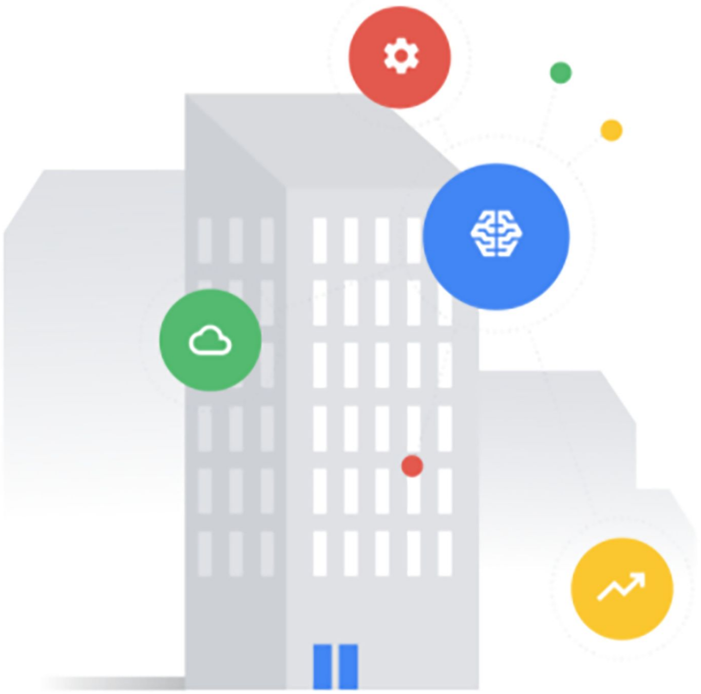
Google

Module 1 | **Lesson 7**

Digital Buildings Ontology (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!

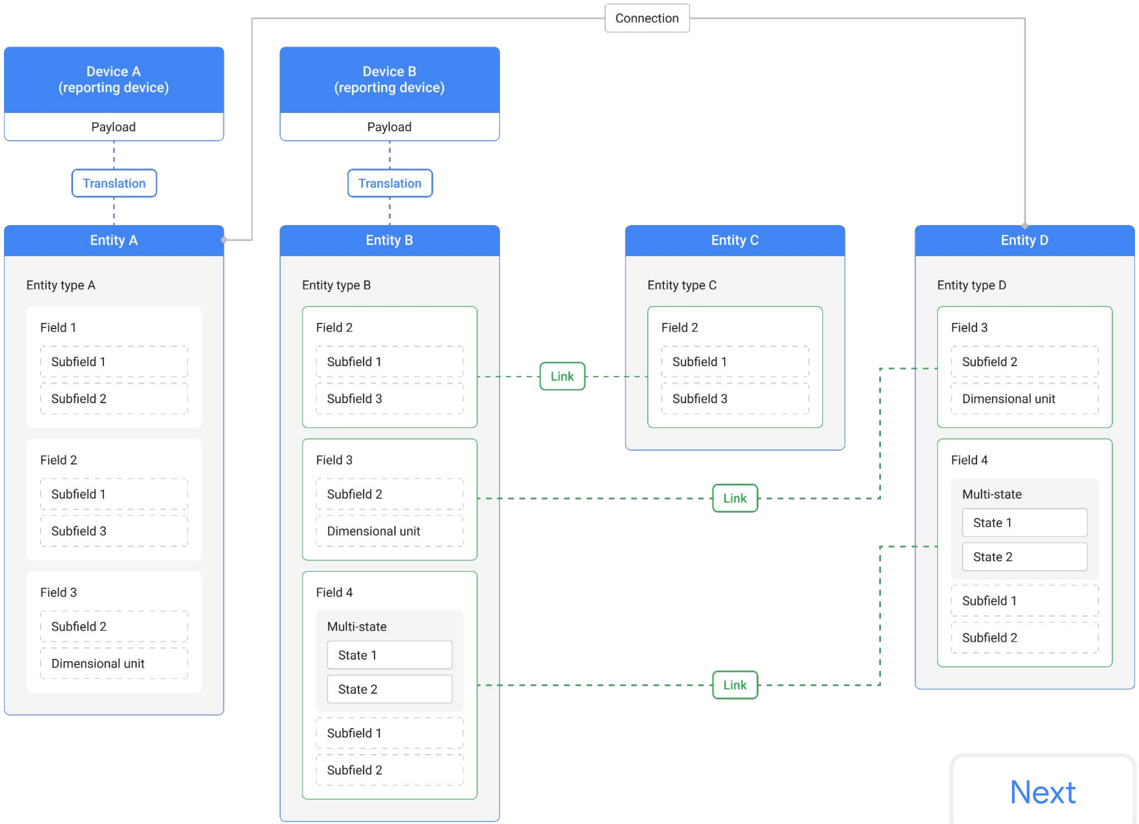
Conceptual model revisited

Here's another look at the DBO conceptual model from Lesson 2.

In this lesson, you'll explore one modeling concept from the abstract model. Remember, the following modeling concepts are used to describe the relationships that can occur between entities:

- Mappings
  - Translations
  - Links
- Connections

Do you see these concepts in the diagram?



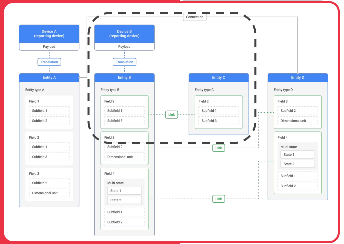
Back

Next

Google

Lesson 7.1

Mappings

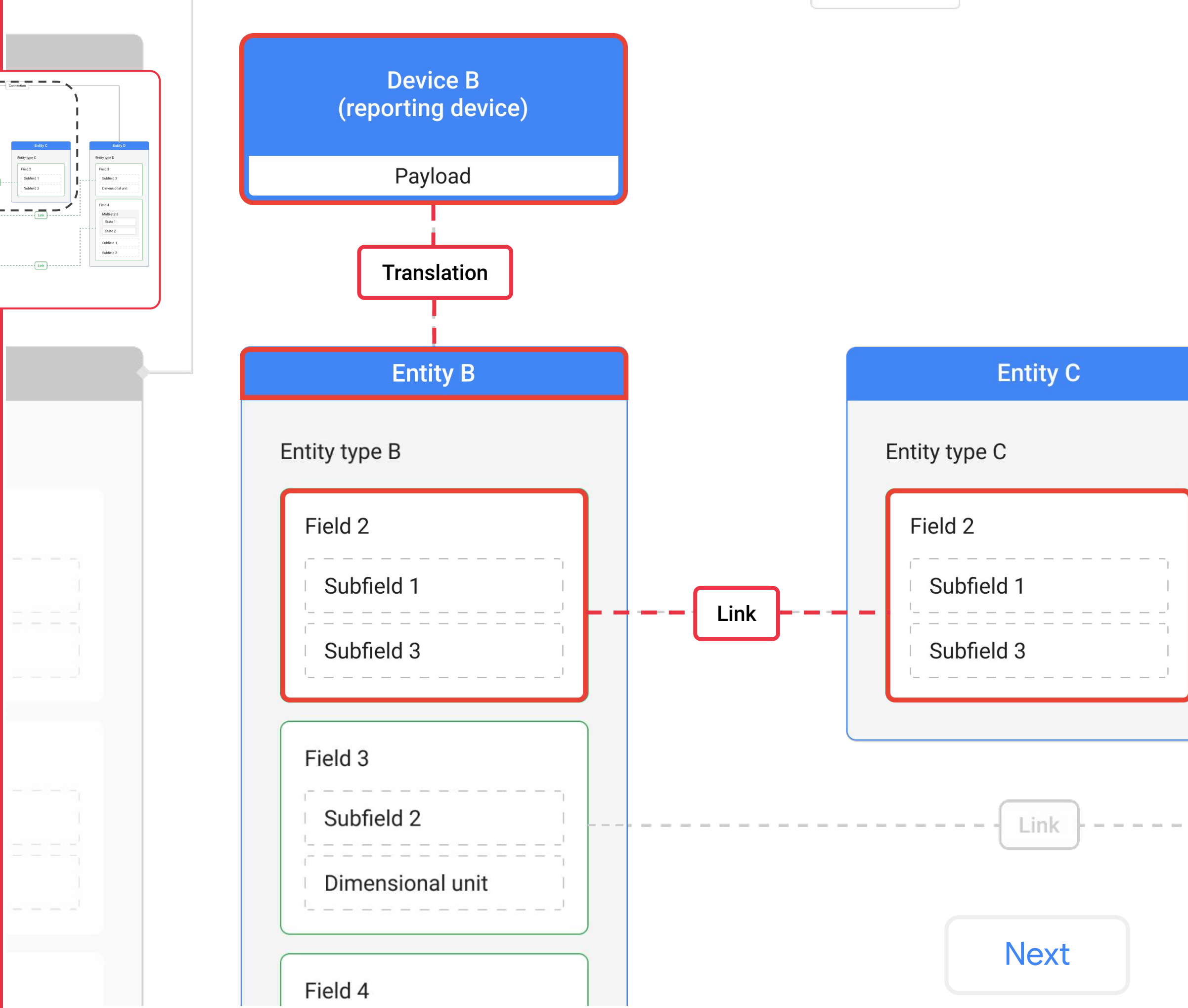


What you'll learn about:

- Mappings
- Logical entities

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

- Describe the purpose of mapping.
- Recognize different types of mappings.
- Describe what is a logical entity.



Back

Next

Google

What's a mapping?

A mapping shows how a payload should be interpreted by associating the equipment installed in a building with the modeling concepts in the DBO.

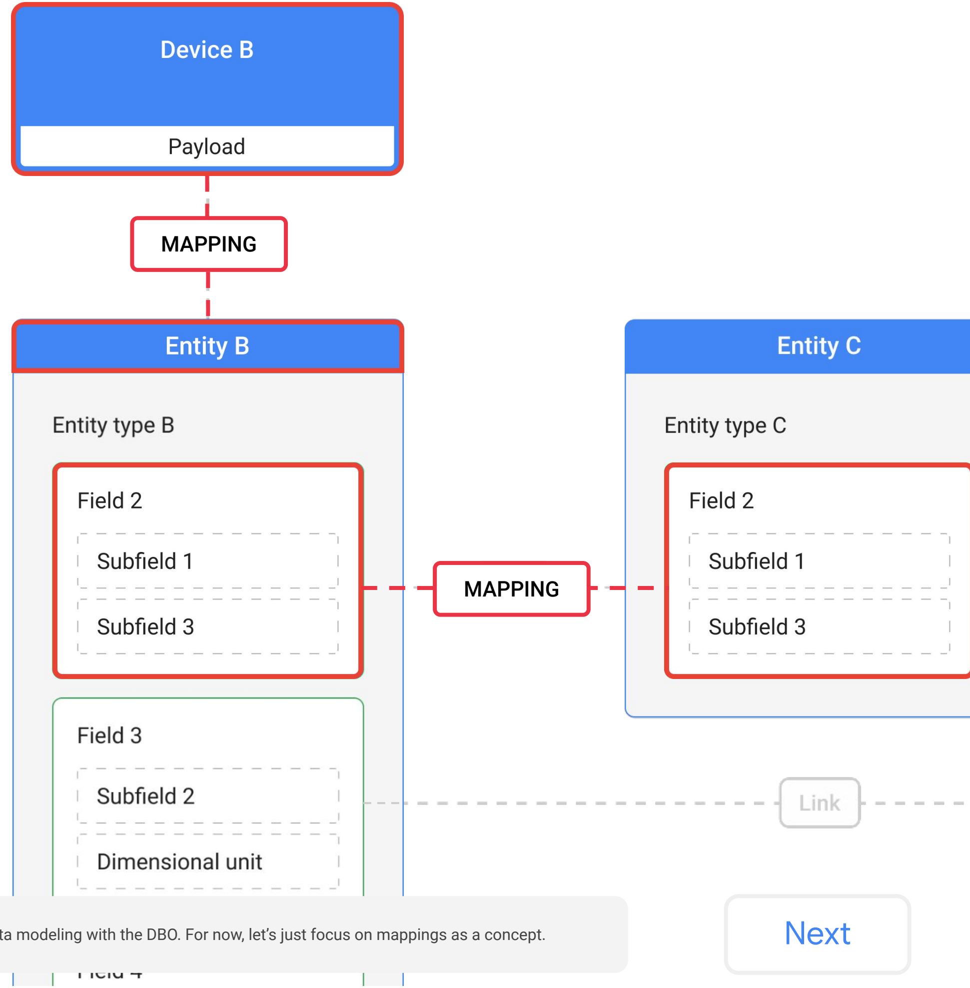
Logical entities

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

You'll always focus on logical entities when mapping devices or systems to the DBO. Simply put, logical entities are the things we care to model, such as fans or meters.

Mappings can be applied in different ways depending on what we care to model.

- Mappings can associate or translate the data from a device's payload to an entity (i.e., a concrete modeling concept) and the fields of its entity type (i.e., an abstract modeling concept).
- Mappings can also be used to pass or link data from one entity and the fields of its entity type to another.



Back

Note: You'll learn more about the application of mappings in Module 3: Data modeling with the DBO. For now, let's just focus on mappings as a concept.

Next

Google

Types of mappings

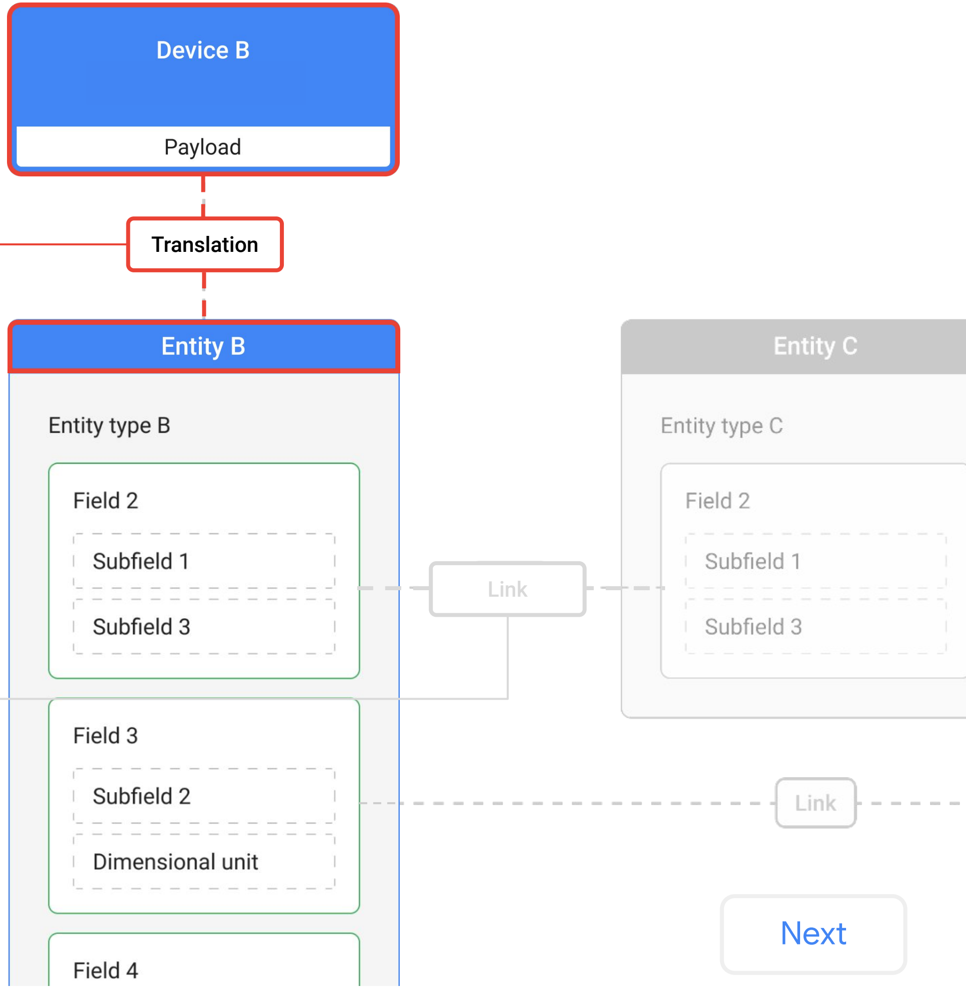
There are two types of mappings in the DBO that are used to associate building equipment to abstract and canonical concepts.

Translations

A translation is a mapping between a device in the real world and its corresponding concrete and abstract concepts in the DBO. Translations convert the information contained in the device's native data payload into the DBO format.

Links

A link is a mapping between the standard fields of two entities to pass data between them. Links are used in conjunction with translations whenever a device's native payload can't be mapped one-to-one with a single abstract concept in the DBO.



Back

Next

Google

Types of mappings

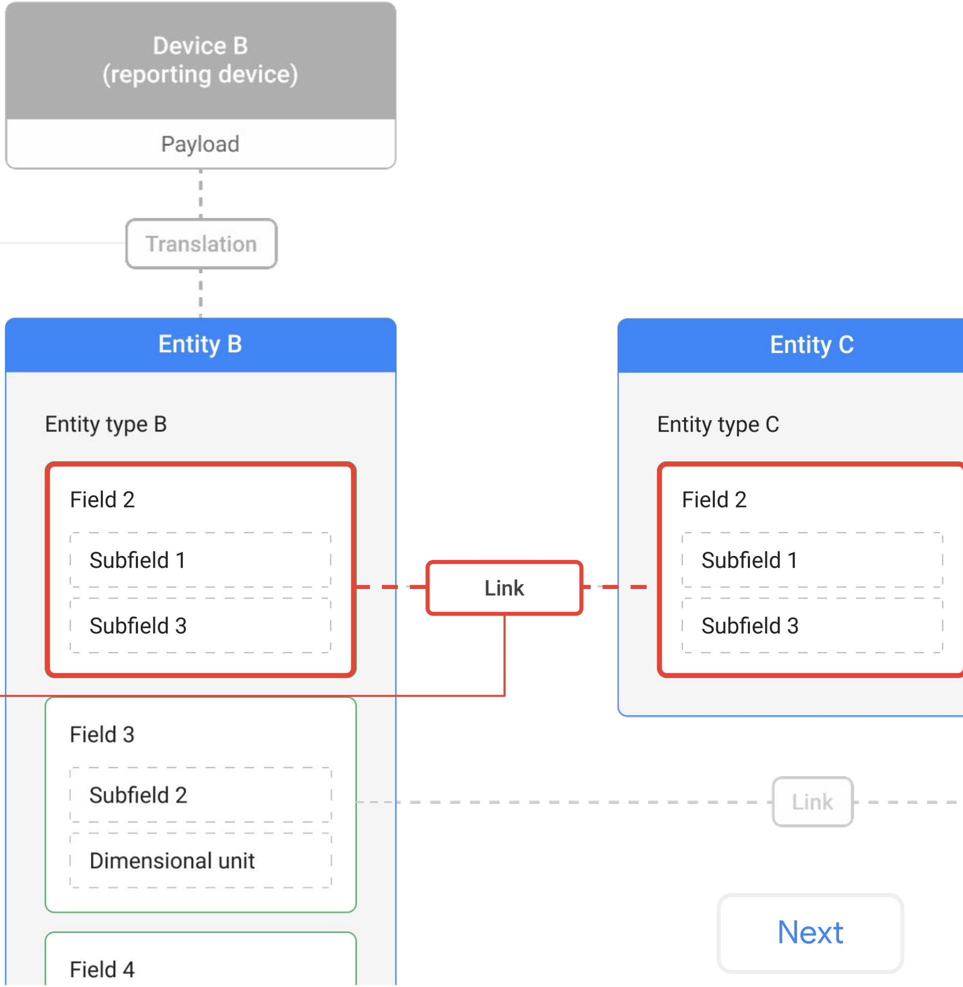
There are two types of mappings in the DBO that are used to associate building equipment to abstract and canonical concepts.

Translations

A translation is a mapping between a device in the real world and its corresponding concrete and abstract concepts in the DBO. Translations convert the information contained in the device's native data payload into the DBO format.

Links

A link is a mapping between the standard fields of two entities to pass data between them. Links are used in conjunction with translations whenever a device's native payload cannot be mapped one-to-one with a single abstract concept in the DBO.



Back

Next

Google

Lesson 7.2

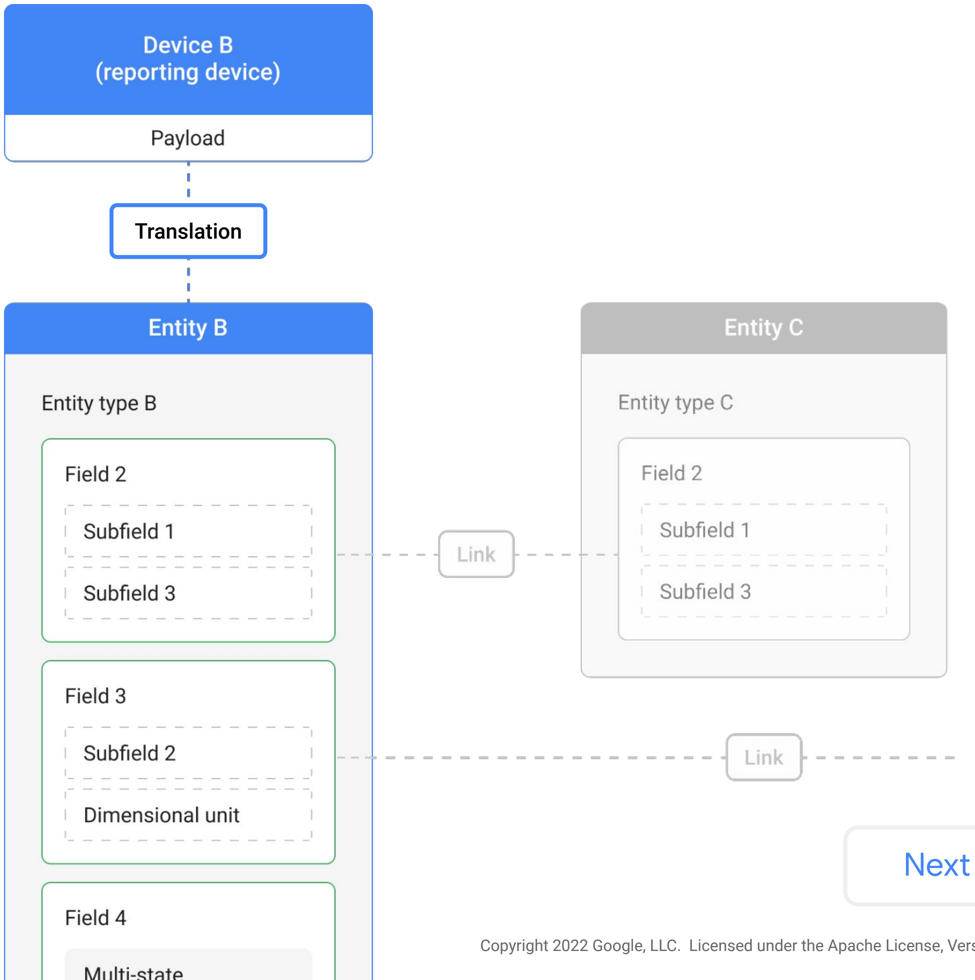
Translations

What you'll learn about:

- Translation configuration
- Reporting devices
- Reporting entities
- Point mapping

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

- Recognize the correct configuration of a translation.
- Understand the relationship between reporting devices, reporting entities, and translations.
- Describe the need for point mapping.



Back

Next

Google

Translations

A **translation** is a mapping between a device in the real world and its corresponding concepts in the DBO.

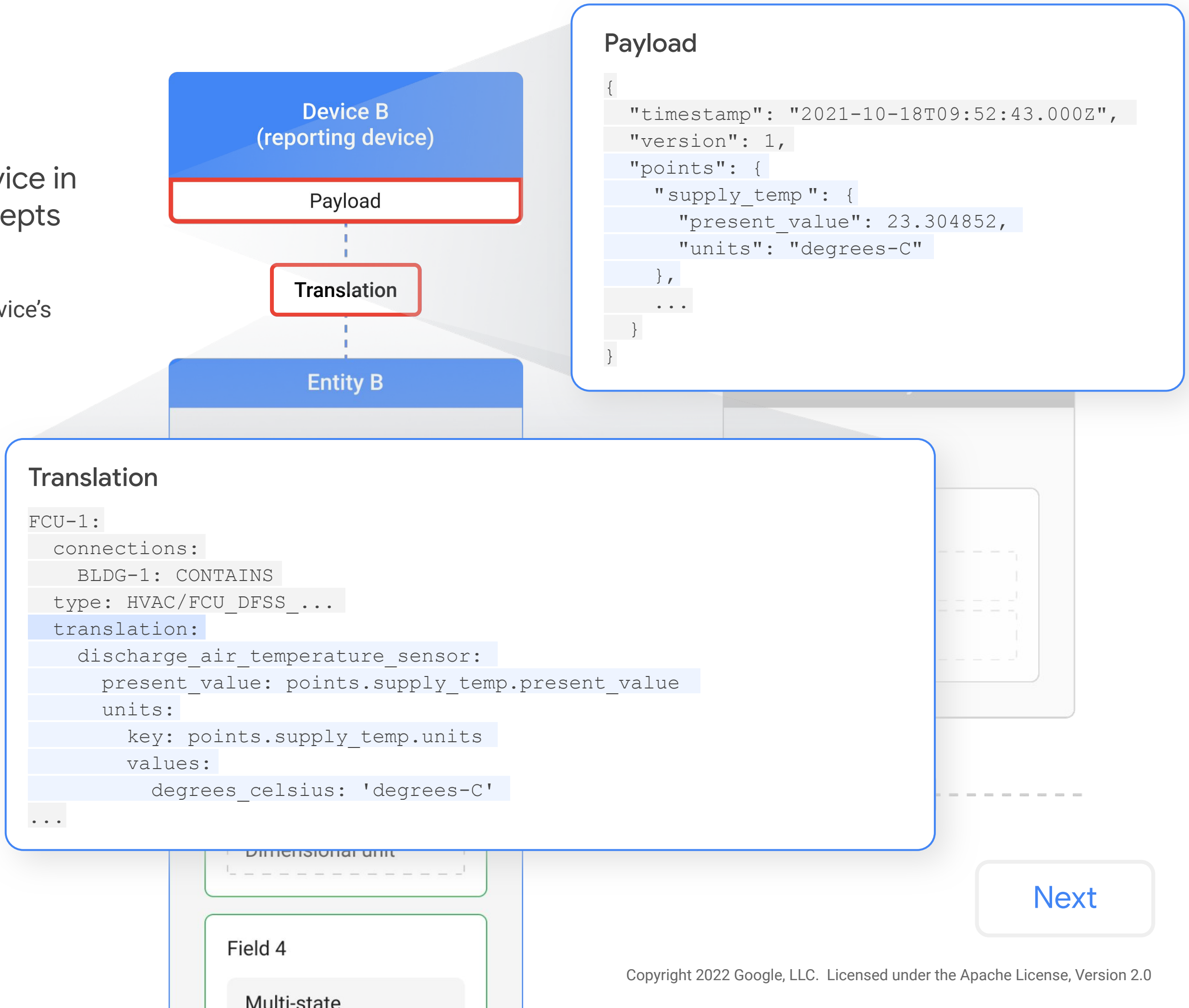
Translations convert the information contained in the device's native data payload into the DBO format.

To the right are samples of an actual payload and its translation.

Do you see the point `supply_temp` in the payload sample?

How about in the translation sample?

> **Note:** The translation sample is an abridged version of the full code. The `...` indicates abbreviations in the code. Moving forward, you'll notice most samples in this lesson are shortened using `...`.



Back

Next

Google

Translation configuration

Let's break down this translation sample.

First, you'll see the name of an entity.

This is the reporting entity, which is the concrete modeling concept that represents the reporting device that sends data.

```
FCU-1:
  connections:
    BLDG-1: CONTAINS
  type: HVAC/FCU_DFSS_...
  translation:
    discharge_air_temperature_sensor:
      present_value: points.supply_temp.present_value
      units:
        key: points.supply_temp.units
        values:
          degrees_celsius: 'degrees-C'
...
```

Back

Next

Google

Translation configuration (continued)

Let's break down this translation sample.

Next, you'll notice a few abstract modeling concepts.

There's the entity type, which represents all of the abstract concepts that describe the named reporting entity.

There are fields listed inside the **`translation`** block. These are fields from the DBO that map to specific points from the device's native payload.

```
FCU-1:
  connections:
    BLDG-1: CONTAINS
  type: HVAC/FCU_DFSS_...
  translation:
    discharge_air_temperature_sensor:
      present_value: points.supply_temp.present_value
      units:
        key: points.supply_temp.units
        values:
          degrees_celsius: 'degrees-C'
...
```

Back

Next

Translation configuration (continued)

Let's break down this translation sample.

Don't mistake some things for a field inside the field block. Here's the DBO standard field.

Highlighted inside the field block are specific points from the device's native payload. These points are what's converted into the DBO format using a translation.

That means in this sample, the point `supply_temp` is translated to the field `discharge_air_temperature_sensor`.

```
FCU-1:
  connections:
    BLDG-1: CONTAINS
  type: HVAC/FCU_DFSS_...
  translation:
    discharge_air_temperature_sensor:
      present_value: points.supply_temp.present_value
      units:
        key: points.supply_temp.units
        values:
          degrees_celsius: 'degrees-C'
...
```

Payload


```
{
  "timestamp": "2021-10-18T09:52:43.000Z",
  "version": 1,
  "points": {
    "supply_temp": {
      "present_value": 23.304852,
      "units": "degrees-C"
    },
    ...
  }
}
```


Back

Next

Translation configuration (continued)

Let's break down this translation sample.

Also inside the field block is information about the field's specific dimensional units or states. In this sample, only units are specified because it isn't multi-state.

Finally, notice the . separator used to indicate the access path to the nested keys in the JSON payload.

```
FCU-1:
  connections:
    BLDG-1: CONTAINS
  type: HVAC/FCU_DFSS_...
  translation:
    discharge_air_temperature_sensor:
      present_value: points.supply_temp.present_value
      units:
        key: points.supply_temp.units
        values:
          degrees_celsius: 'degrees-C'
...
```

Payload


```
{
  "timestamp": "2021-10-18T09:52:43.000Z",
  "version": 1,
  "points": {
    "supply_temp": {
      "present_value": 23.304852,
      "units": "degrees-C"
    },
    ...
  }
}
```


Back

Next

Google

Mapping with translations

A **translation** maps the native payload of a **reporting device** to the standard fields of a **reporting entity**.

What's a reporting device?

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

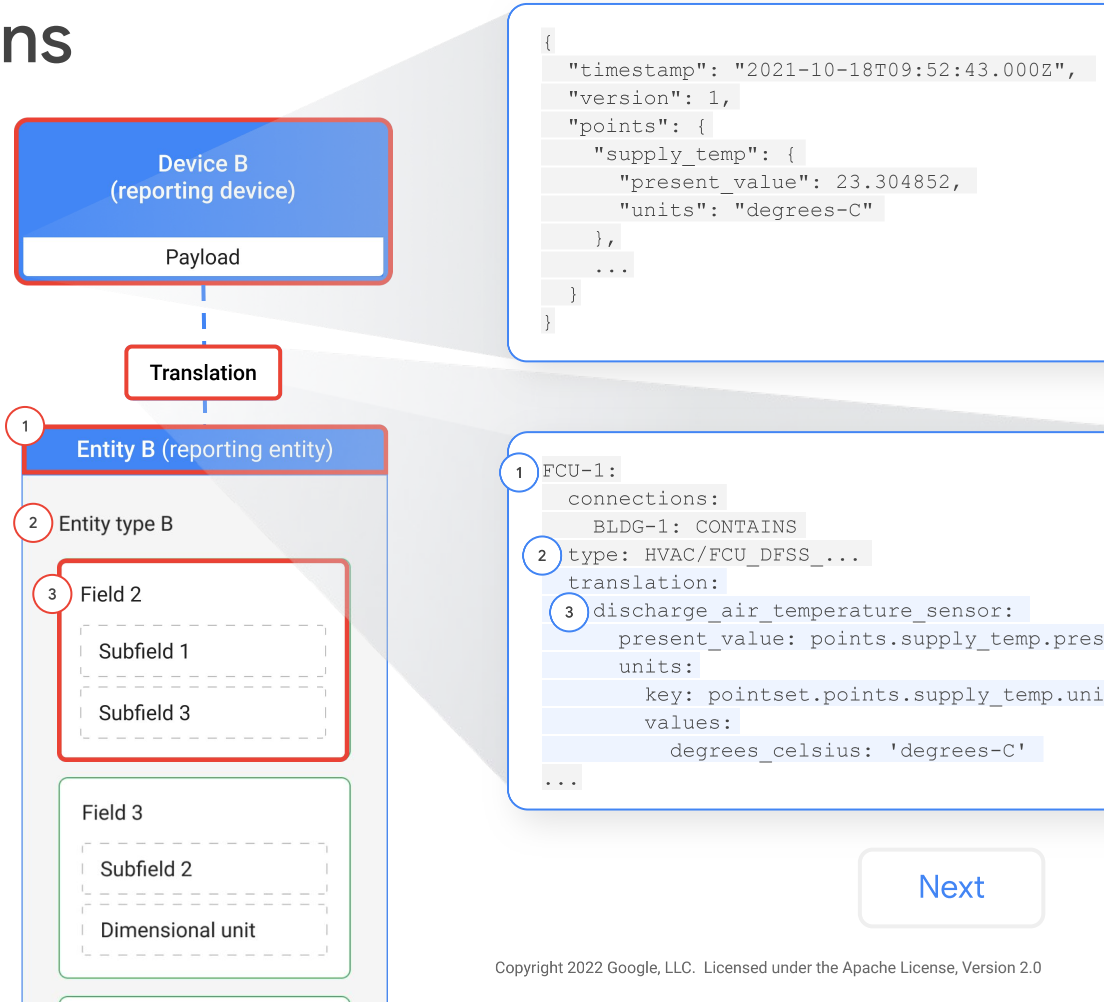
Some examples include:

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

What's a reporting entity?

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

Like all entities, a reporting entity has an entity type, which groups fields and other abstract modeling concepts that describe its properties.



Back

Next

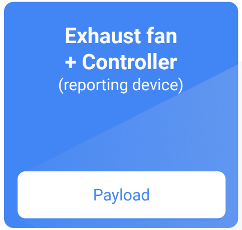
Mapping with translations (continued)

A **translation** maps the native payload of a **reporting device** to the standard fields of a **reporting entity**.

Example

Let’s say you have an exhaust fan with a controller that you want to model.

The controller sends the fan’s native payload to the cloud, which makes it a reporting device. However, the cloud does not recognize its native payload.



Payload


```
{
  "timestamp": "2021-10-18T09:52:43.000Z",
  "points": {
    "fan_speed": {
      "present_value": 60,
      "units": "hertz"
    },
    "fan_pwr": {
      "present_value": 12.77,
      "units": "kw"
    }
  }
}
```


Back

*Click **Next** to inspect the translation in building config syntax.*

Next

Mapping with translations (continued)

A **translation** maps the native payload of a **reporting device** to the standard fields of a **reporting entity**.

Example (continued)

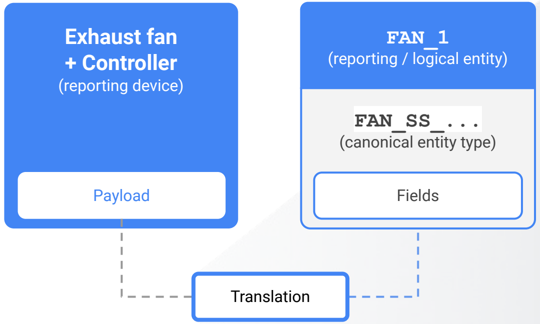
Let's say you have an exhaust fan with a controller that you want to model.

The controller sends the fan's native payload to the cloud, which makes it a reporting device. However, the cloud does not recognize its native payload.

The exhaust fan is a type of `FAN_SS_`..., which is a canonical entity type. We'll name the reporting entity `FAN_1`.

Using a translation, you can map the payload to the reporting entity `FAN_1` and the standard fields of `FAN_SS_`.... Since this is a one-to-one mapping, `FAN_1` is also a logical entity.

Now the exhaust fan is able to send its translated payload, and the cloud can recognize it.



Translation

```
FAN-1:
  type: FAN_SS_…
  translation:
    speed_frequency_sensor:
      present_value: points.fan_speed.present_value
      units:
        key: points.fan_speed.units
        value:
          hertz: 'hertz'
    power_sensor:
      present_value: points.fan_pwr.present_value
      units:
        key: points.fan_pwr.units
        value:
          kilowatts: 'kw'
```

Back

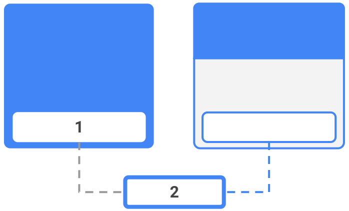
*Click **Next** for a summary of this example.*

Next

Google

Mapping with translations (continued)

A **translation** maps the native payload of a **reporting device** to the standard fields of a **reporting entity**.



Example summary

This mapping requires the following entities:

- One logical entity
- One reporting entity
- One canonical entity

In this particular example, the logical, reporting, and canonical entities are all **`FAN_1`**.

1. Payload

```
{
  "timestamp": "2021-10-18T09:52:43.000Z",
  "points": {
    "fan_speed": {
      "present_value": 60,
      "units": "hertz"
    },
    "fan_pwr": {
      "present_value": 12.77,
      "units": "kw"
    }
  }
}
```

2. Translation

```
FAN-1:
  type: FAN_SS_…
  translation:
    speed_frequency_sensor :
      present_value: points. fan_speed .present_value
      units:
        key: points. fan_speed .units
        value:
          hertz: 'hertz'
    power_sensor :
      present_value: points. fan_pwr .present_value
      units:
        key: points. fan_pwr .units
        value:
          kilowatts: 'kw'
```

Back

Next

Google

Point mapping with translations

A translation “tells” the system exactly which points from the native payload that we care about.

A reporting device can send a lot of points in its payload. However, not every point needs to be mapped.

Google BOS only cares about specific types of data:

- Measured telemetry
- Setpoint telemetry
- Control states

Using translations, you can pinpoint useful data points from a reporting device’s payload to map to the fields of a reporting entity. This allows you to omit unnecessary points.

Example

Let’s say you have a controller that collects data for a fan.

The controller sends all of this data to the cloud in a single payload, which makes it a reporting device. However, BOS does not require the **`status_alarm`** data point.

The fan is a type of **`FAN_SS`**..., which is a canonical entity type. We’ll name the reporting entity **`FAN_3`**.

Using a translation, you can choose to map only the useful points from the payload and omit unnecessary ones like **`status_alarm`**.

Now the fan is able to send its translated payload, and the cloud can recognize it.

Payload


```
{
  "timestamp": "2021-10-18T09:52:43.000Z",
  "points": {
    "fan_speed": {
      "present_value": 60,
      "units": "hertz"
    },
    "status_alarm": {
      "present_value": OFF
    }
  }
}
```


Translation


```
FAN-3:
  type: FAN_SS_…
  translation:
    speed_frequency_sensor:
      present_value: pointsfan_speed.present_value
      units:
        key: points.fan_speed.units
        value:
          hertz: 'hertz'
```


Back

Note: You’ll learn more about required device data and determining what needs to be modeled in Module 3: Data modeling with the DBO.

Next

Lesson 7

Knowledge check 1

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

- The next slide will have a question about translations.
- Review the question and select the correct response.
- After this knowledge check, you'll move on to learn about mapping with links.

You won't be able to move forward until the correct answer is selected.

Back

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

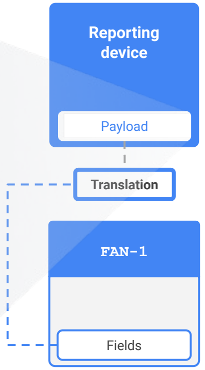
Next

Google

Knowledge check 1

A sample payload is on the right.


```
{
  "timestamp": "2021-10-18T09:52:43.000Z",
  "points": {
    "fan_speed": {
      "present_value": 60,
      "units": "hertz"
    },
    "fan_pwr": {
      "present_value": 12.77,
      "units": "kw"
    }
  }
}
```

Given this payload, which translation is configured correctly?

Select the best answer from the options listed below.

```
FAN-1:
  translation:
    speed_frequency_sensor:
      present_value: points.fan_speed.present_value
      units:
        key: points.fan_speed.units
        value:
          hertz: 'hertz'
    power_sensor:
      present_value: points.fan_pwr.present_value
      units:
        key: points.fan_pwr.units
        value:
          kilowatts: 'kw'
```

```
FAN-1:
  type: FAN_SS_…
  translation:
    speed_frequency_sensor:
      present_value: points.fan_speed.present_value
      units:
        key: points.fan_speed.units
        value:
          hertz: 'hertz'
    power_sensor:
      present_value: points.fan_pwr.present_value
      units:
        key: points.fan_pwr.units
        value:
          kilowatts: 'kw'
```

```
FAN-1:
  type: FAN_SS_…
  translation:
    speed_frequency_sensor:
      present_value: points.fan_pwr.present_value
      units:
        key: points.fan_pwr.units
        value:
          hertz: 'hertz'
    power_sensor:
      present_value: points.fan_speed.present_value
      units:
        key: points.fan_speed.units
        value:
          kilowatts: 'kw'
```

Back

Note: Don't worry! We've already provided the standard fields for you. For this question, pay attention to the configuration of each option based on the information you're given from the sample.

Next

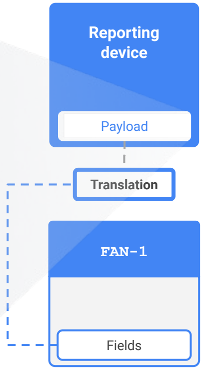
Google

Hmm, that's not right! 🤔

This translation is missing an entity type. Remember, an entity should always have an entity type.

Try again

```
{
  "timestamp": "2021-10-18T09:52:43.000Z",
  "points": {
    "fan_speed": {
      "present_value": 60,
      "units": "hertz"
    },
    "fan_pwr": {
      "present_value": 12.77,
      "units": "kw"
    }
  }
}
```



Given this payload, which translation is configured correctly?

Select the best answer from the options listed below.

```
FAN-1:
  translation:
    speed_frequency_sensor:
      present_value: points.fan_speed.present_value
      units:
        key: points.fan_speed.units
        value:
          hertz: 'hertz'
    power_sensor:
      present_value: points.fan_pwr.present_value
      units:
        key: points.fan_pwr.units
        value:
          kilowatts: 'kw'
```

```
FAN-1:
  type: FAN_SS_…
  translation:
    speed_frequency_sensor:
      present_value: points.fan_speed.present_value
      units:
        key: points.fan_speed.units
        value:
          hertz: 'hertz'
    power_sensor:
      present_value: points.fan_pwr.present_value
      units:
        key: points.fan_pwr.units
        value:
          kilowatts: 'kw'
```

```
FAN-1:
  type: FAN_SS_…
  translation:
    speed_frequency_sensor:
      present_value: points.fan_pwr.present_value
      units:
        key: points.fan_pwr.units
        value:
          hertz: 'hertz'
    power_sensor:
      present_value: points.fan_speed.present_value
      units:
        key: points.fan_speed.units
        value:
          kilowatts: 'kw'
```

Back

Next

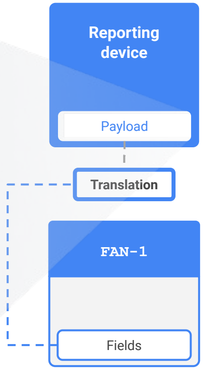
Google

That's right! 🎉

This translation does the following:

- The point **`fan_speed`** translates to the field **`speed_frequency_sensor`**.
- The point **`fan_pwr`** translates to the field **`power_sensor`**.


```
{
  "timestamp": "2021-10-18T09:52:43.000Z",
  "points": {
    "fan_speed": {
      "present_value": 60,
      "units": "hertz"
    },
    "fan_pwr": {
      "present_value": 12.77,
      "units": "kw"
    }
  }
}
```

Given this payload, which translation is configured correctly?

Select the best answer from the options listed below.

```
FAN-1:
  translation:
    speed_frequency_sensor:
      present_value: points.fan_speed.present_value
      units:
        key: points.fan_speed.units
        value:
          hertz: 'hertz'
    power_sensor:
      present_value: points.fan_pwr.present_value
      units:
        key: points.fan_pwr.units
        value:
          kilowatts: 'kw'
```

```
FAN-1:
  type: FAN_SS_…
  translation:
    speed_frequency_sensor:
      present_value: points.fan_speed.present_value
      units:
        key: points.fan_speed.units
        value:
          hertz: 'hertz'
    power_sensor:
      present_value: points.fan_pwr.present_value
      units:
        key: points.fan_pwr.units
        value:
          kilowatts: 'kw'
```

```
FAN-1:
  type: FAN_SS_…
  translation:
    speed_frequency_sensor:
      present_value: points.fan_pwr.present_value
      units:
        key: points.fan_pwr.units
        value:
          hertz: 'hertz'
    power_sensor:
      present_value: points.fan_speed.present_value
      units:
        key: points.fan_speed.units
        value:
          kilowatts: 'kw'
```

Back

*Click **Next** to move on to and learn about mapping with links.*

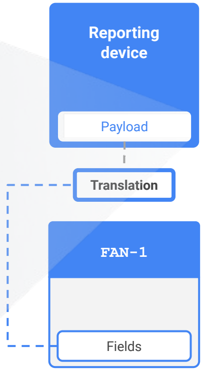
Next

Hmm, that's not right! 🤔

Look closely! It doesn't make much sense to translate the point `fan_pwr` to the field `speed_frequency_sensor`.

Try again


```
{
  "timestamp": "2021-10-18T09:52:43.000Z",
  "points": {
    "fan_speed": {
      "present_value": 60,
      "units": "hertz"
    },
    "fan_pwr": {
      "present_value": 12.77,
      "units": "kw"
    }
  }
}
```

Given this payload, which translation is configured correctly?

Select the best answer from the options listed below.

```
FAN-1:
  translation:
    speed_frequency_sensor:
      present_value: points.fan_speed.present_value
      units:
        key: points.fan_speed.units
        value:
          hertz: 'hertz'
    power_sensor:
      present_value: points.fan_pwr.present_value
      units:
        key: points.fan_pwr.units
        value:
          kilowatts: 'kw'
```

```
FAN-1:
  type: FAN_SS_…
  translation:
    speed_frequency_sensor:
      present_value: points.fan_speed.present_value
      units:
        key: points.fan_speed.units
        value:
          hertz: 'hertz'
    power_sensor:
      present_value: points.fan_pwr.present_value
      units:
        key: points.fan_pwr.units
        value:
          kilowatts: 'kw'
```

```
FAN-1:
  type: FAN_SS_…
  translation:
    speed_frequency_sensor:
      present_value: points.fan_pwr.present_value
      units:
        key: points.fan_pwr.units
        value:
          hertz: 'hertz'
    power_sensor:
      present_value: points.fan_speed.present_value
      units:
        key: points.fan_speed.units
        value:
          kilowatts: 'kw'
```

Back

Next

Google

Lesson 7.3

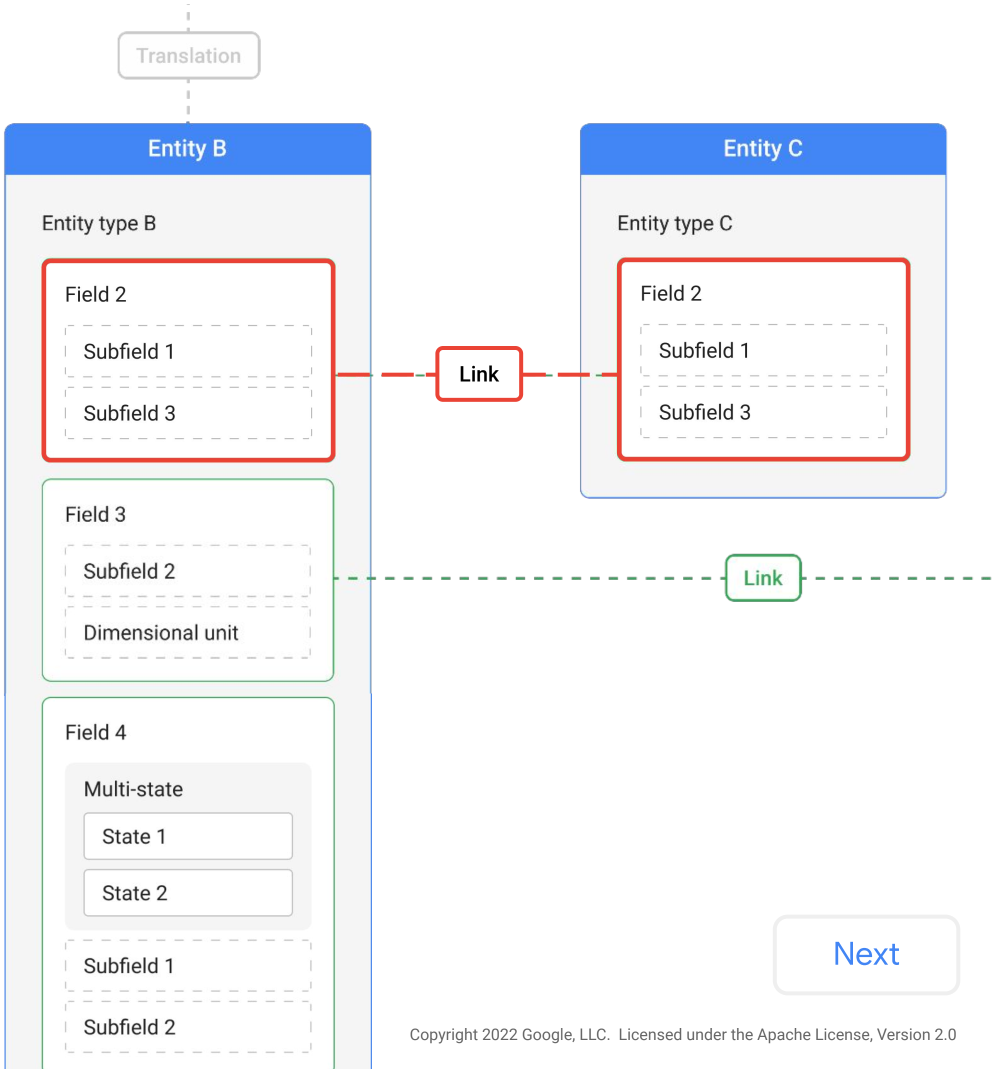
Links

What you'll learn about:

- Link configuration
- Virtual entities
- Passthrough entities

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

- Recognize the correct configuration of a link.
- Determine when a virtual entity is needed.
- Understand the dependencies between links and translations for data modeling.
- Determine when a passthrough entity is needed.



Back

Next

Google

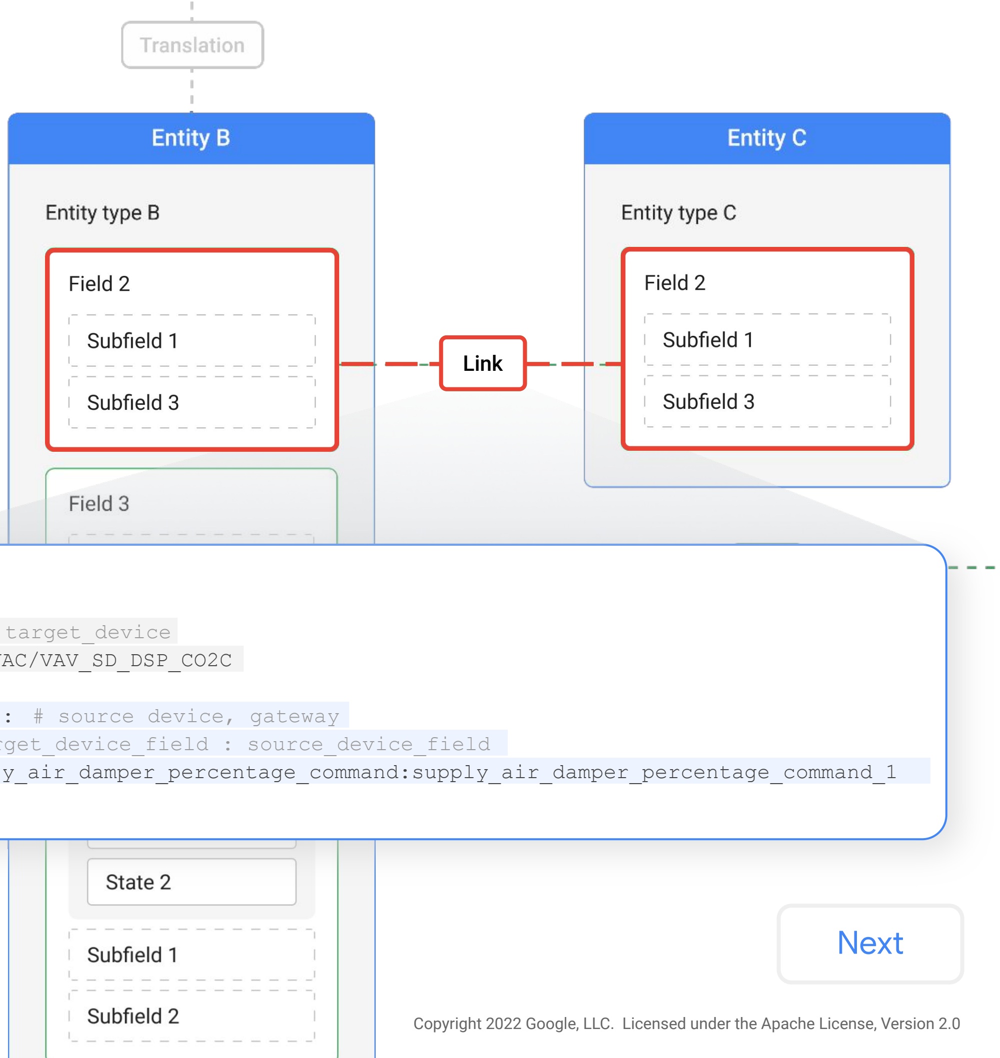
Links

A **link** is a mapping between the standard fields of two entity types.

Links are used together with translations whenever a device's native payload can't be mapped one-to-one with a single abstract concept in the DBO.

To the right is a sample of a link. You'll notice there isn't a sample payload, though.

Can you think of why there isn't a sample payload here?
If not, that's alright! You'll find out in a few slides.



Link

```
VAV-32:  # target_device
  type: HVAC/VAV_SD_DSP_CO2C
  links:
    GTWY-1:  # source device, gateway
     # target_device_field : source_device_field
     supply_air_damper_percentage_command:supply_air_damper_percentage_command_1
...
```

Back

Next

Link configuration

Let's break down this link sample.

First, you'll see the name of an entity. This is the target entity, which defines the link.

Inside the `links` block, you'll see the name of another entity. This is the source entity, which lists its fields that are linked to the target entity.

```
VAV-32: # target_device
  type: HVAC/VAV_SD_DSP_CO2C
  links:
    GTWY-1: # source device, gateway
      # target_device_field : source_device_field
      supply_air_damper_percentage_command: supply_air_damper_percentage_command_1
      ...
```

Back

Note: The concept of source and target entities is also important for connections, which you'll learn about in Lesson 8: Connections.

Next

Link configuration (continued)

Let's break down this link sample.

Next, you'll notice a few abstract modeling concepts.

There's the entity type, which indicates what type of entity is the target.

As an entity type, it represents all of the abstract concepts that describe the named entity.

In the source entity block, you'll see lines with two fields.

```
VAV-32: # target_device
  type: HVAC/VAV_SD_DSP_CO2C
  links:
    GTWY-1: # source device, gateway
      # target_device_field : source_device_field
      supply_air_damper_percentage_command:supply_air_damper_percentage_command_1
      ...
```

The first field is from the target entity.

The second field is from the source entity.

Back

Note: Compared to a translation, you don't see points in the link configuration. This is because a link only maps fields, and a field by itself does not deal with the payload of a device. A field needs to be mapped to a device's payload and its specific points with a translation.

Next

Mapping with links

Linking the standard fields of one entity to the standard fields of another entity results in a **virtual entity**.

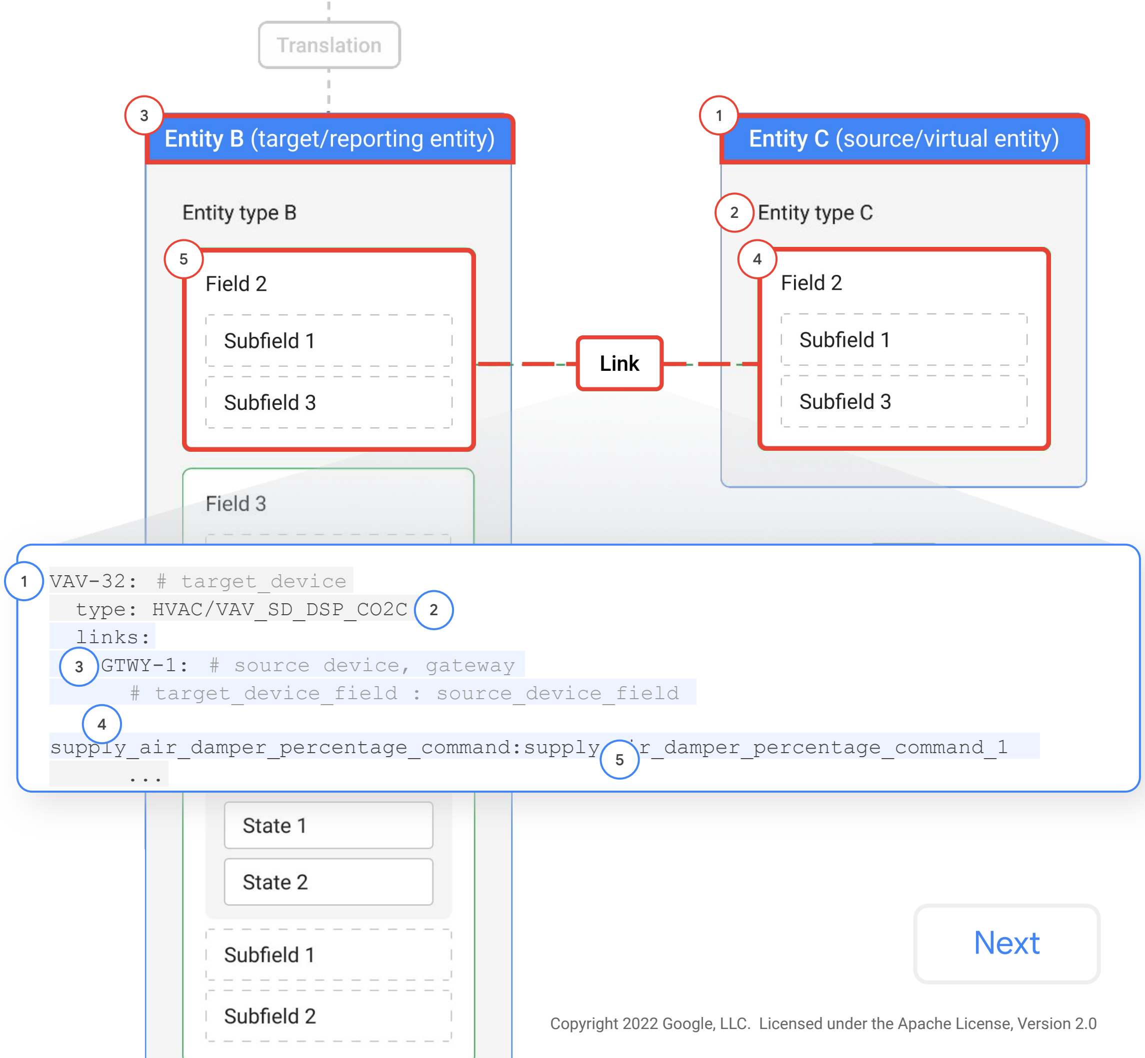
What's a virtual entity?

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.

Why are virtual entities needed?

Virtual entities are needed any time a reporting device can't map one-to-one with a single logical entity. Some use cases include:

- You want to include data from one reporting device in multiple logical entities.
- You want to model a reporting device with multiple components that send their payload via the same network controller.



Back

Next

Mapping with links and translations

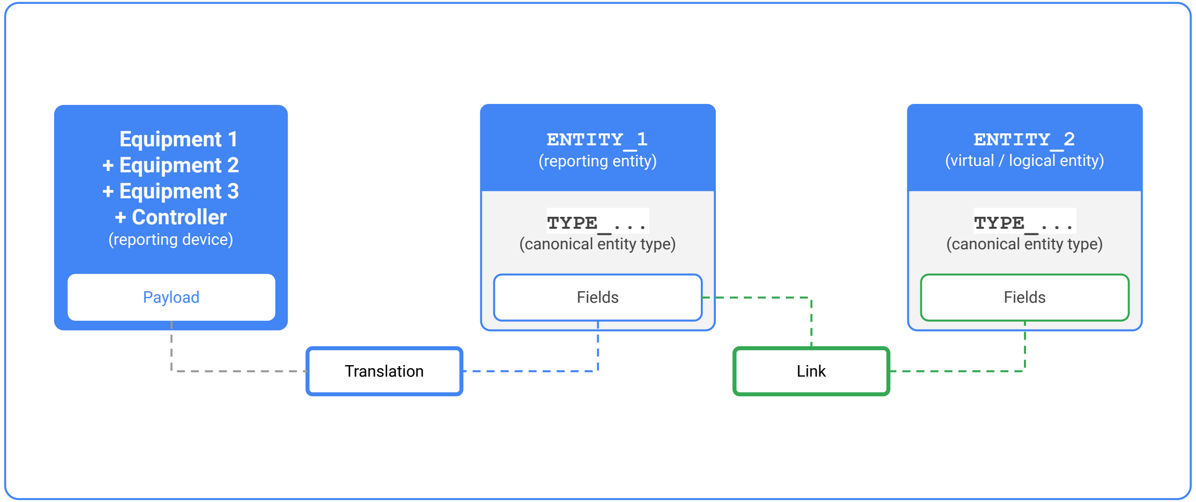
Links are used in conjunction with **translations** to map the payload of a **reporting device** to the fields of a **virtual entity**.

Links can't do the job alone

Remember, the purpose of mapping is to associate the native payload of a building's equipment with the abstract and canonical concepts of the DBO. Since links only map fields to other fields, they don't deal with a payload directly.

Links must have a valid translation between a reporting device's payload and a reporting entity's fields.

Since these aren't one-to-one mappings, there can be many different ways to combine links and translations.



Back

Note: A model using only a translation mapping can be valid. However, a model only using a link mapping is never valid.

Next

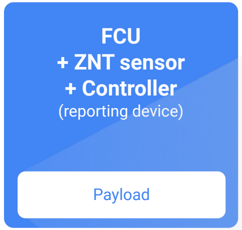
Mapping with links and translations (continued)

Links are used in conjunction with **translations** to map the payload of a **reporting device** to the fields of a **virtual entity**.

Example

Let's say you have a fan coil unit (FCU) with a zone temperature (ZNT) sensor and a single controller. You want to model the ZNT sensor as its own device because it's in a different room than the FCU.

The FCU's controller sends the device's native payload to the cloud, which makes it a reporting device. However, the cloud doesn't recognize its native payload.



Payload


```
{
  "timestamp": "2021-10-18T09:52:43.000Z",
  "points": {
    "zone_temperature": {
      "present_value": 72.43,
      "units": "degrees-F"
    },
    "zone_temperature_setpoint": {
      "present_value": 71,
      "units": "degrees-F"
    },
    "discharge_temp": {
      "present_value": 55.232,
      "units": "degrees-F"
    },
    "status_alarm": {
      "present_value": OFF
    }
  }
}
```


Back

*Click **Next** to inspect the translation in building config syntax.*

Next

Google

Mapping with links and translations (continued)

Links are used in conjunction with **translations** to map the payload of a **reporting device** to the fields of a **virtual entity**.

Example (continued)

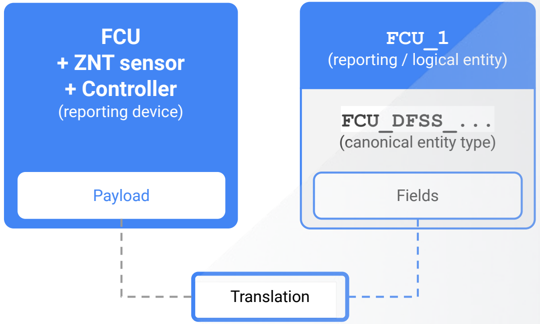
Let's say you have a fan coil unit (FCU) with a zone temperature (ZNT) sensor and a single controller. You want to model the ZNT sensor as its own device because it's in a different room than the FCU.

The FCU's controller sends the device's native payload to the cloud, which makes it a reporting device. However, the cloud doesn't recognize its native payload.

The FCU is a type of `FCU_DFSS_`..., which is a canonical entity type. We'll name the reporting entity `FCU-1`.

Using a translation, you can map the payload to the reporting entity `FCU-1` and standard fields of `FCU_DFSS_`.... Since this is a one-to-one mapping, `FCU-1` also a logical entity.

Now the FCU can send its translated payload, and the cloud can recognize it. However, the ZNT isn't modeled as its own device. It's just a field living on the FCU.



Translation

```
FCU-1:
  type: FCU_DFSS_… # Curated type because this is a logical entity.
  translation:
    # NOTE: The alarm is not translated.
    # Not everything in the payload must be translated.
    zone_air_temperature_sensor:
      present_value: points.zone_temperature.present_value
      units:
        key: points.zone_temperature.units
        value:
          degrees_fahrenheit: 'degrees-F'
    zone_air_temperature_setpoint:
      present_value: points.zone_temperature_setpoint.present_value
      units:
        key: points.zone_temperature_setpoint.units
        value:
          degrees_fahrenheit: 'degrees-F'
    discharge_air_temperature_sensor:
      present_value: points.discharge_temp.present_value
      units:
        key: points.discharge_temp.units
        value:
          degrees_fahrenheit: 'degrees-F'
...
```

Back

*Click **Next** to inspect the link in building config syntax.*

Next

Mapping with links and translations (continued)

Links are used in conjunction with **translations** to map the payload of a **reporting device** to the fields of a **virtual entity**.

Example (continued)

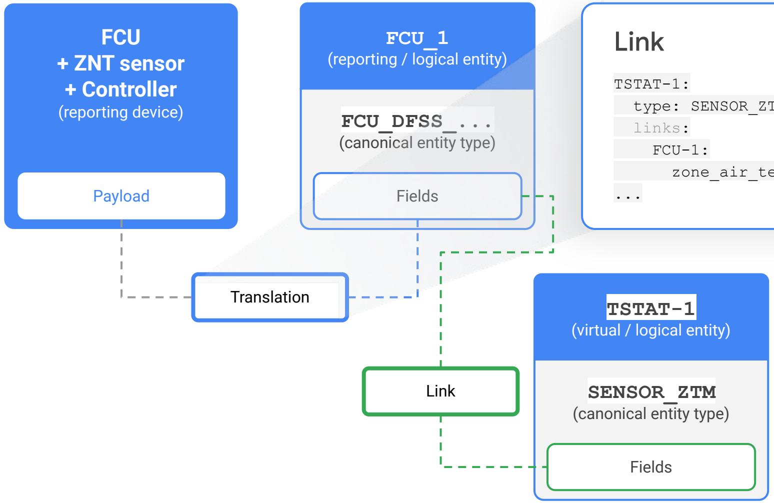
Let's say you have a fan coil unit (FCU) with a zone temperature (ZNT) sensor and a single controller. You want to model the ZNT sensor as its own device because it's in a different room than the FCU.

Using a translation, you've mapped the reporting device's payload to the entity `FCU-1` and the standard fields of the entity type `FCU_DFSS_`....

The ZNT sensor is a type of `SENSOR_ZTM` and a canonical entity type. Since it's something we care to model, we'll name this virtual entity `TSTAT-1`.

However, the ZNT sensor can't map one-to-one to the fields of `SENSOR_ZTM`, because the reporting device's payload is already mapped to `FCU-1`.

Using links, you can map the fields of the reporting entity `FCU-1` to the fields of the virtual entity `TSTAT-1`. Now these entities share the same ZNT sensor value.



Link

```
TSTAT-1:
  type: SENSOR_ZTM_… # Curated type because this is a logical entity
  links:
    FCU-1:
      zone_air_temperature_sensor: zone_air_temperature_sensor
...
```

Back

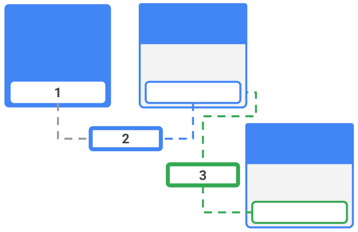
*Click **Next** for a summary of this example.*

Next

Google

Mapping with links and translations (continued)

Links are used in conjunction with **translations** to map the payload of a **reporting device** to the fields of a **virtual entity**.



Example summary

This mapping requires the following:

- Two logical entities: `FCU-1`, `TSTAT-1`
  - One reporting entity: `FCU-1`
  - One virtual entity: `TSTAT-1`
- Two canonical entity types: `FCU_DFSS_...`, `SENSOR_ZTM_...`

1. Payload

```
{
  "timestamp": "2021-10-18T09:52:43.000Z",
  "points": {
    "zone_temperature": {
      "present_value": 72.43,
      "units": "degrees-F"
    },
    "zone_temperature_setpoint": {
      "present_value": 71,
      "units": "degrees-F"
    },
    "discharge_temp": {
      "present_value": 55.232,
      "units": "degrees-F"
    },
    "status_alarm": {
      "present_value": OFF
    }
  }
}
```

2. Translation

```
FCU-1:
  type: FCU_DFSS_… # Curated type because this is a logical entity.
  translation:
    # NOTE: The alarm is not translated.
    # Not everything in the payload must be translated.
    zone_air_temperature_sensor:
      present_value: points.zone_temperature.present_value
      units:
        key: points.zone_temperature.units
        value:
          degrees_fahrenheit: 'degrees-F'
    zone_air_temperature_setpoint:
      present_value: points.zone_temperature_setpoint.present_value
      units:
        key: points.zone_temperature_setpoint.units
        value:
          degrees_fahrenheit: 'degrees-F'
    discharge_air_temperature_sensor:
      present_value: points.discharge_temp.present_value
      units:
        key: points.discharge_temp.units
        value:
          degrees_fahrenheit: 'degrees-F'
...
```

3. Link

```
TSTAT-1:
  type: SENSOR_ZTM_… # Curated type because this is
  links:
    FCU-1:
      zone_air_temperature_sensor: zone_air_temperature_sensor
...
```

Back

Next

Lesson 7

Knowledge check 2

Back

Google

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

- The next slide will have a question about translations.
- Review the question and select the correct response.
- After this knowledge check, you'll move on to learn about mapping with passthrough entities.

You won't be able to move forward until the correct answer is selected.

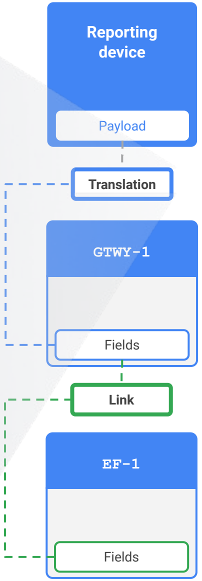
*Click **Next** when you're ready to begin.*

Next

Knowledge check 2

A sample translation is on the right.

```
GTWY-1:
  type: GTWY_NC_1 # Not curated
  translation:
    speed_frequency_sensor_1:
      present_value: points.ef_1_speed.present_value
      units:
        key: points.ef_1_speed.units
        value:
          hertz: 'hertz'
    speed_frequency_sensor_2:
      present_value: points.ef_2_speed.present_value
      units:
        key: points.ef_2_speed.units
        value:
          hertz: 'hertz'
    power_sensor_1:
      present_value: points.ef_1_kw.present_value
      units:
        key: points.ef_1_kw.units
        value:
          kilowatts: 'kw'
    power_sensor_2:
      present_value: points.ef_2_kw.present_value
      units:
        key: points.ef_2_kw.units
        value:
          kilowatts: 'kw'
...
```



Given this translation, which link is configured correctly?

Select the best answer from the options listed below.

```
EF-1:
  type: FAN_SS_… # Curated
  links:
      speed_frequency_sensor: speed_frequency_sensor_1
      power_sensor: power_sensor_1
...
```

```
EF-1:
  type: FAN_SS_… # Curated
  links:
    GTWY-1:
      speed_frequency_sensor_1: speed_frequency_sensor
      power_sensor_1: power_sensor
...
```

```
EF-1:
  type: FAN_SS_… # Curated
  links:
    GTWY-1:
      speed_frequency_sensor: speed_frequency_sensor_1
      power_sensor: power_sensor_1
...
```

Back

Note: Don't worry! We've already provided the standard fields for you. For this question, pay attention to the configuration of each option based on the information you're given from the sample.

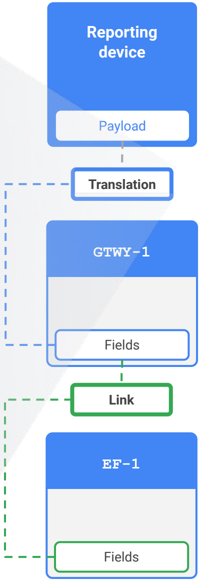
Next

Hmm, that’s not right! 🤔

This link isn’t configured correctly because it doesn’t name the source entity. In this sample, the source entity is `GTWY-1` and the target entity is `EF-1`.

Try again

```
GTWY-1:
  type: GTWY_NC_1 # Not curated
  translation:
    speed_frequency_sensor_1:
      present_value: points.ef_1_speed.present_value
      units:
        key: points.ef_1_speed.units
        value:
          hertz: 'hertz'
    speed_frequency_sensor_2:
      present_value: points.ef_2_speed.present_value
      units:
        key: points.ef_2_speed.units
        value:
          hertz: 'hertz'
    power_sensor_1:
      present_value: points.ef_1_kw.present_value
      units:
        key: points.ef_1_kw.units
        value:
          kilowatts: 'kw'
    power_sensor_2:
      present_value: points.ef_2_kw.present_value
      units:
        key: points.ef_2_kw.units
        value:
          kilowatts: 'kw'
...
```



Given this translation, which link is configured correctly?

Select the best answer from the options listed below.

```
EF-1:
  type: FAN_SS_… # Curated
  links:
      speed_frequency_sensor: speed_frequency_sensor_1
      power_sensor: power_sensor_1
...
```

```
EF-1:
  type: FAN_SS_… # Curated
  links:
    GTWY-1:
      speed_frequency_sensor_1: speed_frequency_sensor
      power_sensor_1: power_sensor
...
```

```
EF-1:
  type: FAN_SS_… # Curated
  links:
    GTWY-1:
      speed_frequency_sensor: speed_frequency_sensor_1
      power_sensor: power_sensor_1
...
```

Back

Next

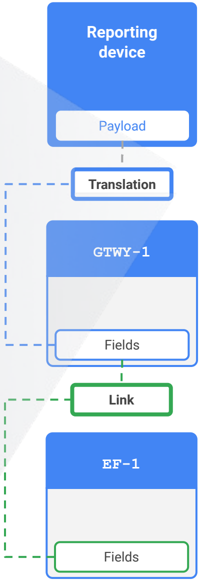
Google

Hmm, that's not right! 🤔

This link isn't configured correctly because the fields aren't listed in the correct order in the source entity block. The first field should be from the target entity **`EF-1`**, and the second field should be from the source entity **`GTWY-1`**.

Try again

```
GTWY-1:
  type: GTWY_NC_1 # Not curated
  translation:
    speed_frequency_sensor_1:
      present_value: points.ef_1_speed.present_value
      units:
        key: points.ef_1_speed.units
        value:
          hertz: 'hertz'
    speed_frequency_sensor_2:
      present_value: points.ef_2_speed.present_value
      units:
        key: points.ef_2_speed.units
        value:
          hertz: 'hertz'
    power_sensor_1:
      present_value: points.ef_1_kw.present_value
      units:
        key: points.ef_1_kw.units
        value:
          kilowatts: 'kw'
    power_sensor_2:
      present_value: points.ef_2_kw.present_value
      units:
        key: points.ef_2_kw.units
        value:
          kilowatts: 'kw'
...
```



Given this translation, which link is configured correctly?

Select the best answer from the options listed below.

```
EF-1:
  type: FAN_SS_… # Curated
  links:
      speed_frequency_sensor: speed_frequency_sensor_1
      power_sensor: power_sensor_1
...
```

```
EF-1:
  type: FAN_SS_… # Curated
  links:
    GTWY-1:
      speed_frequency_sensor_1: speed_frequency_sensor
      power_sensor_1: power_sensor
...
```

```
EF-1:
  type: FAN_SS_… # Curated
  links:
    GTWY-1:
      speed_frequency_sensor: speed_frequency_sensor_1
      power_sensor: power_sensor_1
...
```

Back

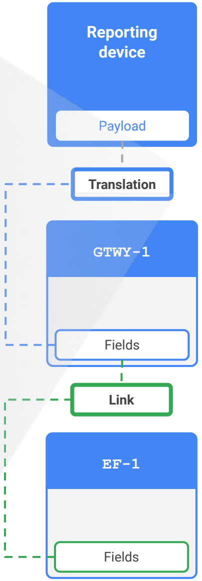
Next

That’s right! 🎉

This link is configured correctly, because:

- It names the target entity **`EF-1`**.
- It has a canonical entity type **`FAN_SS_...`**.
- It names the source entity **`GTWY-1`**.
- It lists the linked fields in the correct order:
  - First field is from the target entity.
  - Second field is from the source entity.

```
GTWY-1:
  type: GTWY_NC_1 # Not curated
  translation:
    speed_frequency_sensor_1:
      present_value: points.ef_1_speed.present_value
      units:
        key: points.ef_1_speed.units
        value:
          hertz: 'hertz'
    speed_frequency_sensor_2:
      present_value: points.ef_2_speed.present_value
      units:
        key: points.ef_2_speed.units
        value:
          hertz: 'hertz'
    power_sensor_1:
      present_value: points.ef_1_kw.present_value
      units:
        key: points.ef_1_kw.units
        value:
          kilowatts: 'kw'
    power_sensor_2:
      present_value: points.ef_2_kw.present_value
      units:
        key: points.ef_2_kw.units
        value:
          kilowatts: 'kw'
...
```



Given this translation, which link is configured correctly?

Select the best answer from the options listed below.

```
EF-1:
  type: FAN_SS_… # Curated
  links:
      speed_frequency_sensor: speed_frequency_sensor_1
      power_sensor: power_sensor_1
...
```

```
EF-1:
  type: FAN_SS_… # Curated
  links:
    GTWY-1:
      speed_frequency_sensor_1: speed_frequency_sensor
      power_sensor_1: power_sensor
...
```

```
EF-1:
  type: FAN_SS_… # Curated
  links:
    GTWY-1:
      speed_frequency_sensor: speed_frequency_sensor_1
      power_sensor: power_sensor_1
...
```

Back

*Click **Next** to move on and learn about mapping with passthrough entities.*

Next

Google

Mapping with a passthrough entity

A **passthrough entity** may be needed if a **reporting device** can't map one-to-one with a **logical entity**.

What's a passthrough entity?

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

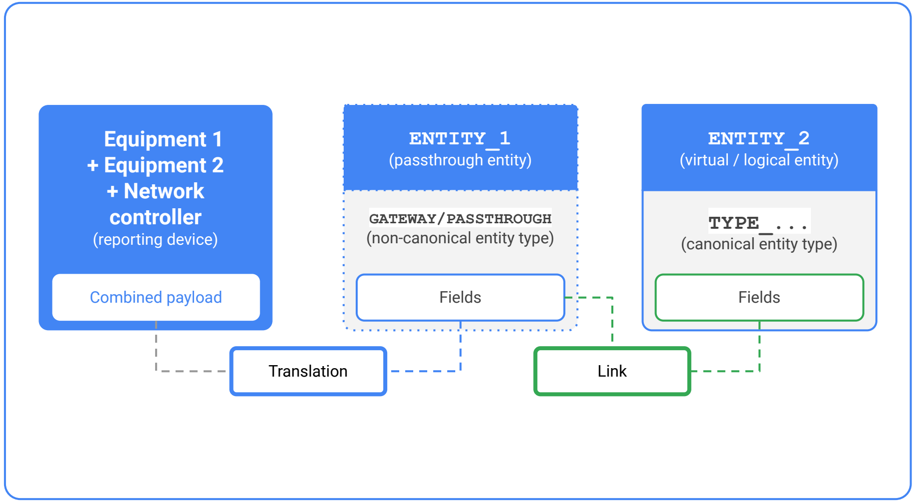
Why are passthrough entities needed?

The only use case for a passthrough entity is when a network controller needs to be mapped. Network controllers send a combined payload for multiple devices, but there isn't a canonical type for them in the DBO.

In these instances, the network controller's combined payload needs to be translated to a passthrough entity, then linked to a virtual entity for each device in the combined payload.

What's a passthrough entity type?

Passthrough entities use the special entity type `GATEWAYS/PASSTHROUGH`. It's a non-canonical type with a unique feature allowing the passthrough entity to have any standard field defined on it for translations.



Note: Passthrough entity types are reserved exclusively for the use of passthrough entities.

Mapping with a passthrough entity (continued)

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

Example

Let's say you have two exhaust fans that you'd like to model. The fans have a network controller that sends their combined payload to the cloud, which makes it a reporting device. However, the cloud doesn't recognize the combined native payload.

Exhaust fan 1
+ Exhaust fan 2
+ Network controller
(reporting device)

Combined payload

Combined payload


```
{
  "timestamp": "2021-10-18T09:52:43.000Z",
  "points": {
    "ef_1_speed": {
      "present_value": 60,
      "units": "hertz"
    },
    "ef_2_speed": {
      "present_value": 45,
      "units": "hertz"
    },
    "ef_1_kw": {
      "present_value": 12.77,
      "units": "kw"
    }
    "ef_2_kw": {
      "present_value": 6.753,
      "units": "kw"
    },
  }
}
```


Back

*Click **Next** to inspect the passthrough entity in building configuration format.*

Next

Mapping with a passthrough entity (continued)

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

Example (continued)

Let's say you have two exhaust fans that you'd like to model. The fans have a network controller that sends their combined payload to the cloud, which makes it a reporting device. However, the cloud doesn't recognize the combined native payload.

The network controller doesn't have a canonical entity type. Instead, `GATEWAYS/PASSTHROUGH` is needed, which is a non-canonical entity type reserved exclusively for instances involving network controllers and passthrough entities.

We'll name a passthrough entity for the network controller `GTWY_1` and associate it with `GATEWAYS/PASSTHROUGH`.

Now the network controller's combined payload can be translated.

Exhaust fan 1 + Exhaust fan 2 + Network controller
(reporting device)

Combined payload

`GTWY_1`
(passthrough entity)

`GATEWAYS/PASSTHROUGH`
(non-canonical entity type)

Passthrough entity type

```
PASSTHROUGH:
 id: "5942334369254342656"
 description: "A device that provides
translations for virtual entities."
 allow_undefined_fields: true
```

Back

*Click **Next** to inspect the translation in building configuration format.*

Next

Google

Mapping with a passthrough entity (continued)

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

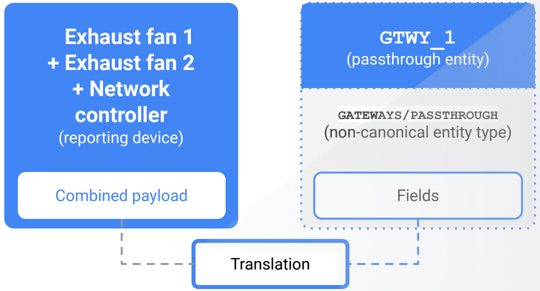
Example (continued)

Let's say you have two exhaust fans that you'd like to model. The fans have a network controller that sends their combined payload to the cloud, which makes it a reporting device. However, the cloud doesn't recognize the combined native payload.

You've created the passthrough entity `GTWY_1`. Because it's a passthrough entity, the entity type `GATEWAYS/PASSTHROUGH` is used.

Using a translation, you can map the combined payload to the passthrough entity `GTWY_1` and any required standard fields.

Now the network controller is able to send its translated payload, and the cloud can recognize it. However, the exhaust fans aren't modeled as their own devices yet.



Translation

```
GTWY-1:
  type: GATEWAYS/PASSTHROUGH
  translation:
    speed_frequency_sensor_1:
      present_value: points.ef_1_speed.present_value
      units:
        key: points.ef_1_speed.units
        value:
          hertz: 'hertz'
    speed_frequency_sensor_2:
      present_value: points.ef_2_speed.present_value
      units:
        key: points.ef_2_speed.units
        value:
          hertz: 'hertz'
    power_sensor_1:
      present_value: points.ef_1_kw.present_value
      units:
        key: points.ef_1_kw.units
        value:
          kilowatts: 'kw'
    power_sensor_2:
      present_value: points.ef_2_kw.present_value
      units:
        key: points.ef_2_kw.units
        value:
          kilowatts: 'kw'
...
```


*Click **Next** to inspect the links in building configuration format.*

Mapping with a passthrough entity (continued)

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

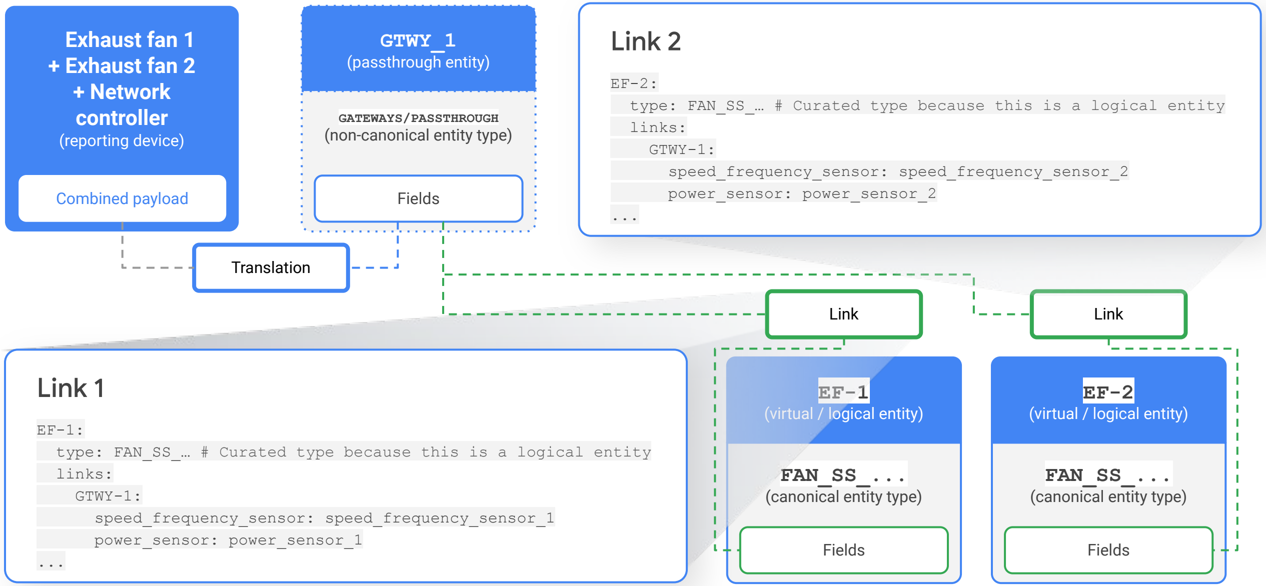
Example (continued)

Let's say you have two exhaust fans that you'd like to model. The fans have a network controller that sends their combined payload to the cloud, which makes it a reporting device. However, the cloud doesn't recognize the combined native payload.

Using a translation, you mapped the combined payload to the passthrough entity `GTWY_1` and any required standard fields.

The exhaust fans are types of `FAN_SS_...` and canonical entity types. Since we care to model the fans, we'll name these logical entities `EF-1` and `EF-2`.

Using links, you can map the fields of `GTWY-1` to the fields of `EF-1` and `EF-2`. Now the cloud recognizes the combined native payload of the two exhaust fans.



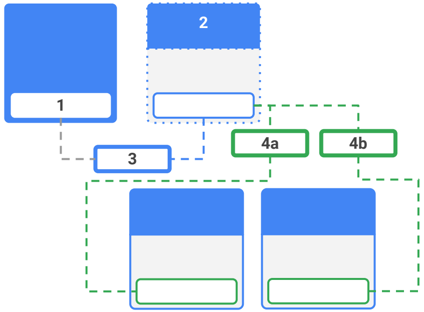
Back

*Click **Next** for a summary of this example.*

Next

Mapping with a passthrough entity (continued)

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



Example summary

This mapping requires the following:

- Two logical entities: **`EF-1`**, **`EF-2`**
- Two virtual entities: **`EF-1`**, **`EF-2`**
- One passthrough entity: **`GTWY-1`**
- One canonical entity type: **`FAN_SS_...`**

1. Payload


```
{
  "timestamp": "2021-10-18T09:52:43.000Z",
  "points": {
    "ef_1_speed": {
      "present_value": 60,
      "units": "hertz"
    },
    "ef_2_speed": {
      "present_value": 45,
      "units": "hertz"
    },
    "ef_1_kw": {
      "present_value": 12.77,
      "units": "kw"
    }
    "ef_2_kw": {
      "present_value": 6.753,
      "units": "kw"
  }
}
```


2. Passthrough entity type

```
PASSTHROUGH:
 id: "5942334369254342656"
 description: "A device that provides t
entities."
 allow_undefined_fields: true
```

3. Translation

```
GTWY-1:
  type: GATEWAYS/PASSTHROUGH
  translation:
    speed_frequency_sensor_1:
      present_value: points.ef_1_speed.pre
      units:
        key: points.ef_1_speed.units
        value:
          hertz: 'hertz'
    speed_frequency_sensor_2:
      present_value: points.ef_2_speed.present_value
      units:
        key: points.ef_2_speed.units
        value:
          hertz: 'hertz'
    power_sensor_1:
      present_value: points.ef_1_kw.presen
      units:
        key: points.ef_1_kw.units
        value:
          kilowatts: 'kw'
    power_sensor_2:
      present_value: points.ef_2_kw.presen
      units:
        key: points.ef_2_kw.units
        value:
          kilowatts: 'kw'
...
```

4a. Link 1

```
EF-1:
  type: FAN_SS_...# Curated type because this is a logical
entity
  links:
    GTWY-1:
      speed_frequency_sensor_1: speed_frequency_sensor
      power_sensor_1: power_sensor
...
```

4b. Link 2

```
EF-2:
  type: FAN_SS_...# Curated type because this is a logical
entity
  links:
    GTWY-1:
      speed_frequency_sensor_2: speed_frequency_sensor
      power_sensor_2: power_sensor
...
```

Back

Next

Lesson 7

Knowledge check 3

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

- The next slides will have one scenario with six questions to think about.
- Answer each question on your own and select the question to check your answer.
- After this knowledge check, you’ll wrap up Lesson 7.

Back

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

Next

Google

Knowledge check 3

On the roof of a building, there are:

- Five identical, standard AHUs. Each AHU sends its own payload to the cloud through their individual onboard controllers
- Two identical, standard fans. Their payload is combined and sent to the cloud through a network controller.
- Three identical, standard heat pumps. Their payload is combined and sent to the cloud through another network controller.

Think about each question below. Select the question to check your answer.

How many **entities** are required to model this scenario?

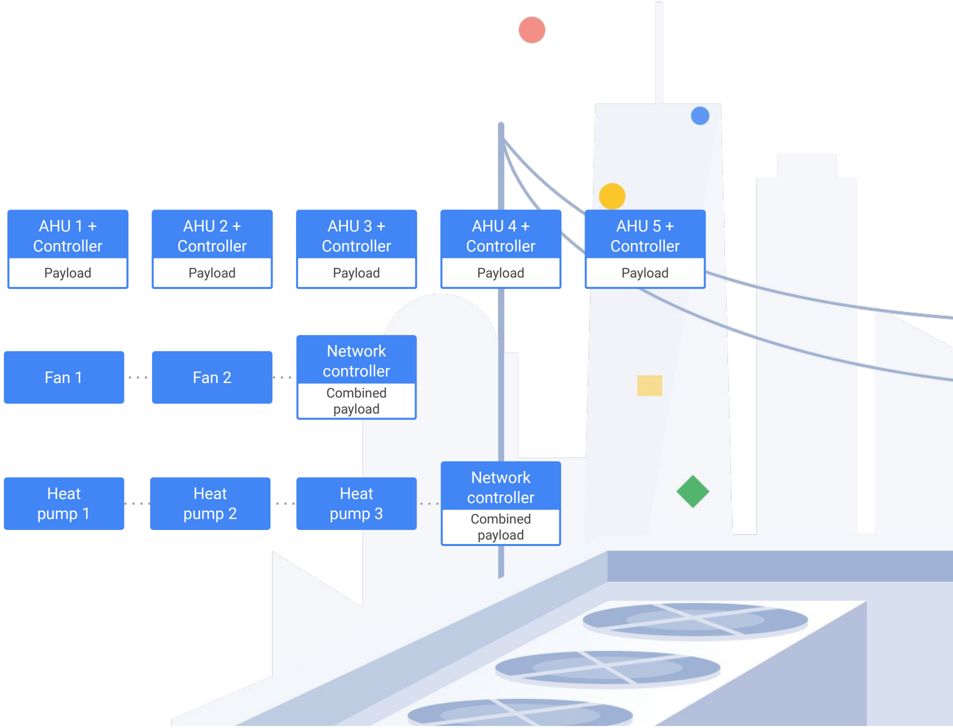
How many **logical entities** are required to model this scenario?

How many **canonical entity types** are required to model this scenario?

How many **reporting entities** are required to model this scenario?

How many **passthrough entities** are required to model this scenario?

How many **virtual entities** are required to model this scenario?



Back

Next

Google

Knowledge check 3

On the roof of a building, there are:

- Five identical, standard AHUs. Each AHU sends its own payload to the cloud through their individual onboard controllers
- Two identical, standard fans. Their payload is combined and sent to the cloud through a network controller.
- Three identical, standard heat pumps. Their payload is combined and sent to the cloud through another network controller.

Think about each question below. Select the question to check your answer.

How many **entities** are required to model this scenario?

How many **logical entities** are required to model this scenario?

How many **canonical entity types** are required to model this scenario?

How many **reporting entities** are required to model this scenario?

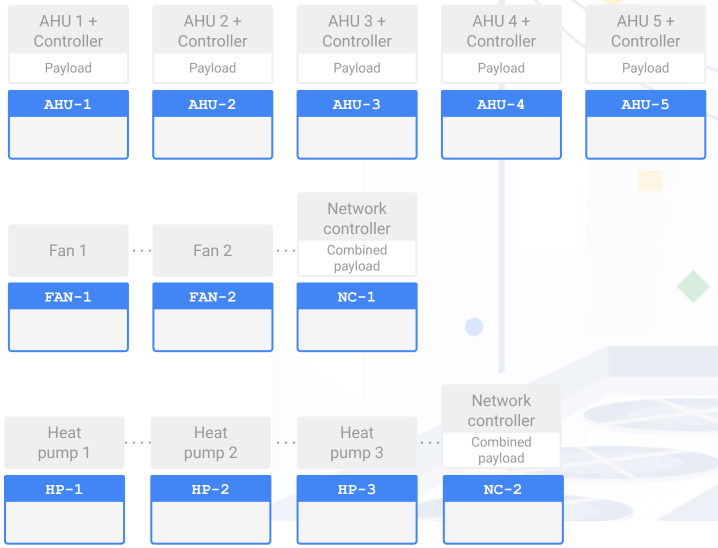
How many **passthrough entities** are required to model this scenario?

How many **virtual entities** are required to model this scenario?

Back

Check your answer! 👨‍🏫

A total of **twelve entities** are needed to model this scenario.
Each piece of equipment and network controller needs its own entity.



Next

Google

Knowledge check 3

On the roof of a building, there are:

- Five identical, standard AHUs. Each AHU sends its own payload to the cloud through their individual onboard controllers
- Two identical, standard fans. Their payload is combined and sent to the cloud through a network controller.
- Three identical, standard heat pumps. Their payload is combined and sent to the cloud through another network controller.

Think about each question below. Select the question to check your answer.

How many **entities** are required to model this scenario?

How many **logical entities** are required to model this scenario?

How many **canonical entity types** are required to model this scenario?

How many **reporting entities** are required to model this scenario?

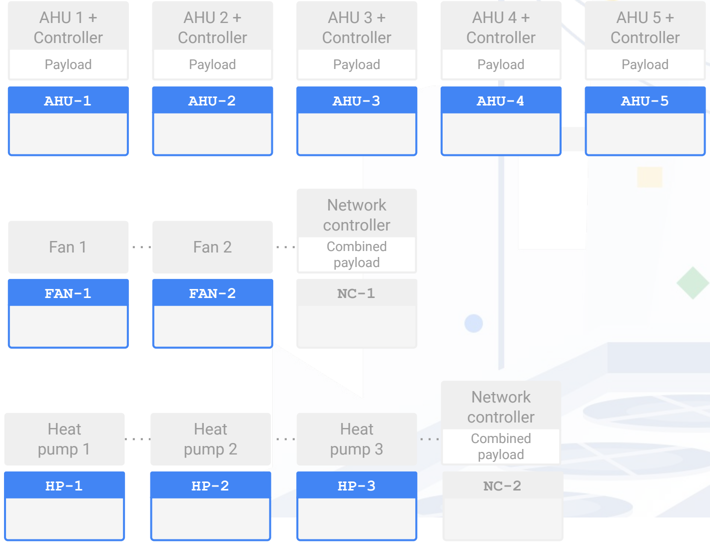
How many **passthrough entities** are required to model this scenario?

How many **virtual entities** are required to model this scenario?

Back

Check your answer!

Ten logical entities are needed: one for each device that needs to be modeled. A logical entity is also called a canonical entity. It maps one-to-one with a canonical entity type.



Next

Knowledge check 3

On the roof of a building, there are:

- Five identical, standard AHUs. Each AHU sends its own payload to the cloud through their individual onboard controllers
- Two identical, standard fans. Their payload is combined and sent to the cloud through a network controller.
- Three identical, standard heat pumps. Their payload is combined and sent to the cloud through another network controller.

Think about each question below. Select the question to check your answer.

How many **entities** are required to model this scenario?

How many **logical entities** are required to model this scenario?

How many **canonical entity types** are required to model this scenario?

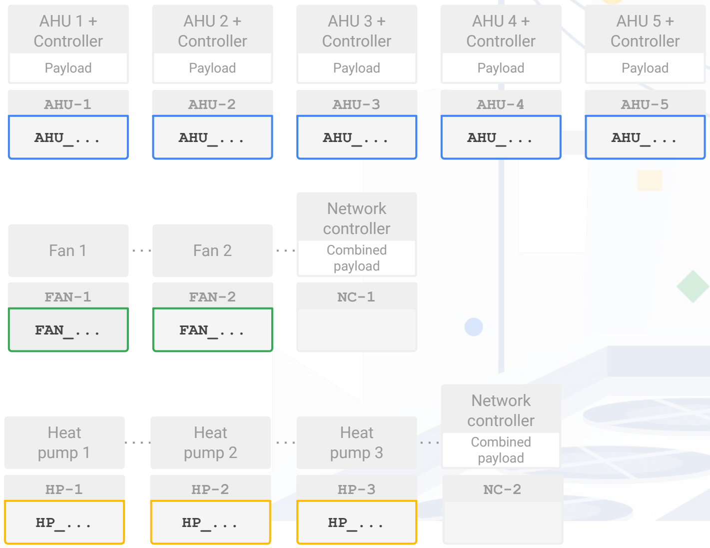
How many **reporting entities** are required to model this scenario?

How many **passthrough entities** are required to model this scenario?

How many **virtual entities** are required to model this scenario?

Check your answer!

Three canonical entity types are needed. One for the AHUs, one for the fans, and one for the heat pumps. Network controllers don't have a canonical type.



Back

Next

Knowledge check 3

On the roof of a building, there are:

- Five identical, standard AHUs. Each AHU sends its own payload to the cloud through their individual onboard controllers
- Two identical, standard fans. Their payload is combined and sent to the cloud through a network controller.
- Three identical, standard heat pumps. Their payload is combined and sent to the cloud through another network controller.

Think about each question below. Select the question to check your answer.

How many **entities** are required to model this scenario?

How many **logical entities** are required to model this scenario?

How many **canonical entity types** are required to model this scenario?

How many **reporting entities** are required to model this scenario?

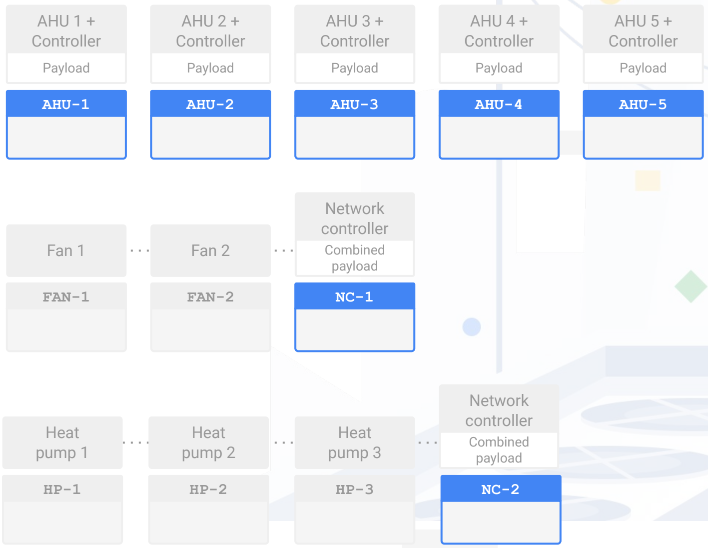
How many **passthrough entities** are required to model this scenario?

How many **virtual entities** are required to model this scenario?

Back

Check your answer!

Seven reporting entities are needed. One reporting entity for each reporting device (i.e., things that sends a payload of data).



Next

Knowledge check 3

On the roof of a building, there are:

- Five identical, standard AHUs. Each AHU sends its own payload to the cloud through their individual onboard controllers
- Two identical, standard fans. Their payload is combined and sent to the cloud through a network controller.
- Three identical, standard heat pumps. Their payload is combined and sent to the cloud through another network controller.

Think about each question below. Select the question to check your answer.

How many **entities** are required to model this scenario?

How many **logical entities** are required to model this scenario?

How many **canonical entity types** are required to model this scenario?

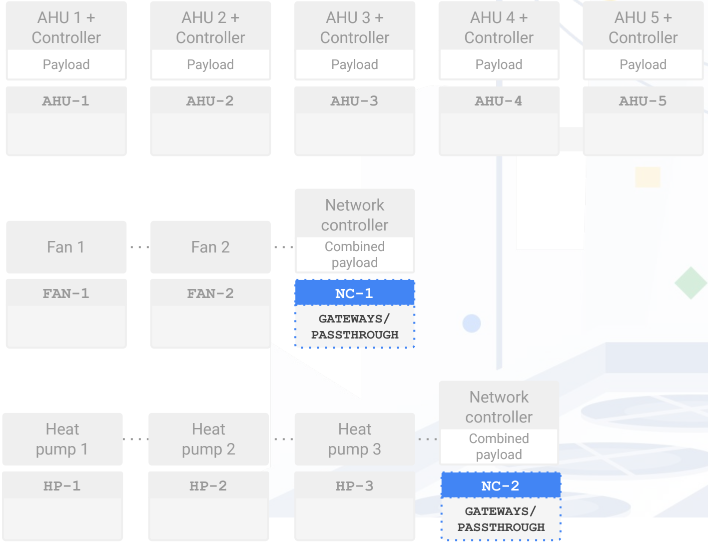
How many **reporting entities** are required to model this scenario?

How many **passthrough entities** are required to model this scenario?

How many **virtual entities** are required to model this scenario?

Check your answer! 👨‍🏫

Two passthrough entities are needed. The two network controllers can't map one-to-one with a logical entity. The passthrough entities will use the entity type `GATEWAYS/PASSTHROUGH`.



Back

Next

Google

Knowledge check 3

On the roof of a building, there are:

- Five identical, standard AHUs. Each AHU sends its own payload to the cloud through their individual onboard controllers
- Two identical, standard fans. Their payload is combined and sent to the cloud through a network controller.
- Three identical, standard heat pumps. Their payload is combined and sent to the cloud through another network controller.

Think about each question below. Select the question to check your answer.

How many **entities** are required to model this scenario?

How many **logical entities** are required to model this scenario?

How many **canonical entity types** are required to model this scenario?

How many **reporting entities** are required to model this scenario?

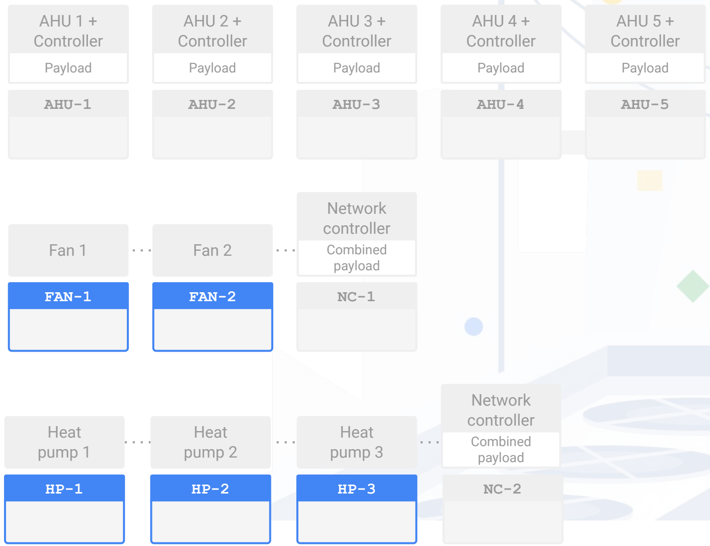
How many **passthrough entities** are required to model this scenario?

How many **virtual entities** are required to model this scenario?

Back

Check your answer! 👩‍🏫

Five virtual entities are needed. These devices need to be modeled separately from their reporting entities.



Next

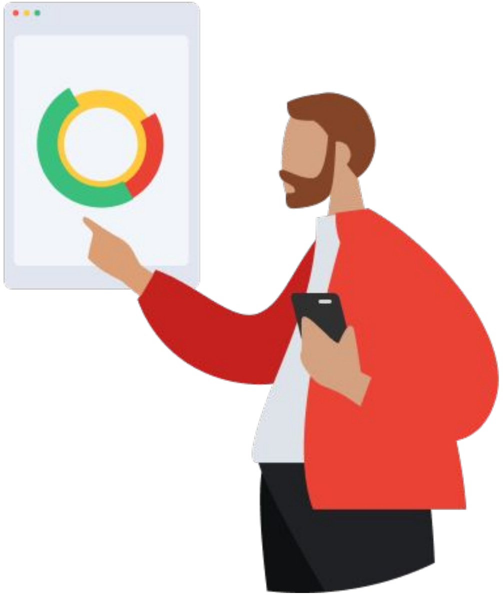
Lesson 7 summary

Let’s review what you learned about:

- Mappings and logical devices
- Translations, reporting devices, reporting entities, and point mapping
- Links, virtual entities, and passthrough entities

Now you should be able to:

- Describe the purpose of mapping.
- Recognize the correct configuration of a translation.
- Recognize the correct configuration of a link.
- Describe the need for point mapping.
- Understand the dependencies between links and translations for data modeling.
- Determine when reporting entities, virtual entities, and passthrough entities are needed.



Back

Next

You completed Lesson 7!

Now's a great time to take a quick break before starting Lesson 8.

Ready for Lesson 8?

Let's go!

Back

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

Google

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 questions: udmi-discuss@

Helpful resources

Bookmark these resources for future reference.

- digitalbuildings / ontology
  Contains the documentation and configuration files for the DBO.
- Building Configuration
  Document that describes the configuration format for mapping concrete assets to a model.