



Module 2 | Lesson 6



Data modeling with the DBO



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.

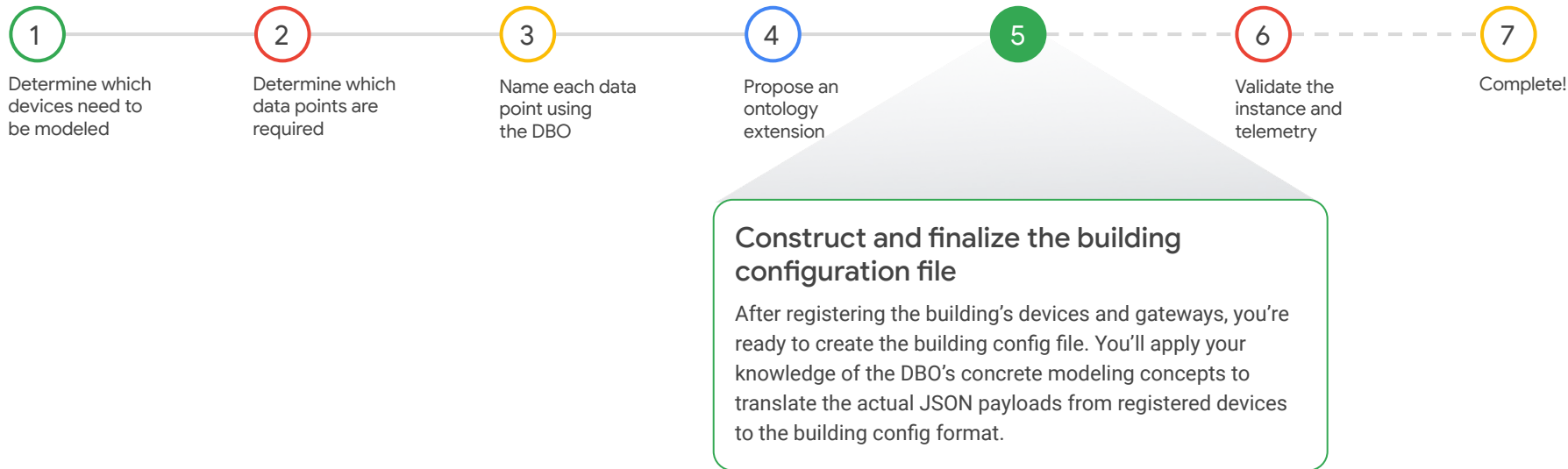
Ready to get started?

Let's go!

Workflow revisited

Here's the recommended workflow for data modeling from Lesson 1.

In this lesson, you'll walk through the fifth step of data modeling with the DBO.



[Back](#)

[Next](#)



Lesson 6

Construct and finalize the building configuration file

[Back](#)

What you'll learn about:

- The contents and format of building configuration files
- Constructing a building config file
- Finalizing a building config file

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

- Recognize the contents and format of a building config file.
- Construct a building config using the configuration format to define:
 - Translations for devices.
 - Entities for spaces, zones, and control groups.
 - Connections for spaces, devices, zones, and control groups.
 - Links for devices.
- Finalize a building config by generating GUIDs.

[Next](#)

Building configuration file

A **building configuration file** maps real-world devices to the Digital Buildings Ontology (DBO).

Also known as a building configuration file or simply building config for short, these files are an important part of every digital building project that uses the DBO. Building configs make a building's data useful and recognizable across any deployment by mapping the data communicated to Cloud IoT using the DBO.

Example

```
1  US-MTV-1234:  
2      id: FACILITIES/buildings/36166722673234  
3      type: FACILITIES/BUILDING  
4  
5  EF - 3 Restroom / Bldg:  
6      cloud_device_id: '2804802894218214135'  
7      connections:  
8          US-MTV-SB55:  
9              - CONTAINS  
10         id: CDM/2804808941814135  
11         translation:  
12             run_command:  
13                 present_value: data.binary-output_1.present-value  
14                 states:  
15                     'OFF': inactive  
16                     'ON': active  
17             run_status:  
18                 present_value: data.binary-input_1.present-value  
19                 states:  
20                     'OFF': inactive  
21                     'ON': active  
22         type: HVAC/FAN_SS
```

[Back](#)[Next](#)

Building configuration file (continued)

What's contained in a building config file?

All of the relevant information about a building and its installed equipment is encoded in a building config file. Its contents should be able to describe what devices exist, what types they apply, which devices connect to which other devices, what devices serve which zone, etc.

Some data elements are expected in every building config including:

- The spaces in the building (i.e., Building, Rooms, Floors) with their unique names.
- Each logical entity and its associated entity type.
- Each reporting device that's registered in Cloud IoT.
- Link mappings between the points of reporting devices and logical devices if the two are not the same.
- Translation mappings between device-native point names and the standard field names for each reporting device.
- **FEEDS** connections between chained equipment in HVAC or power systems and between terminal units and Zones.
- **CONTROLS** connections between switch groups and fixtures and between switches and switch groups.
- **CONTAINS** connections between zones or switch groups and rooms.
- **CONTAINS** connections between all entities and floors (or a more specific space, if known).
- **CONTAINS** connections between buildings and floors and between floors and rooms.

[Back](#)[Next](#)

Building configuration file (continued)

How many building config files are needed?

Short answer: at least one building config file is needed. It doesn't matter how many building config files are created as long as the information between them is consistent.

Breaking up a building config

A single building config can be broken up into separate files for ease of management as long as the connections between objects in the files are valid and the objects themselves are valid.

Sometimes, it will be most convenient to produce one file for a project in a small area of a building.

Sometimes, it will be most convenient to break the systems apart into individual files to prevent thousands of devices from being in one gigantic file. For example, all FCUs in one file, all FACILITIES in another, etc.

Use your best judgment for your specific project's needs.

[Back](#)[Next](#)

Building configuration file (continued)

What's needed to construct a building config?

While constructing a building config, you'll need the following:

- ☒ Rough-in model with named data points
- ☒ JSON payload(s) from the building's registered device(s) and gateway(s)
- ☒ Tools
 - JSON formatter like [go/jsonformatter](#)
 - Text editor like [Sublime Text](#)

For finalizing a building config, you'll also need:

- ☒ A version of Python installed on your machine (see [python.org](#))
- ☒ The Digital Buildings toolkit installed on your machine (see [instructions](#))
- ☒ The GUID Generator installed on your machine (see [instructions](#))

Note: In the future, we'll supply tooling to help you easily create a building config file from a standard "rough-in" template. For now, you'll need to perform this conversion between the collected information and the building config files as a manual process.

[Back](#)[Next](#)

Contents of the building config

The building config contains an entity for every reporting device, space, and zone in a building.

Click on each item to reveal more info about building config contents.

Reporting devices

Here are the entities that need to be defined for devices, spaces, and zones:

Spaces

Logical entities

A **logical entity** (also known as a canonical entity) is the concrete instance of any device, system, or entity that maps one-to-one with a canonical entity type in the DBO.

Passthrough entities

A **passthrough entity** is a reporting entity that does nothing more than pass data from a network controller to logical entities.

Zones and control groups

Reporting entities

A **reporting entity** is the concrete instance of a reporting device expressed in the building configuration file.

Virtual entities

A **virtual entity** is a representation of a logical entity constructed by linking the fields of a reporting entity to the fields of a logical entity.

[Back](#)

[Next](#)

Contents of the building config

The building config contains an entity for every reporting device, space, and zone in a building.

Click on each item to reveal more info about building config contents.

Reporting devices

Spaces

Zones and control groups

Reporting devices

A **reporting device** is any device or system that generates and sends a payload of data to Cloud IoT.

A building config should have an entity for every reporting device in the building.

Some examples include:

- A controller for an individual device
- A network controller for multiple devices

[Back](#)

[Next](#)

Contents of the building config

The building config contains an entity for every reporting device, space, and zone in a building.

Click on each item to reveal more info about building config contents.

- Reporting devices
- Spaces**
- Zones and control groups

Spaces

A building config should have an entity for the building itself and its individual floors and rooms.

Config format

Building

```
UK-LON-S2:  
  type: FACILITIES/BUILDING  
  id: FACILITIES/123456
```

Room

```
UK-LON-S2-1-1C3G:  
  type: FACILITIES/ROOM  
  id: FACILITIES/2345678  
  connections:  
    UK-LON-S2-2: CONTAINS
```

Floor

```
UK-LON-S2-1:  
  type: FACILITIES/FLOOR  
  id: FACILITIES/3456789  
  connections:  
    UK-LON-S2: CONTAINS
```

[Back](#)

[Next](#)

Contents of the building config

The building config contains an entity for every reporting device, space, and zone in a building.

Click on each item to reveal more info about building config contents.

Reporting devices

Spaces

Zones and control groups

Zones and control groups

Most buildings have other logically defined areas or groups that aren't strictly a reporting device or space. Some examples include an HVAC zone or a lighting control group.

A building config should have an entity for every zone and control group. These entities are usually neither reporting entities nor virtual entities – they have no telemetry fields at all.

Config format

HVAC zone

```
ZONE-123:  
  type: HVAC/ZONE  
  connections:  
    UK-LON-6PS-1: CONTAINS  
    VAV-123: FEEDS
```

Lighting zone

```
LZ-234:  
  type: LIGHTING/ZONE  
  connections:  
    UK-LON-6PS-1: CONTAINS  
    SW-456: CONTROLS
```

[Back](#)

[Next](#)

Configuration format

A building config is constructed using the configuration format.

Sample entity

This is a generic entity with all possible top-level fields pulled from the [building_config.md](#). You can use this as a basic template to start a new building config .yaml file using your preferred text editor.

Config format

```
ENTITY-NAME:
  type: NAMESPACE/A_DIGITAL_BUILDINGS_ENTITY_TYPE
  id: SYSTEM/ID1234
  guid: 1937y5198347190
  cloud_device_id: device-id-from-cloud-iot-registry
  translation:
    zone air temperature sensor:
      present_value: "points.temp_1.present_value"
      units:
        key: "pointset.points.temp_1.units"
        values:
          degrees celsius: "degC"
    supply air isolation damper command:
      present_value: "points.damper_1.present_value"
      states:
        OPEN: "1"
        CLOSED:
          - "2"
          - "3"
  connections:
    # Listed entities are sources on connections
    ANOTHER-ENTITY: FEEDS
    A-THIRD-ENTITY: CONTAINS
  links:
    A-FOURTH-ENTITY: # source device
      # target device field: source device field
      supply air damper position command: supply air damper_command_1
      zone_air_temperature: zone_air_temperature_sensor_1
```

[Back](#)

Note: By now, you should be able to define an entity for a reporting device. Later in this lesson, we'll walk through how to define entities for spaces in a building. For a refresher about reporting entities specifically, see previous lessons in Module 3 and [Module 1, Lesson 6](#).

[Next](#)



Define translations

Begin constructing the building config by converting a reporting device's native payload into the DBO format.

A building config needs to map a reporting device's native payload to concepts in the DBO. This is done by defining translations that map a reporting device's native payload to its corresponding reporting entity.

[Back](#)

Config format

```
ENTITY-NAME:
...
cloud device_id: device-id-from-cloud-iot-registry
translation:
  zone air temperature sensor:
    present_value: "points.temp_1.present_value"
    units:
      key: "pointset.points.temp_1.units"
      values:
        degrees celsius: "degC"
    supply air isolation damper command:
      present_value: "points.damper_1.present_value"
      states:
        OPEN: "1"
        CLOSED:
          - "2"
          - "3"
    zone_air_temperature_setpoint: MISSING
...
```

Note: A **translation** is a mapping between a device in the real world and its corresponding concepts in the DBO. Buildings, floors, and rooms won't usually have translations defined on them. Review [Module 1, Lesson 7](#) or [building_config.md](#) for more info.

[Next](#)

Define translations

Translations are defined on reporting entities. They are configured by listing the required fields from the reporting entity's entity type inside the `translation` block. Optional fields are omitted unless they need to be mapped.

Paths to the payload

Within each field block, information is provided about the following:

- The `present_value` of the corresponding point in the JSON payload. This defines the fully qualified path in the payload that contains the value of this field.
- The `units` if the `present_value` is a dimensional number. This defines the dimensional unit for the value of this field.
 - The `key` defines the fully qualified path in the payload that represents the units.
 - The `values` map a standard unit to the value in the payload that represents the units.
- The `states` if the `present_value` is a multi-state. This maps the native state values to standard state values.
- If the device lacks a required field for its entity type, the field should be marked `MISSING`. This is very important, and you will often use this when a device looks like a canonical device but does not send all of the data for that entity type definition.

Config format

```
ENTITY-NAME:
...
cloud device_id: device-id-from-cloud-iot-registry
translation:
  zone air temperature sensor:
    present_value: "points.temp_1.present_value"
    units:
      key: "pointset.points.temp_1.units"
      values:
        degrees celsius: "degC"
    supply air isolation damper command:
      present_value: "points.damper_1.present_value"
      states:
        OPEN: "1"
        CLOSED:
          - "2"
          - "3"
    zone air temperature setpoint: MISSING
...
```

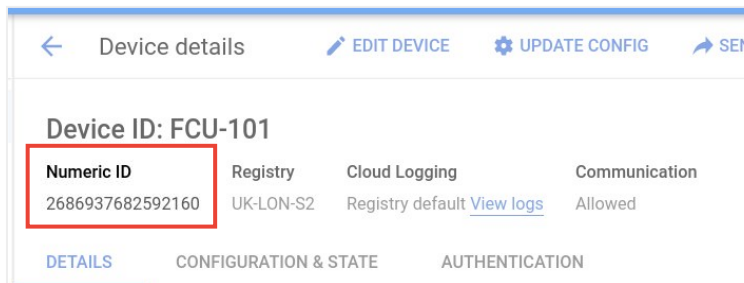
[Back](#)[Next](#)

Define translations (continued)

Translations require `cloud_device_id`

The `cloud_device_id` field is mandatory any time a translation is defined on a reporting entity in order for it to validate properly.

The `cloud_device_id` is retrieved from the Cloud IoT Registry and can be found for every device that's registered to the cloud. It's the Numeric ID shown in the screenshot below.



Config format

```
ENTITY-NAME:
...
cloud_device_id: device-id-from-cloud-iot-registry
translation:
  zone air temperature sensor:
    present_value: "points.temp_1.present_value"
    units:
      key: "pointset.points.temp_1.units"
      values:
        degrees celsius: "degC"
    supply air isolation damper command:
      present_value: "points.damper_1.present_value"
      states:
        OPEN: "1"
        CLOSED:
          - "2"
          - "3"
    zone_air_temperature_setpoint: MISSING
...
```

[Back](#)

[Next](#)

Define translations (continued)

Steps to define a translation

After defining a reporting entity, in the building config file:

1. Enter the **translation:** block.
2. Within the **translation:** block, list the required fields of the reporting entity's entity type. Also list any optional fields that need to be mapped.
3. For each required field, qualify the path to the payload.
 - Confirm there is a point from the payload that corresponds with the field.
 - If there isn't, enter **MISSING** and repeat Step 3 for each required field.
 - Enter a **present_value:** line and qualify the point path using this format: `points.name_of_point.present_value`.
 - Determine whether the point is a dimensional number or a multi-state.
 - If the point is a dimensional number, enter the **units:** block to define the dimensional unit for the point. Within this block:
 - Enter the **key:** line and qualify the point path using this format: `points.name_of_point.units`.
 - Enter the **values:** line and map the standard unit to the point's unit using this format: `standard_unit: "pointUNIT"`.
 - Refer to [units.yaml](#) for all standard dimensional units.
 - If the point is a multi-state, enter the **states:** block to define the states for the point. Within this block:
 - Enter each standard state followed by a `:`. Refer to [states.yaml](#) for all standard states.
 - After each state, enter the point's state enclosed in `" "`.
4. Repeat Step 3 for each required field that needs to be translated.
5. Enter the `cloud_device_ID:` line and retrieve the reporting device's Numeric ID from the Cloud IoT Registry.
6. Save your work.

[Back](#)

[Next](#)

Lesson 6

Practice 1



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

- For the duration of this lesson, you'll use the configuration format to construct a building config for an exhaust fan.
- The next slides will walk through the steps to set up a new building config file and translate an entity.
- After this practice activity, you'll move on to define entities for the rest of a building.
- Keep the file you create easily accessible for additional practice activities.

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

[Back](#)

[Next](#)

Practice 1

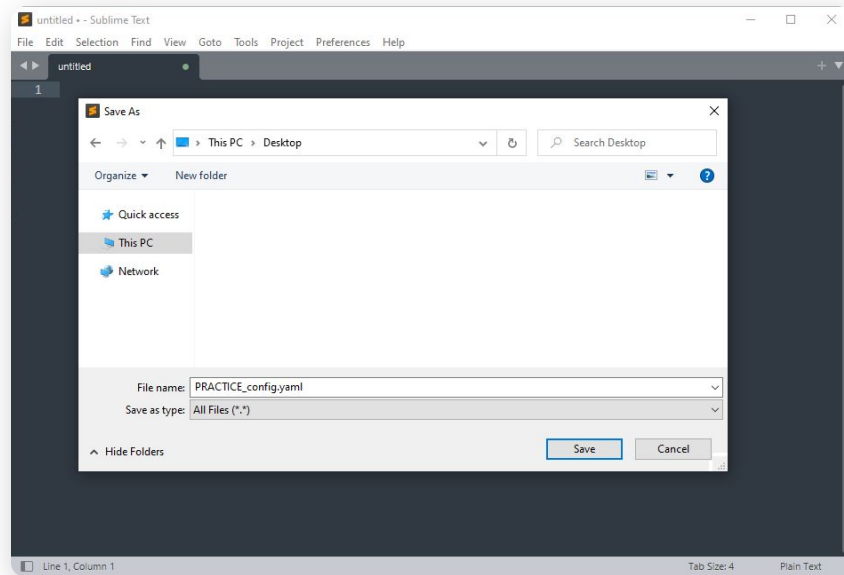
First, create a new building config file.

Follow the steps to displayed below.

Steps

1. Create a new file in your preferred text editor.
2. Save the file as "PRACTICE_config.yaml" for easy reference in this lesson's activities.

Text editor



[Back](#)

*When you're ready, click **Next** to continue this practice activity.*

[Next](#)

Practice 1 (continued)

Next, let's revisit the exhaust fan we defined in Lesson 5 and define its translations.

In the previous lesson, we defined new ontology concepts to describe an exhaust fan (**EF-1**) that operates to maintain low radon levels in a space.

Here's our rough-in with the fields we arrived at and a payload the cloud may receive from the exhaust fan.

Rough-in sheet

	A	B	C	D	E	F
1	Equipment Name	Point Name	Units	Description	Entity Type	Field
2	EF-1	radon_lvl	PPM	Detected radon level.	FAN_SS_RNC	zone_air_radon_concentration_sensor
3	EF-1	radon_lvl_stpt	PPM	Radon level setpoint; threshold where the fan turns on and off.		zone_air_radon_concentration_setpoint
4	EF-1	fan_ss	NO-UNITS	Fan command to run		run_command
5	EF-1	fan_sts	NO-UNITS	Fan feedback, indicating it is running.		run_status
6	EF-1	fan_alarm	NO-UNITS	Fan alarm, indicating it has failed.		

Return to “PRACTICE_config.yaml” in your text editor and define a translation to map the exhaust fan’s native payload to the reporting entity **EF-1**.

Use the information provided above.

Payload

```
{
  "timestamp": "2021-08-18T15:33:06.000Z",
  "version": 1,
  "points":
  {
    "fan_ss":
    {
      "present_value": false,
      "units": "No-units"
    },
    "fan_sts":
    {
      "present_value": false,
      "units": "No-units"
    },
    "radon_lvl":
    {
      "present_value": 18.622520000000002,
      "units": "PPM"
    },
    "radon_lvl_stpt":
    {
      "present_value": 20.0,
      "units": "PPM"
    },
    "fan_alarm":
    {
      "present_value": false,
      "units": "No-units"
    }
  },
  "device_id": "EF-1",
  "type": "udmi"
}
```

[Back](#)

When you're ready, click **Next** to check your work.

[Next](#)

Check your work!



This is what our building config file looks like after defining the translation on **EF-1**.

Did you end up with something similar?

Notice how the field `fan_alarm` is not in the translation but is contained in the payload. This is totally valid, and gives you the ability to pass data that you don't necessarily care to model to the cloud.



```
16
17 EF-1:
18   type: HVAC/FAN_SS_RNC
19   id: 12345
20   cloud_device_id: abc123
21   translation:
22     zone_air_radon_concentration_sensor:
23       present_value: "points.radon_lvl.present_value"
24       units:
25         key: "pointset.points.radon_lvl.units"
26         values:
27           parts_per_million: "PPM"
28     zone_air_radon_concentration_setpoint:
29       present_value: "points.radon_lvl_stpt.present_value"
30       units:
31         key: "pointset.points.radon_lvl_stpt.units"
32         values:
33           parts_per_million: "PPM"
34     run_command:
35       present_value: "points.fan_ss.present_value"
36       states:
37         ON: "true"
38         OFF: "false"
39     run_status:
40       present_value: "points.fan_sts.present_value"
41       states:
42         ON: "true"
43         OFF: "false"
44
```

[Back](#)

Keep this file easily accessible for the next activity.
Click **Next** to complete this activity and move on to defining additional entities.

[Next](#)



Define entities

Continue constructing the building config by defining entities for a building's spaces.

A building config needs to establish the context for reporting entities that are within a building. This is done by defining entities for the building and its relevant floors and rooms.

[Back](#)

Config format

```
ENTITY-NAME:
  type: NAMESPACE/A_DIGITAL_BUILDINGS_ENTITY_TYPE
  id: SYSTEM/ID1234
  guid:
  ...
ANOTHER-ENTITY:
  type: NAMESPACE/A_DIGITAL_BUILDINGS_ENTITY_TYPE
  id: SYSTEM/ID2345
  guid:
  ...
A-THIRD-ENTITY:
  type: NAMESPACE/A_DIGITAL_BUILDINGS_ENTITY_TYPE
  id: SYSTEM/ID3456
  guid:
  ...
A-FOURTH-ENTITY:
  type: NAMESPACE/A_DIGITAL_BUILDINGS_ENTITY_TYPE
  id: SYSTEM/ID4567
  guid:
  ...
```

[Next](#)

Define entities

Within each space entity, information is provided about the following:

Entity name : Refer to the `device_id` in the JSON payload or the name that is specified in the mechanical drawings (if you are assigning these in net-new devices).

Entity type : Refer to your rough-in and [Facilities.yaml](#) in the FACILITIES namespace.

ID : This is automatically generated by our system.

GUID: A globally unique id generated by the GUID Generator tool later in the process.

Config format

Building

```
BUILDING:  
  type: FACILITIES/BUILDING  
  id: FACILITIES/123456  
  guid:
```

Floor

```
BUILDING-FLOOR:  
  type: FACILITIES/FLOOR  
  id: FACILITIES/3456789  
  guid:  
  ...
```

Room

```
BUILDING-FLOOR-ROOM:  
  type: FACILITIES/ROOM  
  id: FACILITIES/2345678  
  guid:  
  ...
```

[Back](#)

Note: The `id` line will be deprecated in the future. For now, you'll include the `id` and `guid` lines.

[Next](#)

Define entities (continued)

Steps to define an entity

1. Review the JSON payload to identify the `device_id` for spaces (building, floors, and rooms), devices, zones, or control groups that need to be modeled. This is the entity name.
 - **Note:** Try using a JSON formatter like go/jsonformatter to convert the payload into a more readable format. If using Sublime or another IDE, there are usually built-in JSON formatters that you can also use.
2. In the building config file, enter the `device_id` as the entity name.
3. Enter the `type:` line to identify the entity type and properly qualify its namespace.
 - For spaces, refer to [Facilities.yaml](#).
 - For devices, refer to your rough-in and the global and child namespaces in [digitalbuildings / ontology / yaml / resources](#).
4. Enter lines for the `id:` and `guid:`. These will be generated later.
5. Repeat Steps 1-4 for each required entity.
6. Save your work.

[Back](#)

[Next](#)

Lesson 6

Practice 2



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

- Picking up where you left off, you'll use the "PRACTICE_config.yaml" file you created in this lesson's first practice activity and continue constructing a building config that includes the exhaust fan **EF-1**.
- The next slides will walk through defining entities for a building's spaces to put **EF-1** into the context of its building.
- After this practice activity, you'll move on to define connections.
- Keep the file you're working on easily accessible for additional practice activities.

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

[Back](#)

[Next](#)

Practice 2

Continuing from the previous practice activity, let's put **EF-1** into the context of a building.

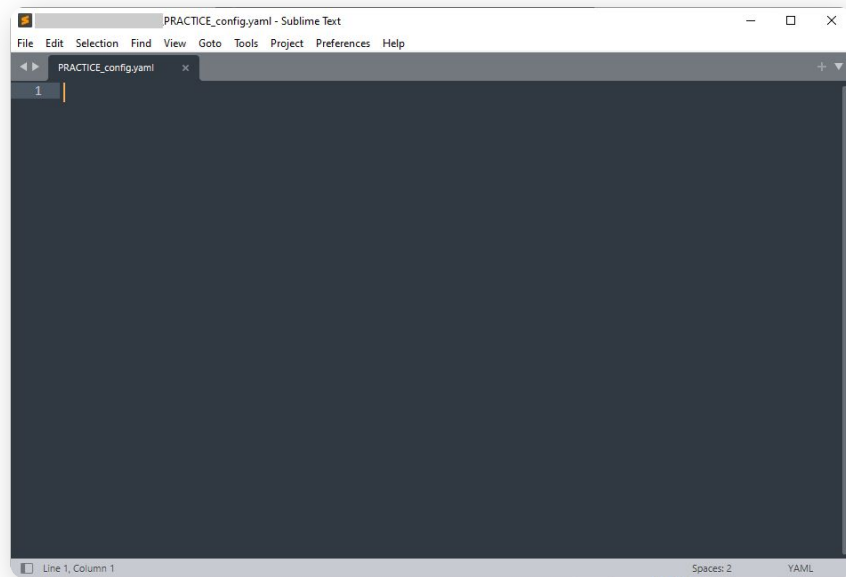
According to the payload and the project documents you previously received about the building, you've been able to pinpoint the exact location of the exhaust fan EF-1:

- Building: **US-MTV-1111**
- Floor: **US-MTV-1111-1**
- Room: **US-MTV-1111-1-LAB**

Return to “PRACTICE_config.yaml” in your text editor and define entities for the space containing **EF-1**.

Use the information provided above and the proper config format.

Text editor



[Back](#)

When you're ready, click **Next** to continue this practice activity.

[Next](#)

Check your work!

This is what our building config file looks like after defining the entities for the space containing **EF-1**.

Did you end up with something similar?



```
1
2  US-MTV-1111:
3    type: FACILITIES/BUILDING
4    id: FACILITIES/123456
5
6
7  US-MTV-1111-1:
8    type: FACILITIES/FLOOR
9    id: FACILITIES/3456789
10
11
12  US-MTV-1111-1-LAB:
13    type: FACILITIES/ROOM
14    id: FACILITIES/2345678
15
16
17  EF-1:
18    type: HVAC/FAN_SS_RNC
19    id: 12345
20    cloud_device_id: abc123
21    translation:
22      zone_air_radon_concentration_sensor:
23        present_value: "points.radon_lvl.present_value"
24        units:
25          key: "pointset.points.radon_lvl.units"
26          values:
27            parts_per_million: "PPM"
28      zone_air_radon_concentration_setpoint:
29        present_value: "points.radon_lvl_stpt.present_value"
30        units:
31          key: "pointset.points.radon_lvl_stpt.units"
32          values:
33            parts_per_million: "PPM"
34      run_command:
35        present_value: "points.fan_ss.present_value"
36        states:
37          ON: "true"
38          OFF: "false"
39      run_status:
40        present_value: "points.fan_sts.present_value"
41        states:
42          ON: "true"
43          OFF: "false"
44
```

[Back](#)

Keep this file easily accessible for the next activity.
Click **Next** to complete this activity and move on to defining connections.

[Next](#)



Define connections

Continue constructing the building config by defining connections between spaces and devices.

A building config needs to describe the various system and spatial relationships between the building's spaces and devices. This is done by defining connections between the entities it includes.

[Back](#)

Config format

```
ENTITY-NAME:
  type: NAMESPACE/A_DIGITAL_BUILDINGS_ENTITY_TYPE
  id: SYSTEM/ID1234
  guid:
  ...
  connections:
    # Listed entities are sources on connections
    ANOTHER-ENTITY: FEEDS
  ...
ANOTHER-ENTITY:
  type: NAMESPACE/A_DIGITAL_BUILDINGS_ENTITY_TYPE
  id: SYSTEM/ID2345
  guid:
  ...
```

[Next](#)

Define connections

Connections are defined on the target entity and not the source entity.

Building config

Spaces

Buildings

Connections aren't defined on the building entity.

Floors

A building should be defined on the target floor entity using a **CONTAINS** connection.

```
US-MTV-1111-1:
...
connections:
  US-MTV-1111: CONTAINS
...
```

Room

A floor should be defined on the target room entity using a **CONTAINS** connection.

```
US-MTV-1111-1-LAB:
...
connections:
  US-MTV-1111-1: CONTAINS
...
```

Devices, zones, and control groups

A floor should be defined on the target device entity using a **CONTAINS** connection. In addition, if any other devices, zones, or control groups have a relationship with the device entity, it should be defined on the target device entity using the most appropriate connection type. For available connection types, see [connections.yaml](#).

```
ENTITY-NAME:
...
connections:
  US-MTV-1111-1: CONTAINS
  ANOTHER-ENTITY: FEEDS
...
```

[Back](#)

Note: A **connection** is a directional relationship from a source to a target entity that's always defined on the target entity. Review [Module 1, Lesson 8](#) for more info.

[Next](#)

Define connections (continued)

Steps to define a connection

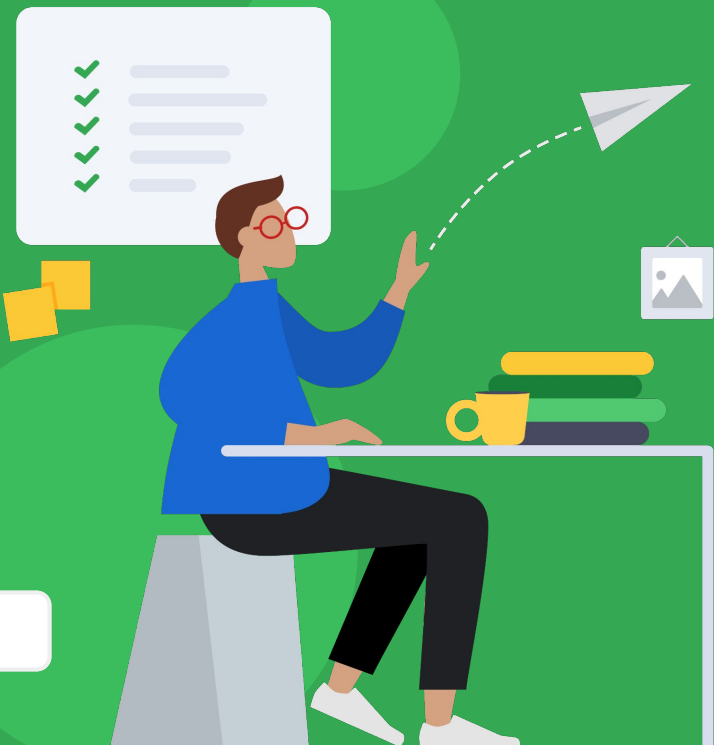
1. In the building config file, locate the entities you previously defined for spaces (building, floors, and rooms), devices, zones, and control groups that need to be modeled.
2. For each entity, enter the `connections:` block below the `id:` and define the connection type.
 - Buildings: Connections aren't defined on the building entity.
 - Floors: A building should be defined on the target floor entity using a `CONTAINS` connection.
 - Room: A floor should be defined on the target room entity using a `CONTAINS` connection.
 - Devices connected to a space: A floor should be defined on the target device entity using a `CONTAINS` connection.
 - Devices connected to another device: Other devices, zones, or control groups should be defined on the target device entity using the most appropriate connection type. Refer to [connections.yaml](#) for all connection types.
3. Save your work.

[Back](#)

[Next](#)

Lesson 6

Practice 3



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

- Picking up where you left off, you'll use the "PRACTICE_config.yaml" file you created in this lesson's first practice activity and continue constructing a building config that includes the exhaust fan **EF-1**.
- The next slides will walk through defining connections between a building's spaces and devices.
- After this practice activity, you'll move on to define links.
- Keep the file you're working on easily accessible for additional practice activities.

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

[Back](#)

[Next](#)

Practice 3

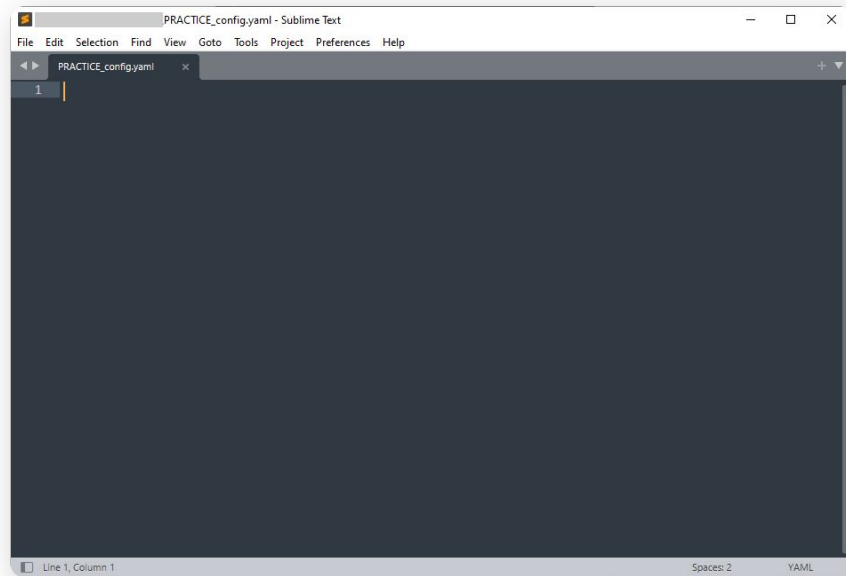
Continuing from the previous practice activity, let's define connections within **EF-1**'s building.

According to the payload and the project documents you previously received about the building, you identified that **EF-1** feeds air to the room **US-MTV-1111-1-LAB**. You also identified the room **US-MTV-1111-1-LAB** is on the floor **US-MTV-1111-1** inside the building **US-MTV-1111**.

Return to “PRACTICE_config.yaml” in your text editor and define connections between all of the entities you’ve defined so far.

Use the information provided above and the proper config format.

Text editor



[Back](#)

When you're ready, click **Next** to continue this practice activity.

[Next](#)

Check your work!

This is what our building config file looks like after defining the connections between the entities we previously defined.
Did you end up with something similar?



```
1
2 ▼ US-MTV-1111:
3   type: FACILITIES/BUILDING
4   id: FACILITIES/123456
5
6
7 ▼ US-MTV-1111-1:
8   type: FACILITIES/FLOOR
9   id: FACILITIES/3456789
10  connections:
11    - US-MTV-1111: CONTAINS
12
13 ▼ US-MTV-1111-1-LAB:
14   type: FACILITIES/ROOM
15   id: FACILITIES/2345678
16   connections:
17     - US-MTV-1111-1: CONTAINS
18
19 ▼ EF-1:
20   type: HVAC/FAN_SS_RNC
21   id: 12345
22   cloud_device_id: abc123
23   connections:
24     - US-MTV-1111-1-LAB: FEEDS
25   translation:
26     zone_air_radon_concentration_sensor:
27       present_value: "points.radon_lvl.present_value"
28     units:
29       key: "pointset.points.radon_lvl.units"
30       values:
31         parts_per_million: "PPM"
32     zone_air_radon_concentration_setpoint:
33       present_value: "points.radon_lvl_stpt.present_value"
34     units:
35       key: "pointset.points.radon_lvl_stpt.units"
36       values:
37         parts_per_million: "PPM"
38     run_command:
39       present_value: "points.fan_ss.present_value"
40     states:
41       ON: "true"
42       OFF: "false"
43     run_status:
44       present_value: "points.fan_sts.present_value"
45     states:
46       ON: "true"
47       OFF: "false"
48
```

[Back](#)

Keep this file easily accessible for the next activity.
Click **Next** to complete this activity and move on to defining connections.

[Next](#)



Define links

Continue constructing the building config by defining links between the standard fields of two entity types.

A building config needs to map a reporting device's native payload to concepts in the DBO. When a translation doesn't result in a one-to-one mapping of a reporting entity, a link will need to be defined between the reporting entity and a logical entity.

[Back](#)

Config format

```
ENTITY-NAME:  
...  
links:  
  A-FOURTH-ENTITY: # source device  
    # target device field: source device_field  
    supply air damper position_command:  
supply air damper command 1  
  zone_air_temperature: zone_air_temperature_sensor_1  
...
```

[Next](#)

Define links

Building config

Spaces

Buildings, floors, and rooms won't usually have links defined on them.

Devices

If a link is needed, it is always defined on the target entity. Links are configured by naming the source entity inside the **links** block, then listing the target entity type's standard field that correlates with the source entity type's standard field.

```
ENTITY-NAME:
  type: NAMESPACE/A_DIGITAL_BUILDINGS_ENTITY_TYPE
  ...
  links:
    AN-ENTITY: # source device
               # target device field: source device field
               supply air damper position_command:
supply air damper command 1
               zone_air_temperature: zone_air_temperature_sensor_1
```

Zones and control groups

Zones and control groups are entities, but won't usually have translations or links defined on them.

[Back](#)

Note: A link is a mapping between the standard fields of two entity types. They're used in conjunction with translations to map virtual entities and passthrough entities. Review [Module 1, Lesson 7](#) for more info.

[Next](#)

Define links (continued)

Steps to define a link

1. In the building config file, locate an entity that needs to define links. This is the target entity.
2. Below its defined connections, enter the `links:` block.
3. Enter the name of the source entity.
4. Enter a standard field of the target entity followed by a `:`.
5. After the `:`, enter the correlating standard field of the source entity.
6. Repeat steps 4-5 for each field that needs to be linked.
7. Repeat steps 2-6 for each entity that needs to define links.
8. Save your work.

[Back](#)[Next](#)

Lesson 6

Practice 4



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

- Picking up where you left off, you'll use the "PRACTICE_config.yaml" file you created in this lesson's first practice activity and continue constructing a building config that includes the exhaust fan **EF-1**.
- The next slides will walk through defining a link between a reporting entity and logical entity.
- After this practice activity, you'll move on to generate GUIDs.
- Keep the file you're working on easily accessible for additional practice activities.

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

[Back](#)

[Next](#)

Practice 4

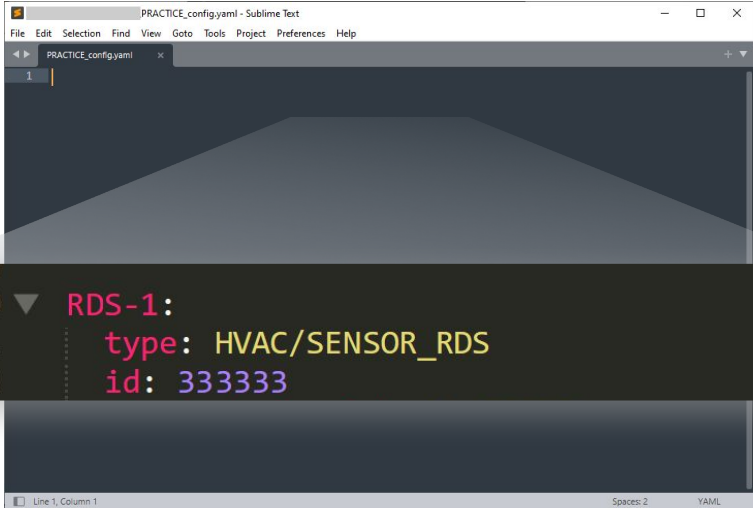
Continuing from the previous practice activity, let's model the radon sensor as a separate entity and link it to the existing exhaust fan.

Occasionally, it can be beneficial to model the sensors themselves as separate entities.

For now, let's assume there exists an entity type for the radon sensor called `HVAC/SENSOR_RDS` and it only requires the field `zone_air_radon_concentration_sensor`. Let's also assume this particular sensor will be called `RDS-1` and that it's located in the lab `US-MTV-1111-1-LAB`.

Return to “PRACTICE_config.yaml” in your text editor and define a new entity for the described radon sensor. Be sure to assign all the necessary information.

Use the information provided above and the proper config format.



```
PRACTICE_config.yaml - Sublime Text
File Edit Selection Find View Goto Tools Project Preferences Help
PRACTICE_config.yaml
1
49
50 ▼ RDS-1:
51     type: HVAC/SENSOR_RDS
52     id: 333333
Line 1, Column 1 Spaces: 2 YAML
```

[Back](#)

When you're ready, click **Next** to continue this practice activity.

[Next](#)

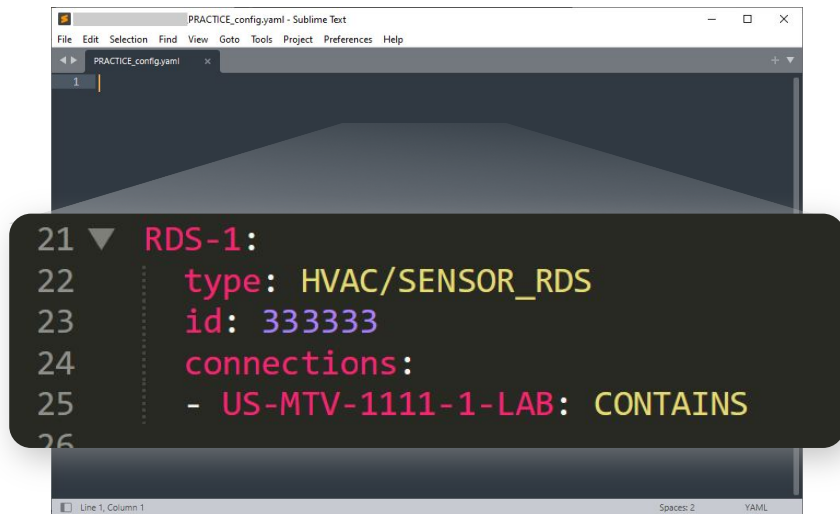
Practice 4 (continued)

Next, let's link the new radon sensor **RDS-1** to the exhaust fan **EF-1**.

In this case, **RDS-1** would be the target entity and **EF-1** would be the source entity. Remember, the field `zone_air_radon_concentration_sensor` is the only one the radon sensor requires.

Return to “**PRACTICE_config.yaml**” in your text editor and define a link between the standard fields of **RDS-1** and **EF-1**.

Use the information provided above and the proper config format.



```
PRACTICE_config.yaml - Sublime Text
File Edit Selection Find View Goto Tools Project Preferences Help
PRACTICE_config.yaml x
1
21 ▼ RDS-1:
22     type: HVAC/SENSOR_RDS
23     id: 333333
24     connections:
25     - US-MTV-1111-1-LAB: CONTAINS
26
Line 1, Column 1 Spaces: 2 YAML
```

[Back](#)

When you're ready, click **Next** to check your work.

[Next](#)

Check your work!

This is what our building config file looks like after defining a link on the target entity **RDS-1** to the source entity **EF-1**.

Did you end up with something similar?



```
4
5 EF-1:
6   type: HVAC/FAN_SS_RNC
7   id: 12345
8   cloud_device_id: abc123
9   connections:
10    - US-MTV-1111-1-LAB: FEEDS
11    - US-MTV-1111-1-LAB: CONTAINS
12   translation:
13     run_command:
14       present_value: "points.fan_ss.present_value"
15     states:
16       ON: "true"
17       OFF: "false"
18     run_status:
19       present_value: "points.fan_sts.present_value"
20     states:
21       OFF: "false"
22       ON: "true"
23   zone_air_radon_concentration_sensor:
24     present_value: "points.radon_lvl.present_value"
25     units:
26       key: "points.radon_lvl.units"
27       values:
28         parts_per_million: "PPM"
29   zone_air_radon_concentration_setpoint:
30     present_value: "points.radon_lvl_stpt.present_value"
31     units:
32       key: "points.radon_lvl_stpt.units"
33       values:
34         parts_per_million: "PPM"
35
36 RDS-1:
37   type: HVAC/SENSOR_RDS
38   id: 333333
39   connections:
40    - US-MTV-1111-1-LAB: CONTAINS
41   links:
42     EF-1:
43       zone_air_radon_concentration_sensor: zone_air_radon_concentration_sensor
44
```

[Back](#)

Keep this file easily accessible for the next activity.
Click **Next** to complete this activity and move on to generating GUIDs.

[Next](#)



Generate GUIDs

Finalize the building config file by generating GUIDs for its included entities.

You can do this using the GUID Generator. To get started, simply install the [GUID Generator](#) and [Digital Buildings toolkit](#) from the Digital Buildings Project GitHub repo and run it from your machine.

[Back](#)

Config format

```
ENTITY-NAME:
  type: NAMESPACE/A_DIGITAL_BUILDINGS_ENTITY_TYPE
  id: SYSTEM/ID1234
  guid: 1234-5678-9012-3456
...
ANOTHER-ENTITY:
  type: NAMESPACE/A_DIGITAL_BUILDINGS_ENTITY_TYPE
  id: SYSTEM/ID2345
  guid: 2345-6789-0123-4567
...
A-THIRD-ENTITY:
  type: NAMESPACE/A_DIGITAL_BUILDINGS_ENTITY_TYPE
  id: SYSTEM/ID3456
  guid: 3456-7890-1234-5678
...
A-FOURTH-ENTITY:
  type: NAMESPACE/A_DIGITAL_BUILDINGS_ENTITY_TYPE
  id: SYSTEM/ID4567
  guid: 4567-8901-2345-6789
...
```

[Next](#)

Generate GUIDs

Steps to generate GUIDs

1. Open your terminal.
2. Run GUID Generator via the DB toolkit using the command: `python toolkit.py input --input path/to/YOUR_BUILDING_CONFIG.yaml --generate`
3. Return to your building config file and see the GUIDs have been automatically appended to the defined entities.

[Back](#)

Note: If you have already populated the GUID for an entity, the generator will not add a new one. This gives you the ability to add them as you go, without fear of overwriting your previous work.

[Next](#)

Lesson 6

Practice 5



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

- Picking up where you left off, you'll use the "PRACTICE_config.yaml" file you created in this lesson's first practice activity and finalize construction of a building config that includes the exhaust fan **EF-1**.
- The next slides will walk through generating GUIDs for the entities defined in the building config.
- If you haven't done so already, install the [GUID Generator](#) and [Digital Buildings toolkit](#) on your machine.
- After this practice activity, you'll wrap up Lesson 6.

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

Next

Practice 5

Continuing from the previous practice activity, let's generate GUIDs for the entities defined in your practice building config file.

So far, all of the entities in "PRACTICE_config.yaml" have not had a GUID added to them yet.

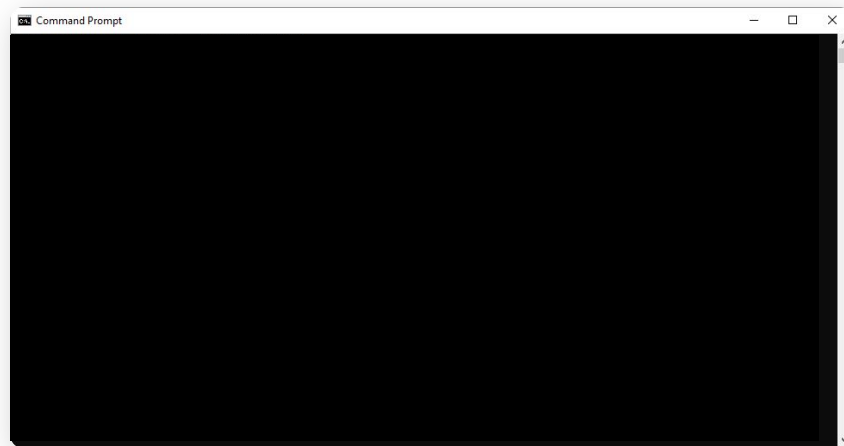
Generate GUIDs using the GUID Generator.

Follow the steps to displayed below.

Steps

1. Open your terminal.
2. Run GUID Generator via the DB toolkit using the command:

```
python toolkit.py input --input  
path/to/YOUR_BUILDING_CONFIG.yaml --generate
```
3. Return to your building config file and see the GUIDs have been automatically appended to the defined entities.



Note: In order to generate GUIDs for this practice activity, you must have the [GUID Generator](#) and [Digital Buildings toolkit](#) already installed on your machine.

Back

When you're ready, click **Next** to check your work.

Next

Check your work!



This is what **EF-1** looks like after generating a GUID for it. All other entities in our building should also have a GUID now.

Did you end up with something similar?

Note that running the GUID Generator will also append some config metadata. This can be deleted or ignored. It won't affect subsequent validations.



```
EF-1:
type: HVAC/FAN_SS
id: 12345
cloud_device_id: 1234567
translation:
  run_command:
    present_value: points.fan_ss.present_value
  states:
    ON: true
    OFF: false
  run_status:
    present_value: points.fan_sts.present_value
    states:
      OFF: false
      ON: true
guid: 474a4afb-e753-4443-90f7-21612f305cb8
```

[Back](#)

Click **Next** to complete this activity and wrap up this lesson.

[Next](#)

Repeat as needed

To construct and finalize a building configuration file, you'll repeat these steps as needed to map real-world devices to the Digital Buildings Ontology (DBO).

Click on each item to review the step-by-step instructions.

Create a new building config file

Define translations

Define entities

Define connections

Define links

Generate GUIDs



[Back](#)

[Next](#)

Repeat as needed

To construct and finalize a building configuration file, you'll repeat these steps as needed to map real-world devices to the Digital Buildings Ontology (DBO).

Click on each item to review the step-by-step instructions.

Create a new building config file

Define translations

Define entities

Define connections

Define links

Generate GUIDs

Steps to create a new building config file

1. Create a new file in your preferred text editor.
2. Add any reporting entities that may have already been defined in your rough-in model or from an ontology extension.
 - **Tip:** To get started without any known entities, you can copy the [config format](#) and paste into the text editor.
3. Save as a .yaml file.

[Back](#)

[Next](#)

Repeat as needed

To construct and finalize a building configuration file, you'll repeat these steps as needed to map real-world devices to the Digital Buildings Ontology (DBO).

Click on each item to review the step-by-step instructions.

Create a new building config file

Define translations

Define entities

Define connections

Define links

Generate GUIDs

Steps to define translations

1. Return to your building config file in your preferred text editor.
2. Locate reporting entities for the reporting devices that need translations.
3. Below any defined connections and/or links, enter the **translation:** block.
4. Within the **translation:** block, list the required fields of the reporting entity's entity type. Also list any optional fields that need to be mapped.
5. For each required field, qualify the path to the payload.
 - Confirm there is a point from the payload that corresponds with the field.
 - If there isn't, enter **MISSING** and Step 5 for each required field.
 - Enter a **present_value:** line and qualify the point path using this format: `points.name_of_point.present_value`.
 - Determine whether the point is a dimensional number or a multi-state.
 - If the point is a dimensional number, enter the **units:** block to define the dimensional unit for the point. Within this block:
 - Enter the **key:** line and qualify the point path using this format: `points.name_of_point.units`.
 - Enter the **values:** line and map the standard unit to the point's unit using this format: `standard_unit: "pointUNIT"`.
 - Refer to [units.yaml](#) for all standard dimensional units.
 - If the point is a multi-state, enter the **states:** block to define the states for the point. Within this block:
 - Enter each standard state followed by a `:`. Refer to [states.yaml](#) for all standard states.
 - After each state, enter the point's state enclosed in `" "`.
6. Repeat Step 5 for each required field that needs to be translated.
7. Enter the `cloud_device_ID:` line and retrieve the reporting device's Numeric ID from the Cloud IoT Registry.
8. Save your work.

Back

Next

Repeat as needed

To construct and finalize a building configuration file, you'll repeat these steps as needed to map real-world devices to the Digital Buildings Ontology (DBO).

Click on each item to review the step-by-step instructions.

Create a new building config file

Define translations

Define entities

Define connections

Define links

Generate GUIDs

Steps to define entities

1. Review the JSON payload to identify the `device_id` for spaces (building, floors, and rooms), devices, zones, or control groups that need to be modeled. This is the entity name.
 - **Note:** Try using a JSON formatter like [go/jsonformatter](#) to convert the payload into a more readable format.
2. In the building config file, enter the `device_id` as the entity name.
3. Enter the `type:` line to identify the entity type and properly qualify its namespace.
 - For spaces, refer to [Facilities.yaml](#).
 - For devices, refer to your rough-in and the global and child namespaces in [digitalbuildings / ontology / yaml / resources](#).
4. Enter lines for the `id:` and `guid:`. These will be generated later.
5. Repeat Steps 1-4 for each required entity.
6. Save your work.

[Back](#)

[Next](#)

Repeat as needed

To construct and finalize a building configuration file, you'll repeat these steps as needed to map real-world devices to the Digital Buildings Ontology (DBO).

Click on each item to review the step-by-step instructions.

Create a new building config file

Define translations

Define entities

Define connections

Define links

Generate GUIDs

Steps to define connections

1. In the building config file, locate the entities you previously defined for spaces (building, floors, and rooms), devices, zones, and control groups that need to be modeled.
2. For each entity, enter the **connections:** block below the **id:** and define the connection type.
 - Buildings: Connections aren't defined on the building entity.
 - Floors: A building should be defined on the target floor entity using a **CONTAINS** connection.
 - Room: A floor should be defined on the target room entity using a **CONTAINS** connection.
 - Devices connected to a space: A floor should be defined on the target device entity using a **CONTAINS** connection.
 - Devices connected to another device: Other devices, zones, or control groups should be defined on the target device entity using the most appropriate connection type. Refer to [connections.yaml](#) for all connection types.
3. Save your work.

Back

Next

Repeat as needed

To construct and finalize a building configuration file, you'll repeat these steps as needed to map real-world devices to the Digital Buildings Ontology (DBO).

Click on each item to review the step-by-step instructions.

Create a new building config file

Define translations

Define entities

Define connections

Define links

Generate GUIDs

Steps to define links

1. In the building config file, locate an entity that needs to define links. This is the target entity.
2. Below its defined connections, enter the **links:** block.
3. Enter the name of the source entity.
4. Enter a standard field of the target entity followed by a **:**.
5. After the **:**, enter the correlating standard field of the source entity.
6. Repeat steps 4-5 for each field that needs to be linked.
7. Repeat steps 2-6 for each entity that needs to define links.
8. Save your work.

[Back](#)

[Next](#)

Repeat as needed

To construct and finalize a building configuration file, you'll repeat these steps as needed to map real-world devices to the Digital Buildings Ontology (DBO).

Click on each item to review the step-by-step instructions.

Create a new building config file

Define translations

Define entities

Define connections

Define links

Generate GUIDs

Steps to generate GUIDs

1. Open your terminal.
2. Run GUID Generator via the DB toolkit using the command: `python toolkit.py input --input path/to/YOUR_BUILDING_CONFIG.yaml --generate`
3. Return to your building config file and see the GUIDs have been automatically appended to the defined entities.

[Back](#)

[Next](#)

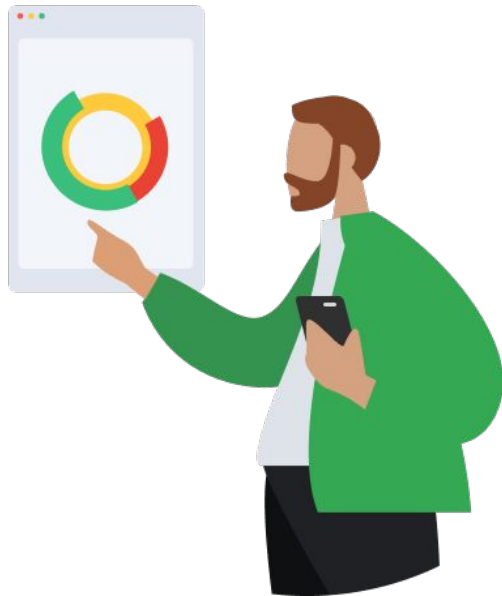
Lesson 6 summary

Let's review what you learned about:

- The contents and format of building configuration files
- Constructing a building config file
- Finalizing a building config file

Now you should be able to:

- Recognize the contents and format of a building config file.
- Construct a building config using the configuration format to define:
 - Translations for devices.
 - Entities for spaces, zones, and control groups.
 - Connections for spaces, devices, zones, and control groups.
 - Links for devices.
- Finalize a building config by generating GUIDs.



[Back](#)

[Next](#)

You completed Lesson 6!

Now's a great time to take a quick break before starting Lesson 7.

Ready for Lesson 7?

Let's go!

Back

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

Have questions?

For future reference, keep these contacts and resources easily accessible for technical and procedural questions.

Key contacts

- For DBO questions: Trevor (tsodorff@) or Charbel (charbelk@)
- For UDMI: udmi-discuss@

Key resources

Bookmark these resources for future reference.

- [Digital Buildings Project GitHub](#)
Contains source code, tooling, and documentation for the DBO.
- [digitalbuildings / ontology / docs / building_config.md](#)
Describes the building configuration format for mapping concrete assets.
- [GUID Generator](#) and [Digital Buildings toolkit](#)
Used to generate and assign GUIDs to entities in a building config file.