Google

Lesson 3

● ● ● ● ● ●
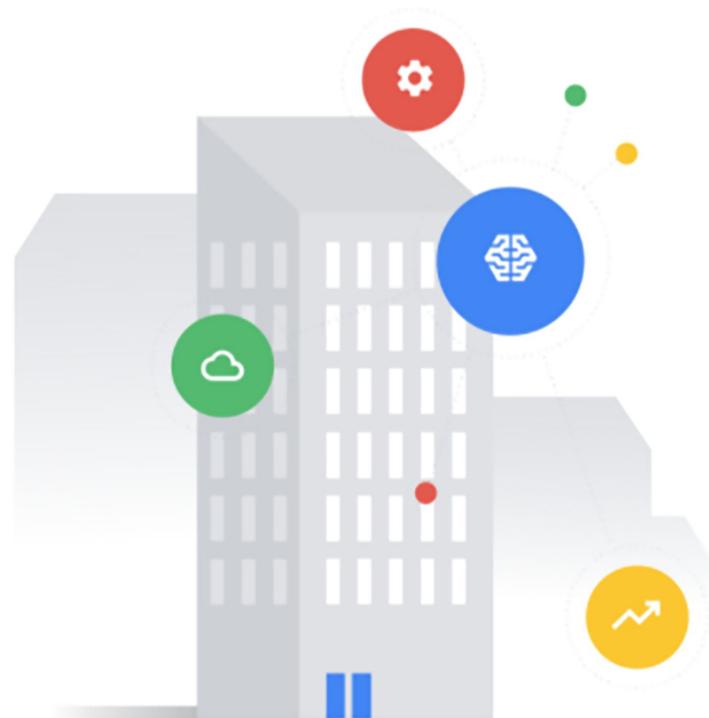
UDMI site model and device-to-cloud basics

Before you get started

This onboarding deck has interactive features and activities that enable a self-guided learning experience.
To help you get started, here are two tips for viewing and navigating through the deck.

① **View this deck in presentation mode.**

- To enter presentation mode, you can either:
  - Click the **Present** or **Slideshow** button in the top-right corner of this page.
  - Press **Ctrl+F5** (Windows), **Cmd+Enter** (macOS), or **Ctrl+Search+5** (Chrome OS) on your keyboard.
- To exit presentation mode, press the **Esc** key on your keyboard.

② **Navigate by clicking the buttons and links.**

- Click the **Back** or **Next** buttons to go backward or forward in the deck. Moving forward, you'll find them in the bottom corners of every slide.
- Click blue text to go to another slide in this deck or open a new page in your browser.
- For the best learning experience, using your keyboard or mouse wheel to navigate is discouraged.

Ready to get started? **Let's go!**

Lesson 3

UDMI site model and device-to-cloud basics

What you'll learn about:

- UDMI site model set up
- How to register devices to GCP IoT Core using UDMI site model

By the end of this lesson, you'll be able to:

- Set up a UDMI site model.
- Register devices in Google Cloud using the registrar tool and a UDMI site model.
- Run and analyze the pubber tool.
- Navigate Pub/Sub topics and subscription.
- Manually verify that a device works.

Back

Next

Google

Introduction to Google Cloud IoT Core

For the purposes of UDMI, a device is defined as a processing unit that's capable of connecting to the internet and exchanging data with the cloud. Devices in this sense are often referred to as "smart devices" or "connected devices."

But with all of these different devices sending data up to the cloud, you need a way to manage those devices and the information they produce. That's where the Google Cloud Internet of Things (IoT) Core comes in.
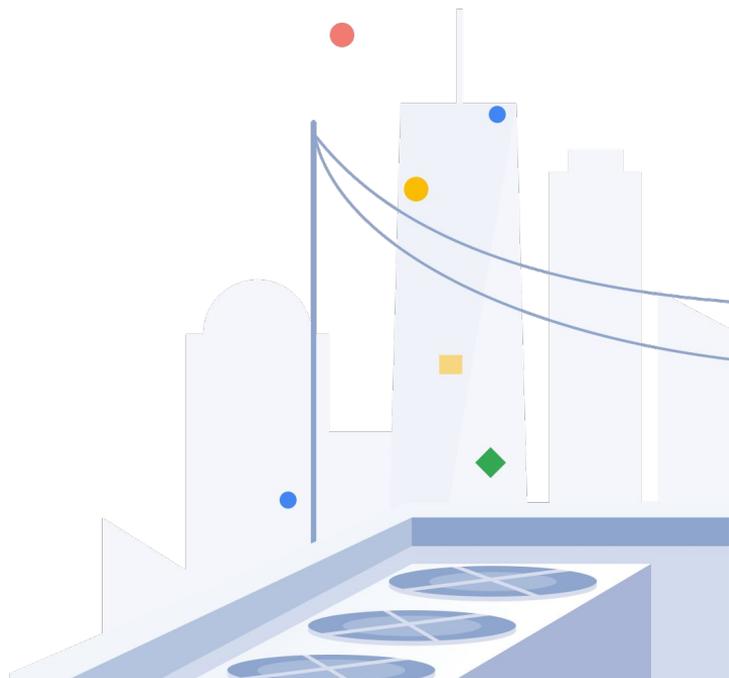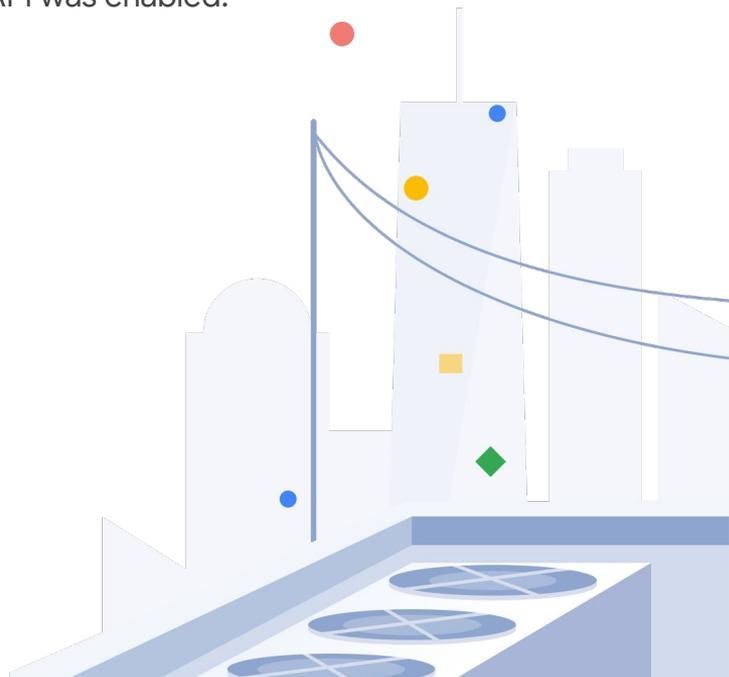
Back

Next

Introduction to Google Cloud IoT Core (continued)

When your Google Cloud Project (GCP) was set up, the Google Cloud IoT API was enabled.

This API allows you to use the Google Cloud IoT Core. As its
product page states:

> IoT Core is a fully managed service that allows you to easily and
> securely connect, manage, and ingest data from globally
> dispersed devices.
>
> IoT Core, in combination with other services on Google Cloud,
> provides a complete solution for collecting, processing,
> analyzing, and visualizing IoT data in real time to support
> improved operational efficiency.

Back

Next

Introduction to Google Cloud IoT Core (continued)

Cloud IoT Core is made up of three components:

- A **device manager** for registering devices with the service, so that you can then monitor and configure them. These devices are stored in collections called IoT Core registries. To learn more on how to set up an IoT Core registry, refer to Lesson 2.

- A **protocol bridge** (MQTT) that devices can use to connect to the Google Cloud Platform. To learn more about MQTT, refer to Publishing over the MQTT bridge.

- An **authentication** function to confirm devices' identities. Each device has a private key. It uses that key to sign a JSON Web Token (JWT), which is then passed to Cloud IoT Core as proof that the device is indeed the device it claims to be.
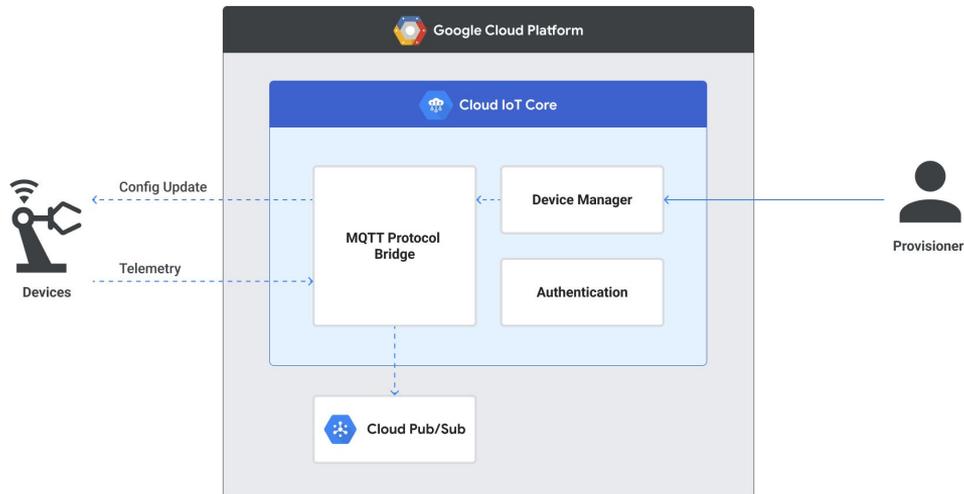
Back

Next

The UDMI site model

While the IoT Core Bridge allows devices to communicate back and forth with the cloud, it doesn't tell us which devices we should expect to see, what role they should play, or what data they should be transmitting. To know that information, we need a UDMI site model.

A project site model is a file structure that describes the devices installed on-premise and how they communicate with the cloud. It contains information necessary to specify the configuration for a particular site. Since the site model is a logical representation of the underlying information, it can be applied against different cloud projects or device configurations to ensure that things are configured appropriately.

A small example of a UDMI site model can be found on github in the udmi_site_model repository. This sample can be used for testing or as a base for a site model.

It's expected that each site in production will have its own git repo, so that it can be maintained and updated in a source controlled way.
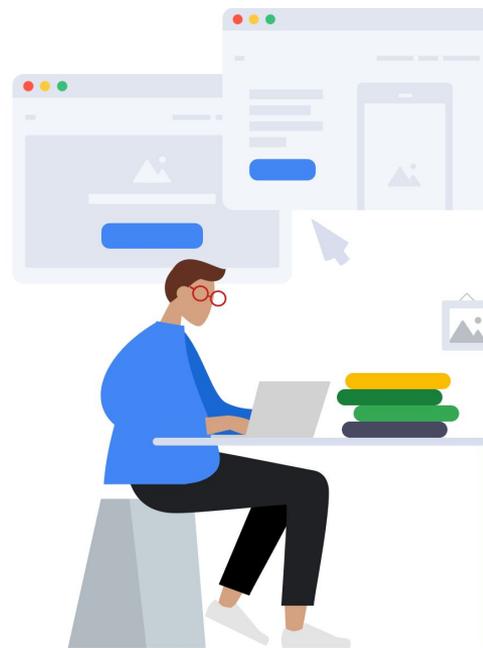
Back

Next

Your first UDMI site model

For these tutorials, we're going to use the [UDMI site model example](#).

To download it, open a terminal window and set the directory to the UDMI directory. Then type `bin/clone_model`. This will download the site model example into the `sites/udmi_site_model/` directory within the UDMI folder.

Tip: Storing your site models locally in either the `sites/` directory or in your home directory allows for easy access.

Back

Next

Your first UDMI site model (continued)

Next, let's edit the cloud_iot_config.json file.

This file contains some basic information about the GCP Cloud Project this site model is associated with.

- First, change the values for **site_name** and **registry_id** to match the name of your site registry. If you're using the values given in the tutorial in [Lesson 2: Cloud setup,](#) change the values to ZZ-PER-FECTA. If you've been given a registry id, then use that.

- Next, ensure the **cloud_region** matches the cloud region the site registry is in. You can find this information on the registry page in GCP.

Example

```
{
  "cloud_region":"us-central1",
  "site_name": "ZZ-PER-FECTA",
  "registry_id": "ZZ-PER-FECTA"
}
```
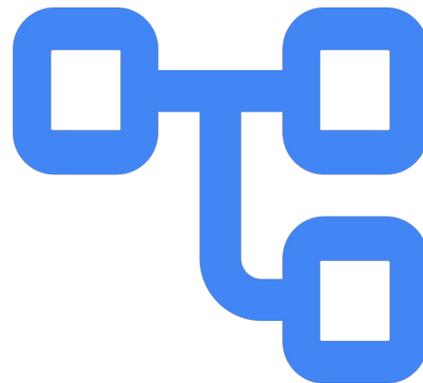
Back

Next

Your first UDMI site model (continued)

Located within the UDMI site model, the `devices` directory contains a set of descriptions for individual devices in the system.

Each device directory name corresponds to the Cloud IoT Core device entry and represents the canonical name for the device. Inside of each device directory will be various pieces of information used by different tools for managing the devices.

Back

Next

Metadata

Each device's directory also contains the device's `metadata.json` file.

This file is a description about the device, providing specifications about how the device should be configured and expectations about what the device should be doing. It's used to provide the basic information for registering devices in an IoT Core registry, generation of other template files (e.g., *config* block) associated with the device, and as a means for runtime validation of what the device is actually sending.

The metadata schema which describes all the different fields and requirements can be viewed here: https://faucetsdn.github.io/udmi/gencode/docs/metadata.html.

Additional information about the `metadata.json` file can be found here: https://faucetsdn.github.io/udmi/docs/specs/metadata.html.

Back

Next

Metadata

Here is an example of the minimum contents of a metadata file.

```
"version": 1,
"timestamp": "2018-08-26T21:39:29.364Z",
"system": {
  "location": {
    "site": "US-SFO-XYY",
    "section": "NW-2F",
    "position": {
      "x": 10,
      "y": 20
    }
  },
  "physical_tag": {
    "asset": {
      "guid": "bim://04aEp5ymD_$u5IxhJN2aGi",
      "site": "US-SFO-XYY",
      "name": "AHU-1"
    }
  }
},
"cloud": {
  "auth_type": "ES256"
},
"pointset": {
  "points": {
    "return_air_temperature_sensor": {
      "units": "Degrees-Celsius",
    },
    "room_setpoint": {
      "writable": true,
      "units": "Degrees-Celsius",
    }
  }
}
}
```

Information about the physical location of the device

Information that is printed on a physical tag

Information about how the device connects to GCP IoT Core

The different points which are to be sent by a device
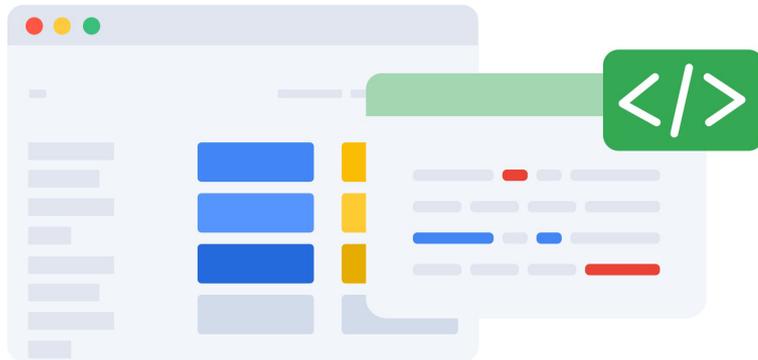
Back

Next

Google

Generating keys

Now that you've downloaded the site model, run the following command:

```
bin/genkeys sites/udmi_site_model
```

This command generates keys for devices in the site model
which don't have keys.

> **Note:** Since we are using the example site model, this step is redundant in this case, as the keys already exist. However, it's good to get in the habit of running this command at this point in the process.

Back

Next

The registrar tool

Now that we have a site model, it's time to register the devices in that model to Cloud IoT.

To do that, we'll use the **registrar**, a utility program that registers and updates devices in Cloud IoT based on the site model. You can learn more about the registrar tool here:
https://faucetsdn.github.io/udmi/docs/tools/registrar

To run the registrar tool, follow the instructions below.

1. Run registrar to validate a site model, ensuring that it is free of schema errors and other issues. The command to do this follows this pattern:
   bin/registrar PATH_TO_SITE_MODEL

2. Run registrar to register devices in the cloud project. The command to do this follows this pattern:
   bin/registrar PATH_TO_SITE_MODEL GCP_PROJECT_ID

Example
```
bin/registrar sites/udmi_site_model
```

Example
```
bin/registrar sites/udmi_site_model udmi-learning
```

Back

Next

The registrar tool (continued)

As the registrar tool works registering devices, it produces command line output which indicates what's happening and a summary at the end.

To the right is an example of such output.

Example

```
Loading local device SNS-4

Loading local device AHU-1
Loading local device GAT-123
Loading local device AHU-22
Finished loading 4 local devices.
Retrieved 0 devices from registry...
Fetched 0 devices from cloud registry projects/udmi-learning/locations/us-central1/registries/ZZ-PER-FECTA
Created new device entry GAT-123
Created new device entry AHU-1
Created new device entry SNS-4
Created new device entry AHU-22
Finished processing 4 device records...
Binding SNS-4 to gateway GAT-123
Binding AHU-22 to gateway GAT-123
Updated 4 devices (out of 4)
Skipped 4 not-in-cloud devices.

Summary:
  Device Clean: 4
Out of 4 total.
Registrar complete, exit 0
```

Back

Next

The registrar tool (continued)

The registar tool also creates a file titled
`registration_summary.json` in the site model
directory. This file includes a summary of the registration
and any devices which had errors or issues.

To the right is an example of such output.

Example

```json
{
  "Clean" : {
    "AHU-1" : "2022-03-31T09:34:14Z",
    "AHU-22" : "2022-03-31T09:34:14Z",
    "GAT-123" : "2022-03-31T09:34:14Z",
    "SNS-4" : "2022-03-31T09:34:14Z"
  },
  "Version" : {
    "main" : "1.3.12-53-gd573d5fb"
  }
}
```

Back

Next

The registrar tool (continued)

Once the registrar tool is finished, you can check the device registry GCP page to confirm that the devices have been successfully added.

To do this, search for GCP IoT Core on the GCP, then select your registry name (in this case, ZZ-PER-FECTA).

Devices are listed under the "devices" section.

☐	Device ID	Communication
☐	AHU-1	✅ Allowed
☐	AHU-22	✅ Allowed
☐	SNS-4	✅ Allowed

Gateway devices are listed under the "gateway" section.

| ☐	Gateway ID	Communication
☐	GAT-123	✅ Allowed

Back

Next

Pubber tool

Pubber simulates other devices registered to your UDMI model.

We've confirmed that devices have been registered correctly. Now let's double check that they can send data to the cloud correctly. To do that, we'll use a utility tool called pubber.

Pubber simulates other devices registered to your UDMI model. It comes in handy when testing your device setup by isolating the setup from anything external. If pubber doesn't work, it's a sign that there's an issue with the GCP configuration. If it does work and there's still an issue, it's a sign that there's a problem with the actual device.
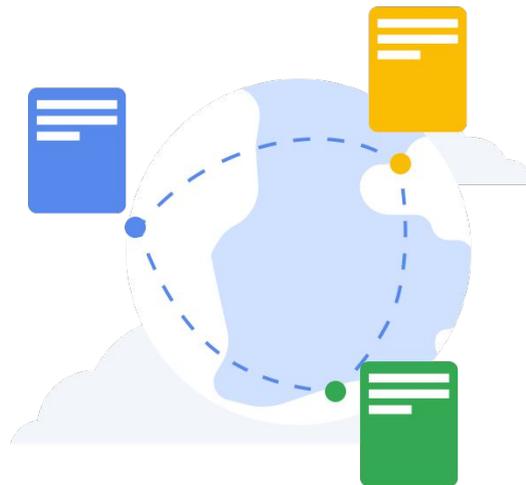
Back

Next

Pubber tool (continued)

We want pubber to simulate one of the devices in our site model.

To do that, we need to point pubber toward our site model and tell it which device to simulate. That command uses the following pattern:

`bin/pubber SITE_PATH PROJECT_ID DEVICE_ID SERIAL_NO`

In this particular instance, let's have pubber simulate the device AHU-1 in our site model. The command for that would be:

`bin/pubber sites/udmi_site_model udmi-learning AHU-1 1234`

> **Note:** You don't have to use a device's actual serial number for the command to work. You can use any random string or number.

You'll get a message saying "Building pubber" before the tool starts sending messages to the cloud.

Back

Next

Pubber tool (continued)

Once pubber is running, a verbose log is outputted to show you what's going on, including when state messages are sent.

Telemetry (pointset events) are reported incrementality (e.g., when messages 1, 100, 200, are sent).

Some example output messages can be seen below.

Example 1

```
INFO daq.pubber.Pubber - 2022-05-16T10:15:15Z 2022-05-16T10:15:15Z received config 2022-05-10T15:43:37Z
```

Example 2

```
INFO daq.pubber.Pubber - 2022-05-16T10:15:19Z State update:
```

Example 3

```
INFO daq.pubber.Pubber - 2022-05-16T10:25:20Z 2022-05-16T10:25:20Z sending test message #200
```

Back

Next

Knowledge check

Let's take a moment to reflect on what you've learned so far.

- In the upcoming slides, you'll see a task performed by a utility.
- Review each question and select which utility performs that task.

If there are more than two answer options, you won't be able to move forward until the correct answer is selected.

*Click **Next** when you're ready to begin.*

Back

Next

Google

Knowledge check 1

Which utility can be used to simulate a device sending messages to the cloud?

Select the best answer from the options listed below.

The registrar utility

The pubber utility

The validator utility

The sequencer utility

Back

Next

Knowledge check 1

Which utility can be used to simulate a device sending messages to the cloud?

Select the best answer from the options listed below.

The registrar utility

The pubber utility

The validator utility

The sequencer utility

Close... but not quite right! 🤔

Sorry, the registrar utility isn't used to simulate a device sending messages. Take another look at the answer choices and try again.

Try again

Back

Next

Knowledge check 1

Which utility can be used to simulate a device sending messages to the cloud?

Select the best answer from the options listed below.

The registrar utility

The pubber utility

The validator utility

The sequencer utility

That's right! 🎉

The pubber utility is used to simulate a device sending messages.

Back

Next

Knowledge check 1

Which utility can be used to simulate a device sending messages to the cloud?

Select the best answer from the options listed below.

The registrar utility	The pubber utility
The validator utility	The sequencer utility

Close... but not quite right! 🤔

Sorry, the validator utility isn't used to simulate a device sending messages. Take another look at the answer choices and try again.

Try again

Back

Next

Knowledge check 1

Which utility can be used to simulate a device sending messages to the cloud?

Select the best answer from the options listed below.

The registrar utility

The pubber utility

The validator utility

The sequencer utility

Close... but not quite right! 🤔

Sorry, the sequencer utility isn't used to simulate a device sending messages. Take another look at the answer choices and try again.

Try again

Back

Next

Knowledge check 2

Which utility can be used to validate a site model?

Select the best answer from the options listed below.

The registrar utility	The pubber utility
The validator utility	The sequencer utility

Back

Next

Knowledge check 2

Which utility can be used to validate a site model?

Select the best answer from the options listed below.

The registrar utility	The pubber utility
The validator utility	The sequencer utility

That's right! 🎉

You can use the registrar to validate a site model by using the following command in your terminal:

```
bin/registrar PATH_TO_SITE_MODEL
```

Back

Next

Knowledge check 2

Which utility can be used to validate a site model?

Select the best answer from the options listed below.

The registrar utility

The pubber utility

The validator utility

The sequencer utility

Close... but not quite right! 🤔

Sorry, the pubber utility isn't used to validate a site model. Take another look at the answer choices and try again.

Try again

Back

Next

Knowledge check 2

Which utility can be used to validate a site model?

Select the best answer from the options listed below.

The registrar utility

The pubber utility

The validator utility

The sequencer utility

Close... but not quite right! 🤔

Sorry, the validator utility isn't used to validate a site model. (It actually validates streaming telemetry, which is a different thing!) Take another look at the answer choices and try again.

Try again

Back

Next

Knowledge check 2

Which utility can be used to validate a site model?

Select the best answer from the options listed below.

The registrar utility

The pubber utility

The validator utility

The sequencer utility

Close... but not quite right! 🤔

Sorry, the sequencer utility isn't used to validate a site model. Take another look at the answer choices and try again.

Try again

Back

Next

Device diagnostics

Now that we have pubber simulating a device and sending data to the cloud, we need to check if that data is being received properly.

To access that data, you'll need to have subscriptions for the Pub/Sub topic **udmi_target**. This is the main topic for device communication, and provides a solid overview of device traffic. Examples of data found in this topic include main telemetry publish events, augmented state messages, config, and more.

> **Note:** For most MSIs, a subscription will have already been set up for you. To learn more about creating Pub/Sub subscriptions, see this article or check out Lesson 2: Cloud setup.

To view the messages a device is sending, return to the Google Cloud Console and go to the Pub/Sub section. Open up the subscription you were given or that you created earlier, then select the **Messages** tab. Next, select **Pull** and the table will start to populate with messages being sent from your pubber device. To view the contents of a message, simply select that message to open it.

Back

Next

Device diagnostics (continued)

Let's start by taking a look at the device pubber is simulating in the Google Cloud Console.

To do that, head back to the Google Cloud Console and select the registry you created earlier.

	Device ID	Communication	Last seen	Cloud Logging
☐	AHU-1	✅ Allowed	15 Mar 2022, 22:27:12	Registry default
☐	AHU-22	✅ Allowed	—	Registry default
☐	SNS-4	✅ Allowed	—	Registry default

Filter Enter exact device ID

Back

Next

Device diagnostics (continued)

Next, select the device that pubber is impersonating.

Doing so allows you to access a wealth of useful dates and times in the "Details" tab.

Device ID: AHU-1

Numeric ID	Registry	Cloud Logging	Communication
3050118358611915	UK-LON-XXX	Registry default View logs	Allowed

DETAILS   CONFIGURATION AND STATE   AUTHENTICATION

Latest activity

Heartbeat (MQTT only)	—
Telemetry event received	15 Mar 2022, 22:30:22
Device state event received	15 Mar 2022, 22:09:32
Config sent	15 Mar 2022, 22:09:23
Zone config. ACK (MQTT only)	15 Mar 2022, 22:09:23
Error	15 Mar 2022, 21:04:08
Error status and message	[4] mqtt: SERVER: The connection was closed because MQTT keep-alive c...

˅ SHOW MORE

Back

Next

Device diagnostics (continued)

By selecting the "Configuration and State" tab, you can see configuration and state messages that have been received from the device.

Back

Next

Device diagnostics (continued)

Select any message to see its full contents.

Cloud update: 15 March 2022

| Latest State | ✓ | STATE | Cloud update 22:09 | {"timestamp":"2022-03-15T22:09:32 |

| Cloud update | 22:09 |

Format

○ Base64
● Text

{"timestamp":"2022-03-15T22:09:32Z","version":1,"system":{"make_model":"DAQ_pubber","serial_no":"1234","firmwa ⎘ :{"

REVERT

Back

Next

Device diagnostics (continued)

If you select **Update Config,** you can manually update the device's configuration.

Navigate Pub/Sub topics and subscription

One of the benefits of using IoT Core is the ability to view aggregated device data via Pub/Sub.

Let's take a closer look at how to navigate through Pub/Sub topics and subscriptions. To access Pub/Sub data, return to the Google Cloud Console and then go to the Pub/Sub section.

Pub/Sub subscription

To view the messages a device is sending, select **Subscriptions**. Then select **udmi_target_subscriptions**.

> **Note:** If you're using a Google-managed project, you'll need to access the subscription directly via the URL provided to you.

By default, you should start in the "Messages" tab. Select **Pull**.

The table will start to populate with messages being sent from your pubber device (and any other devices publishing data at the same time).
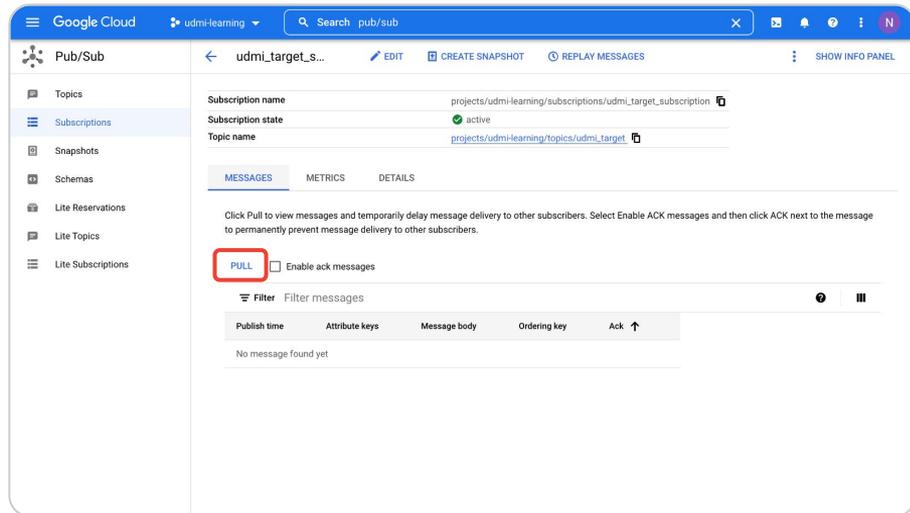
Back

Next

Navigate Pub/Sub topics and subscription (continued)

To view the contents of a message, simply select the arrow to expand the row. Scroll horizontally to view all message attributes.

Navigate Pub/Sub topics and subscription (continued)

Select the Metrics tab to see some high level aggregations.
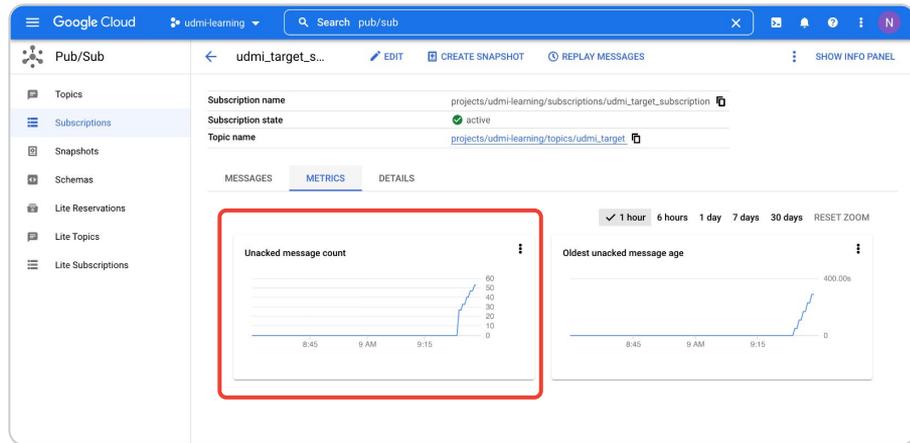
Knowledge check

Let's take a moment to reflect on what you've learned so far.

- The next slides will have a question related to viewing device messages within Google Cloud Project.

- Review each question and select the correct response.

If there are more than two answer options, you won't be able to move forward until the correct answer is selected.

*Click **Next** when you're ready to begin.*

Back

Next

Google

Knowledge check 3

Which of the following is NOT a method for viewing a message received from a device?

Select the best answer from the options listed below.

From the "Topic" page within the Pub/Sub section of the Google Cloud Project

From the "Subscriptions" page within the Pub/Sub section of the Google Cloud Project

From the device page within the registry on Google Cloud Project

Back

Next

Knowledge check 3

Which of the following is NOT a method for viewing a message received from a device?

Select the best answer from the options listed below.

From the "Topic" page within the Pub/Sub section of the Google Cloud Project

From the "Subscriptions" page within the Pub/Sub section of the Google Cloud Project

From the device page within the registry on Google Cloud Project

That's right! 🎉

While the "Topic" page of the Pub/Sub section of GCP shows high-level aggregations of message data, you can't view the content of individual device messages on that page.

Back

Next

Knowledge check 3

Which of the following is NOT a method for viewing a message received from a device?

Select the best answer from the options listed below.

From the "Topic" page within the Pub/Sub section of the Google Cloud Project

From the "Subscriptions" page within the Pub/Sub section of the Google Cloud Project

From the device page within the registry on Google Cloud Project

Close… but not quite right! 🤔

Actually, it is possible to view the contents of individual messages from the "Subscriptions" page by viewing the Messages subtab. Take a closer look at the answers and think about the purpose of each location, then try answering the question again.

Try again

Back

Next

Knowledge check 3

Which of the following is NOT a method for viewing a message received from a device?

Select the best answer from the options listed below.

From the "Topic" page within the Pub/Sub section of the Google Cloud Project

From the "Subscriptions" page within the Pub/Sub section of the Google Cloud Project

From the device page within the registry on Google Cloud Project

Close… but not quite right! 🤔

Actually, it is possible to view the contents of individual messages by clicking on any messages seen with the "Configuration and State" subtab of a device page. Take a closer look at the answers and think about the purpose of each location, then try answering the question again.

Try again

Back

Next

Lesson 3

Practice

Now, let's take a moment to apply what you've learned so far.

- The next slide will present the steps to add a new device to the site model.

- Review the steps and practice on your own.

*Click **Next** when you're ready to begin.*

Back

Next

Google

Adding a new device to the site model

1. In the devices directory of the site model, create a directory named "FCU-1."

2. In this new directory, create a new file, metadata.json (or copy the metadata from another device to use as a template).

3. Fill in or update the required details and add a few points.

4. Run the genkeys tool to generate the keys.

5. Run the registrar tool to validate the site model and register the new device on GCP.

6. Check that the device has been registered to the GCP.

7. Run pubber with the new device (FCU-1), and verify the device is online and sending data by checking its state and telemetry messages.

Back

Next

Lesson 3 summary

Let's review what you learned about:

- UDMI site model set up
- How to register devices to GCP IoT Core using UDMI site model

Now you should be able to:

- Set up a UDMI site model.
- Register devices in Google Cloud using the registrar tool and a UDMI site model.
- Run and analyze the pubber tool.
- Navigate Pub/Sub topics and subscription.
- Manually verify that a device works.

Back

Next

You completed Lesson 3!

Now's a great time to take a quick break before starting Lesson 4.

Ready for Lesson 4?

Let's go!

Helpful resources

Bookmark these resources for future reference.

- UDMI Project GitHub
  Contains specification for management and operation of IoT systems.
- Git Documentation
  Contains various sources of information about Git contributed by the Git community.

Back

*Press the **Esc** key on your keyboard to exit presentation mode.*

Google