Contents

Building healthcare solutions with Azure IoT Central

2/4/2020 • 2 minutes to read • Edit Online

Learn to build healthcare solutions with Azure IoT Central using application templates.

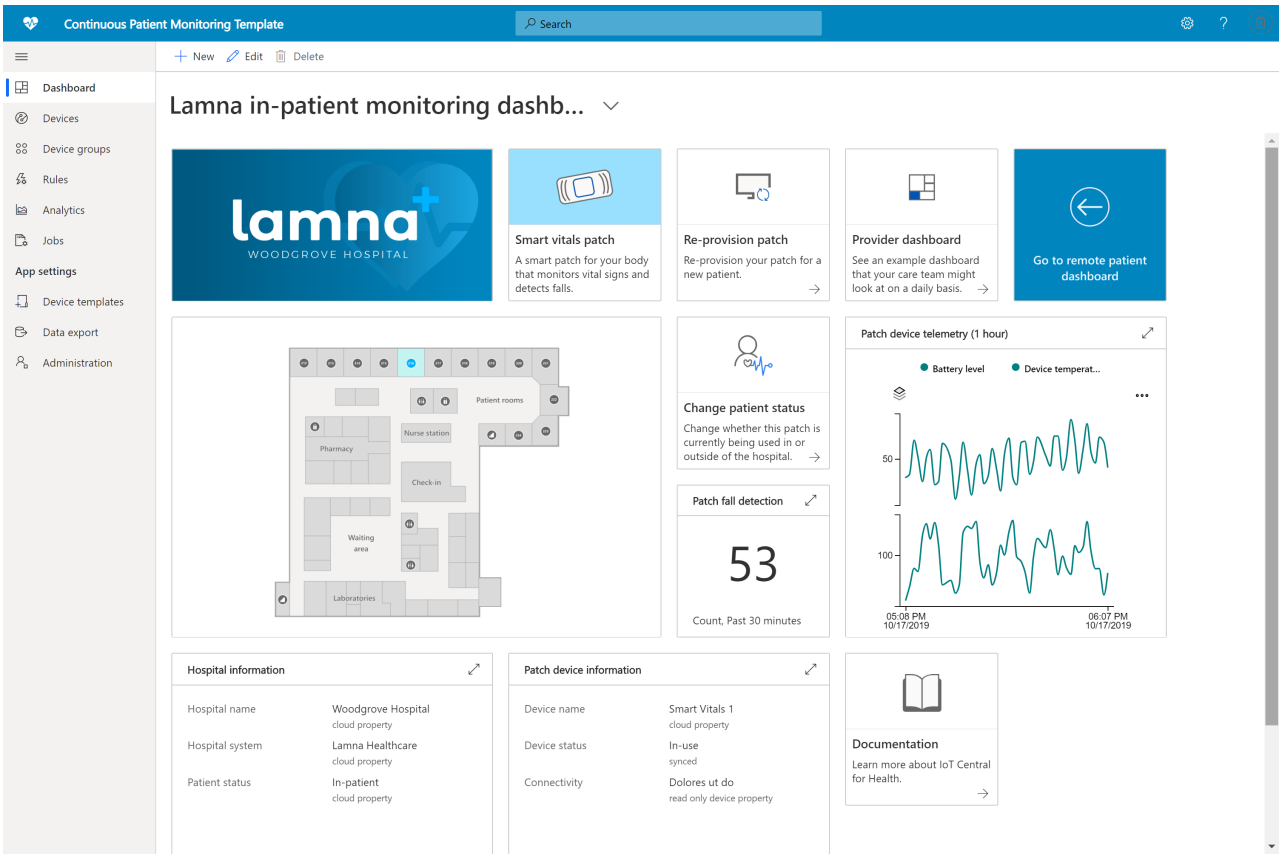
What is continuous patient monitoring template?

In the healthcare IoT space, Continuous Patient Monitoring is one of the key enablers of reducing the risk of readmissions, managing chronic diseases more effectively, and improving patient outcomes. Continuous Patient Monitoring can be split into two major categories:

1. **In-patient monitoring**: Using medical wearables and other devices in the hospital, care teams can monitor patient vital signs and medical conditions without having to send a nurse to check up on a patient multiple times a day. Care teams can understand the moment that a patient needs critical attention through notifications and prioritizes their time effectively.
2. **Remote patient monitoring**: By using medical wearables and patient reported outcomes (PROs) to monitor patients outside of the hospital, the risk of readmission can be lowered. Data from chronic disease patients and rehabilitation patients can be collected to ensure that patients are adhering to care plans and that alerts of patient deterioration can be surfaced to care teams before they become critical.

This application template can be used to build solutions for both categories of Continuous Patient Monitoring. The benefits include:

- Seamlessly connect different kinds of medical wearables to an IoT Central instance.
- Monitor and manage the devices to ensure they remain healthy.
- Create custom rules around device data to trigger appropriate alerts.
- Export your patient health data to the Azure API for FHIR, a compliant data store.
- Export the aggregated insights into existing or new business applications.



Next steps

To get started building a Continuous Patient monitoring solution:

- Deploy the application template
- See an example architecture

Tutorial: Deploy and walkthrough a continuous patient monitoring app template

2/4/2020 • 4 minutes to read • Edit Online

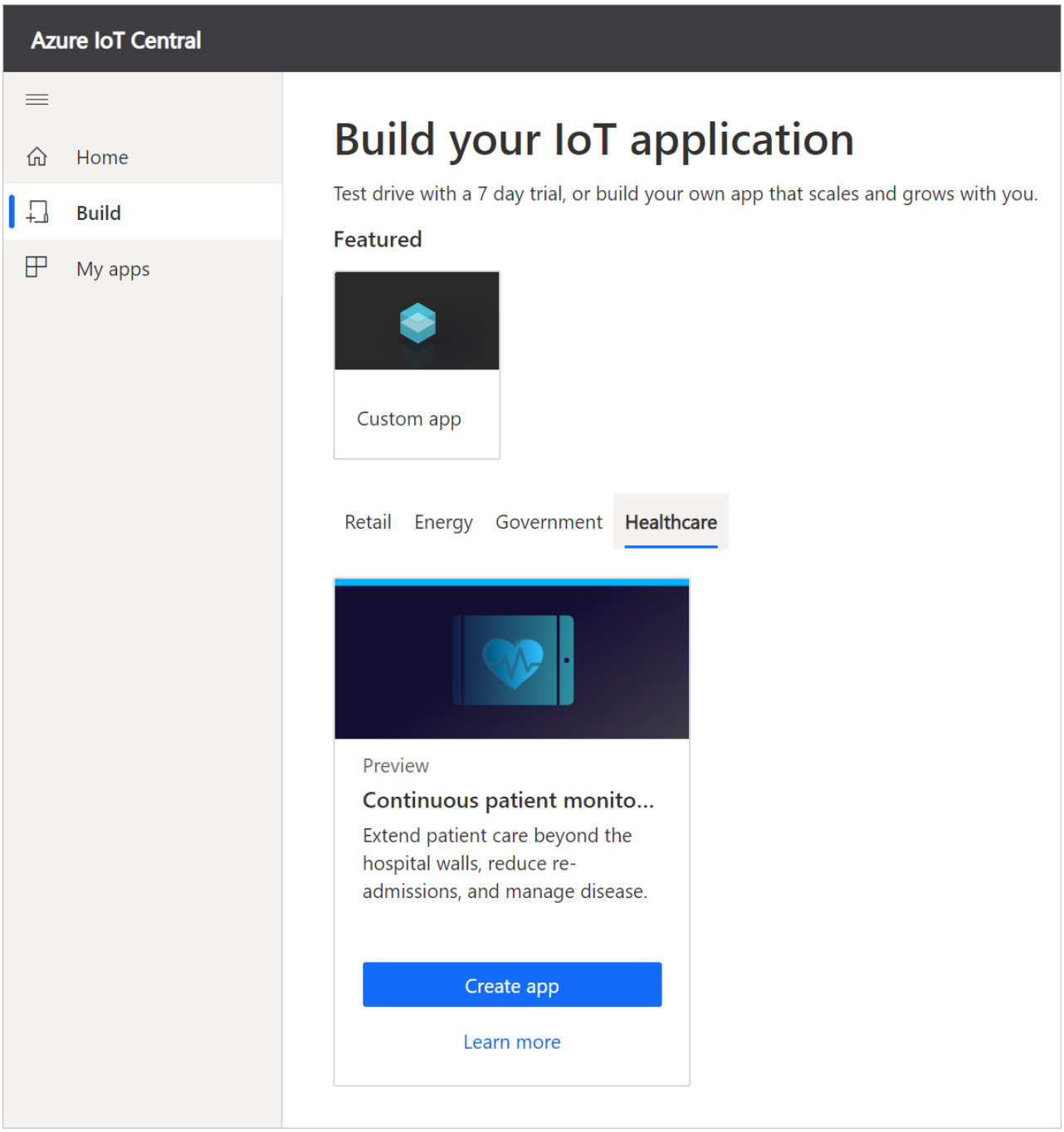
This tutorial shows you, as a solution builder, how to get started by deploying an IoT Central continuous patient monitoring application template. You will learn how to deploy the template, what's included out of the box, and what you can do next.

In this tutorial, you learn how to:

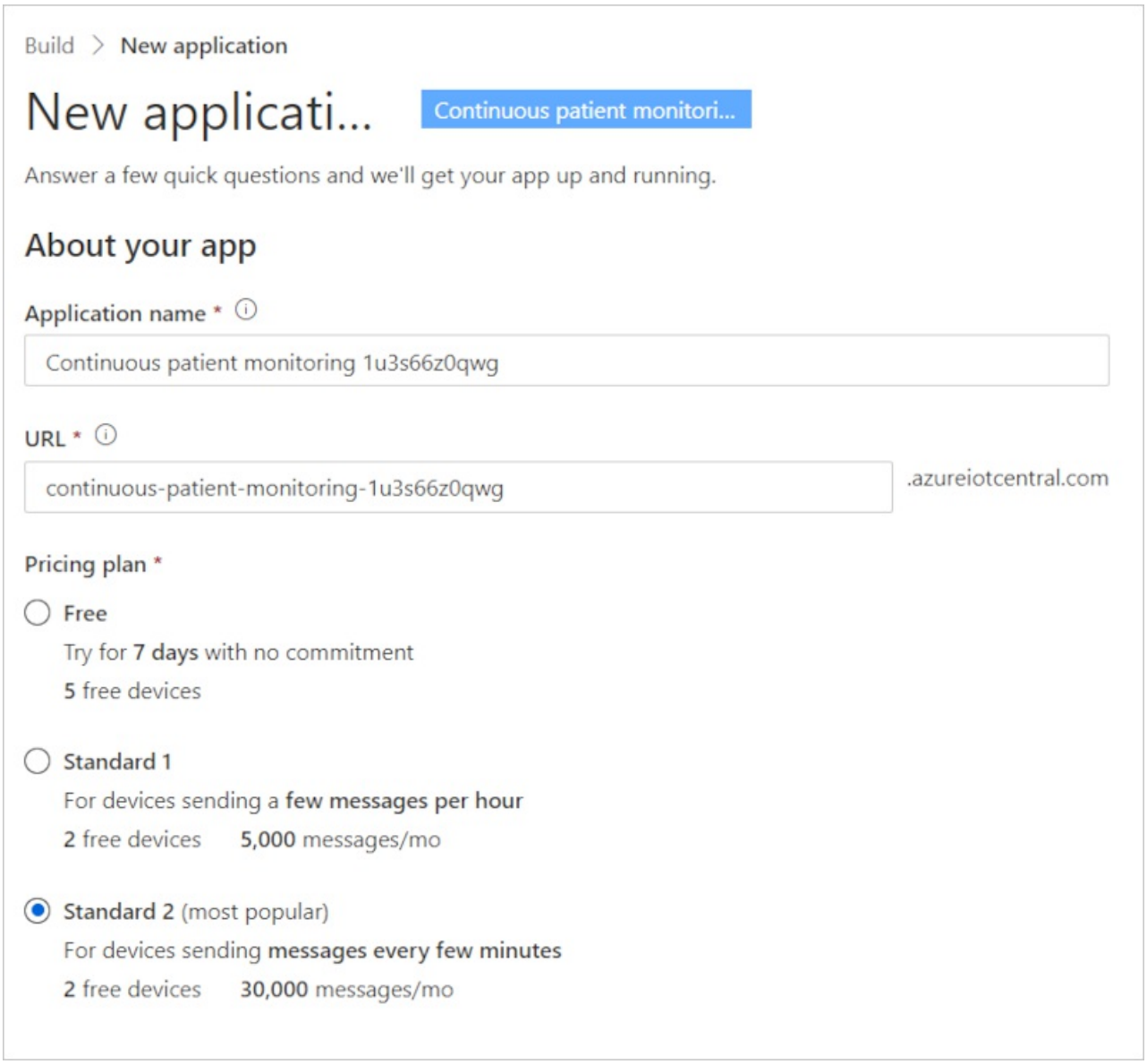
- Create an application template
- Walk through the application template

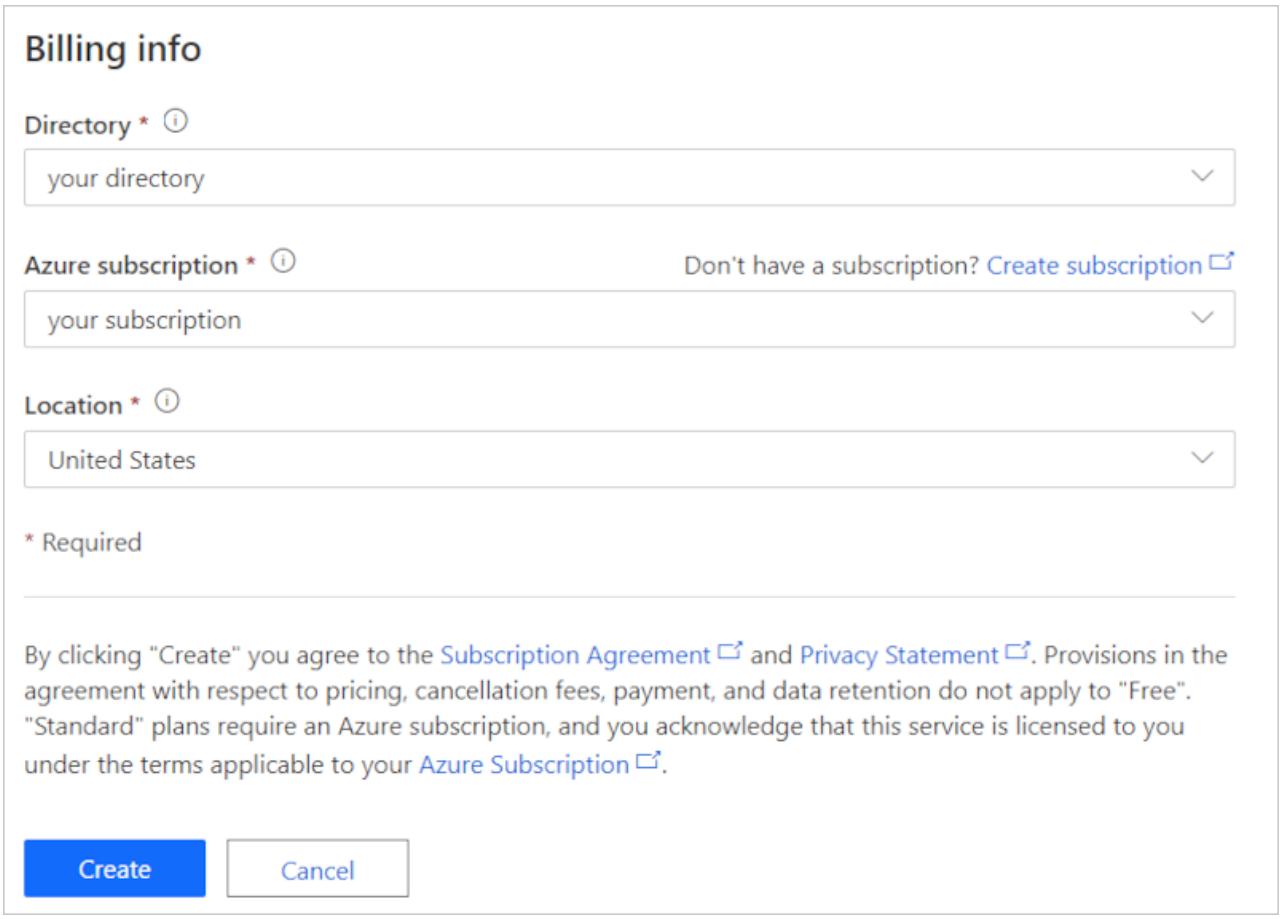
Create an application template

Navigate to the Azure IoT Central application manager website. Select **Build** from the left-hand navigation bar and then click the **Healthcare** tab.



Click the **Create app** button to begin creating your application and then sign in with a Microsoft personal, work, or school account. It will take you to the **New application** page.


Build > New application
New applicati...
Continuous patient monitori...
Answer a few quick questions and we'll get your app up and running.
About your app
Application name *
Continuous patient monitoring 1u3s66z0qwg
URL *
continuous-patient-monitoring-1u3s66z0qwg
.azureiotcentral.com
Pricing plan *
Free
Try for 7 days with no commitment
5 free devices
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



To create your application:

1. Azure IoT Central automatically suggests an application name based on the template you've selected. You can accept this name or enter your own friendly application name, such as **Continuous Patient Monitoring**. Azure IoT Central also generates a unique URL prefix for you based on the application name. You're free to change this URL prefix to something more memorable if you'd like.

2. You can select whether you would like to create the application using the *free* pricing plan or one of the *standard* pricing plans. Applications you create using the free plan are free for seven days before they expire and allow up to five free devices. You can move an application from the free plan to a standard pricing plan at any time before it expires. If you choose the free plan, you need to enter your contact information and choose whether to receive information and tips from Microsoft. Applications you create using a standard plan support up to two free devices and require you to enter your Azure subscription information for billing.

3. Select **Create** at the bottom of the page to deploy your application.

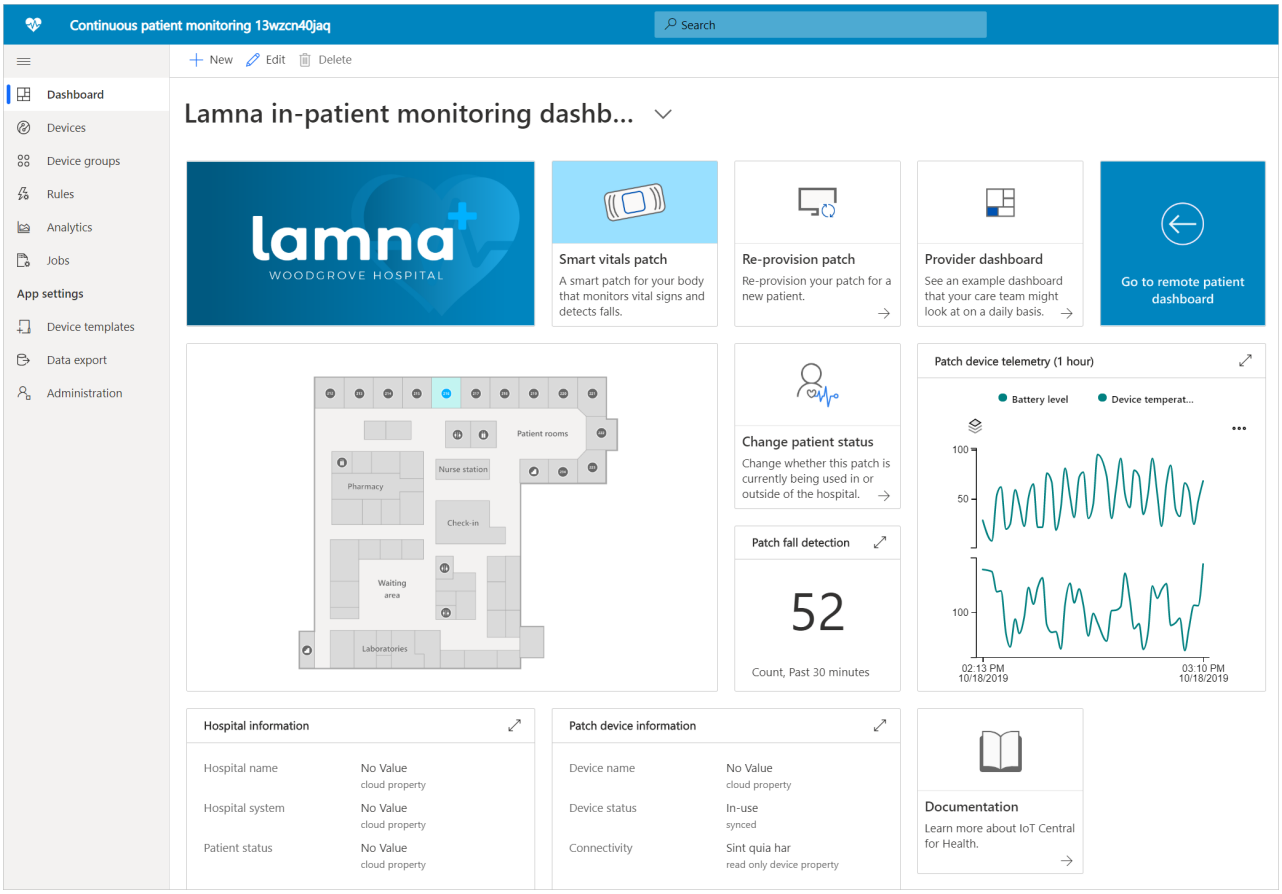
Walk through the application template

Dashboards

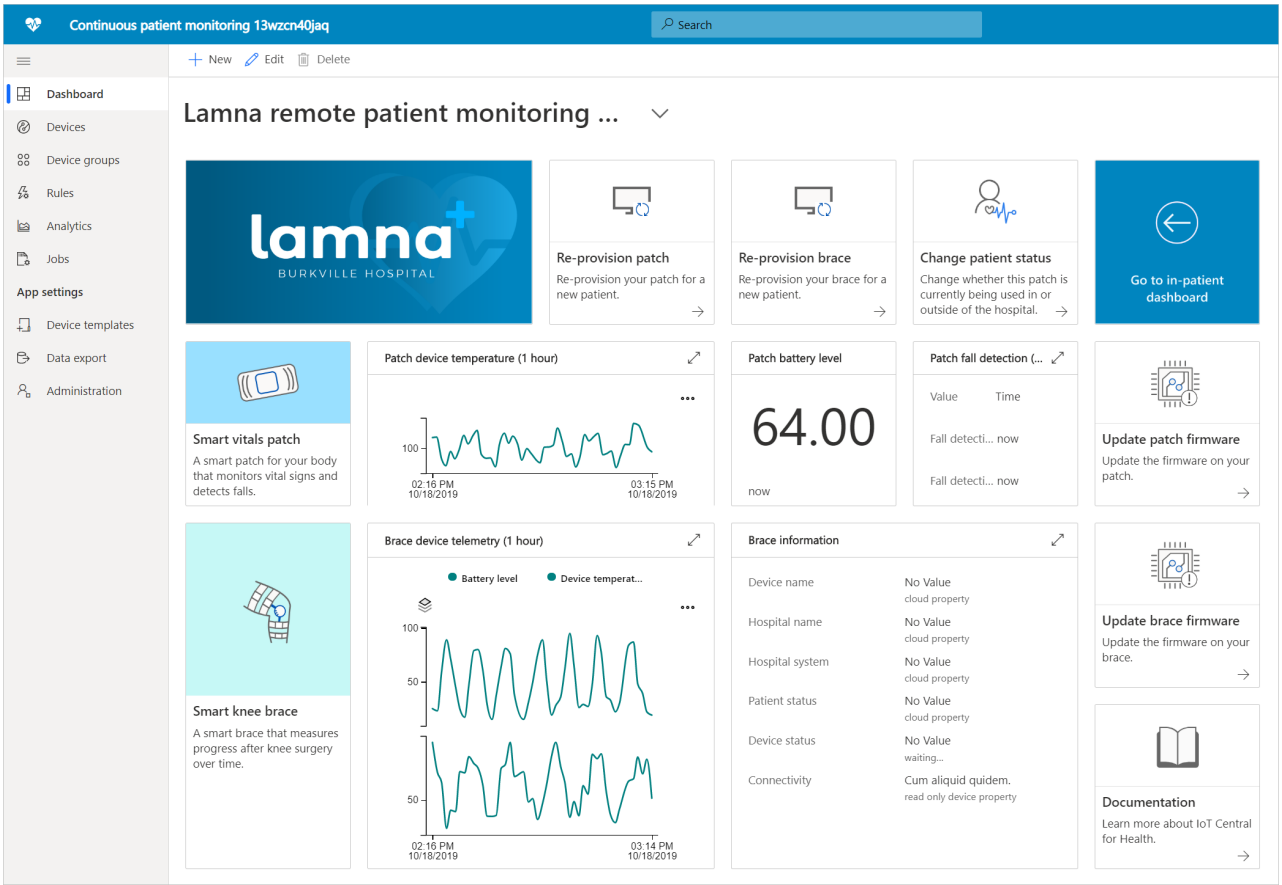
After deploying the app template, you'll first land on the **Lamna in-patient monitoring dashboard**. Lamna Healthcare is a fictitious hospital system that contains two hospitals: Woodgrove Hospital and Burkville Hospital. On this operator dashboard for Woodgrove Hospital, you'll see information and telemetry about the devices in this template along with a set of commands, jobs, and actions that you can take. From the dashboard you can:

- See device telemetry and properties such as the **battery level** of your device or its **connectivity** status.
- View the **floor plan** and location of the Smart Vitals Patch device.
- **Reprovision** the Smart Vitals Patch for a new patient.
- See an example of a **provider dashboard** that a hospital care team might see to track their patients.

- Change the **patient status** of your device to indicate if the device is being used for an in-patient or remote scenario.



You can also click on **Go to remote patient dashboard** to see the second operator dashboard used for Burkville Hospital. This dashboard contains a similar set of actions, telemetry, and information. In addition, you can see multiple devices being used and have the ability to **update the firmware** on each.

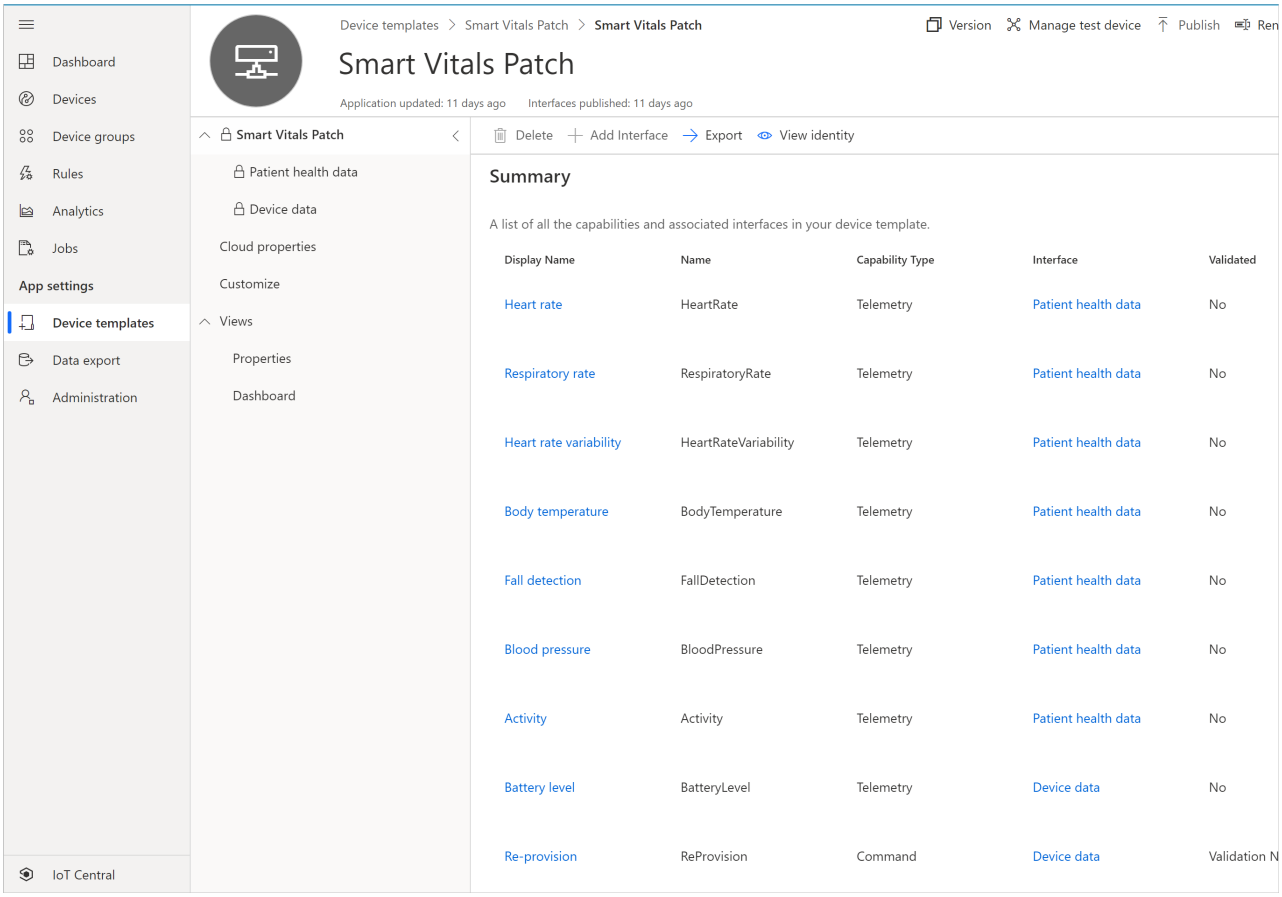


On both dashboards, you can always link back to this documentation.

Device templates

If you click on the **Device templates** tab, you will see that there are two different device types that are part of the template:

- **Smart Vitals Patch**: This device represents a patch that measures different types of vitals signs. It can be used for monitoring patients in and outside of the hospital. If you click on the template, you'll see that in addition to sending device data such as battery level and device temperature, the patch is also sending patient health data such as respiratory rate and blood pressure.
- **Smart Knee Brace**: This device represents a knee brace that patients might use when recovering from a knee replacement surgery. If you click on this template, you'll see capabilities such as range of motion and acceleration, in addition to device data.

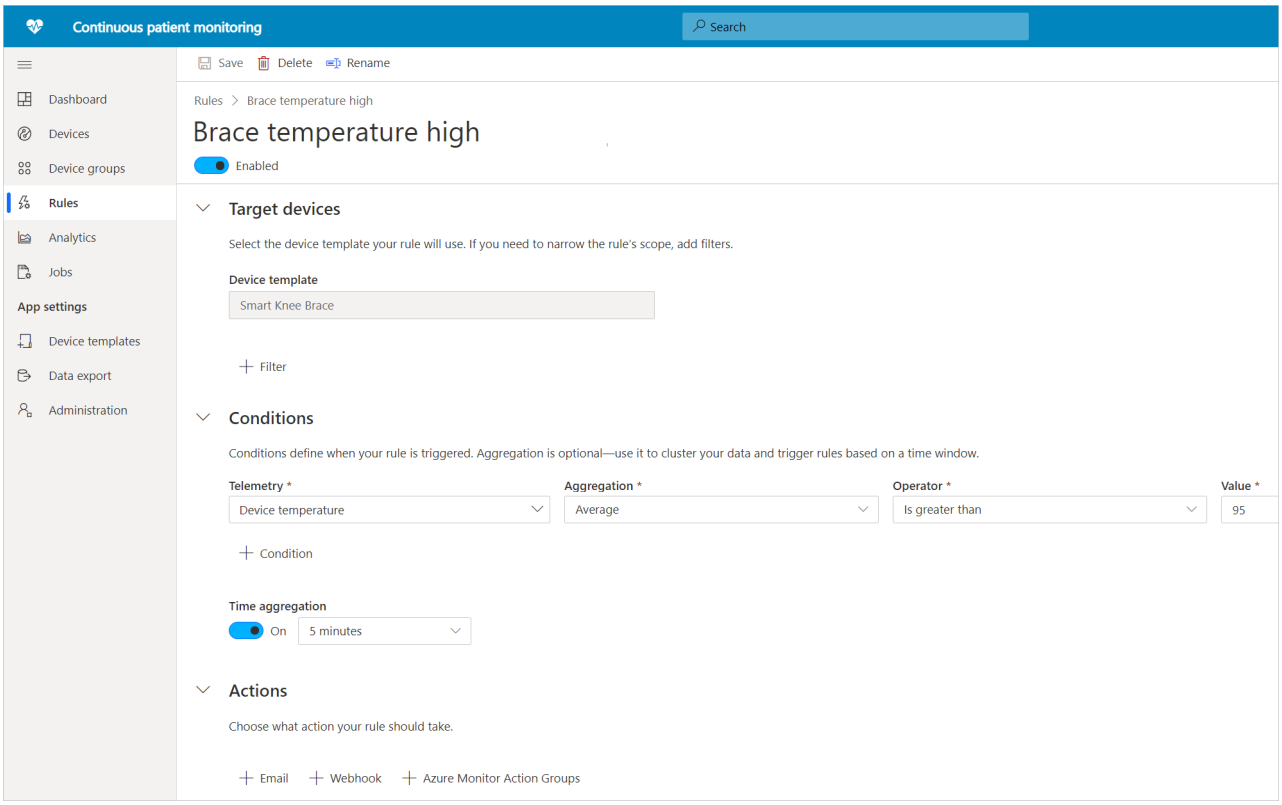


If you click on the **Device groups** tab, you will also see that these device templates automatically have device groups created for them.

Rules

When jumping to the rules tab, you will see three rules that exist in the application template:

- **Brace temperature high**: This rule is triggered when the device temperature of the Smart Knee Brace is greater than 95°F over a 5-minute window. You could use this rule to alert the patient and care team, and cool the device down remotely.
- **Fall detected**: This rule is triggered if a patient fall is detected. You could use this rule to configure an action to deploy an operational team to assist the patient who has fallen.
- **Patch battery low**: This rule is triggered when the battery level on the device goes below 10%. You could use this rule to trigger a notification to the patient to charge their device.



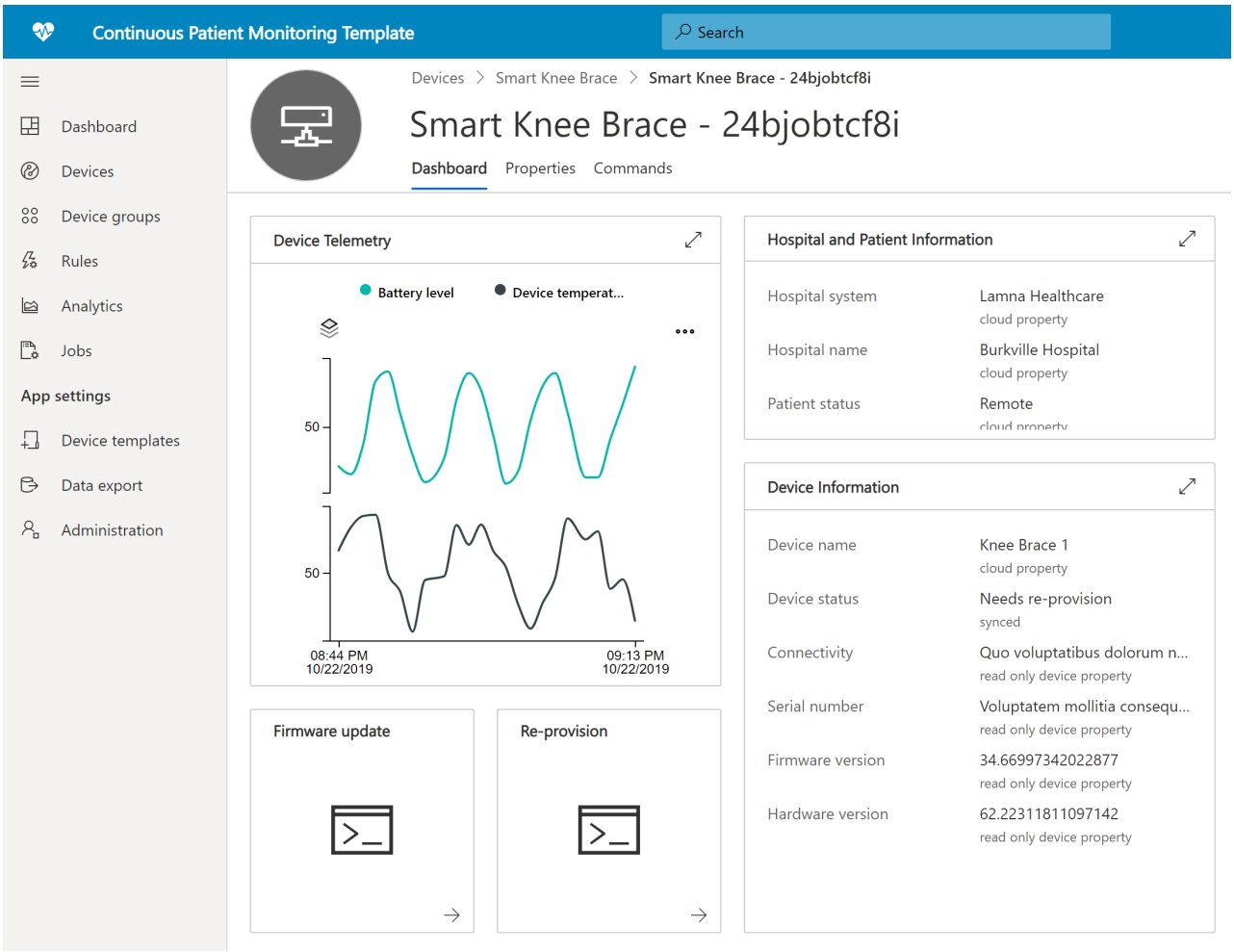
Devices

Click on the **Devices** tab and then select an instance of the **Smart Knee Brace**. You will see that there are three views to explore information about the particular device that you've selected. These views are created and published when building the device template for your device, which means they'll be consistent across all devices that you connect or simulate.

The **Dashboard** view gives an overview of telemetry and properties that are coming from the device that are operator-oriented.

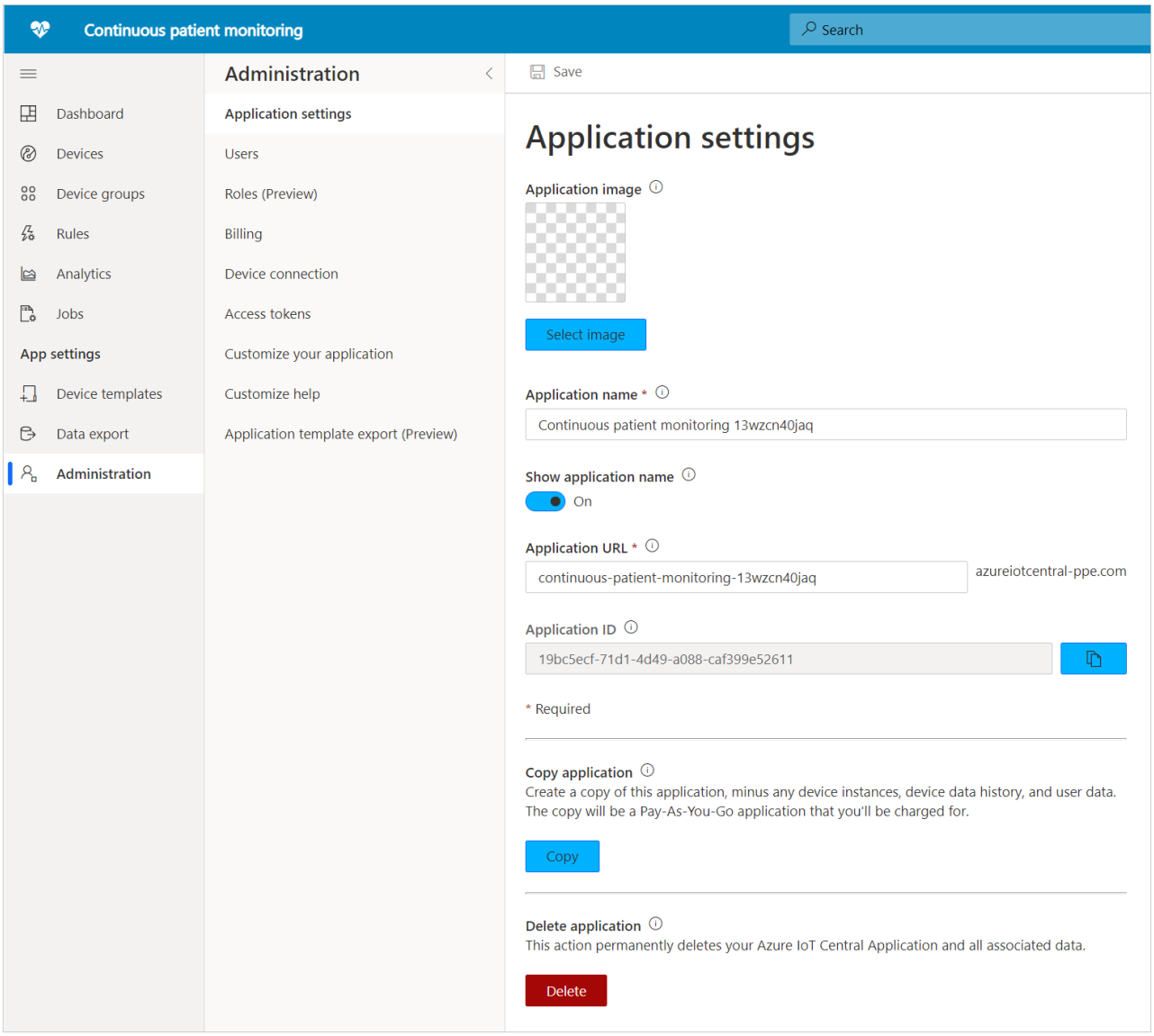
The **Properties** tab will allow you to edit cloud properties and read/write device properties.

The **Commands** tab will allow you to run commands that have been modeled as part of your device template.



Clean up resources

If you're not going to continue to use this application, delete the application by visiting **Administration** > **Application settings** and click **Delete**.



Next steps

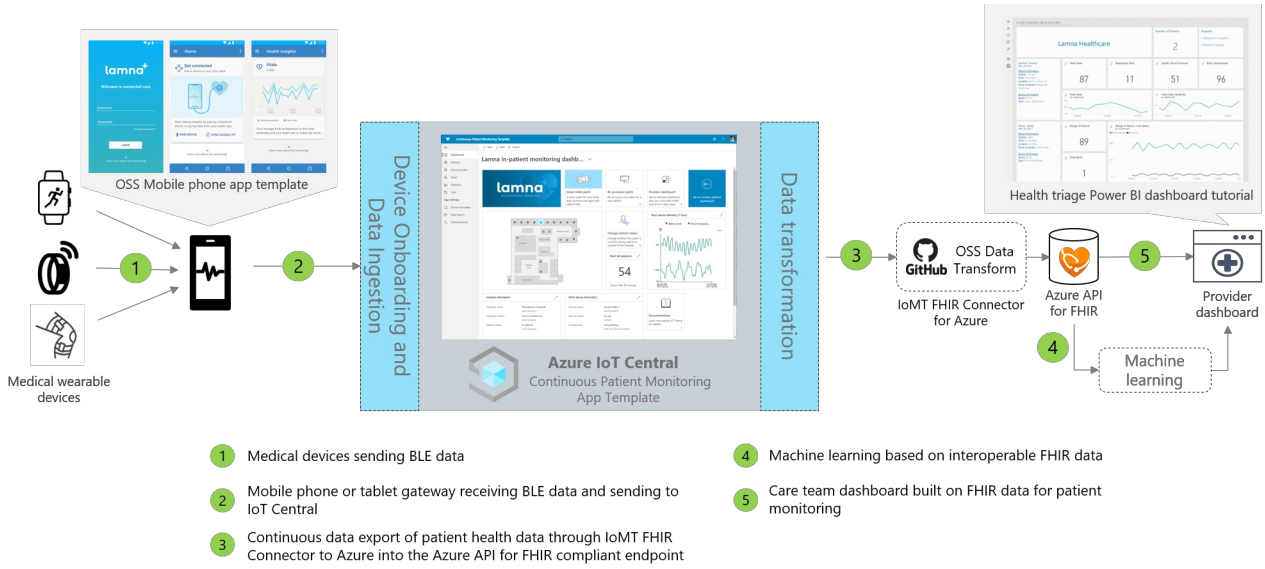
Advance to the next article to learn how to create a provider dashboard that connects to your IoT Central application.

Build a provider dashboard

Continuous patient monitoring architecture

2/4/2020 • 2 minutes to read • Edit Online

Continuous patient monitoring solutions can be built by using the app template provided and using the architecture that is outlined below as guidance.



1. Medical devices communicating using Bluetooth Low Energy (BLE)
2. Mobile phone gateway receiving BLE data and sending to IoT Central
3. Continuous data export of patient health data to the Azure API for FHIR®
4. Machine learning based on interoperable data
5. Care team dashboard built on FHIR data

Details

This section outlines each part of the architecture diagram in more detail.

BLE medical devices

Many medical wearables used in the healthcare IoT space are Bluetooth Low Energy devices. They're unable to speak directly to the cloud and will need to pass through a gateway. This architecture suggests using a mobile phone application as this gateway.

Mobile phone gateway

The mobile phone application's primary function is to ingest BLE data from medical devices and communicate it to Azure IoT Central. Additionally, the app can help guide patients through a device setup and provisioning flow and help them see a view of their personal health data. Other solutions can use a tablet gateway or a static gateway if inside a hospital room to achieve the same communication flow.

Export to Azure API for FHIR®

Azure IoT Central is HIPAA-compliant and HITRUST® certified, but you may also want to send patient health-related data to the Azure API for FHIR. Azure API for FHIR is a fully managed, standards-based, compliant API for clinical health data that enables you to create new systems of engagement with your health data. It enables rapid exchange of data through FHIR APIs, backed by a managed Platform-as-a Service (PaaS) offering in the cloud. Using the Continuous Data Export functionality of IoT Central, you can send data to the Azure API for FHIR.

Machine learning

After aggregating your data and translating it into FHIR format, you can build machine learning models that can enrich insights and enable smarter decision-making for your care team. There are different kinds of services that can be used to build, train, and deploy machine learning models. More information about how to use Azure's machine learning offerings can be found in our machine learning documentation.

Provider dashboard

The data located in the Azure API for FHIR can be used to build a patient insights dashboard or can be directly integrated into an EMR to help care teams visualize patient status. Care teams can use this dashboard to take care of patients in need of assistance and spot early warning signs of deterioration. To learn how to build a Power BI real-time provider dashboard, follow our how-to guide.

Next steps

- Learn how to deploy a continuous patient monitoring application template

Get connected to Azure IoT Central

3/16/2020 • 13 minutes to read • Edit Online

This article describes the options for connecting your devices to an Azure IoT Central application.

Typically, you must register a device in your application before it can connect. However, IoT Central does support scenarios where devices can connect without first being registered.

IoT Central uses the Azure IoT Hub Device Provisioning service (DPS) to manage the connection process. A device first connects to a DPS endpoint to retrieve the information it needs to connect to your application. Internally, your IoT Central application uses an IoT hub to handle device connectivity. Using DPS enables:

- IoT Central to support onboarding and connecting devices at scale.
- You to generate device credentials and configure the devices offline without registering the devices through IoT Central UI.
- You to use your own device IDs to register devices in IoT Central. Using your own device IDs simplifies integration with existing back-office systems.
- A single, consistent way to connect devices to IoT Central.

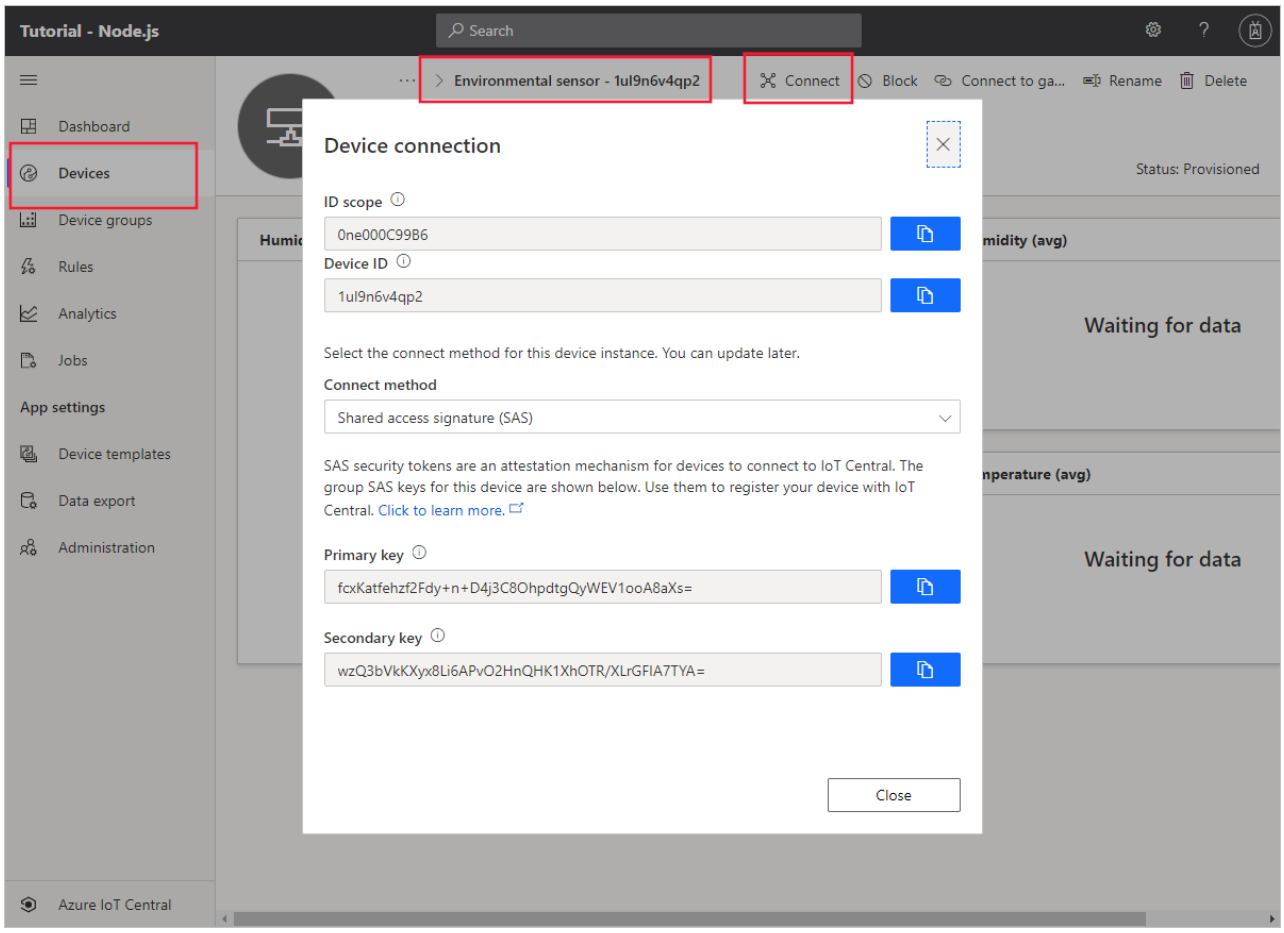
To secure the communication between a device and your application, IoT Central supports both shared access signatures (SAS) and X.509 certificates. X.509 certificates are recommended in production environments.

This article describes the following use cases:

- Connect a single device using SAS
- Connect devices at scale using SAS
- Connect devices at scale using X.509 certificates - the recommended approach for production environments.
- Connect devices without first registering them
- Connect devices that use DPS individual enrollments
- Connect devices using IoT Plug and Play (preview) features

Connect a single device

This approach is useful when you're experimenting with IoT Central or testing devices. You can use the device connection SAS keys from your IoT Central application to connect a device to your IoT Central application. Copy the *device SAS key* from the connection information for a registered device:



To learn more, see the Create and connect a Node.js client application to your Azure IoT Central application tutorial.

Connect devices at scale using SAS

To connect devices to IoT Central at scale using SAS keys, you need to register and then set up the devices:

Register devices in bulk

To register a large number of devices with your IoT Central application, use a CSV file to import device IDs and device names.

To retrieve the connection information for the imported devices, export a CSV file from your IoT Central application. The exported CSV file includes the device IDs and the SAS keys.

Set up your devices

Use the connection information from the export file in your device code to enable your devices to connect and send data to IoT to your IoT Central application. You also need the DPS **ID scope** for your application. You can find this value in **Administration > Device connection**.

> **NOTE**
> To learn how you can connect devices without first registering them in IoT Central, see Connect without first registering devices.

Connect devices using X.509 certificates

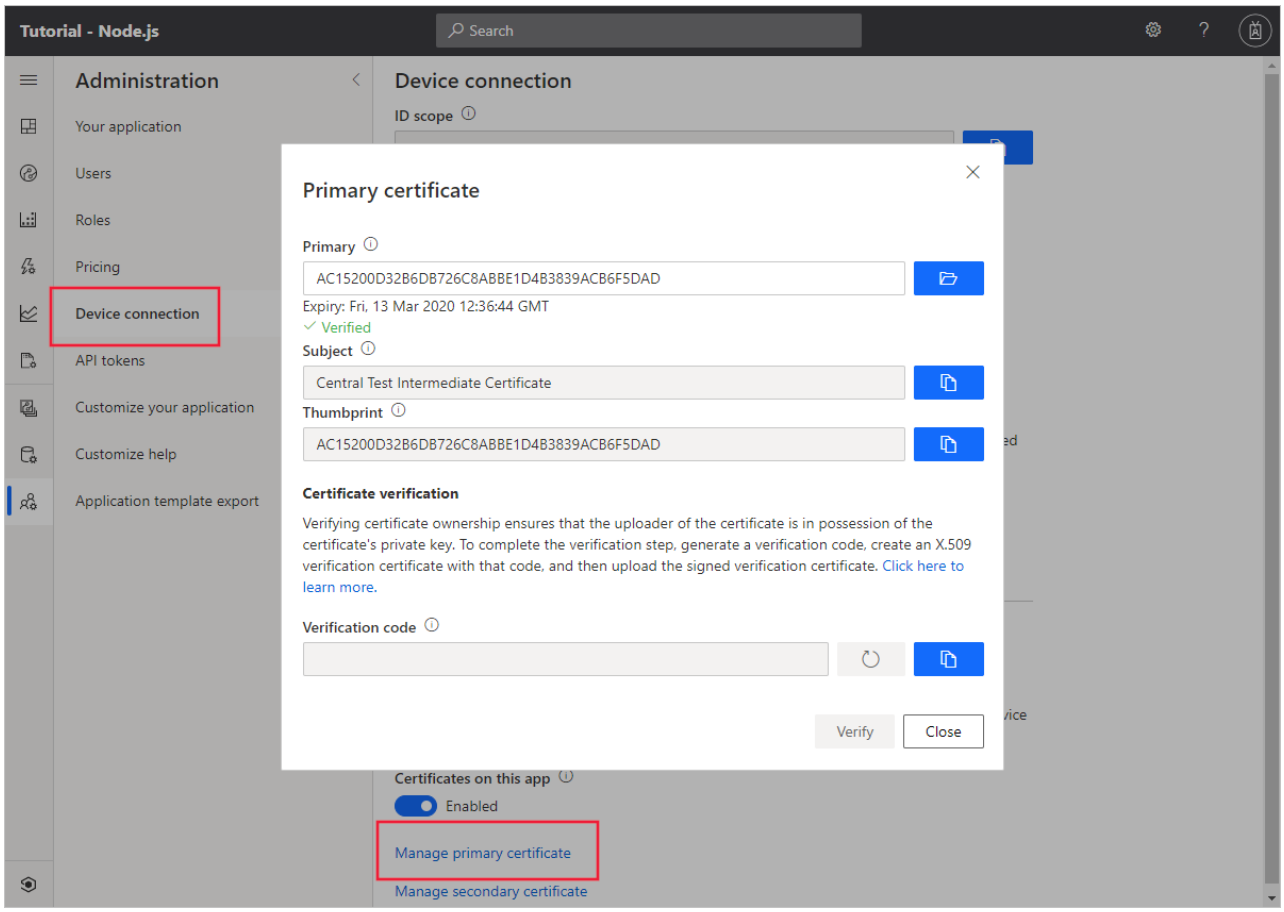
In a production environment, using X.509 certificates is the recommended device authentication mechanism for IoT Central. To learn more, see Device Authentication using X.509 CA Certificates.

Before you connect a device with an X.509 certificate, add and verify an intermediate or root X.509 certificate in

your application. Devices must use leaf X.509 certificates generated from the root or intermediate certificate.

Add and verify a root or intermediate certificate

Navigate to **Administration > Device Connection > Manage primary certificate** and add the X.509 root or intermediate certificate you're using to generate the device certificates.



Verifying certificate ownership ensures that the person uploading the certificate has the certificate's private key. To verify the certificate:

1. Select the button next to **Verification Code** to generate a code.
2. Create an X.509 verification certificate with the verification code you generated in the previous step. Save the certificate as a .cer file.
3. Upload the signed verification certificate and select **Verify**. The certificate is marked as **Verified** when the verification is successful.

If you have a security breach or your primary certificate is set to expire, use the secondary certificate to reduce downtime. You can continue to provision devices using the secondary certificate while you update the primary certificate.

Register and connect devices

To bulk connect devices using X.509 certificates, first register the devices in your application, by using a CSV file to import the device IDs and device names. The device IDs should all be in lower case.

Generate X.509 leaf certificates for your devices using the uploaded root or intermediate certificate. Use the **Device ID** as the `CNAME` value in the leaf certificates. Your device code needs the **ID scope** value for your application, the **device ID**, and the corresponding device certificate.

For testing purposes only

For testing only, you can use the following utilities to generate root, intermediate, and device certificates:

- Tools for the Azure IoT Device Provisioning Device SDK: a collection of Node.js tools that you can use to generate

and verify X.509 certificates and keys.

- If you're using a DevKit device, this command-line tool generates a CA certificate that you can add to your IoT Central application to verify the certificates.
- Manage test CA certificates for samples and tutorials: a collection of PowerShell and Bash scripts to:
  - Create a certificate chain.
  - Save the certificates as .cer files to upload to your IoT Central application.
  - Use the verification code from the IoT Central application to generate the verification certificate.
  - Create leaf certificates for your devices using your device IDs as a parameter to the tool.

Further reference

- Sample implementation for RaspberryPi
- Sample device client in C

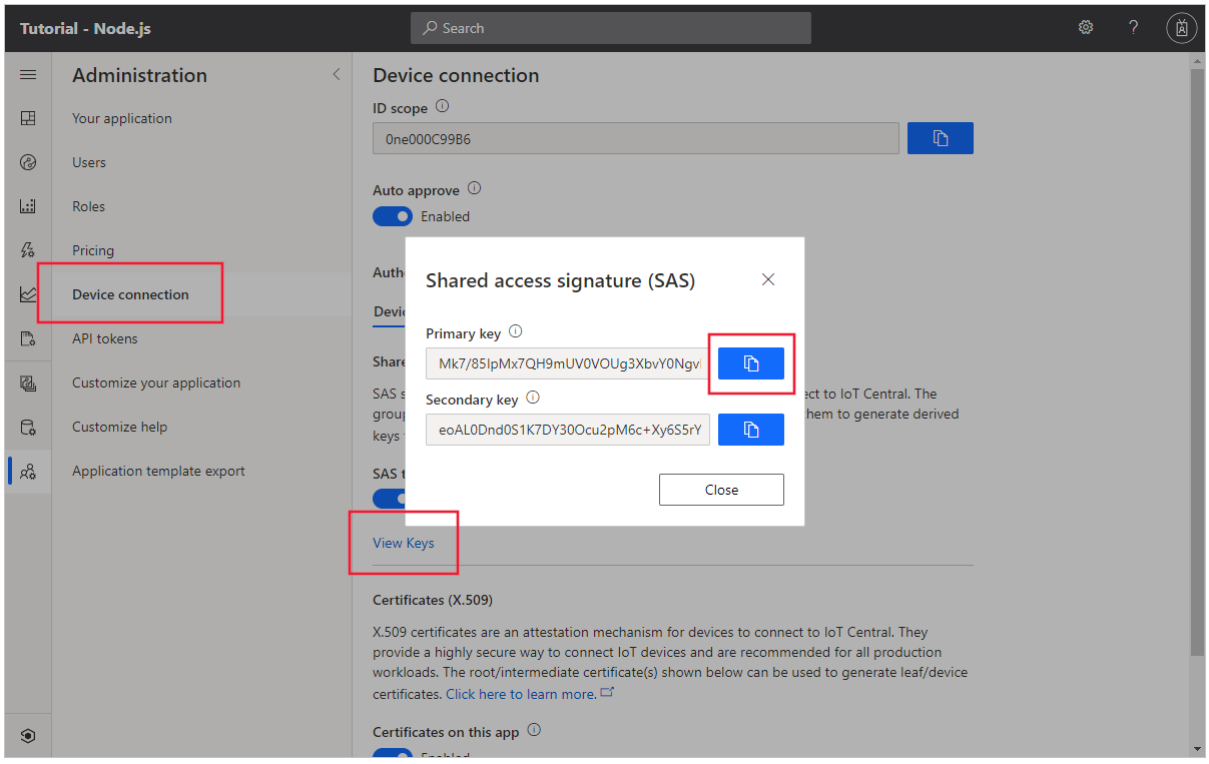
Connect without registering devices

The previously described scenarios all require you to register devices in your application before they connect. IoT Central also enables OEMs to mass manufacture devices that can connect without first being registered. An OEM generates suitable device credentials, and configures the devices in the factory. When a customer turns on a device for the first time, it connects to DPS, which then automatically connects the device to the correct IoT Central application. An IoT Central operator must approve the device before it starts sending data to the application.

The flow is slightly different depending on whether the devices use SAS tokens or X.509 certificates:

Connect devices that use SAS tokens without registering

1. Copy the IoT Central application's group primary key:



2. Use the dps-keygen tool to generate the device SAS keys. Use the group primary key from the previous step. The device IDs must be lower-case:

```
dps-keygen -mk:<group primary key> -di:<device ID>
```

3. The OEM flashes each device with a device ID, a generated device SAS key, and the application **ID scope** value.

4. When you switch on a device, it first connects to DPS to retrieve its IoT Central registration information.

   The device initially has a device status **Unassociated** on the **Devices** page and isn't assigned to a device template. On the **Devices** page, **Migrate** the device to the appropriate device template. Device provisioning is now complete, the device status is now **Provisioned**, and the device can start sending data.

   On the **Administration > Device connection** page, the **Auto approve** option controls whether you need to manually approve the device before it can start sending data.

   > **NOTE**
   > To learn how automatically associate a device with a device template, see Connect devices with IoT Plug and Play (preview).

Connect devices that use X.509 certificates without registering

1. Add and verify a root or intermediate X.509 certificate to your IoT Central application.(#connect-devices-using-x509-certificates)
2. Generate the leaf-certificates for your devices using the root or intermediate certificate you added to your IoT Central application. Use lower-case device IDs as the `CNAME` in the leaf certificates.
3. The OEM flashes each device with a device ID, a generated left X.509 certificate, and the application **ID scope** value.
4. When you switch on a device, it first connects to DPS to retrieve its IoT Central registration information.

   The device initially has a device status **Unassociated** on the **Devices** page and isn't assigned to a device template. On the **Devices** page, **Migrate** the device to the appropriate device template. Device provisioning is now complete, the device status is now **Provisioned**, and the device can start sending data.

   On the **Administration > Device connection** page, the **Auto approve** option controls whether you need to manually approve the device before it can start sending data.

   > **NOTE**
   > To learn how automatically associate a device with a device template, see Connect devices with IoT Plug and Play (preview).

Individual enrollment-based device connectivity

For customers connecting devices that each have their own authentication credentials, use individual enrollments. An individual enrollment is an entry for a single device that is allowed to connect. Individual enrollments can use either X.509 leaf certificates or SAS tokens (from a physical or virtual trusted platform module) as attestation mechanisms. The device ID (also known as registration ID) in an individual enrollment is alphanumeric, lowercase, and may contain hyphens. For more information, see DPS individual enrollment.

> **NOTE**
> When you create an individual enrollment for a device, it takes precedence over the default group enrollment options in your IoT Central application.

Creating individual enrollments

IoT Central supports the following attestation mechanisms for individual enrollments:

- **Symmetric key attestation:** Symmetric key attestation is a simple approach to authenticating a device

with the DPS instance. To create an individual enrollment that uses symmetric keys, open the **Device Connection** page, select **Individual enrollment** as the connection method, and **Shared access signature (SAS)** as the mechanism. Enter base64 encoded primary and secondary keys, and save your changes. Use the **ID scope**, **Device ID**, and either the primary or secondary key to connect your device.

> **TIP**
> For testing, you can use **OpenSSL** to generate base64 encoded keys: `openssl rand -base64 64`

- **X.509 certificates:** To create an individual enrollment with X.509 certificates, open the **Device Connection** page, select **Individual enrollment** as the connection method, and **Certificates (X.509)** as the mechanism. Device certificates used with an individual enrollment entry have a requirement that the issuer and subject CN are set to the device ID.

  > **TIP**
  > For testing, you can use Tools for the Azure IoT Device Provisioning Device SDK for Node.js to generate a self-signed certificate: `node create_test_cert.js device "mytestdevice"`

- **Trusted Platform Module (TPM) attestation:** A TPM is a type of hardware security module. Using a TPM is one of the most secure ways to connect a device. This article assumes you're using a discrete, firmware, or integrated TPM. Software emulated TPMs are well suited for prototyping or testing, but they don't provide the same level of security as discrete, firmware, or integrated TPMs. Don't use software TPMs in production. To create an individual enrollment that uses a TPM, open the **Device Connection** page, select **Individual enrollment** as the connection method, and **TPM** as the mechanism. Enter the TPM endorsement key and save the device connection information.

Connect devices with IoT Plug and Play (preview)

One of the key features of IoT Plug and Play (preview) with IoT Central is the ability to associate device templates automatically on device connection. Along with device credentials, devices can now send the **CapabilityModelId** as part of the device registration call. This capability enables IoT Central to discover and associate the device template with the device. The discovery process works as follows:

1. Associates with the device template if it's already published in the IoT Central application.
2. Fetches from the public repository of published and certified capability models.

Below is the format of the additional payload the device would send during the DPS registration call

```
'__iot:interfaces': {
    CapabilityModelId: <this is the URN for the capability model>
}
```

> **NOTE**
> Note that the **Auto approve** option on **Administration** > **Device connection** must be enabled for devices to automatically connect, discover the device template, and start sending data.

Device status values

When a real device connects to your IoT Central application, its device status changes as follows:

1. The device status is first **Registered**. This status means the device is created in IoT Central, and has a device ID. A device is registered when:

   - A new real device is added on the **Devices** page.
   - A set of devices is added using **Import** on the **Devices** page.
2. The device status changes to **Provisioned** when the device that connected to your IoT Central application with valid credentials completes the provisioning step. In this step, the device uses DPS to automatically retrieve a connection string from the IoT Hub used by your IoT Central application. The device can now connect to IoT Central and start sending data.

3. An operator can block a device. When a device is blocked, it can't send data to your IoT Central application. Blocked devices have a status of **Blocked**. An operator must reset the device before it can resume sending data. When an operator unblocks a device the status returns to its previous value, **Registered** or **Provisioned**.

4. If the device status is **Waiting for Approval**, it means the **Auto approve** option is disabled. An operator must explicitly approve a device before it starts sending data. Devices not registered manually on the **Devices** page, but connected with valid credentials will have the device status **Waiting for Approval**. Operators can approve these devices from the **Devices** page using the **Approve** button.

5. If the device status is **Unassociated**, it means the device connecting to IoT Central doesn't have an associated device template. This situation typically happens in the following scenarios:

   - A set of devices is added using **Import** on the **Devices** page without specifying the device template.
   - A device was registered manually on the **Devices** page without specifying the device template. The device then connected with valid credentials.

   The Operator can associate a device to a device template from the **Devices** page using the **Migrate** button.

Best practices

Don't persist or cache the device connection string that DPS returns when you first connect the device. To reconnect a device, go through the standard device registration flow to get the correct device connection string. If the device caches the connection string, the device software runs into the risk of having a stale connection string if IoT Central updates the underlying Azure IoT hub it uses.

SDK support

The Azure Device SDKs offer the easiest way for you implement your device code. The following device SDKs are available:

- Azure IoT SDK for C
- Azure IoT SDK for Python
- Azure IoT SDK for Node.js
- Azure IoT SDK for Java
- Azure IoT SDK for .NET

SDK features and IoT Hub connectivity

All device communication with IoT Hub uses the following IoT Hub connectivity options:

- Device-to-cloud messaging
- Device twins

The following table summarizes how Azure IoT Central device features map on to IoT Hub features:

AZURE IOT CENTRAL	AZURE IOT HUB
Telemetry	Device-to-cloud messaging
Property	Device twin reported properties
Property (writeable)	Device twin desired and reported properties
Command	Direct methods

To learn more about using the Device SDKs, see Connect a DevDiv kit device to your Azure IoT Central application for example code.

Protocols

The Device SDKs support the following network protocols for connecting to an IoT hub:

- MQTT
- AMQP
- HTTPS

For information about these difference protocols and guidance on choosing one, see Choose a communication protocol.

If your device can't use any of the supported protocols, you can use Azure IoT Edge to do protocol conversion. IoT Edge supports other intelligence-on-the-edge scenarios to offload processing to the edge from the Azure IoT Central application.

Security

All data exchanged between devices and your Azure IoT Central is encrypted. IoT Hub authenticates every request from a device that connects to any of the device-facing IoT Hub endpoints. To avoid exchanging credentials over the wire, a device uses signed tokens to authenticate. For more information, see, Control access to IoT Hub.

Next steps

Now that you've learned about device connectivity in Azure IoT Central, here are the suggested next steps:

- Prepare and connect a DevKit device
- C SDK: Provisioning Device Client SDK

What are application templates?

2/4/2020 • 2 minutes to read • Edit Online

Application templates in Azure IoT Central are a tool to help solution builders kickstart their IoT solution development. You can use app templates for everything from getting a feel for what is possible, to fully customizing and your application for resale to your customers.

Application templates consist of:

- Sample operator dashboards
- Sample device templates
- Simulated devices producing real-time data
- Pre-configured rules and jobs
- Rich documentation including tutorials and how-tos

You choose the application template when you create your application. You can't change the template after the application is created.

Custom templates

If you want to create your application from scratch, choose one of the two custom application templates:

- Custom application
- Custom application (legacy)

Choose the **Custom application** template unless you have a specific reason to use the legacy template.

Industry focused templates

Azure IoT Central is an industry agnostic application platform. Application templates are industry focused examples available for these industries today, with more to come in the future:

- Retail
  - Connected logistics
  - Digital distribution center
  - In-store analytics - condition monitoring
  - In-store analytics - checkout
  - Smart Inventory Management
- Energy
  - Smart meter monitoring
  - Solar panel monitoring
- Government
  - Connected waste management
  - Water consumption monitoring
  - Water quality monitoring
- Healthcare.
  - Continuous patient monitoring

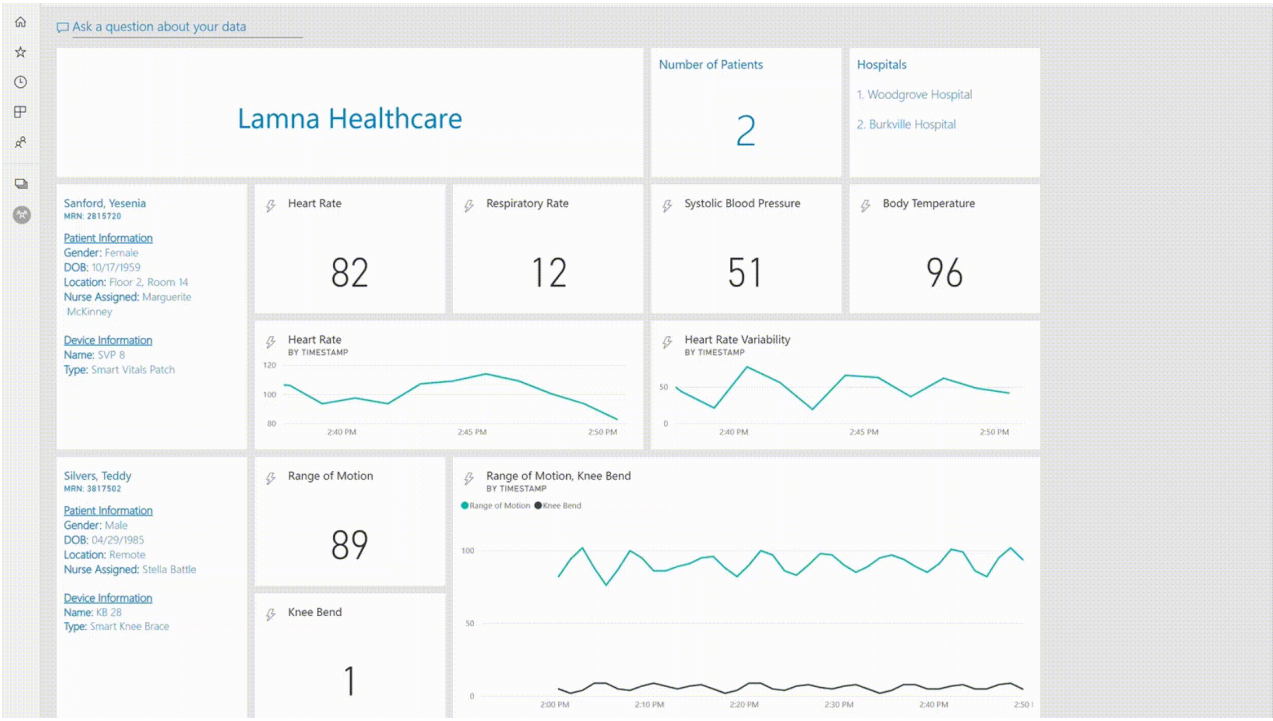
Next steps

Now that you know what IoT Central application templates are, get started by creating an IoT Central Application.

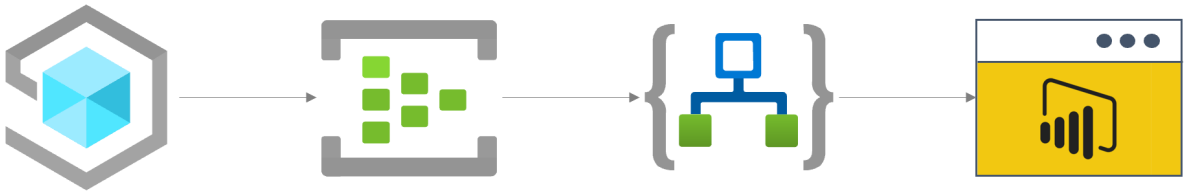
Tutorial: Build a Power BI provider dashboard

2/4/2020 • 6 minutes to read • Edit Online

When building your continuous patient monitoring solution, you can also create a dashboard for a hospital care team to visualize patient data. In this tutorial, you will learn how to create a Power BI real-time streaming dashboard from your IoT Central continuous patient monitoring application template.



The basic architecture will follow this structure:



In this tutorial, you learn how to:

- Export data from Azure IoT Central to Azure Event Hubs
- Set up a Power BI streaming dataset
- Connect your Logic App to Azure Event Hubs
- Stream data to Power BI from your Logic App
- Build a real-time dashboard for patient vitals

Prerequisites

- An Azure subscription. If you don't have an Azure subscription, sign up for a free Azure account.
- An Azure IoT Central continuous patient monitoring application template. If you don't have one already, you

can follow steps to Deploy an application template.

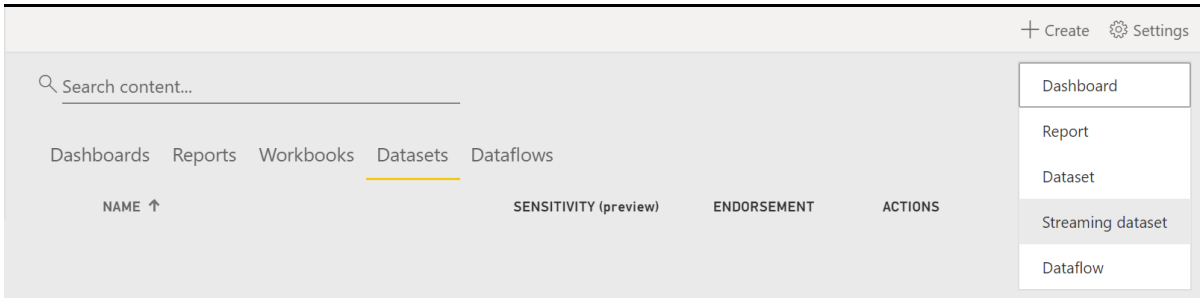
- An Azure Event Hubs namespace and Event Hub.
- The Logic App that you want to access your Event Hub. To start your Logic App with an Azure Event Hubs trigger, you need a blank Logic App.
- A Power BI service account. If you don't have one already, you can create a free trial account for Power BI service. If you haven't used Power BI before, it might be helpful to go through Get started with Power BI.

Set up a continuous data export to Azure Event Hubs

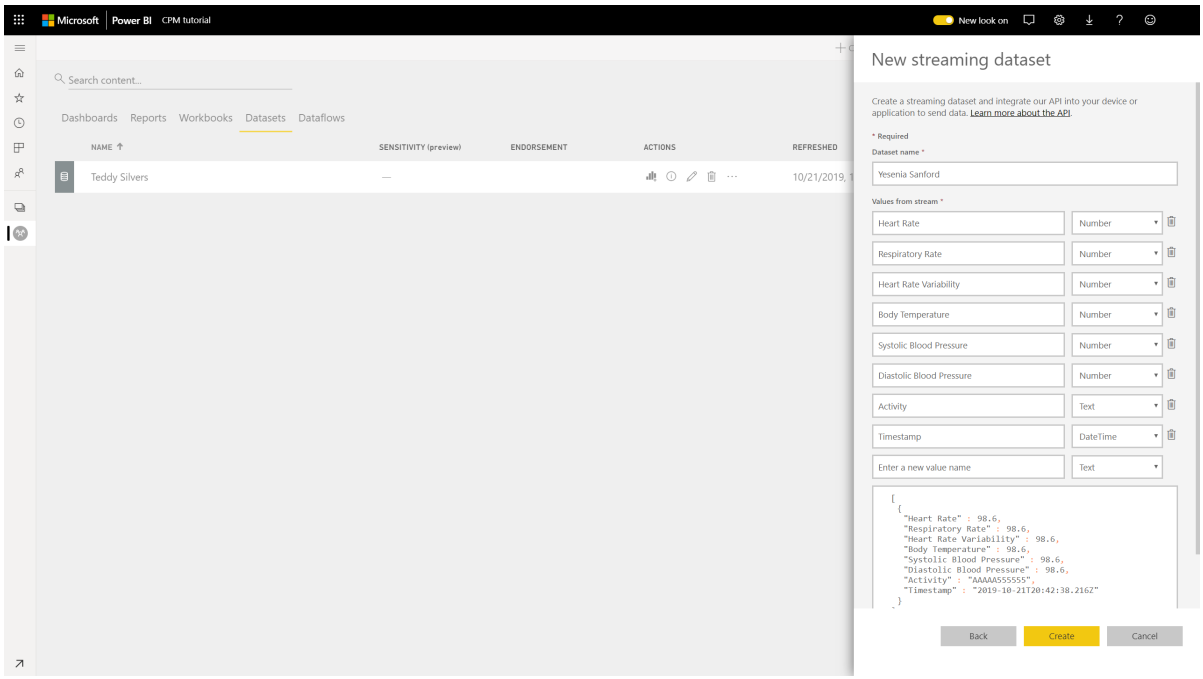
You will first need to set up a continuous data export from your Azure IoT Central app template to the Azure Event Hub in your subscription. You can do so by following the steps in this Azure IoT Central tutorial for Exporting to Event Hubs. You will only need to export for the telemetry for the purposes of this tutorial.

Create a Power BI streaming dataset

1. Sign in to your Power BI account.
2. In your preferred Workspace, create a new streaming dataset by selecting the **+ Create** button in the upper-right corner of the toolbar. You will need to create a separate dataset for each patient that you would like to have on your dashboard.



3. Choose **API** for the source of your dataset.
4. Enter a **name** (for example, a patient's name) for your dataset and then fill out the values from your stream. You can see an example below based on values coming from the simulated devices in the continuous patient monitoring application template. The example has two patients:
   - Teddy Silvers, who has data from the Smart Knee Brace
   - Yesenia Sanford, who has data from the Smart Vitals Patch



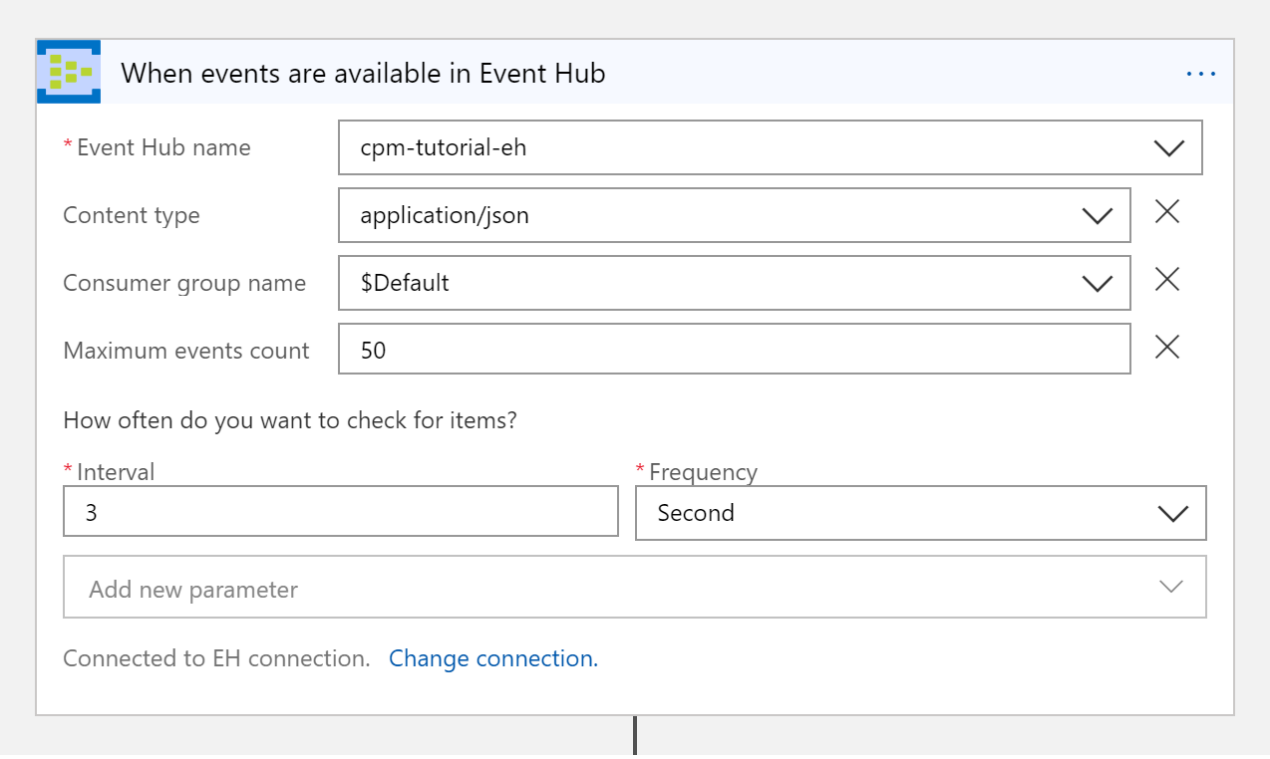
To learn more about streaming datasets in Power BI, you can read this document on real-time streaming in Power BI.

Connect your Logic App to Azure Event Hubs

To connect your Logic App to Azure Event Hubs, you can follow the instructions outlined in this document on Sending events with Azure Event Hubs and Azure Logic Apps. Here are some suggested parameters:

PARAMETER	VALUE
Content type	application/json
Interval	3
Frequency	Second

At the end of this step, your Logic App Designer should look like this:



Stream data to Power BI from your Logic App

The next step will be to parse the data coming from your Event Hub to stream it into the Power BI datasets that you have previously created.

1. Before you can do this, you will need to understand the JSON payload that is being sent from your device to your Event Hub. You can do so by looking at this sample schema and modifying it to match your schema or using Service Bus explorer to inspect the messages. If you are using the continuous patient monitoring applications, your messages will look like this:

Smart Vitals Patch telemetry

```
{
  "HeartRate": 80,
  "RespiratoryRate": 12,
  "HeartRateVariability": 64,
  "BodyTemperature": 99.08839032397609,
  "BloodPressure": {
    "Systolic": 23,
    "Diastolic": 34
  },
  "Activity": "walking"
}
```

Smart Knee Brace telemetry

```
{
  "Acceleration": {
    "x": 72.73510947763711,
    "y": 72.73510947763711,
    "z": 72.73510947763711
  },
  "RangeOfMotion": 123,
  "KneeBend": 3
}
```

Properties

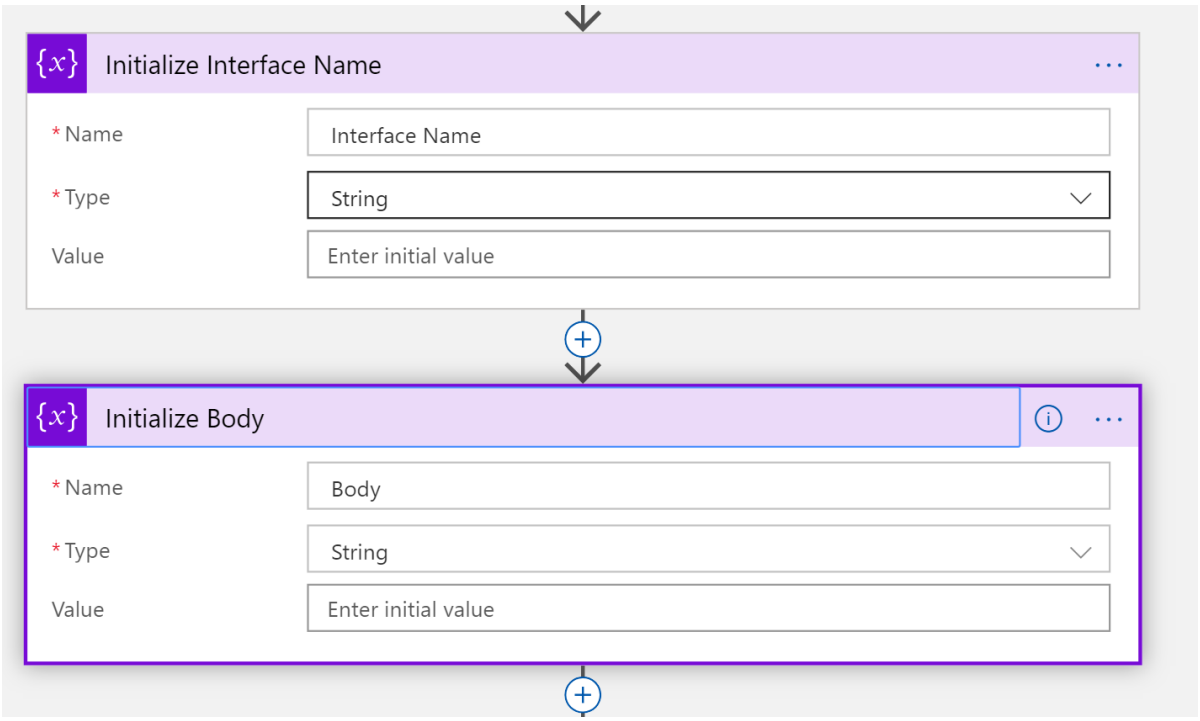
```
{
  "iothub-connection-device-id": "1qsi9p8t5l2",
  "iothub-connection-auth-method": "{\"scope\":\"device\",\"type\":\"sas\",
\"issuer\":\"iothub\",\"acceptingIpFilterRule\":null}",
  "iothub-connection-auth-generation-id": "637063718586331040",
  "iothub-enqueuedtime": 1571681440990,
  "iothub-message-source": "Telemetry",
  "iothub-interface-name": "Patient_health_data_3bk",
  "x-opt-sequence-number": 7,
  "x-opt-offset": "3672",
  "x-opt-enqueued-time": 1571681441317
}
```

2. Now that you have inspected your JSON payloads, go back to your Logic App Designer and select **+ New Step**. Search and add **Initialize variable** as your next step and enter the following parameters:

PARAMETER	VALUE
Name	Interface Name
Type	String

Hit **Save**.

3. Add another variable called **Body** with Type as **String**. Your Logic App will have these actions added:

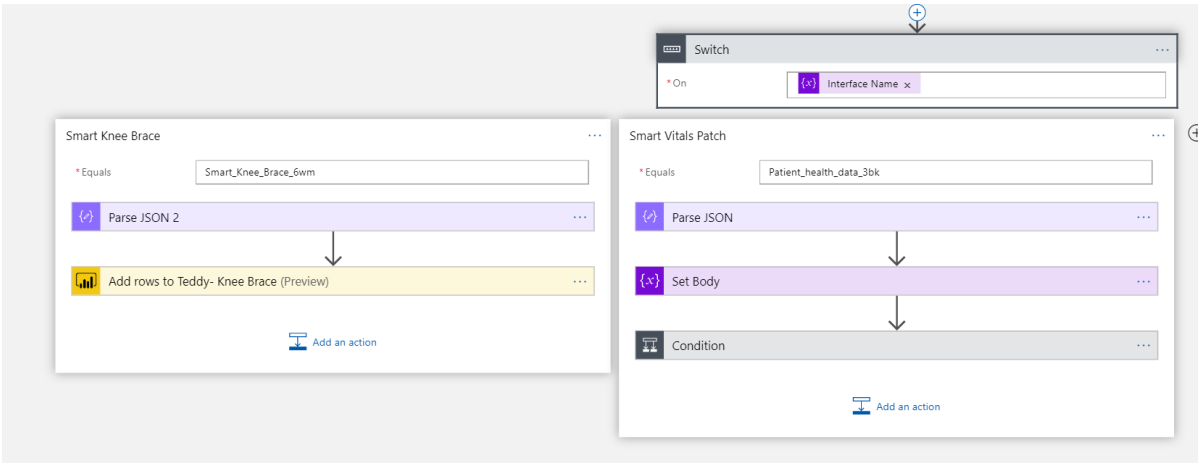


4. Select **+ New Step** and add a **Parse JSON** action. Rename this to **Parse Properties**. For the Content, choose **Properties** coming from the Event Hub. Select **Use sample payload to generate schema** at the bottom, and paste the sample payload from the Properties section above.

5. Next, choose the **Set variable** action and update your **Interface Name** variable with the **iothub-interface-name** from the parsed JSON properties.

6. Add a **Split** Control as your next action and choose the **Interface Name** variable as the On parameter. You will use this to funnel the data to the correct dataset.

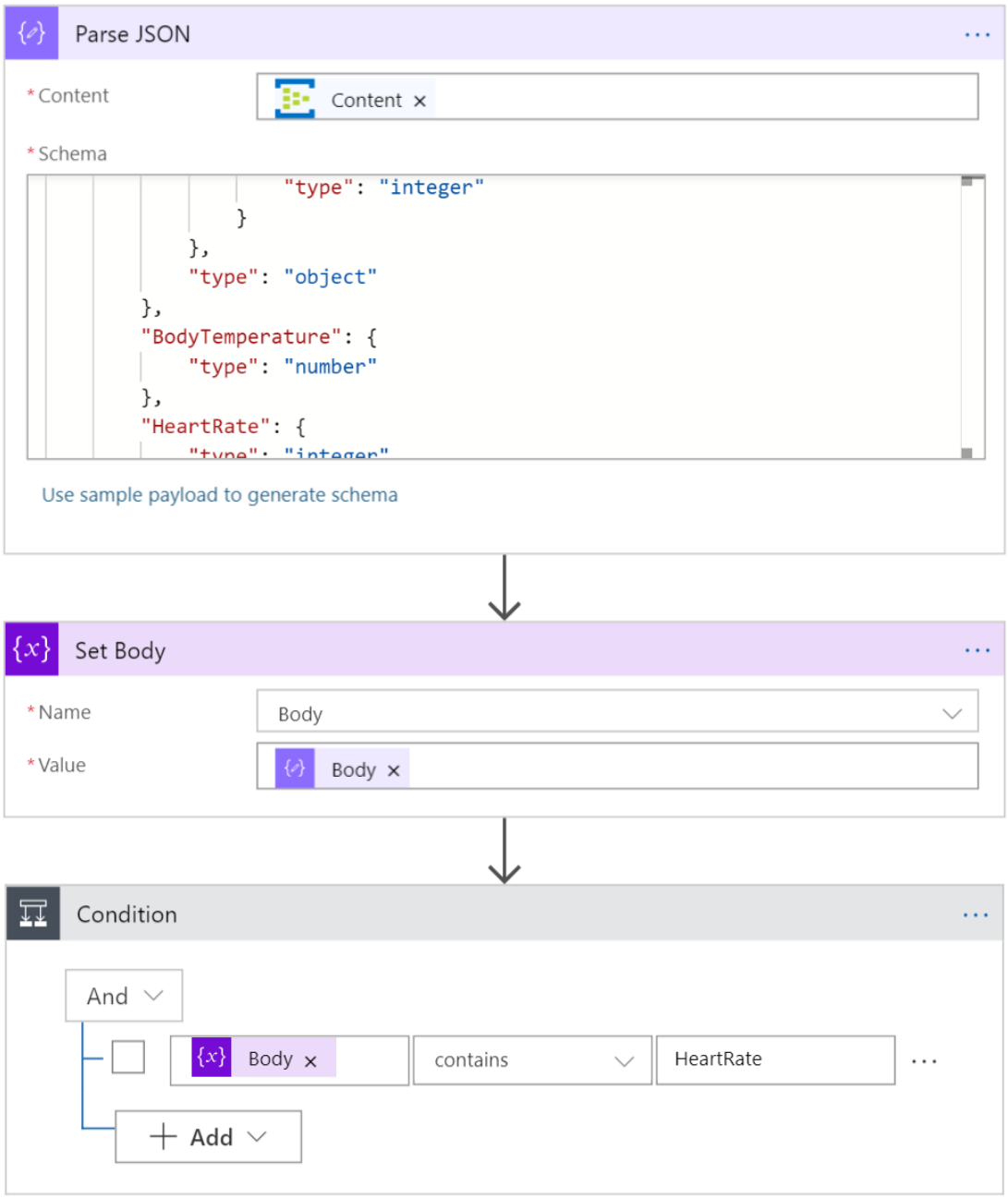
7. In your Azure IoT Central application, find the Interface Name for the Smart Vitals Patch health data and the Smart Knee Brace health data from the **Device Templates** view. Create two different cases for the **Switch** Control for each Interface Name and rename the control appropriately. You can set the Default case to use the **Terminate** Control and choose what status you would like to show.



8. For the **Smart Vitals Patch** case, add a **Parse JSON** action. For the Content, choose **Content** coming from the Event Hub. Copy and paste the sample payloads for the Smart Vitals Patch above to generate the schema.

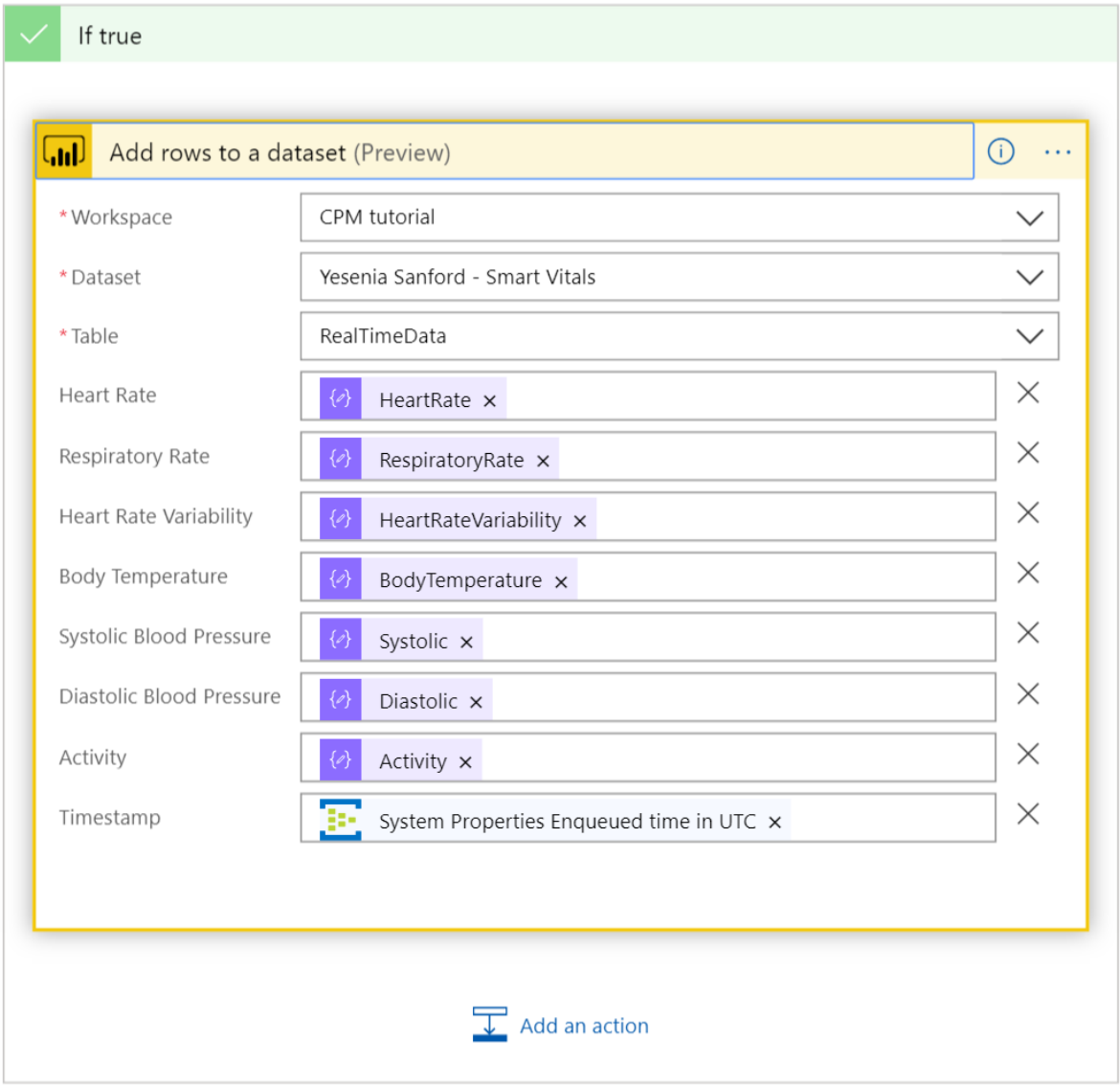
9. Add a **Set variable** action and update the **Body** variable with the **Body** from the parsed JSON in Step 7.

10. Add a **Condition** Control as your next action and set the condition to **Body**, **contains**, **HeartRate**. This will make sure that you have the right set of data coming from the Smart Vitals Patch before populating the Power BI dataset. Steps 7-9 will look like this:

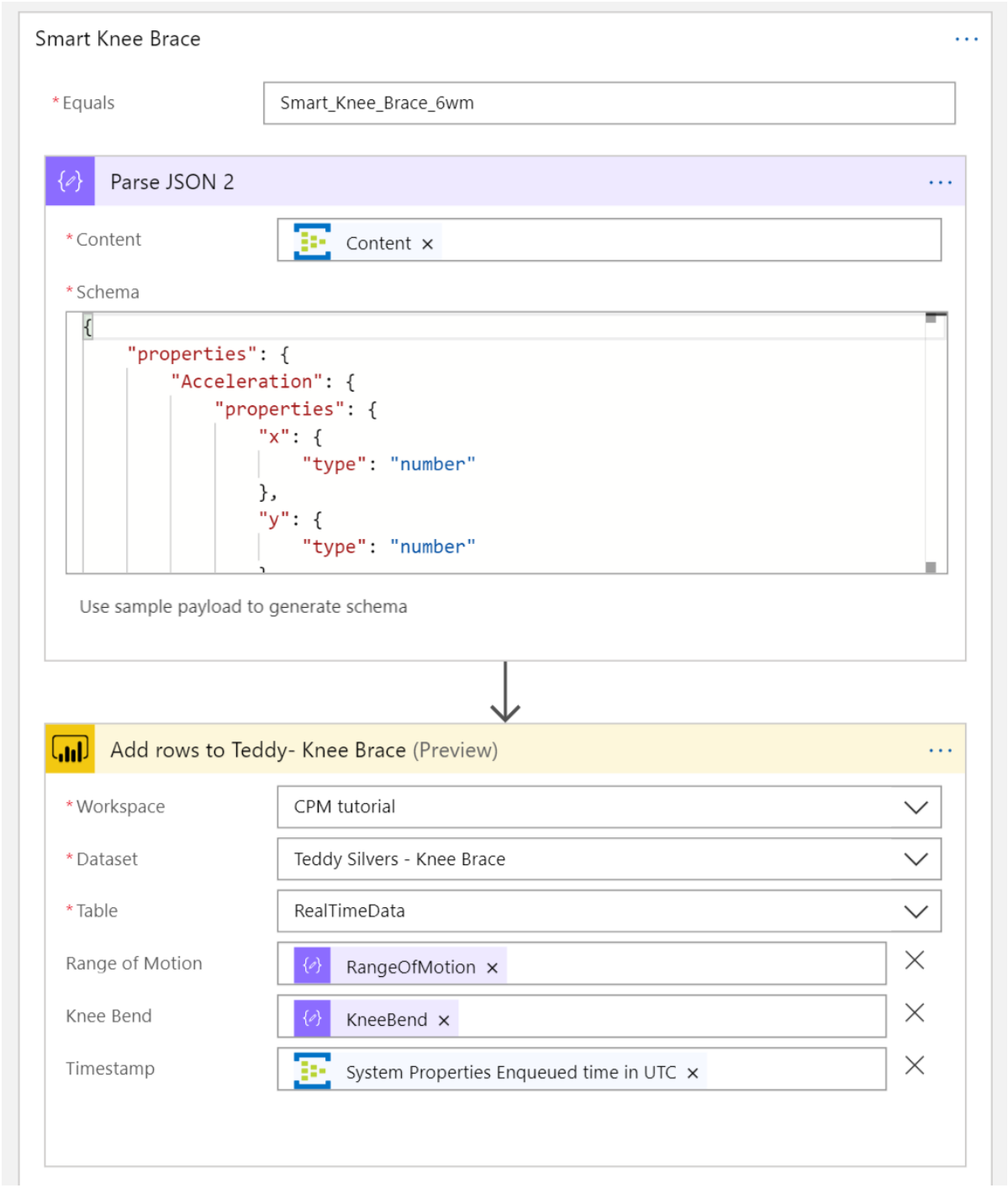


11. For the **True** case of the Condition, add an action that calls the **Add rows to a dataset** Power BI functionality. You will have to sign into Power BI for this. Your **False** case can again use the **Terminate** control.

12. Choose the appropriate **Workspace**, **Dataset**, and **Table**. Map the parameters that you specified when creating your streaming dataset in Power BI to the parsed JSON values that are coming from your Event Hub. Your filled-out actions should look like this:



13. For the **Smart Knee Brace** switch case, add a **Parse JSON** action to parse the content, similar to Step 7. Then **Add rows to a dataset** to update your Teddy Silvers dataset in Power BI.

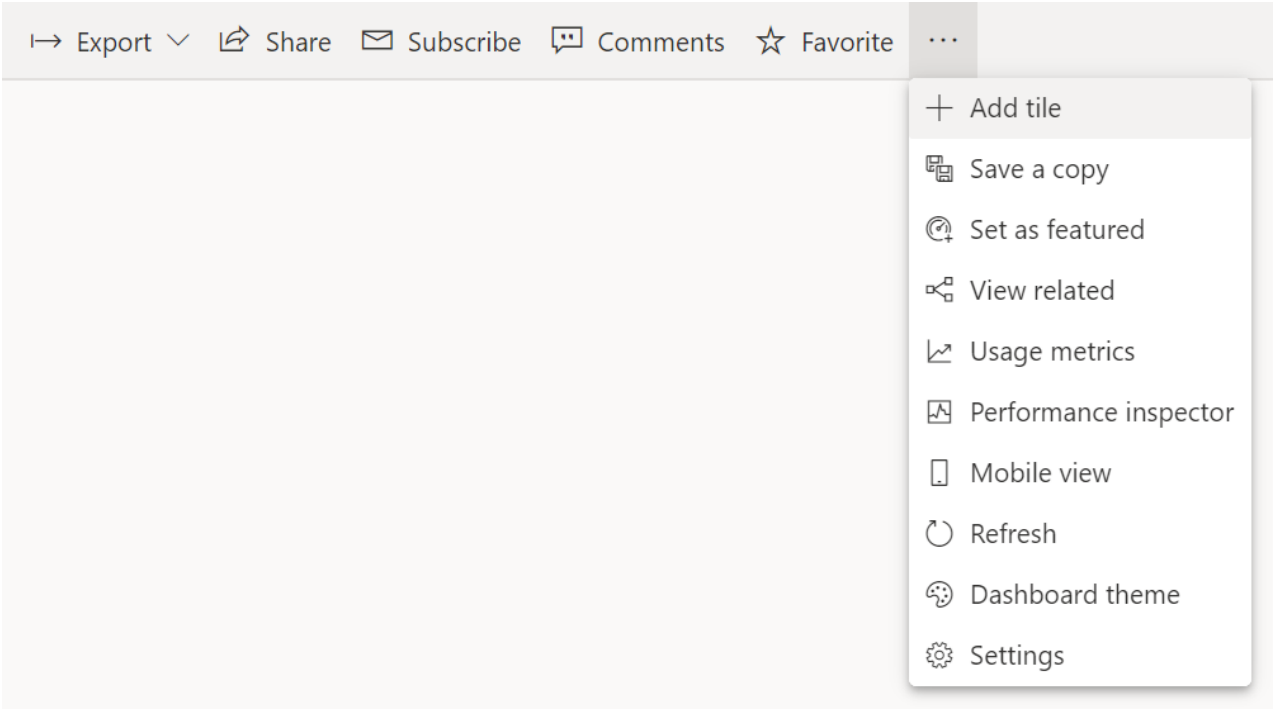


14. Press **Save** and then run your Logic App.

Build a real-time dashboard for patient vitals

Now go back to Power BI and select **+ Create** to create a new **Dashboard**. Give your dashboard a name and hit **Create**.

Select the three dots in the top navigation bar and then select **+ Add tile**.



Choose the type of tile you would like to add and customize your app however you'd like.

Clean up resources

If you're not going to continue to use this application, delete your resources with the following steps:

1. From the Azure portal, you can delete the Event Hub and Logic Apps resources that you created.
2. For your IoT Central application, go to the Administration tab and select **Delete**.

Next steps

- Review the continuous patient monitoring architecture guidance.

Connect an MXChip IoT DevKit device to your Azure IoT Central application

3/24/2020 • 3 minutes to read • Edit Online

This article shows you how to connect an MXChip IoT DevKit (DevKit) device to an Azure IoT Central application. The device uses the certified IoT Plug and Play (preview) model for the DevKit device to configure its connection to IoT Central.

In this how-to article, you:

- Get the connection details from your IoT Central application.
- Prepare the device and connect it to your IoT Central application.
- View the telemetry and properties from the device in IoT Central.

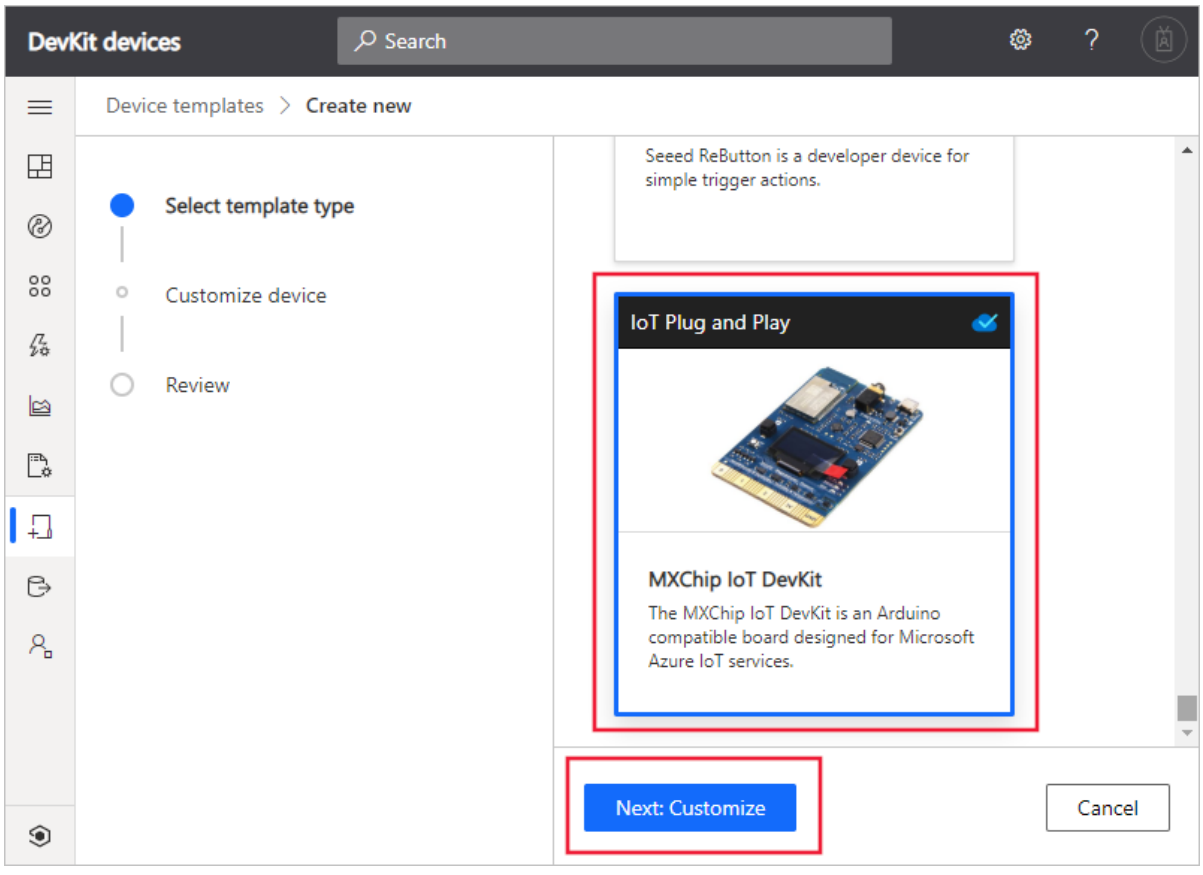
Prerequisites

To complete the steps in this article, you need the following resources:

- A DevKit device.
- An IoT Central application. You can follow the steps in Create an IoT Central application.

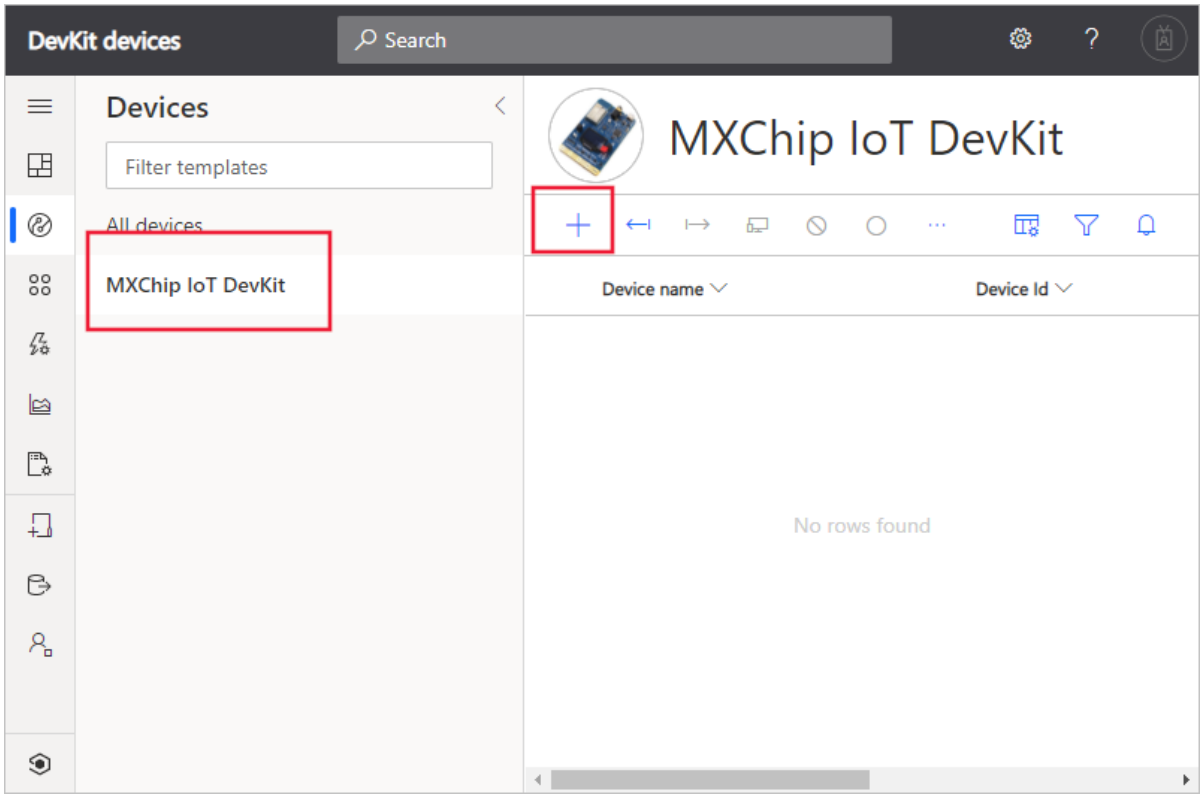
Get device connection details

1. In your Azure IoT Central application, select the **Device Templates** tab and select **+ New**. In the section **Use a preconfigured device template**, select **MXChip IoT DevKit**.

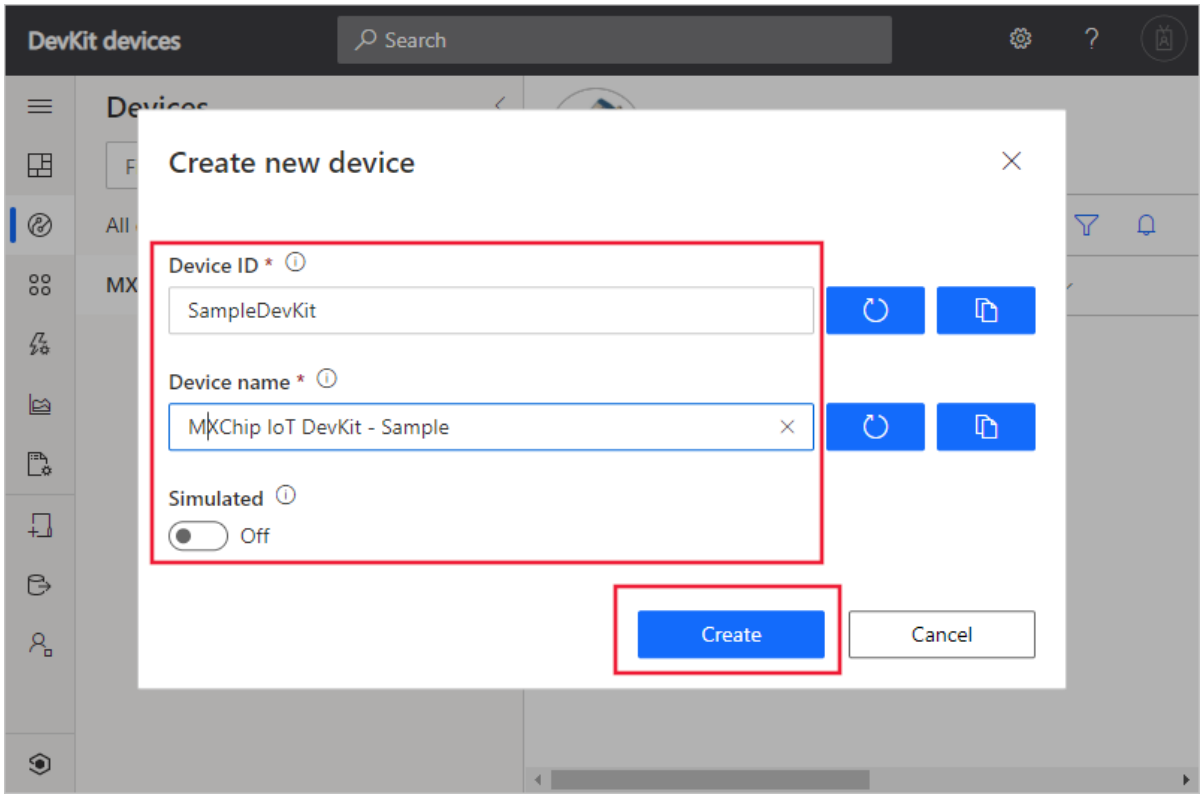


2. Select **Next: Customize** and then **Create**.

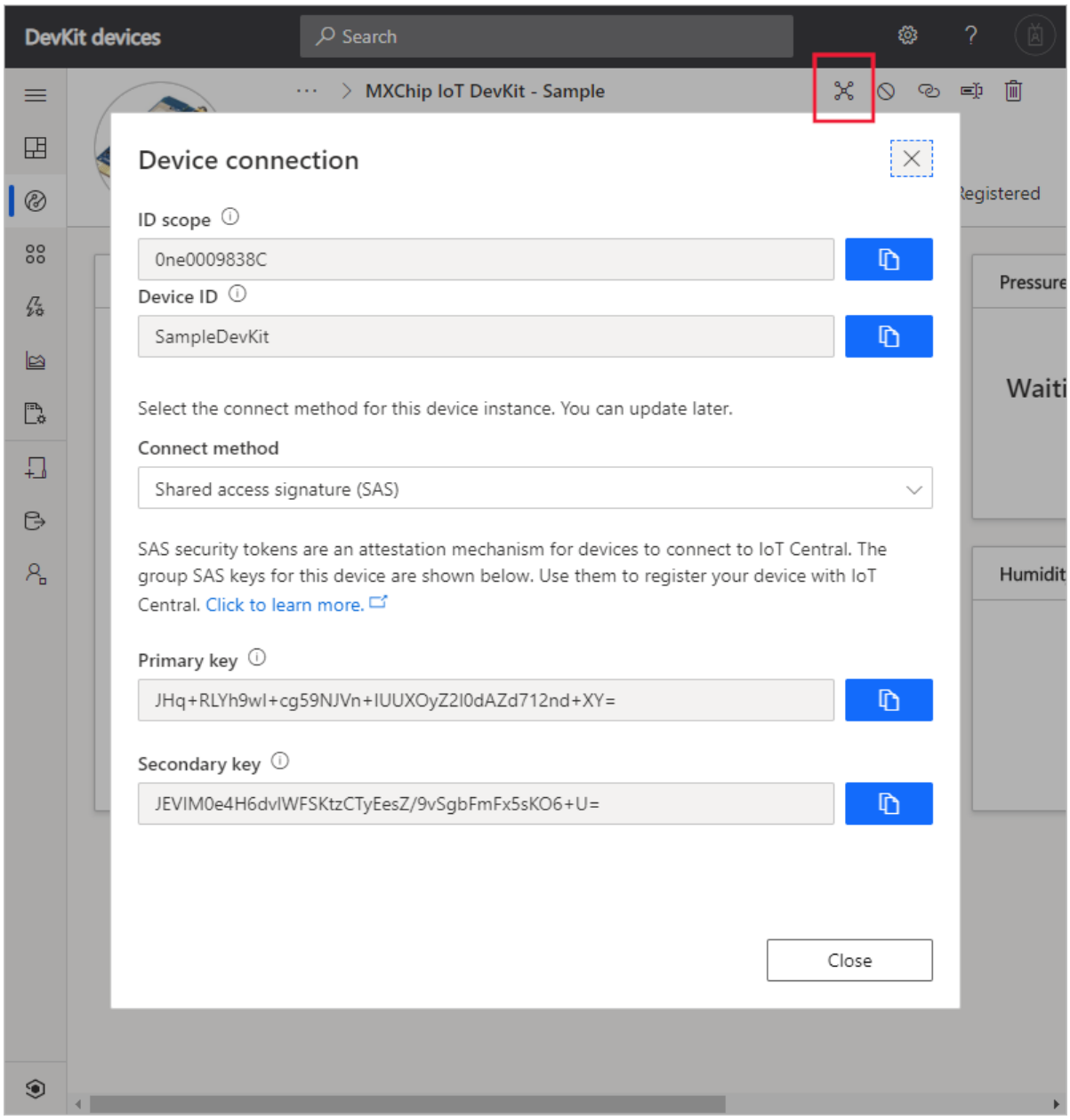
3. Select **Devices** tab. In the devices list, select **MXChip IoT DevKit** and select **+ New** to create a new device from the template.



4. In the pop-up window, enter the **Device ID** as `SampleDevKit` and **Device Name** as `MXChip IoT DevKit - Sample`. Make sure the **Simulated** option is off. Then select **Create**.



5. Select the device you created and then select **Connect**. Make a note of the **ID Scope**, **Device ID**, and **Primary key**. You need these values later in this how-to article.



Prepare the device

1. Download the latest pre-built Azure IoT Central Plug and Play (preview) firmware for the DevKit device from GitHub.

2. Connect the DevKit device to your development machine using a USB cable. In Windows, a file explorer window opens on a drive mapped to the storage on the DevKit device. For example, the drive might be called **AZ3166 (D:)**.

3. Drag the **iotc_devkit.bin** file onto the drive window. When the copying is complete, the device reboots with the new firmware.

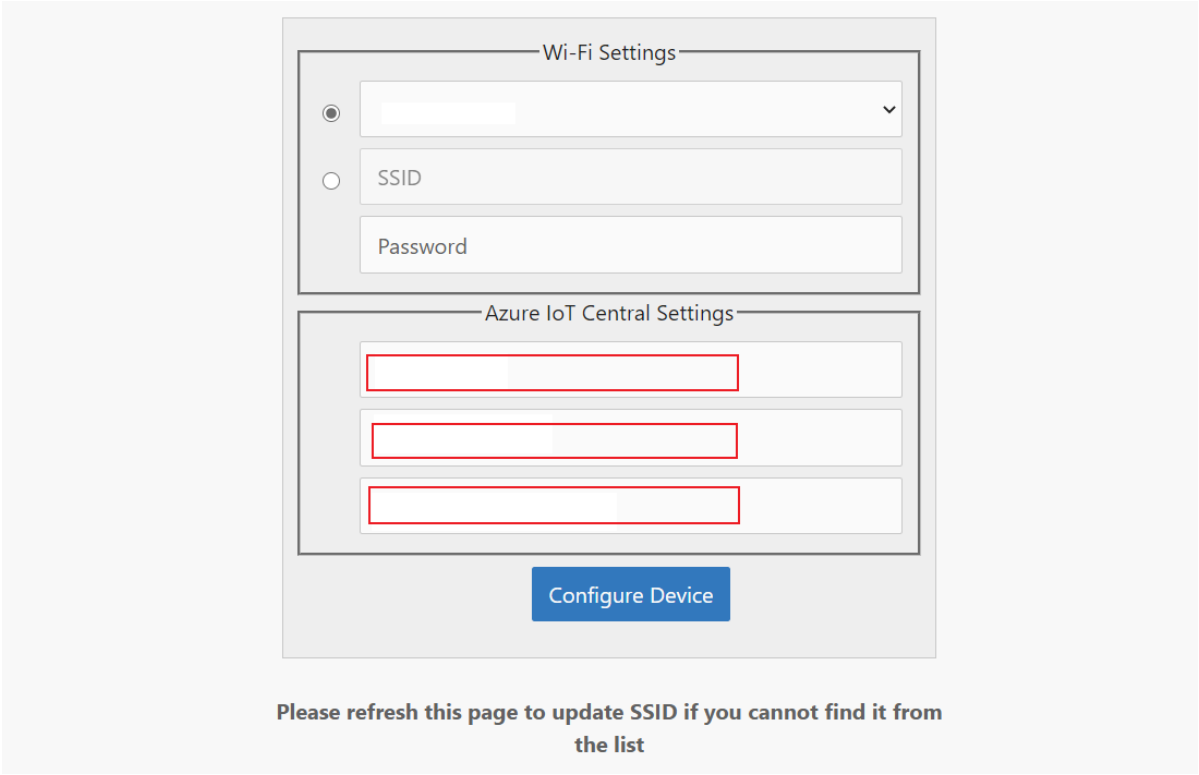
   > **NOTE**
   > If you see errors on the screen such as **No Wi-Fi**, this is because the DevKit has not yet been connected to WiFi.

4. On the DevKit, hold down **button B**, push and release the **Reset** button, and then release **button B**. The device is now in access point mode. To confirm, the screen displays "IoT DevKit - AP" and the configuration portal IP address.

5. On your computer or tablet, connect to the WiFi network name shown on the screen of the device. The WiFi

network starts with **AZ-** followed by the MAC address. When you connect to this network, you don't have internet access. This state is expected, and you only connect to this network for a short time while you configure the device.

6. Open your web browser and navigate to http://192.168.0.1/. The following web page displays:

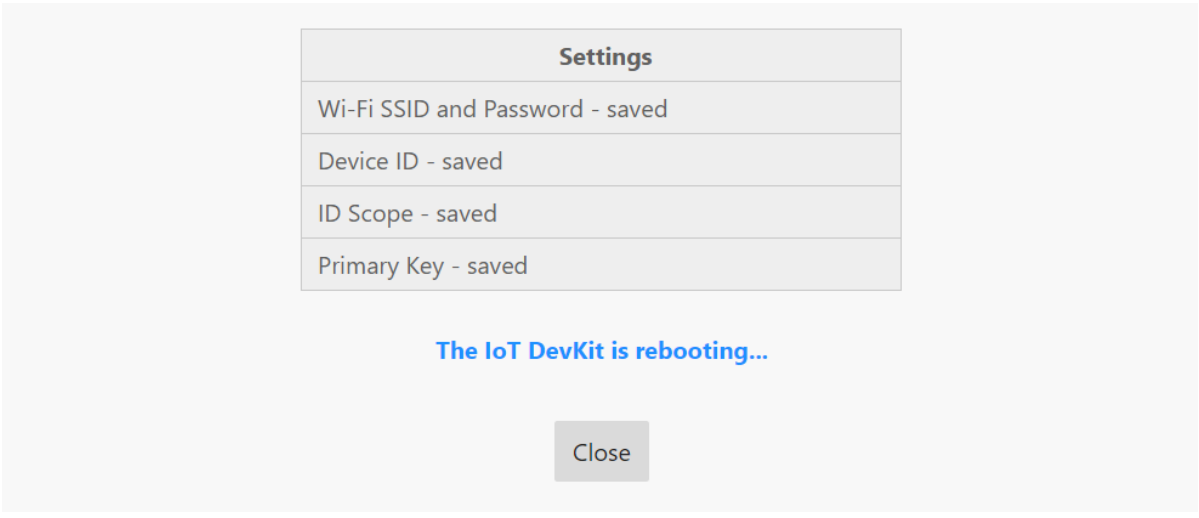


On the web page, enter:

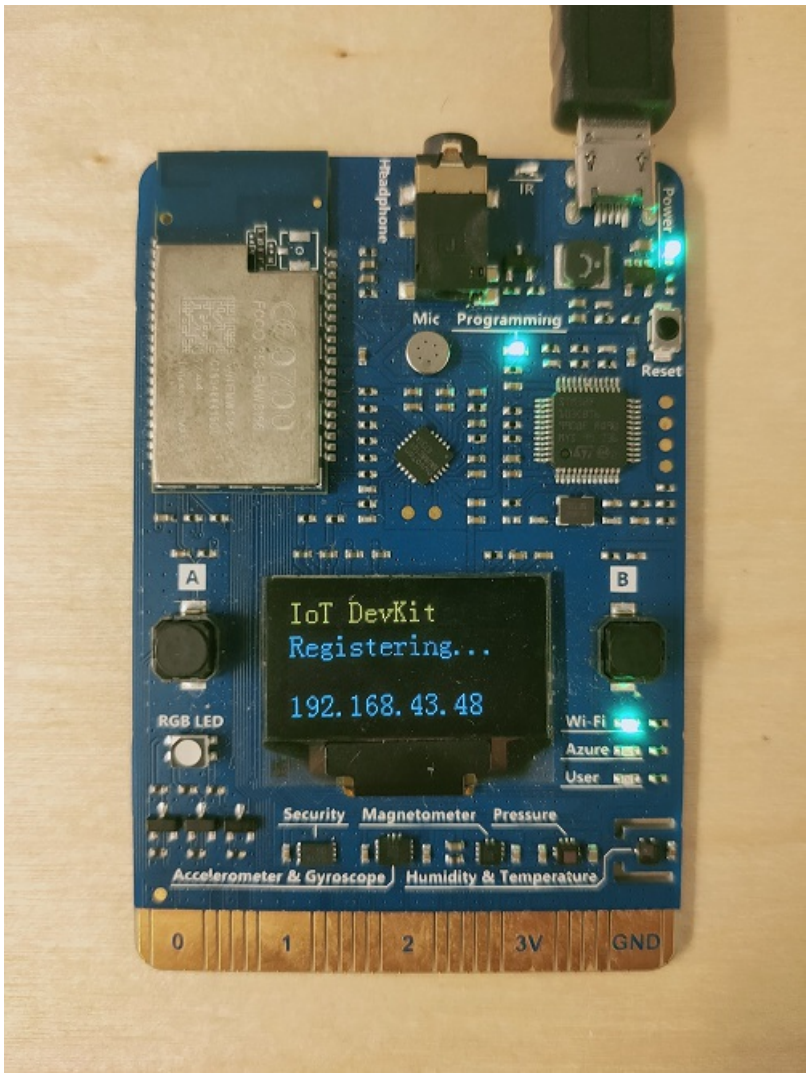
- The name of your WiFi network (SSID).
- Your WiFi network password.
- The connection details: enter the **Device ID**, **ID Scope**, and **SAS Primary Key** you made a note of previously.

> **NOTE**
> Currently, the IoT DevKit only can connect to 2.4 GHz Wi-Fi, 5 GHz is not supported due to hardware restrictions.

7. Choose **Configure Device**, the DevKit device reboots and runs the application:



The DevKit screen displays a confirmation that the application is running:

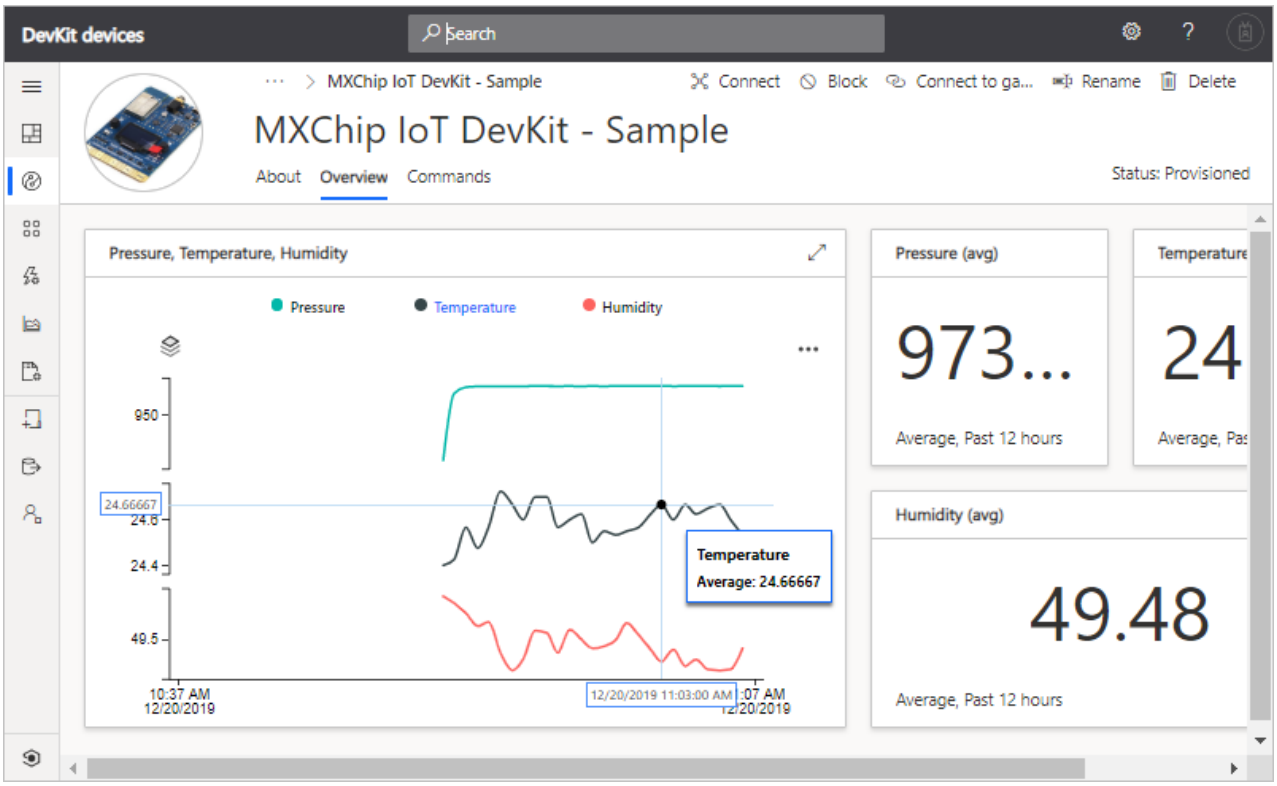


The DevKit first registers a new device in IoT Central application and then starts sending data.

View the telemetry

In this step, you view the telemetry in your Azure IoT Central application.

In your IoT Central application, select **Devices** tab, select the device you added. In the **Overview** tab, you can see the telemetry from the DevKit device:



Review the code

To review the code or modify and compile it, go to the Code Samples.

Next steps

Now that you've learned how to connect a DevKit device to your Azure IoT Central application, the suggested next step is to learn how to set up a custom device template for your own IoT device.

Connect a Rigado Cascade 500 gateway device to your Azure IoT Central application

3/24/2020 • 2 minutes to read • Edit Online

This article describes how, as a solution builder, you can connect a Rigado Cascade 500 gateway device to your Microsoft Azure IoT Central application.

What is Cascade 500?

Cascade 500 IoT gateway is a hardware offering from Rigado that is included as part of their Cascade Edge-as-a-Service solution. It provides commercial IoT project and product teams with flexible edge computing power, a robust containerized application environment, and a wide variety of wireless device connectivity options, including Bluetooth 5, LTE, & Wi-Fi.

Cascade 500 is pre-certified for Azure IoT Plug and Play (preview) allowing our solution builders to easily onboard the device into their end to end solutions. The Cascade gateway allows you to wirelessly connect to a variety of condition monitoring sensors that are in proximity to the gateway device. These sensors can be onboarded into IoT Central via the gateway device.

Prerequisites

To step through this how-to guide, you need the following resources:

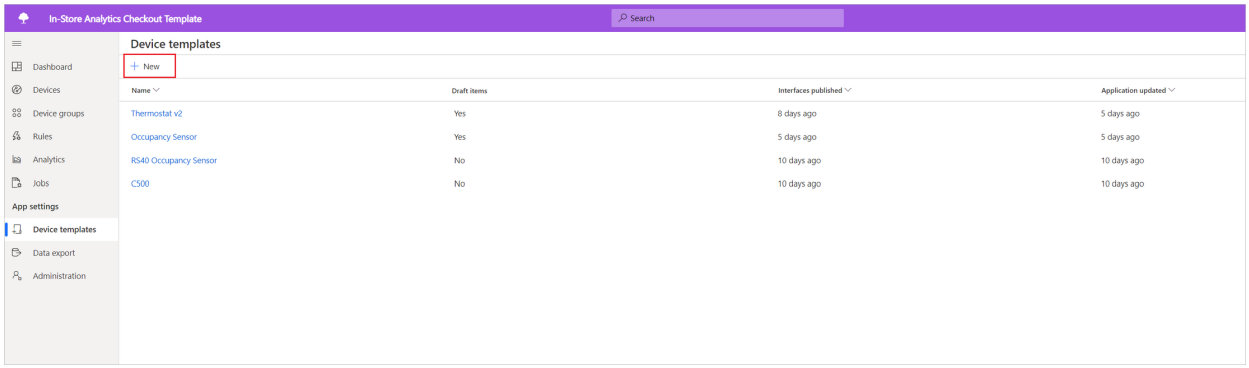
- A Rigado Cascade 500 device. For more information, please visit Rigado.
- An Azure IoT Central application. For more information, see the create a new application.

Add a device template

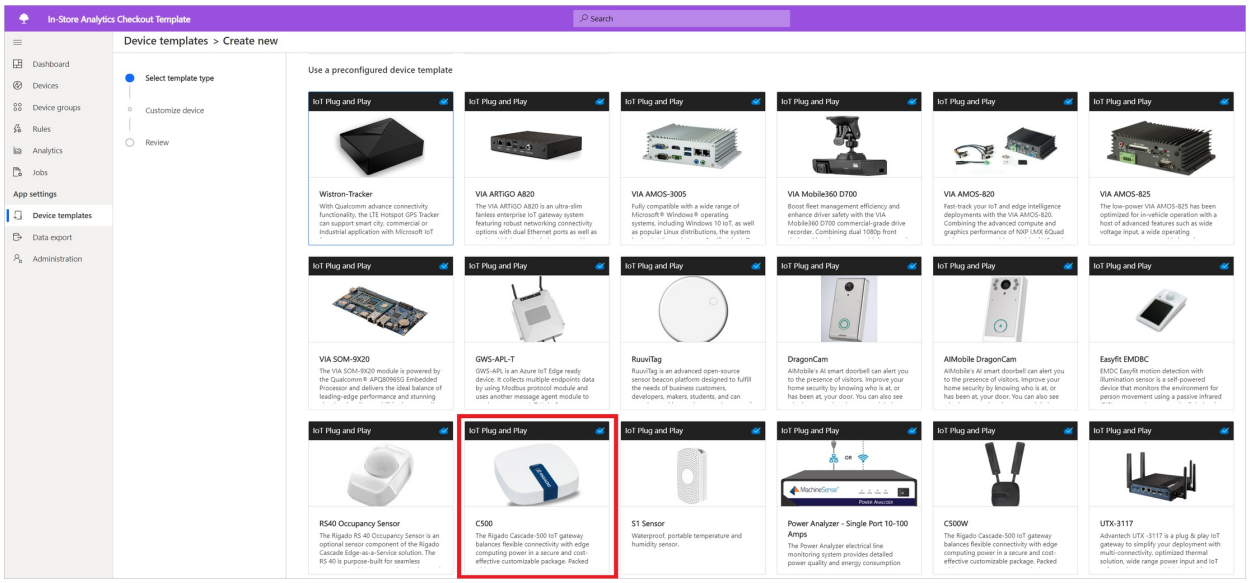
In order to onboard a Cascade 500 gateway device into your Azure IoT Central application instance, you will need to configure a corresponding device template within your application.

To add a Cascade 500 device template:

1. Navigate to the ***Device Templates*** tab in the left pane, select **+ New**:



2. The page gives you an option to ***Create a custom template*** or ***Use a preconfigured device template***
3. Select the C500 device template from the list of preconfigured device templates as shown below:

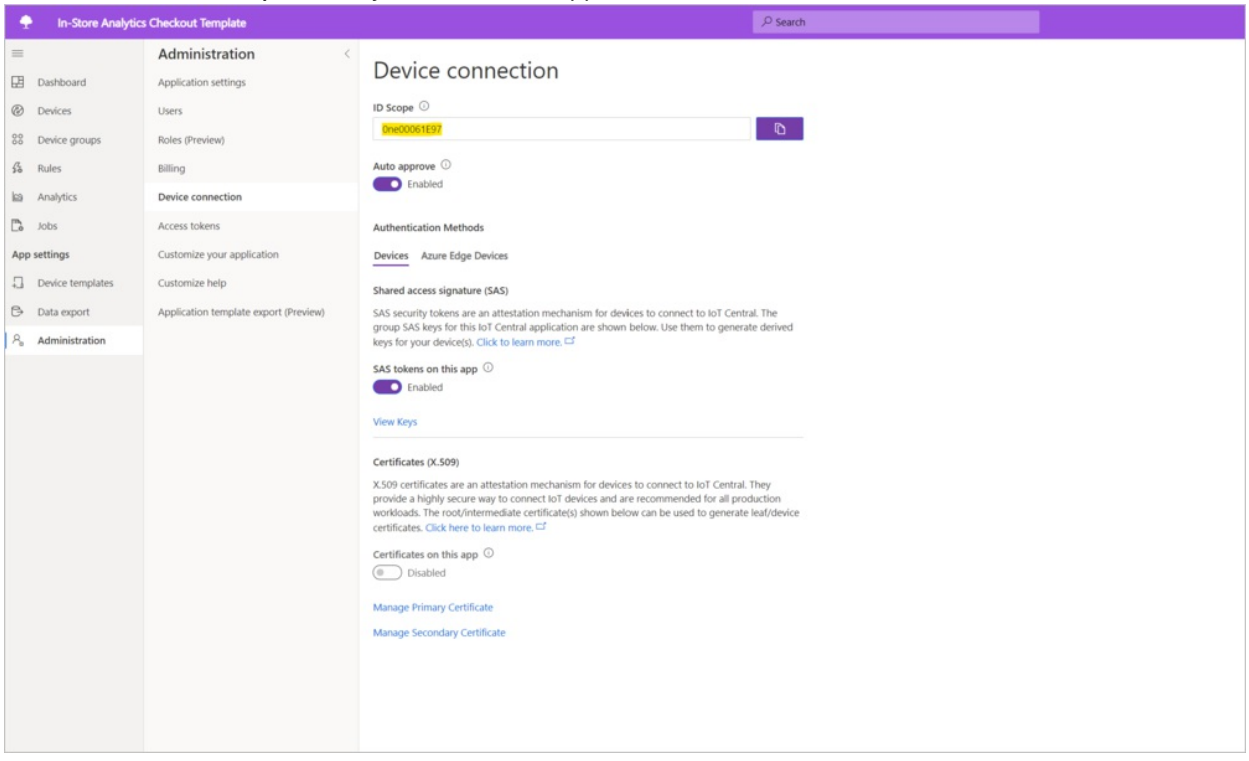


4. Select ***Next: Customize*** to continue to the next step.
5. On the next screen, select ***Create*** to onboard the C500 device template into your IoT Central application.

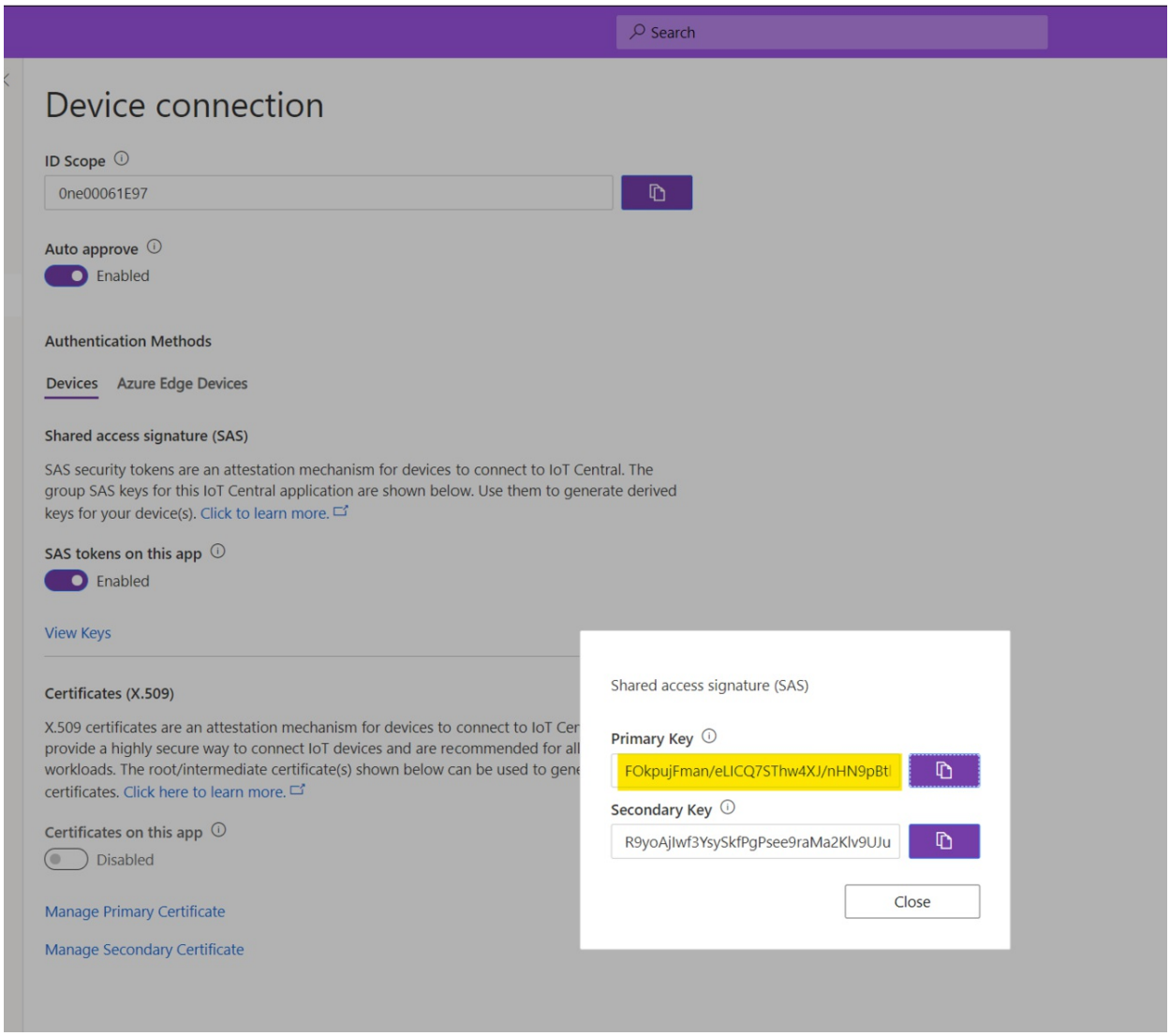
Retrieve application connection details

You will now need to retrieve the **Scope ID** and **Primary key** for your Azure IoT Central application in order to connect the Cascade 500 device.

1. Navigate to **Administration** in the left pane and click on **Device connection**.
2. Make a note of the **Scope ID** for your IoT Central application.



3. Now click on **View Keys** and make a note of the **Primary key**

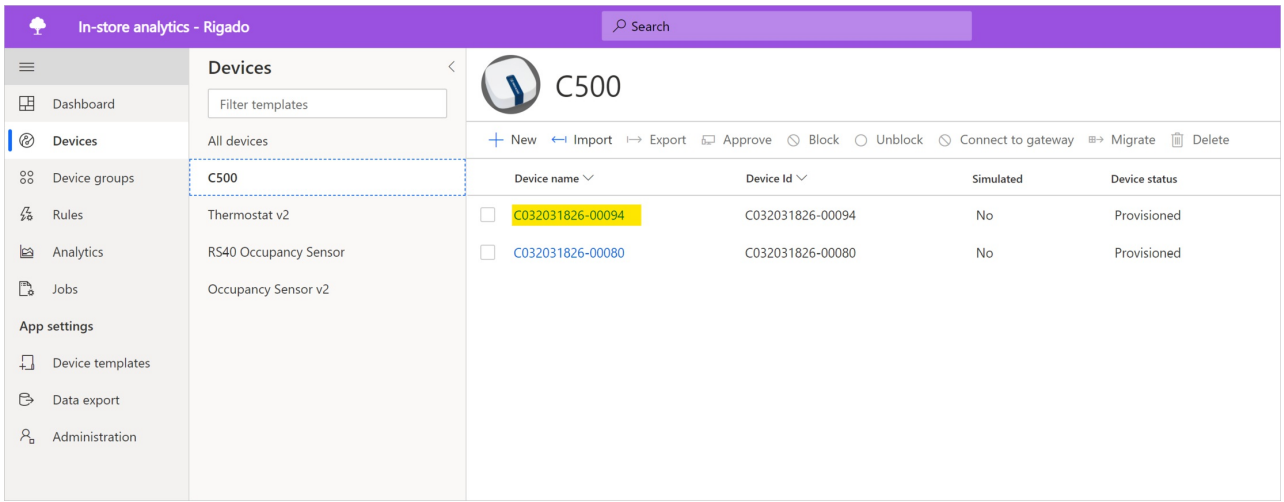


Contact Rigado to connect the gateway

In order to connect the Cascade 500 device to your IoT Central application, you will need to contact Rigado and provide them with the application connection details from the above steps.

Once the device is connected to the internet, Rigado will be able to push down a configuration update down to the Cascade 500 gateway device through a secure channel.

This update will apply the IoT Central connection details on the Cascade 500 device and it will appear in your devices list.



You are now ready to use your C500 device in your IoT Central application!

Next steps

Now that you've learned how to connect a Rigado Cascade 500 to your Azure IoT Central application, the suggested next step is to learn how to create an in-store analytics application to build an end to end solution.

Monitor device connectivity using Azure CLI

4/14/2020 • 2 minutes to read • Edit Online

This topic applies to builders and administrators.

Use the Azure CLI IoT extension to see messages your devices are sending to IoT Central and observe changes in the device twin. You can use this tool to debug and observe device connectivity and diagnose issues of device messages not reaching the cloud or devices not responding to twin changes.

Visit the Azure CLI extensions reference for more details

Prerequisites

- Azure CLI installed and is version 2.0.7 or higher. Check the version of your Azure CLI by running `az --version`. Learn how to install and update from the Azure CLI docs
- A work or school account in Azure, added as a user in an IoT Central application.

Install the IoT Central extension

Run the following command from your command line to install:

```
az extension add --name azure-iot
```

Check the version of the extension by running:

```
az --version
```

You should see the azure-iot extension is 0.8.1 or higher. If it is not, run:

```
az extension update --name azure-iot
```

Using the extension

The following sections describe common commands and options that you can use when you run `az iot central`. To view the full set of commands and options, pass `--help` to `az iot central` or any of its subcommands.

Login

Start by signing into the Azure CLI.

```
az login
```

Get the Application ID of your IoT Central app

In **Administration/Application Settings**, copy the **Application ID**. You use this value in later steps.

Monitor messages

Monitor the messages that are being sent to your IoT Central app from your devices. The output includes all headers and annotations.

```
az iot central app monitor-events --app-id <app-id> --properties all
```

View device properties

View the current read and read/write device properties for a given device.

```
az iot central device-twin show --app-id <app-id> --device-id <device-id>
```

Next steps

Now that you've learned how to use the IoT Central Explorer, the suggested next step is to explore managing devices IoT Central.

Manage devices in your Azure IoT Central application

3/24/2020 • 5 minutes to read • Edit Online

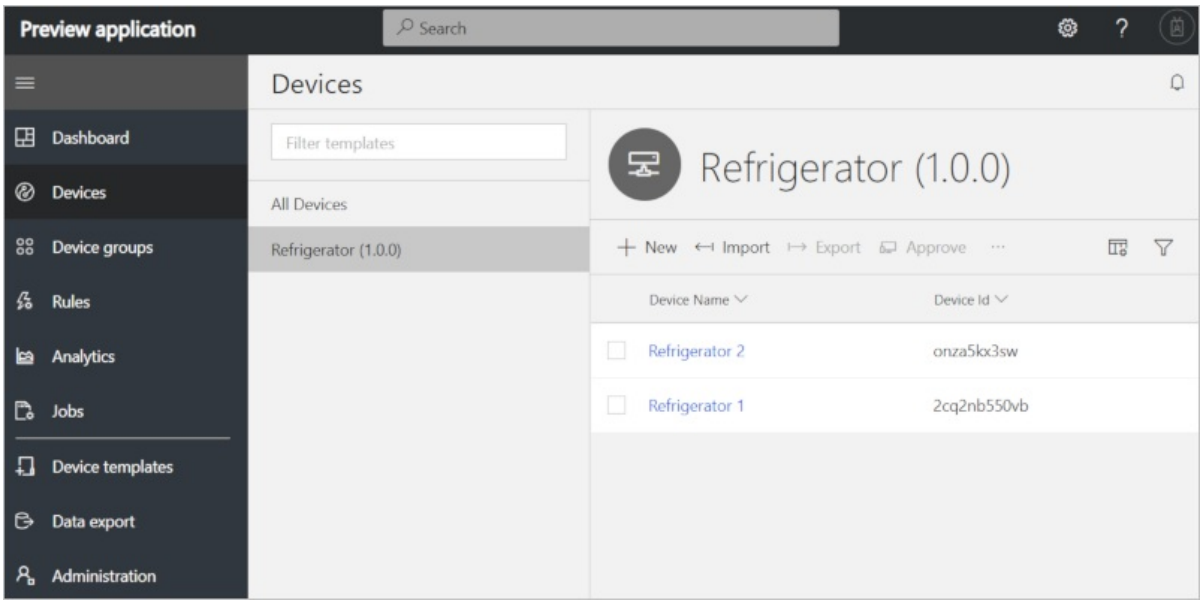
This article describes how, as an operator, to manage devices in your Azure IoT Central application. As an operator, you can:

- Use the **Devices** page to view, add, and delete devices connected to your Azure IoT Central application.
- Maintain an up-to-date inventory of your devices.
- Keep your device metadata up-to-date by changing the values stored in the device properties from your views.
- Control the behavior of your devices by updating a setting on a specific device from your views.

View your devices

To view an individual device:

1. Choose **Devices** on the left pane. Here you see a list of all devices and of your device templates.
2. Choose a device template.
3. In the right-hand pane of the **Devices** page, you see a list of devices created from that device template. Choose an individual device to see the device details page for that device:



Add a device

To add a device to your Azure IoT Central application:

1. Choose **Devices** on the left pane.
2. Choose the device template from which you want to create a device.
3. Choose + **New**.
4. Turn the **Simulated** toggle to **On** or **Off**. A real device is for a physical device that you connect to your Azure IoT Central application. A simulated device has sample data generated for you by Azure IoT Central.

5. Click **Create**.

6. This device now appears in your device list for this template. Select the device to see the device details page that contains all views for the device.

Import devices

To connect large number of devices to your application, you can bulk import devices from a CSV file. The CSV file should have the following columns and headers:

- **IOTC_DeviceID** - the device ID should be all lowercase.
- **IOTC_DeviceName** - this column is optional.

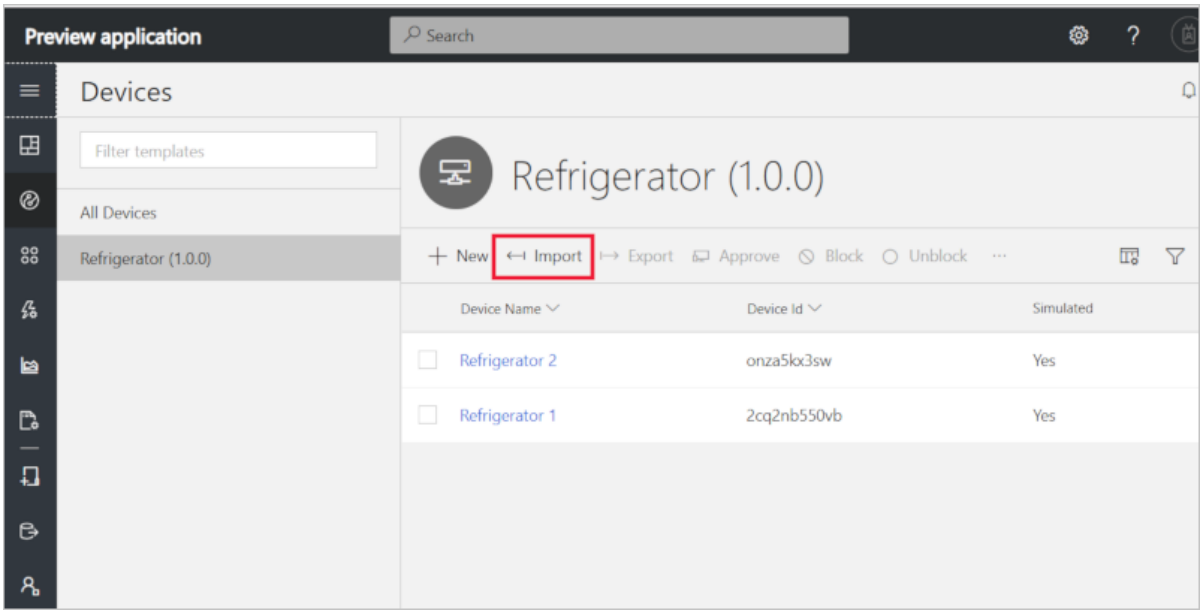
To bulk-register devices in your application:

1. Choose **Devices** on the left pane.

2. On the left panel, choose the device template for which you want to bulk create the devices.

> **NOTE**
> If you don't have a device template yet then you can import devices under **All devices** and register them without a template. After devices have been imported, you can then migrate them to a template.

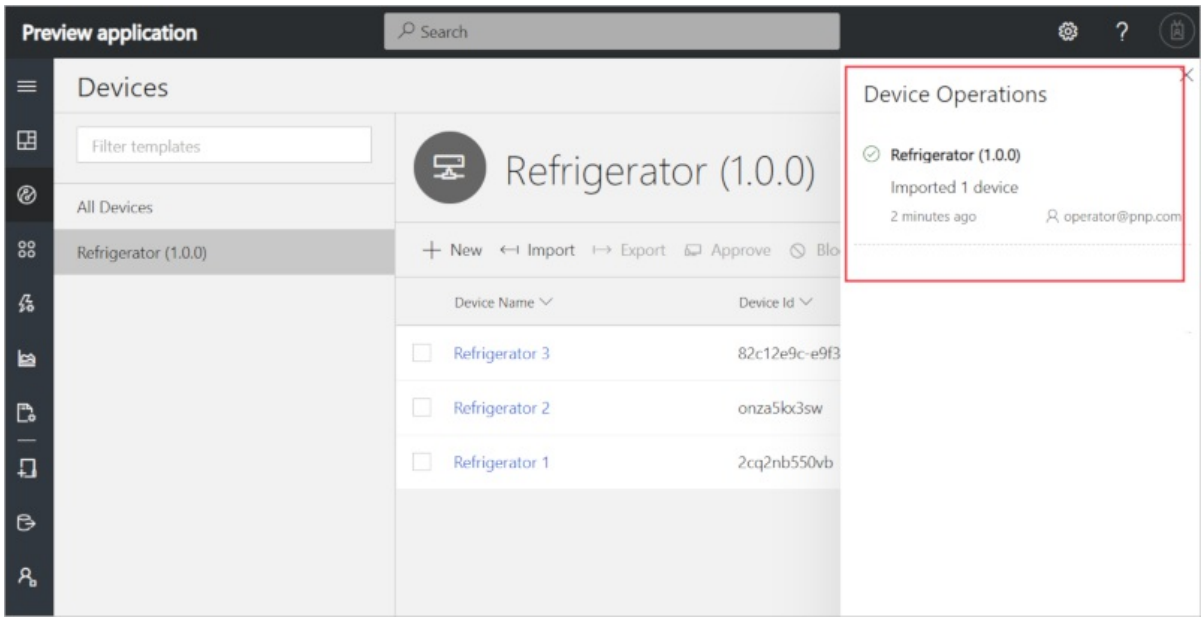
3. Select **Import**.



4. Select the CSV file that has the list of Device IDs to be imported.

5. Device import starts once the file has been uploaded. You can track the import status in the Device Operations panel. This panel appears automatically after the import starts or you can access it through the bell icon in the top right-hand corner.

6. Once the import completes, a success message is shown in the Device Operations panel.

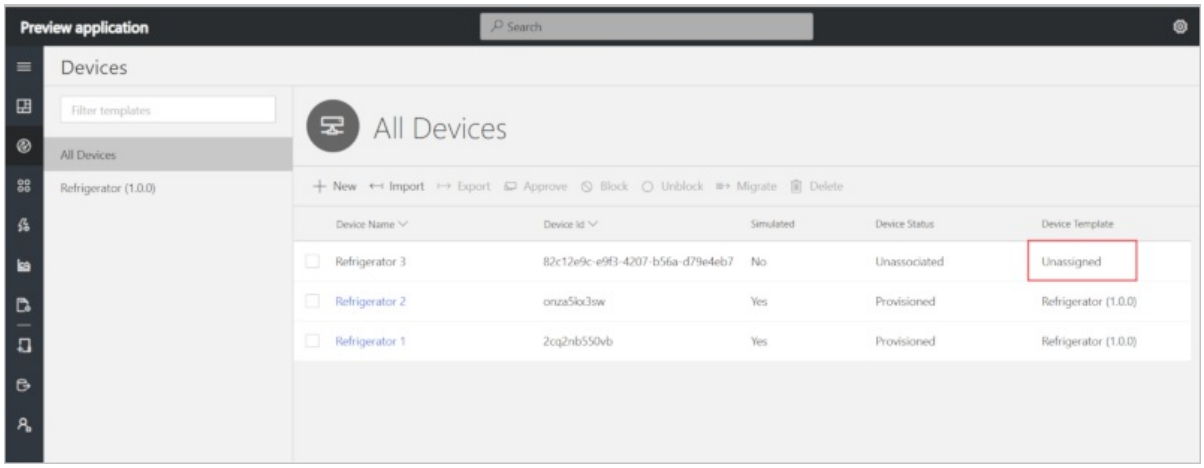


If the device import operation fails, you see an error message on the Device Operations panel. A log file capturing all the errors is generated that you can download.

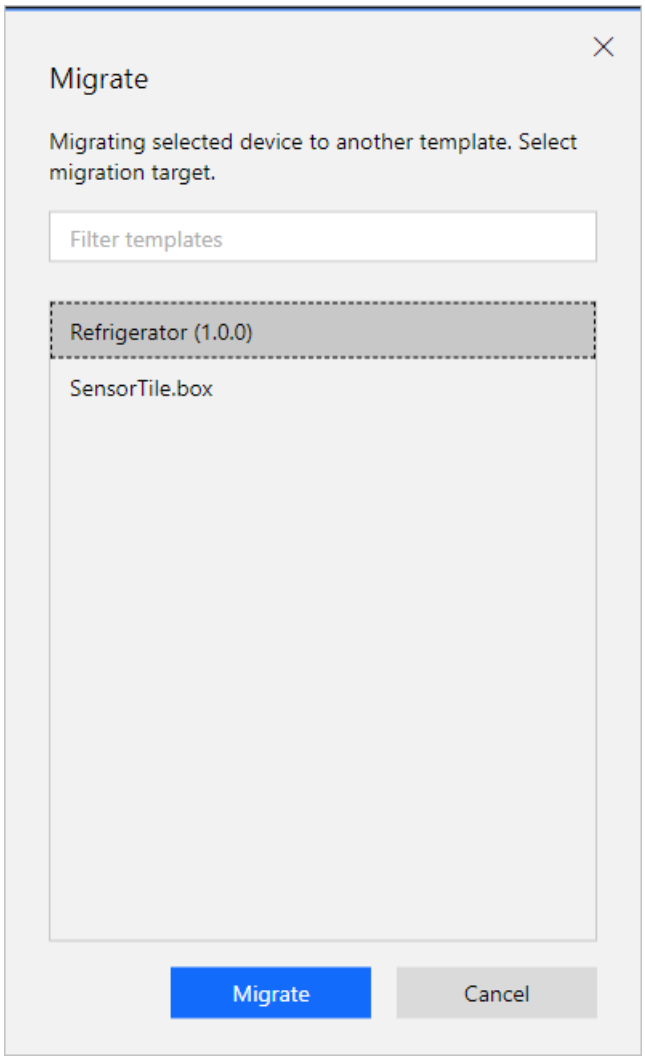
Migrating devices to a template

If you register devices by starting the import under **All devices**, then the devices are created without any device template association. Devices must be associated with a template to explore the data and other details about the device. Follow these steps to associate devices with a template:

1. Choose **Devices** on the left pane.
2. On the left panel, choose **All devices**:



3. Use the filter on the grid to determine if the value in the **Device Template** column is "Unassociated" for any of your devices.
4. Select the devices you want to associate with a template:
5. Select **Migrate**:



6. Choose the template from the list of available templates and select **Migrate**.

7. The selected devices are associated with the device template you chose.

Export devices

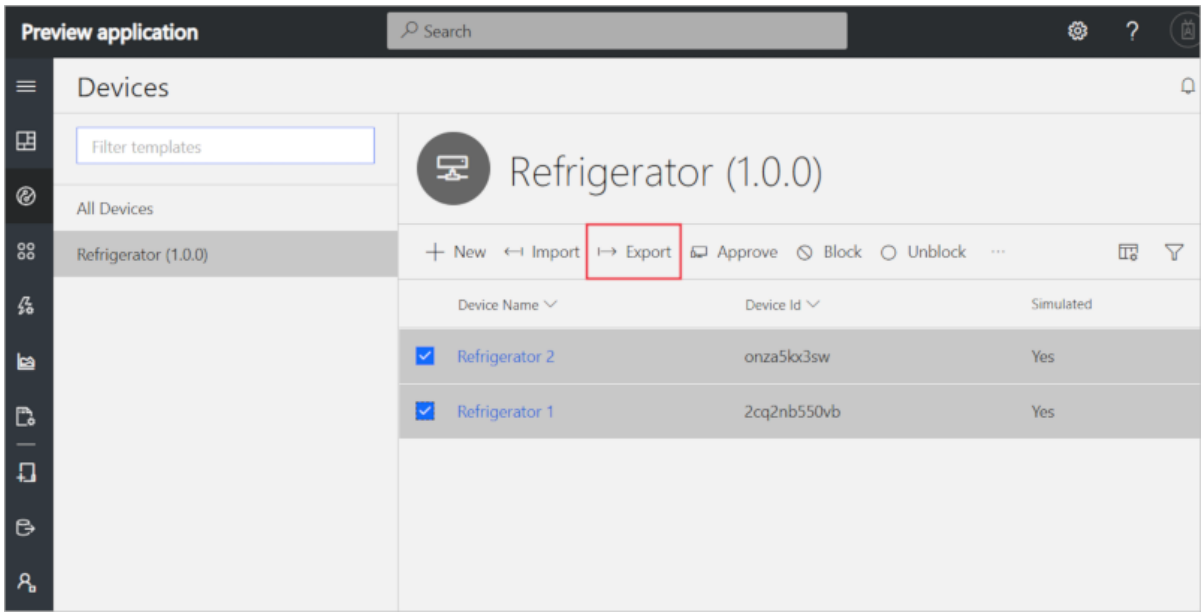
To connect a real device to IoT Central, you need its connection string. You can export device details in bulk to get the information you need to create device connection strings. The export process creates a CSV file with the device identity, device name, and keys for all the selected devices.

To bulk export devices from your application:

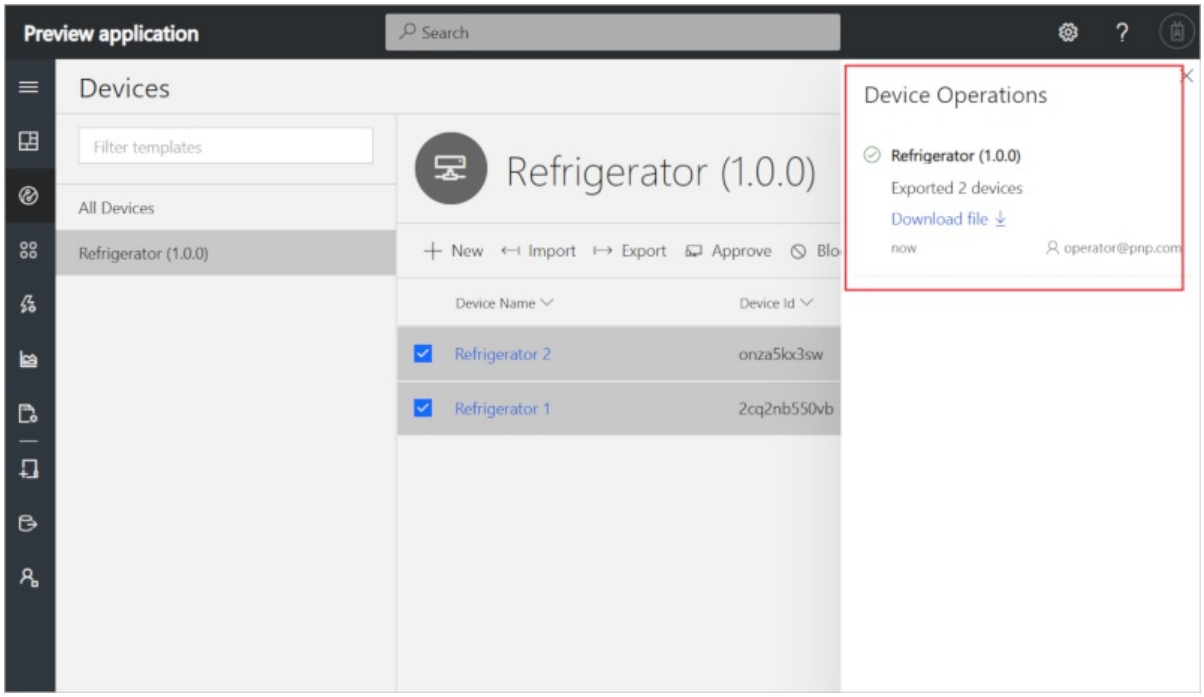
1. Choose **Devices** on the left pane.

2. On the left pane, choose the device template from which you want to export the devices.

3. Select the devices that you want to export and then select the **Export** action.



4. The export process starts. You can track the status using the Device Operations panel.
5. When the export completes, a success message is shown along with a link to download the generated file.
6. Select the **Download File** link to download the file to a local folder on the disk.



7. The exported CSV file contains the following columns: device ID, device name, device keys, and X509 certificate thumbprints:

   - IOTC_DEVICEID
   - IOTC_DEVICENAME
   - IOTC_SASKEY_PRIMARY
   - IOTC_SASKEY_SECONDARY
   - IOTC_X509THUMBPRINT_PRIMARY
   - IOTC_X509THUMBPRINT_SECONDARY

For more information about connection strings and connecting real devices to your IoT Central application, see Device connectivity in Azure IoT Central.

Delete a device

To delete either a real or simulated device from your Azure IoT Central application:

1. Choose **Devices** on the left pane.
2. Choose the device template of the device you want to delete.
3. Use the filter tools to filter and search for your devices. Check the box next to the devices to delete.
4. Choose **Delete**. You can track the status of this deletion in your Device Operations panel.

Change a property

Cloud properties are the device metadata associated with the device, such as city and serial number. Writeable properties control the behavior of a device. In other words, they enable you to provide inputs to your device. Device properties are set by the device and are read-only within IoT Central. You can view and update properties on the **Device Details** views for your device.

1. Choose **Devices** on the left pane.
2. Choose the device template of the device whose properties you want to change and select the target device.
3. Choose the view that contains properties for your device, this view enables you to input values and select **Save** at the top of the page. Here you see the properties your device has and their current values. Cloud properties and writeable properties have editable fields, while device properties are read-only. For writeable properties, you can see their sync status at the bottom of the field.
4. Modify the properties to the values you need. You can modify multiple properties at a time and update them all at the same time.
5. Choose **Save**. If you saved writeable properties, the values are sent to your device. When the device confirms the change for the writeable property, the status returns back to **synced**. If you saved a cloud property, the value is updated.

Next steps

Now that you've learned how to manage devices in your Azure IoT Central application, here is the suggested next step:

How to use device groups

Create and run a job in your Azure IoT Central application

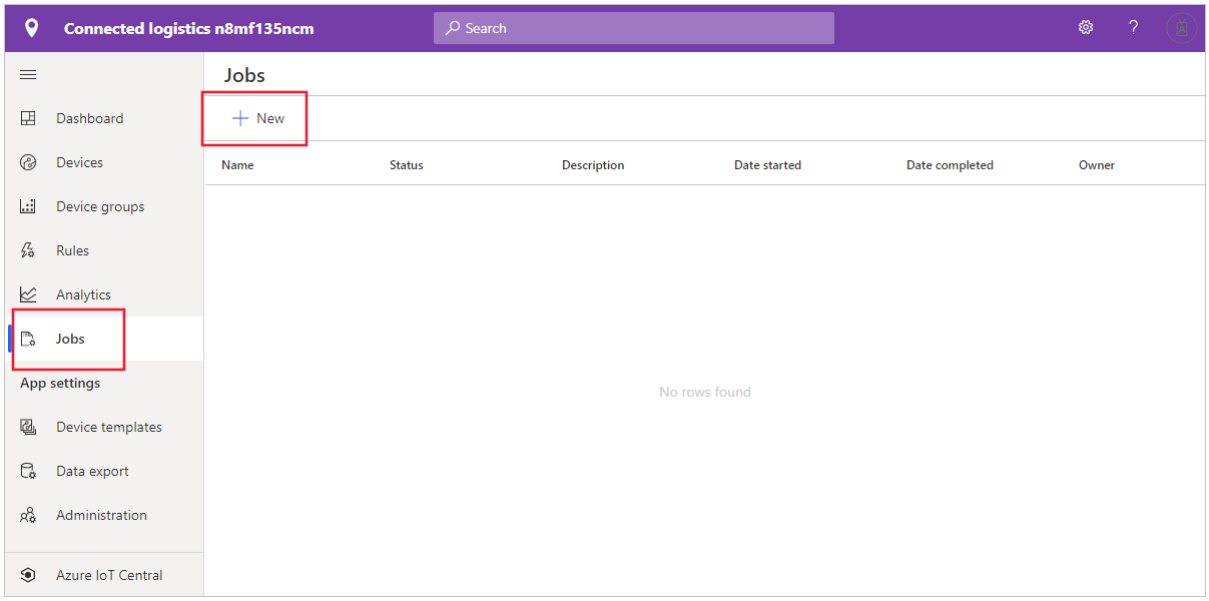
3/24/2020 • 3 minutes to read • Edit Online

You can use Microsoft Azure IoT Central to manage your connected devices at scale using jobs. Jobs let you do bulk updates to device properties and run commands. This article shows you how to get started using jobs in your own application.

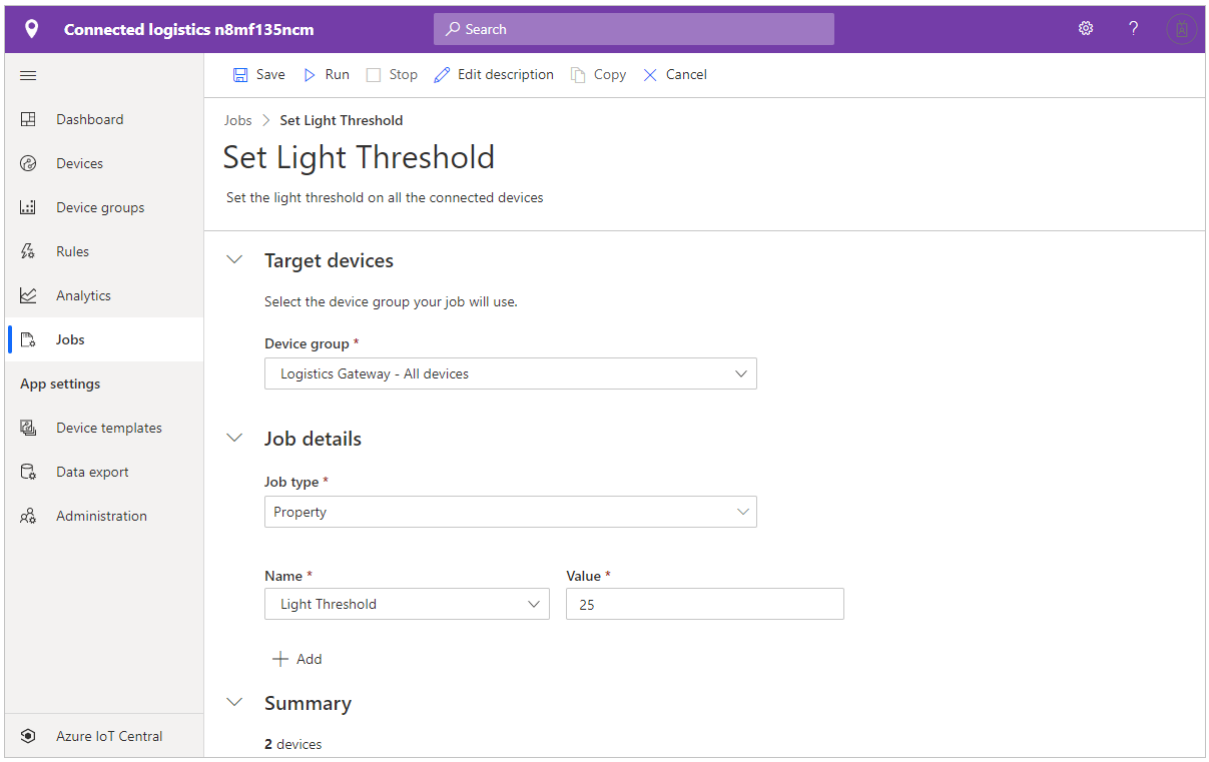
Create and run a job

This section shows you how to create and run a job. It shows you how to set the light threshold for a group of logistic gateway devices.

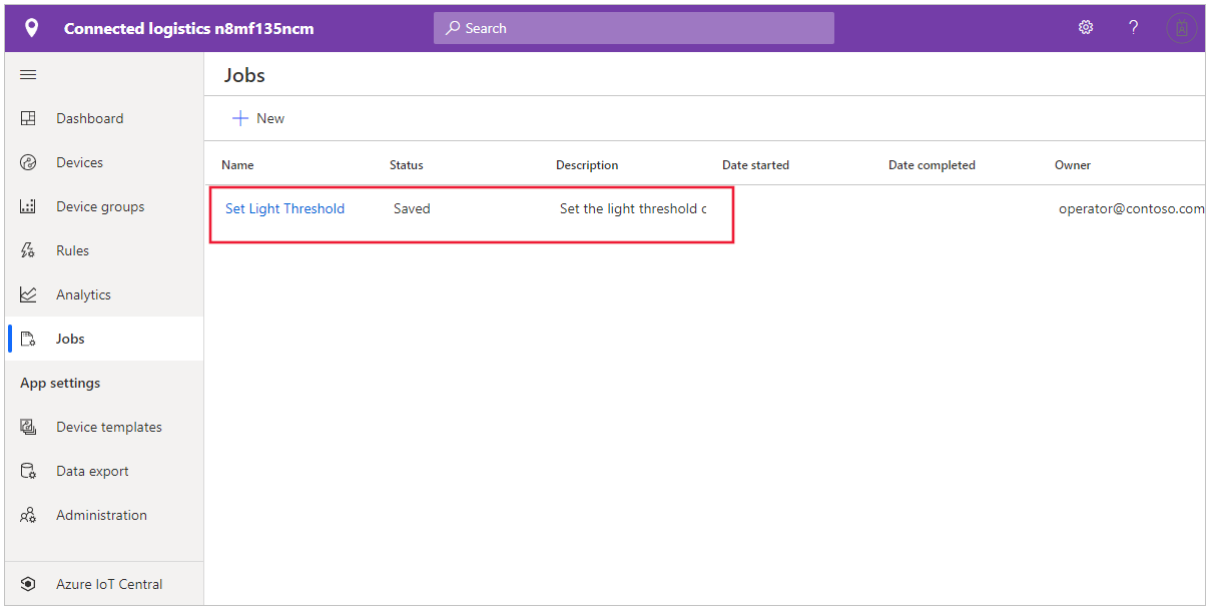
1. Navigate to **Jobs** from the left pane.
2. Select **+ New** to create a new job:



3. Enter a name and description to identify the job you're creating.
4. Select the target device group you want your job to apply to. You can see how many devices your job configuration applies to in the **Summary** section.
5. Next, choose either **Cloud property**, **Property** or **Command** as the type of job to configure. To set up a **Property** job configuration, select a property and set its new value. To set up a **Command**, choose the command to run. A property job can set multiple properties:

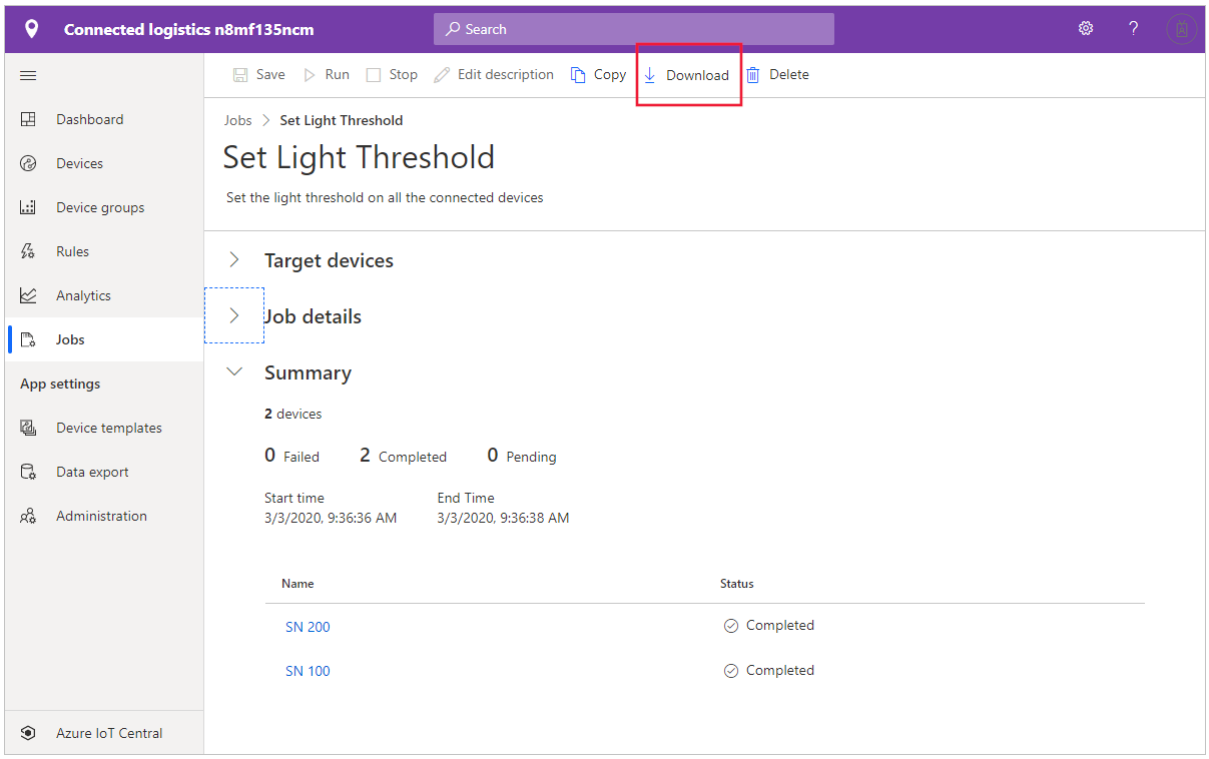


6. After creating your job, choose **Run** or **Save**. The job now appears on your main **Jobs** page. On this page, you can see your currently running job and the history of any previously run or saved jobs. Your saved job can be opened again at any time to continue editing it or to run it:



> **NOTE**
>
> You can view up 30 days of history for your previously run jobs.

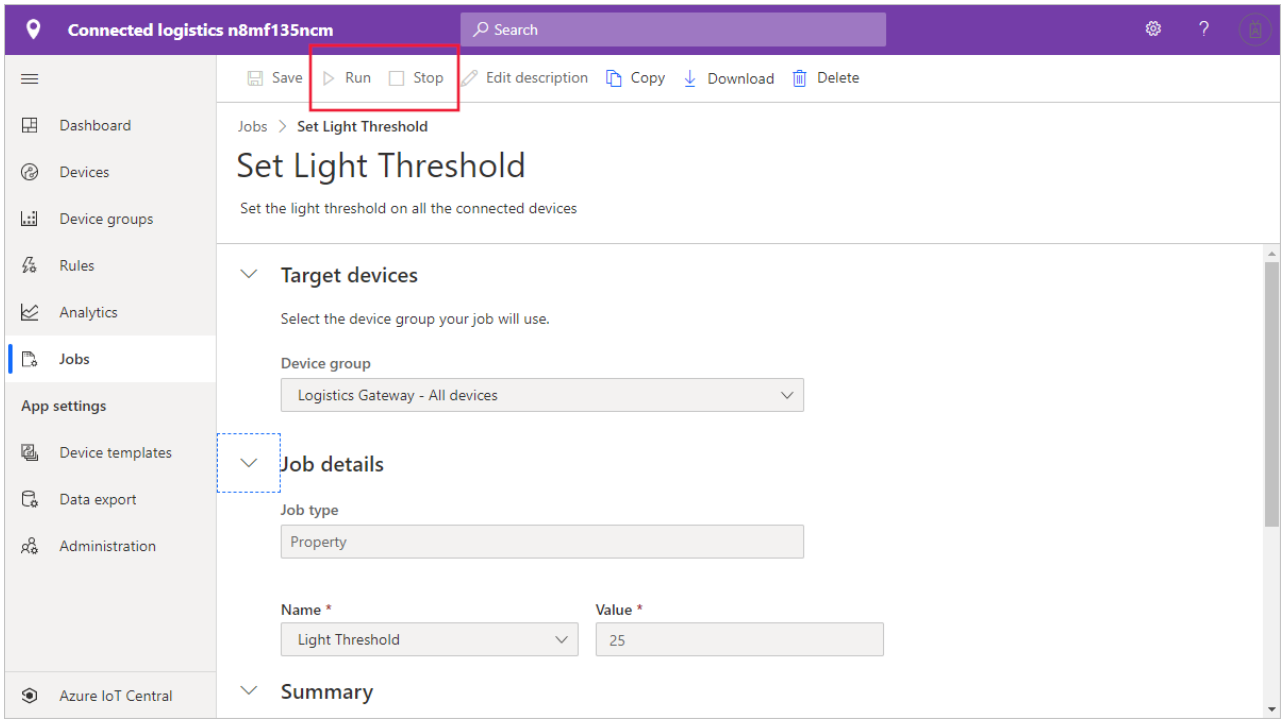
7. To get an overview of your job, select the job to view from the list. This overview contains the job details, devices, and device status values. From this overview, you can also select **Download Job Details** to download a CSV file of your job details, including the devices and their status values. This information can be useful for troubleshooting:



Manage a job

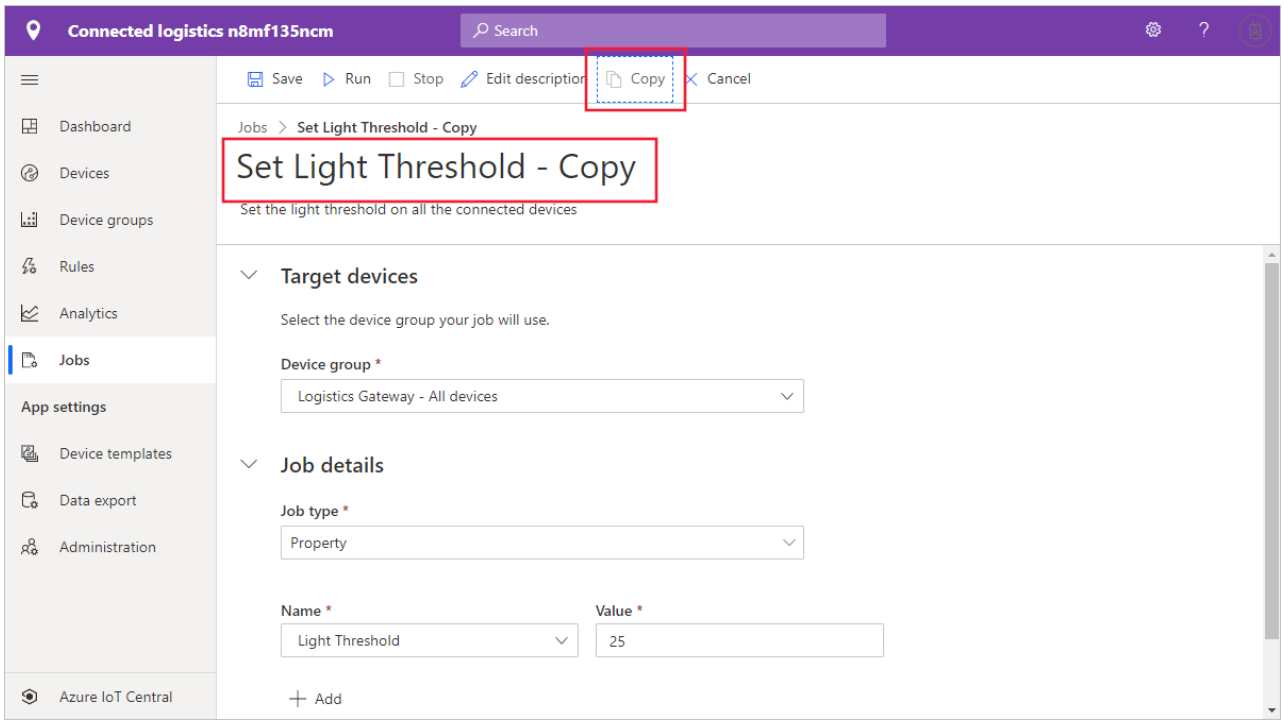
To stop one of your running jobs, open it and select **Stop**. The job status changes to reflect the job is stopped. The **Summary** section shows which devices have completed, failed, or are still pending.

To run a job that's currently stopped, select it, and then select **Run**. The job status changes to reflect the job is now running again. The **Summary** section continues to update with the latest progress.



Copy a job

To copy one of your existing jobs, select it on the **Jobs** page and select **Copy**. A copy of the job configuration opens for you to edit, and **Copy** is appended to the job name. You can save or run the new job:



View the job status

After a job is created, the **Status** column updates with the latest status message of the job. The following table lists the possible status values:

STATUS MESSAGE	STATUS MEANING
Completed	This job has been executed on all devices.
Failed	This job has failed and not fully executed on devices.
Pending	This job hasn't yet begun executing on devices.
Running	This job is currently executing on devices.
Stopped	This job has been manually stopped by a user.

The status message is followed by an overview of the devices in the job. The following table lists the possible device status values:

STATUS MESSAGE	STATUS MEANING
Succeeded	The number of devices that the job successfully executed on.
Failed	The number of devices that the job has failed to execute on.

View the device status

To view the status of the job and all the affected devices, open the job. To download a CSV file that includes the job details, including the list of devices and their status values, select **Download job details**. Next to each device name, you see one of the following status messages:

STATUS MESSAGE	STATUS MEANING
Completed	The job has been executed on this device.
Failed	The job has failed to execute on this device. The error message shows more information.
Pending	The job hasn't yet executed on this device.

NOTE

If a device has been deleted, you can't select the device. It displays as deleted with the device ID.

Next steps

Now that you've learned how to create jobs in your Azure IoT Central application, here are some next steps:

- Manage your devices
- Version your device template

Create a new device template version

3/24/2020 • 3 minutes to read • Edit Online

Azure IoT Central allows rapid development of IoT Applications. You can quickly iterate over your device template designs by adding, editing, or deleting device capabilities, views, and customizations. Once you have published your device template, the device capability model shows as **Published** with lock icons next to the model. In order to make changes to the device capability model, you will need to create a new version of the device template. Meanwhile the cloud properties, customizations, and views can all be edited at any time without needing to version the device template. Once you have saved any of these changes, you can publish the device template to make the latest changes available for the operator to view in Device Explorer.

> **NOTE**
> To learn more about how to create a device template see Set up and manage a device template

Add customizations to the device template without versioning

Certain elements of your device capabilities can be edited without needing to version your device template and interfaces. For example, some of these fields include display name, semantic type, minimum value, maximum value, decimal places, color, unit, display unit, comment, and description. To add one of these customizations:

1. Go to the **Device Templates** page.
2. Select the device template you wish to customize.
3. Choose the **Customize** tab.
4. All of the capabilities defined in your device capability model will be listed here. All of the fields you can edit here can be saved and used across your application, without needing to version your device template. If there are fields you wish to edit that are read-only, you will need to version your device template to change these. Select a field you wish to edit and enter in any new values.
5. Click **Save**. Now these values will override anything that was initially saved in your device template and will be used across the application.

Versioning a device template

Creating a new version of your device template will create a draft version of the template where the device capability model can be edited. Any published interfaces will remain published until they are individually versioned. In order to modify a published interface, you must first create a new device template version.

The device template should only be versioned when you are trying to edit a part of the device capability model that you can not edit in the customizations section of the device template.

In order to version a device template:

1. Go to the **Device Templates** page.
2. Select the device template you are trying to version.
3. Click the **Version** button at the top of the page and give the template a new name. We have suggested a new name for you which can be edited.
4. Click **Create**.
5. Now your device template is in draft mode. You will see your interfaces are still locked and must be individually versioned to be edited.

Versioning an interface

Versioning an interface allows you to add, update, and remove the capabilities inside the interface you had already created.

In order to version an interface:

1. Go to the **Device Templates** page.
2. Select the device template you have in a draft mode.
3. Select the interface that is in published mode that you wish to version and edit.
4. Click the **Version** button at the top of the interface page.
5. Click **Create**.
6. Now your interface is in draft mode. You will be able to add or edit capabilities to your interface without breaking existing customizations and views.

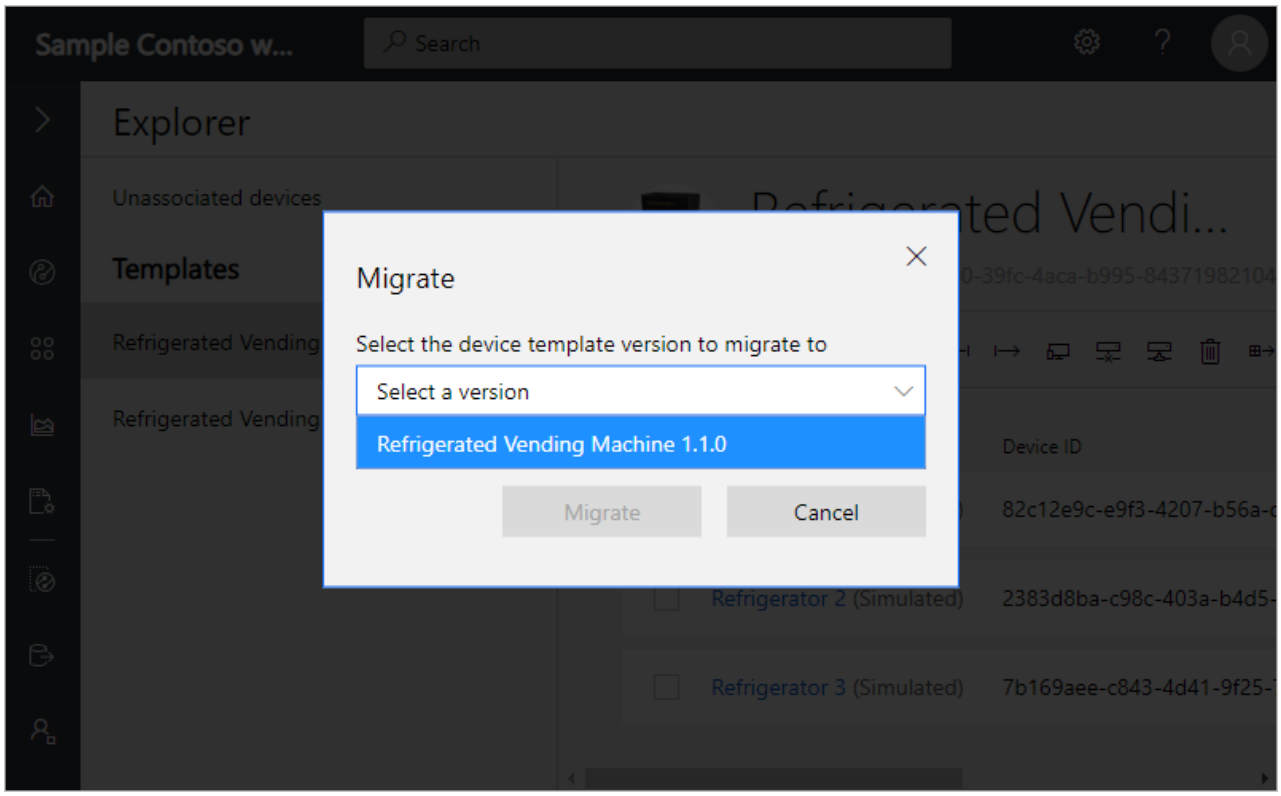
> **NOTE**
> Standard interfaces published by Azure IoT can not be versioned or edited. These standard interfaces are used for device certification.

> **NOTE**
> Once the interface has been published, you can not delete any of it's capabilities even in a draft mode. Capabilities can only be edited or added to the interface in draft mode.

Migrate a device across device template versions

You can create multiple versions of the device template. Over time, you will have multiple connected devices using these device templates. You can migrate devices from one version of your device template to another. The following steps describe how to migrate a device:

1. Go to the **Device Explorer** page.
2. Select the device you need to migrate to another version.
3. Choose **Migrate**.
4. Select the device template with the version number you want to migrate the device to and choose **Migrate**.



Next steps

Now that you have learned how to use device template versions in your Azure IoT Central application, here is the suggested next step:

How to create telemetry rules

How to use analytics to analyze device data

3/24/2020 • 4 minutes to read • Edit Online

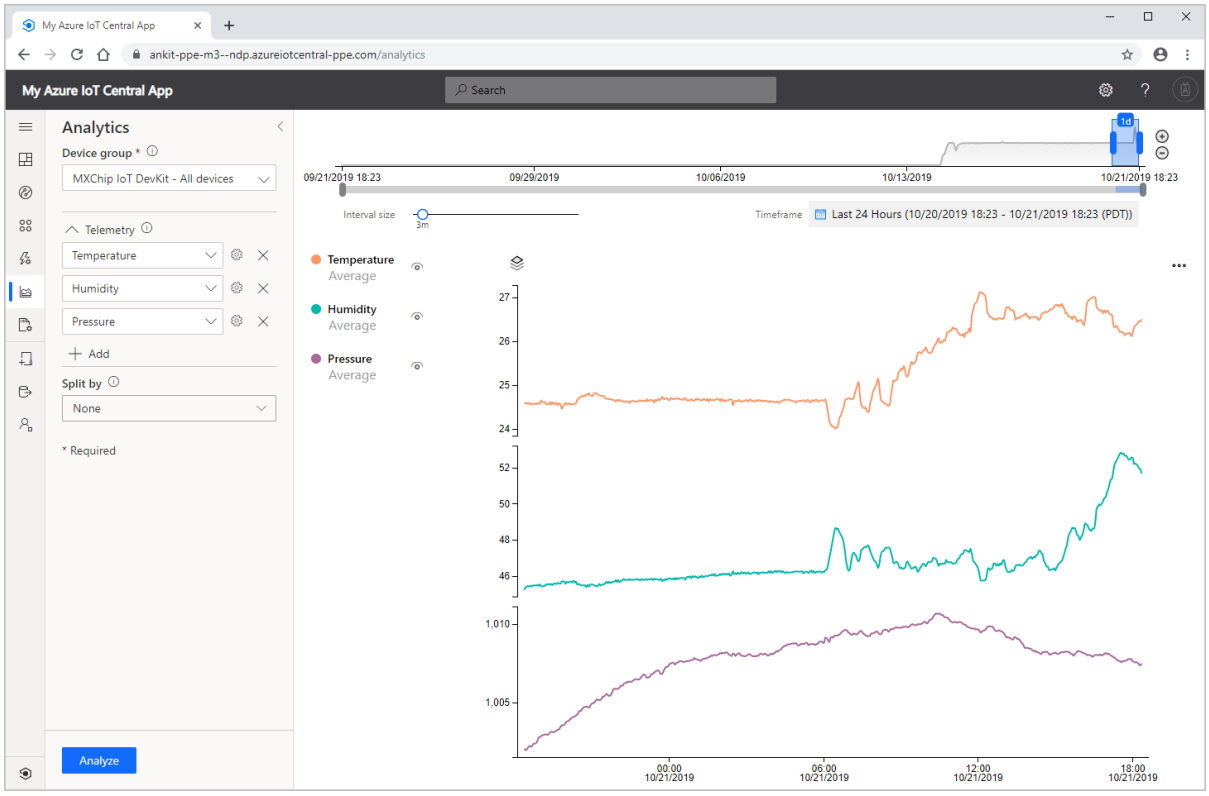
This article applies to operators, builders, and administrators.

Azure IoT Central provides rich analytics capabilities to analyze historical trends and correlate various telemetries from your devices. To get started, visit **Analytics** on the left pane.

Understanding the Analytics UI

Analytics user interface is made of three main components:

- **Data configuration panel:** On the configuration panel, start by selecting the device group for which you want to analyze the data. Next, select the telemetry that you want to analyze and select the aggregation method for each telemetry. **Split By** control helps to group the data by using the device properties as dimensions.
- **Time control:** Time control is used to select the duration for which you want to analyze the data. You can drag either end of the time slider to select the time span. Time control also has an **Interval size** slider that controls the bucket or the interval size used to aggregate the data.
- **Chart control:** Chart control visualizes the data as a line chart. You can toggle the visibility of specific lines by interacting with the chart legend.



Querying your data

You'll need to start by choosing a **device group**, and the telemetry that you want to analyze. Once you're done, select **Analyze** to start visualizing your data.

- **Device group:** A device group is a user-defined group of your devices. For example, all Refrigerators in Oakland, or All version 2.0 wind turbines.

- **Telemetry:** Select the telemetry that you want to analyze and explore. You can select multiple telemetries to analyze together. Default aggregation method is set to Average for numerical and Count for string data-type respectively. Supported aggregation methods for Numeric data types are Average, Maximum, Minimum, Count and, Sum. Supported aggregation methods for string data type are count.

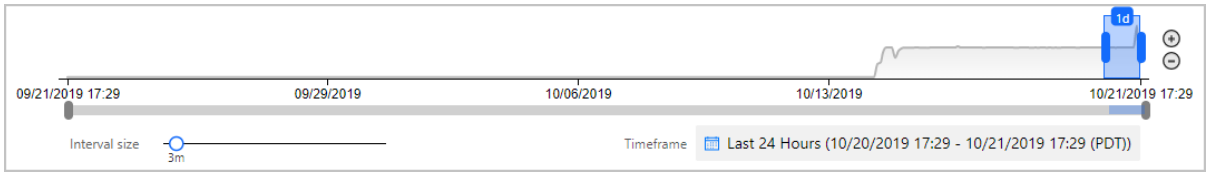
- **Split by:** 'Split by' control helps to group the data by using the device properties as dimensions. Values of the device and cloud properties are joined along with the telemetry as and when it is sent by the device. If the cloud or device property has been updated, then you will see the telemetry grouped by different values on the chart.

  > **TIP**
  > To view data for each device separately, select Device Id in the 'Split by' control.

Interacting with your data

Once you've queried your data, you can start visualizing it on the line chart. You can show/hide telemetry, change the time duration, view telemetry in a data grid.

- **Time editor panel:** By default we'll retrieve data from the past one day. You can drag either end of the time slider to change the time duration. You can also use the calendar control to select one of the predefined time buckets or select a custom time range. Time control also has an **Interval size** slider that controls the bucket or the interval size used to aggregate the data.



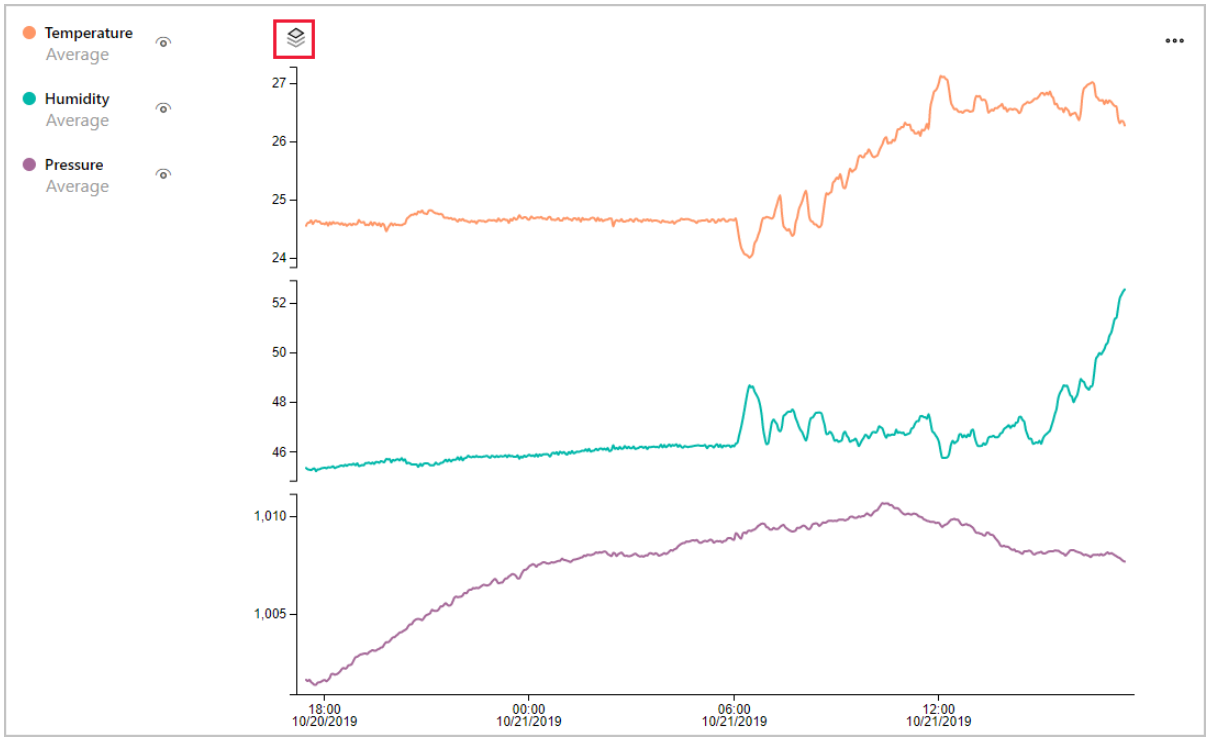
  - **Inner date range slider tool**: Use the two endpoint controls by dragging them over the time span you want. This inner date range is constrained by the outer date range slider control.
  - **Outer date range slider control**: Use the endpoint controls to select the outer date range, which will be available for your inner date range control.
  - **Increase and decrease date range buttons**: Increase or decrease your time span by selecting either button for the interval you want.
  - **Interval-size slider**: Use it to zoom in and out of intervals over the same time span. This action provides more precise control of movement between large slices of time. You can use it to see granular, high-resolution views of your data, even down to milliseconds. The slider's default starting point is set as the most optimal view of the data from your selection, which balances resolution, query speed, and granularity.
  - **Date range picker**: With this web control, you can easily select the date and time ranges you want. You can also use the control to switch between different time zones. After you make the changes to apply to your current workspace, select Save.

  > **TIP**
  > Interval size is determined dynamically based on the selected time span. Smaller time spans will enable aggregating the data into very granular intervals of up to a few seconds.

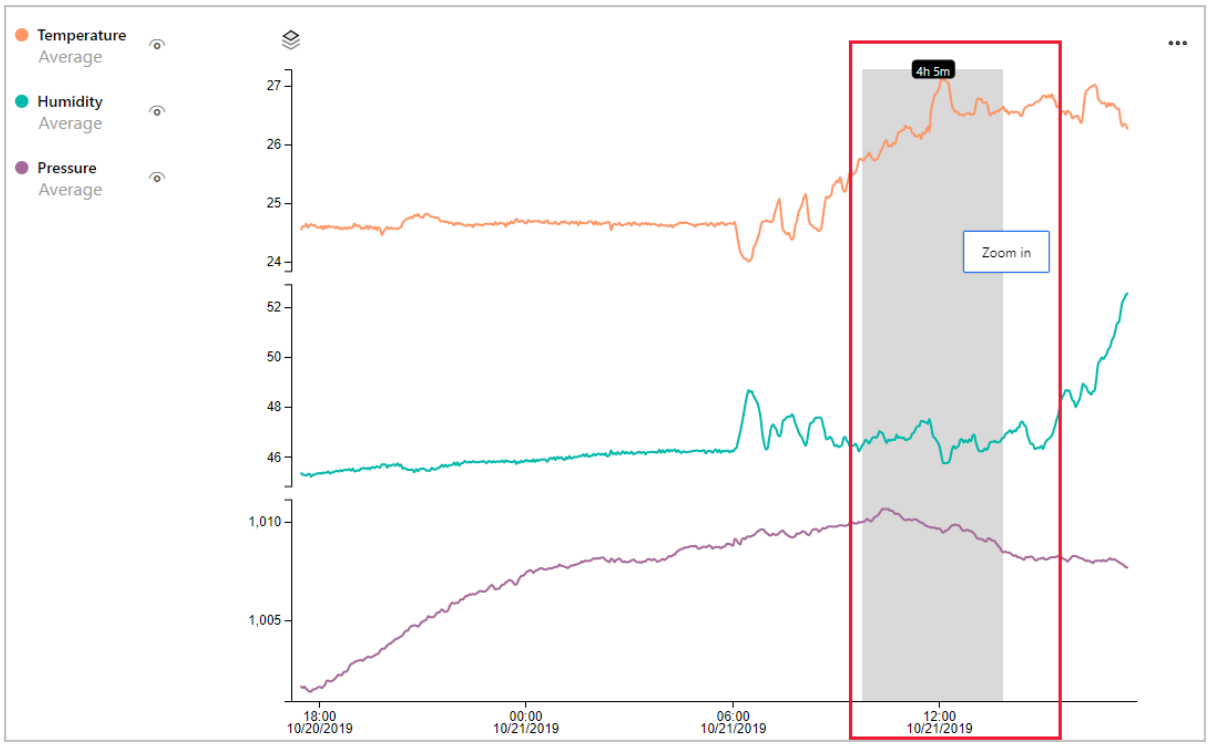
- **Chart Legend:** Chart legend shows the selected telemetry on the chart. You can hover over each item on

the legend to bring it into focus on the chart. When using 'Split By', the telemetry is grouped by the respective values of the selected dimension. You can toggle the visibility of each specific telemetry or the whole group by clicking on the group name.

- **Y-axis format control:** y-axis mode cycles through the available y-axis view options. This control is available only when different telemetries are being visualized. You can set the y-axis by choosing from one of three modes:
  - **Stacked:** A graph for every telemetry is stacked and each of the graphs have their own y-axis. This mode is set as default.
  - **Shared:** A graph for every telemetry is plotted against the same y-axis.
  - **Overlap:** Use it to stack multiple lines on the same y-axis, with the y-axis data changing based on the selected line.

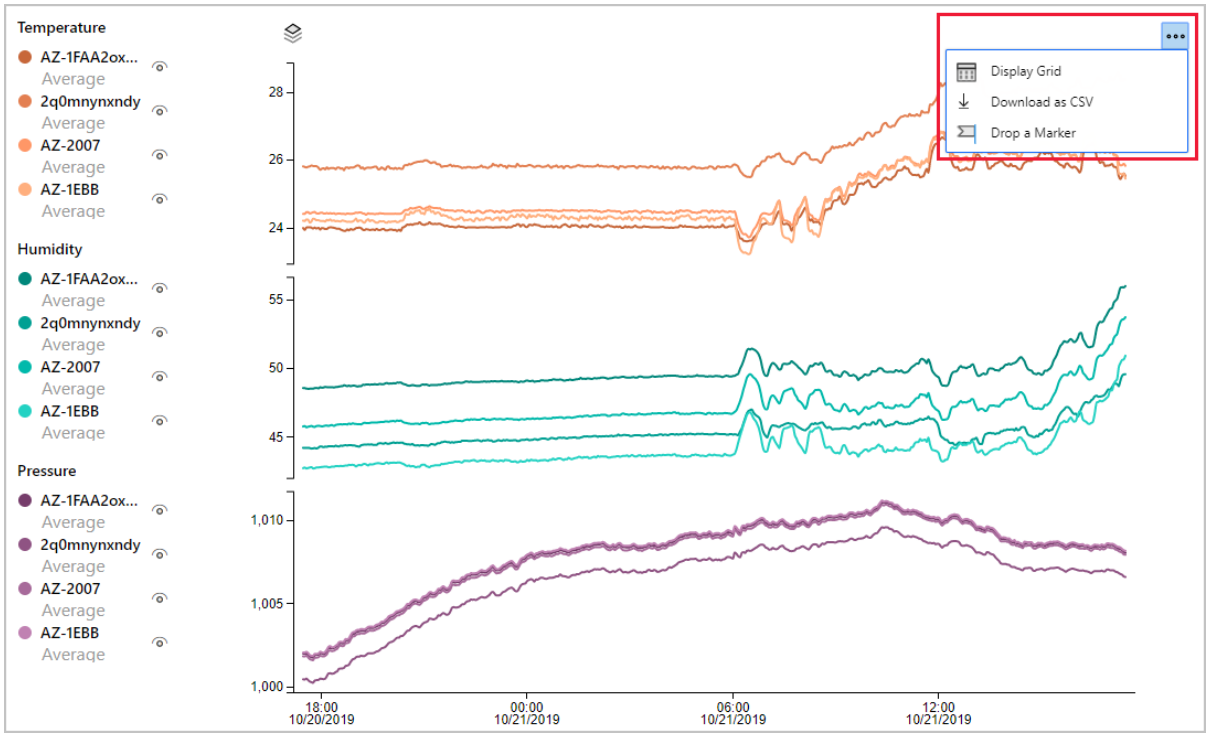


- **Zoom control:** Zoom lets you drill further into your data. If you find a time period you'd like to focus on within your result set, use your mouse pointer to grab the area and then drag it to the endpoint of your choice. Then right click on the selected area and click Zoom.



Under the ellipsis, there are more chart controls to interact with the data.

- **Display Grid:** Your results are available in a table format, enabling you to view the specific value for each data point.
- **Drop a Marker**: 'Drop Marker' control provides a way to anchor certain data points on the chart. It is useful when you are trying to compare data for multiple lines across different time periods.



Create webhook actions on rules in Azure IoT Central

3/24/2020 • 2 minutes to read • Edit Online

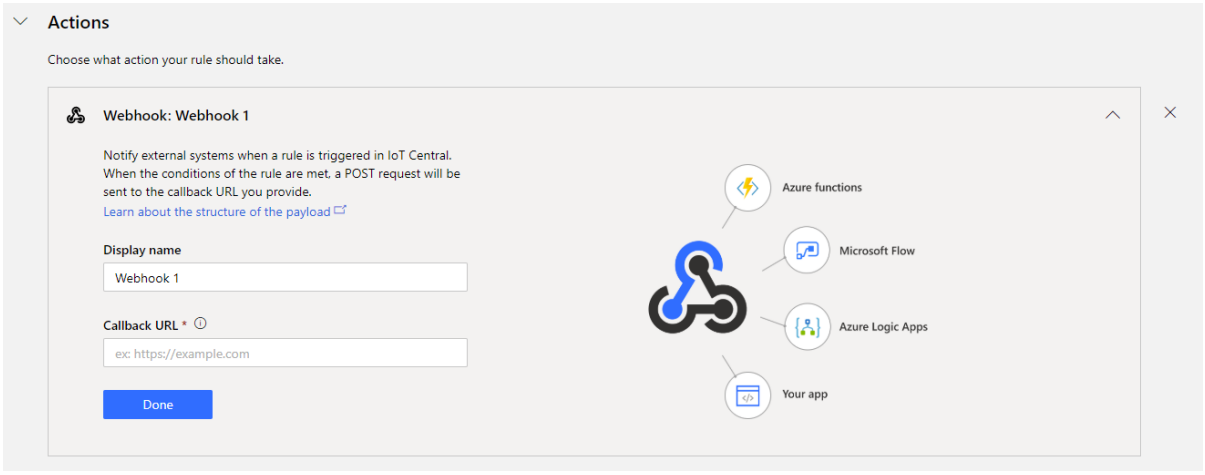
This topic applies to builders and administrators.

Webhooks enable you to connect your IoT Central app to other applications and services for remote monitoring and notifications. Webhooks automatically notify other applications and services you connect whenever a rule is triggered in your IoT Central app. Your IoT Central app sends a POST request to the other application's HTTP endpoint whenever a rule is triggered. The payload contains device details and rule trigger details.

Set up the webhook

In this example, you connect to RequestBin to get notified when rules fire using webhooks.

1. Open RequestBin.
2. Create a new RequestBin and copy the **Bin URL**.
3. Create a telemetry rule. Save the rule and add a new action.



4. Choose the webhook action and provide a display name and paste the Bin URL as the Callback URL.
5. Save the rule.

Now when the rule is triggered, you see a new request appear in RequestBin.

Payload

When a rule is triggered, an HTTP POST request is made to the callback URL containing a json payload with the telemetry, device, rule, and application details. The payload could look like the following:

```
{
    "id": "<id>",
    "displayName": "Webhook 1",
    "timestamp": "2019-10-24T18:27:13.538Z",
    "rule": {
        "id": "<id>",
        "displayName": "High temp alert",
        "enabled": true
    },
    "device": {
        "id": "mx1",
        "displayName": "MXChip IoT DevKit - mx1",
        "instanceOf": "<device-template-id>",
        "simulated": true,
        "provisioned": true,
        "approved": true
    },
    "data": [{
        "@id": "<id>",
        "@type": ["Telemetry"],
        "name": "temperature",
        "displayName": "Temperature",
        "value": 66.27310467496761,
        "interfaceInstanceName": "sensors"
    }],
    "application": {
        "id": "<id>",
        "displayName": "x - Store Analytics Checkout---PnP",
        "subdomain": "<subdomain>",
        "host": "<host>"
    }
}
```


Known limitations

Currently there is no programmatic way of subscribing/unsubscribing from these webhooks through an API.

If you have ideas for how to improve this feature, post your suggestions to our User voice forum.

Next steps

Now that you've learned how to set up and use webhooks, the suggested next step is to explore configuring Azure Monitor Action Groups.

Export IoT data to destinations in Azure

4/7/2020 • 13 minutes to read • Edit Online

This topic applies to administrators.

This article describes how to use the data export feature in Azure IoT Central. This feature lets you export your data continuously to **Azure Event Hubs**, **Azure Service Bus**, or **Azure Blob storage** instances. Data export uses the JSON format and can include telemetry, device information, and device template information. Use the exported data for:

- Warm-path insights and analytics. This option includes triggering custom rules in Azure Stream Analytics, triggering custom workflows in Azure Logic Apps, or passing it through Azure Functions to be transformed.
- Cold-path analytics such as training models in Azure Machine Learning or long-term trend analysis in Microsoft Power BI.

> **NOTE**
> When you turn on data export, you get only the data from that moment onward. Currently, data can't be retrieved for a time when data export was off. To retain more historical data, turn on data export early.

Prerequisites

You must be an administrator in your IoT Central application, or have Data export permissions.

Set up export destination

Your export destination must exist before you configure your data export.

Create Event Hubs namespace

If you don't have an existing Event Hubs namespace to export to, follow these steps:

1. Create a new Event Hubs namespace in the Azure portal. You can learn more in Azure Event Hubs docs.
2. Choose a subscription. You can export data to other subscriptions that aren't in the same subscription as your IoT Central application. You connect using a connection string in this case.
3. Create an event hub in your Event Hubs namespace. Go to your namespace, and select **+ Event Hub** at the top to create an event hub instance.

Create Service Bus namespace

If you don't have an existing Service Bus namespace to export to, follow these steps:

1. Create a new Service Bus namespace in the Azure portal. You can learn more in Azure Service Bus docs.
2. Choose a subscription. You can export data to other subscriptions that aren't in the same subscription as your IoT Central application. You connect using a connection string in this case.
3. To create a queue or topic to export to, go to your Service Bus namespace, and select **+ Queue** or **+ Topic**.

When you choose Service Bus as an export destination, the queues and topics must not have Sessions or Duplicate Detection enabled. If either of those options are enabled, some messages won't arrive in your queue or topic.

Create storage account

If you don't have an existing Azure storage account to export to, follow these steps:

1. Create a new storage account in the Azure portal. You can learn more about creating new Azure Blob storage accounts or Azure Data Lake Storage v2 storage accounts. Data export can only write data to storage accounts that support block blobs. The following list shows the known compatible storage account types:

PERFORMANCE TIER	ACCOUNT TYPE
Standard	General Purpose V2
Standard	General Purpose V1
Standard	Blob storage
Premium	Block Blob storage

2. Create a container in your storage account. Go to your storage account. Under **Blob Service**, select **Browse Blobs**. Select **+ Container** at the top to create a new container.

Set up data export

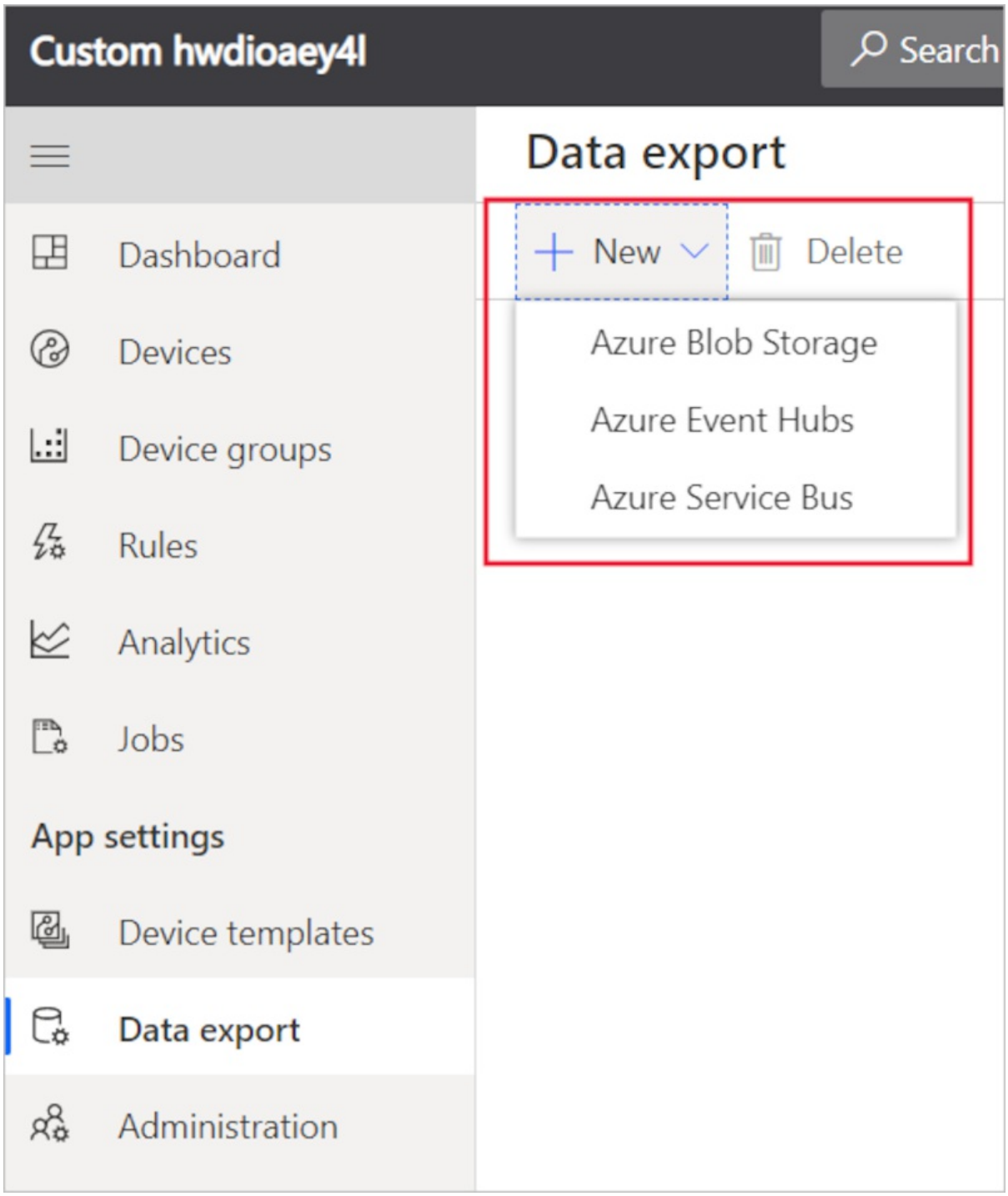
Now that you have a destination to export data to, follow these steps to set up data export.

1. Sign in to your IoT Central application.

2. In the left pane, select **Data export**.

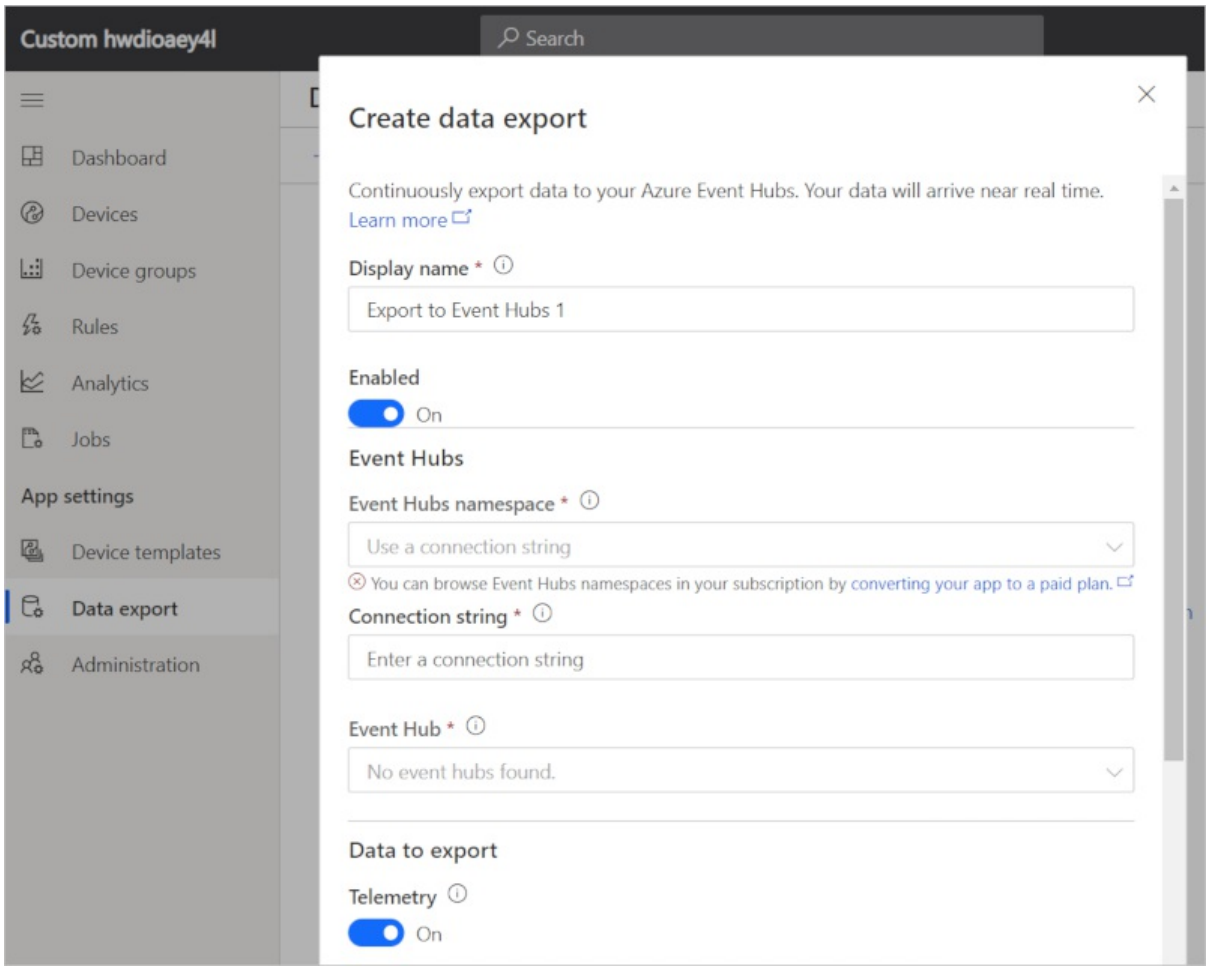
> **TIP**
> If you don't see **Data export** in the left pane, then you don't have permissions to configure data export in your app. Talk to an administrator to set up data export.

3. Select the **+ New** button in the top right. Choose one of **Azure Event Hubs**, **Azure Service Bus**, or **Azure Blob storage** as the destination of your export. The maximum number of exports per application is five.



4. In the drop-down list box, select your **Event Hubs namespace**, **Service Bus namespace**, **Storage Account namespace**, or **Enter a connection string**.

   - You only see storage accounts, Event Hubs namespaces, and Service Bus namespaces in the same subscription as your IoT Central application. If you want to export to a destination outside of this subscription, choose **Enter a connection string** and see the next step.
   - For apps created using the free pricing plan, the only way to configure data export is through a connection string. Apps on the free pricing plan don't have an associated Azure subscription.



5. (Optional) If you chose **Enter a connection string**, a new box appears for you to paste your connection string. To get the connection string for your:

   - Event Hubs or Service Bus, go to the namespace in the Azure portal:
     - Under **Settings**, select **Shared Access Policies**
     - Choose the default **RootManageSharedAccessKey** or create a new one
     - Copy either the primary or secondary connection string
   - Storage account, go to the storage account in the Azure portal:
     - Under **Settings**, select **Access keys**
     - Copy either the key1 connection string or the key2 connection string

6. Choose an event hub, queue, topic, or container from the drop-down list box.

7. Under **Data to export**, choose the types of data to export by setting the type to **On**.

8. To turn on data export, make sure the **Enabled** toggle is **On**. Select **Save**.

9. After a few minutes, your data appears in your chosen destination.

Export contents and format

Exported telemetry data contains the entirety of the message your devices sent to IoT Central, not just the telemetry values themselves. Exported devices data contains changes to properties and metadata of all devices, and exported device templates contains changes to all device templates.

For Event Hubs and Service Bus, data is exported in near-realtime. The data is in the `body` property and is in JSON format. See below for examples.

For Blob storage, data is exported once per minute, with each file containing the batch of changes since the last exported file. Exported data is placed in three folders in JSON format. The default paths in your storage account are:

- Telemetry: *{container}/{app-id}/telemetry/{YYYY}/{MM}/{dd}/{hh}/{mm}/{filename}*
- Devices: *{container}/{app-id}/devices/{YYYY}/{MM}/{dd}/{hh}/{mm}/{filename}*
- Device templates: *{container}/{app-id}/deviceTemplates/{YYYY}/{MM}/{dd}/{hh}/{mm}/{filename}*

To browse the exported files in the Azure portal, navigate to the file and select the **Edit blob** tab.

Telemetry

For Event Hubs and Service Bus, IoT Central exports a new message quickly after it receives the message from a device. Each exported message contains the full message the device sent in the body property in JSON format.

For Blob storage, messages are batched and exported once per minute. The exported files use the same format as the message files exported by IoT Hub message routing to blob storage.

> **NOTE**
> For Blob storage, ensure that your devices are sending messages that have `contentType: application/JSON` and `contentEncoding:utf-8` (or `utf-16`, `utf-32`). See the IoT Hub documentation for an example.

The device that sent the telemetry is represented by the device ID (see the following sections). To get the names of the devices, export device data and correlate each message by using the **connectionDeviceId** that matches the **deviceId** of the device message.

The following example shows a message received from an event hub or Service Bus queue or topic:

```
{
  "temp":81.129693132351775,
  "humid":59.488071477541247,
  "EventProcessedUtcTime":"2020-04-07T09:41:15.2877981Z",
  "PartitionId":0,
  "EventEnqueuedUtcTime":"2020-04-07T09:38:32.7380000Z"
}
```

This message doesn't include the device ID of the sending device.

To retrieve the device ID from the message data in an Azure Stream Analytics query, use the GetMetadataPropertyValue function. For an example, see the query in Extend Azure IoT Central with custom rules using Stream Analytics, Azure Functions, and SendGrid.

To retrieve the device ID in an Azure Databricks or Apache Spark workspace, use systemProperties. For an example, see the Databricks workspace in Extend Azure IoT Central with custom analytics using Azure Databricks.

The following example shows a record exported to blob storage:

```
{
  "EnqueuedTimeUtc":"2019-09-26T17:46:09.8870000Z",
  "Properties":{

  },
  "SystemProperties":{
    "connectionDeviceId":"<deviceid>",
    "connectionAuthMethod":"
{\"scope\":\"device\",\"type\":\"sas\",\"issuer\":\"iothub\",\"acceptingIpFilterRule\":null}",
    "connectionDeviceGenerationId":"637051167384630591",
    "contentType":"application/json",
    "contentEncoding":"utf-8",
    "enqueuedTime":"2019-09-26T17:46:09.8870000Z"
  },
  "Body":{
    "temp":49.91322758395974,
    "humid":49.61214852573155,
    "pm25":25.87332214661367
  }
}
```


Devices

Each message or record in a snapshot represents one or more changes to a device and its device and cloud properties since the last exported message. The message includes the:

- `id` of the device in IoT Central
- `displayName` of the device
- Device template ID in `instanceOf`
- `simulated` flag, true if the device is a simulated device
- `provisioned` flag, true if the device has been provisioned
- `approved` flag, true if the device has been approved to send data
- Property values
- `properties` including device and cloud properties values

Deleted devices aren't exported. Currently, there are no indicators in exported messages for deleted devices.

For Event Hubs and Service Bus, IoT Central sends messages containing device data to your event hub or Service Bus queue or topic in near real time.

For Blob storage, a new snapshot containing all the changes since the last one written is exported once per minute.

The following example message shows information about devices and properties data in an event hub or Service Bus queue or topic:

```
{
  "body":{
    "id": "<device Id>",
    "etag": "<etag>",
    "displayName": "Sensor 1",
    "instanceOf": "<device template Id>",
    "simulated": false,
    "provisioned": true,
    "approved": true,
    "properties": {
        "sensorComponent": {
            "setTemp": "30",
            "fwVersion": "2.0.1",
            "status": { "first": "first", "second": "second" },
            "$metadata": {
                "setTemp": {
                    "desiredValue": "30",
                    "desiredVersion": 3,
                    "desiredTimestamp": "2020-02-01T17:15:08.9284049Z",
                    "ackVersion": 3
                },
                "fwVersion": { "ackVersion": 3 },
                "status": {
                    "desiredValue": {
                        "first": "first",
                        "second": "second"
                    },
                    "desiredVersion": 2,
                    "desiredTimestamp": "2020-02-01T17:15:08.9284049Z",
                    "ackVersion": 2
                }
            },

        }
    },
    "installDate": { "installDate": "2020-02-01" }
},
  "annotations":{
    "iotcentral-message-source":"devices",
    "x-opt-partition-key":"<partitionKey>",
    "x-opt-sequence-number":39740,
    "x-opt-offset":"<offset>",
    "x-opt-enqueued-time":1539274959654
  },
  "partitionKey":"<partitionKey>",
  "sequenceNumber":39740,
  "enqueuedTimeUtc":"2020-02-01T18:14:49.3820326Z",
  "offset":"<offset>"
}
```


This snapshot is an example message that shows devices and properties data in Blob storage. Exported files contain a single line per record.

```
{
  "id": "<device Id>",
  "etag": "<etag>",
  "displayName": "Sensor 1",
  "instanceOf": "<device template Id>",
  "simulated": false,
  "provisioned": true,
  "approved": true,
  "properties": {
      "sensorComponent": {
          "setTemp": "30",
          "fwVersion": "2.0.1",
          "status": { "first": "first", "second": "second" },
          "$metadata": {
              "setTemp": {
                  "desiredValue": "30",
                  "desiredVersion": 3,
                  "desiredTimestamp": "2020-02-01T17:15:08.9284049Z",
                  "ackVersion": 3
              },
              "fwVersion": { "ackVersion": 3 },
              "status": {
                  "desiredValue": {
                      "first": "first",
                      "second": "second"
                  },
                  "desiredVersion": 2,
                  "desiredTimestamp": "2020-02-01T17:15:08.9284049Z",
                  "ackVersion": 2
              }
          },

      }
  },
  "installDate": { "installDate": "2020-02-01" }
}
```

Device templates

Each message or snapshot record represents one or more changes to a published device template since the last exported message. Information sent in each message or record includes:

- `id` of the device template that matches the `instanceOf` of the devices stream above
- `displayName` of the device template
- The device `capabilityModel` including its `interfaces`, and the telemetry, properties, and commands definitions
- `cloudProperties` definitions
- Overrides and initial values, inline with the `capabilityModel`

Deleted device templates aren't exported. Currently, there are no indicators in exported messages for deleted device templates.

For Event Hubs and Service Bus, IoT Central sends messages containing device template data to your event hub or Service Bus queue or topic in near real time.

For Blob storage, a new snapshot containing all the changes since the last one written is exported once per minute.

This example shows a message about device templates data in event hub or Service Bus queue or topic:

```
{
  "body":{
      "id": "<device template id>",
      "etag": "<etag>"
```

```
    "etag": "<etag>",
    "types": ["DeviceModel"],
    "displayName": "Sensor template",
    "capabilityModel": {
        "@id": "<capability model id>",
        "@type": ["CapabilityModel"],
        "contents": [],
        "implements": [
            {
                "@id": "<component Id>",
                "@type": ["InterfaceInstance"],
                "name": "sensorComponent",
                "schema": {
                    "@id": "<interface Id>",
                    "@type": ["Interface"],
                    "displayName": "Sensor interface",
                    "contents": [
                        {
                            "@id": "<id>",
                            "@type": ["Telemetry"],
                            "displayName": "Humidity",
                            "name": "humidity",
                            "schema": "double"
                        },
                        {
                            "@id": "<id>",
                            "@type": ["Telemetry", "SemanticType/Event"],
                            "displayName": "Error event",
                            "name": "error",
                            "schema": "integer"
                        },
                        {
                            "@id": "<id>",
                            "@type": ["Property"],
                            "displayName": "Set temperature",
                            "name": "setTemp",
                            "writable": true,
                            "schema": "integer",
                            "unit": "Units/Temperature/fahrenheit",
                            "initialValue": "30"
                        },
                        {
                            "@id": "<id>",
                            "@type": ["Property"],
                            "displayName": "Firmware version read only",
                            "name": "fwversion",
                            "schema": "string"
                        },
                        {
                            "@id": "<id>",
                            "@type": ["Property"],
                            "displayName": "Display status",
                            "name": "status",
                            "writable": true,
                            "schema": {
                                "@id": "urn:testInterface:status:obj:ka8iw8wka:1",
                                "@type": ["Object"]
                            }
                        },
                        {
                            "@id": "<id>",
                            "@type": ["Command"],
                            "commandType": "synchronous",
                            "request": {
                                "@id": "<id>",
                                "@type": ["SchemaField"],
                                "displayName": "Configuration",
                                "name": "config",
                                "schema": "string"
```

```
                            },
                            "response": {
                                "@id": "<id>",
                                "@type": ["SchemaField"],
                                "displayName": "Response",
                                "name": "response",
                                "schema": "string"
                            },
                            "displayName": "Configure sensor",
                            "name": "sensorConfig"
                        }
                    ]
                }
            }
        ],
        "displayName": "Sensor capability model"
    },
    "solutionModel": {
        "@id": "<id>",
        "@type": ["SolutionModel"],
        "cloudProperties": [
            {
                "@id": "<id>",
                "@type": ["CloudProperty"],
                "displayName": "Install date",
                "name": "installDate",
                "schema": "dateTime",
                "valueDetail": {
                    "@id": "<id>",
                    "@type": ["ValueDetail/DateTimeValueDetail"]
                }
            }
        ]
    }
  },
    "annotations":{
      "iotcentral-message-source":"deviceTemplates",
      "x-opt-partition-key":"<partitionKey>",
      "x-opt-sequence-number":25315,
      "x-opt-offset":"<offset>",
      "x-opt-enqueued-time":1539274985085
    },
    "partitionKey":"<partitionKey>",
    "sequenceNumber":25315,
    "enqueuedTimeUtc":"2019-10-02T16:23:05.085Z",
    "offset":"<offset>"
  }
}
```

This example snapshot shows a message that contains device and properties data in Blob storage. Exported files contain a single line per record.

```
{
      "id": "<device template id>",
      "etag": "<etag>",
      "types": ["DeviceModel"],
      "displayName": "Sensor template",
      "capabilityModel": {
          "@id": "<capability model id>",
          "@type": ["CapabilityModel"],
          "contents": [],
          "implements": [
              {
                  "@id": "<component Id>",
                  "@type": ["InterfaceInstance"],
                  "name": "Sensor component",
                  "schema": {
```

```
                    "@id": "<interface Id>",
                    "@type": ["Interface"],
                    "displayName": "Sensor interface",
                    "contents": [
                        {
                            "@id": "<id>",
                            "@type": ["Telemetry"],
                            "displayName": "Humidity",
                            "name": "humidity",
                            "schema": "double"
                        },
                        {
                            "@id": "<id>",
                            "@type": ["Telemetry", "SemanticType/Event"],
                            "displayName": "Error event",
                            "name": "error",
                            "schema": "integer"
                        },
                        {
                            "@id": "<id>",
                            "@type": ["Property"],
                            "displayName": "Set temperature",
                            "name": "setTemp",
                            "writable": true,
                            "schema": "integer",
                            "unit": "Units/Temperature/fahrenheit",
                            "initialValue": "30"
                        },
                        {
                            "@id": "<id>",
                            "@type": ["Property"],
                            "displayName": "Firmware version read only",
                            "name": "fwversion",
                            "schema": "string"
                        },
                        {
                            "@id": "<id>",
                            "@type": ["Property"],
                            "displayName": "Display status",
                            "name": "status",
                            "writable": true,
                            "schema": {
                                "@id": "urn:testInterface:status:obj:ka8iw8wka:1",
                                "@type": ["Object"]
                            }
                        },
                        {
                            "@id": "<id>",
                            "@type": ["Command"],
                            "commandType": "synchronous",
                            "request": {
                                "@id": "<id>",
                                "@type": ["SchemaField"],
                                "displayName": "Configuration",
                                "name": "config",
                                "schema": "string"
                            },
                            "response": {
                                "@id": "<id>",
                                "@type": ["SchemaField"],
                                "displayName": "Response",
                                "name": "response",
                                "schema": "string"
                            },
                            "displayName": "Configure sensor",
                            "name": "sensorconfig"
                        }
                    ]
                }
```

```
            }
        ],
        "displayName": "Sensor capability model"
    },
    "solutionModel": {
        "@id": "<id>",
        "@type": ["SolutionModel"],
        "cloudProperties": [
            {
                "@id": "<id>",
                "@type": ["CloudProperty"],
                "displayName": "Install date",
                "name": "installDate",
                "schema": "dateTime",
                "valueDetail": {
                    "@id": "<id>",
                    "@type": ["ValueDetail/DateTimeValueDetail"]
                }
            }
        ]
    }
}
```

Data format change notice

> **NOTE**
>
> The telemetry stream data format is unaffected by this change. Only the devices and device templates streams of data are affected.

If you have an existing data export in your preview application with the *Devices* and *Device templates* streams turned on, update your export by **30 June 2020**. This requirement applies to exports to Azure Blob storage, Azure Event Hubs, and Azure Service Bus.

Starting 3 February 2020, all new exports in applications with Devices and Device templates enabled will have the data format described above. All exports created before this date remain on the old data format until 30 June 2020, at which time these exports will automatically be migrated to the new data format. The new data format matches the device, device property, device cloud property, and device template objects in the IoT Central public API.

For **Devices**, notable differences between the old data format and the new data format include:

- `@id` for device is removed, `deviceId` is renamed to `id`
- `provisioned` flag is added to describe the provisioning status of the device
- `approved` flag is added to describe the approval state of the device
- `properties` including device and cloud properties, matches entities in the public API

For **Device templates**, notable differences between the old data format and the new data format include:

- `@id` for device template is renamed to `id`
- `@type` for the device template is renamed to `types`, and is now an array

Devices (format deprecated as of 3 February 2020)

```
{
  "@id":"<id-value>",
  "@type":"Device",
  "displayName":"Airbox",
  "data":{
    "$cloudProperties":{
        "Color":"blue"
    },
    "EnvironmentalSensor":{
      "thsensormodel":{
        "reported":{
          "value":"Neque quia et voluptatem veritatis assumenda consequuntur quod.",
          "$lastUpdatedTimestamp":"2019-09-30T20:35:43.8478978Z"
        }
      },
      "pm25sensormodel":{
        "reported":{
          "value":"Aut alias odio.",
          "$lastUpdatedTimestamp":"2019-09-30T20:35:43.8478978Z"
        }
      }
    },
    "urn_azureiot_DeviceManagement_DeviceInformation":{
      "totalStorage":{
        "reported":{
          "value":27900.9730905171,
          "$lastUpdatedTimestamp":"2019-09-30T20:35:43.8478978Z"
        }
      },
      "totalMemory":{
        "reported":{
          "value":4667.82916715811,
          "$lastUpdatedTimestamp":"2019-09-30T20:35:43.8478978Z"
        }
      }
    }
  },
  "instanceOf":"<template-id>",
  "deviceId":"<device-id>",
  "simulated":true
}
```


Device templates (format deprecated as of 3 February 2020)


```
{
  "@id":"<template-id>",
  "@type":"DeviceModelDefinition",
  "displayName":"Airbox",
  "capabilityModel":{
    "@id":"<id>",
    "@type":"CapabilityModel",
    "implements":[
      {
        "@id":"<id>",
        "@type":"InterfaceInstance",
        "name":"EnvironmentalSensor",
        "schema":{
          "@id":"<id>",
          "@type":"Interface",
          "comment":"Requires temperature and humidity sensors.",
          "description":"Provides functionality to report temperature, humidity. Provides telemetry, commands
and read-write properties",
          "displayName":"Environmental Sensor",
          "contents":[
            {
              "@id":"<id>",
```

```
            "@type":"Telemetry",
            "description":"Current temperature on the device",
            "displayName":"Temperature",
            "name":"temp",
            "schema":"double",
            "unit":"Units/Temperature/celsius",
            "valueDetail":{
              "@id":"<id>",
              "@type":"ValueDetail/NumberValueDetail",
              "minValue":{
                "@value":"50"
              }
            },
            "visualizationDetail":{
              "@id":"<id>",
              "@type":"VisualizationDetail"
            }
          },
          {
            "@id":"<id>",
            "@type":"Telemetry",
            "description":"Current humidity on the device",
            "displayName":"Humidity",
            "name":"humid",
            "schema":"integer"
          },
          {
            "@id":"<id>",
            "@type":"Telemetry",
            "description":"Current PM2.5 on the device",
            "displayName":"PM2.5",
            "name":"pm25",
            "schema":"integer"
          },
          {
            "@id":"<id>",
            "@type":"Property",
            "description":"T&H Sensor Model Name",
            "displayName":"T&H Sensor Model",
            "name":"thsensormodel",
            "schema":"string"
          },
          {
            "@id":"<id>",
            "@type":"Property",
            "description":"PM2.5 Sensor Model Name",
            "displayName":"PM2.5 Sensor Model",
            "name":"pm25sensormodel",
            "schema":"string"
          }
        ]
      }
    },
    {
      "@id":"<id>",
      "@type":"InterfaceInstance",
      "name":"urn_azureiot_DeviceManagement_DeviceInformation",
      "schema":{
        "@id":"<id>",
        "@type":"Interface",
        "displayName":"Device information",
        "contents":[
          {
            "@id":"<id>",
            "@type":"Property",
            "comment":"Total available storage on the device in kilobytes. Ex. 20480000 kilobytes.",
            "displayName":"Total storage",
            "name":"totalStorage",
            "displayUnit":"kilobytes",
```

```
              "schema":"long"
            },
            {
              "@id":"<id>",
              "@type":"Property",
              "comment":"Total available memory on the device in kilobytes. Ex. 256000 kilobytes.",
              "displayName":"Total memory",
              "name":"totalMemory",
              "displayUnit":"kilobytes",
              "schema":"long"
            }
          ]
        }
      }
    ],
    "displayName":"AAEONAirbox52"
  },
  "solutionModel":{
    "@id":"<id>",
    "@type":"SolutionModel",
    "cloudProperties":[
      {
        "@id":"<id>",
        "@type":"CloudProperty",
        "displayName":"Color",
        "name":"Color",
        "schema":"string",
        "valueDetail":{
          "@id":"<id>",
          "@type":"ValueDetail/StringValueDetail"
        },
        "visualizationDetail":{
          "@id":"<id>",
          "@type":"VisualizationDetail"
        }
      }
    ]
  }
}
```

Next steps

Now that you know how to export your data to Azure Event Hubs, Azure Service Bus, and Azure Blob storage, continue to the next step:

How to create webhooks

Extend Azure IoT Central with custom rules using Stream Analytics, Azure Functions, and SendGrid

3/24/2020 • 8 minutes to read • Edit Online

This how-to guide shows you, as a solution developer, how to extend your IoT Central application with custom rules and notifications. The example shows sending a notification to an operator when a device stops sending telemetry. The solution uses an Azure Stream Analytics query to detect when a device has stopped sending telemetry. The Stream Analytics job uses Azure Functions to send notification emails using SendGrid.

This how-to guide shows you how to extend IoT Central beyond what it can already do with the built-in rules and actions.

In this how-to guide, you learn how to:

- Stream telemetry from an IoT Central application using *continuous data export*.
- Create a Stream Analytics query that detects when a device has stopped sending data.
- Send an email notification using the Azure Functions and SendGrid services.

Prerequisites

To complete the steps in this how-to guide, you need an active Azure subscription.

If you don't have an Azure subscription, create a free account before you begin.

IoT Central application

Create an IoT Central application on the Azure IoT Central application manager website with the following settings:

SETTING	VALUE
Pricing plan	Standard
Application template	In-store analytics – condition monitoring
Application name	Accept the default or choose your own name
URL	Accept the default or choose your own unique URL prefix
Directory	Your Azure Active Directory tenant
Azure subscription	Your Azure subscription
Region	Your nearest region

The examples and screenshots in this article use the **United States** region. Choose a location close to you and make sure you create all your resources in the same region.

This application template includes two simulated thermostat devices that send telemetry.

Resource group

Use the Azure portal to create a resource group called **DetectStoppedDevices** to contain the other resources you create. Create your Azure resources in the same location as your IoT Central application.

Event Hubs namespace

Use the Azure portal to create an Event Hubs namespace with the following settings:

SETTING	VALUE
Name	Choose your namespace name
Pricing tier	Basic
Subscription	Your subscription
Resource group	DetectStoppedDevices
Location	East US
Throughput Units	1

Stream Analytics job

Use the Azure portal to create a Stream Analytics job with the following settings:

SETTING	VALUE
Name	Choose your job name
Subscription	Your subscription
Resource group	DetectStoppedDevices
Location	East US
Hosting environment	Cloud
Streaming units	3

Function app

Use the Azure portal to create a function app with the following settings:

SETTING	VALUE
App name	Choose your function app name
Subscription	Your subscription
Resource group	DetectStoppedDevices
OS	Windows
Hosting Plan	Consumption Plan
Location	East US
Runtime Stack	.NET

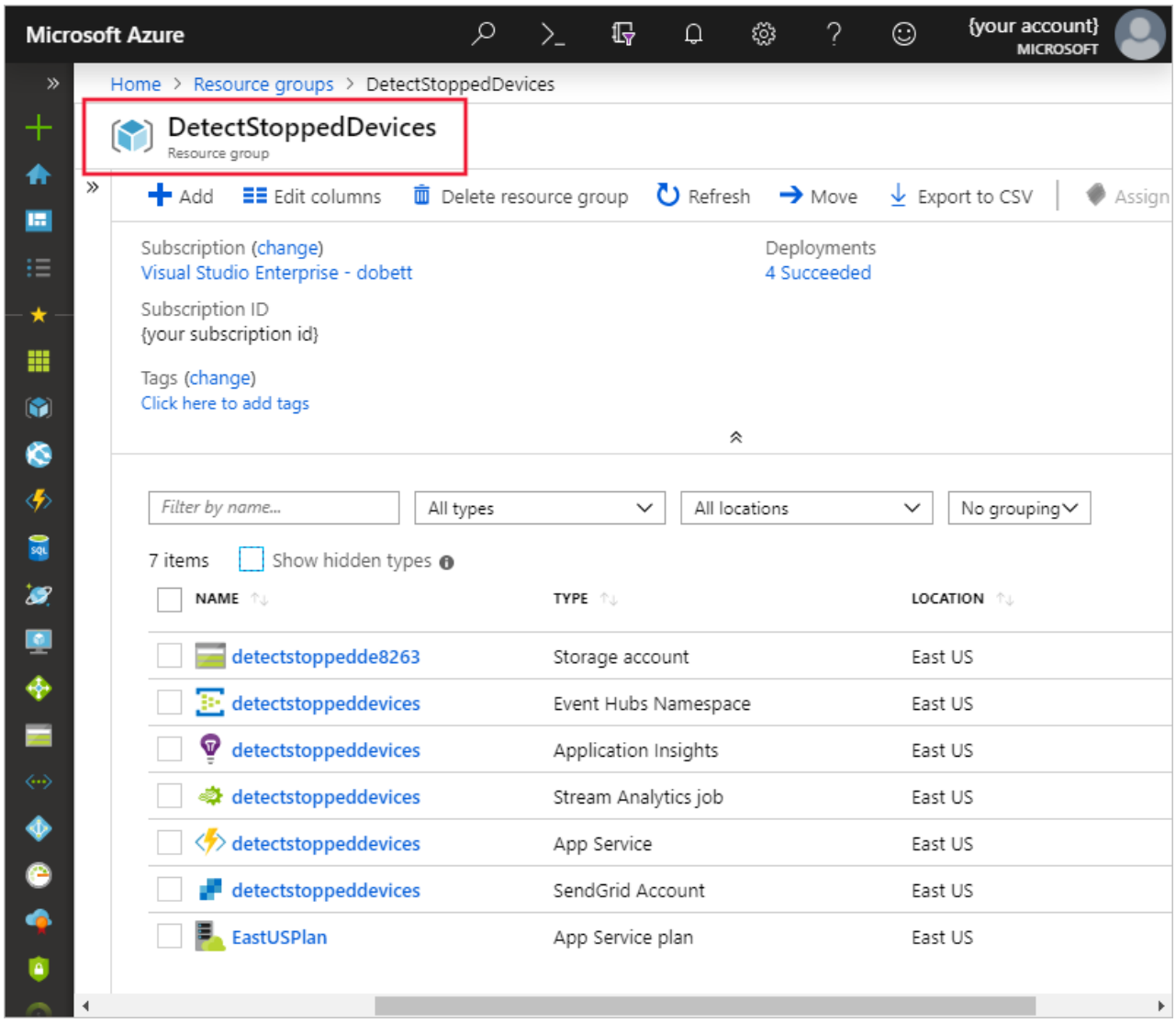
SETTING	VALUE
Storage	Create new

SendGrid account

Use the Azure portal to create a SendGrid account with the following settings:

SETTING	VALUE
Name	Choose your SendGrid account name
Password	Create a password
Subscription	Your subscription
Resource group	DetectStoppedDevices
Pricing tier	F1 Free
Contact information	Fill out required information

When you've created all the required resources, your **DetectStoppedDevices** resource group looks like the following screenshot:

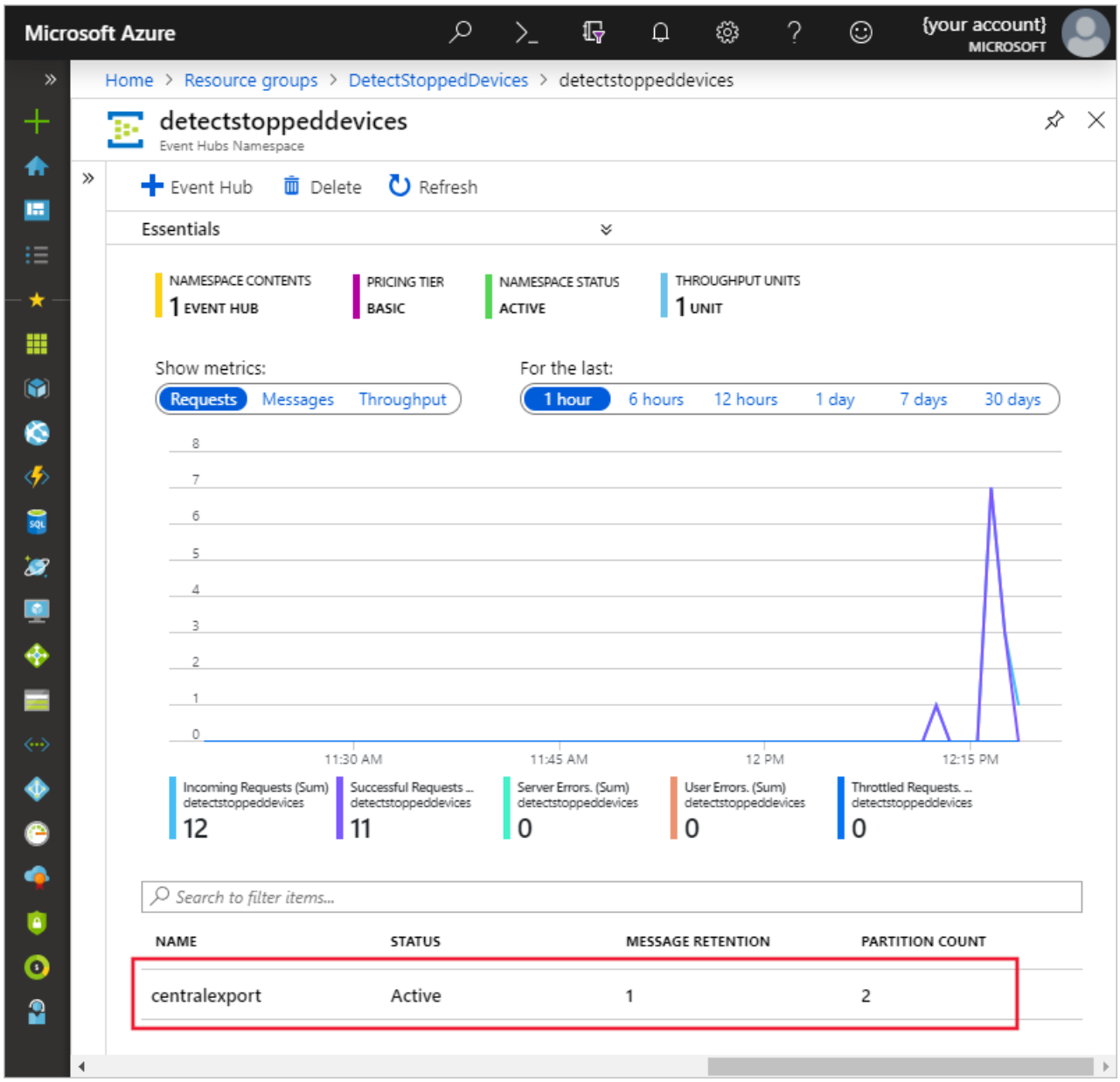


Create an event hub

You can configure an IoT Central application to continuously export telemetry to an event hub. In this section, you create an event hub to receive telemetry from your IoT Central application. The event hub delivers the telemetry to your Stream Analytics job for processing.

1. In the Azure portal, navigate to your Event Hubs namespace and select **+ Event Hub**.
2. Name your event hub **centralexport**, and select **Create**.

Your Event Hubs namespace looks like the following screenshot:

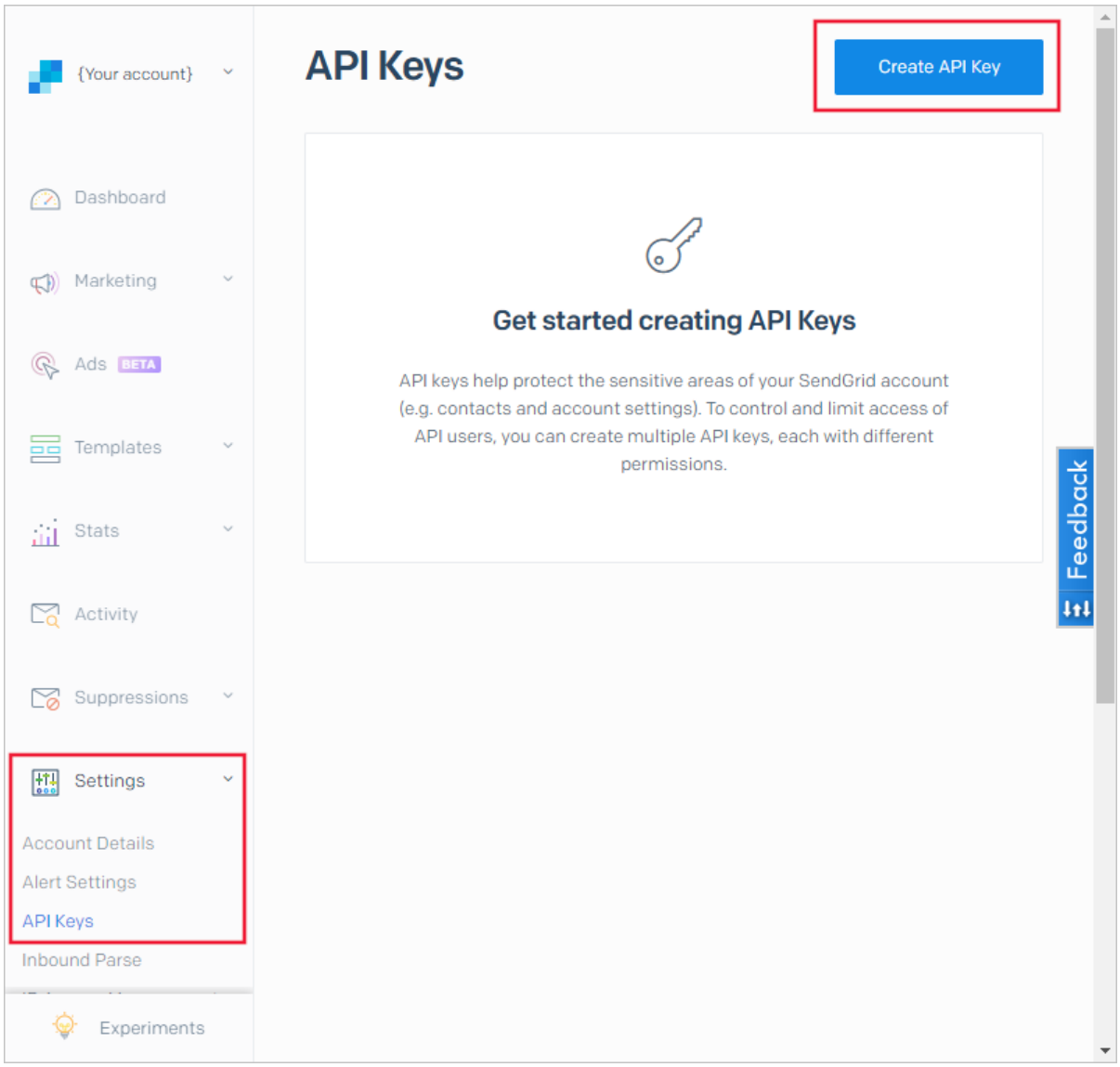


Get SendGrid API key

Your function app needs a SendGrid API key to send email messages. To create a SendGrid API key:

1. In the Azure portal, navigate to your SendGrid account. Then choose **Manage** to access your SendGrid account.

2. In your SendGrid account, choose **Settings**, then **API Keys**. Choose **Create API Key**:



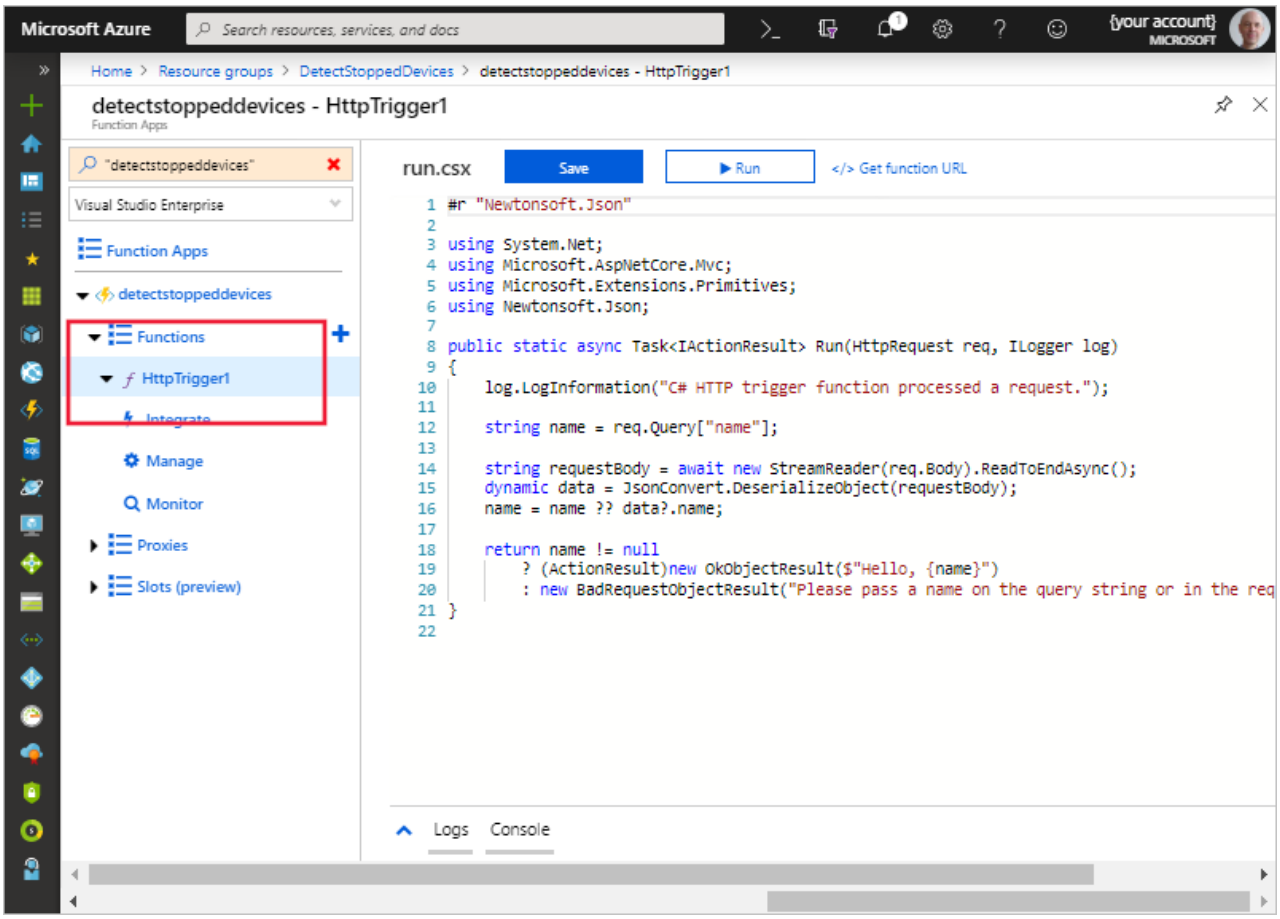
3. On the **Create API Key** page, create a key named **AzureFunctionAccess** with **Full Access** permissions.
4. Make a note of the API Key, you need it when you configure your function app.

Define the function

This solution uses an Azure Functions app to send an email notification when the Stream Analytics job detects a stopped device. To create your function app:

1. In the Azure portal, navigate to the **App Service** instance in the **DetectStoppedDevices** resource group.
2. Select **+** to create a new function.
3. On the **CHOOSE A DEVELOPMENT ENVIRONMENT** page, choose **In-portal** and then select **Continue**.
4. On the **CREATE A FUNCTION** page, choose **Webhook + API** and then select **Create**.

The portal creates a default function called **HttpTrigger1**:

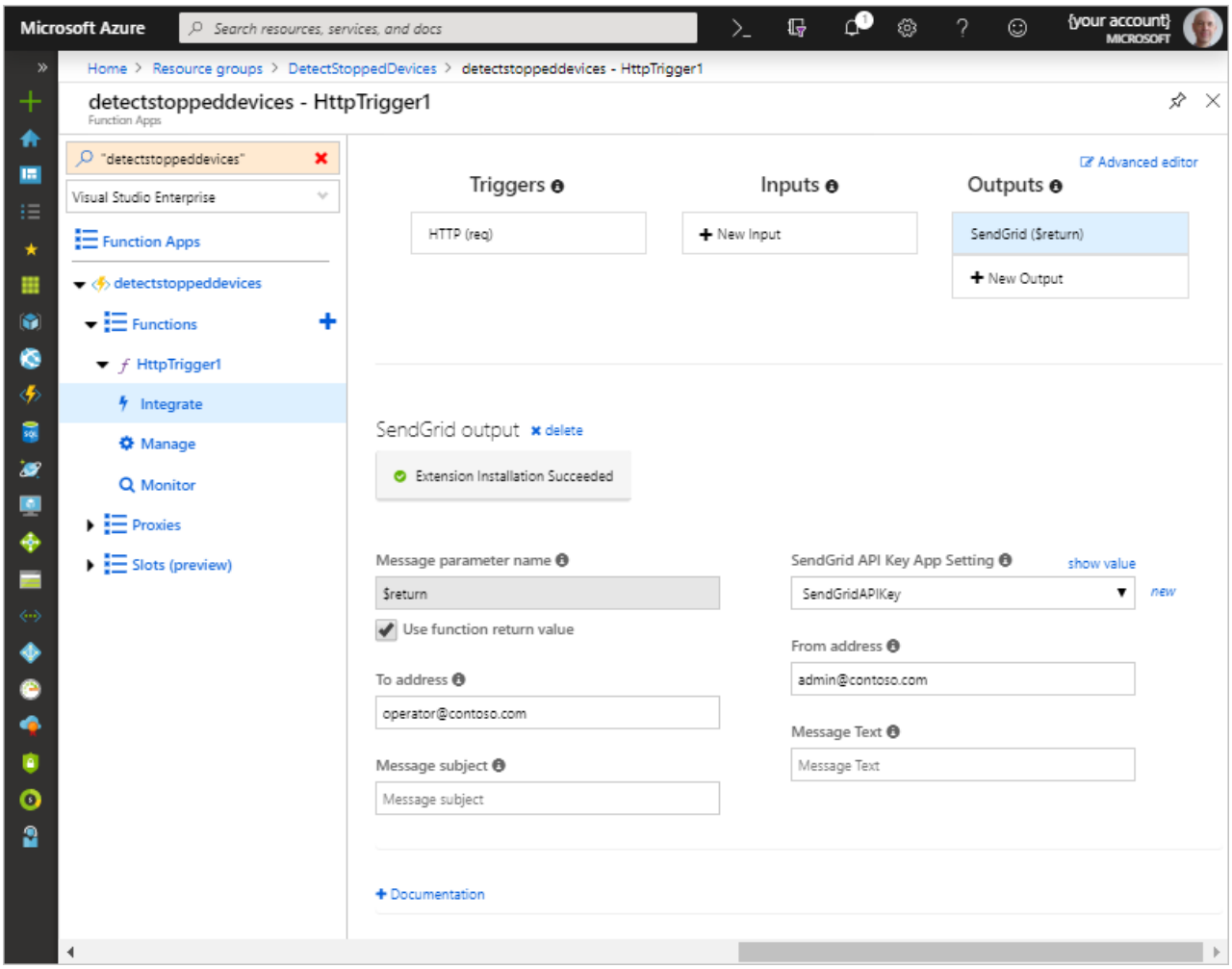


Configure function bindings

To send emails with SendGrid, you need to configure the bindings for your function as follows:

1. Select **Integrate**, choose the output **HTTP ($return)**, and then select **delete**.
2. Choose **+ New Output**, then choose **SendGrid**, and then choose **Select**. Choose **Install** to install the SendGrid extension.
3. When the installation completes, select **Use function return value**. Add a valid **To address** to receive email notifications. Add a valid **From address** to use as the email sender.
4. Select **new** next to **SendGrid API Key App Setting**. Enter **SendGridAPIKey** as the key, and the SendGrid API key you noted previously as the value. Then select **Create**.
5. Choose **Save** to save the SendGrid bindings for your function.

The integrate settings look like the following screenshot:



Add the function code

To implement your function, add the C# code to parse the incoming HTTP request and send the emails as follows:

1. Choose the **HttpTrigger1** function in your function app and replace the C# code with the following code:

```
#r "Newtonsoft.Json"
#r "..\bin\SendGrid.dll"

using System;
using SendGrid.Helpers.Mail;
using Microsoft.Azure.WebJobs.Host;
using Microsoft.AspNetCore.Mvc;
using Microsoft.Extensions.Primitives;
using Newtonsoft.Json;

public static SendGridMessage Run(HttpRequest req, ILogger log)
{
    string requestBody = new StreamReader(req.Body).ReadToEnd();
    log.LogInformation(requestBody);
    var notifications = JsonConvert.DeserializeObject<IList<Notification>>(requestBody);

    SendGridMessage message = new SendGridMessage();
    message.Subject = "Contoso device notification";

    var content = "The following device(s) have stopped sending telemetry:<br/><br/><table><tr>
<th>Device ID</th><th>Time</th></tr>";
    foreach(var notification in notifications) {
        log.LogInformation($"No message received - Device: {notification.deviceid}, Time:
{notification.time}");
        content += $"<tr><td>{notification.deviceid}</td><td>{notification.time}</td></tr>";
    }
    content += "</table>";
    message.AddContent("text/html", content);

    return message;
}

public class Notification
{
    public string deviceid { get; set; }
    public string time { get; set; }
}
```

You may see an error message until you save the new code.

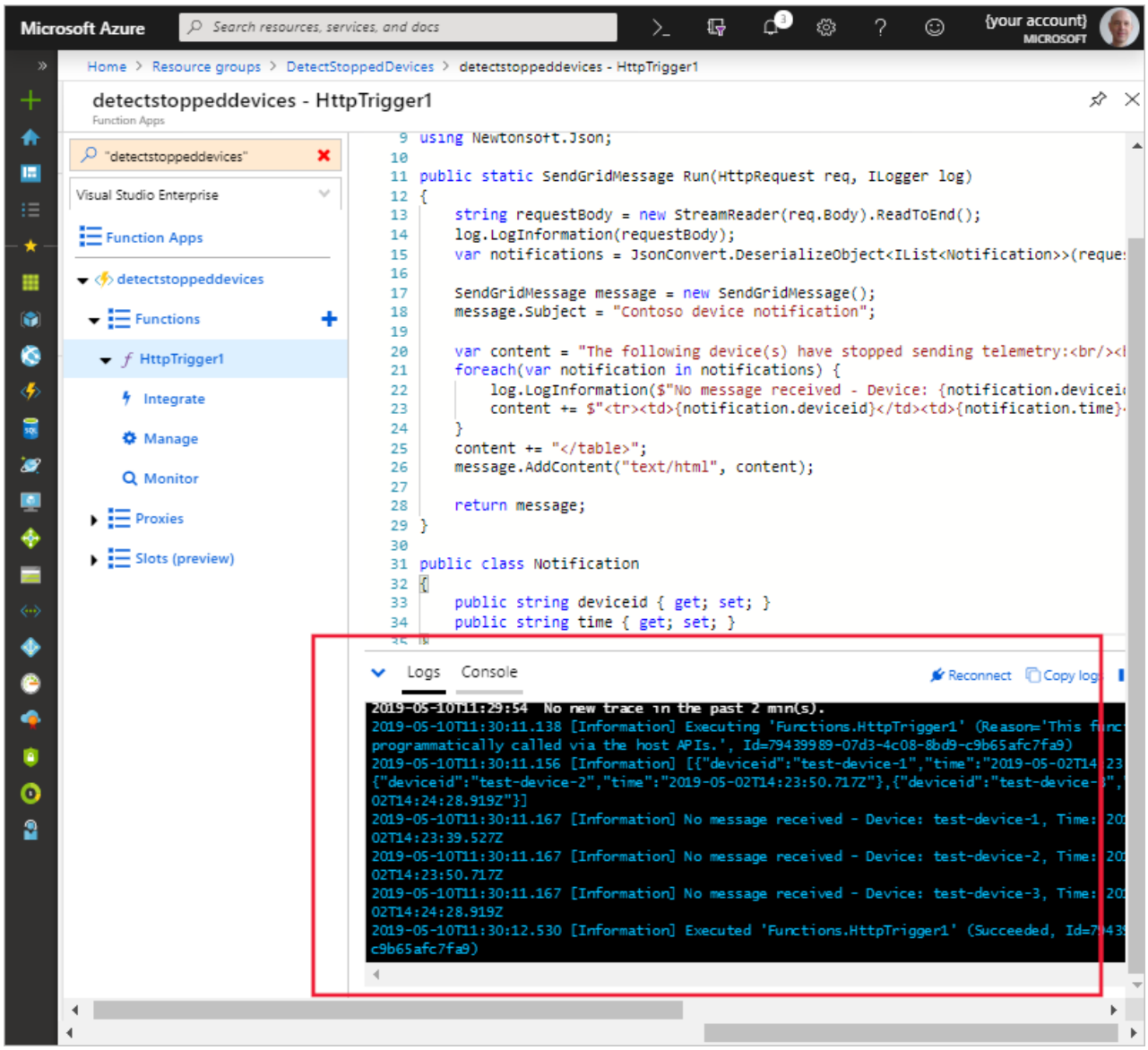
2. Select **Save** to save the function.

Test the function works

To test the function in the portal, first choose **Logs** at the bottom of the code editor. Then choose **Test** to the right of the code editor. Use the following JSON as the **Request body**:

```
[{"deviceid":"test-device-1","time":"2019-05-02T14:23:39.527Z"},{"deviceid":"test-device-2","time":"2019-05-
02T14:23:50.717Z"},{"deviceid":"test-device-3","time":"2019-05-02T14:24:28.919Z"}]
```

The function log messages appear in the **Logs** panel:



After a few minutes, the **To** email address receives an email with the following content:

```
The following device(s) have stopped sending telemetry:

Device ID    Time
test-device-1    2019-05-02T14:23:39.527Z
test-device-2    2019-05-02T14:23:50.717Z
test-device-3    2019-05-02T14:24:28.919Z
```

Add Stream Analytics query

This solution uses a Stream Analytics query to detect when a device stops sending telemetry for more than 120 seconds. The query uses the telemetry from the event hub as its input. The job sends the query results to the function app. In this section, you configure the Stream Analytics job:

1. In the Azure portal, navigate to your Stream Analytics job, under **Jobs topology** select **Inputs**, choose **+ Add stream input**, and then choose **Event Hub**.

2. Use the information in the following table to configure the input using the event hub you created previously, then choose **Save**:

SETTING	VALUE
Input alias	centraltelemetry
Subscription	Your subscription
Event Hub namespace	Your Event Hub namespace
Event Hub name	Use existing - **centralexport**

3. Under **Jobs topology**, select **Outputs**, choose **+ Add**, and then choose **Azure function**.
4. Use the information in the following table to configure the output, then choose **Save**:

SETTING	VALUE
Output alias	emailnotification
Subscription	Your subscription
Function app	Your function app
Function	HttpTrigger1

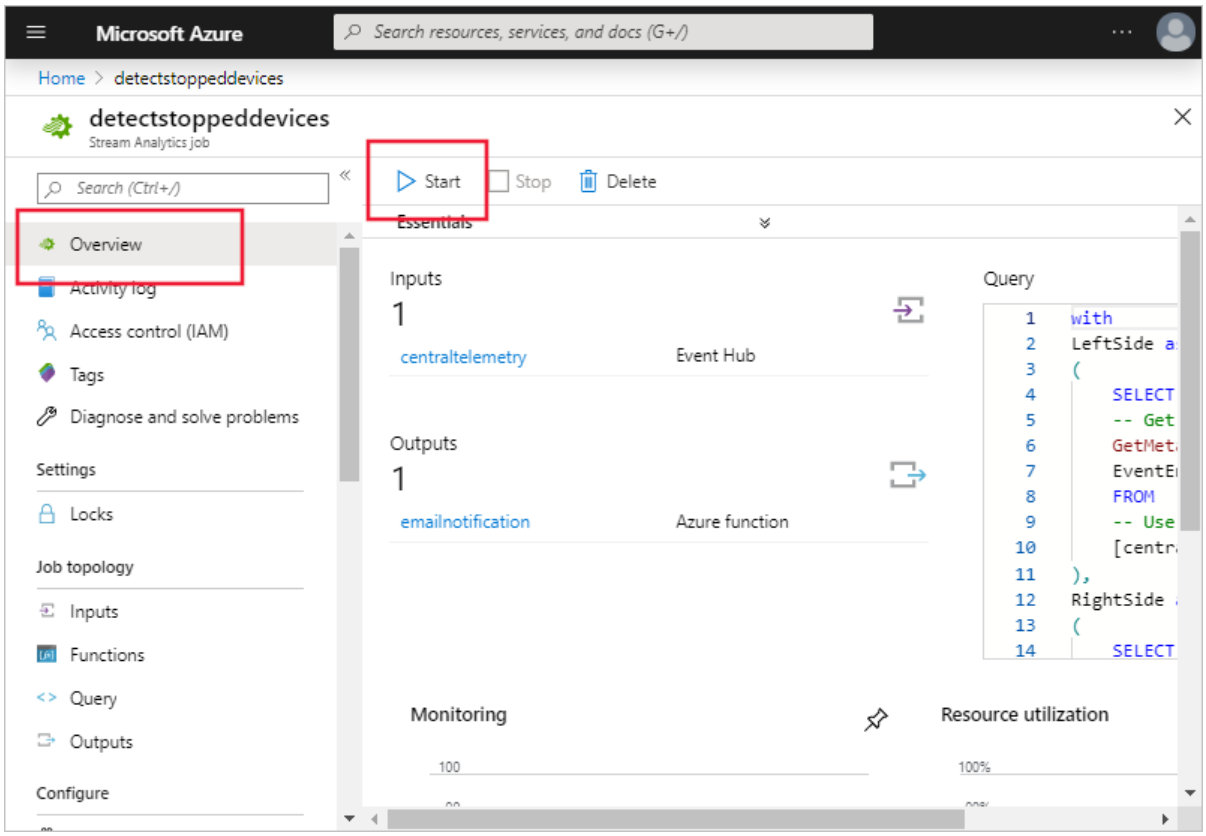
5. Under **Jobs topology**, select **Query** and replace the existing query with the following SQL:

```
with
LeftSide as
(
    SELECT
    -- Get the device ID from the message metadata and create a column
    GetMetadataPropertyValue([centraltelemetry], '[EventHub].[IoTConnectionDeviceId]') as deviceid1,
    EventEnqueuedUtcTime AS time1
    FROM
    -- Use the event enqueued time for time-based operations
    [centraltelemetry] TIMESTAMP BY EventEnqueuedUtcTime
),
RightSide as
(
    SELECT
    -- Get the device ID from the message metadata and create a column
    GetMetadataPropertyValue([centraltelemetry], '[EventHub].[IoTConnectionDeviceId]') as deviceid2,
    EventEnqueuedUtcTime AS time2
    FROM
    -- Use the event enqueued time for time-based operations
    [centraltelemetry] TIMESTAMP BY EventEnqueuedUtcTime
)

SELECT
    LeftSide.deviceid1 as deviceid,
    LeftSide.time1 as time
INTO
    [emailnotification]
FROM
    LeftSide
    LEFT OUTER JOIN
    RightSide
    ON
    LeftSide.deviceid1=RightSide.deviceid2 AND DATEDIFF(second,LeftSide,RightSide) BETWEEN 1 AND 120
    where
    -- Find records where a device didn't send a message 120 seconds
    RightSide.deviceid2 is NULL
```

6. Select **Save**.

7. To start the Stream Analytics job, choose **Overview**, then **Start**, then **Now**, and then **Start**:

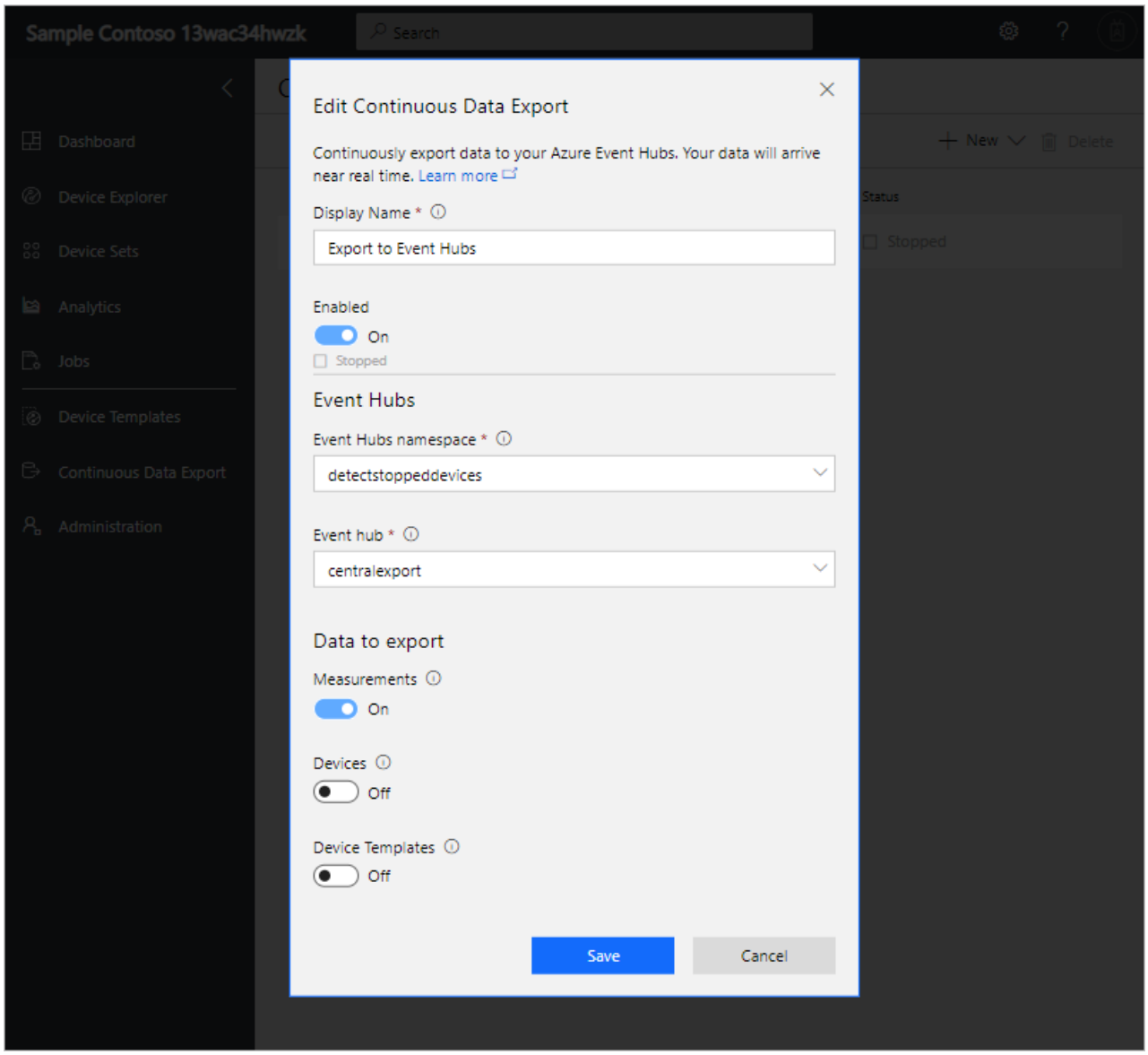


Configure export in IoT Central

On the Azure IoT Central application manager website, navigate to the IoT Central application you created from the Contoso template. In this section, you configure the application to stream the telemetry from its simulated devices to your event hub. To configure the export:

1. Navigate to the **Data Export** page, select **+ New**, and then **Azure Event Hubs**.
2. Use the following settings to configure the export, then select **Save**:

SETTING	VALUE
Display Name	Export to Event Hubs
Enabled	On
Event Hubs namespace	Your Event Hubs namespace name
Event hub	centralexport
Measurements	On
Devices	Off
Device Templates	Off



Wait until the export status is **Running** before you continue.

Test

To test the solution, you can disable the continuous data export from IoT Central to simulated stopped devices:

1. In your IoT Central application, navigate to the **Data Export** page and select the **Export to Event Hubs** export configuration.
2. Set **Enabled** to **Off** and choose **Save**.
3. After at least two minutes, the **To** email address receives one or more emails that look like the following example content:

```
The following device(s) have stopped sending telemetry:

Device ID         Time
Thermostat-Zone1  2019-11-01T12:45:14.686Z
```

Tidy up

To tidy up after this how-to and avoid unnecessary costs, delete the **DetectStoppedDevices** resource group in the Azure portal.

You can delete the IoT Central application from the **Management** page within the application.

Next steps

In this how-to guide, you learned how to:

- Stream telemetry from an IoT Central application using *continuous data export*.
- Create a Stream Analytics query that detects when a device has stopped sending data.
- Send an email notification using the Azure Functions and SendGrid services.

Now that you know how to create custom rules and notifications, the suggested next step is to learn how to Extend Azure IoT Central with custom analytics.

Extend Azure IoT Central with custom analytics using Azure Databricks

3/24/2020 • 5 minutes to read • Edit Online

This how-to guide shows you, as a solution developer, how to extend your IoT Central application with custom analytics and visualizations. The example uses an Azure Databricks workspace to analyze the IoT Central telemetry stream and to generate visualizations such as box plots.

This how-to guide shows you how to extend IoT Central beyond what it can already do with the built-in analytics tools.

In this how-to guide, you learn how to:

- Stream telemetry from an IoT Central application using *continuous data export*.
- Create an Azure Databricks environment to analyze and plot device telemetry.

Prerequisites

To complete the steps in this how-to guide, you need an active Azure subscription.

If you don't have an Azure subscription, create a free account before you begin.

IoT Central application

Create an IoT Central application on the Azure IoT Central application manager website with the following settings:

SETTING	VALUE
Pricing plan	Standard
Application template	In-store analytics – condition monitoring
Application name	Accept the default or choose your own name
URL	Accept the default or choose your own unique URL prefix
Directory	Your Azure Active Directory tenant
Azure subscription	Your Azure subscription
Region	Your nearest region

The examples and screenshots in this article use the **United States** region. Choose a location close to you and make sure you create all your resources in the same region.

This application template includes two simulated thermostat devices that send telemetry.

Resource group

Use the Azure portal to create a resource group called **IoTCentralAnalysis** to contain the other resources you create. Create your Azure resources in the same location as your IoT Central application.

Event Hubs namespace

Use the Azure portal to create an Event Hubs namespace with the following settings:

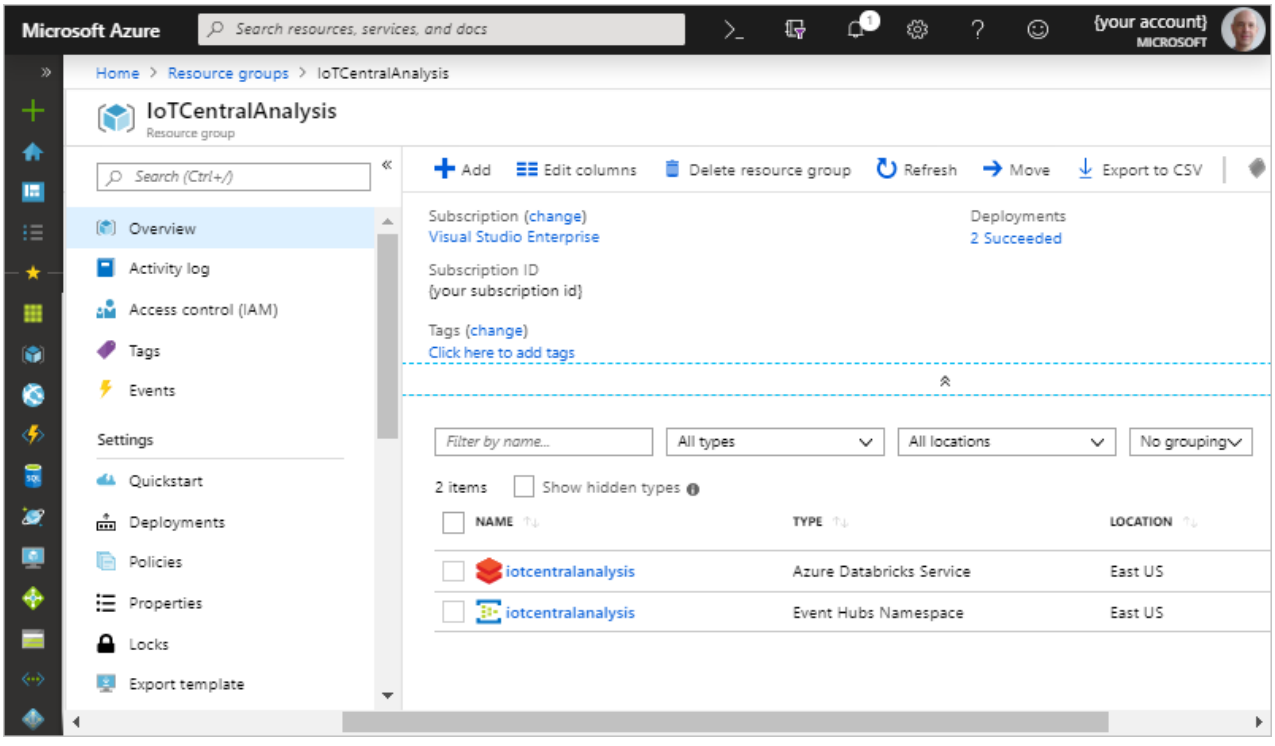
SETTING	VALUE
Name	Choose your namespace name
Pricing tier	Basic
Subscription	Your subscription
Resource group	IoTCentralAnalysis
Location	East US
Throughput Units	1

Azure Databricks workspace

Use the Azure portal to create an Azure Databricks Service with the following settings:

SETTING	VALUE
Workspace name	Choose your workspace name
Subscription	Your subscription
Resource group	IoTCentralAnalysis
Location	East US
Pricing Tier	Standard

When you've created the required resources, your **IoTCentralAnalysis** resource group looks like the following screenshot:

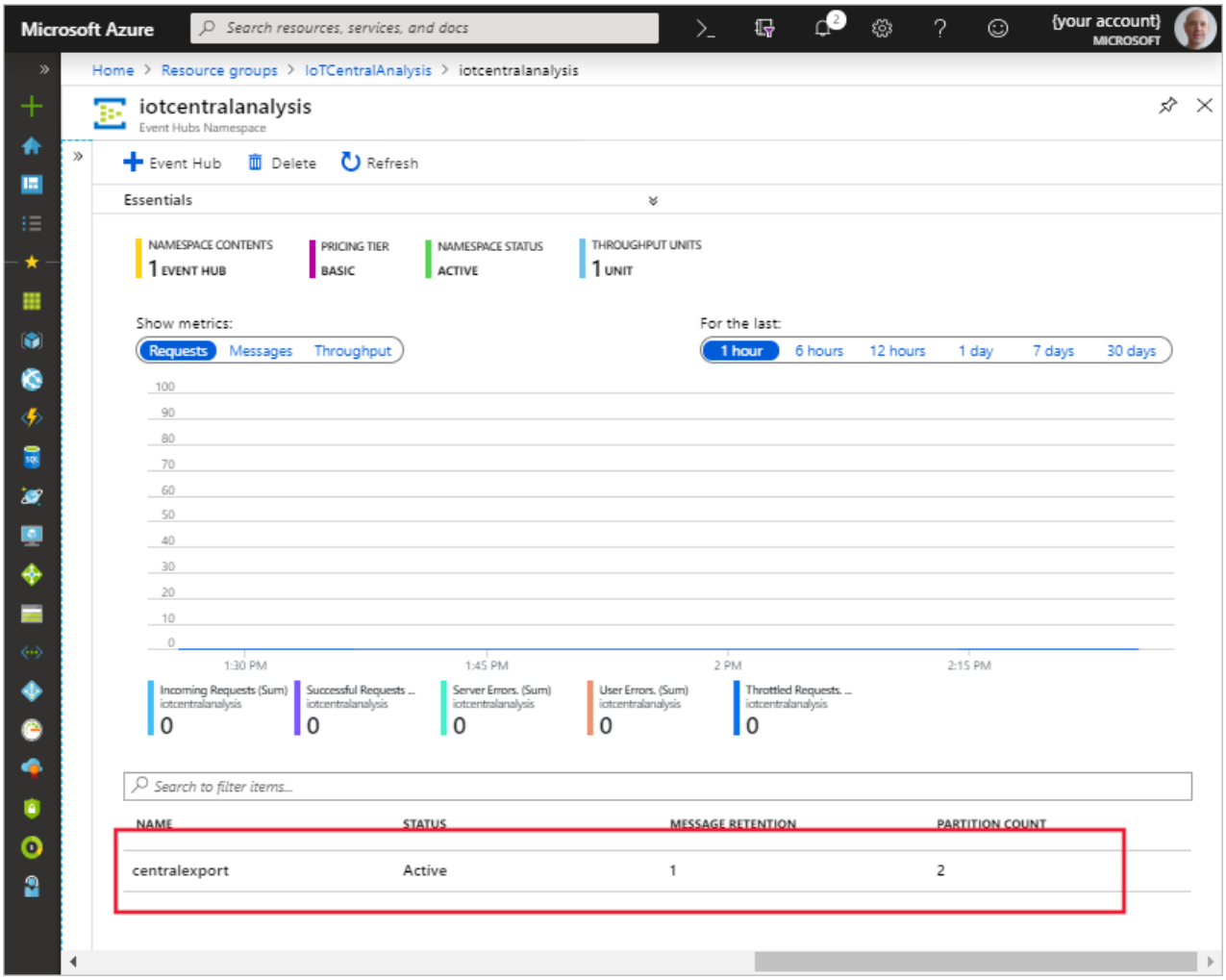


Create an event hub

You can configure an IoT Central application to continuously export telemetry to an event hub. In this section, you create an event hub to receive telemetry from your IoT Central application. The event hub delivers the telemetry to your Stream Analytics job for processing.

1. In the Azure portal, navigate to your Event Hubs namespace and select **+ Event Hub**.
2. Name your event hub **centralexport**, and select **Create**.
3. In the list of event hubs in your namespace, select **centralexport**. Then choose **Shared access policies**.
4. Select **+ Add**. Create a policy named **Listen** with the **Listen** claim.
5. When the policy is ready, select it in the list, and then copy the **Connection string-primary key** value.
6. Make a note of this connection string, you use it later when you configure your Databricks notebook to read from the event hub.

Your Event Hubs namespace looks like the following screenshot:



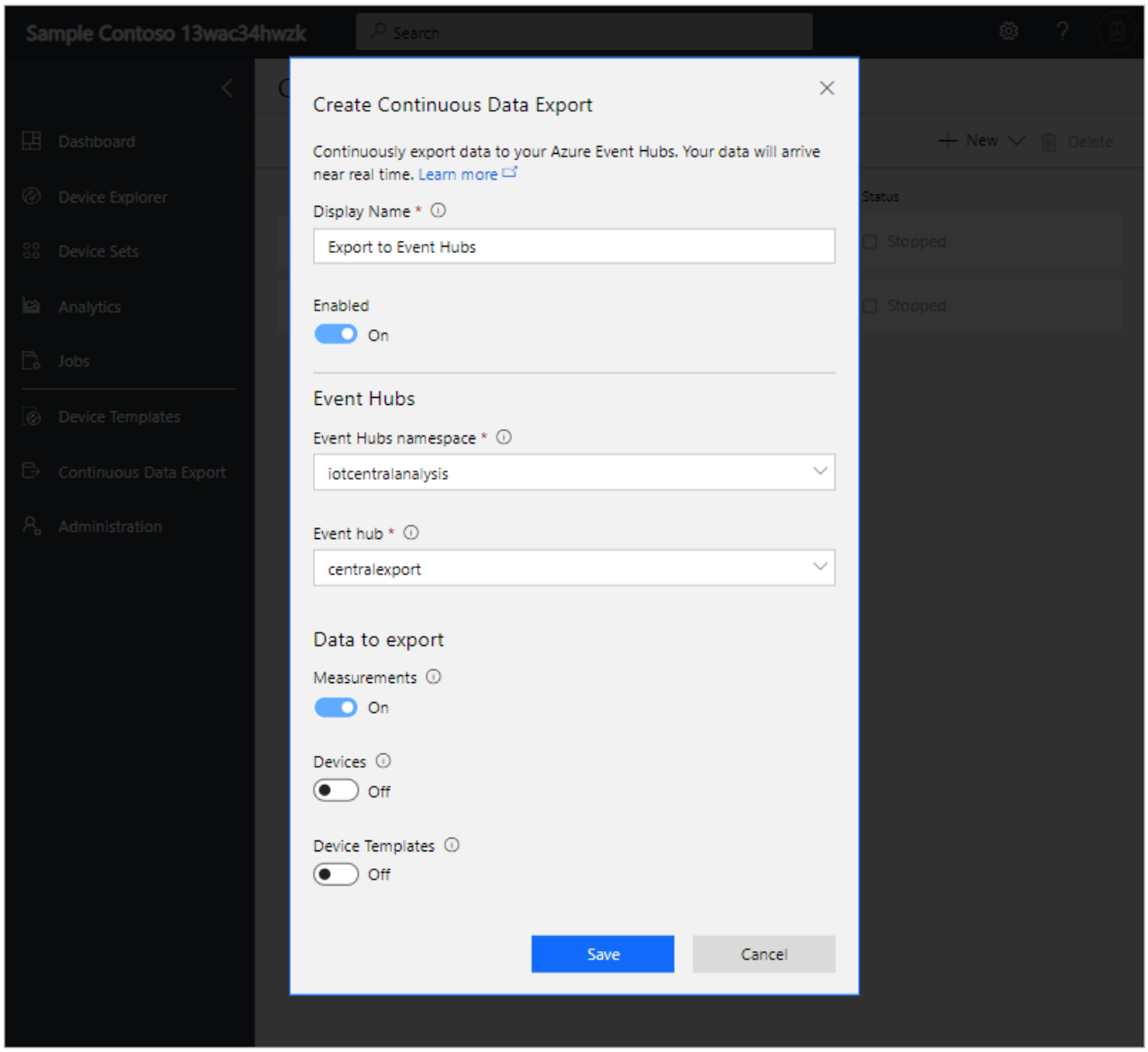
Configure export in IoT Central

On the Azure IoT Central application manager website, navigate to the IoT Central application you created from the Contoso template. In this section, you configure the application to stream the telemetry from its simulated devices to your event hub. To configure the export:

1. Navigate to the **Data Export** page, select **+ New**, and then **Azure Event Hubs**.

2. Use the following settings to configure the export, then select **Save**:

SETTING	VALUE
Display Name	Export to Event Hubs
Enabled	On
Event Hubs namespace	Your Event Hubs namespace name
Event hub	centralexport
Measurements	On
Devices	Off
Device Templates	Off



Wait until the export status is **Running** before you continue.

Configure Databricks workspace

In the Azure portal, navigate to your Azure Databricks service and select **Launch Workspace**. A new tab opens in your browser and signs you in to your workspace.

Create a cluster

On the **Azure Databricks** page, under the list of common tasks, select **New Cluster**.

Use the information in the following table to create your cluster:

SETTING	VALUE
Cluster Name	centralanalysis
Cluster Mode	Standard
Databricks Runtime Version	5.5 LTS (Scala 2.11, Spark 2.4.3)
Python Version	3
Enable Autoscaling	No
Terminate after minutes of inactivity	30
Worker Type	Standard_DS3_v2
Workers	1
Driver Type	Same as worker

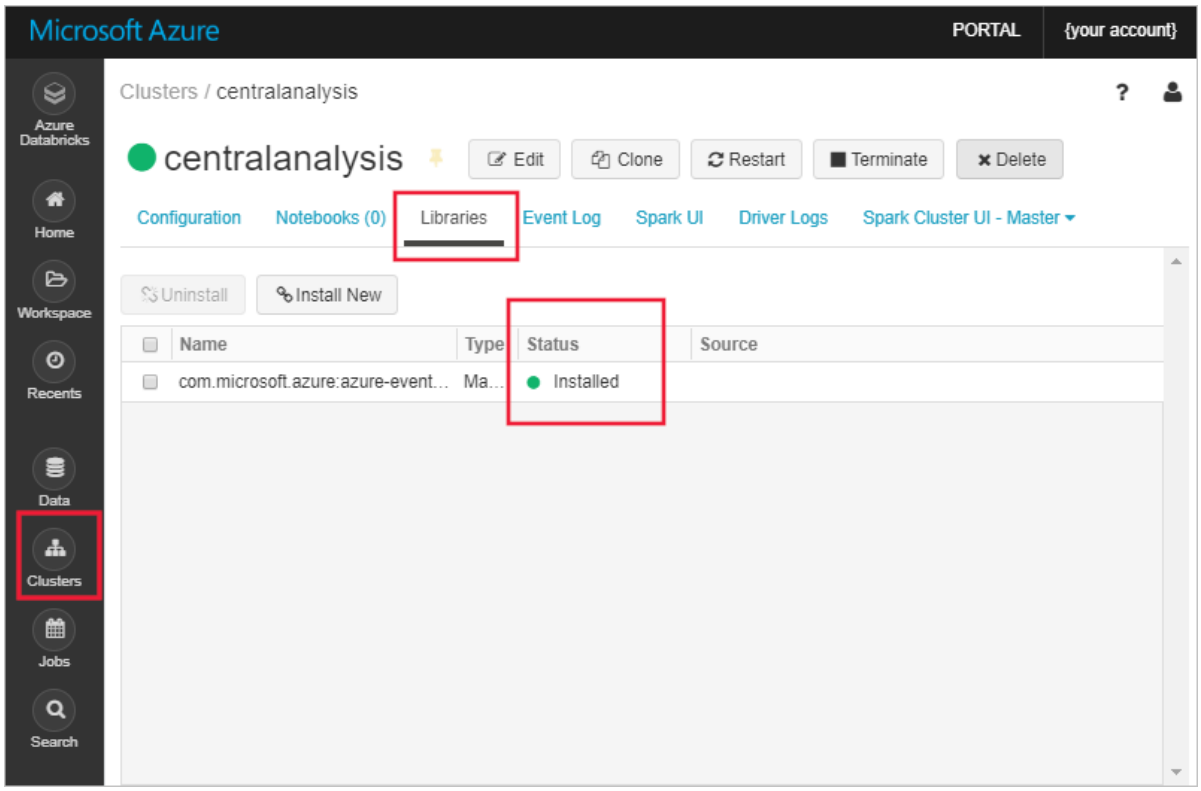
Creating a cluster may take several minutes, wait for the cluster creation to complete before you continue.

Install libraries

On the **Clusters** page, wait until the cluster state is **Running**.

The following steps show you how to import the library your sample needs into the cluster:

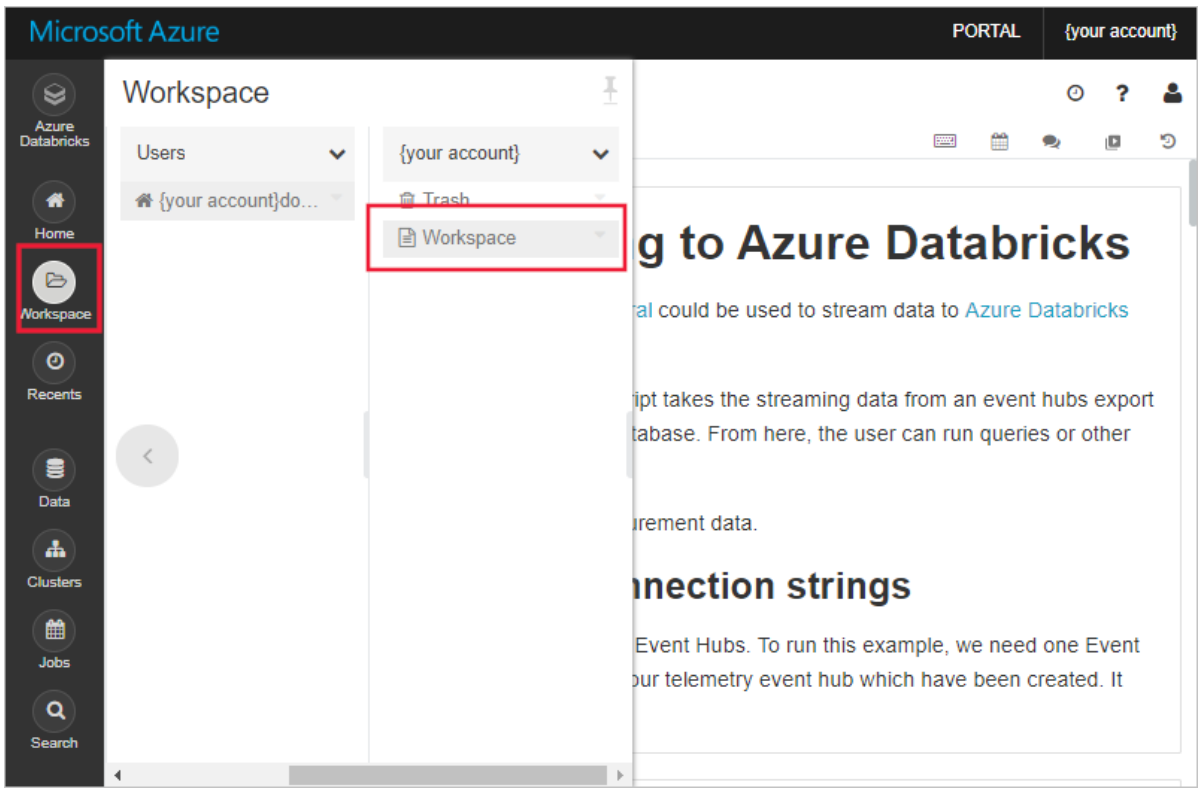
1. On the **Clusters** page, wait until the state of the **centralanalysis** interactive cluster is **Running**.
2. Select the cluster and then choose the **Libraries** tab.
3. On the **Libraries** tab, choose **Install New**.
4. On the **Install Library** page, choose **Maven** as the library source.
5. In the **Coordinates** textbox, enter the following value:
   `com.microsoft.azure:azure-eventhubs-spark_2.11:2.3.10`
6. Choose **Install** to install the library on the cluster.
7. The library status is now **Installed**:



Import a Databricks notebook

Use the following steps to import a Databricks notebook that contains the Python code to analyze and visualize your IoT Central telemetry:

1. Navigate to the **Workspace** page in your Databricks environment. Select the dropdown next to your account name and then choose **Import**.
2. Choose to import from a URL and enter the following address: https://github.com/Azure-Samples/iot-central-docs-samples/blob/master/databricks/IoT%20Central%20Analysis.dbc?raw=true
3. To import the notebook, choose **Import**.
4. Select the **Workspace** to view the imported notebook:



5. Edit the code in the first Python cell to add the Event Hubs connection string you saved previously:

```
from pyspark.sql.functions import *
from pyspark.sql.types import *

###### Event Hub Connection strings ######
telementryEventHubConfig = {
  'eventhubs.connectionString' : '{your Event Hubs connection string}'
}
```

Run analysis

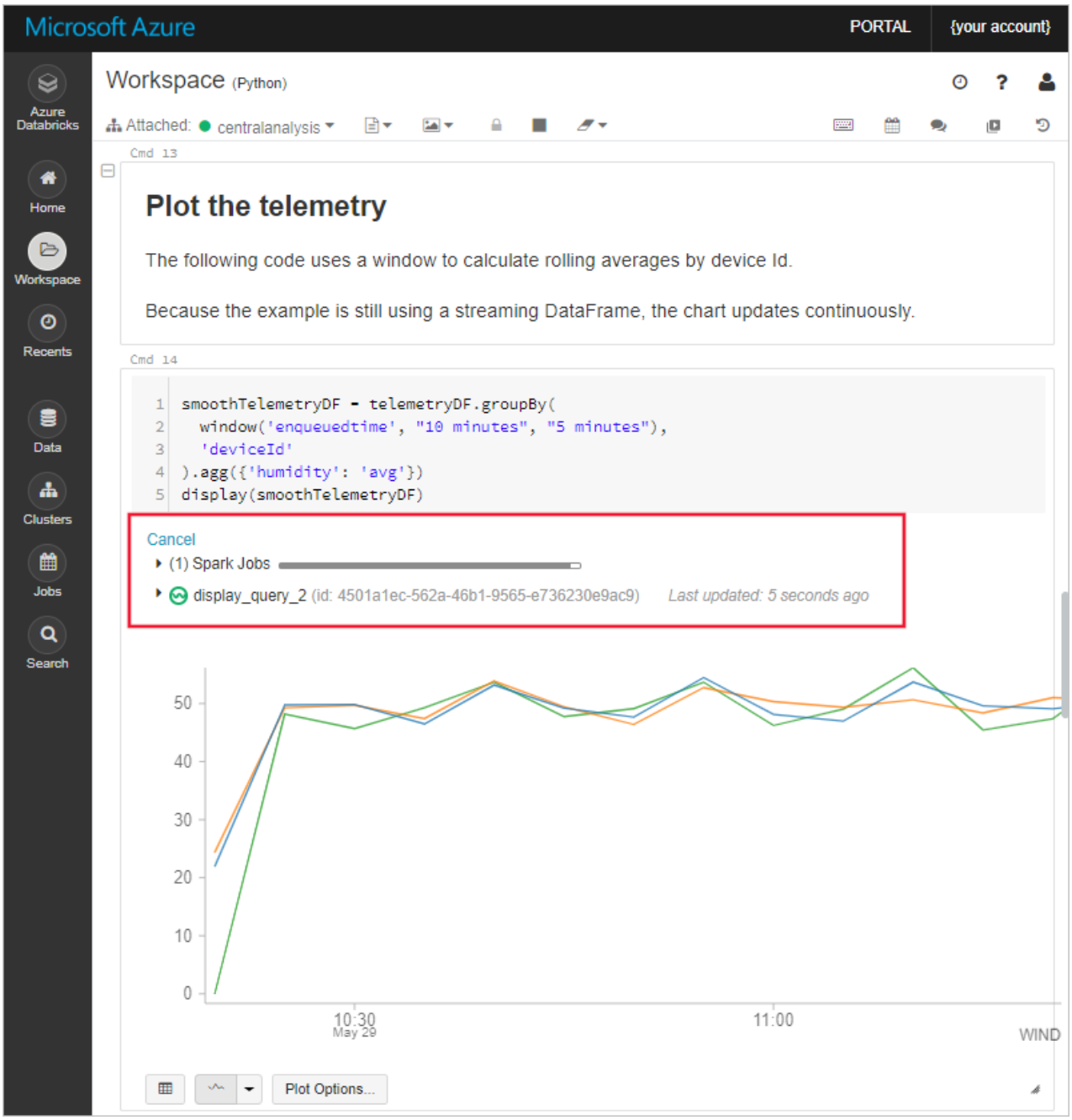
To run the analysis, you must attach the notebook to the cluster:

1. Select **Detached** and then select the **centralanalysis** cluster.
2. If the cluster isn't running, start it.
3. To start the notebook, select the run button.

You may see an error in the last cell. If so, check the previous cells are running, wait a minute for some data to be written to storage, and then run the last cell again.

View smoothed data

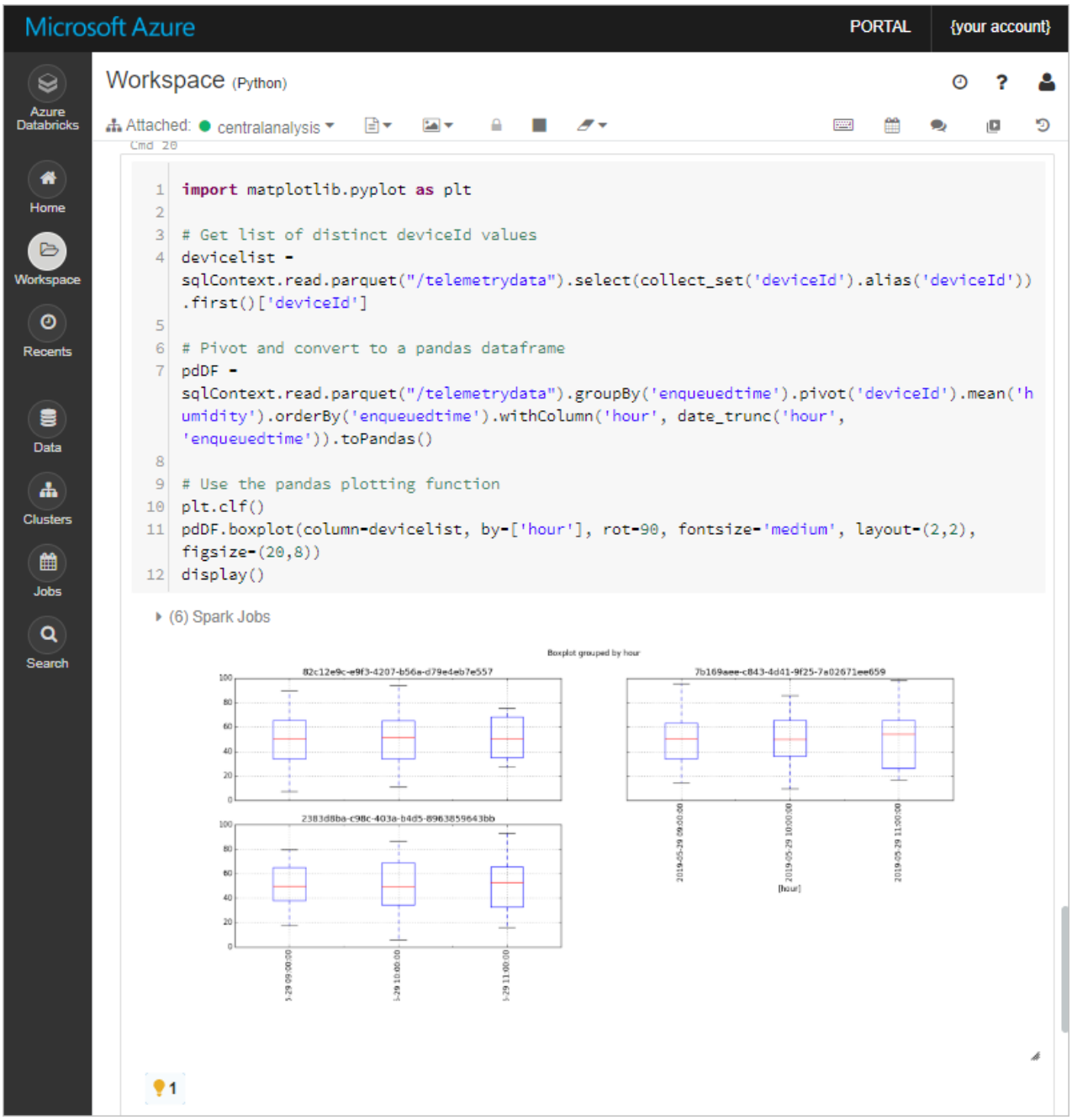
In the notebook, scroll down to cell 14 to see a plot of the rolling average humidity by device type. This plot continuously updates as streaming telemetry arrives:



You can resize the chart in the notebook.

View box plots

In the notebook, scroll down to cell 20 to see the box plots. The box plots are based on static data so to update them you must rerun the cell:



You can resize the plots in the notebook.

Tidy up

To tidy up after this how-to and avoid unnecessary costs, delete the **IoTCentralAnalysis** resource group in the Azure portal.

You can delete the IoT Central application from the **Management** page within the application.

Next steps

In this how-to guide, you learned how to:

- Stream telemetry from an IoT Central application using *continuous data export*.
- Create an Azure Databricks environment to analyze and plot telemetry data.

Now that you know how to create custom analytics, the suggested next step is to learn how to Manage your application.

Change IoT Central application settings

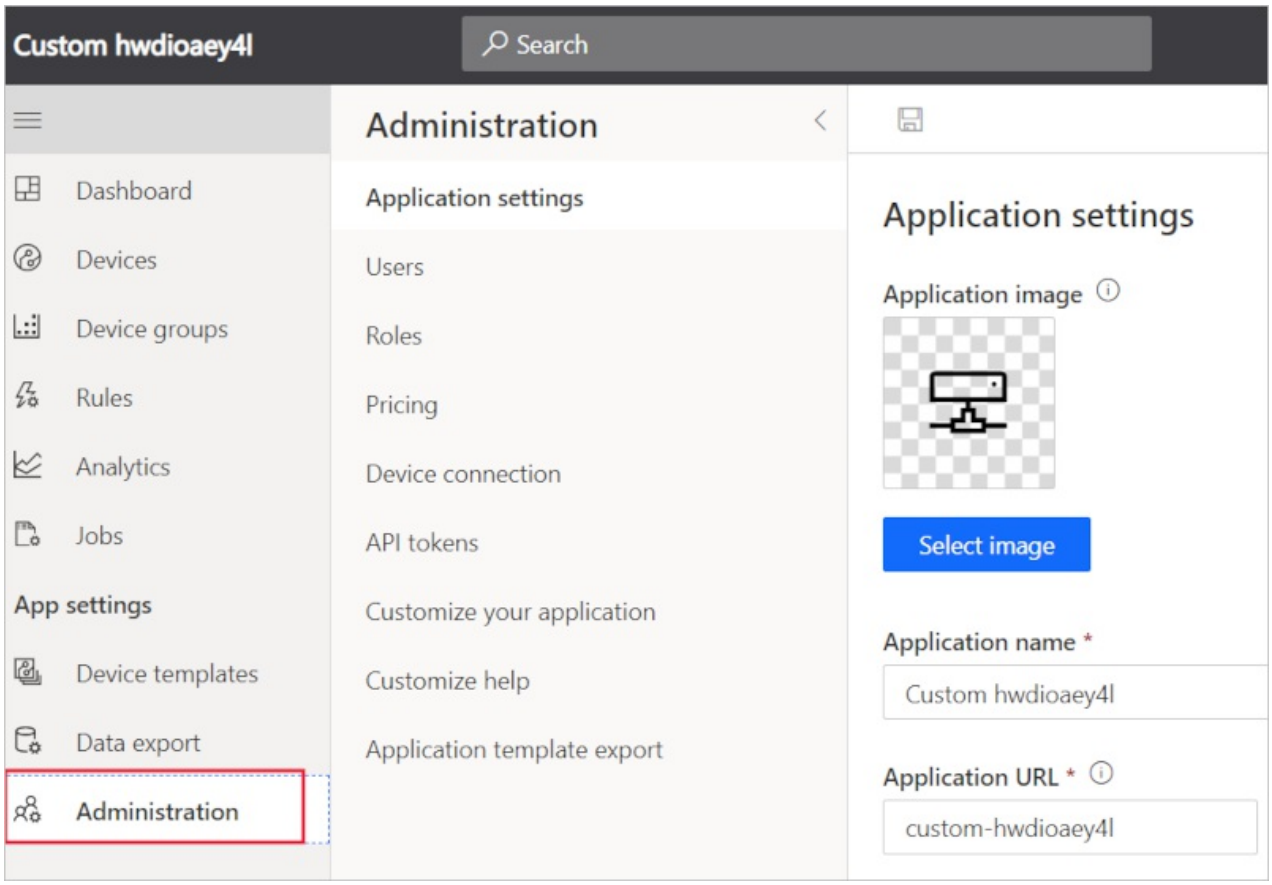
3/24/2020 • 2 minutes to read • Edit Online

This article describes how, as an administrator, you can manage application by changing application name and URL, uploading image, and delete an application in your Azure IoT Central application.

To access and use the **Administration** section, you must be in the **Administrator** role for an Azure IoT Central application. If you create an Azure IoT Central application, you're automatically assigned to the **Administrator** role for that application.

Change application name and URL

In the **Application Settings** page, you can change the name and URL of your application, then select **Save**.



If your administrator creates a custom theme for your application, this page includes an option to hide the **Application Name** in the UI. This option is useful if the application logo in the custom theme includes the application name. For more information, see Customize the Azure IoT Central UI.

> **NOTE**
> If you change your URL, your old URL can be taken by another Azure IoT Central customer. If that happens, it is no longer available for you to use. When you change your URL, the old URL no longer works, and you need to notify your users about the new URL to use.

Delete an application

Use the **Delete** button to permanently delete your IoT Central application. This action permanently deletes all data

that's associated with the application.

> **NOTE**
> To delete an application, you must also have permissions to delete resources in the Azure subscription you chose when you created the application. To learn more, see Use role-based access control to manage access to your Azure subscription resources.

Manage programmatically

IoT Central Azure Resource Manager SDK packages are available for Node, Python, C#, Ruby, Java, and Go. You can use these packages to create, list, update, or delete IoT Central applications. The packages include helpers to manage authentication and error handling.

You can find examples of how to use the Azure Resource Manager SDKs at https://github.com/emgarten/iotcentral-arm-sdk-examples.

To learn more, see the following GitHub repositories and packages:

LANGUAGE	REPOSITORY	PACKAGE
Node	https://github.com/Azure/azure-sdk-for-node	https://www.npmjs.com/package/azure-arm-iotcentral
Python	https://github.com/Azure/azure-sdk-for-python	https://pypi.org/project/azure-mgmt-iotcentral
C#	https://github.com/Azure/azure-sdk-for-net	https://www.nuget.org/packages/Microsoft.Azure.Management.IotCentral
Ruby	https://github.com/Azure/azure-sdk-for-ruby	https://rubygems.org/gems/azure_mgmt_iot_central
Java	https://github.com/Azure/azure-sdk-for-java	https://search.maven.org/search?q=a:azure-mgmt-iotcentral
Go	https://github.com/Azure/azure-sdk-for-go	https://github.com/Azure/azure-sdk-for-go

Next steps

Now that you've learned about how to administer your Azure IoT Central application, the suggested next step is to learn about Manage users and roles in Azure IoT Central.

Export your application

3/24/2020 • 3 minutes to read • Edit Online

This article describes how, as a solution manager, to export an IoT Central application to be able to reuse it.

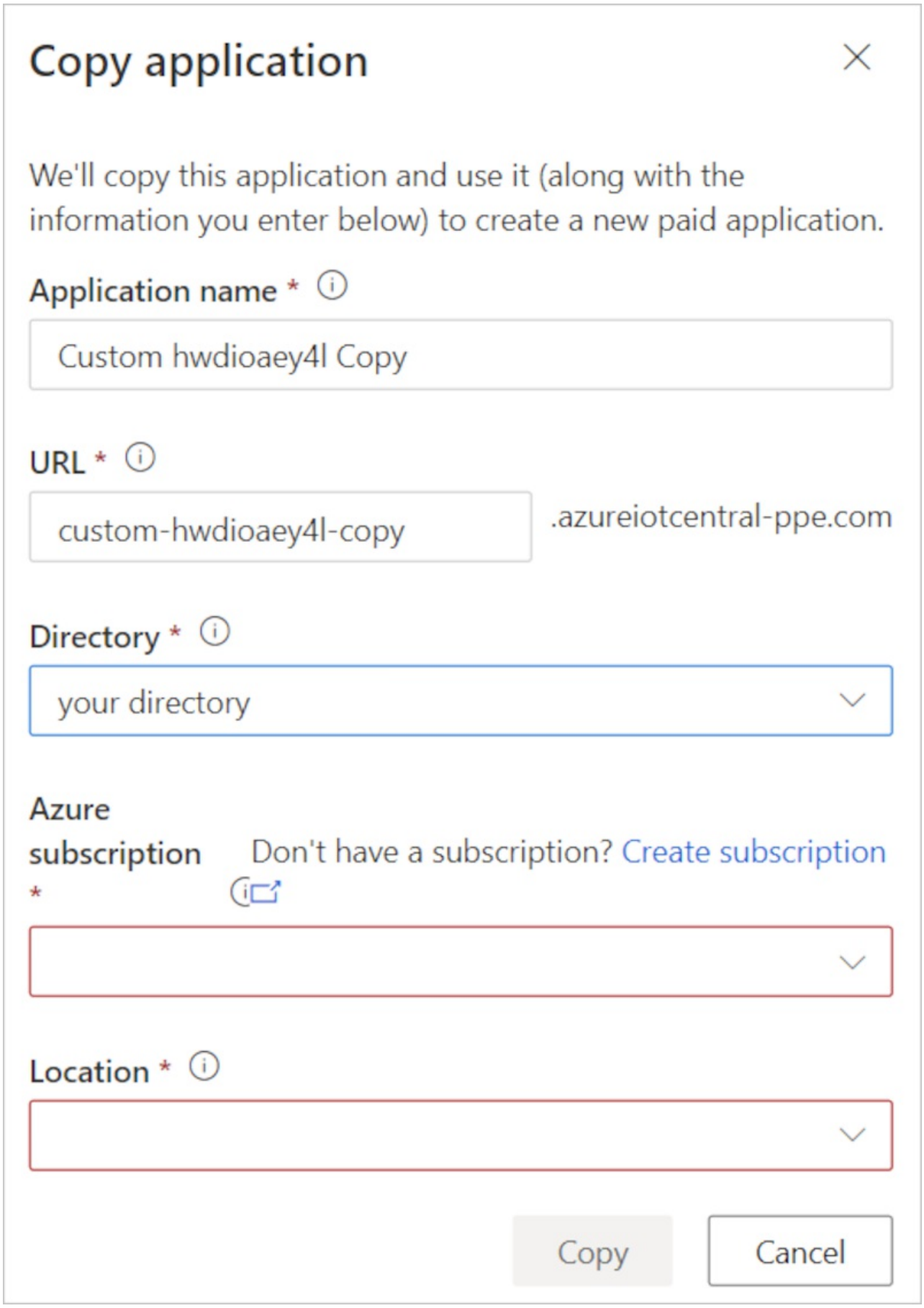
You have two options:

- You can create a copy of your application if you just need to create a duplicate copy of your application.
- You can create an application template from your application if you plan to create multiple copies.

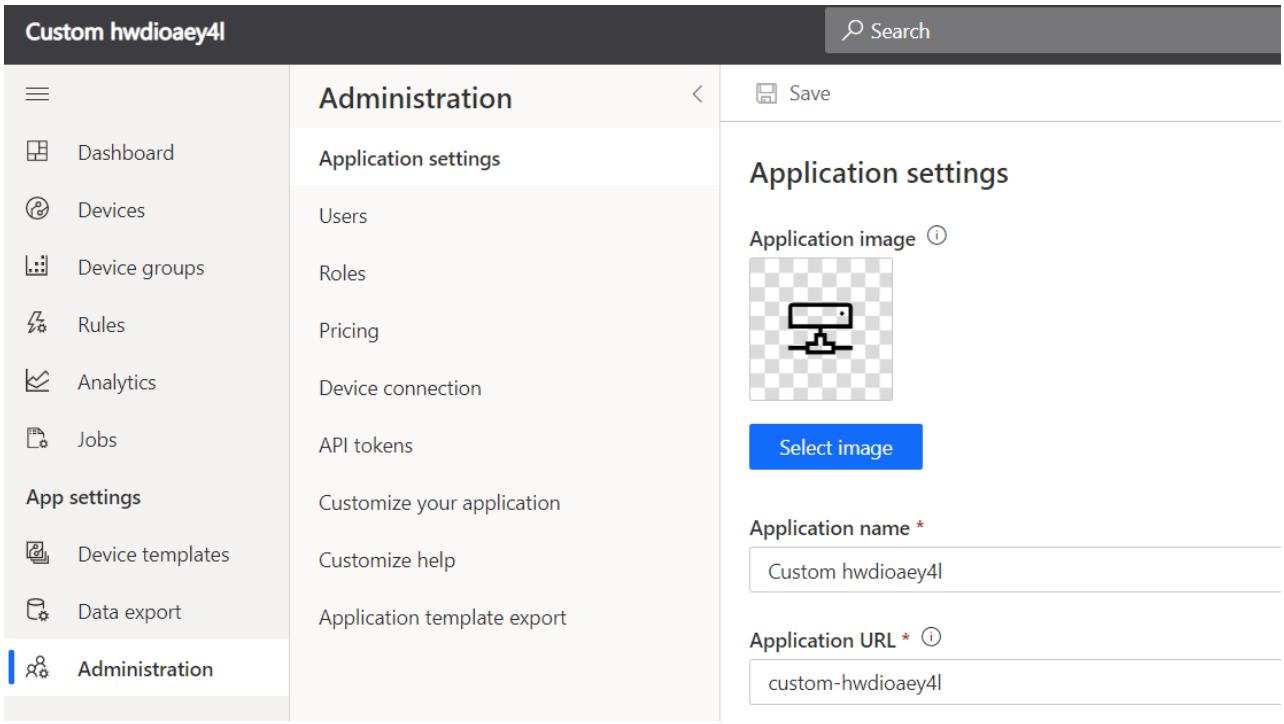
Copy your application

You can create a copy of any application, minus any device instances, device data history, and user data. The copy uses a standard pricing plan that you'll be billed for. You can't create an application that uses the free pricing plan by copying an application.

Select **Copy**. In the dialog box, enter the details for the new application. Then select **Copy** to confirm that you want to continue. To learn more about the fields in the form, see the Create an application quickstart.



After the app copy operation succeeds, you can navigate to the new application using the link.



Copying an application also copies the definition of rules and email action. Some actions, such as Flow and Logic Apps, are tied to specific rules via the Rule ID. When a rule is copied to a different application, it gets its own Rule ID. In this case, users will have to create a new action and then associate the new rule with it. In general, it's a good idea to check the rules and actions to make sure they're up-to-date in the new app.

> **WARNING**
> If a dashboard includes tiles that display information about specific devices, then those tiles show **The requested resource was not found** in the new application. You must reconfigure these tiles to display information about devices in your new application.

Create an application template

When you create an Azure IoT Central application, you have a choice of built-in sample templates. You can also create your own application templates from existing IoT Central applications. You can then use your own application templates when you create new applications.

When you create an application template, it includes the following items from your existing application:

- The default application dashboard, including the dashboard layout and all the tiles you've defined.
- Device templates, including measurements, settings, properties, commands, and dashboard.
- Rules. All rule definitions are included. However actions, except for email actions, aren't included.
- Device sets, including their conditions and dashboards.

> **WARNING**
> If a dashboard includes tiles that display information about specific devices, then those tiles show **The requested resource was not found** in the new application. You must reconfigure these tiles to display information about devices in your new application.

When you create an application template, it doesn't include the following items:

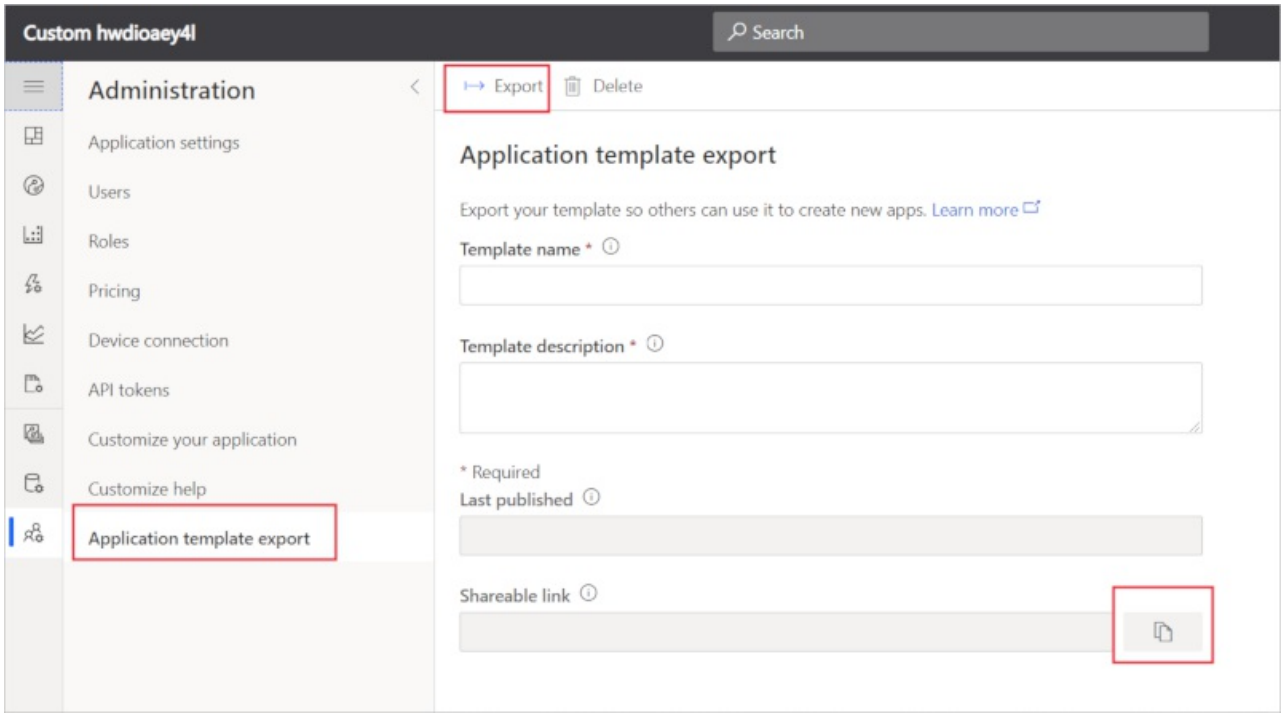
- Devices
- Users

- Job definitions
- Continuous data export definitions

Add these items manually to any applications created from an application template.

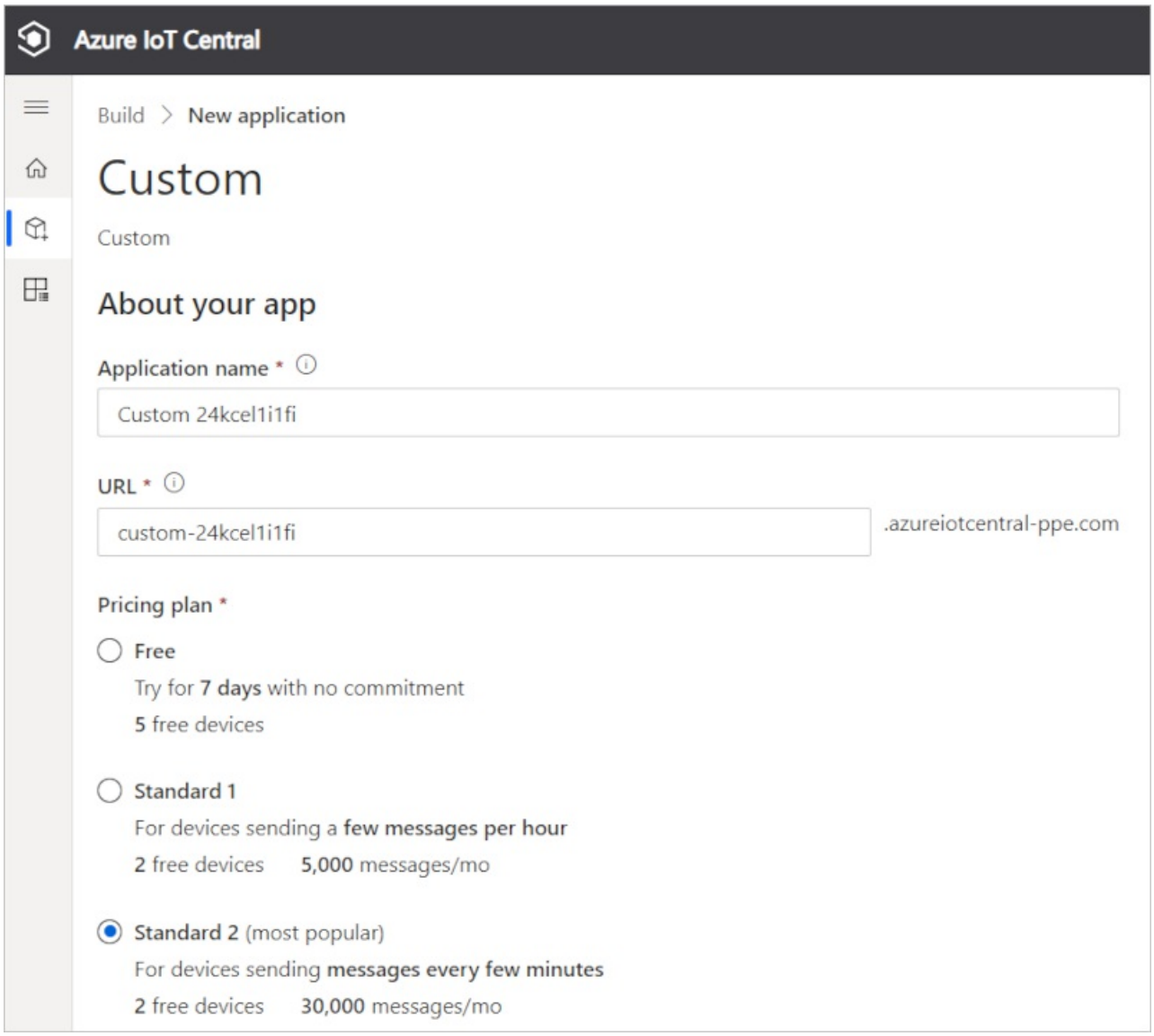
To create an application template from an existing IoT Central application:

1. Go to the **Administration** section in your application.
2. Select **Application Template Export**.
3. On the **Application Template Export** page, enter a name and description for your template.
4. Select the **Export** button to create the application template. You can now copy the **Shareable Link** that enables someone to create a new application from the template:



Use an application template

To use an application template to create a new IoT Central application, you need a previously created **Shareable Link**. Paste the **Shareable Link** into your browser's address bar. The **Create an application** page displays with your custom application template selected:



Select your pricing plan and fill out the other fields on the form. Then select **Create** to create a new IoT Central application from the application template.

Manage application templates

On the **Application Template Export** page, you can delete or update the application template.

If you delete an application template, you can no longer use the previously generated shareable link to create new applications.

To update your application template, change the template name or description on the **Application Template Export** page. Then select the **Export** button again. This action generates a new **Shareable link** and invalidates any previous **Shareable link** URL.

Next steps

Now that you've learned how to use application templates, the suggested next step is to learn how to Manage IoT Central from the Azure portal

Manage users and roles in your IoT Central application

3/27/2020 • 6 minutes to read • Edit Online

This article describes how, as an administrator, you can add, edit, and delete users in your Azure IoT Central application. The article also describes how to manage roles in your Azure IoT Central application.

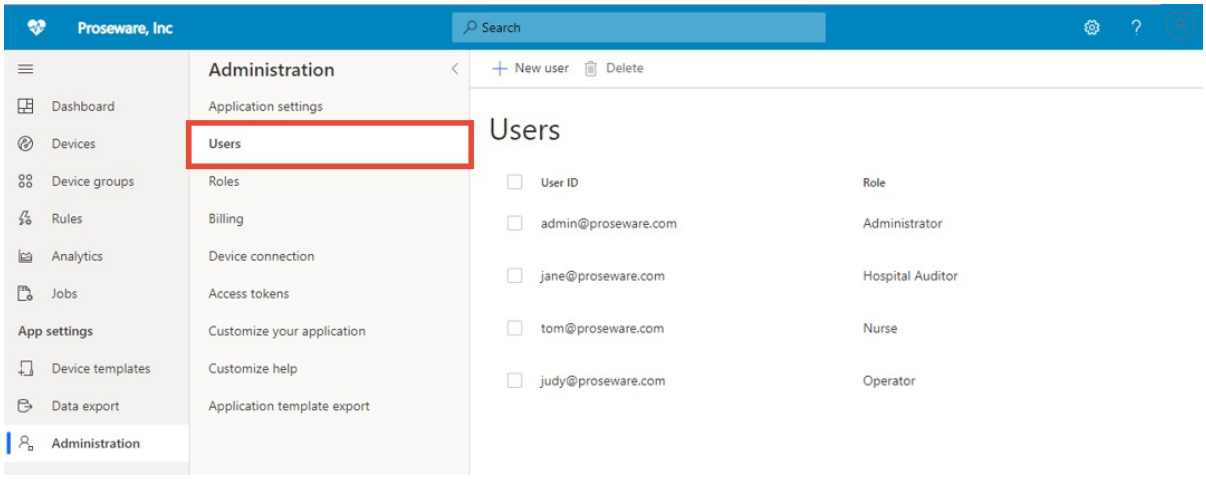
To access and use the **Administration** section, you must be in the **Administrator** role for an Azure IoT Central application. If you create an Azure IoT Central application, you're automatically added to the **Administrator** role for that application.

Add users

Every user must have a user account before they can sign in and access an Azure IoT Central application. Microsoft Accounts and Azure Active Directory accounts are supported in Azure IoT Central. Azure Active Directory groups aren't currently supported in Azure IoT Central.

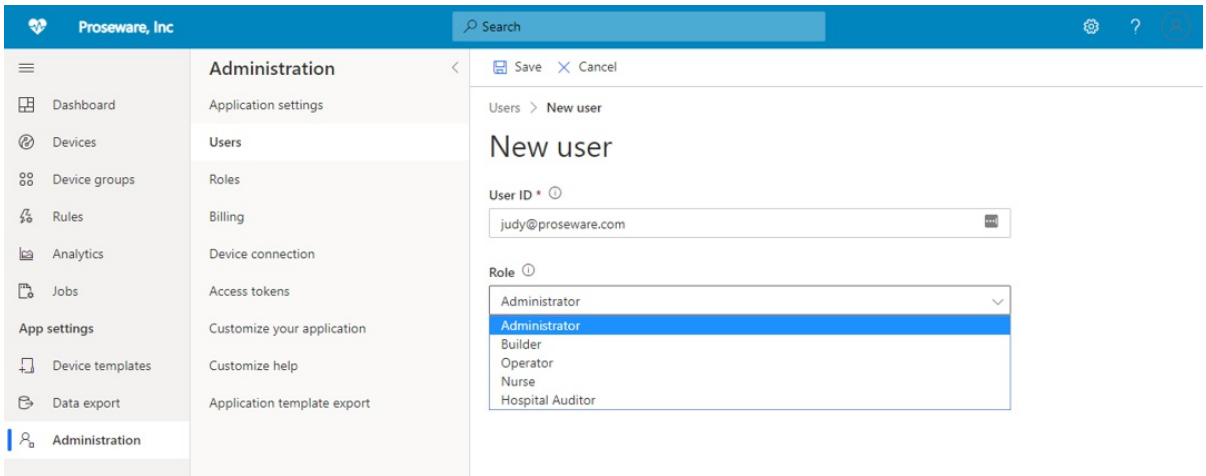
For more information, see Microsoft account help and Quickstart: Add new users to Azure Active Directory.

1. To add a user to an IoT Central application, go to the **Users** page in the **Administration** section.



2. To add a user, on the **Users** page, choose **+ Add user**.

3. Choose a role for the user from the **Role** drop-down menu. Learn more about roles in the Manage roles section of this article.



> **NOTE**
> A user who is in a custom role that grants them the permission to add other users, can only add users to a role with same or fewer permissions than their own role.

If an IoT Central user ID is deleted from Azure Active Directory and then readded, the user won't be able to sign in the IoT Central application. To re-enable access, the IoT Central administrator should delete and readd the user in the application.

Edit the roles that are assigned to users

Roles can't be changed after they're assigned. To change the role that's assigned to a user, delete the user, and then add the user again with a different role.

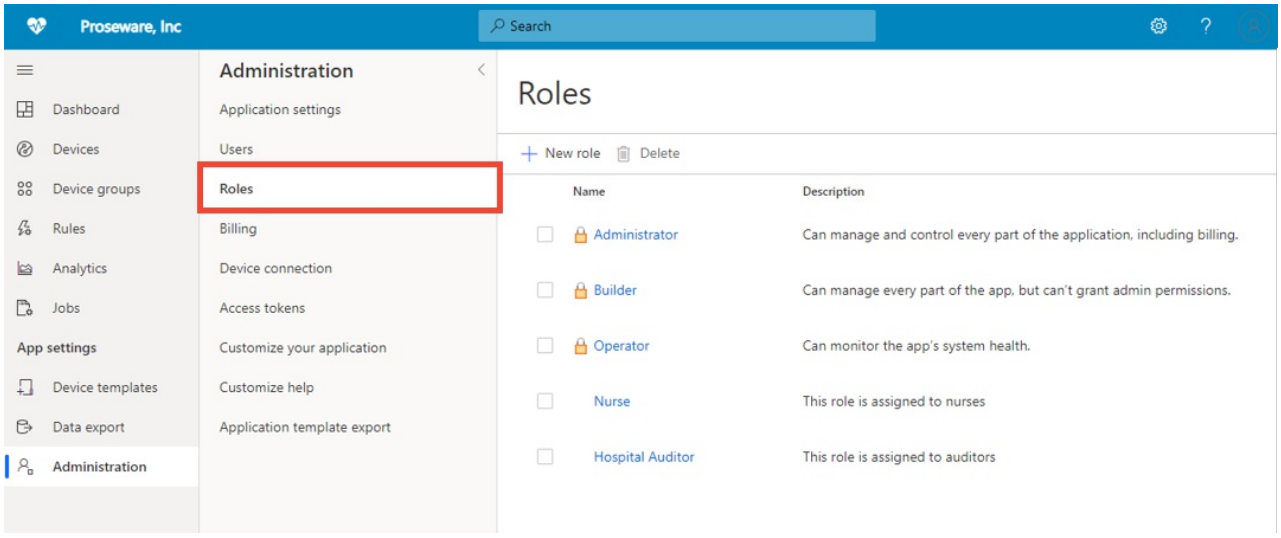
> **NOTE**
> The roles assigned are specific to IoT Central application and cannot be managed from the Azure Portal.

Delete users

To delete users, select one or more check boxes on the **Users** page. Then select **Delete**.

Manage roles

Roles enable you to control who within your organization is allowed to do various tasks in IoT Central. There are three built-in roles you can assign to users of your application. You can also create custom roles if you require finer-grained control.



Administrator

Users in the **Administrator** role can manage and control every part of the application, including billing.

The user who creates an application is automatically assigned to the **Administrator** role. There must always be at least one user in the **Administrator** role.

Builder

Users in the **Builder** role can manage every part of the app, but can't make changes on the Administration or Continuous Data Export tabs.

Operator

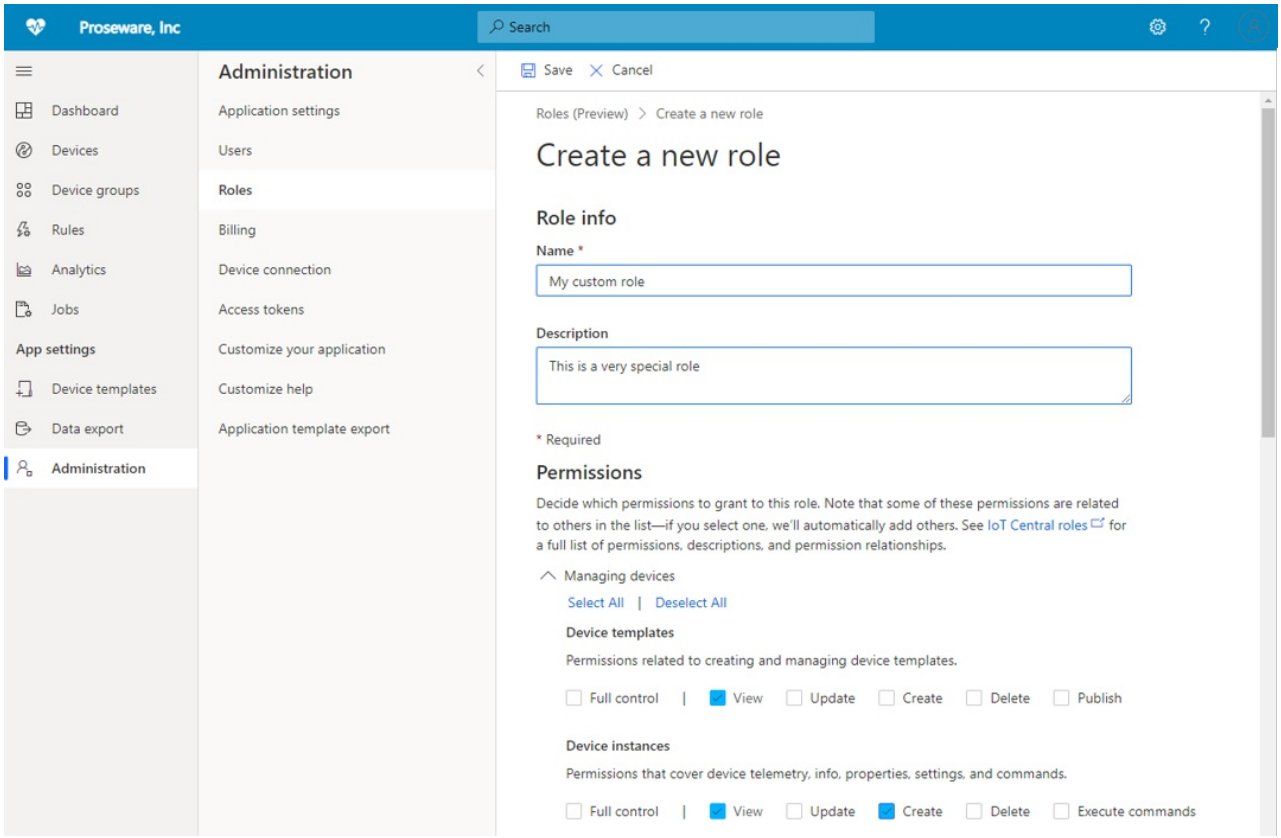
Users in the **Operator** role can monitor device health and status. They aren't allowed to make changes to device

templates or to administer the application. Operators can add and delete devices, manage device sets, and run analytics and jobs.

Create a custom role

If your solution requires finer-grained access controls, you can create custom roles with custom sets of permissions. To create a custom role, navigate to the **Roles** page in the **Administration** section of your application. Then select **+ New role**, and add a name and description for your role. Select the permissions your role requires and then select **Save**.

You can add users to your custom role in the same way that you add users to a built-in role.



Custom role options

When you define a custom role, you choose the set of permissions that a user is granted if they're a member of the role. Some permissions are dependent on others. For example, if you add the **Update application dashboards** permission to a role, the **View application dashboards** permission is automatically added. The following tables summarize the available permissions, and their dependencies, you can use when creating custom roles.

Managing devices

Device template permissions

NAME	DEPENDENCIES
View	None
Manage	View Other dependencies: View device instances
Full Control	View, Manage Other dependencies: View device instances

Device instance permissions

NAME	DEPENDENCIES
View	None Other dependencies: View device templates and device groups
Update	View Other dependencies: View device templates and device groups
Create	View Other dependencies: View device templates and device groups
Delete	View Other dependencies: View device templates and device groups
Execute Commands	Update, View Other dependencies: View device templates and device groups
Full Control	View, Update, Create, Delete, Execute Commands Other dependencies: View device templates and device groups

Device groups permissions

NAME	DEPENDENCIES
View	None Other dependencies: View device templates and device instances
Update	View Other dependencies: View device templates and device instances
Create	View, Update Other dependencies: View device templates and device instances
Delete	View Other dependencies: View device templates and device instances
Full Control	View, Update, Create, Delete Other dependencies: View device templates and device instances

Device connectivity management permissions

NAME	DEPENDENCIES
Read instance	None Other dependencies: View device templates, device groups, device instances
Manage instance	None
Read global	None

NAME	DEPENDENCIES
Manage global	Read Global
Full Control	Read instance, Manage instance, Read global, Manage global. Other dependencies: View device templates, device groups, device instances

Jobs permissions

NAME	DEPENDENCIES
View	None Other dependencies: View device templates, device instances, and device groups
Update	View Other dependencies: View device templates, device instances, and device groups
Create	View, Update Other dependencies: View device templates, device instances, and device groups
Delete	View Other dependencies: View device templates, device instances, and device groups
Execute	View Other dependencies: View device templates, device instances, and device groups; Update device instances; Execute commands on device instances
Full Control	View, Update, Create, Delete, Execute Other dependencies: View device templates, device instances, and device groups; Update device instances; Execute commands on device instances

Rules permissions

NAME	DEPENDENCIES
View	None Other dependencies: View device templates
Update	View Other dependencies: View device templates
Create	View, Update Other dependencies: View device templates
Delete	View Other dependencies: View device templates
Full Control	View, Update, Create, Delete Other dependencies: View device templates

Managing the app

Application settings permissions

NAME	DEPENDENCIES
View	None
Update	View
Copy	View Other dependencies: View device templates, device instances, device groups, dashboards, data export, branding, help links, custom roles, rules
Delete	View
Full Control	View, Update, Copy, Delete Other dependencies: View device templates, device groups, application dashboards, data export, branding, help links, custom roles, rules

Application template export permissions

NAME	DEPENDENCIES
View	None
Export	View Other dependencies: View device templates, device instances, device groups, dashboards, data export, branding, help links, custom roles, rules
Full Control	View, Export Other dependencies: View device templates, device groups, application dashboards, data export, branding, help links, custom roles, rules

Billing permissions

NAME	DEPENDENCIES
Manage	None
Full Control	Manage

Managing users and roles

Custom roles permissions

NAME	DEPENDENCIES
View	None
Update	View
Create	View, Update

NAME	DEPENDENCIES
Delete	View
Full Control	View, Update, Create, Delete

User management permissions

NAME	DEPENDENCIES
View	None Other dependencies: View custom roles
Add	View Other dependencies: View custom roles
Delete	View Other dependencies: View custom roles
Full Control	View, Add, Delete Other dependencies: View custom roles

> **NOTE**
> A user who is in a custom role that grants them the permission to add other users, can only add users to a role with same or fewer permissions than their own role.

Customizing the app

Application dashboard permissions

NAME	DEPENDENCIES
View	None
Update	View
Create	View, Update
Delete	View
Full Control	View, Update, Create, Delete

Personal dashboards permissions

NAME	DEPENDENCIES
View	None
Update	View
Create	View, Update
Delete	View

NAME	DEPENDENCIES
Full Control	View, Update, Create, Delete

Branding, favicon, and colors permissions

NAME	DEPENDENCIES
View	None
Update	View
Full Control	View, Update

Help links permissions

NAME	DEPENDENCIES
View	None
Update	View
Full Control	View, Update

Extending the app

Data export permissions

NAME	DEPENDENCIES
View	None
Update	View
Create	View, Update
Delete	View
Full Control	View, Update, Create, Delete

API token permissions

NAME	DEPENDENCIES
View	None
Create	View
Delete	View
Full Control	View, Create, Delete

Next steps

Now that you've learned about how to manage users and roles in your Azure IoT Central application, the suggested next step is to learn how to Manage your bill.

Manage your bill in an IoT Central application

3/24/2020 • 2 minutes to read • Edit Online

This article describes how, as an administrator, you can manage your bill in Azure IoT Central application in the administration section. You will learn how you can move your application from the free pricing plan to a standard pricing plan, and also how to upgrade or downgrade your pricing plan.

To access and use the **Administration** section, you must be in the *Administrator* role or have a *custom user role* that allows you to view billing for an Azure IoT Central application. If you create an Azure IoT Central application, you're automatically assigned to the **Administrator** role for that application.

Move from free to standard pricing plan

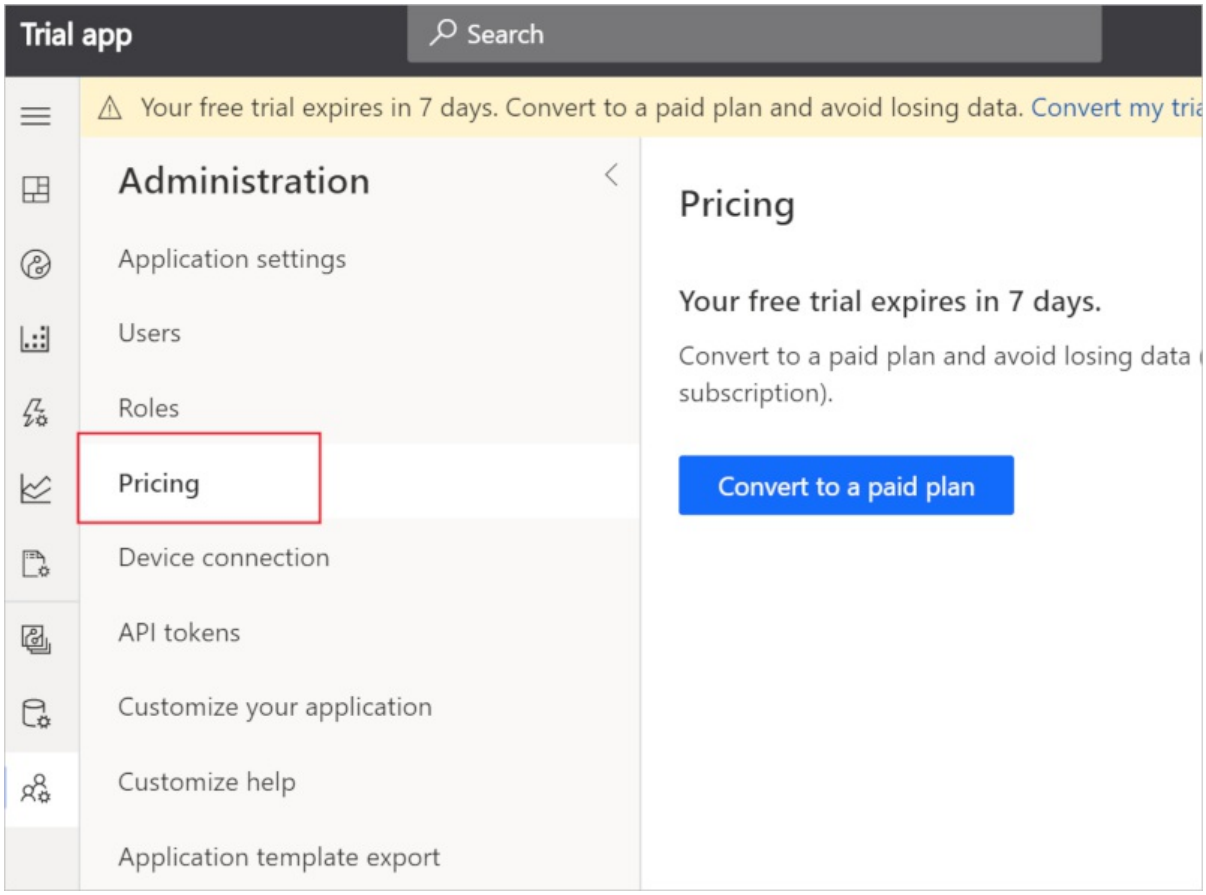
- Applications that use the free pricing plan are free for seven days before they expire. In order to avoid losing data you can move them to a standard pricing plan at any time before they expire.
- Applications that use a standard pricing plan are charged per device, with the first two devices free, per application.

Learn more about pricing on the Azure IoT Central pricing page.

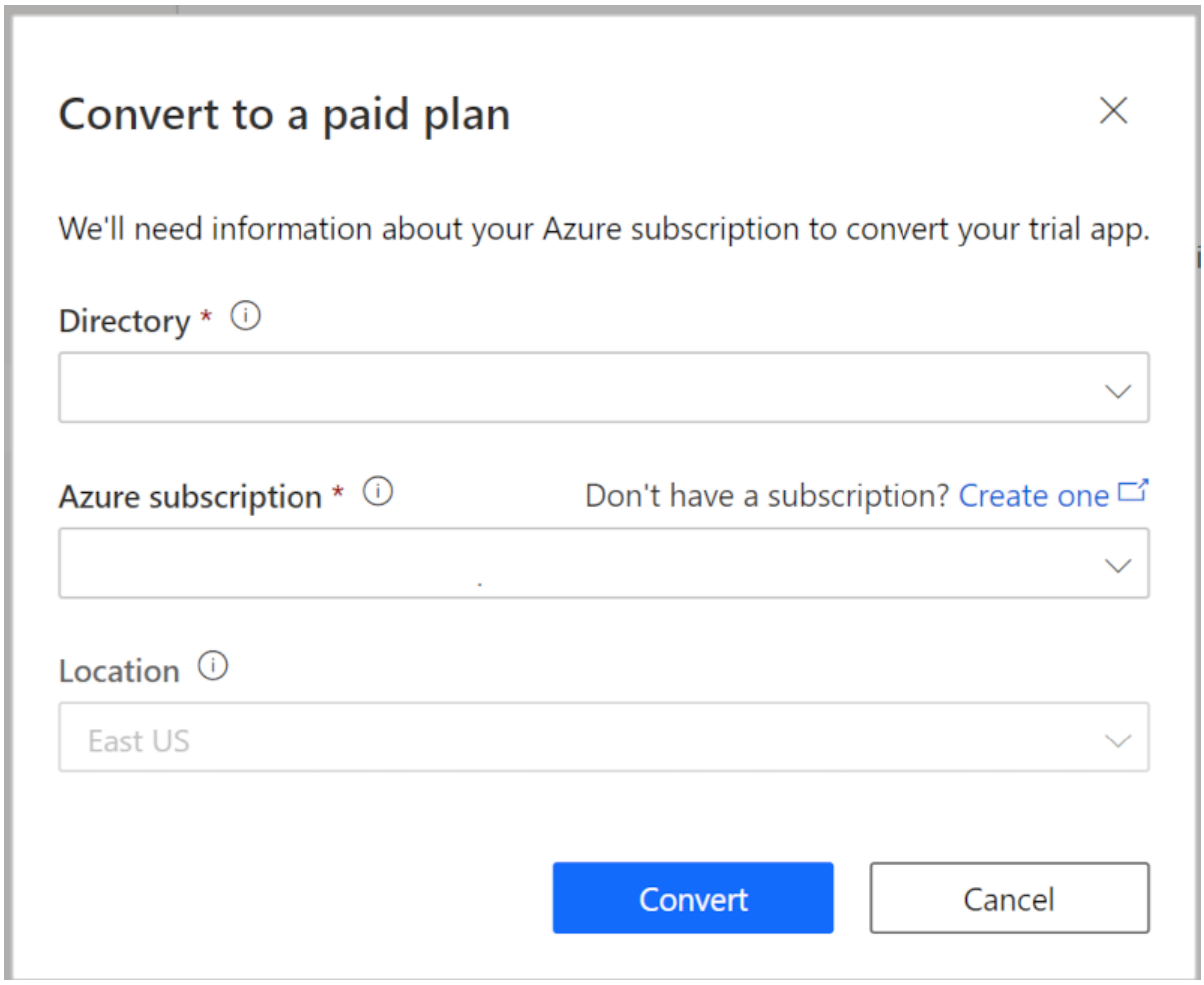
In the pricing section, you can move your application from the free to a standard pricing plan.

To complete this self-service process, follow these steps:

1. Go to the **Pricing** page in the **Administration** section.



2. Select **Convert to a paid plan**.



3. Select the appropriate Azure Active Directory, and then the Azure subscription to use for your application that uses a paid plan.
4. After you select **Convert**, your application now uses a paid plan and you start getting billed.

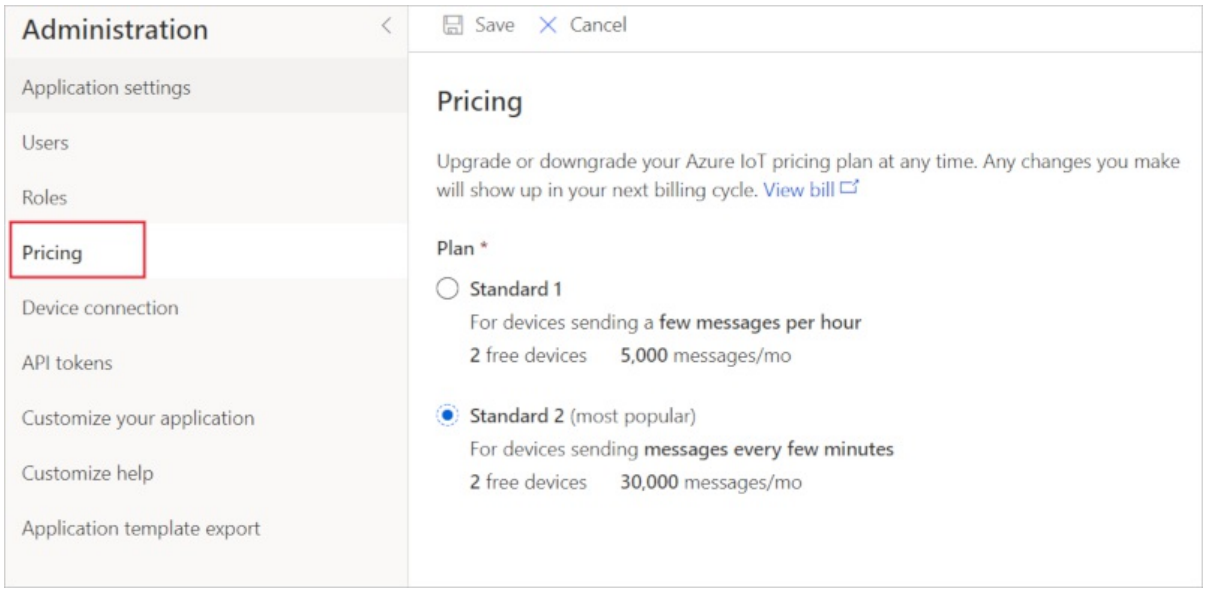
> **NOTE**
> By default, you are converted to a *Standard 2* pricing plan.

How to change your application pricing plan

Applications that use a standard pricing plan are charged per device, with the first two devices free, per application.

In the pricing section, you can upgrade or downgrade your Azure IoT pricing plan at any time.

1. Go to the **Pricing** page in the **Administration** section.



2. Select the **Plan** and click **Save** to upgrade or downgrade.

View your bill

1. Select the appropriate Azure Active Directory, and then the Azure subscription to use for your application that uses a paid plan.

2. After you select **Convert**, your application now uses a paid plan and you start getting billed.

> **NOTE**
> By default, you are converted to a *Standard 2* pricing plan.

Next steps

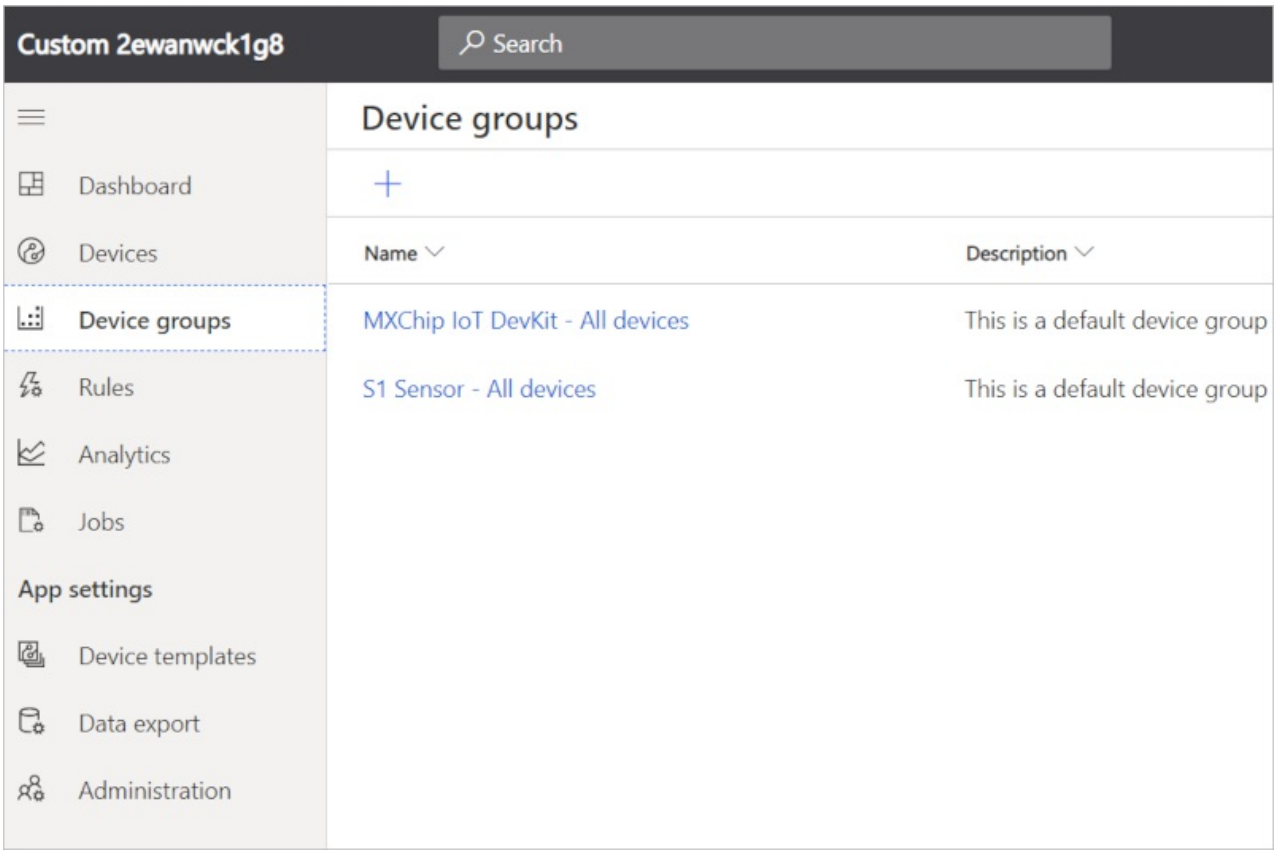
Now that you've learned about how to manage your bill in Azure IoT Central application, the suggested next step is to learn about Customize application UI in Azure IoT Central.

Customize the Azure IoT Central UI

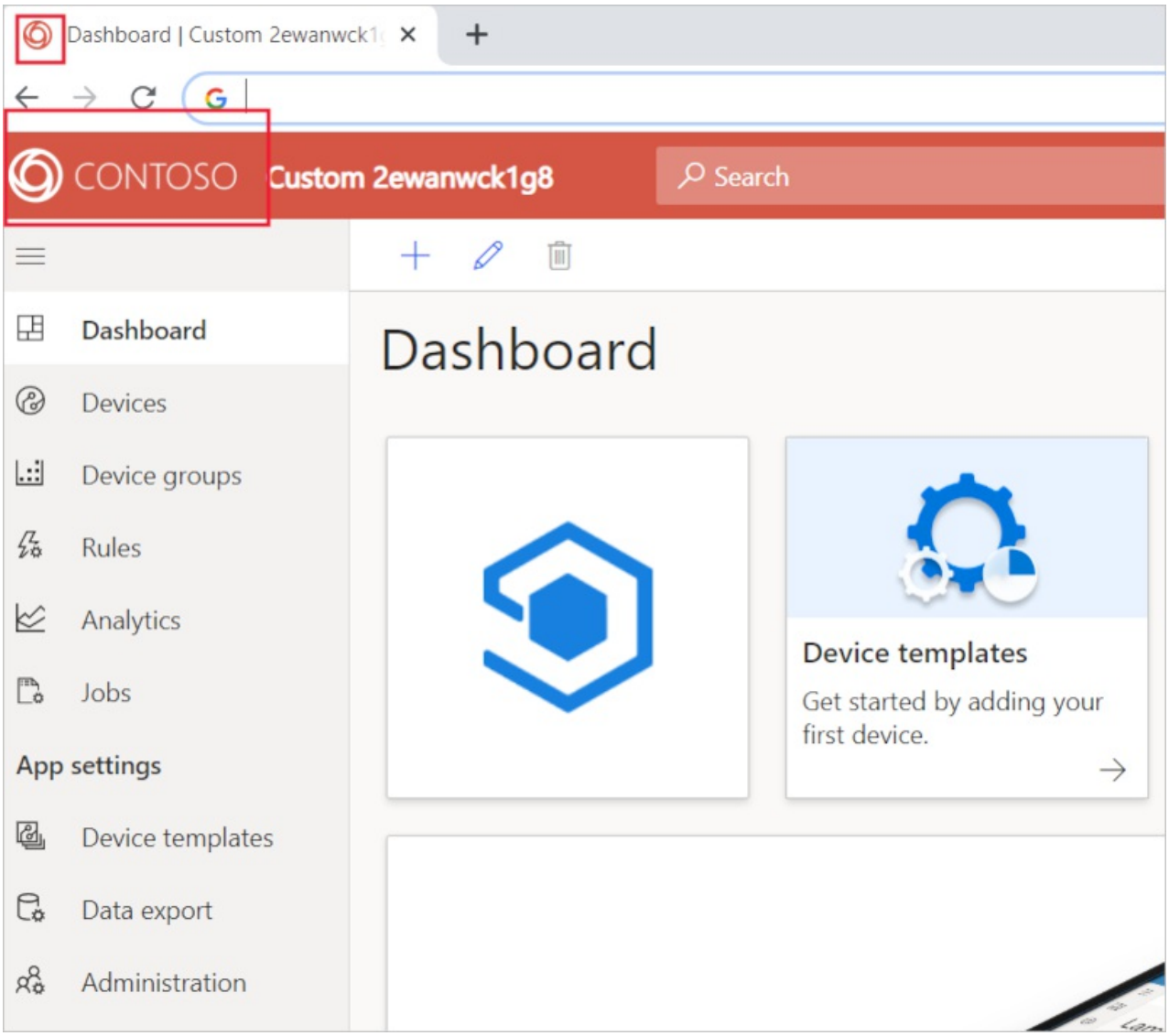
3/24/2020 • 2 minutes to read • Edit Online

This article describes how, as an administrator, you can customize the UI of your application by applying custom themes and modifying the help links to point to your own custom help resources.

The following screenshot shows a page using the standard theme:

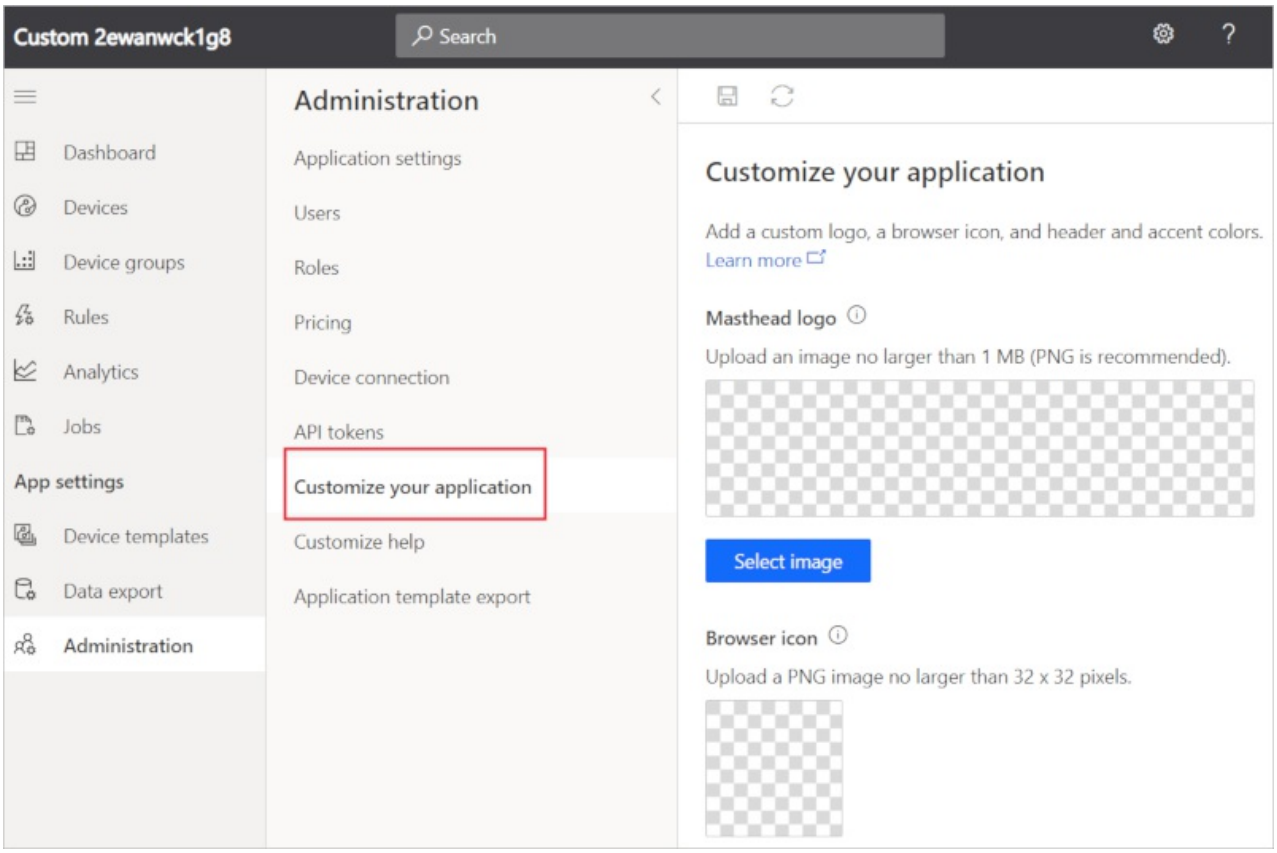


The following screenshot shows a page using a custom screenshot with the customized UI elements highlighted:



Create theme

To create a custom theme, navigate to the **Customize your application** page in the **Administration** section:



On this page, you can customize the following aspects of your application:

Application logo

A PNG image, no larger than 1 MB, with a transparent background. This logo displays to the left on the IoT Central application title bar.

If your logo image includes the name of your application, you can hide the application name text. For more information, see Manage your application.

Browser icon (favicon)

A PNG image, no larger than 32 x 32 pixels, with a transparent background. A web browser can use this image in the address bar, history, bookmarks, and browser tab.

Browser colors

You can change the color of the page header and the color used for accenting buttons and other highlights. Use a six character hex color value in the format `##ff6347`. For more information about **HEX Value** color notation, see HTML Colors.

> **NOTE**
> You can always revert back to the default options on the **Customize your application** page.

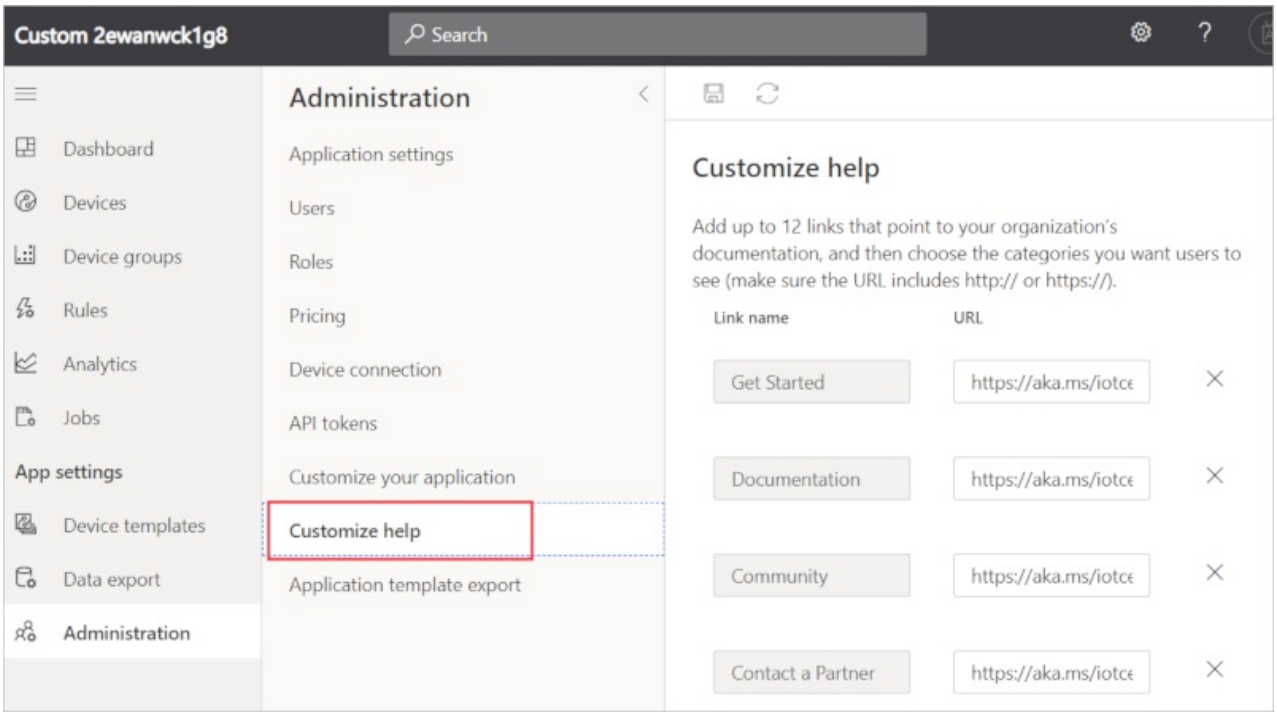
Changes for operators

If an administrator creates a custom theme, then operators and other users of your application can no longer choose a theme in **Settings**.

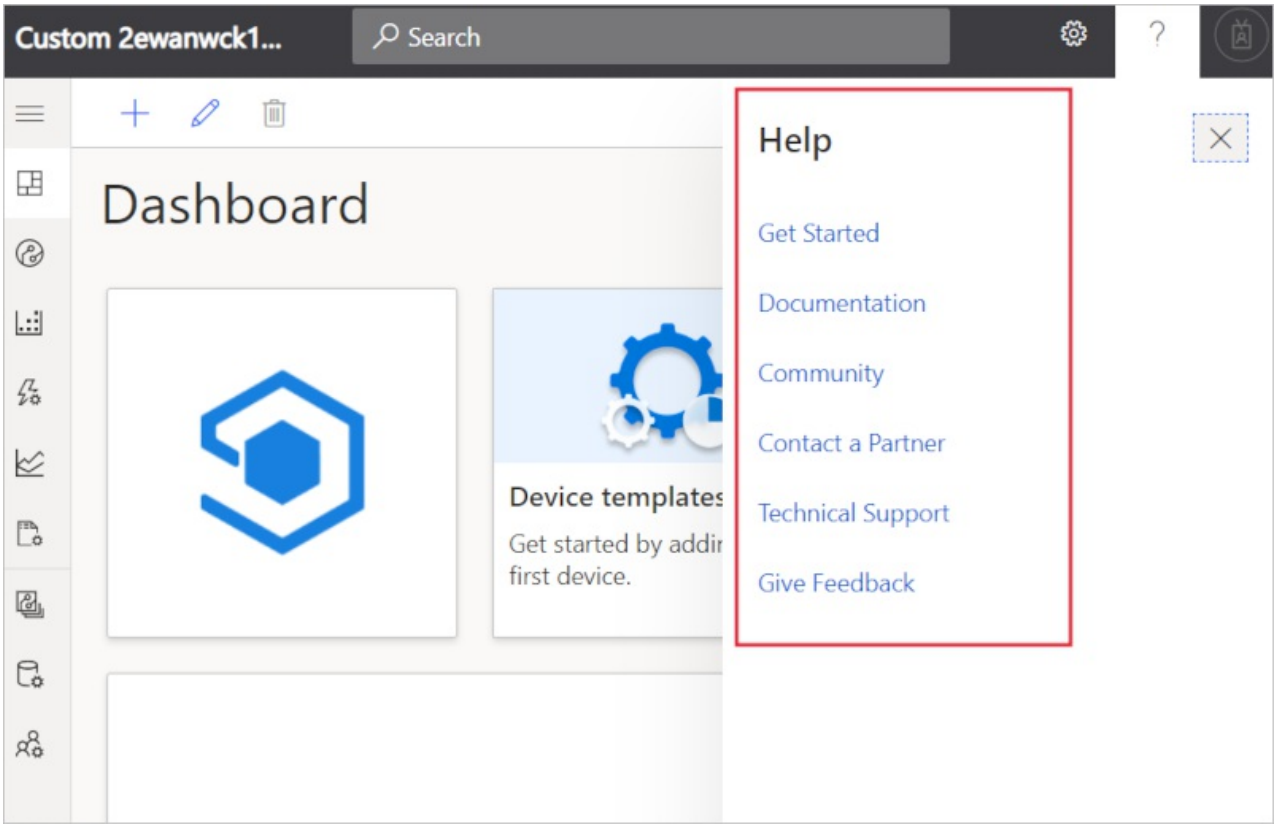
Replace help links

To provide custom help information to your operators and other users, you can modify the links on the application **Help** menu.

To modify the help links, navigate to the **Customize help** page in the **Administration** section:



You can also add new entries to the help menu and remove default entries:



NOTE

You can always revert back to the default help links on the **Customize help** page.

Next steps

Now that you've learned how to customize the UI in your IoT Central application, here are some suggested next steps:

- Administer your application

- Add tiles to your dashboard

Configure the application dashboard

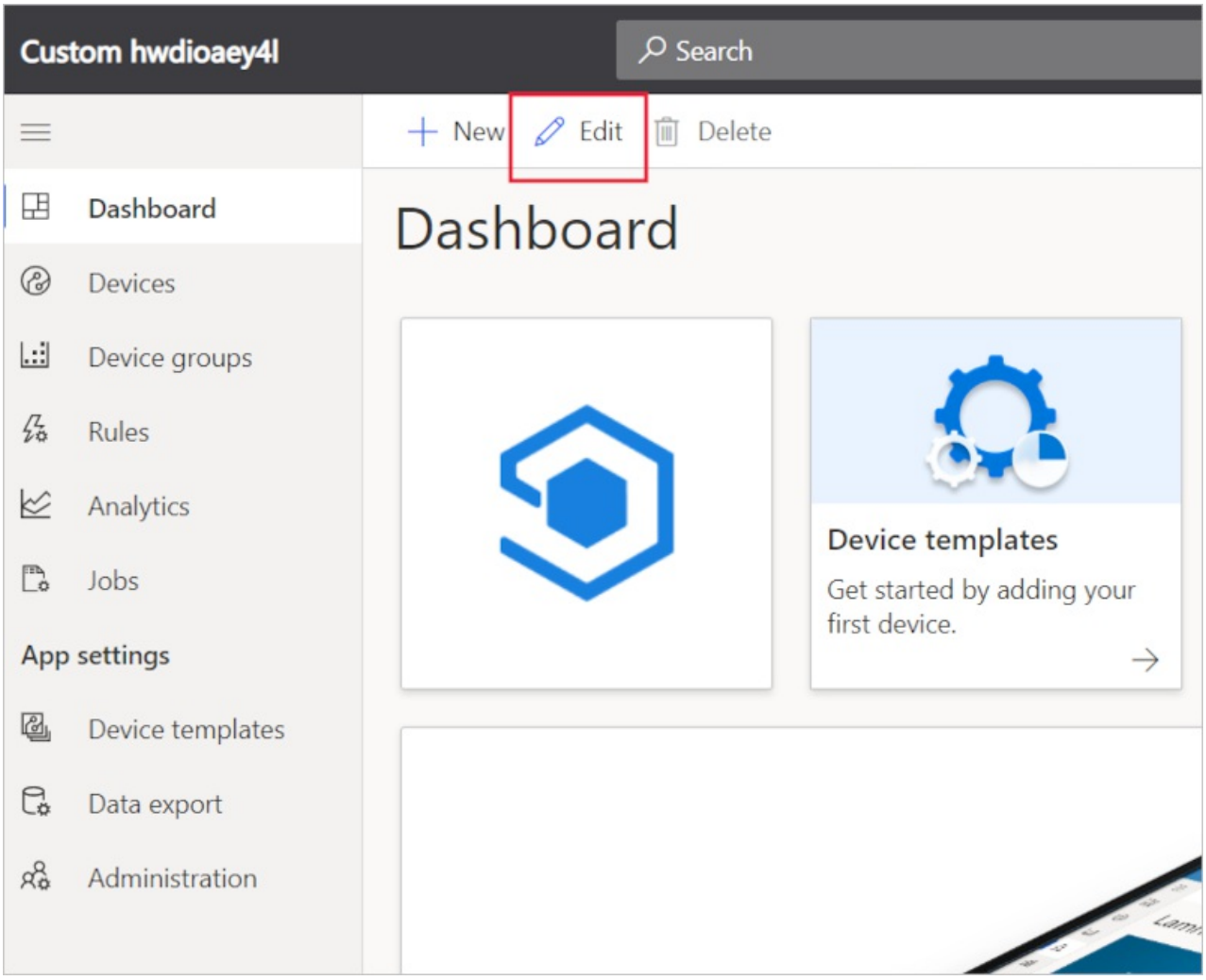
4/14/2020 • 4 minutes to read • Edit Online

The **Dashboard** is the page that loads when users who have access to the application navigate to the application's URL. If you created your application from one of the **Application Templates**, your application will have a pre-defined dashboard to start. If you created your application from the **Legacy application** application template, your dashboard will be blank to start.

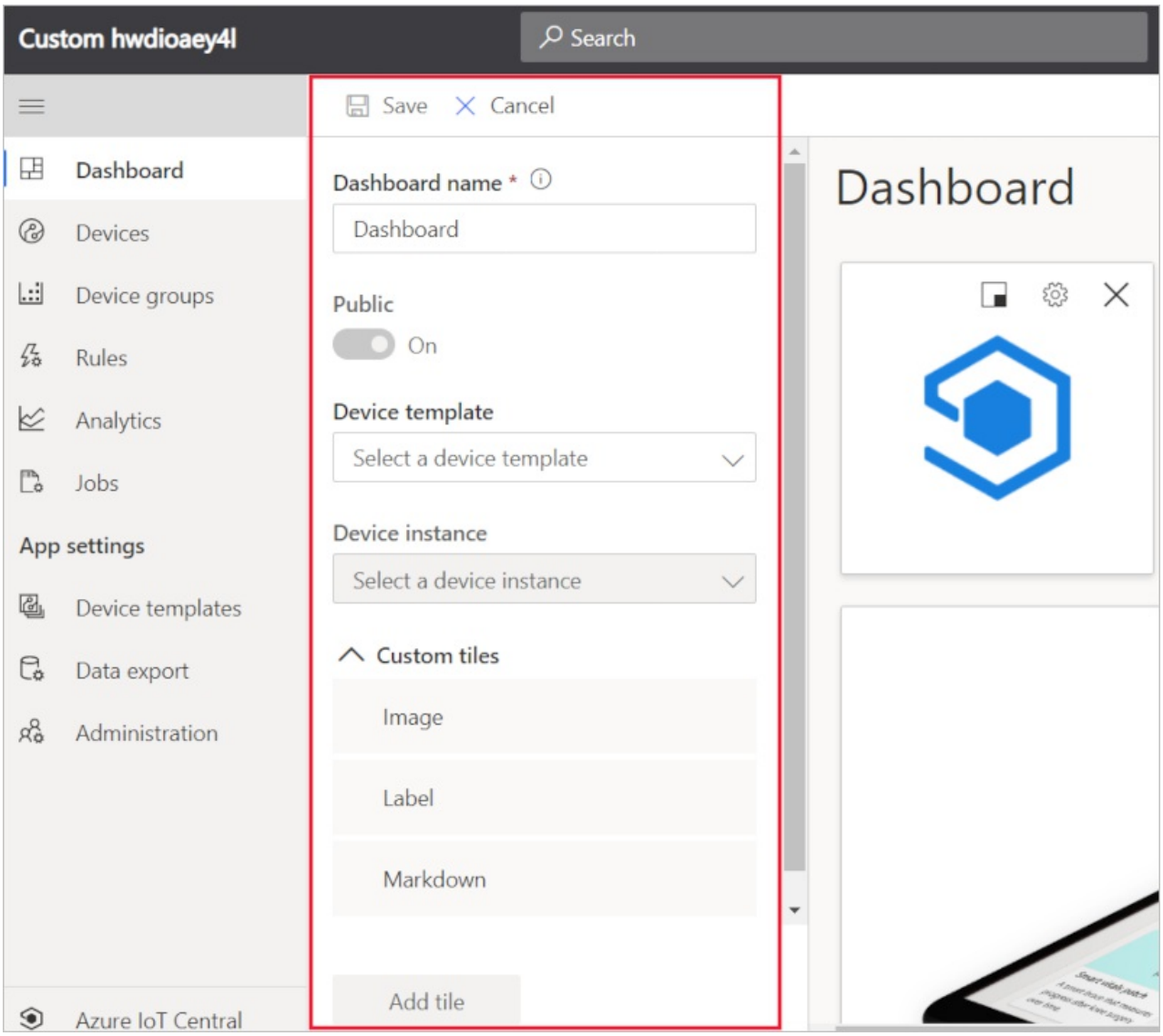
> **NOTE**
> Users can create multiple dashboards in addition to the default application dashboard. These dashboards can be personal to the user only, or shared across all users of the application.

Add tiles

The following screenshot shows the dashboard in an application created from the **Custom Application** template. To customize the default dashboard for your application, select **Edit** at the top-left of the page.

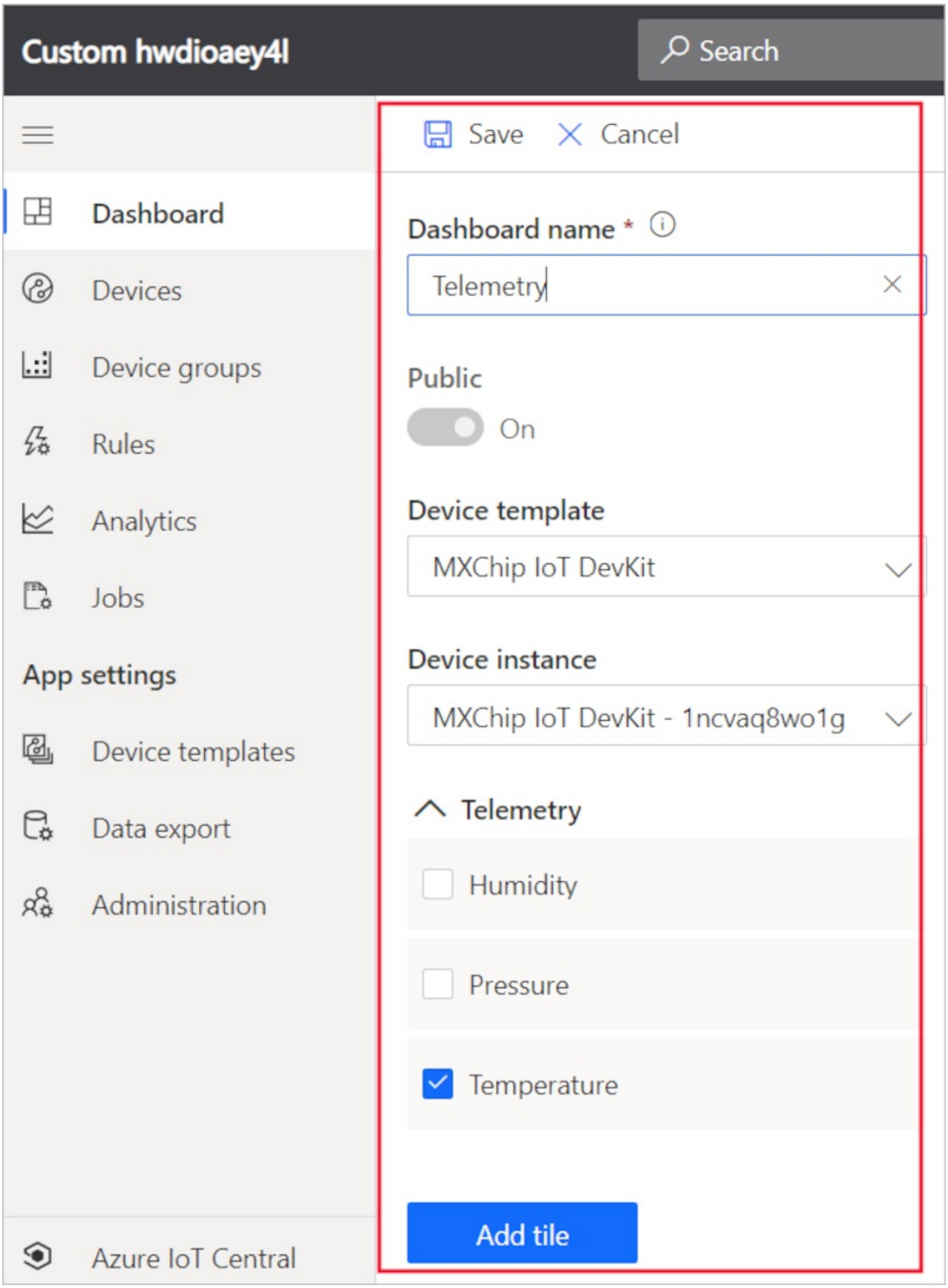


Selecting **Edit** opens the dashboard library panel. The library contains the tiles and dashboard primitives you can use to customize the dashboard.

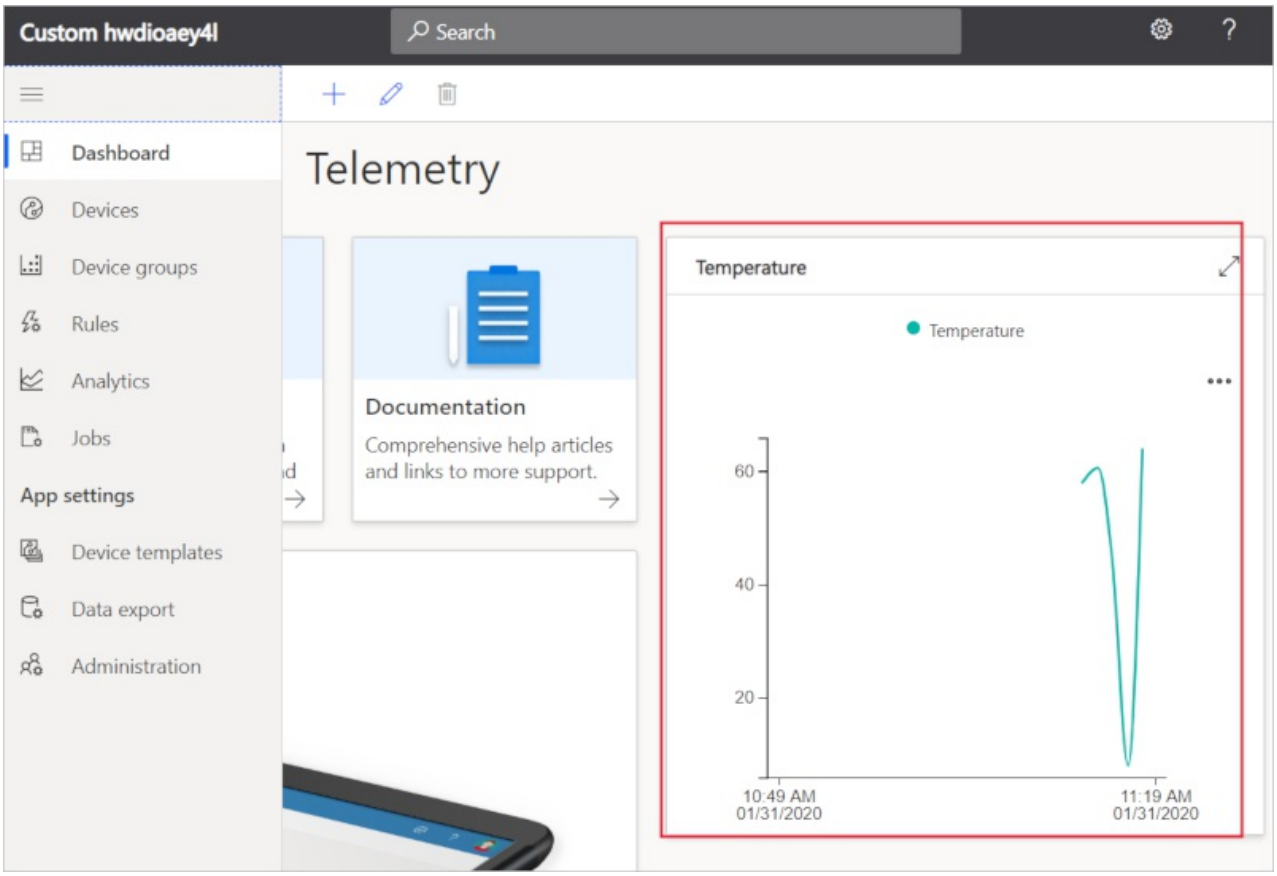


For example, you can add a **Telemetry** tile for the current temperature of the device. To do so:

1. Select a **Device Template**
2. Select a **Device Instance** for the device you want to see on a dashboard tile. Then you will see a list of the device's properties that can be used on the tile.
3. To create the tile on the dashboard, click on **Temperature** and drag it to the dashboard area. You can also click the checkbox next to **Temperature** and click **Combine**. The following screenshot shows selecting a Device Template and Device Instance then creating a Temperature Telemetry tile on the dashboard.
4. Select **Save** in the top left to save the tile to the dashboard.

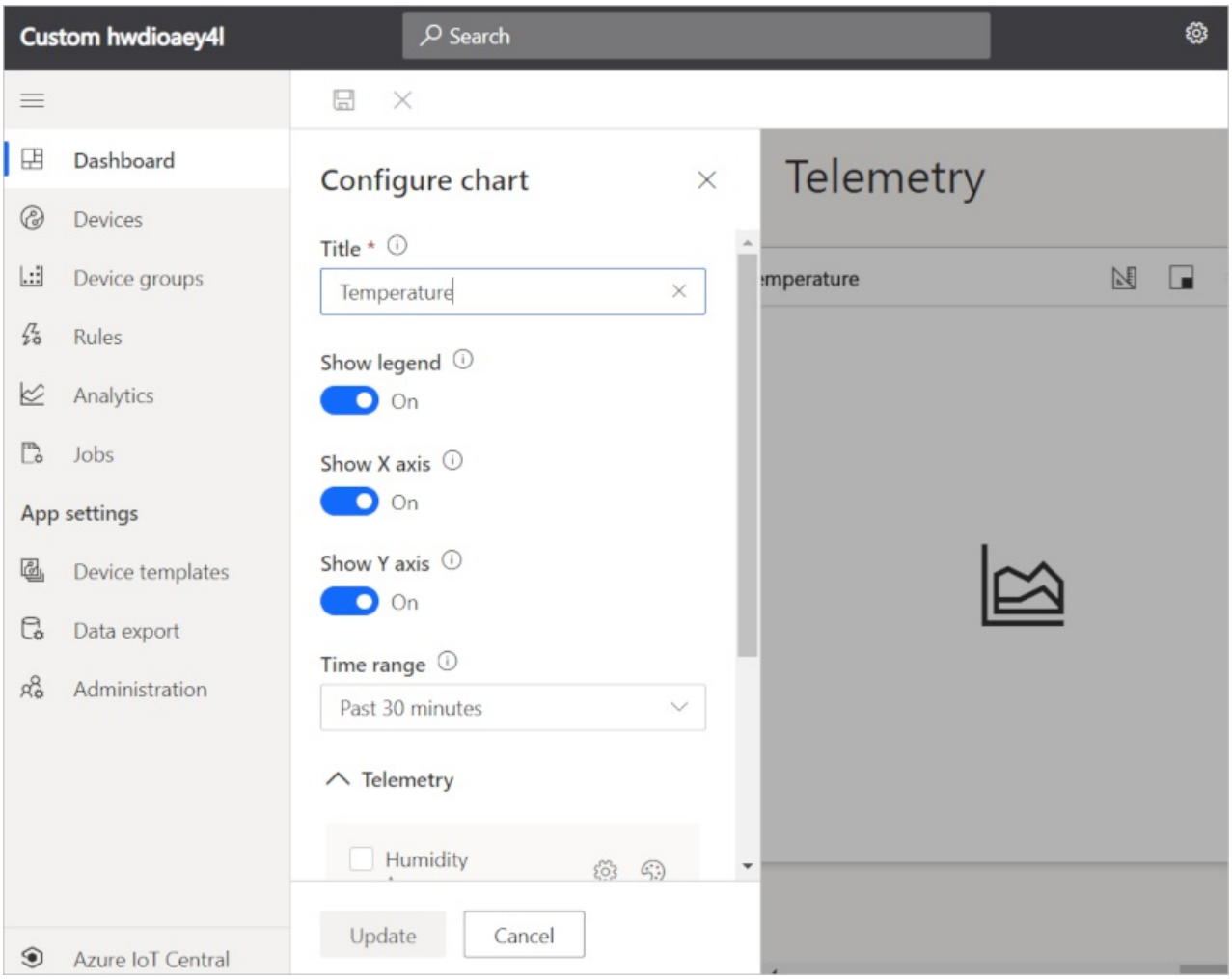


Now when an operator views the default application dashboard, they see the new tile with the **Temperature** for the device. Each tile has a pre-selected graph, chart, etc. that will be displayed when the tile is created. However, users can choose to edit and change this visualization.

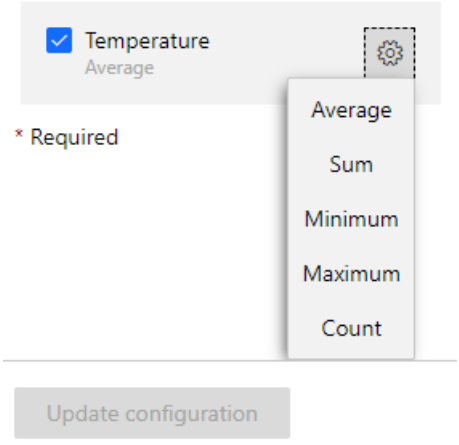


Edit tiles

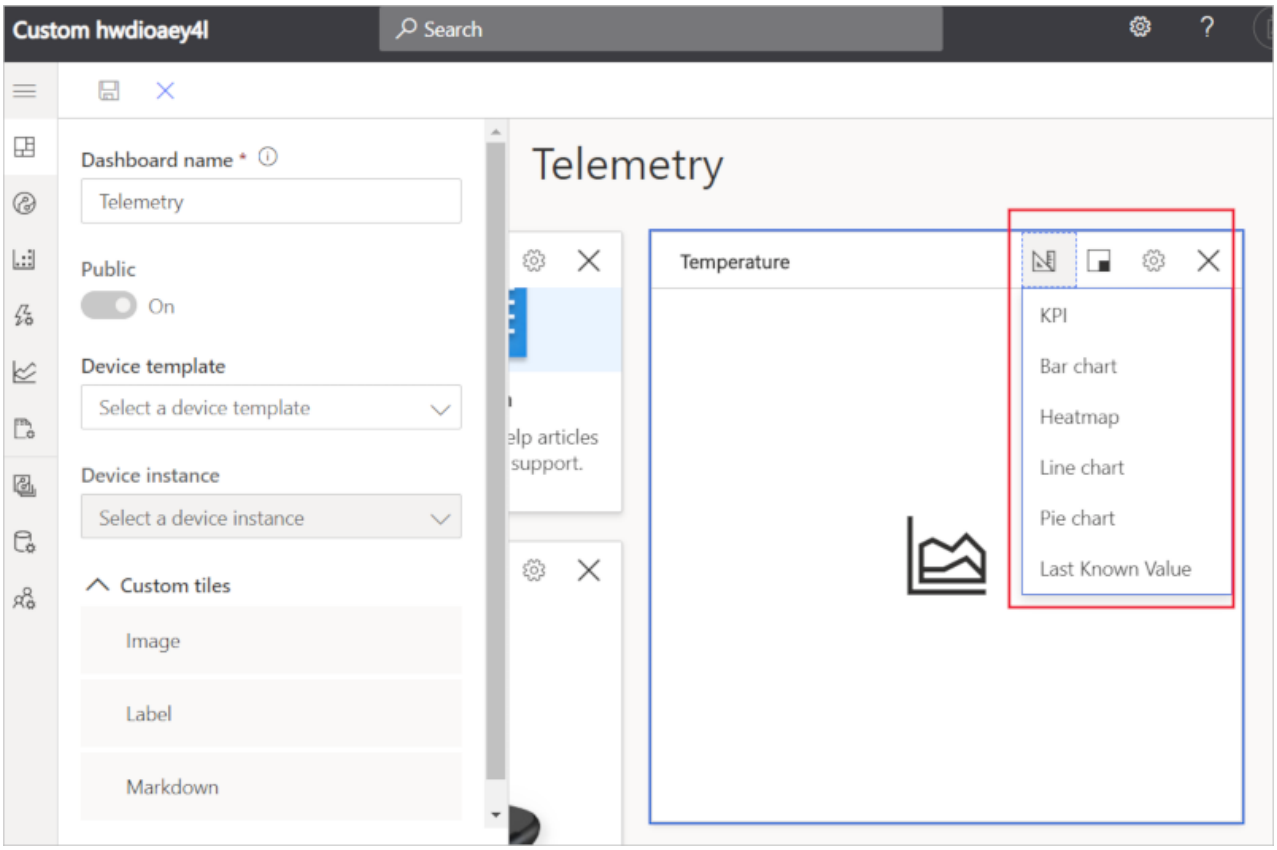
To edit a tile on the dashboard, first click **Edit** at the top left of the page, which will open edit mode for the dashboard and all its tiles.



Then click the **Gear** icon in the top-right corner of the tile you wish to edit. Here you can edit aspects of the tile including its title, its visualization, aggregation, etc.



You can also change the chart visualization by clicking the **Ruler** icon on the tile.



Tile types

The following table summarizes the usage of tiles in Azure IoT Central:

TILE	DASHBOARD	DESCRIPTION
Content	Application and device set dashboards	Markdown supported tiles are clickable tiles that display heading and description text. You can also use this tile as a link tile to enable a user to navigate to a URL related to your application.
Image	Application and device set dashboards	Image tiles display a custom image and can be clickable. Use an image tile to add graphics to a dashboard and optionally enable a user to navigate to a URL relevant to your application.
Label	Application dashboards	Label tiles display custom text on a dashboard. You can choose the size of the text. Use a label tile to add relevant information to the dashboard such descriptions, contact details, or help.
Map	Application and device dashboards	Map tiles display the location of a device on a map. You can also display up to 100 points of a device's location history. For example, you can a display sampled route of where a device has been on the past week.

TILE	DASHBOARD	DESCRIPTION
Line Chart	Application and device dashboards	Line chart tiles display a chart of aggregate measurement for a device for a time period. For example, you can display a line chart that shows the average temperature and pressure of a device for the last hour.
Bar Chart	Application and device dashboards	Bar chart tiles display a chart of aggregate measurements for a device for a time period. For example, you can display a bar chart that shows the average temperature and pressure of a device for the last hour.
Pie Chart	Application and device set dashboards	Pie chart tiles display a chart of aggregate measurements for a device for a time period.
Heat Map	Application and device set dashboards	Heat Map tiles display information about the device set, represented as colors.
Event History	Application and device dashboards	Event History tiles display the events for a device over a time period. For example, you can use it to show all the temperature changes for a device during the last hour.
State History	Application and device dashboards	State history tiles display the measurement values for a time period. For example, you can use it to show the temperature values for a device during the last hour.
KPI	Application and device dashboards	KPI tiles display an aggregate telemetry or event measurement for a time period. For example, you can use it to show the maximum temperature reached for a device during the last hour.
Last Known Value	Application and device dashboards	Last known value tiles display the latest value for a telemetry or state measurement. For example, you can use this tile to display the most recent measurements of temperature, pressure and humidity for a device.

Next steps

Now that you've learned how to configure your Azure IoT Central default application dashboard, you can Learn how to create a personal dashboard.

Manage IoT Central from the Azure portal

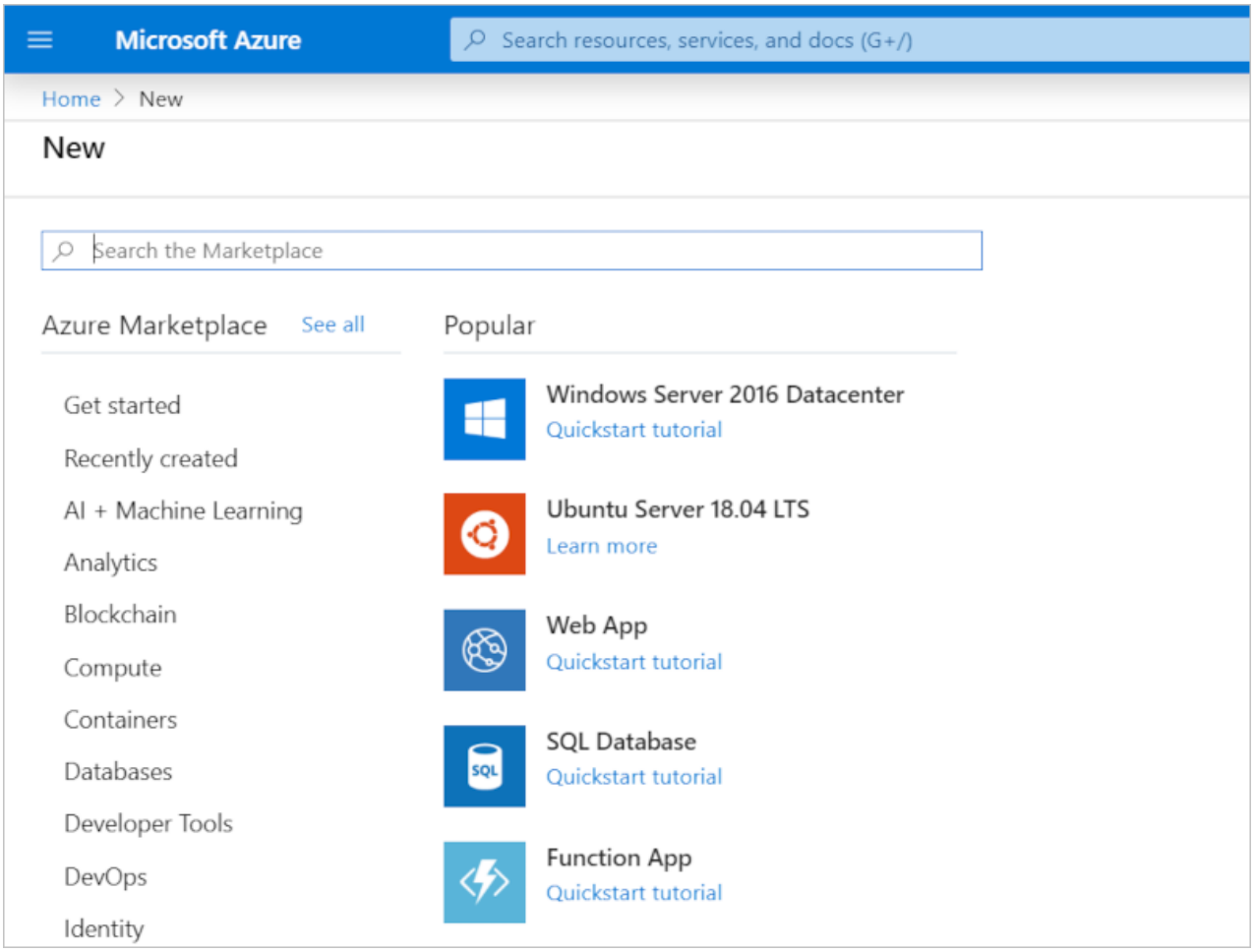
3/24/2020 • 2 minutes to read • Edit Online

Instead of creating and managing IoT Central applications on the Azure IoT Central application manager website, you can use the Azure portal to manage your applications.

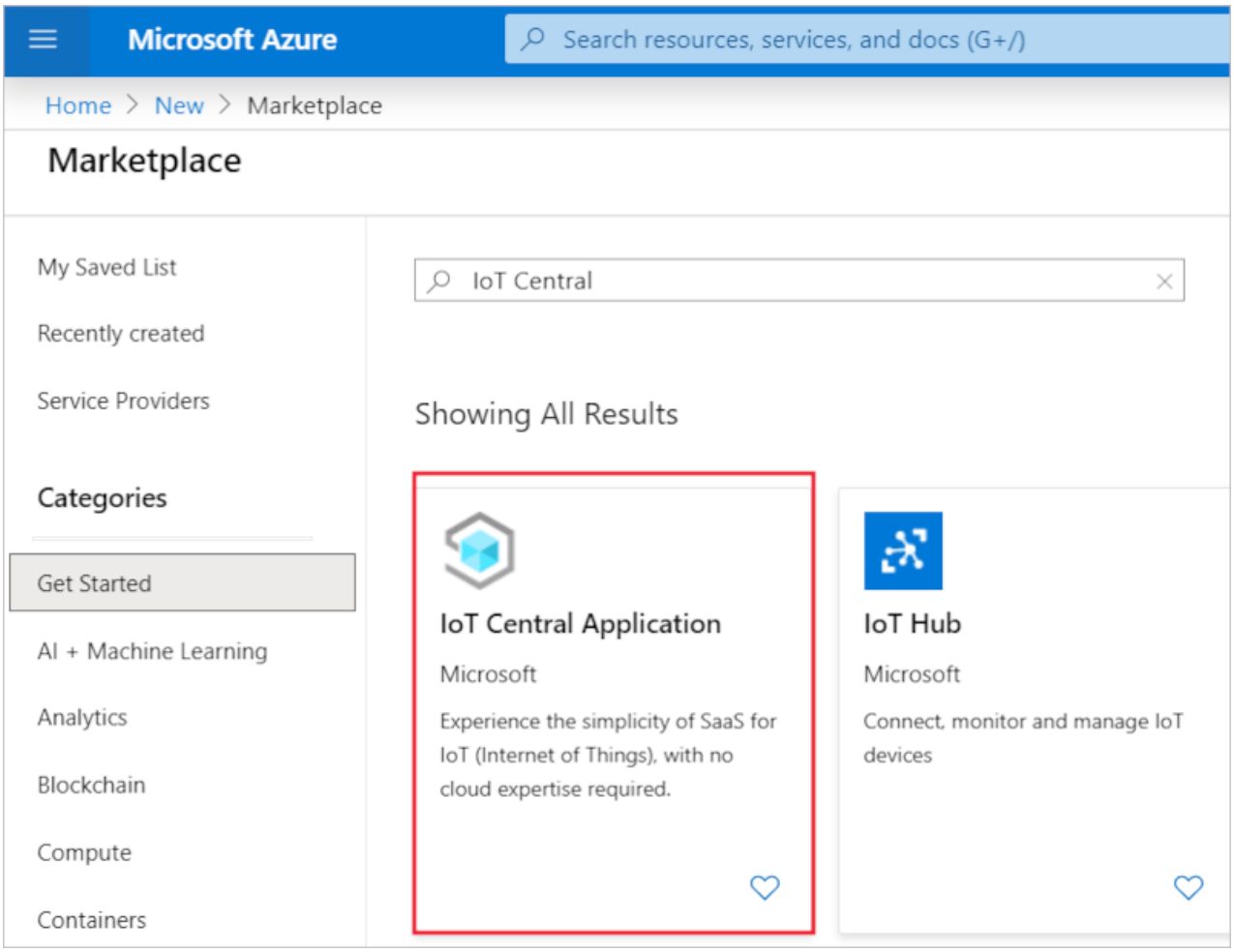
Create IoT Central applications

To create an application, navigate to the Azure portal and select **Create a resource**.

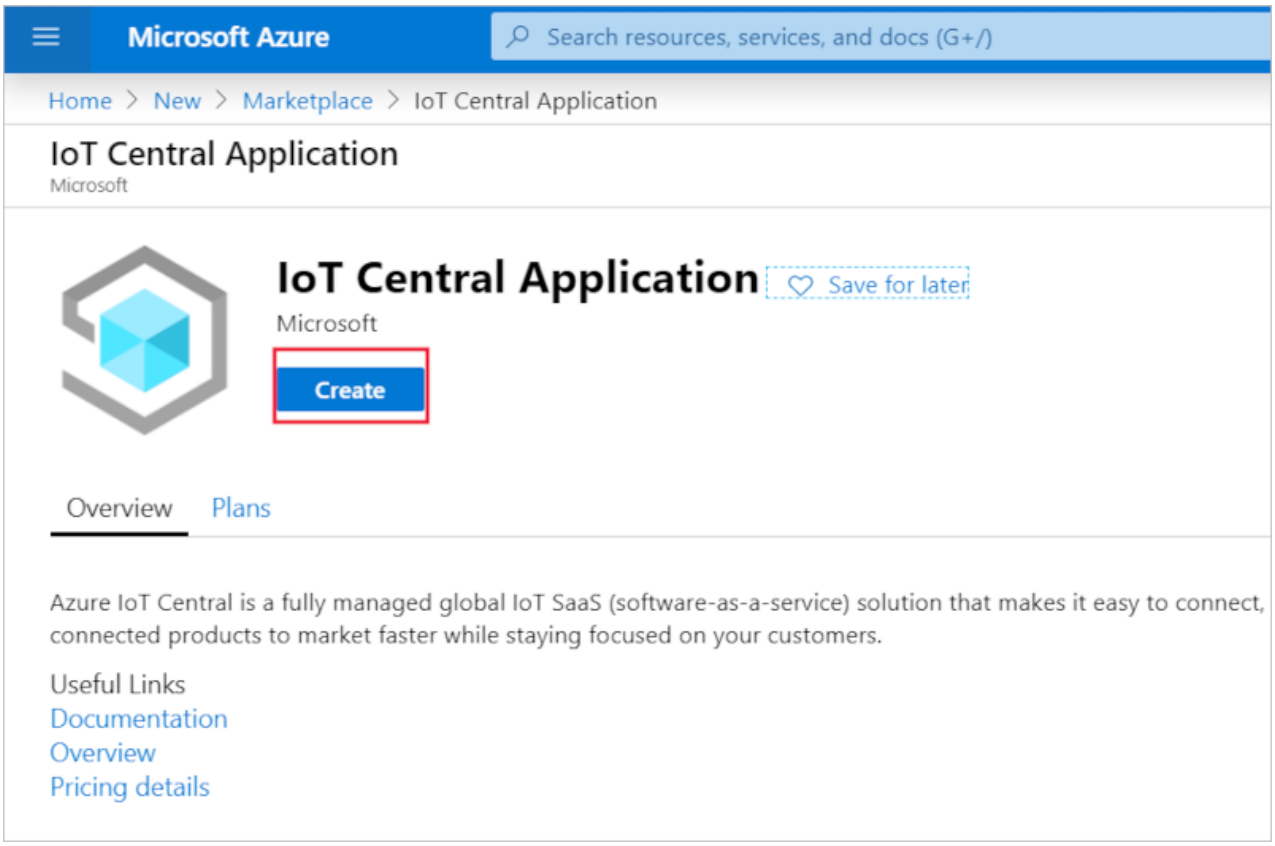
In **Search the Marketplace** bar, type *IoT Central*:



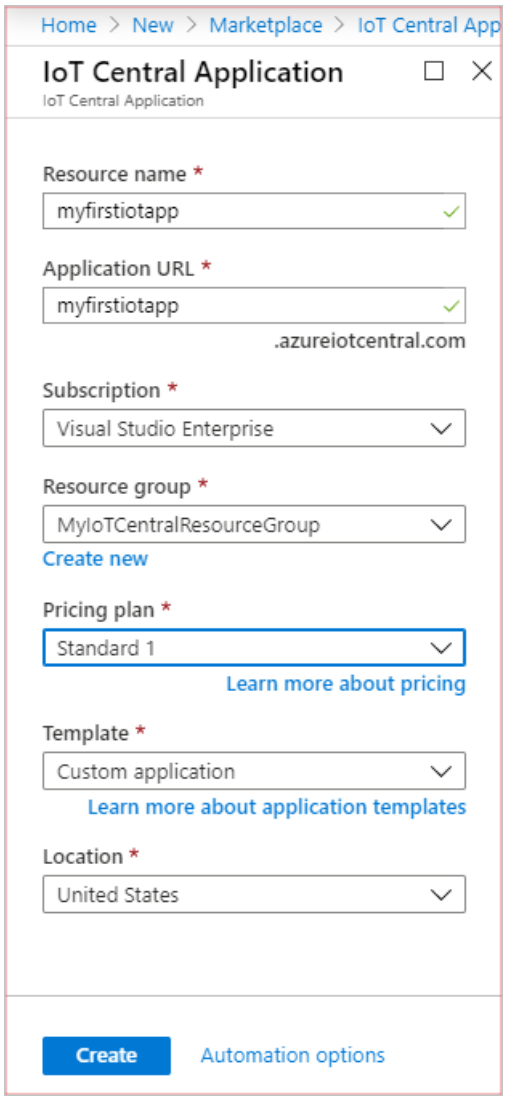
Select the **IoT Central Application** tile in the search results:



Now, select **Create**:



Fill in all the fields in the form. This form is similar to the form you fill out to create applications on the Azure IoT Central application manager website. For more information, see the Create an IoT Central application quickstart.



Location is the geography where you'd like to create your application. Typically, you should choose the location that's physically closest to your devices to get optimal performance. Azure IoT Central is currently available in the **Australia**, **Asia Pacific**, **Europe**, **United States**, **United Kingdom**, and **Japan** geographies. Once you choose a location, you can't move your application to a different location later.

After filling out all fields, select **Create**.

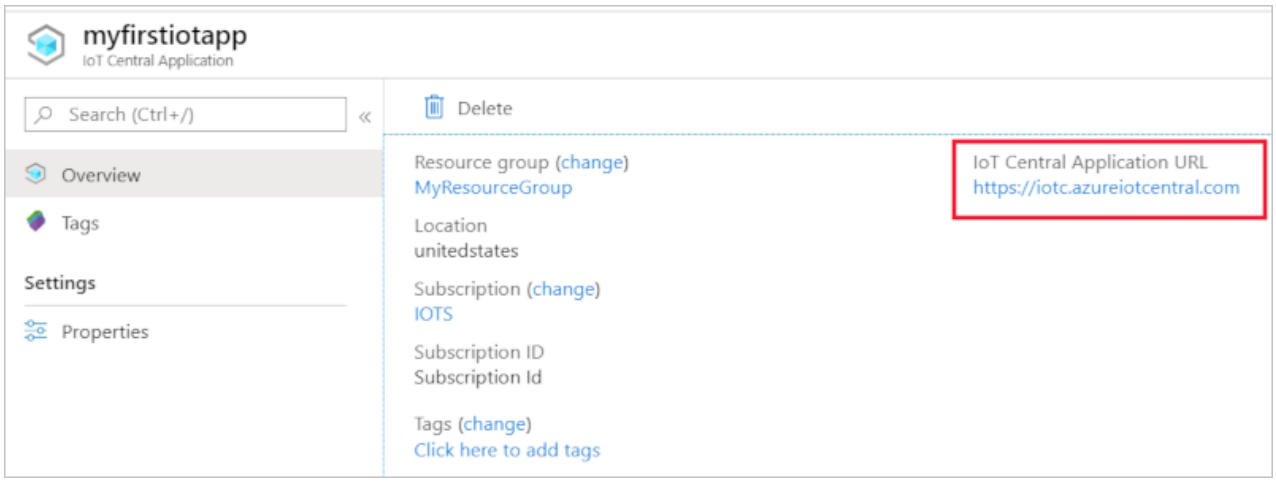
Manage existing IoT Central applications

If you already have an Azure IoT Central application you can delete it, or move it to a different subscription or resource group in the Azure portal.

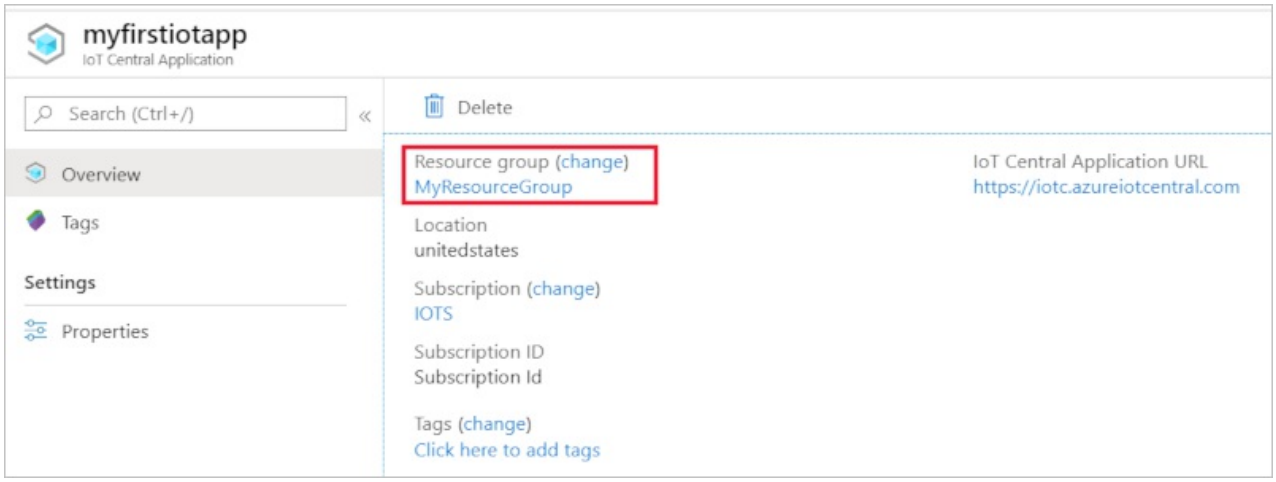
> **NOTE**
> You can't see applications created on the free pricing plan in the Azure portal because they are not associated with your subscription.

To get started, select **All resources** in the portal. Select **Show hidden types** and start typing the name of your application in **Filter by name** to find it. Then select the IoT Central application you'd like to manage.

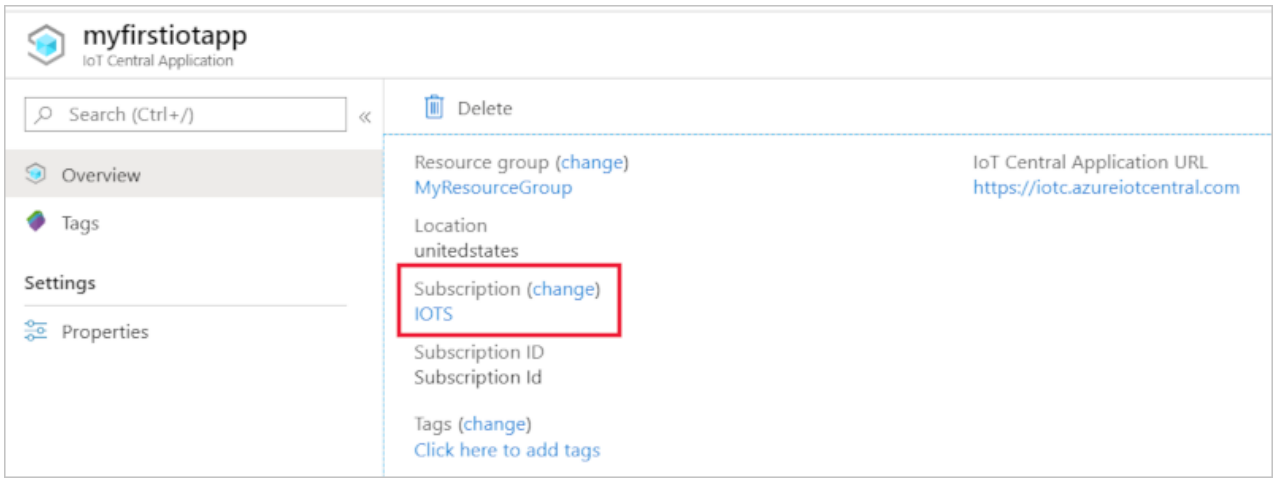
To navigate to the application, select the **IoT Central Application URL**:



To move the application to a different resource group, select **change** beside the resource group. On the **Move resources** page, choose the resource group you'd like to move this application to:



To move the application to a different subscription, select **change** beside the subscription. On the **Move resources** page, choose the subscription you'd like to move this application to:



Next steps

Now that you've learned how to manage Azure IoT Central applications from the Azure portal, here is the suggested next step:

Administer your application

Manage IoT Central from Azure CLI

3/27/2020 • 4 minutes to read • Edit Online

Instead of creating and managing IoT Central applications on the Azure IoT Central application manager website, you can use Azure CLI to manage your applications.

Prerequisites

If you don't have an Azure subscription, create a free account before you begin.

Use Azure Cloud Shell

Azure hosts Azure Cloud Shell, an interactive shell environment that you can use through your browser. You can use either Bash or PowerShell with Cloud Shell to work with Azure services. You can use the Cloud Shell preinstalled commands to run the code in this article without having to install anything on your local environment.

To start Azure Cloud Shell:

OPTION	EXAMPLE/LINK
Select **Try It** in the upper-right corner of a code block. Selecting **Try It** doesn't automatically copy the code to Cloud Shell.	
Go to https://shell.azure.com, or select the **Launch Cloud Shell** button to open Cloud Shell in your browser.	
Select the **Cloud Shell** button on the menu bar at the upper right in the Azure portal.	

To run the code in this article in Azure Cloud Shell:

1. Start Cloud Shell.
2. Select the **Copy** button on a code block to copy the code.
3. Paste the code into the Cloud Shell session by selecting **Ctrl+Shift+V** on Windows and Linux or by selecting **Cmd+Shift+V** on macOS.
4. Select **Enter** to run the code.

If you prefer to run Azure CLI on your local machine, see Install the Azure CLI. When you run Azure CLI locally, use the **az login** command to sign in to Azure before you try the commands in this article.

> **TIP**
> If you need to run your CLI commands in a different Azure subscription, see Change the active subscription.

Install the extension

The commands in this article are part of the **azure-iot** CLI extension. Run the following command to install the extension:

```
az extension add --name azure-iot
```

Create an application

Use the az iotcentral app create command to create an IoT Central application in your Azure subscription. For example:

```
# Create a resource group for the IoT Central application
az group create --location "East US" \
    --name "MyIoTCentralResourceGroup"
```

```
# Create an IoT Central application
az iotcentral app create \
  --resource-group "MyIoTCentralResourceGroup" \
  --name "myiotcentralapp" --subdomain "mysubdomain" \
  --sku ST1 --template "iotc-pnp-preview@1.0.0" \
  --display-name "My Custom Display Name"
```

These commands first create a resource group in the east US region for the application. The following table describes the parameters used with the **az iotcentral app create** command:

PARAMETER	DESCRIPTION
resource-group	The resource group that contains the application. This resource group must already exist in your subscription.
location	By default, this command uses the location from the resource group. Currently, you can create an IoT Central application in the **Australia**, **Asia Pacific**, **Europe**, **United States**, **United Kingdom**, and **Japan** geographies.
name	The name of the application in the Azure portal.
subdomain	The subdomain in the URL of the application. In the example, the application URL is `https://mysubdomain.azureiotcentral.com`.
sku	Currently, you can use either **ST1** or **ST2**. See Azure IoT Central pricing.
template	The application template to use. For more information, see the following table.
display-name	The name of the application as displayed in the UI.

Application templates

TEMPLATE	NAME	DESCRIPTION
iotc-pnp-preview@1.0.0	Custom application	Creates an empty application for you to populate with your own device templates and devices.

TEMPLATE	NAME	DESCRIPTION
iotc-default@1.0.0	Custom application (legacy)	Creates an empty legacy application for you to populate with your own device templates and devices.
iotc-condition@1.0.0	In-store Analytics – Condition Monitoring	Creates an application to connect and monitor a store environment.
iotc-consumption@1.0.0	Water Consumption Monitoring	Creates an application to monitor and control water flow.
iotc-distribution@1.0.0	Digital Distribution Center	Creates an application to improve warehouse output efficiency by digitizing key assets and actions.
iotc-inventory@1.0.0	Smart Inventory Management	Creates an application to automate receiving, product movement, cycle counting, and tracking.
iotc-logistics@1.0.0	Connected Logistics	Creates an application to track your shipments in real time across air, water, and land with location and condition monitoring.
iotc-meter@1.0.0	Smart Meter Analytics	Creates an application to monitor energy consumption, network status, and identify trends to improve customer support and smart meter management.
iotc-mfc@1.0.0	Micro-fulfillment Center	Creates an application to digitally connect and manage a fully automated fulfillment center.
iotc-patient@1.0.0	Continuous Patient Monitoring	Creates an application to extend patient care, reduce readmissions, and manage diseases.
iotc-power@1.0.0	Solar Power Monitoring	Creates an application to monitor solar panel status and energy generation trends.
iotc-quality@1.0.0	Water Quality Monitoring	Creates an application to digitally monitor water quality.
iotc-store@1.0.0	In-store Analytics – Checkout	Creates an application to monitor and manage the checkout flow inside your store.
iotc-waste@1.0.0	Connected Waste Management	Creates an application to monitor waste bins and dispatch field operators.

View your applications

Use the az iotcentral app list command to list your IoT Central applications and view metadata.

Modify an application

Use the az iotcentral app update command to update the metadata of an IoT Central application. For example, to change the display name of your application:

```
az iotcentral app update --name myiotcentralapp \
  --resource-group MyIoTCentralResourceGroup \
  --set displayName="My new display name"
```

Remove an application

Use the az iotcentral app delete command to delete an IoT Central application. For example:

```
az iotcentral app delete --name myiotcentralapp \
  --resource-group MyIoTCentralResourceGroup
```

Next steps

Now that you've learned how to manage Azure IoT Central applications from Azure CLI, here is the suggested next step:

Administer your application

Manage IoT Central from Azure PowerShell

3/27/2020 • 4 minutes to read • Edit Online

Instead of creating and managing IoT Central applications on the Azure IoT Central application manager website, you can use Azure PowerShell to manage your applications.

Prerequisites

If you don't have an Azure subscription, create a free account before you begin.

Use Azure Cloud Shell

Azure hosts Azure Cloud Shell, an interactive shell environment that you can use through your browser. You can use either Bash or PowerShell with Cloud Shell to work with Azure services. You can use the Cloud Shell preinstalled commands to run the code in this article without having to install anything on your local environment.

To start Azure Cloud Shell:

OPTION	EXAMPLE/LINK
Select **Try It** in the upper-right corner of a code block. Selecting **Try It** doesn't automatically copy the code to Cloud Shell.	
Go to https://shell.azure.com, or select the **Launch Cloud Shell** button to open Cloud Shell in your browser.	
Select the **Cloud Shell** button on the menu bar at the upper right in the Azure portal.	

To run the code in this article in Azure Cloud Shell:

1. Start Cloud Shell.
2. Select the **Copy** button on a code block to copy the code.
3. Paste the code into the Cloud Shell session by selecting **Ctrl+Shift+V** on Windows and Linux or by selecting **Cmd+Shift+V** on macOS.
4. Select **Enter** to run the code.

If you prefer to run Azure PowerShell on your local machine, see Install the Azure PowerShell module. When you run Azure PowerShell locally, use the **Connect-AzAccount** cmdlet to sign in to Azure before you try the cmdlets in this article.

> **TIP**
> If you need to run your PowerShell commands in a different Azure subscription, see Change the active subscription.

Install the IoT Central module

Run the following command to check the IoT Central module is installed in your PowerShell environment:

```
Get-InstalledModule -name Az.I*
```

If the list of installed modules doesn't include **Az.IotCentral**, run the following command:

```
Install-Module Az.IotCentral
```

Create an application

Use the New-AzIotCentralApp cmdlet to create an IoT Central application in your Azure subscription. For example:

```
# Create a resource group for the IoT Central application
New-AzResourceGroup -ResourceGroupName "MyIoTCentralResourceGroup" `
  -Location "East US"
```

```
# Create an IoT Central application
New-AzIotCentralApp -ResourceGroupName "MyIoTCentralResourceGroup" `
  -Name "myiotcentralapp" -Subdomain "mysubdomain" `
  -Sku "ST1" -Template "iotc-pnp-preview@1.0.0" `
  -DisplayName "My Custom Display Name"
```

The script first creates a resource group in the east US region for the application. The following table describes the parameters used with the **New-AzIotCentralApp** command:

PARAMETER	DESCRIPTION
ResourceGroupName	The resource group that contains the application. This resource group must already exist in your subscription.
Location	By default, this cmdlet uses the location from the resource group. Currently, you can create an IoT Central application in the **Australia**, **Asia Pacific**, **Europe**, **United States**, **United Kingdom**, and **Japan** geographies.
Name	The name of the application in the Azure portal.
Subdomain	The subdomain in the URL of the application. In the example, the application URL is `https://mysubdomain.azureiotcentral.com`.
Sku	Currently, you can use either **ST1** or **ST2**. See Azure IoT Central pricing.
Template	The application template to use. For more information, see the following table.
DisplayName	The name of the application as displayed in the UI.

Application templates

TEMPLATE	NAME	DESCRIPTION

TEMPLATE	NAME	DESCRIPTION
iotc-pnp-preview@1.0.0	Custom application	Creates an empty application for you to populate with your own device templates and devices.
iotc-default@1.0.0	Custom application (legacy)	Creates an empty legacy application for you to populate with your own device templates and devices.
iotc-condition@1.0.0	In-store Analytics – Condition Monitoring	Creates an application to connect and monitor a store environment.
iotc-consumption@1.0.0	Water Consumption Monitoring	Creates an application to monitor and control water flow.
iotc-distribution@1.0.0	Digital Distribution Center	Creates an application to improve warehouse output efficiency by digitizing key assets and actions.
iotc-inventory@1.0.0	Smart Inventory Management	Creates an application to automate receiving, product movement, cycle counting, and tracking.
iotc-logistics@1.0.0	Connected Logistics	Creates an application to track your shipments in real time across air, water, and land with location and condition monitoring.
iotc-meter@1.0.0	Smart Meter Analytics	Creates an application to monitor energy consumption, network status, and identify trends to improve customer support and smart meter management.
iotc-mfc@1.0.0	Micro-fulfillment Center	Creates an application to digitally connect and manage a fully automated fulfillment center.
iotc-patient@1.0.0	Continuous Patient Monitoring	Creates an application to extend patient care, reduce readmissions, and manage diseases.
iotc-power@1.0.0	Solar Power Monitoring	Creates an application to monitor solar panel status and energy generation trends.
iotc-quality@1.0.0	Water Quality Monitoring	Creates an application to digitally monitor water quality.
iotc-store@1.0.0	In-store Analytics – Checkout	Creates an application to monitor and manage the checkout flow inside your store.
iotc-waste@1.0.0	Connected Waste Management	Creates an application to monitor waste bins and dispatch field operators.

View your IoT Central applications

Use the Get-AzIotCentralApp cmdlet to list your IoT Central applications and view metadata.

Modify an application

Use the Set-AzIotCentralApp cmdlet to update the metadata of an IoT Central application. For example, to change the display name of your application:

```
Set-AzIotCentralApp -Name "myiotcentralapp" `
  -ResourceGroupName "MyIoTCentralResourceGroup" `
  -DisplayName "My new display name"
```

Remove an application

Use the Remove-AzIotCentralApp cmdlet to delete an IoT Central application. For example:

```
Remove-AzIotCentralApp -ResourceGroupName "MyIoTCentralResourceGroup" `
 -Name "myiotcentralapp"
```

Next steps

Now that you've learned how to manage Azure IoT Central applications from Azure PowerShell, here is the suggested next step:

Administer your application

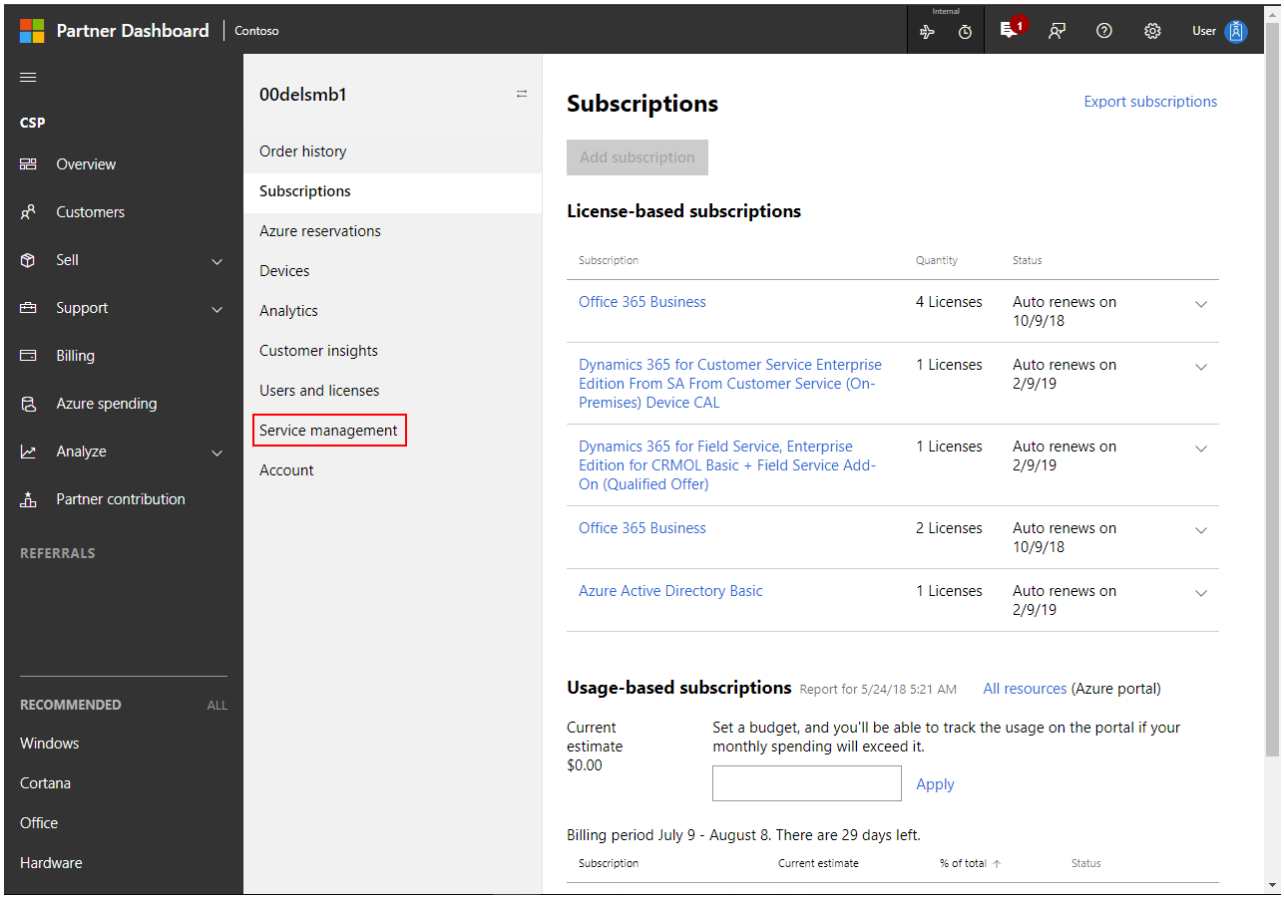
Create and manage an Azure IoT Central application from the CSP portal

4/8/2020 • 3 minutes to read • Edit Online

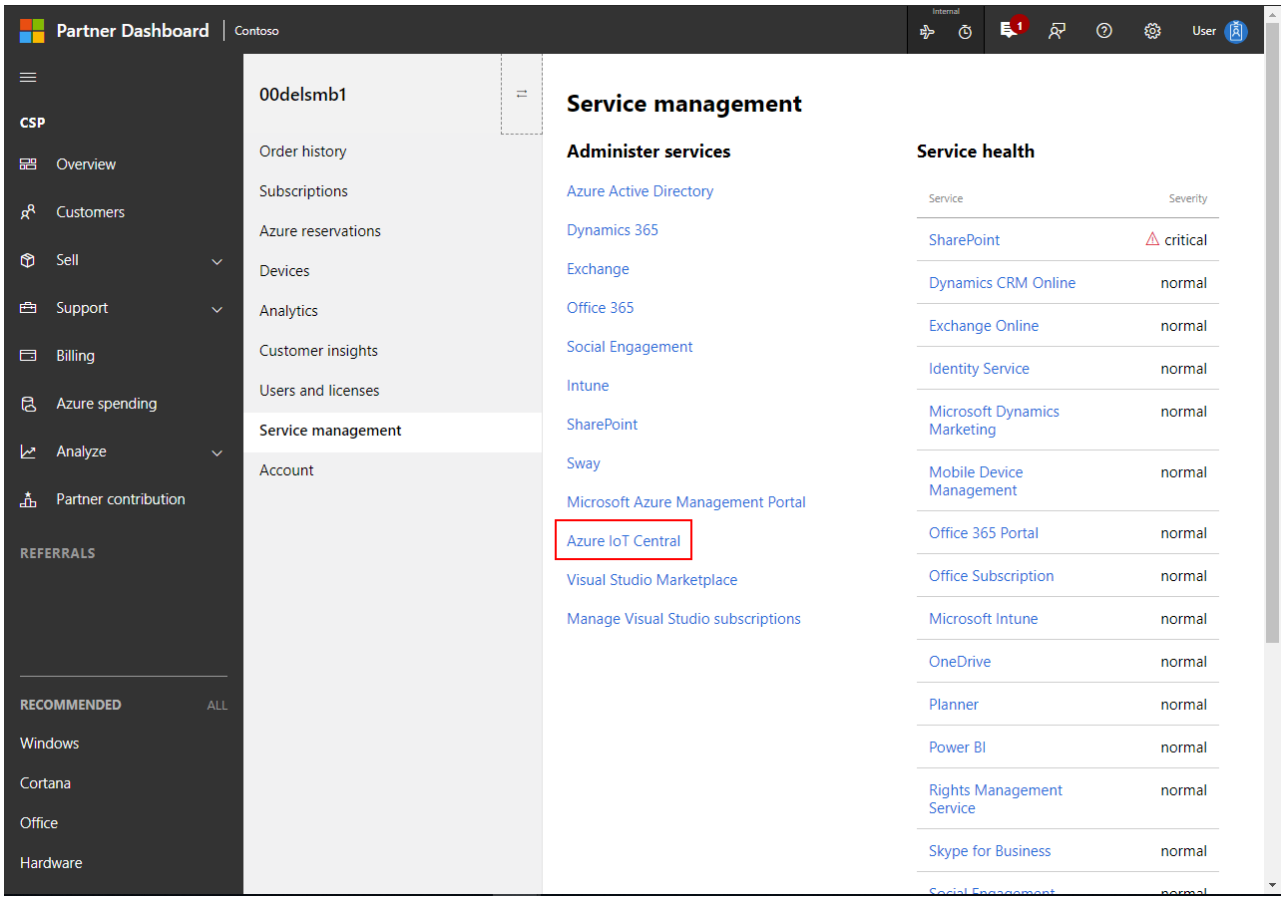
The Microsoft Cloud Solution Provider (CSP) program is a Microsoft Reseller program. Its intent is to provide our channel partners with a one-stop program to resell all Microsoft Commercial Online Services. Learn more about the Cloud Solution Provider program.

As a CSP, you can create and manage Microsoft Azure IoT Central applications on behalf of your customers through the Microsoft Partner Center. When Azure IoT Central applications are created on behalf of customers by CSPs, just like with other CSP managed Azure services, CSPs manage billing for customers. A charge for Azure IoT Central will appear in your total bill in the Microsoft Partner Center.

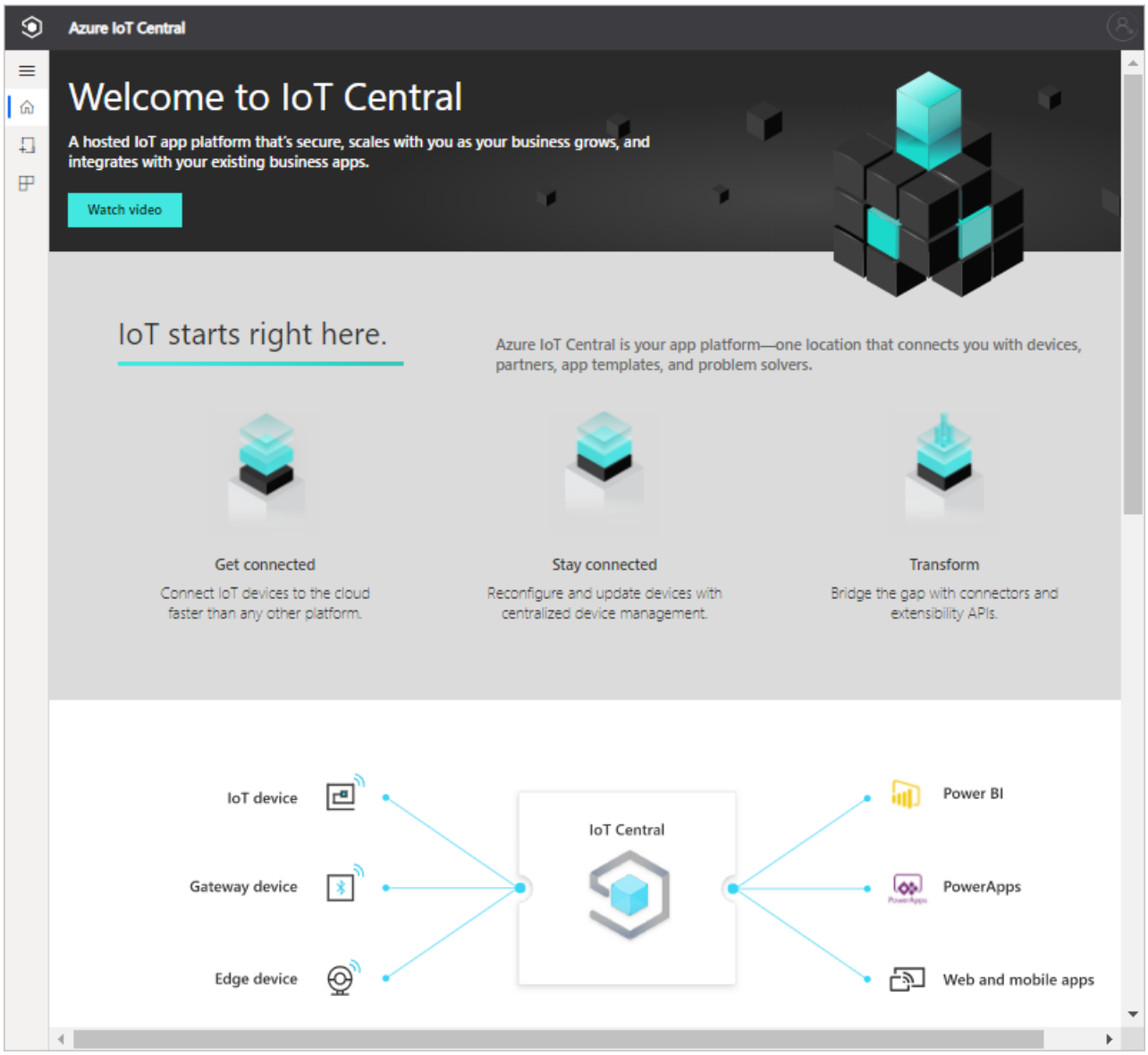
To get started, sign-in to your account on the Microsoft Partner Portal and select a customer for whom you want to create an Azure IoT Central application. Navigate to Service Management for the customer from the left nav.



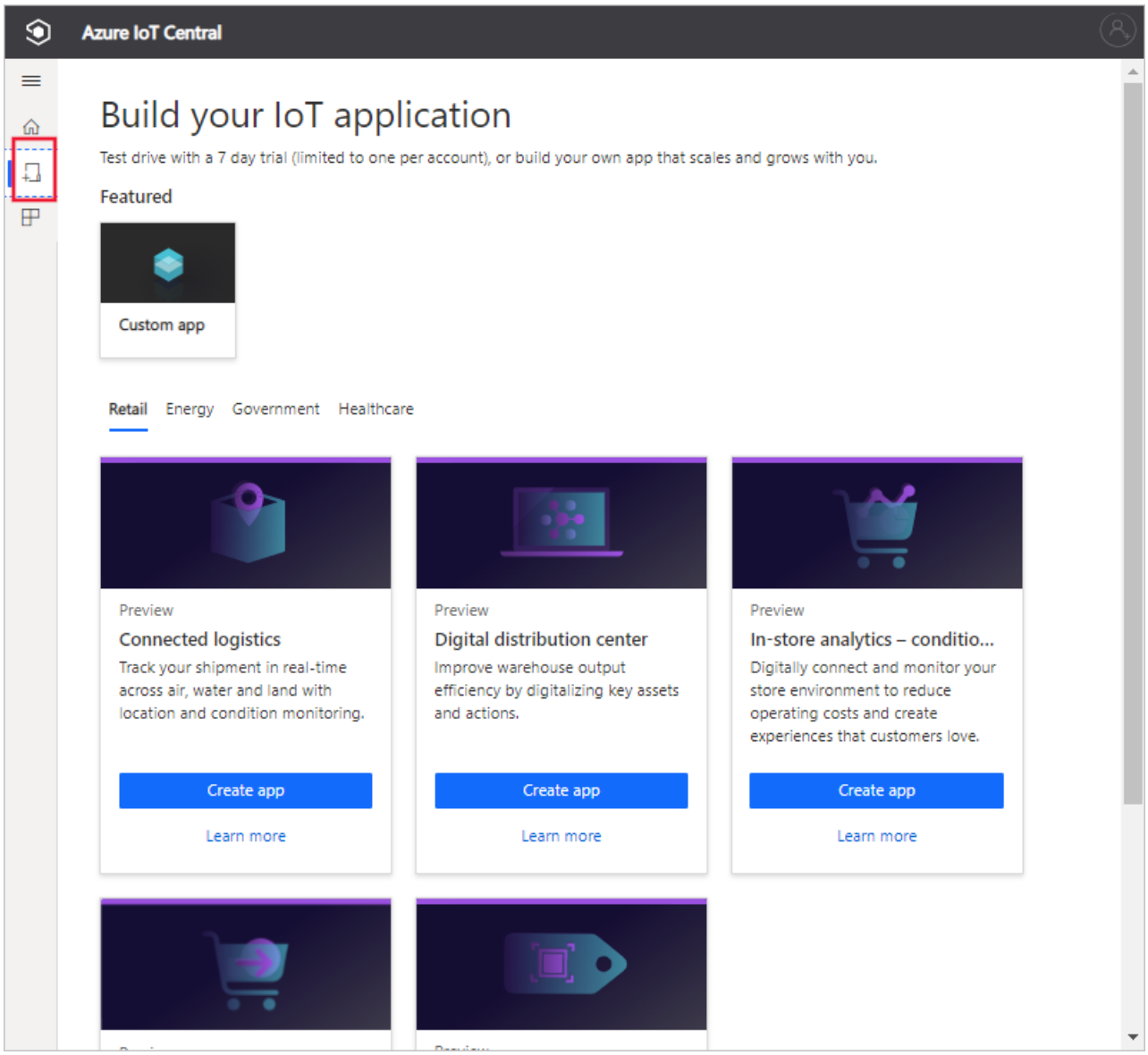
Azure IoT Central is listed as a service available to administer. Select the Azure IoT Central link on the page to create new applications or manage existing applications for this customer.



You land on the Azure IoT Central Application Manager page. Azure IoT Central keeps context that you came from the Microsoft Partner Center and that you came to manage that particular customer. You see this acknowledged in the header of the Application Manager page. From here, you can either navigate to an existing application you had created earlier for this customer to manage or create a new application for the customer.



To create an Azure IoT Central application, select **Build** in the left menu. Choose one of the industry templates, or choose **Custom app** to create an application from scratch. This will load the Application Creation page. You must complete all the fields on this page and then choose **Create**. You find more information about each of the fields below.


Azure IoT Central
Build your IoT application
Test drive with a 7 day trial (limited to one per account), or build your own app that scales and grows with you.
Featured
Custom app
Retail
Energy
Government
Healthcare
Preview
Connected logistics
Track your shipment in real-time across air, water and land with location and condition monitoring.
Create app
Learn more
Preview
Digital distribution center
Improve warehouse output efficiency by digitalizing key assets and actions.
Create app
Learn more
Preview
In-store analytics – conditio...
Digitally connect and monitor your store environment to reduce operating costs and create experiences that customers love.
Create app
Learn more

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 * ⓘ

Custom application

URL * ⓘ

custom-application .azureiotcentral-ppe.com

Application template * ⓘ

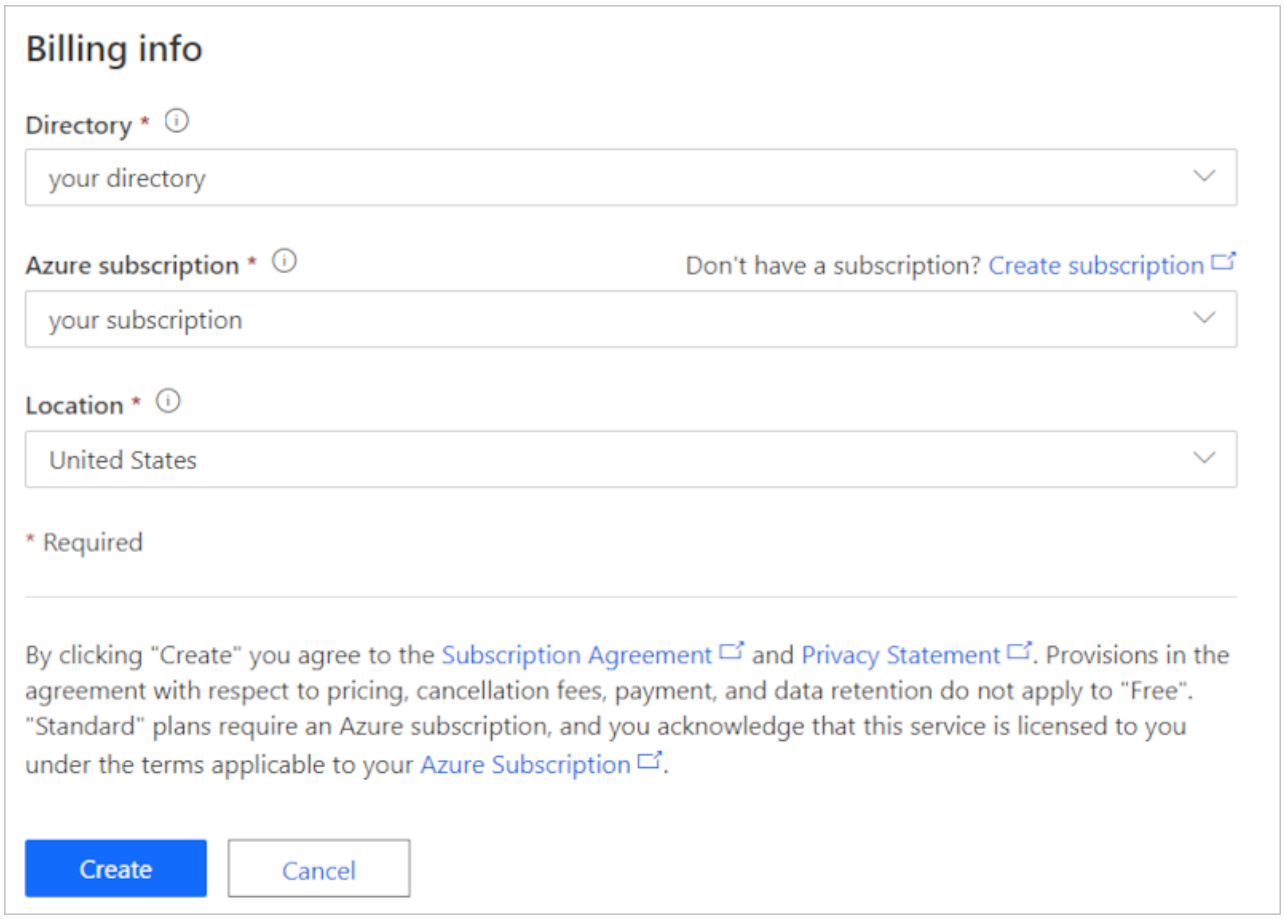
Custom application (legacy)

Pricing plan *

- ○ **Free**
  Try for **7 days** with no commitment
  5 free devices

- ○ **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



Pricing plan

You can only create applications that use a standard pricing plan as a CSP. To showcase Azure IoT Central to your customer, you can create an application that uses the free pricing plan separately. Learn more about the free and standard pricing plans on the Azure IoT Central pricing page.

You can only create applications that use a standard pricing plan as a CSP. To showcase Azure IoT Central to your customer, you can create an application that uses the free pricing plan separately. Learn more about the free and standard pricing plans on the Azure IoT Central pricing page.

Application name

The name of your application is displayed on the **Application Manager** page and within each Azure IoT Central application. You can choose any name for your Azure IoT Central application. Choose a name that makes sense to you and to others in your organization.

Application URL

The application URL is the link to your application. You can save a bookmark to it in your browser or share it with others.

When you enter the name for your application, your application URL is autogenerated. If you prefer, you can choose a different URL for your application. Each Azure IoT Central URL must be unique within Azure IoT Central. You see an error message if the URL you choose has already been taken.

Directory

Since Azure IoT Central has context that you came to manage the customer you selected in the Microsoft Partner Portal, you see just the Azure Active Directory tenant for that customer in the Directory field.

An Azure Active Directory tenant contains user identities, credentials, and other organizational information. Multiple Azure subscriptions can be associated with a single Azure Active Directory tenant.

To learn more, see Azure Active Directory.

Azure subscription

An Azure subscription enables you to create instances of Azure services. Azure IoT Central automatically finds all Azure Subscriptions of the customer to which you have access, and displays them in a dropdown on the **Create Application** page. Choose an Azure subscription to create a new Azure IoT Central Application.

If you don't have an Azure subscription, you can create one in the Microsoft Partner Center. After you create the Azure subscription, navigate back to the **Create Application** page. Your new subscription appears in the **Azure Subscription** drop-down.

To learn more, see Azure subscriptions.

Location

Location is the geography where you'd like to create the application. Typically, you should choose the location that's physically closest to your devices to get optimal performance. Currently, you can create an IoT Central application in the **Australia**, **Asia Pacific**, **Europe**, **United States**, **United Kingdom**, and **Japan** geographies. Once you choose a location, you can't later move your application to a different location.

Application template

Choose the application template you want to use for your application.

Next steps

Now that you have learned how to create an Azure IoT Central application as a CSP, here is the suggested next step:

Administer your application

Create and manage multiple dashboards

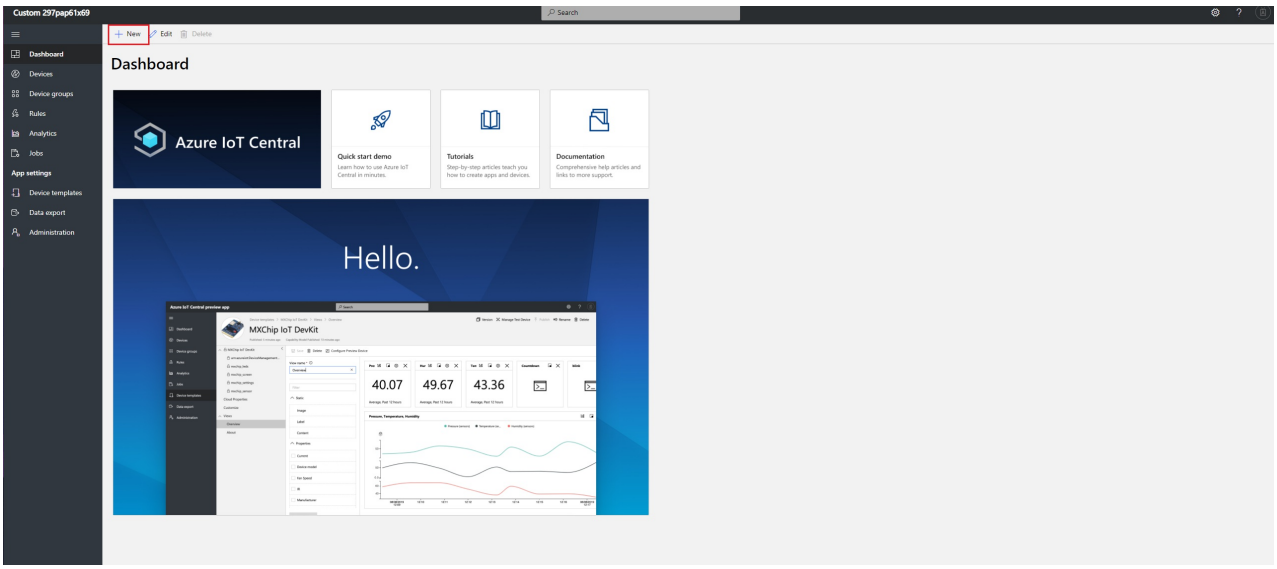
4/8/2020 • 2 minutes to read • Edit Online

The **Dashboard** is the page that loads when you first navigate to your application. An **builder** in your application defines the default application dashboard for all users. You can additionally create your own, personalized application dashboard. You can have several dashboards that display different data and switch between them.

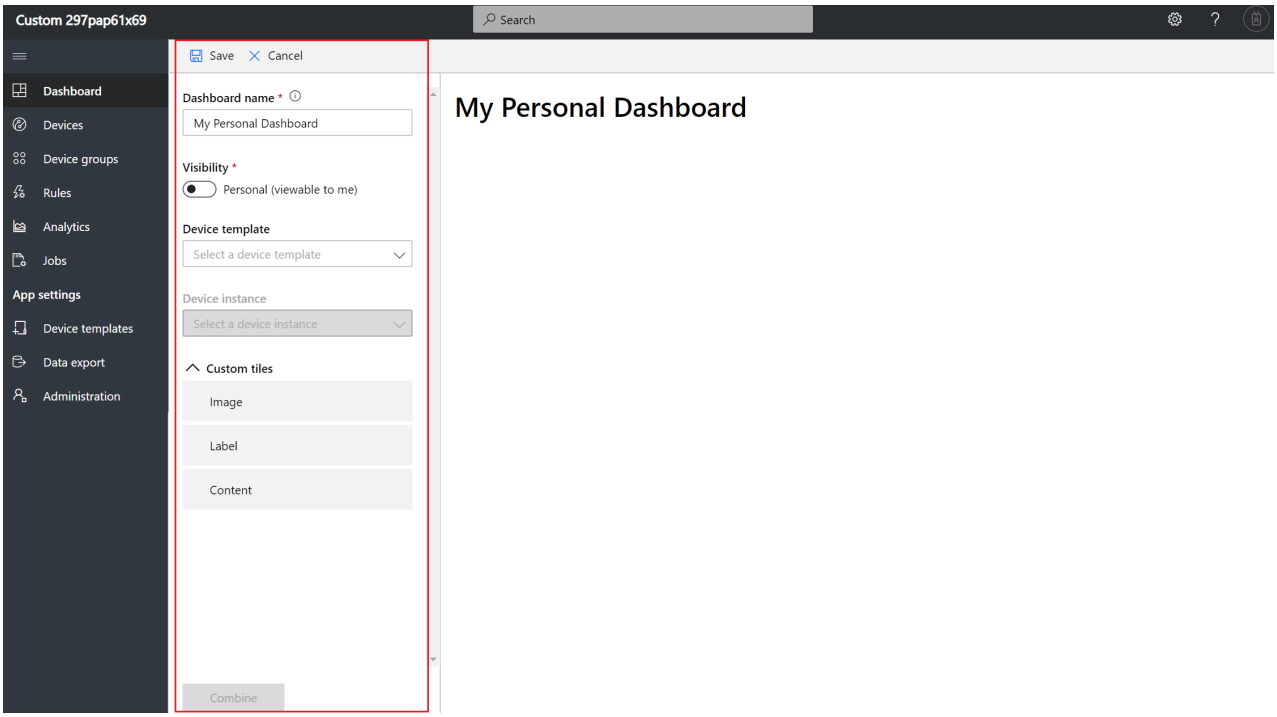
If you are an **admin** of the application, you also can create up to 10 application level dashboards to share with other users of the application. Only **admins** have the ability to create, edit, and delete application level dashboards.

Create dashboard

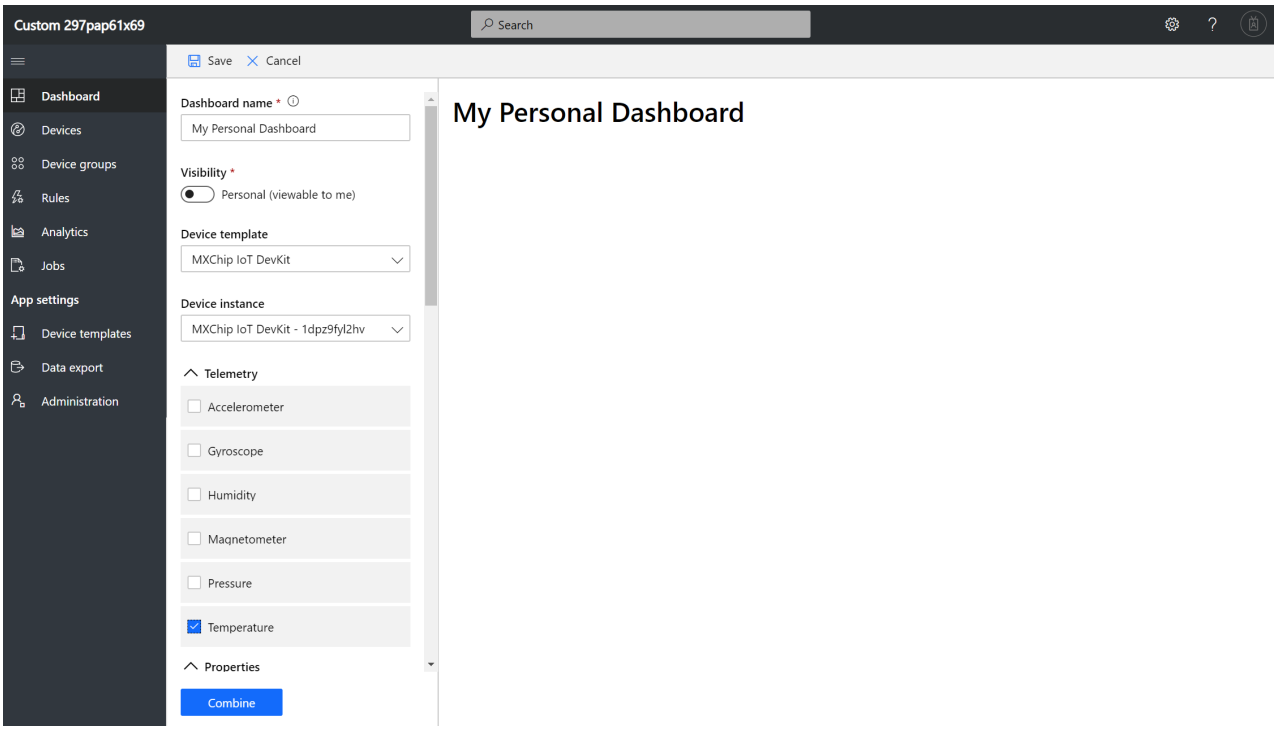
The following screenshot shows the dashboard in an application created from the **Custom Application** template. You can replace the default application dashboard with a personal dashboard, or if you are an admin, another application level dashboard. To do so, select **+ New** at the top left of the page.



Selecting **+ New** opens the dashboard editor. In the editor, you can give your dashboard a name and chose items from the library. The library contains the tiles and dashboard primitives you can use to customize the dashboard.

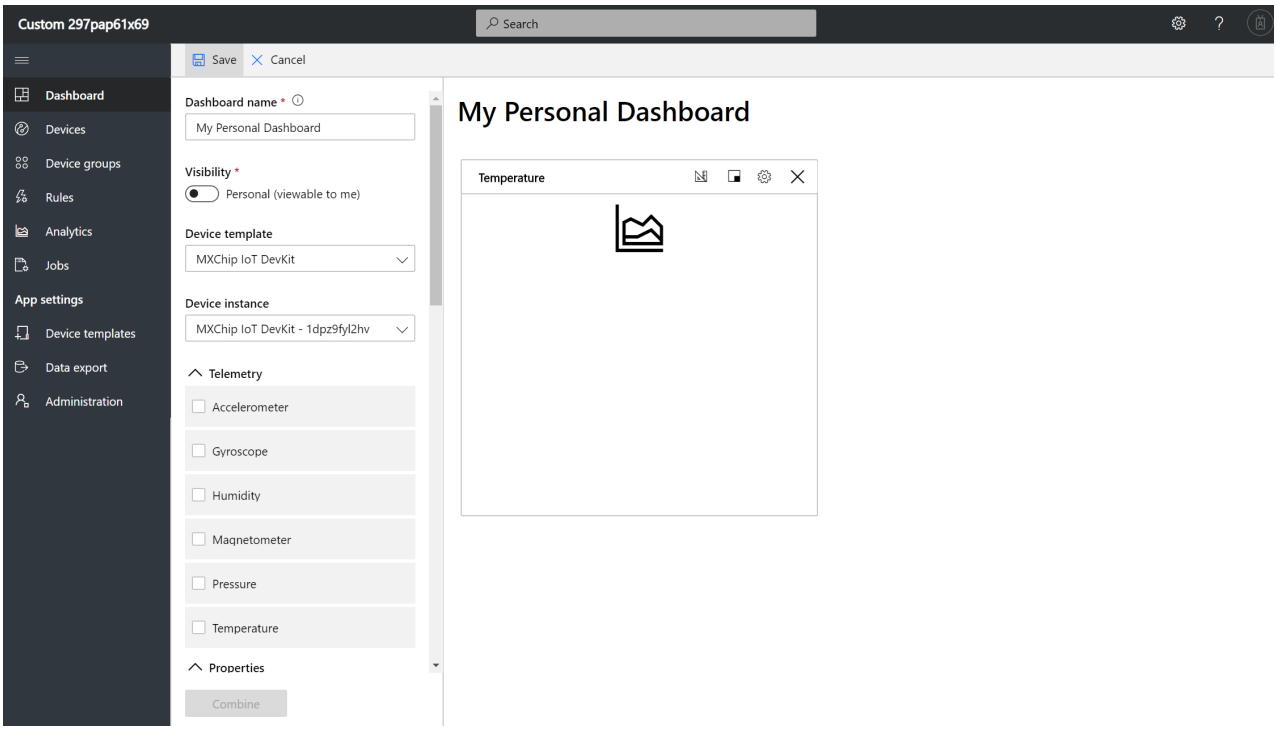


If you are an **admin** of the application, you will be given the option to toggle if you want to create a personal level dashboard or an application level dashboard. If you create a personal level dashboard, only you will be able to see it. If you create an application level dashboard, every user of the application will be able to see it. After entering a title and selecting the type of dashboard you want to create, you can save and add tiles later. Or if you are ready now and have added a device template and device instance, you can go ahead and create your first tile.

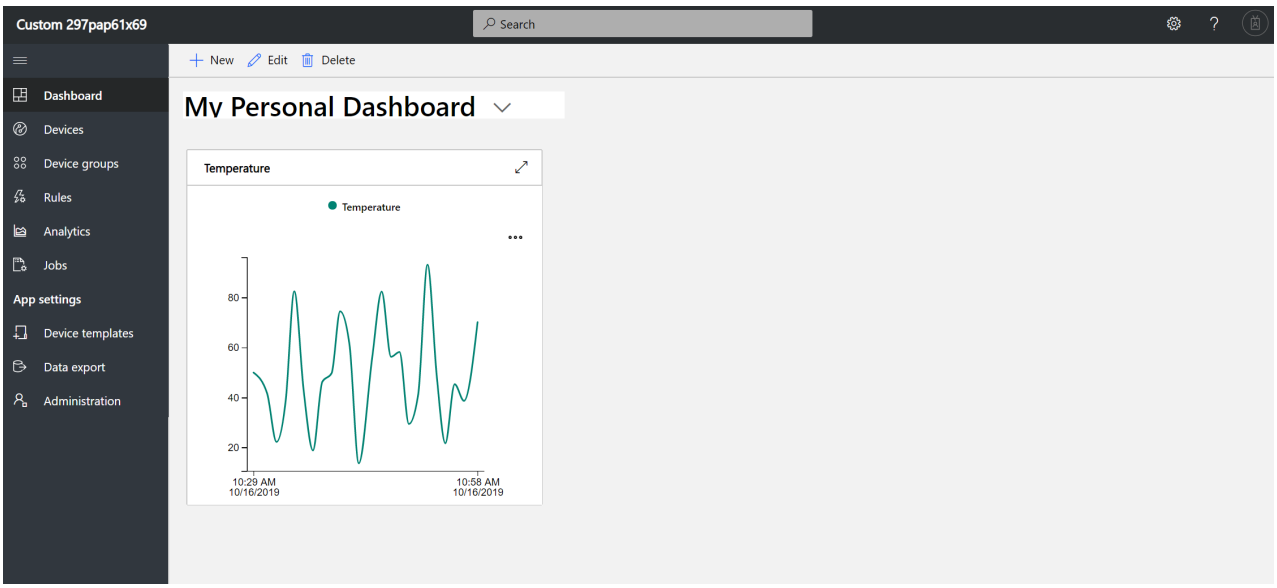


For example, you can add a **Telemetry** tile for the current temperature of the device. To do so:

1. Select a **Device Template**
2. Select a **Device Instance** for the device you want to see on a dashboard tile. Then you will see a list of the device's properties that can be used on the tile.
3. To create the tile on the dashboard, click on **Temperature** and drag it to the dashboard area. You can also click the checkbox next to **Temperature** and click **Combine**. The following screenshot shows selecting a Device Template and Device Instance then creating a Temperature Telemetry tile on the dashboard.
4. Select **Save** in the top left to save the tile to the dashboard.



Now when you view your personal dashboard, you see the new tile with the **Temperature** setting for the device:

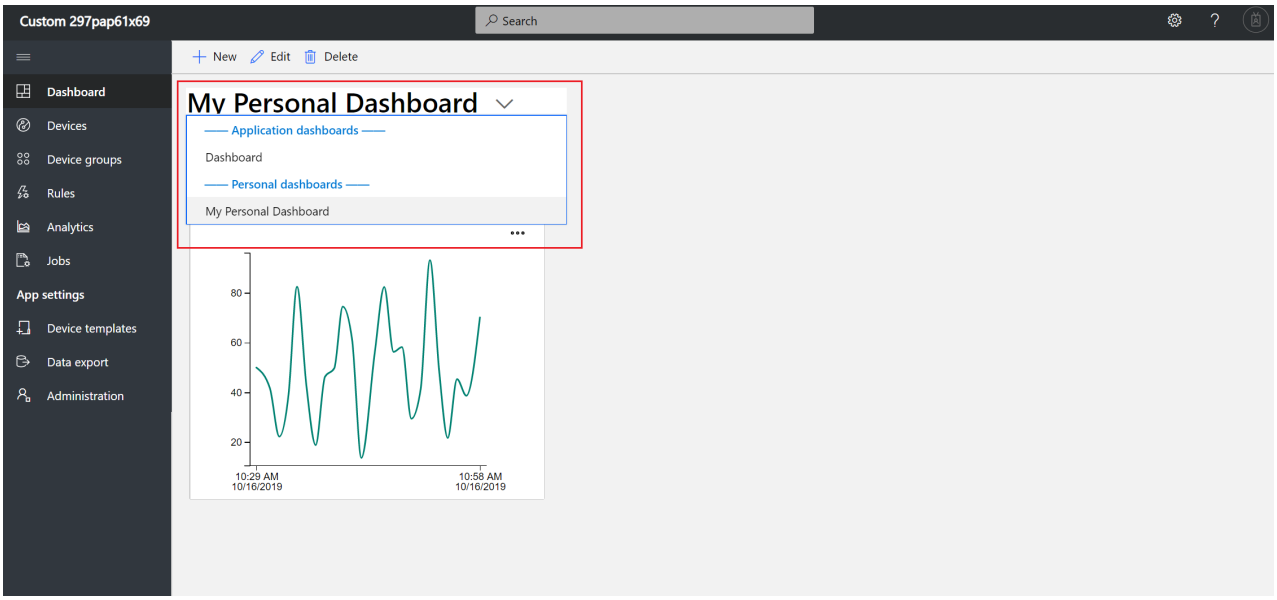


You can explore other tile types in the library to discover how to further customize your personal dashboards.

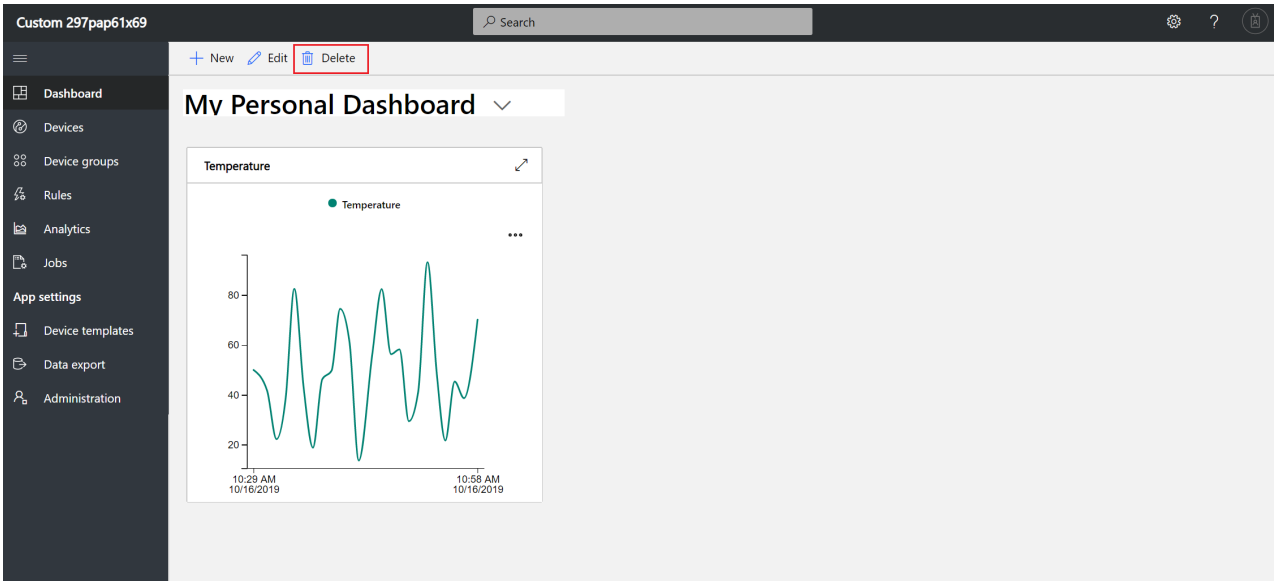
To learn more about how to use tiles in Azure IoT Central, see Add Tiles to your Dashboard.

Manage dashboards

You can have several personal dashboards and switch between them or choose from one of the default application dashboards:



You can edit your personal dashboards and delete any dashboards you no longer need. If you are an **admin**, you also have the ability to edit or delete application level dashboards as well.



Next steps

Now that you've learned how to create and manage personal dashboards, you can Learn how to manage your application preferences

Manage your personal application preferences

3/24/2020 • 2 minutes to read • Edit Online

This article applies to operators, builders, and administrators.

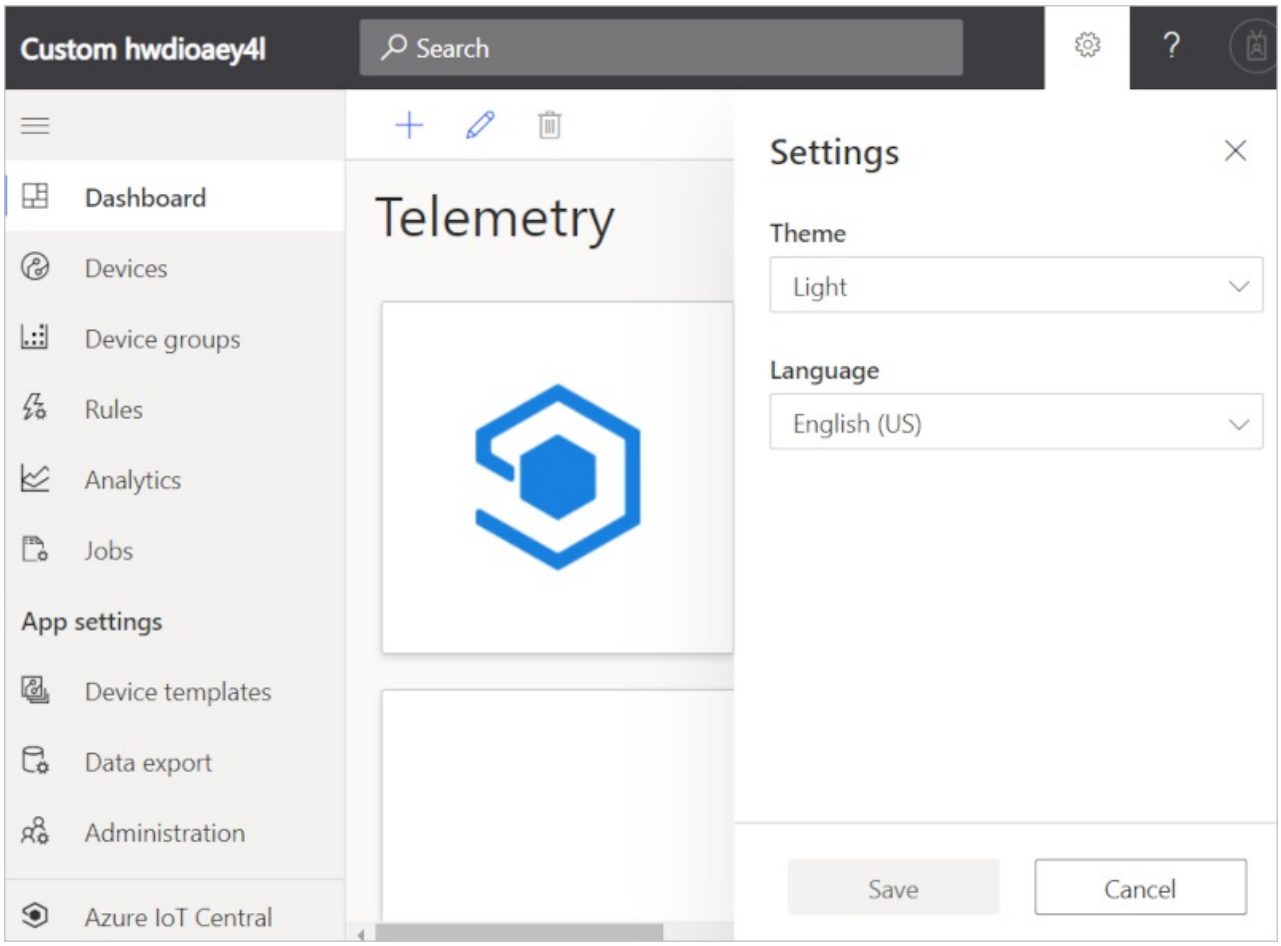
IoT Central provides the flexibility to customize your applications to fit your need. We also provide some flexibility on a per-user basis to customize your own view. This article describes the various customization options that a user can apply to their profile.

Changing language

IoT Central is supported in multiple languages. You can switch your preferred language by using the **language picker** on the settings icon on the top navigation bar. Once you've changed your language, IoT Central remembers your selection and applies it across all your applications. Customization within the application such dashboard images aren't localized.

Changing theme

We have support for both dark theme and light theme. While the light theme is the default, you can change the theme by selecting the settings icon on the top navigation bar.



NOTE

The option to choose between light and dark themes isn't available if your administrator has configured a custom theme for the application.

Next steps

Now that you've learned how to manage your profile in Azure IoT Central, here is the suggested next step:

Toggle live chat

Toggle live chat

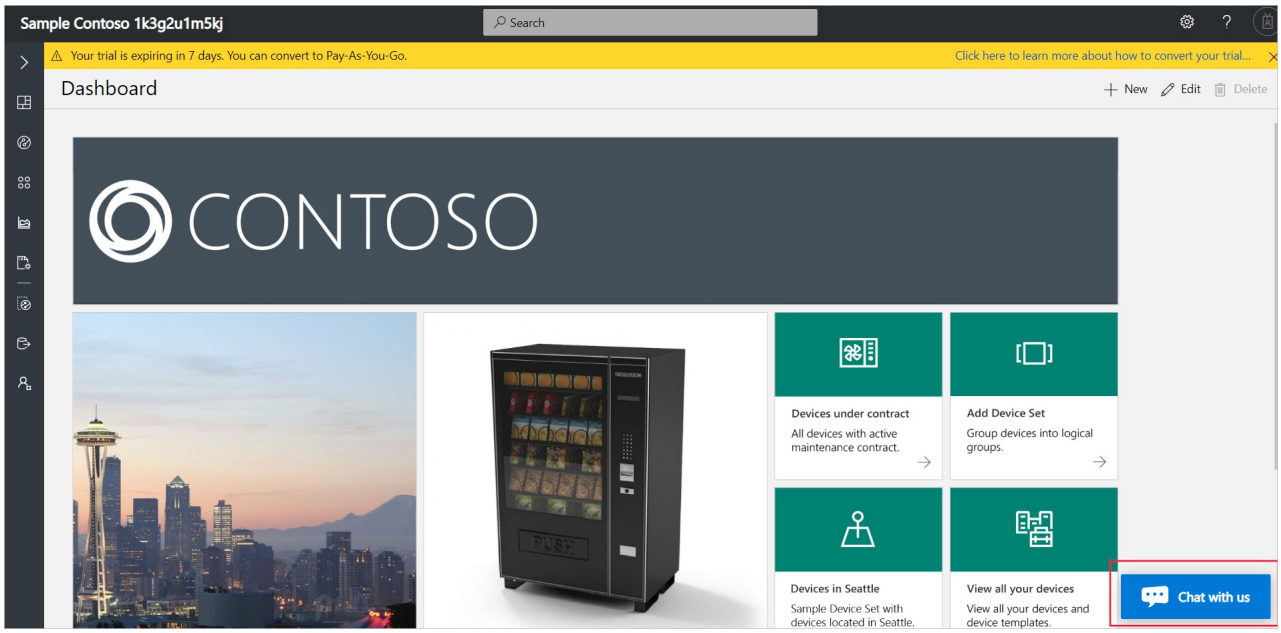
3/24/2020 • 2 minutes to read • Edit Online

This how-to article shows you how to toggle the live chat in your IoT Central application. You can use live chat to access technical support.

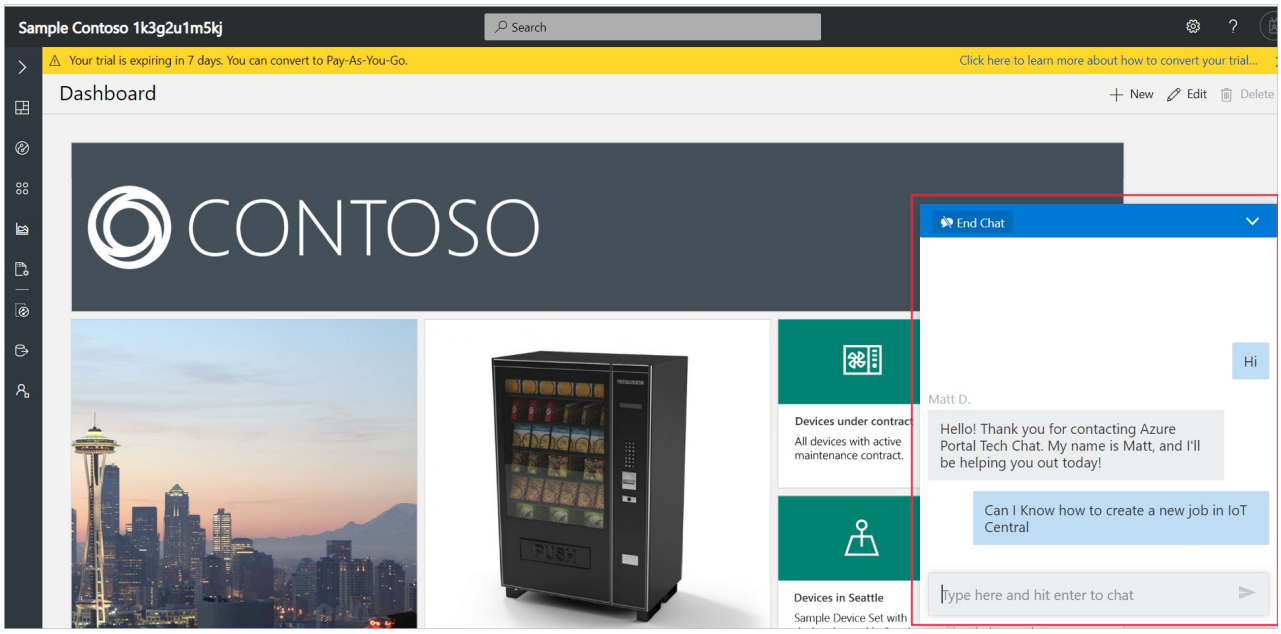
> **NOTE**
> The chat option is available only for applications created using the free pricing plan.

Chat with us

To get technical support, open your IoT Central application and select **Chat with us**.

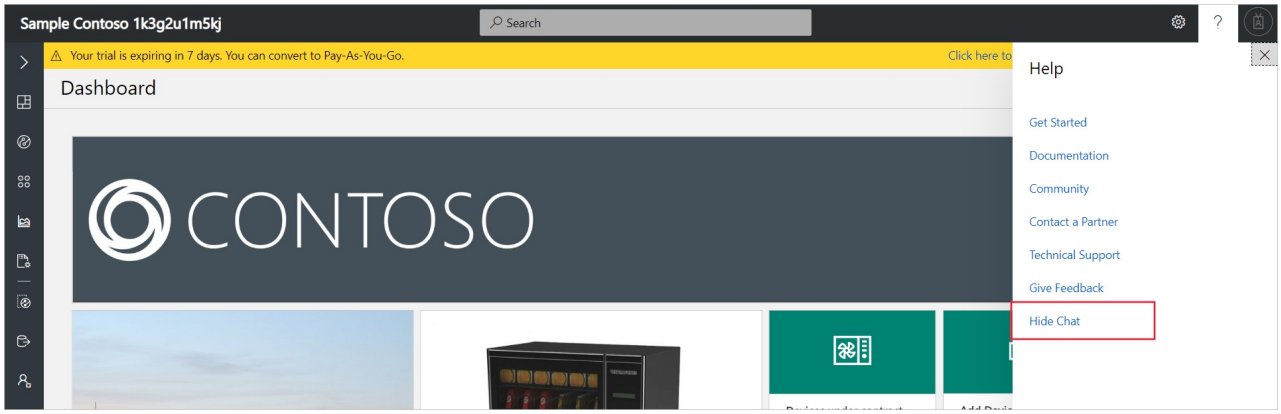


You can type a question as shown in the following screenshot:



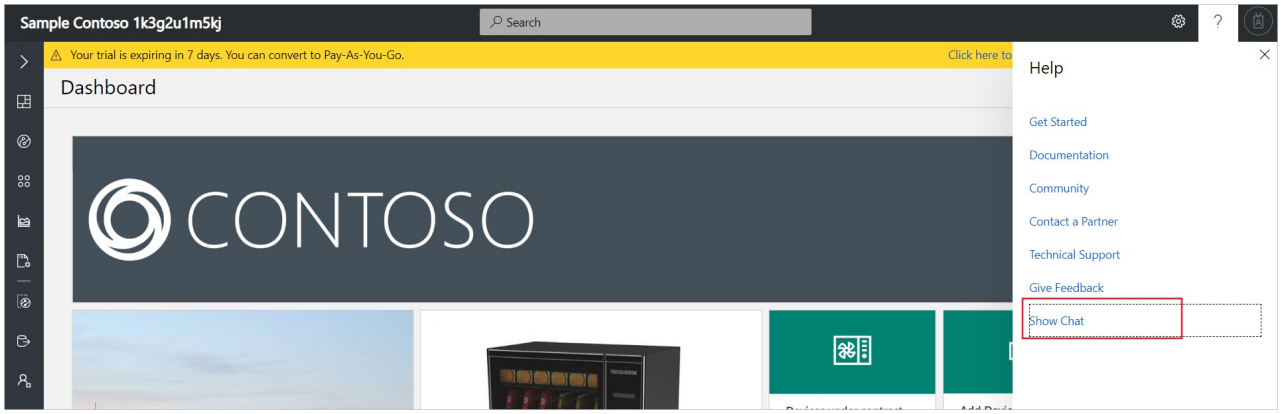
Hide chat

To hide the chat, choose **Hide Chat** in the **Help** panel:



Enable chat

To show the chat, choose **Show Chat** in the **Help** panel:



Next steps

Now that you've learned how to toggle live chat in Azure IoT Central, here is the suggested next step:

Add tiles to your dashboard

Summary of customer data request features

2/4/2020 • 2 minutes to read • Edit Online

Azure IoT Central is a fully managed Internet of Things (IoT) software-as-a-service solution that makes it easy to connect, monitor, and manage your IoT assets at scale, create deep insights from your IoT data, and take informed action.

> **NOTE**
>
> This article provides steps for how to delete personal data from the device or service and can be used to support your obligations under the GDPR. If you're looking for general info about GDPR, see the GDPR section of the Service Trust portal.

Identifying customer data

Azure Active Directory Object-IDs are used to identify users and assign roles. The Azure IoT Central portal displays user email addresses for role assignments but only the Azure Active Directory Object-ID is stored, the email address is dynamically queried from Azure Active Directory. Azure IoT Central administrators can view, export, and delete application users in the user administration section of an Azure IoT Central application.

Within the application, email addresses can be configured to receive alerts. In this case, email addresses are stored within IoT Central and must be managed from the in-app account administration page.

Regarding devices, Microsoft maintains no information and has no access to data that enables device to user correlation. Many of the devices managed in Azure IoT Central are not personal devices, for example a vending machine or coffee maker. Customers may, however, consider some devices to be personally identifiable and at their discretion may maintain their own asset or inventory tracking systems that tie devices to individuals. Azure IoT Central manages and stores all data associated with devices as if it were personal data.

When you use Microsoft enterprise services, Microsoft generates some information, known as system-generated logs. These logs constitute factual actions conducted within the service and diagnostic data related to individual devices, and are not related to user activity. Azure IoT Central system-generated logs are not accessible or exportable by application administrators.

Deleting customer data

The ability to delete user data is only provided through the IoT Central administration page. Application administrators can select the user to be deleted and select **Delete** in the upper right corner of the application to delete the record. Application administrators can also remove individual accounts that are no longer associated with the application in question.

After a user is deleted, no further alerts are emailed to them. However, their email address must be individually removed from each configured alert.

Exporting customer data

The ability to export data is only provided through the IoT Central administration page. Customer data, including assigned roles, can be selected, copied, and pasted by an application administrator.

Links to additional documentation

For more information about account administration, including role definitions, see How to administer your

application.

Supported browsers for Azure IoT Central

10/27/2019 • 2 minutes to read • Edit Online

This article applies to operators, builders, and administrators.

Azure IoT Central can be accessed across most modern desktops, tablets, and browsers. The following article outlines the list of supported browsers and required connectivity.

Supported browsers

We recommend that you use the most up-to-date browser that's compatible with your operating system. The following browsers are supported:

- Microsoft Edge (latest version)
- Safari (latest version, Mac only)
- Chrome (latest version)
- Firefox (latest version)

Required protocols

Azure IoT Central requires that your network supports both the HTTPS and WebSocket protocols for outbound connectivity.