Microsoft

Mitsubishi C-lang Intelligent Function Unit

Demo Setup Guide and Demo Script version 1.0

Hisashi Goto
Sr. IoT Cloud Solution Architect
Microsoft

27th Dec 2021

Microsoft

Generate X.509 Certificates on your PC

Connect devices with X.509 certificates – Node.js

Generate X.509 Certificate (using Node.js)

Using WSL of laptop {C

Unzip master.zip to C:¥temp¥x509deviceCert ← download from Git https://github.com/Azure/azure-iot-sdk-node/archive/master.zip

```
git clone https://github.com/Azure/azure-iot-sdk-python.git
```

Open WSL terminal

```
cd  /mnt/c/temp¥x509deviceCert
```

```
cd azure-iot-sdk-node/provisioning/tools
npm install
```

← or /usr/bin/npm install

Edit create_test_cert.js @line70: days: 365

Create root certificate

```
node create_test_cert.js root mytestroot
```

Create Device Certificate

```
node create_test_cert.js device sample-device-01 mytestroot
```

Device ID of RD55UP12-V

List of Cert files →

Use them for IoT Central→

Copy them to RD55UP12-V →

X.509 cert files created	contents
mytestroot_cert.pem	The **public cert** of the root X509 certificate
mytestroot_key.pem	The **private key** for the root X509 certificate
mytestroot_fullchain.pem	The entire keychain for the root X509 certificate.
sampleDevice01_cert.pem	The **public cert** of the device X509 certificate
sampleDevice01_key.pem	The **private key** for the device X509 certificate
sampleDevice01_fullchain.pem	The entire keychain for the device X509 certificate.

Microsoft

Install Azure IoT SDK for Python

How to Install Azure IoT SDK for Python to RD55UP12-V

How to install Python 3.7

```
wget https://www.python.org/ftp/python/3.7.0/Python-3.7.0.tgz
tar zxvf Python-3.7.0.tgz
cd Python-3.7.0/
./configure --enable-optimizations
make -j4
sudo make altinstall
```

Make it sure Python version == 3.7?

```
python3 --version
```

install python package for Azure IoT

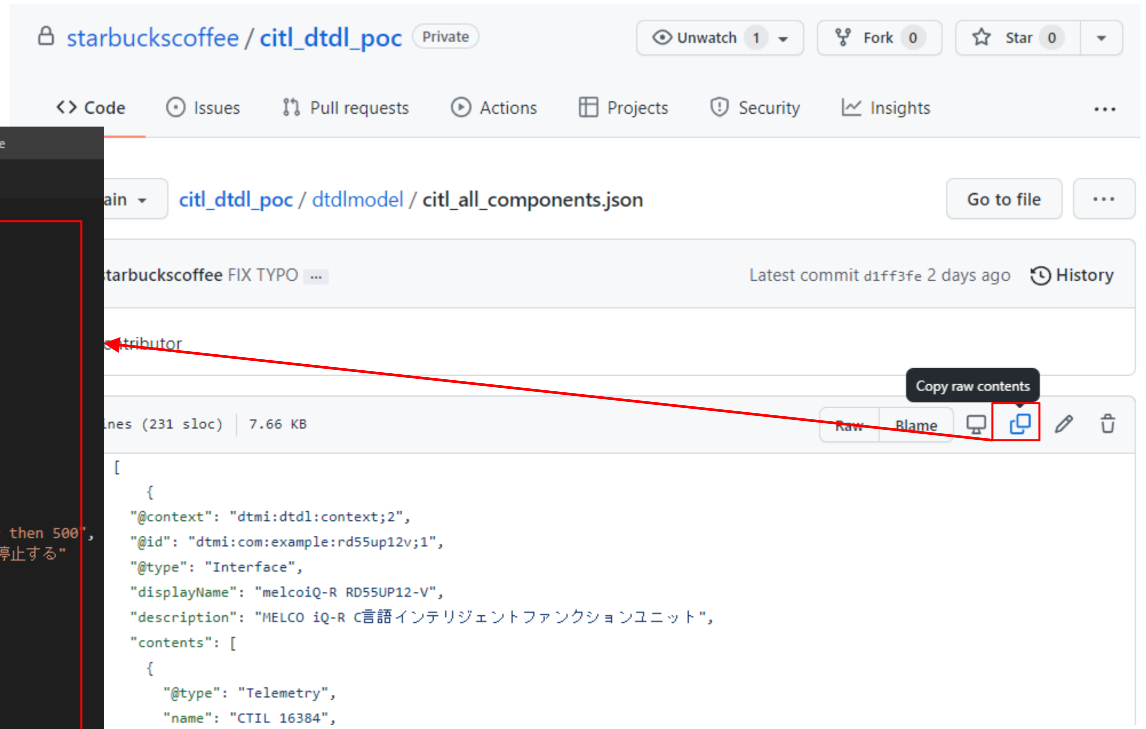
```
export DST="/usr/local/lib/python3.7/site-packages"
pip3 install azure-iot-device -t $DST
```

Microsoft

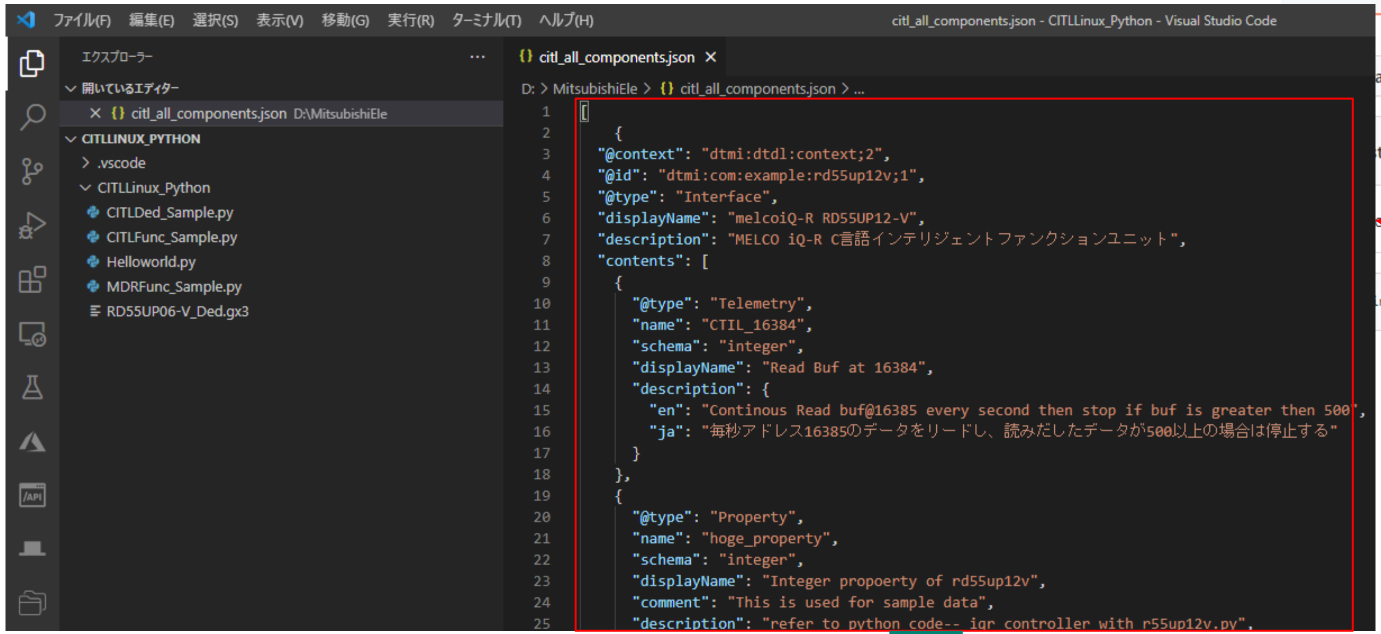
Setting up IoT Central

Set up your IoT Central Service

https://github.com/starbuckscoffee/citl_dtdl_poc/blob/main/dtdlmodel/citl_all_components.json



```
nes (231 sloc)   7.66 KB
[
    {
  "@context": "dtmi:dtdl:context;2",
  "@id": "dtmi:com:example:rd55up12v;1",
  "@type": "Interface",
  "displayName": "melcoiQ-R RD55UP12-V",
  "description": "MELCO iQ-R C言語インテリジェントファンクションユニット",
  "contents": [
    {
      "@type": "Telemetry",
      "name": "CTIL 16384",
```


ファイル(F) 編集(E) 選択(S) 表示(V) 移動(G) 実行(R) ターミナル(T) ヘルプ(H)
citl_all_components.json - CITLLinux_Python - Visual Studio Code
エクスプローラー
開いているエディター
citl_all_components.json D:\MitsubishiEle
CITLLINUX_PYTHON
.vscode
CITLLinux_Python
CITLDed_Sample.py
CITLFunc_Sample.py
Helloworld.py
MDRFunc_Sample.py
RD55UP06-V_Ded.gx3
citl_all_components.json
D: > MitsubishiEle > citl_all_components.json > ...

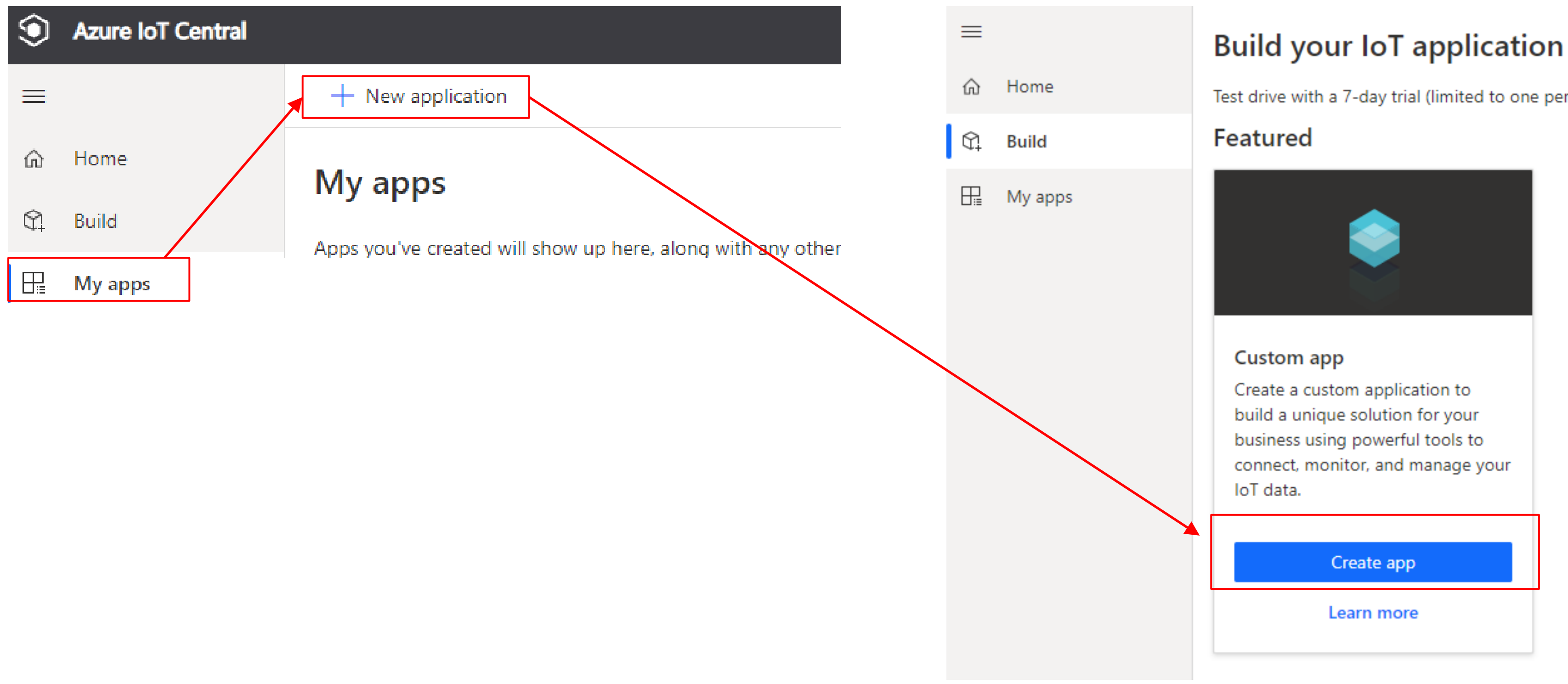
```
[
    {
  "@context": "dtmi:dtdl:context;2",
  "@id": "dtmi:com:example:rd55up12v;1",
  "@type": "Interface",
  "displayName": "melcoiQ-R RD55UP12-V",
  "description": "MELCO iQ-R C言語インテリジェントファンクションユニット",
  "contents": [
    {
      "@type": "Telemetry",
      "name": "CTIL_16384",
      "schema": "integer",
      "displayName": "Read Buf at 16384",
      "description": {
        "en": "Continous Read buf@16385 every second then stop if buf is greater then 500",
        "ja": "毎秒アドレス16385のデータをリードし、読みだしたデータが500以上の場合は停止する"
      }
    },
    {
      "@type": "Property",
      "name": "hoge_property",
      "schema": "integer",
      "displayName": "Integer propoerty of rd55up12v",
      "comment": "This is used for sample data",
      "description": "refer to python code-- igr controller with r55up12v.py",
```

Save it to Local File
File Name: citl_all_components.json

← You will use it when you import Plug and Play model to IoT Central

Set Up your IoT Central Service

https://apps.azureiotcentral.com



My apps

Apps you've created will show up here, along with any other

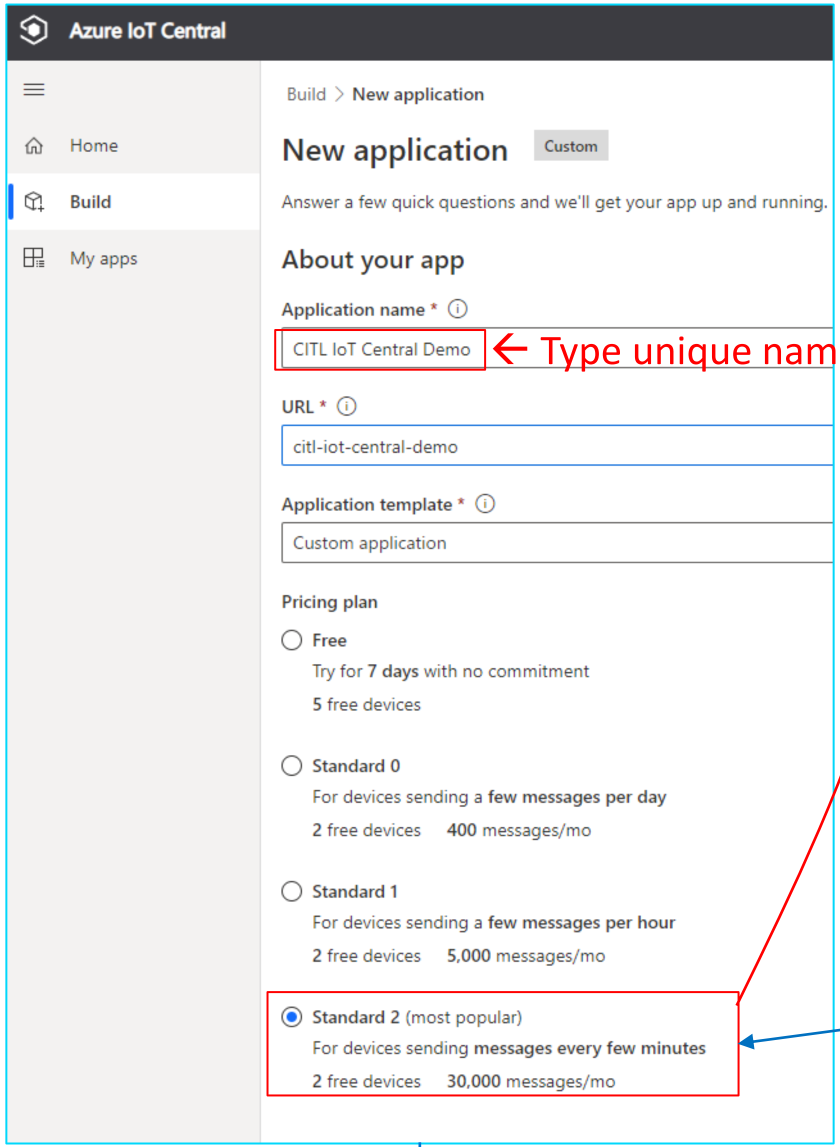
Build your IoT application

Test drive with a 7-day trial (limited to one per account), or b

Custom app

Create a custom application to build a unique solution for your business using powerful tools to connect, monitor, and manage your IoT data.

Set Up your IoT Central Service



Azure IoT Central

Home
Build
My apps

Build > New application

New application Custom

Answer a few quick questions and we'll get your app up and running.

About your app

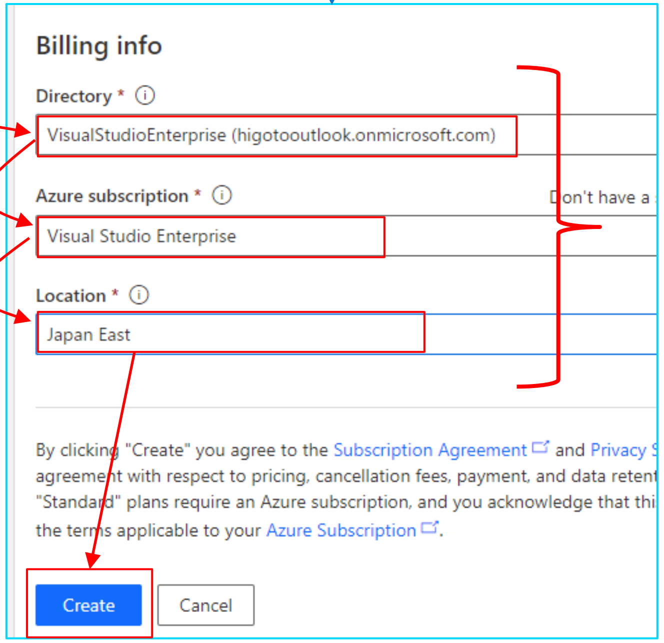
Application name * CITL IoT Central Demo ← Type unique name

URL * citl-iot-central-demo

Application template * Custom application

Pricing plan

- Free
  Try for 7 days with no commitment
  5 free devices
- Standard 0
  For devices sending a few messages per day
  2 free devices 400 messages/mo
- Standard 1
  For devices sending a few messages per hour
  2 free devices 5,000 messages/mo
- [✓ Standard 2 (most popular)]
  For devices sending messages every few minutes
  2 free devices 30,000 messages/mo



Billing info

Directory * VisualStudioEnterprise (higotooutlook.onmicrosoft.com)

Azure subscription * Visual Studio Enterprise Don't have a

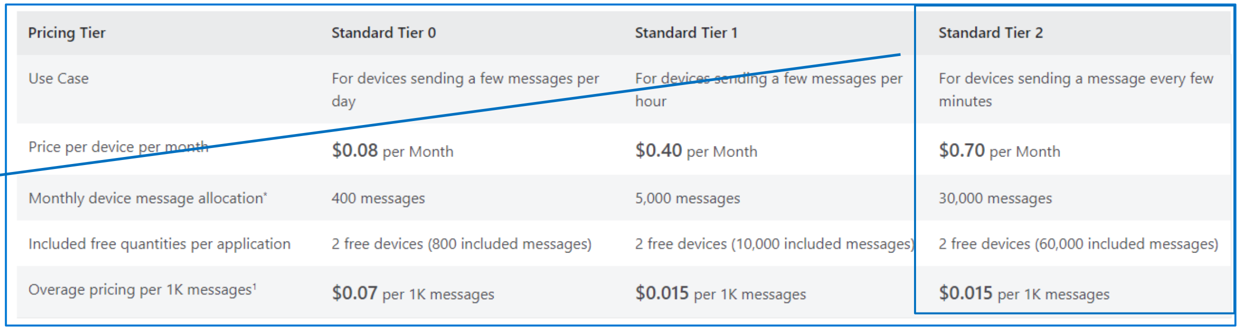
Location * Japan East

By clicking "Create" you agree to the Subscription Agreement and Privacy S agreement with respect to pricing, cancellation fees, payment, and data retent "Standard" plans require an Azure subscription, and you acknowledge that thi the terms applicable to your Azure Subscription.

Create Cancel

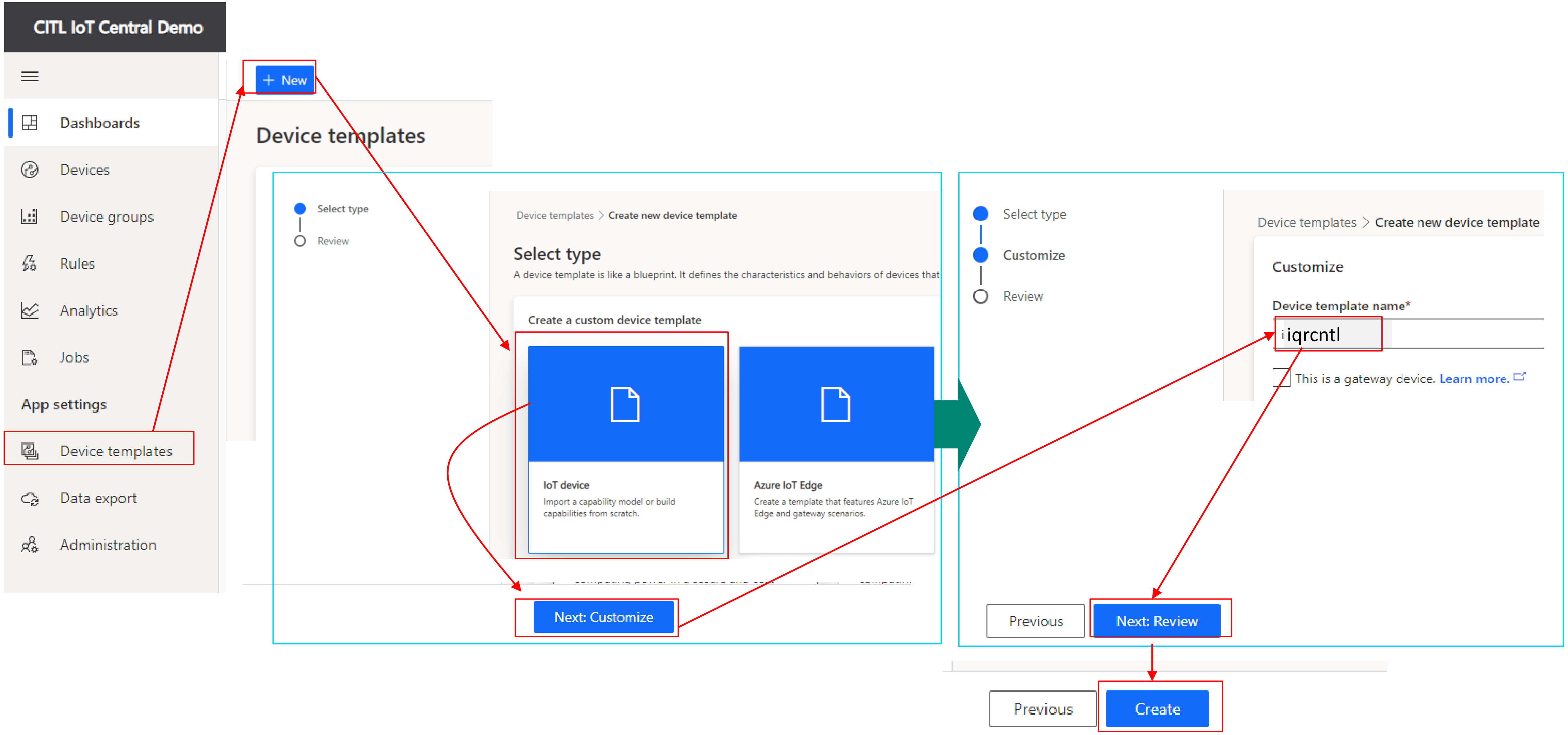
← Select your Azure Subscriptions and Location of IoT Central

Pricing Info

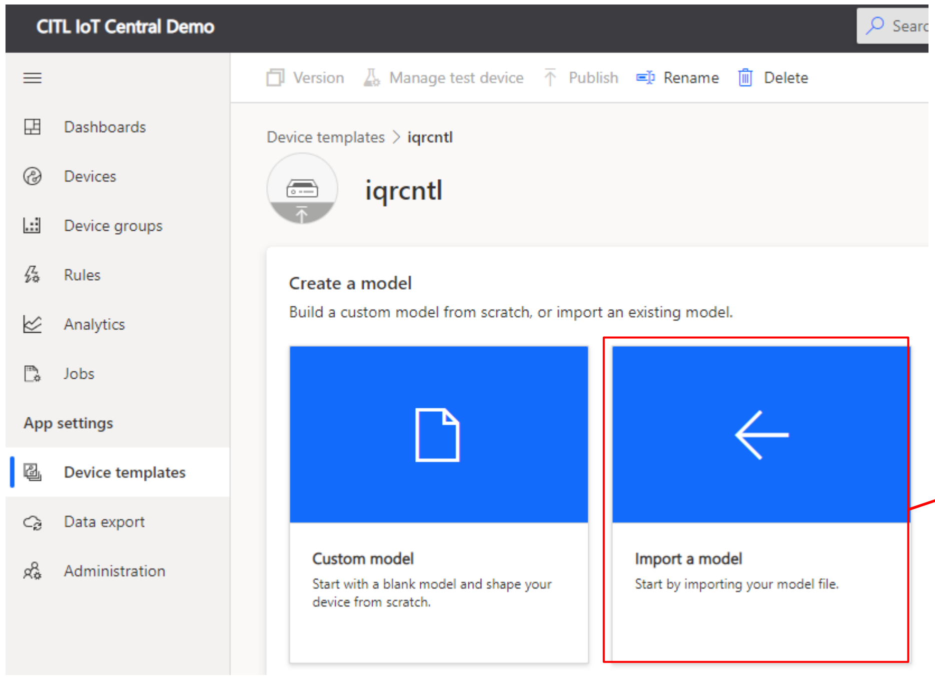


Pricing Tier	Standard Tier 0	Standard Tier 1	Standard Tier 2
Use Case	For devices sending a few messages per day	For devices sending a few messages per hour	For devices sending a message every few minutes
Price per device per month	$0.08 per Month	$0.40 per Month	$0.70 per Month
Monthly device message allocation[1]	400 messages	5,000 messages	30,000 messages
Included free quantities per application	2 free devices (800 included messages)	2 free devices (10,000 included messages)	2 free devices (60,000 included messages)
Overage pricing per 1K messages[1]	$0.07 per 1K messages	$0.015 per 1K messages	$0.015 per 1K messages

Set up your IoT Central Service


CITL IoT Central Demo
Dashboards
Devices
Device groups
Rules
Analytics
Jobs
App settings
Device templates
Data export
Administration
+ New
Device templates
Select type
Review
Device templates > Create new device template
Select type
A device template is like a blueprint. It defines the characteristics and behaviors of devices that
Create a custom device template
IoT device
Import a capability model or build capabilities from scratch.
Azure IoT Edge
Create a template that features Azure IoT Edge and gateway scenarios.
Next: Customize
Select type
Customize
Review
Device templates > Create new device template
Customize
Device template name*
iqrcntl
This is a gateway device. Learn more.
Previous
Next: Review
Previous
Create

Set up your IoT Central Service



Device templates > iqrcntl

iqrcntl

Create a model

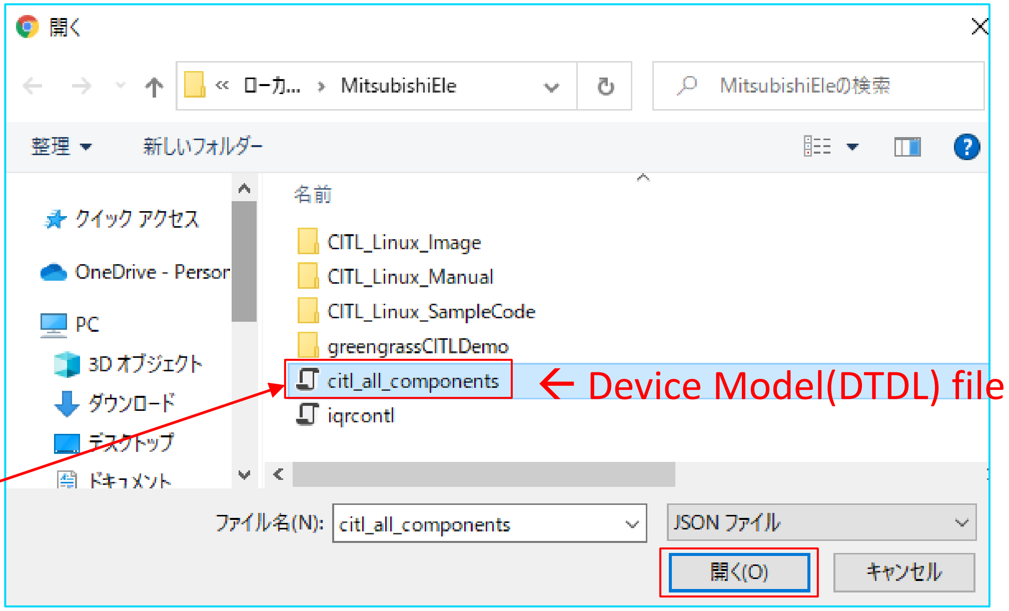
Build a custom model from scratch, or import an existing model.

Custom model

Start with a blank model and shape your device from scratch.

Import a model

Start by importing your model file.

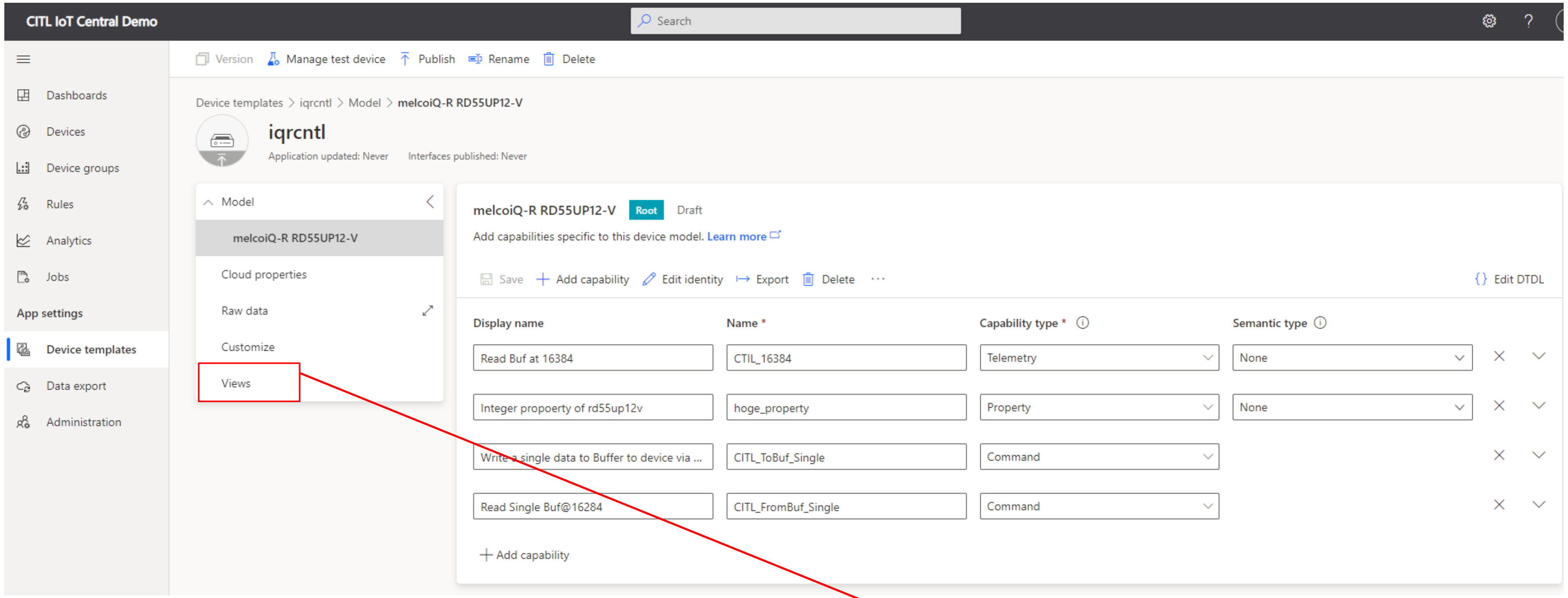


ファイル名(N): citl_all_components

JSON ファイル

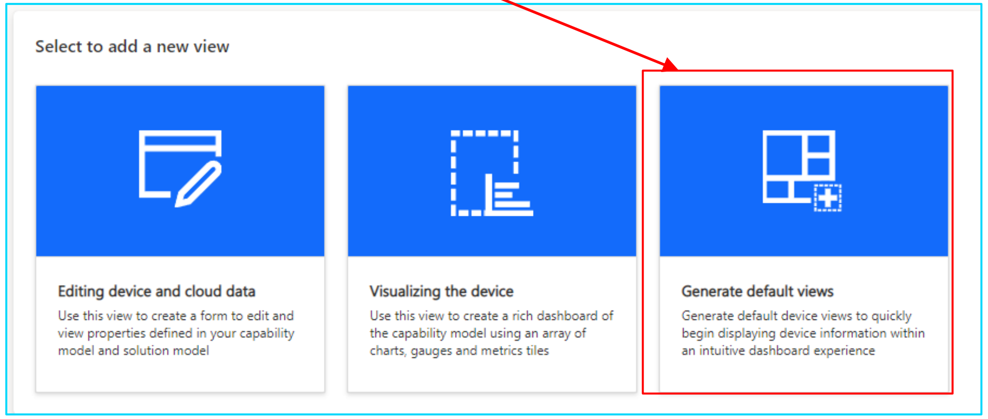
← Device Model(DTDL) file copied from Github

Set up your IoT Central Service



Add capabilities specific to this device model. Learn more

Display name	Name *	Capability type *	Semantic type
Read Buf at 16384	CITL_16384	Telemetry	None
Integer propoerty of rd55up12v	hoge_property	Property	None
Write a single data to Buffer to device via ...	CITL_ToBuf_Single	Command	
Read Single Buf@16284	CITL_FromBuf_Single	Command	



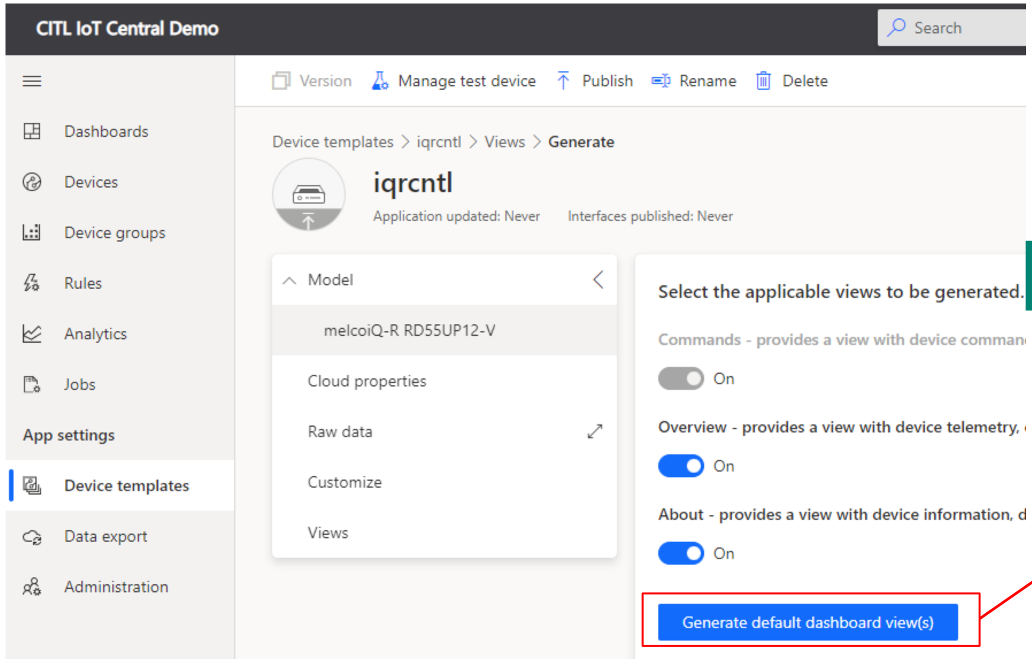
Select to add a new view

Editing device and cloud data
Use this view to create a form to edit and view properties defined in your capability model and solution model

Visualizing the device
Use this view to create a rich dashboard of the capability model using an array of charts, gauges and metrics tiles

Generate default views
Generate default device views to quickly begin displaying device information within an intuitive dashboard experience

Set up your IoT Central Service



CITL IoT Central Demo

Device templates > iqrcntl > Views > Generate

iqrcntl

Application updated: Never   Interfaces published: Never

Model

melcoiQ-R RD55UP12-V

Cloud properties

Raw data

Customize

Views

Select the applicable views to be generated.

Commands - provides a view with device comman

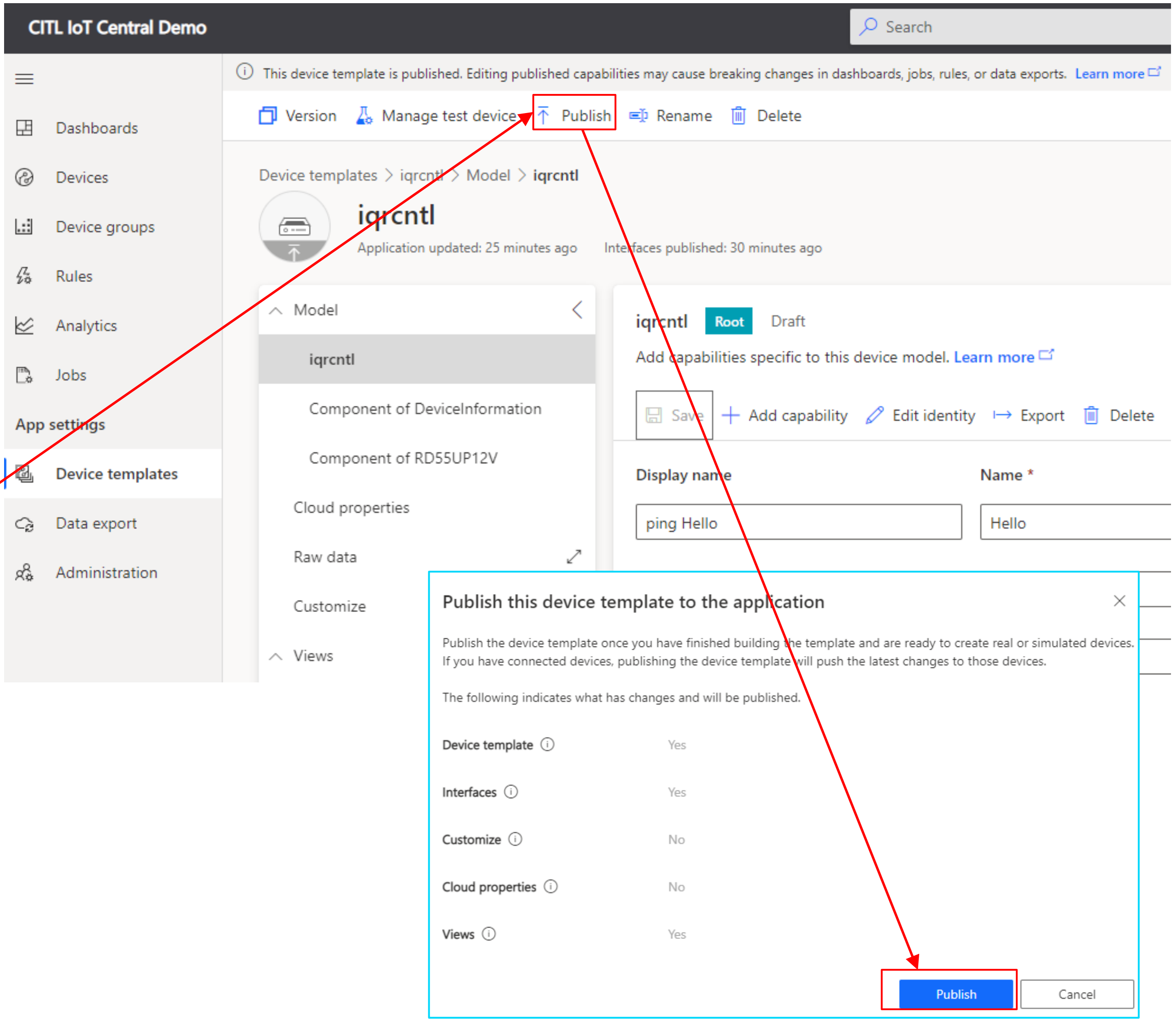
On

Overview - provides a view with device telemetry,

On

About - provides a view with device information, d

On



CITL IoT Central Demo

This device template is published. Editing published capabilities may cause breaking changes in dashboards, jobs, rules, or data exports. Learn more

Device templates > iqrcntl > Model > iqrcntl

iqrcntl

Application updated: 25 minutes ago   Interfaces published: 30 minutes ago

Model

iqrcntl

Component of DeviceInformation

Component of RD55UP12V

Cloud properties

Raw data

Customize

Views

iqrcntl Root Draft

Add capabilities specific to this device model. Learn more

Display name: ping Hello

Name *: Hello

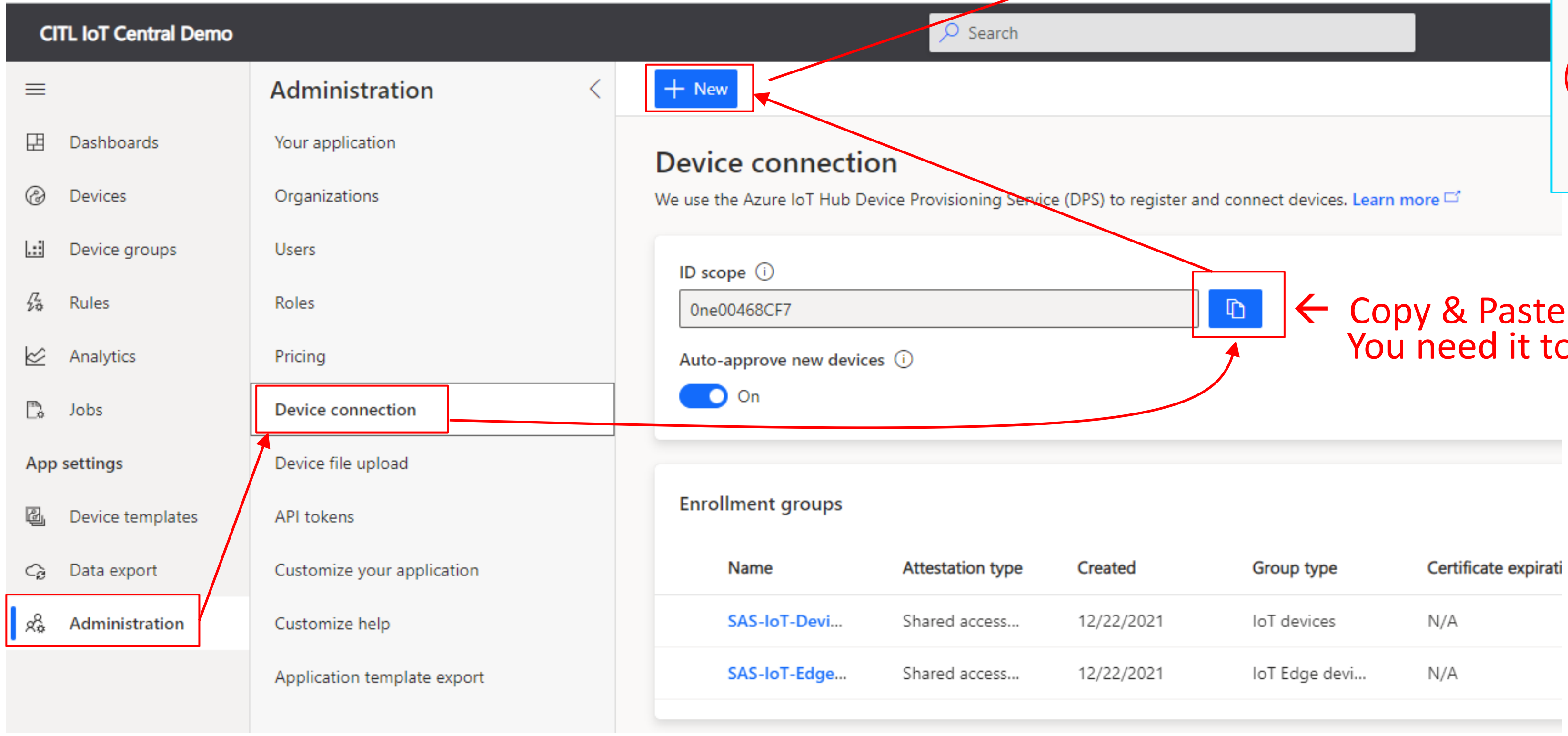
Publish this device template to the application

Publish the device template once you have finished building the template and are ready to create real or simulated devices. If you have connected devices, publishing the device template will push the latest changes to those devices.

The following indicates what has changes and will be published.

Device template	Yes
Interfaces	Yes
Customize	No
Cloud properties	No
Views	Yes

Set up your IoT Central Service



CITL IoT Central Demo

Search

Dashboards, Devices, Device groups, Rules, Analytics, Jobs

App settings

Device templates, Data export, Administration

Administration

Your application, Organizations, Users, Roles, Pricing, Device connection, Device file upload, API tokens, Customize your application, Customize help, Application template export

+ New

Device connection

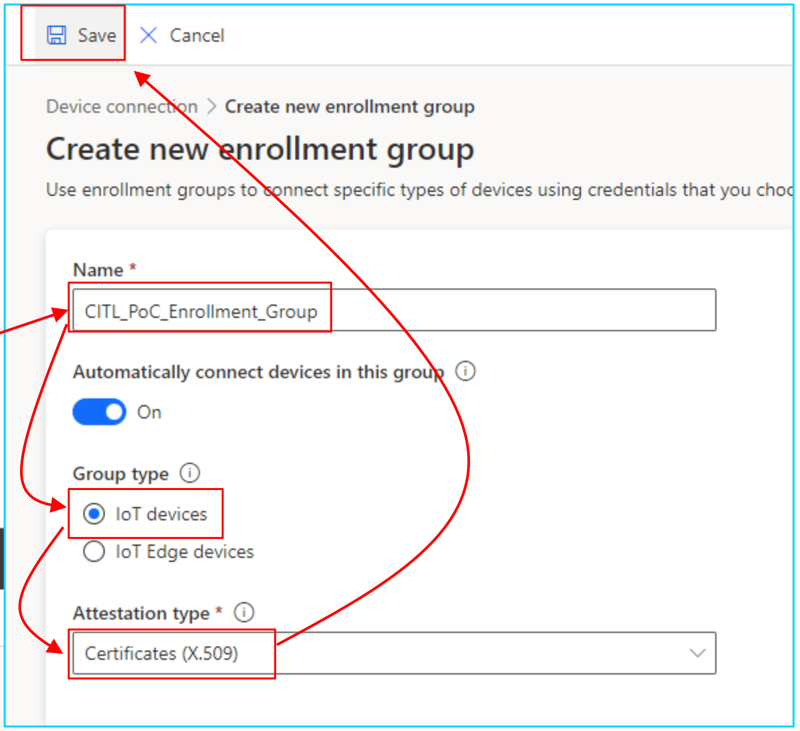
We use the Azure IoT Hub Device Provisioning Service (DPS) to register and connect devices. Learn more

ID scope: 0ne00468CF7

Auto-approve new devices: On

Enrollment groups

Name	Attestation type	Created	Group type	Certificate expirati
SAS-IoT-Devi...	Shared access...	12/22/2021	IoT devices	N/A
SAS-IoT-Edge...	Shared access...	12/22/2021	IoT Edge devi...	N/A



Save Cancel

Device connection > Create new enrollment group

Create new enrollment group

Use enrollment groups to connect specific types of devices using credentials that you cho

Name *: CITL_PoC_Enrollment_Group

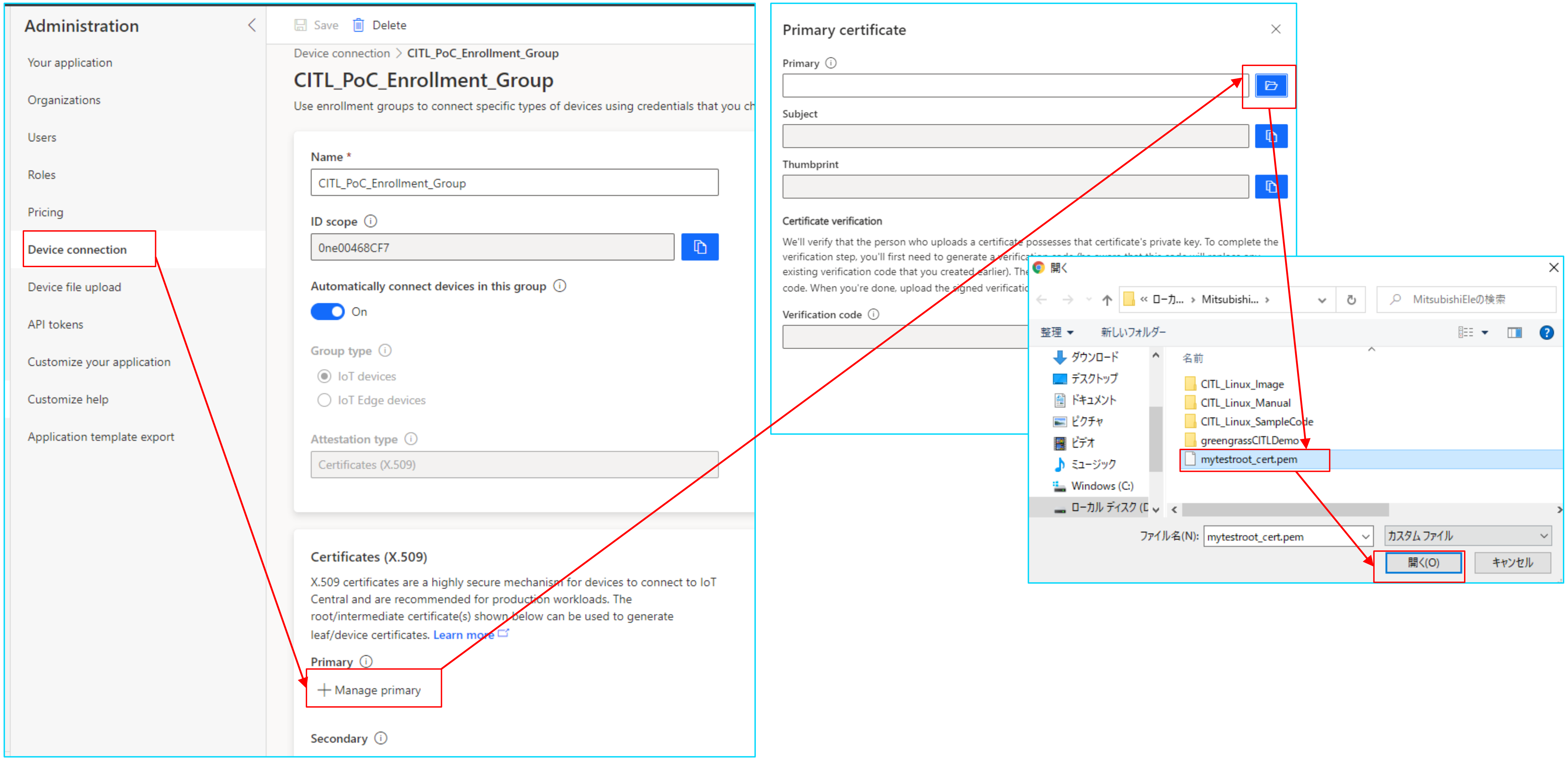
Automatically connect devices in this group: On

Group type: [✓ IoT devices] IoT Edge devices

Attestation type *: Certificates (X.509)

Copy & Paste it to Memopad
You need it to set IOTHUB_DEVICE_DPS_ID_SCOPE later

Set up your IoT Central Service



Administration

Your application

Organizations

Users

Roles

Pricing

Device connection

Device file upload

API tokens

Customize your application

Customize help

Application template export

Save Delete

Device connection > CITL_PoC_Enrollment_Group

CITL_PoC_Enrollment_Group

Use enrollment groups to connect specific types of devices using credentials that you ch

Name *: CITL_PoC_Enrollment_Group

ID scope: 0ne00468CF7

Automatically connect devices in this group: On

Group type: [✓ IoT devices] IoT Edge devices

Attestation type: Certificates (X.509)

Certificates (X.509)

X.509 certificates are a highly secure mechanism for devices to connect to IoT Central and are recommended for production workloads. The root/intermediate certificate(s) shown below can be used to generate leaf/device certificates. Learn more

Primary

+ Manage primary

Secondary

Primary certificate

Primary

Subject

Thumbprint

Certificate verification

We'll verify that the person who uploads a certificate possesses that certificate's private key. To complete the verification step, you'll first need to generate a verificati existing verification code that you created earlier). The code. When you're done, upload the signed verificatio

Verification code

開く

« ローカ... › Mitsubishi... ›

MitsubishiEleの検索

整理 ▾ 新しいフォルダー

ダウンロード

デスクトップ

ドキュメント

ピクチャ

ビデオ

ミュージック

Windows (C:)

ローカル ディスク (D

名前

CITL_Linux_Image

CITL_Linux_Manual

CITL_Linux_SampleCode

greengrassCITLDemo

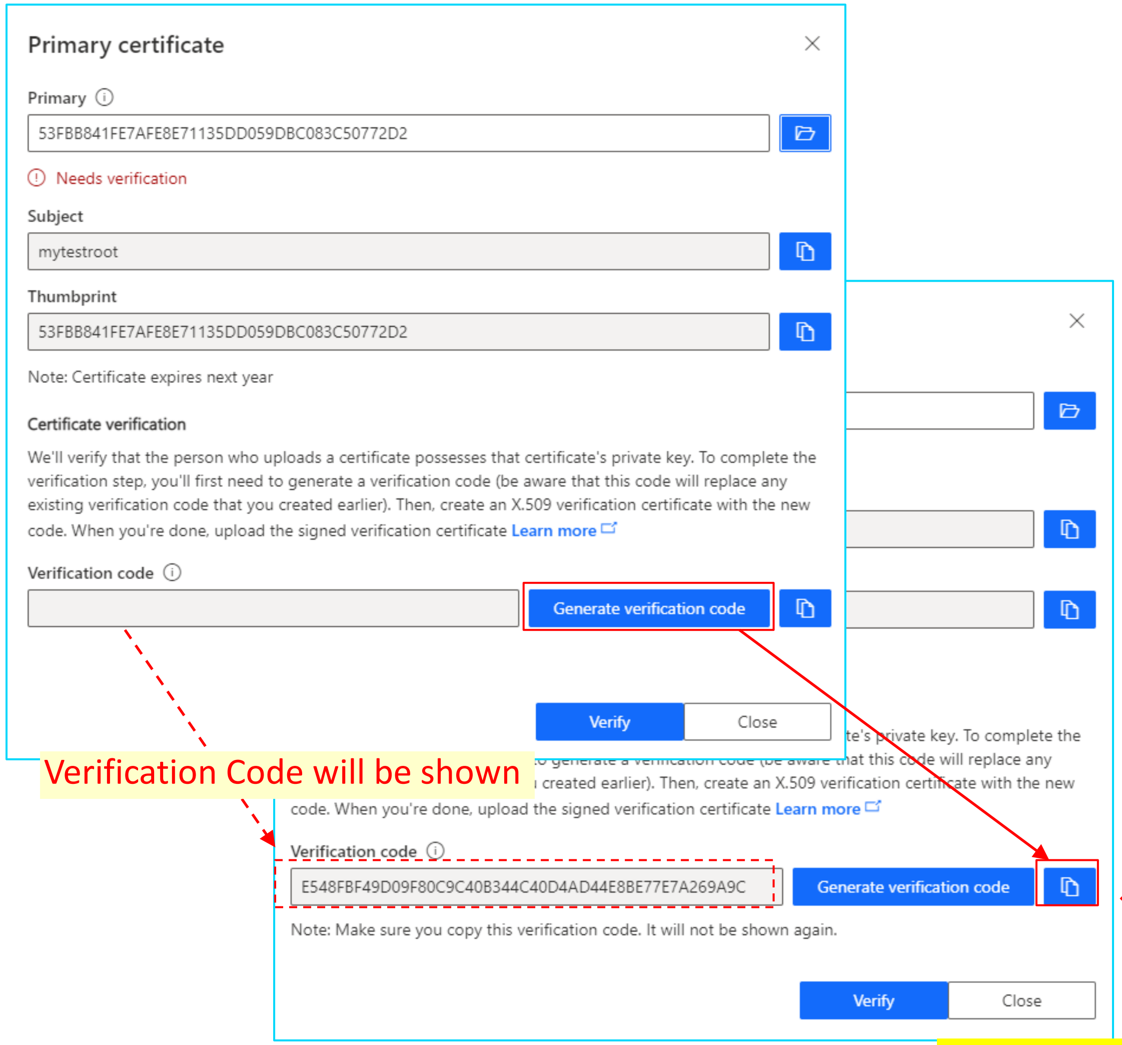
mytestroot_cert.pem

ファイル名(N): mytestroot_cert.pem

カスタム ファイル

開く(O) キャンセル

Set up your IoT Central Service



Primary certificate

Primary

53FBB841FE7AFE8E71135DD059DBC083C50772D2

Needs verification

Subject

mytestroot

Thumbprint

53FBB841FE7AFE8E71135DD059DBC083C50772D2

Note: Certificate expires next year

Certificate verification

We'll verify that the person who uploads a certificate possesses that certificate's private key. To complete the verification step, you'll first need to generate a verification code (be aware that this code will replace any existing verification code that you created earlier). Then, create an X.509 verification certificate with the new code. When you're done, upload the signed verification certificate Learn more

Verification code

E548FBF49D09F80C9C40B344C40D4AD44E8BE77E7A269A9C

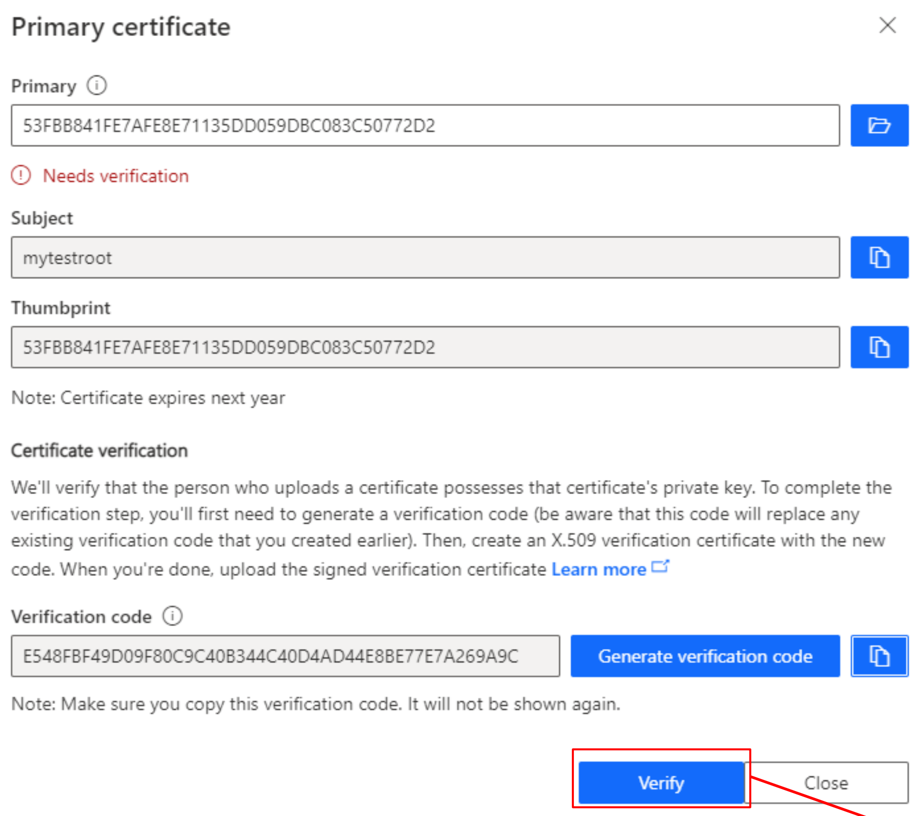
Note: Make sure you copy this verification code. It will not be shown again.

Generate Verification Cert file : verification_cert.pem ← Use it in Next slide

Paste it here

```
node create_test_cert.js verification --ca mytestroot_cert.pem  --key mytestroot_key.pem --nonce  your_verification_code
```

Set up your IoT Central Service



Primary certificate

Primary ⓘ

53FBB841FE7AFE8E71135DD059DBC083C50772D2

Needs verification

Subject

mytestroot

Thumbprint

53FBB841FE7AFE8E71135DD059DBC083C50772D2

Note: Certificate expires next year

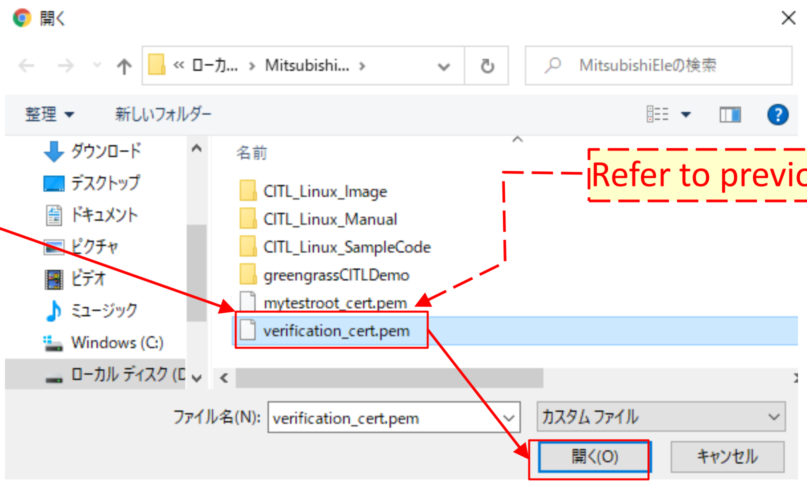
Certificate verification

We'll verify that the person who uploads a certificate possesses that certificate's private key. To complete the verification step, you'll first need to generate a verification code (be aware that this code will replace any existing verification code that you created earlier). Then, create an X.509 verification certificate with the new code. When you're done, upload the signed verification certificate Learn more

Verification code ⓘ

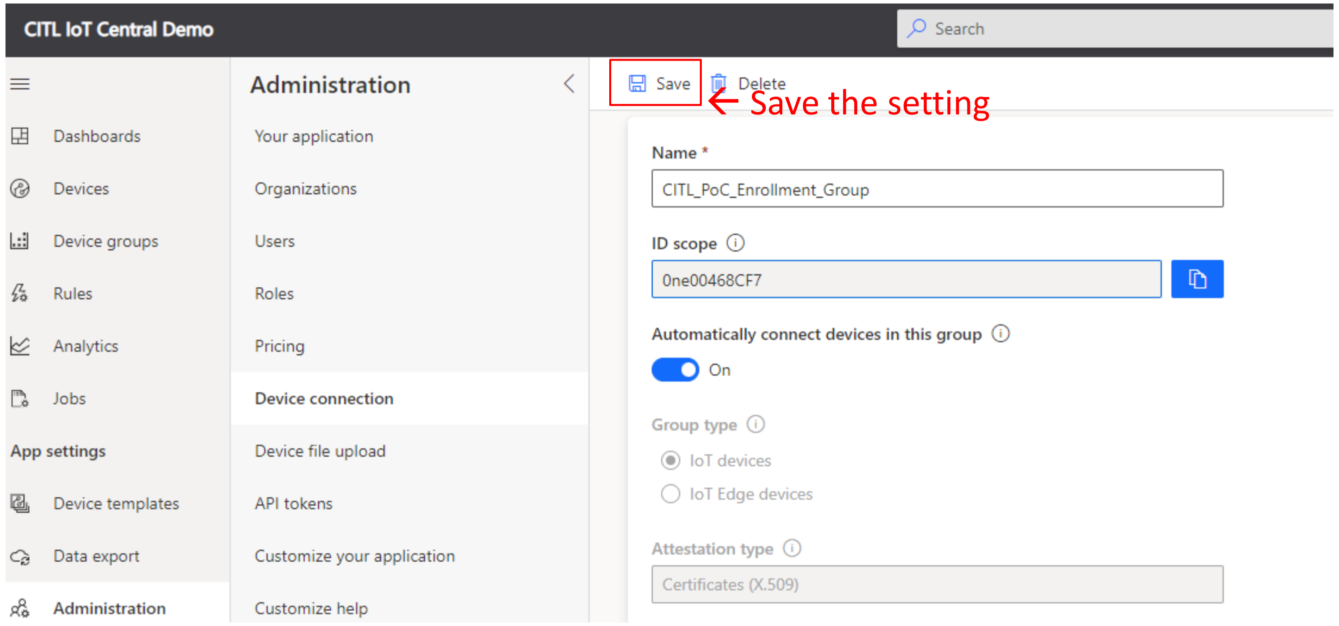
E548FBF49D09F80C9C40B344C40D4AD44E8BE77E7A269A9C

Note: Make sure you copy this verification code. It will not be shown again.



Refer to previous slide

Set up your IoT Central Service



Name *

CITL_PoC_Enrollment_Group

ID scope

0ne00468CF7

Automatically connect devices in this group

On

Group type

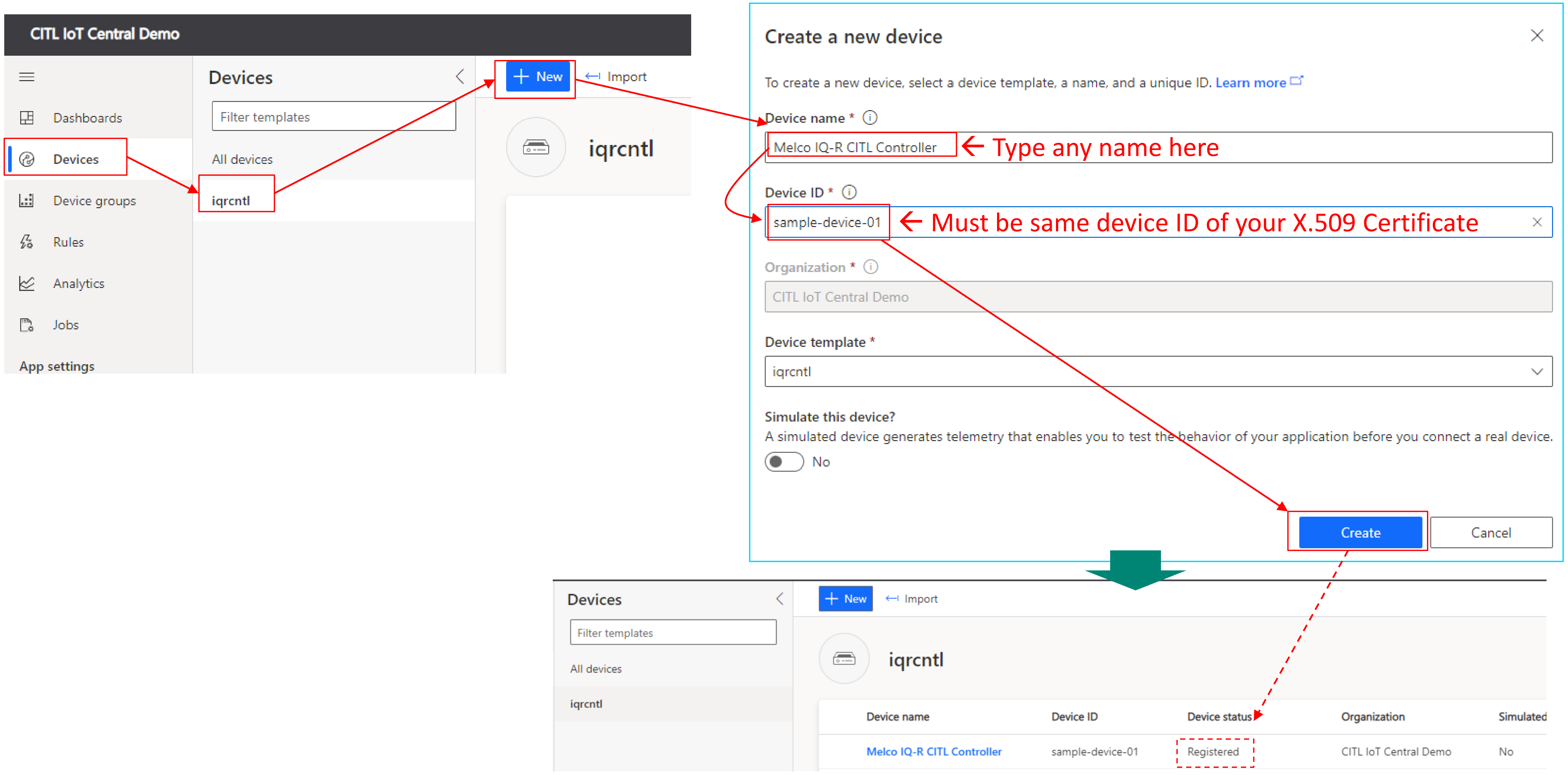
[✓ IoT devices]

IoT Edge devices

Attestation type

Certificates (X.509)

Set up your IoT Central Service



Create a new device

To create a new device, select a device template, a name, and a unique ID. Learn more

Device name *: Melco IQ-R CITL Controller ← Type any name here

Device ID *: sample-device-01 ← Must be same device ID of your X.509 Certificate

Organization *: CITL IoT Central Demo

Device template *: iqrcntl

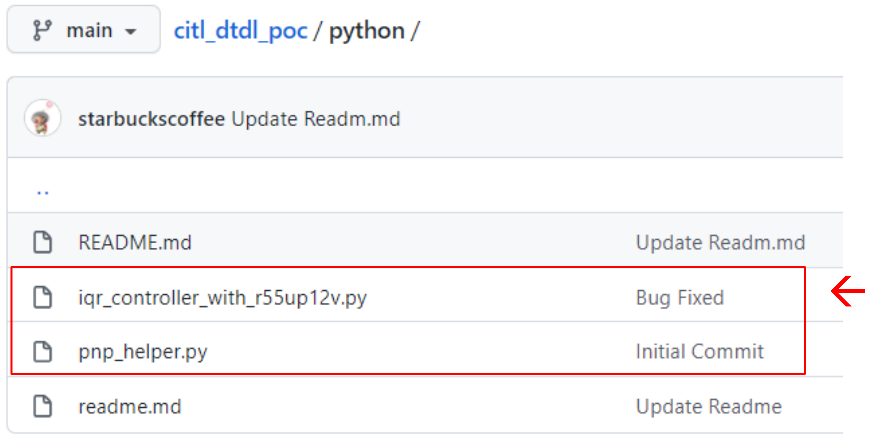
Simulate this device?
A simulated device generates telemetry that enables you to test the behavior of your application before you connect a real device.
No

Create | Cancel

Device name	Device ID	Device status	Organization	Simulated
Melco IQ-R CITL Controller	sample-device-01	Registered	CITL IoT Central Demo	No

Mitsubishi RD55UP12-V Setting

https://github.com/starbuckscoffee/citl_dtdl_poc/tree/main/python



main citl_dtdl_poc / python /

starbuckscoffee Update Readm.md

..

README.md	Update Readm.md
iqr_controller_with_r55up12v.py	Bug Fixed
pnp_helper.py	Initial Commit
readme.md	Update Readme

← You need to copy/download them to RD55UP12V

ip address of PD55UP12-V

Example: how to copy python code to RD55Up12-V

```
scp  C:¥temp¥citl_poc_temp¥citl_dtdl_poc¥python¥iqr_controller_with_r55up12v.py root@192.168.1.10:/root/gitclonedir/iqr_controller_with_r55up12v.py
```

```
scp  C:¥temp¥citl_poc_temp¥citl_dtdl_poc¥python¥pnp_helper.py.py root@192.168.1.10:/root/gitclonedir/pnp_helper.py
```

Example: how to copy X.509 Device Certificates to RD55Up12-V

```
scp  C:¥temp¥citl_poc_temp¥sampleDevice01_cert.pem  root@192.168.1.10:/root/sampleDevice01_cert.pem
```

```
scp  C:¥temp¥citl_poc_temp¥sampleDevice01_key.pem  root@192.168.1.10:/root/sampleDevice01_key.pem
```

Mitsubishi RD55UP12-V Setting

Add these lines to .bashrc

```
export IOTHUB_DEVICE_DPS_ENDPOINT="global.azure-devices-provisioning.net"
export IOTHUB_DEVICE_SECURITY_TYPE="DPS"
export IOTHUB_DEVICE_DPS_ID_SCOPE="0ne00xxxxyy"   ← Replace ID_SCOPE of your Central App
export IOTHUB_DEVICE_DPS_DEVICE_ID="sample-device-01"
export IOTHUB_DEVICE_X509_CERT="/root/sampleDevice01_cert.pem"
export IOTHUB_DEVICE_X509_KEY="/root/sampleDevice01_key.pem"
export PASS_PHRASE=“1234"
export DST="/usr/local/lib/python3.7/site-packages"   ← Environment used for pip3 install command
```

After you save .bashrc

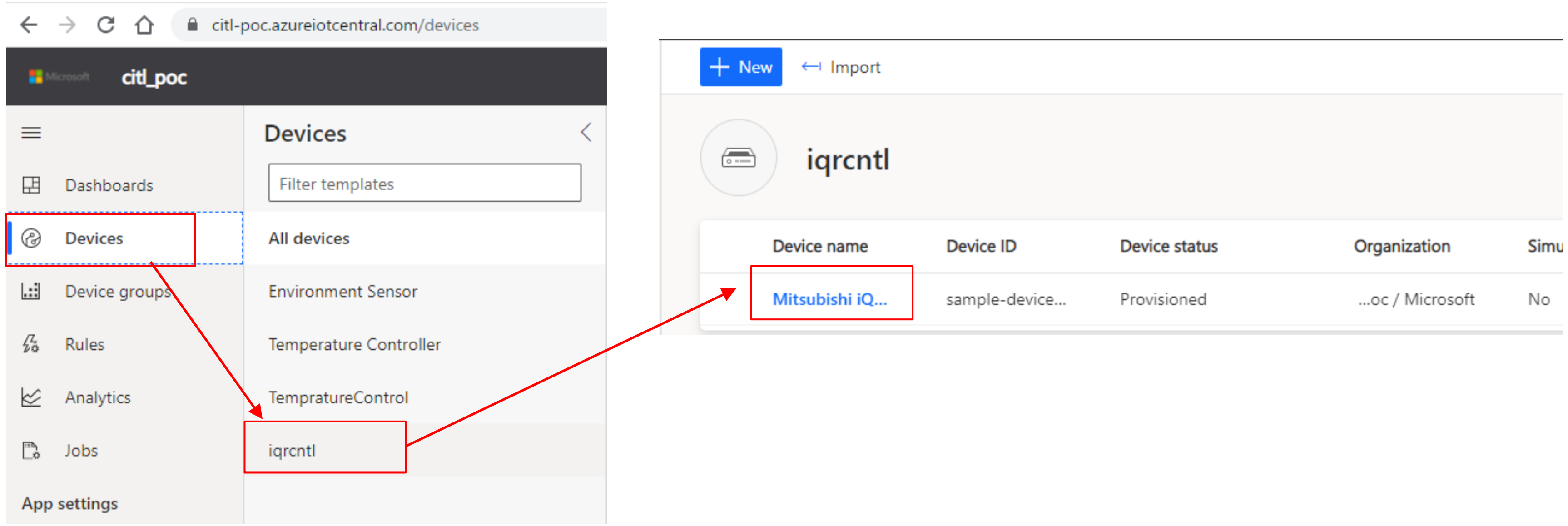
```
source /root/.bashrc
```

Run Python Code on RD55UP12V

Run Python Program on RD55UP12V

```
root/gitclonedir
python3 iqr_controller_with_r55up12v.py
```

IoT Central: Check if Python Program is Up and Running



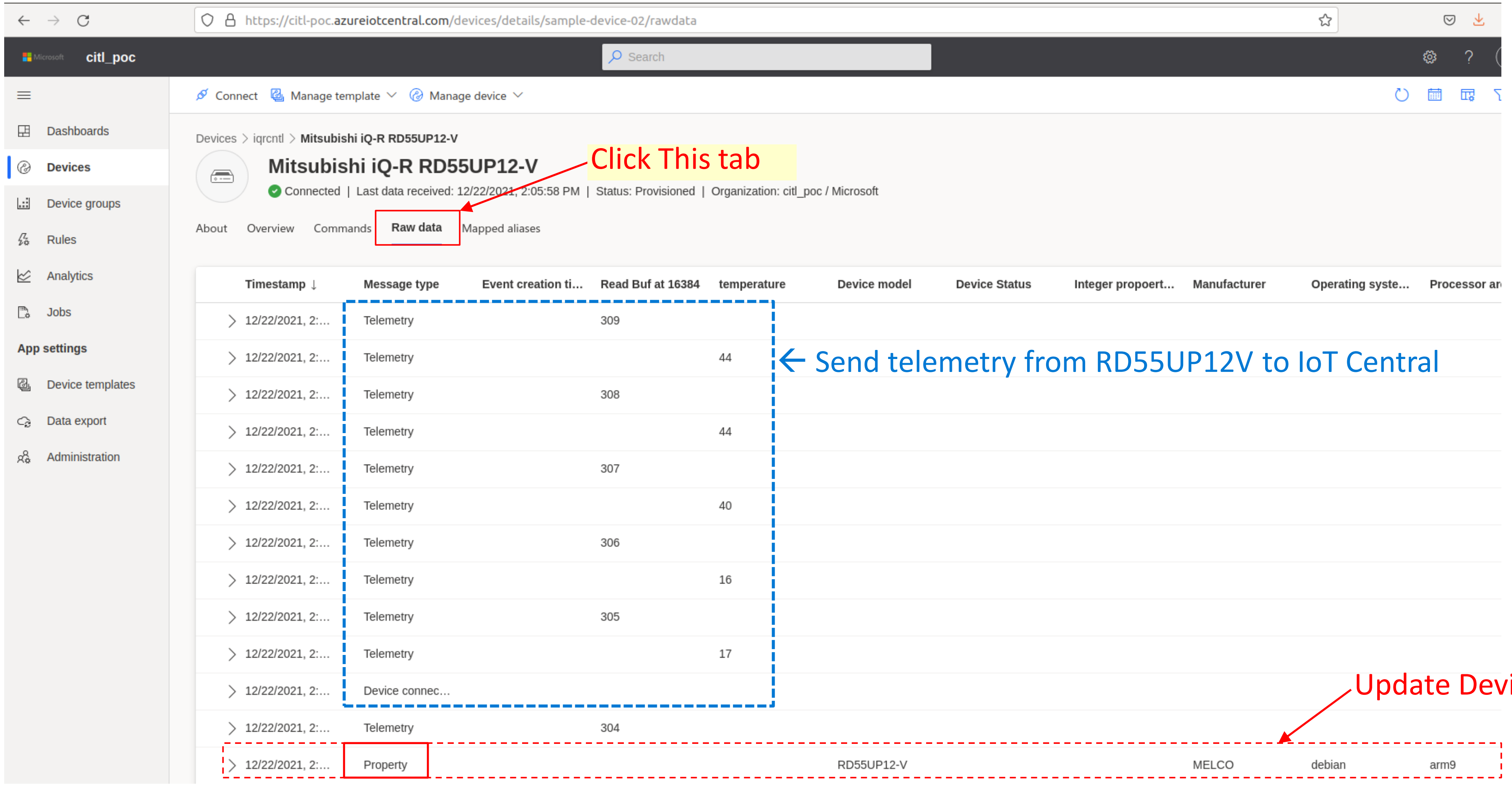
Device name	Device ID	Device status	Organization	Simu
Mitsubishi iQ...	sample-device...	Provisioned	...oc / Microsoft	No

For your info: (do not type "q" until end of demonstration)

How to stop Python program running on RD55UP12V

Type "q" from keyboard

IoT Central: Check if Python Program is Up and Running



Mitsubishi iQ-R RD55UP12-V

Connected | Last data received: 12/22/2021, 2:05:58 PM | Status: Provisioned | Organization: citl_poc / Microsoft

Click This tab

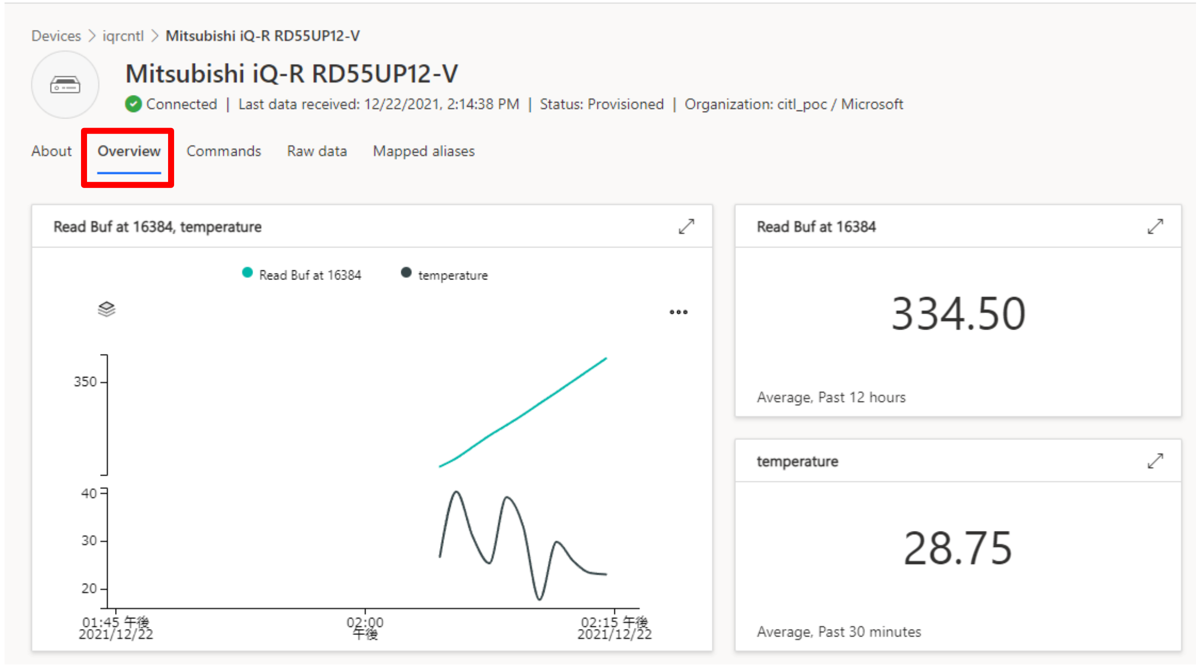
Timestamp ↓	Message type	Event creation ti...	Read Buf at 16384	temperature	Device model	Device Status	Integer propoert...	Manufacturer	Operating syste...	Processor ar
12/22/2021, 2:...	Telemetry		309							
12/22/2021, 2:...	Telemetry			44						
12/22/2021, 2:...	Telemetry		308							
12/22/2021, 2:...	Telemetry			44						
12/22/2021, 2:...	Telemetry		307							
12/22/2021, 2:...	Telemetry			40						
12/22/2021, 2:...	Telemetry		306							
12/22/2021, 2:...	Telemetry			16						
12/22/2021, 2:...	Telemetry		305							
12/22/2021, 2:...	Telemetry			17						
12/22/2021, 2:...	Device connec...									
12/22/2021, 2:...	Telemetry		304							
12/22/2021, 2:...	Property				RD55UP12-V			MELCO	debian	arm9

← Send telemetry from RD55UP12V to IoT Central

Update Device Information

IoT Central -- Visualization

Overview


Devices > iqrcntl > Mitsubishi iQ-R RD55UP12-V

Mitsubishi iQ-R RD55UP12-V

✓ Connected | Last data received: 12/22/2021, 2:14:38 PM | Status: Provisioned | Organization: citl_poc / Microsoft

Read Buf at 16384, temperature

Read Buf at 16384

334.50

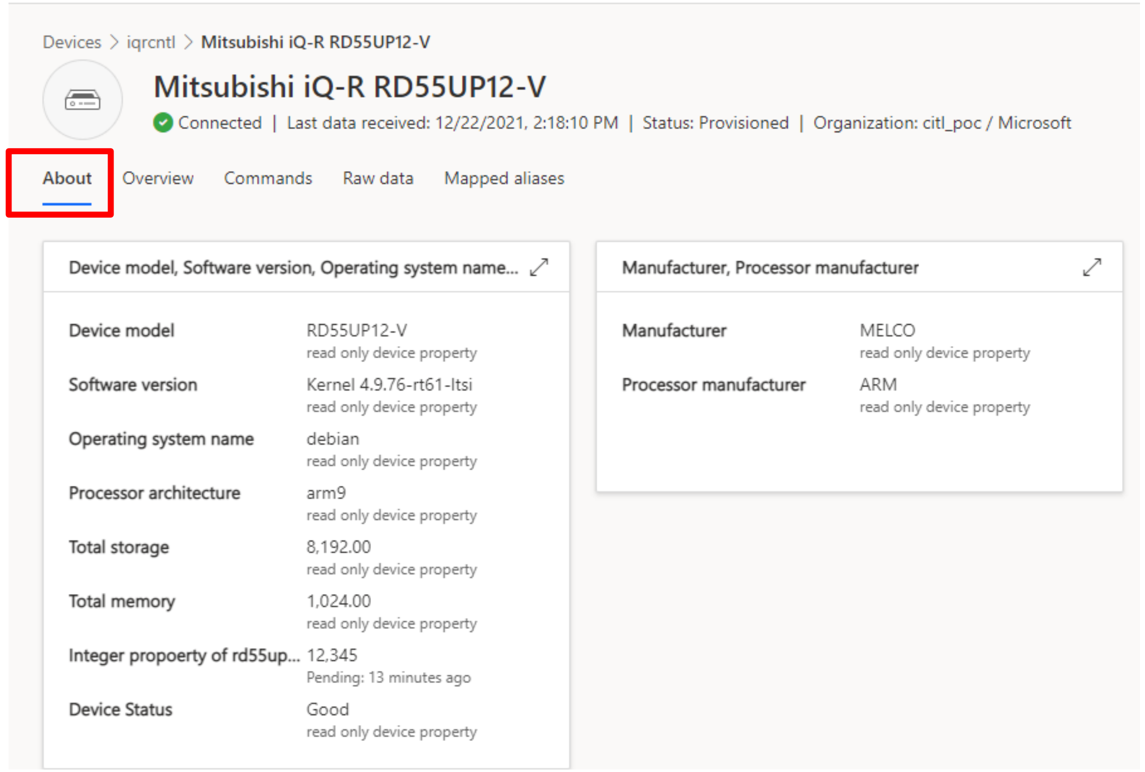
Average, Past 12 hours

temperature

28.75

Average, Past 30 minutes

About – Device Information


Devices > iqrcntl > Mitsubishi iQ-R RD55UP12-V

Mitsubishi iQ-R RD55UP12-V

✓ Connected | Last data received: 12/22/2021, 2:18:10 PM | Status: Provisioned | Organization: citl_poc / Microsoft

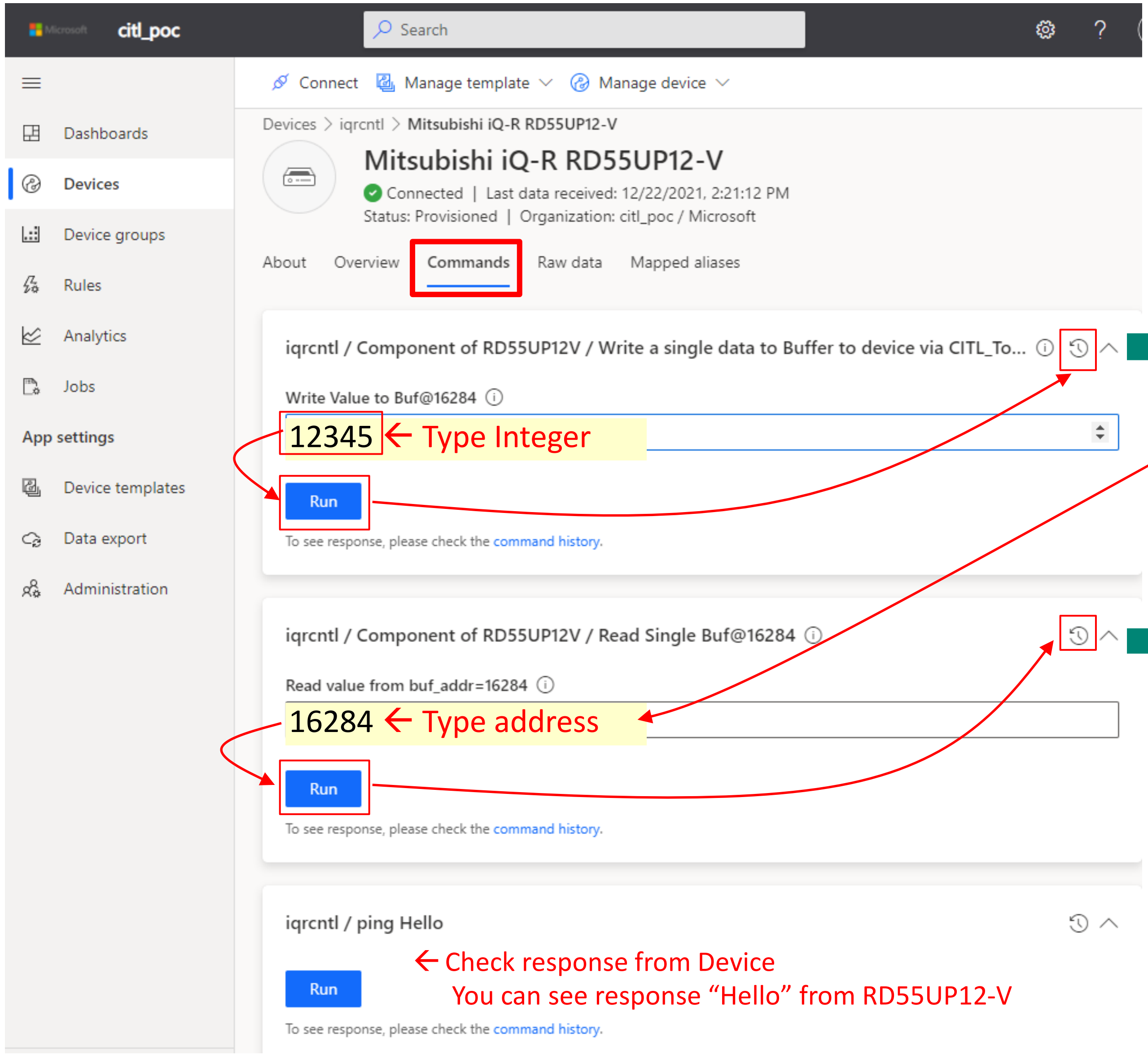
Device model, Software version, Operating system name...

Device model	RD55UP12-V read only device property
Software version	Kernel 4.9.76-rt61-ltsi read only device property
Operating system name	debian read only device property
Processor architecture	arm9 read only device property
Total storage	8,192.00 read only device property
Total memory	1,024.00 read only device property
Integer propoerty of rd55up...	12,345 Pending: 13 minutes ago
Device Status	Good read only device property

Manufacturer, Processor manufacturer

Manufacturer	MELCO read only device property
Processor manufacturer	ARM read only device property

IoT Central – Execute Direct Method



citl_poc

Search

Connect | Manage template | Manage device

Dashboards
Devices
Device groups
Rules
Analytics
Jobs
App settings
Device templates
Data export
Administration

Devices > iqrcntl > Mitsubishi iQ-R RD55UP12-V

Mitsubishi iQ-R RD55UP12-V

Connected | Last data received: 12/22/2021, 2:21:12 PM
Status: Provisioned | Organization: citl_poc / Microsoft

About | Overview | Commands | Raw data | Mapped aliases

iqrcntl / Component of RD55UP12V / Write a single data to Buffer to device via CITL_To...

Write Value to Buf@16284

12345 ← Type Integer

Run

To see response, please check the command history.

iqrcntl / Component of RD55UP12V / Read Single Buf@16284

Read value from buf_addr=16284

16284 ← Type address

Run

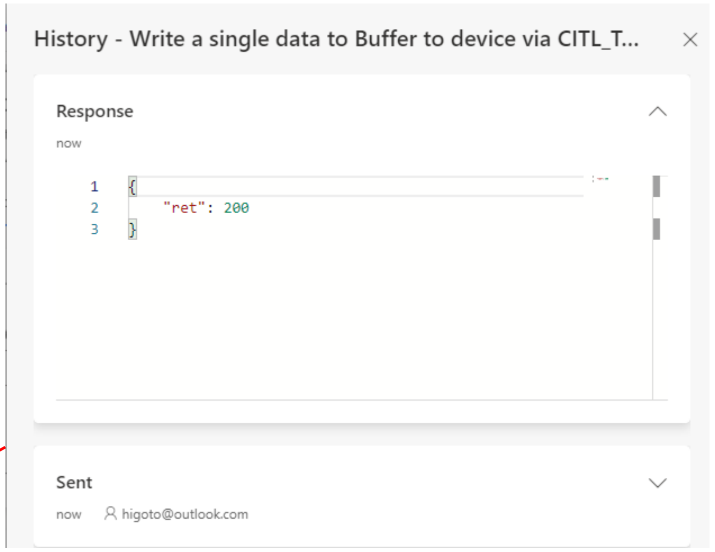
To see response, please check the command history.

iqrcntl / ping Hello

Run

← Check response from Device
You can see response "Hello" from RD55UP12-V

To see response, please check the command history.

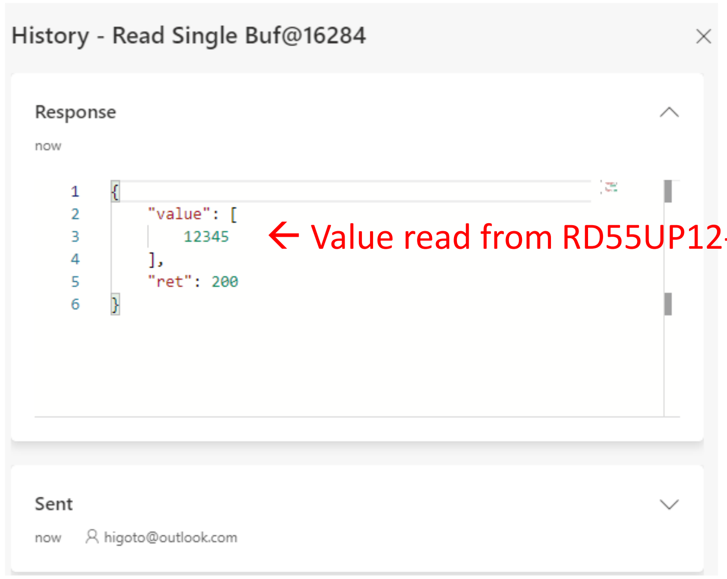


History - Write a single data to Buffer to device via CITL_T...

Response
now

```
{
    "ret": 200
}
```

Sent
now higoto@outlook.com



History - Read Single Buf@16284

Response
now

```
{
    "value": [
        12345
    ],
    "ret": 200
}
```

← Value read from RD55UP12-V@16284

Sent
now higoto@outlook.com

Microsoft