.pem"	+USECMNG: 1,1,"device_cert", "50C3004AAE690124E3D7F96F904D7084" OK	Import the CC in the certificates manager.
AT+USECMNG=1,2,"device_yey", "azure_ztp_device_private_key.pem"	+USECMNG: 1,2,"device_key", "CD879AA22744A7211D3AF5D3BEFAFF29" OK	Import the client PK in the certificates manager.

3.1.3.1 Security profile configuration


Command	Response	Description
AT+USECPRF=0,0,3	OK	Set the certificate validation level 3.
AT+USECPRF=0,2,0	OK	Set automatic the cipher suite.
AT+USECPRF=0,3,"azure_root_ca"	OK	Set the trusted root certificate internal name.
AT+USECPRF=0,4,"SER-IoTHub.azure-devices.net"	OK	Set the expected server hostname.
AT+USECPRF=0,5,"device_cert"	OK	Set the CC internal name.
AT+USECPRF=0,6,"device_key"	OK	Set the client PK internal name.
AT+USECPRF=0,10,"SER-IoTHub.azure-devices.net"	OK	Set the Server Name Indication.




 SNI is a feature of SSL/TLS which uses an additional SSL/TLS extension header to specify the name of the server to which the client is connecting to. SNI configuration may be required to support the certificate handling used with virtual hosting provided by the various SSL/TLS enabled servers mostly in cloud-based infrastructures.

3.2 Azure IoT Hub connection via MQTT protocol

The best way to describe the interaction between a u-blox module and the Azure IoT Hub is through a simple use case. The following example describes an MQTT session that simulates a form of remote temperature control. The u-blox module is the MQTT client responsible for publishing temperature messages and receiving action messages from the Azure IoT server.

3.2.1 Module setup MQTT session

 Make sure to correctly activate an IP data connection before using the AT commands mentioned in this section. This is necessary because a packet switched (PS) data connection must be activated before creating a socket and connecting to the Azure server.

Command	Response	Description
AT+UMQTT=0, "0001234567890abcd"	OK	Set the unique client ID.  Use the device ID as client ID.
AT+UMQTT=2, "SER-IoTHub.azure-devices.net ", 8883	OK	Set the name of the remote server (the above endpoint address) and the server port (TLS MQTT).
AT+UMQTT=4, "SER-IoTHub.azure-devices.net/0001234567890abcd/?api-version=2018-06-30", ""	OK	Set username and password.  For the Username field, use <code>{iothubhostname}/{device_id}/?api-version=2018-06-30</code> , where <code>{iothubhostname}</code> is the full CA Name of the IoT Hub.  For the Password field, use a SAS token. However, note that if a X.509 certificate authentication is used, the SAS token password is not required.
AT+UMQTT=8, "devices/0001234567890abcd/messages/events/"	OK	Set the last will topic.
AT+UMQTT=11, 1, 0	OK	Enable the secure connection option using the profile 0.

3.2.2 MQTT example: connection, subscribe, publish and read

As an example, the following table shows the AT commands used to communicate with the Azure portal (using the module internal MQTT protocol application) and update the device properties.

To update reported properties, the device issues a request to IoT Hub via a publication over a designated MQTT topic. The following sequence describes how a device updates the reported properties in the device twin in IoT Hub:

- A device must first subscribe to the `$/iothub/twin/res/#` topic to receive the operation's responses from IoT Hub.
- A device sends a message that contains the device twin update to the `$/iothub/twin/PATCH/properties/reported/?$rid={request id}` topic. This message includes a **request ID** value.
- The service then sends a response message that contains the new ETag value for the reported properties collection on topic `$/iothub/twin/res/{status}/?$rid={request id}`. This response message uses the same **request ID** as the request.

The request message body contains a JSON document, that contains new values for reported properties.

To retrieve a **device** twin's properties, follow the steps below:

- First, a device subscribes to `$/iothub/twin/res/#`, to receive the operation's responses.

- Then, it sends an empty message to topic `$iothub/twin/GET/?$rid={request id}`, with a populated value for **request ID**.
- The service then sends a response message containing the device twin data on topic `$iothub/twin/res/{status}/?$rid={request id}`, using the same **request ID** as the request.

Command	Response	Description
AT+UMQTTTC=1	OK +UUMQTTTC: 1,1	Connect to the Azure IoT Hub and start a secure MQTT session.
AT+UMQTTTC=4,0,"\$iothub/twin/res/#"	OK +UUMQTTTC: 4,1,0,"\$iothub/twin/res/#"	Subscribe to the heating system control of the ground floor office #1.
AT+UMQTTTC=3,0,0,"\$iothub/twin/PATCH/properties/reported/?\$rid=11","dut.json"	OK +UUMQTTTC: 3,1 +UUMQTTTC: 6,1	Update device twin's reported properties using "MQTT publish a file to a topic". Received a new MQTT message.
AT+UMQTTTC=6	+UMQTTTC: 6,0,40,40,"\$iothub/twin/res/204/?\$rid=11&\$version=4",0,"" OK	Read last received message.
AT+UMQTTTC=2,0,0,0,"\$iothub/twin/GET/?\$rid=100",""	OK +UUMQTTTC: 2,1 +UUMQTTTC: 6,1	Retrieve device twin's properties using "MQTT publish to a topic". Received a new MQTT message.
AT+UMQTTTC=6	+UMQTTTC: 6,0,121,30,"\$iothub/twin/res/200/?\$rid=100",91,"{"desired":{"\$version":1},"reported":{"date":"2020-10-19","status":"success","\$version":4}}" OK	Read last received message.

For more details about the AT commands, see the SARA-R5 series AT commands manual [\[2\]](#). Further details on the IP data connection configuration are available in the SARA-R4 / SARA-R5 internet applications development guide document [\[4\]](#).

4 Integration with Azure IoT Explorer

IoT Explorer is a tool provided by Azure to set up the connection and monitoring of the SARA-R5 module to Azure Cloud applications. The software is provided as a free download and is available at <https://github.com/Azure/azure-iot-explorer/releases>.

The main features are:

- Simple Azure connection configuration
- Device twin and direct method functions
- Real-time Azure monitoring and alarm management
- Advanced diagnostics tools

More details about IoT Explorer functions are available on the official [Azure IoT Explorer page](#).

4.1 IoT Hub connection

The first time that Azure IoT Explorer is executed, the application is prompted for the user's IoT hub connection string. After providing the connection string, select **Connect**.

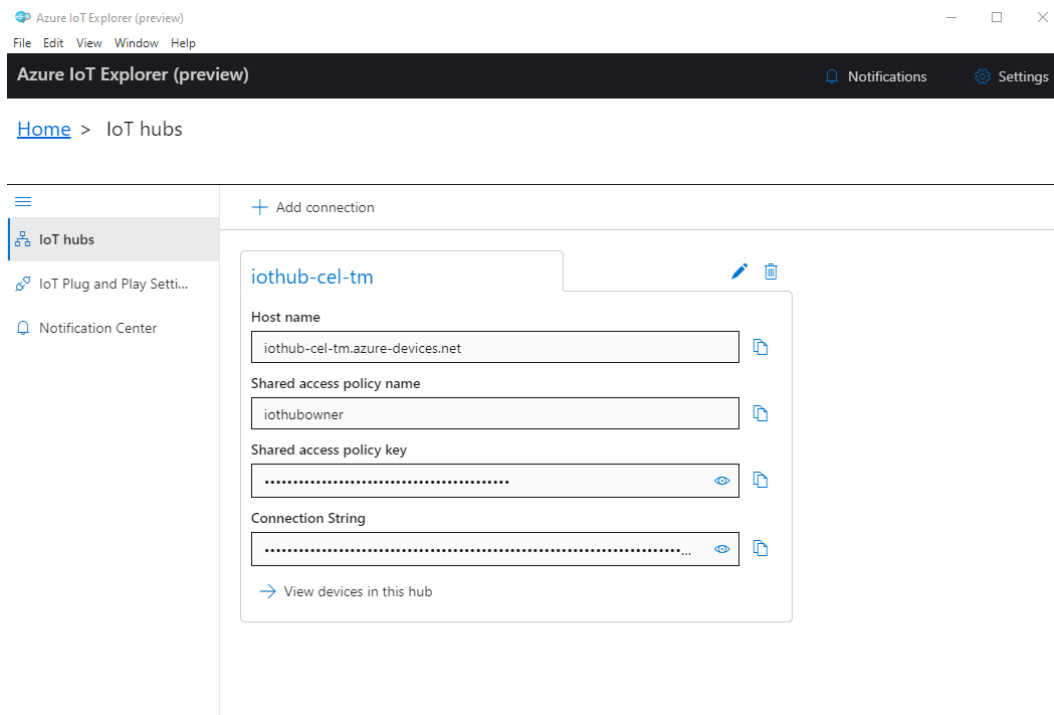


Figure 13: IoT Explorer hub connection

4.2 View Devices

After the tool connects to the IoT hub, it displays the **Devices** list page that lists the device identities registered with the IoT hub. The user can select any entry in the list to see more information.

On the Devices list page, it is possible to:

- Select **New** to register a new device with the Azure IoT hub. Then enter a **Device ID**. Use the default settings to automatically generate authentication keys and enable the connection to the IoT hub.
- Select a device and then select **Delete** to delete a device identity. Review the device details before completing this action to check if this is the correct device identity to delete.

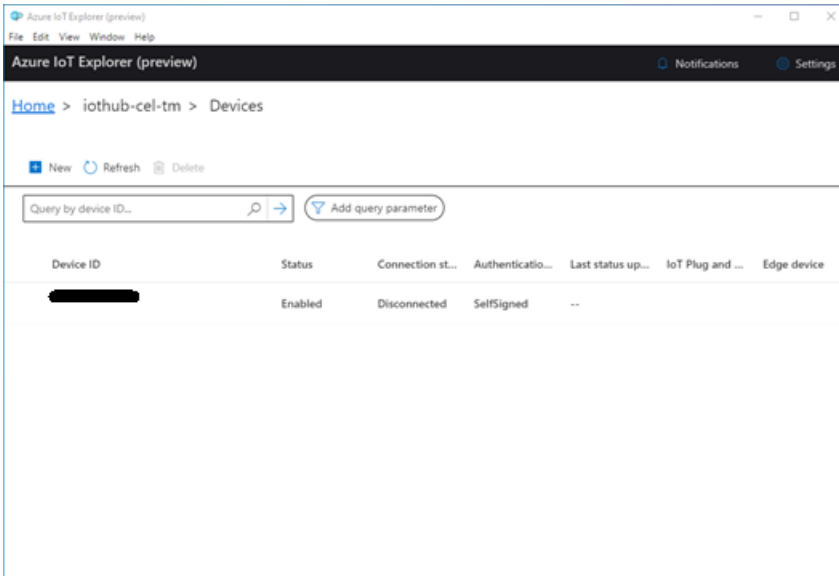


Figure 14: IoT Explorer list of devices

4.3 Interact with a device

On the **Devices** list page, select a value in the **Device ID** column to view the detail page for the registered device. For each device, there are two main sections: **Device** and **Digital Twin**.

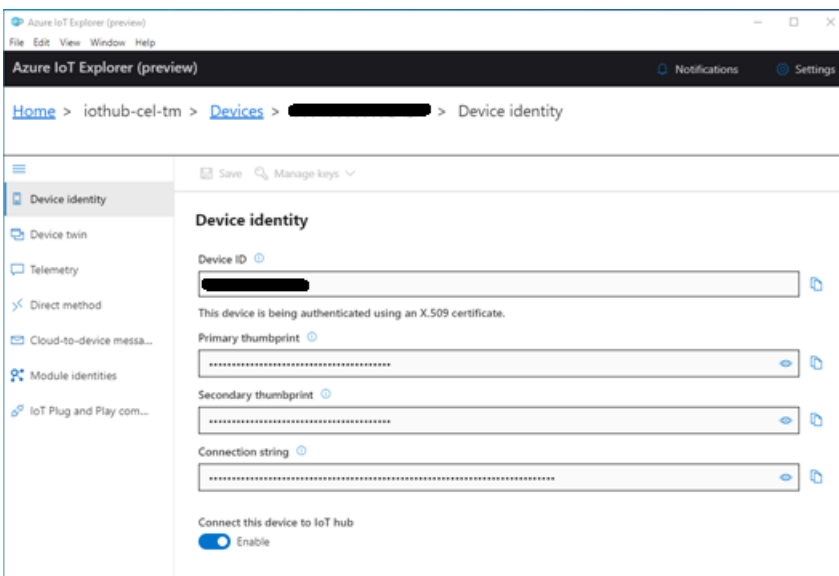


Figure 15: IoT Explorer device identity

The user can access the device twin information on the **Device twin** tab.

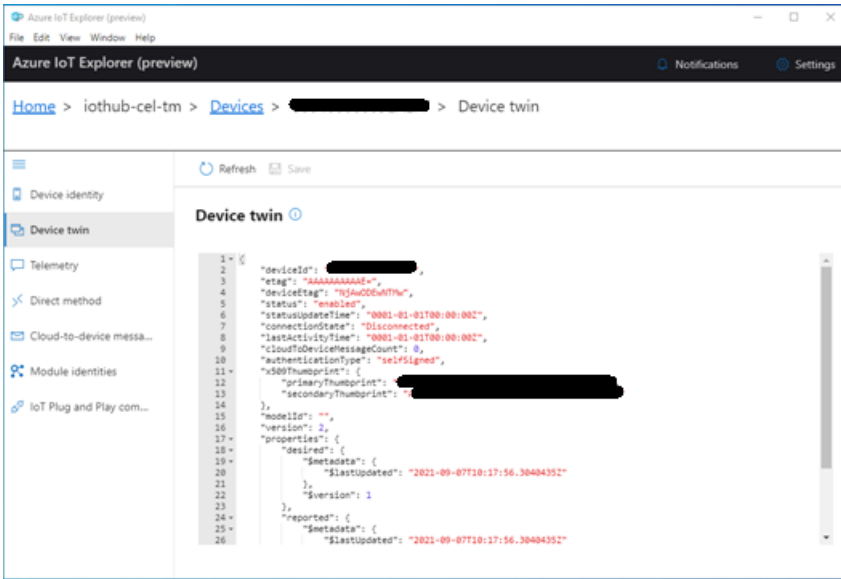


Figure 16: IoT Explorer device twin

Appendix


A Glossary

Abbreviation	Definition
CA	Certificate Authority
CC	Client Certificate
CN	Common Name
CSR	Certificate Signing Request
DPS	Device Provisioning Service
PK	Private Key
PoP	Proof-of-Possession
RoT	Root-of-Trust
ZTP	Zero-Touch Provisioning

Table 1: Explanation of the abbreviations and terms used

Related documentation

- [1] u-blox SARA-R5 series data sheet, [UBX-19016638](#)
- [2] u-blox SARA-R5 series AT commands manual, [UBX-19047455](#)
- [3] u-blox SARA-R5 series system integration manual, [UBX-19041356](#)
- [4] u-blox SARA-R5 internet application development guide, [UBX-20032566](#)
- [5] u-blox Thingstream documentation page: <https://developer.thingstream.io/>

 For product change notifications and regular updates of u-blox documentation, register on our website, www.u-blox.com.

Revision history

Revision	Date	Name	Comments
R01	25-Oct-2021	mreb	Initial release

Contact

For complete contact information, visit us at www.u-blox.com.

u-blox Offices

North, Central and South America

u-blox America, Inc.

Phone: +1 703 483 3180
Email: info_us@u-blox.com

Regional Office West Coast:

Phone: +1 408 573 3640
Email: info_us@u-blox.com

Technical Support:

Phone: +1 703 483 3185
Email: support@u-blox.com

Headquarters

Europe, Middle East, Africa

u-blox AG

Phone: +41 44 722 74 44
Email: info@u-blox.com
Support: support@u-blox.com

Asia, Australia, Pacific

u-blox Singapore Pte. Ltd.

Phone: +65 6734 3811
Email: info_ap@u-blox.com
Support: support_ap@u-blox.com

Regional Office Australia:

Phone: +61 3 9566 7255
Email: info_anz@u-blox.com
Support: support_ap@u-blox.com

Regional Office China (Beijing):

Phone: +86 10 68 133 545
Email: info_cn@u-blox.com
Support: support_cn@u-blox.com

Regional Office China (Chongqing):

Phone: +86 23 6815 1588
Email: info_cn@u-blox.com
Support: support_cn@u-blox.com

Regional Office China (Shanghai):

Phone: +86 21 6090 4832
Email: info_cn@u-blox.com
Support: support_cn@u-blox.com

Regional Office China (Shenzhen):

Phone: +86 755 8627 1083
Email: info_cn@u-blox.com
Support: support_cn@u-blox.com

Regional Office India:

Phone: +91 80 405 092 00
Email: info_in@u-blox.com
Support: support_in@u-blox.com

Regional Office Japan (Osaka):

Phone: +81 6 6941 3660
Email: info_jp@u-blox.com
Support: support_jp@u-blox.com

Regional Office Japan (Tokyo):

Phone: +81 3 5775 3850
Email: info_jp@u-blox.com
Support: support_jp@u-blox.com

Regional Office Korea:

Phone: +82 2 542 0861
Email: info_kr@u-blox.com
Support: support_kr@u-blox.com

Regional Office Taiwan:

Phone: +886 2 2657 1090
Email: info_tw@u-blox.com
Support: support_tw@u-blox.com