



Module 2 | Lesson 4



Data modeling with the DBO



Before you get started

This learning module 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 content.

1 View this content outside of GitHub.

- For the best learning experience, you're encouraged to download a copy so links and other interactive features will be enabled.
- To download a copy of this lesson, click **Download** in the top-right corner of this content block.
- After downloading, open the file in your preferred PDF reader application.

2 Navigate by clicking the buttons and links.

- For the best learning experience, using your keyboard or mouse wheel to navigate is discouraged. However, this is your only option if you're viewing from GitHub.
- If you're viewing this content outside of GitHub:
 - 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.

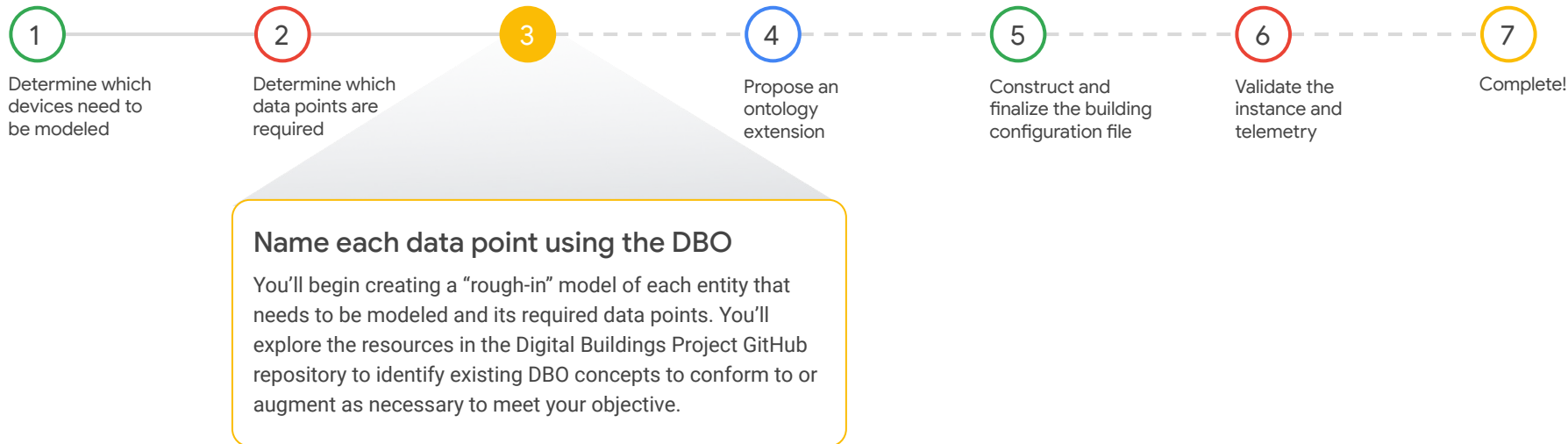
Ready to get started?

Let's go!

Workflow revisited

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

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



[Back](#)

[Next](#)

Lesson 4

Name each data point using the DBO

What you'll learn about:

- Rough-in models

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

- Prepare a new spreadsheet for a rough-in model.
- Rough-in the names of required data points.

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

Rough-in models

You're strongly encouraged to create a **rough-in model** before integrating or modeling data.

What's a rough-in model?

A **rough-in model** is a similar idea to the rough-in stage of any construction project. Just like roughing-in the electrical, mechanical, and plumbing of a building before final connections are made, roughing-in a data model of a digital building project intends to lay out and name the exact data points that will need to be integrated and modeled.

Why create a rough-in model?

There are two major reasons for roughing-in the data and naming data points so early in the workflow:

1. Data integration - Informs how you register devices and gateways that generate UDMI-compliant payloads.
2. Data modeling - Simplifies how you translate UDMI-compliant payloads into DBO-compliant data models.

Roughing-in also allows you to identify gaps in the DBO. While you should always stick with precedent, you may run into something that needs to be modeled but doesn't have a curated field or entity type. In these cases, you'll submit an ontology extension to propose new modeling concept definitions. You'll learn about ontology extensions in the next lesson.

[Back](#)

Note: You'll learn all about the data integration process, including device and gateway registration, in another module. For now, let's just focus on the data modeling workflow.

[Next](#)

Rough-in models

A **rough-in model** is simply a spreadsheet naming all data points that need to be modeled.

Rough-ins aren't meant to be an end deliverable. However, you can use rough-ins to properly plan how you integrate and model data.

How to prepare a new rough-in model

1. Create a new spreadsheet [in your Google Drive](#) or using your preferred application.
2. Set up the following column headings:
 - Namespace
 - General type
 - Entity name
 - Field name
3. Below the column headings, designate each row as a separate data point.

Diagram illustrating the structure of a rough-in model spreadsheet:

- Namespace and General Types** aligned to DBO
- Entity Name** to distinguish logical devices from each other
- Field Name** aligned to DBO
- Each row represents a single point on a piece of equipment

Namespace	General Type	Entity Name	Field Name
HVAC	AHU	AC 1	building_air_static_pressure_sensor
HVAC	AHU	AC 1	building_air_static_pressure_setpoint
HVAC	AHU	AC 1	compressor_run_command
HVAC	AHU	AC 1	compressor_run_status
HVAC	AHU	AC 1	compressor_speed_percentage_command
HVAC	AHU	AC 1	economizer_mode
HVAC	AHU	AC 1	exhaust_air_damper_percentage_command
HVAC	AHU	AC 1	exhaust_fan_run_command
HVAC	AHU	AC 1	exhaust_fan_run_status
HVAC	AHU	AC 1	low_limit_outside_air_damper_percentage_command
HVAC	AHU	AC 1	mixed_air_temperature_sensor
HVAC	AHU	AC 1	outside_air_damper_percentage_command
HVAC	AHU	AC 1	outside_air_temperature_sensor
HVAC	AHU	AC 1	return_air_temperature_sensor
HVAC	AHU	AC 1	supply_air_flowrate_sensor
HVAC	AHU	AC 1	supply_air_static_pressure_sensor
HVAC	AHU	AC 1	supply_air_static_pressure_setpoint
HVAC	AHU	AC 1	supply_air_temperature_sensor
HVAC	AHU	AC 1	supply_air_temperature_setpoint
HVAC	AHU	AC 1	supply_fan_run_command
HVAC	AHU	AC 1	supply_fan_run_status
HVAC	AHU	AC 1	supply_fan_speed_percentage_command
HVAC	AHU	AC 1	supply_fan_speed_percentage_sensor
HVAC	BLR	BLR 1	return_water_isolation_valve_command
HVAC	BLR	BLR 1	return_water_isolation_valve_status
HVAC	BLR	BLR 1	run_command
HVAC	BLR	BLR 1	run_status
HVAC	BLR	BLR 1	supply_water_temperature_sensor
HVAC	BLR	BLR 1	supply_water_temperature_setpoint
HVAC	FAN	EF 10	run_command
HVAC	FAN	EF 10	run_status

[Back](#)

[Next](#)

Lesson 4

Practice 1



[Back](#)

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

- The next slides will walk through the steps to prepare a new spreadsheet for a rough-in model.
- After this practice activity, you'll use the new sheet as you learn about completing the rough-in model.

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

[Next](#)

Practice 1

In this lesson, we're going to create a rough-in model for the **AHU** shown here.

Let's prepare a new spreadsheet for your rough-in model.

Follow the steps below.

Steps

1. Create a new spreadsheet [in your Google Drive](#) or using your preferred application.
2. Set up the following column headings:
 - Namespace
 - General type
 - Entity name
 - Field name
3. Below the column headings, designate each row as a separate data point.

BMS screenshot

Here's the **AHU** from Lesson 3 that you inspected and determined the required data points to model.

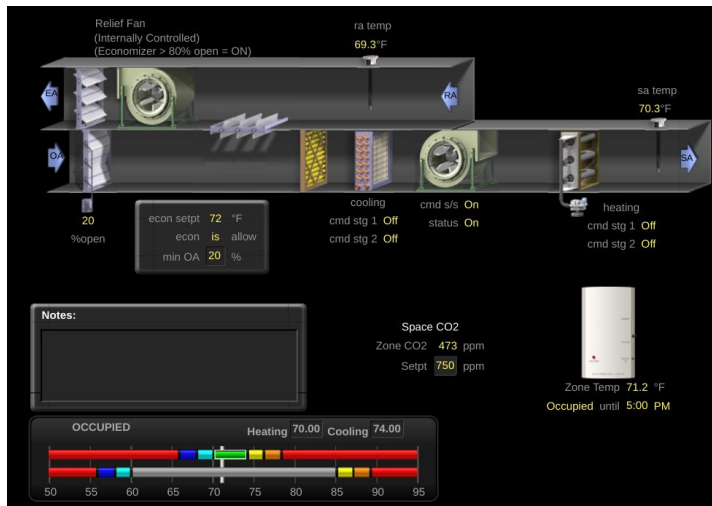


Image source: Google's WebCTRL instances.
WebCTRL is a building automation system owned by Automated Logic.

[Back](#)

*When you're ready, click **Next** to check your work and see how we prepared our rough-in spreadsheet.*

[Next](#)

Practice 1

Check your answer! 

Here's a look at how our spreadsheet turned out. **Did you come up with something similar?**

This is a basic setup for creating a rough-in model. Feel free to take this and customize it to your liking.

Be sure to keep your new rough-in sheet accessible. For the remainder of this lesson, you'll work through a scenario to fill it in by naming data points.

Rough-in sheet

Namespace	General type	Entity name	Field name

Note: Keep this file easily accessible for the next activity in this lesson.

[Back](#)

[Next](#)

Rough in the names of each data point

Moving forward, we'll use our new rough-in sheet to name the data points of this AHU together.

Back

BMS screenshot

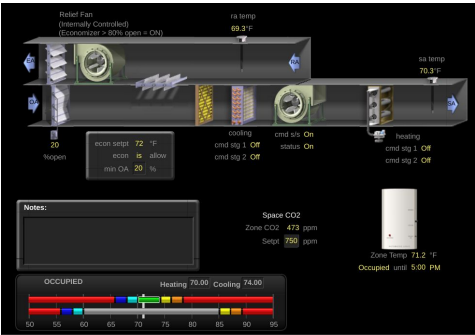


Image source: Google's WebCTRL instances.
WebCTRL is a building automation system owned by Automated Logic.

Rough-in sheet

Namespace	General type	Entity name	Field name
Data point 1	Data point 1	Data point 1	Data point 1
Data point 2	Data point 2	Data point 2	Data point 2
Data point 3	Data point 3	Data point 3	Data point 3
Data point 4	Data point 4	Data point 4	Data point 4
Data point 5	Data point 5	Data point 5	Data point 5

Next

Rough in the names of each data point

Each row of the rough-in model is a separate data point.

Click on each column heading to reveal how to find this information.

Namespace	General type	Entity name	Field name
Data point 1	Data point 1	Data point 1	Data point 1
Data point 2	Data point 2	Data point 2	Data point 2

BMS screenshot

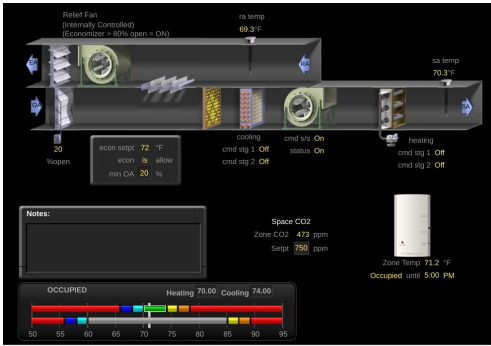


Image source: Google's WebCTRL instances.
WebCTRL is a building automation system owned by Automated Logic.

What happens now?

Row by row, you'll identify the namespace, general type, entity name, and field name that corresponds with every individual data point that you previously determined needs to be included in the data model (revisit Lesson 3 if you're not sure about required data points).

Keep in mind that you don't need to work in this order. It may make more sense for you to identify a point's namespace before its field name or its field name before its entity name. In whatever order you complete each row, make sure you're able to complete each column accordingly.

Back

When you're ready, click **Next** to rough in the "Namespace" column.

Next

Rough in the “Namespace” column

Each row of the rough-in model is a separate data point.

Click on each column heading to reveal how to find this information.

Namespace	General type	Entity name	Field name
HVAC			

BMS screenshot

This device is clearly an **AHU**, and therefore fits within the HVAC namespace.

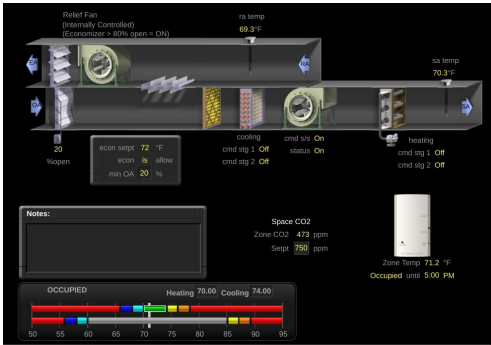


Image source: Google's WebCTRL instances.
WebCTRL is a building automation system owned by Automated Logic.

Namespace

In the “Namespace” column, you’ll identify the application domain of the logical device that generates the row’s data point. This will help you pinpoint in which child namespace folder to look for general types and fields.

By now, you should have already identified devices that need to be modeled (revisit Lesson 2). Identifying the namespace of each device should be fairly straightforward based on your domain knowledge and the DBO’s supported namespaces.

Currently, the following namespaces are supported by the DBO:

- **HVAC:** Includes air handling units (**AHU**), boilers (**BLR**), chillers (**CH**), and more.
- **Lighting:** Includes light fixtures (**LT**), lighting gateways (**LTGW**), emergency lights (**ELT**), and more.
- **Metering:** Includes electrical meters (**EM**), gas meters (**GM**), water meters (**WM**), and more.
- **Electrical:** Includes batteries (**BATT**), uninterruptible power supplies (**UPS**), and panels (**PANEL**).
- **Safety:** Includes smoke detectors (**SD**), fire dampers (**FD**), fire alarm systems (**FAS**), and more.

Refer to:

- Your list of required devices to model (from Lesson 2)
- GENERALTYPES.yaml in each DBO namespace

Back

When you’re ready, click **Next** to rough in the “General type” column.

Next

Rough in the “General type” column

Each row of the rough-in model is a separate data point.

Click on each column heading to reveal how to find this information.

Namespace	General type	Entity name	Field name
	AHU		

BMS screenshot

Since this is clearly an **AHU**, and since that is a general type, that’s what we’ll apply.

We’ll use this in the near future to help us match this device’s fields to an entity type.

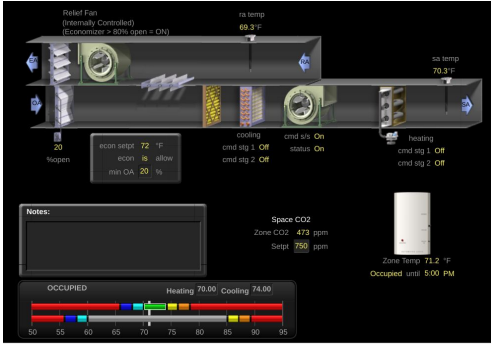


Image source: Google’s WebCTRL instances.
WebCTRL is a building automation system owned by Automated Logic.

General type

In the “General type” column, you’ll name the general entity type of the entity that will be associated with the logical device that generates the row’s data point. This will help you explore canonical and abstract types defined in a child namespace folder to find fields that correlate with the data point.

By now, you should have already determined the devices that will need to be modeled (revisit Lesson 2) along with the general type that classifies the type of entity to associate with the devices (revisit Lesson 3). Inputting the general types of each device based on what you previously determined should be fairly straightforward.

Refer to:

- Your list of required devices to model (from Lesson 2)
- Your list of general types for each device to model (from Lesson 3)
- GENERALTYPES.yaml in each DBO namespace

What if there isn’t a general type for what I need to model?

If a general type hasn’t been defined yet, then you’ll need to propose an ontology extension. See Lesson 5 for more information about extending the ontology.

Back

When you’re ready, click **Next** to rough in the “Entity name” column.

Next

Rough in the “Entity name” column

Each row of the rough-in model is a separate data point.

Click on each column heading to reveal how to find this information.

Namespace	General type	Entity name	Field name
		AHU-1	

BMS screenshot

This screenshot doesn't show a name, but let's assume that it's referred to **AHU-1** within this building. That's the name we'll give it.

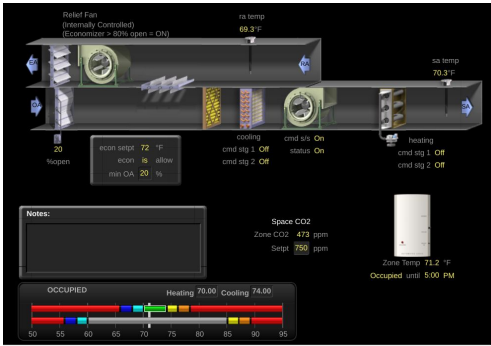


Image source: Google's WebCTRL instances. WebCTRL is a building automation system owned by Automated Logic.

Entity name

In the “Entity name” column, you'll name the entity that will be associated with the logical device that generates the row's data point. This will help you differentiate between general types and field names that are reused by different entities.

By now, you should have already determined the devices that will need to be modeled (revisit Lesson 2). Each device will need to be mapped to an entity. Since you should have also already identified a general type for each device to be modeled (revisit Lesson 3), it should be fairly straightforward to name each entity. The name of an entity usually adopts and enumerates the name of its general type (in a brownfield site, you usually will have to stick with the existing name – except if it is used for multiple devices in the same building).

For example, if you identified three air handling units to model and you determined their general type is **AHU**, then you'd name them **AHU-1**, **AHU-2**, and **AHU-3**.

Refer to:

- Your list of required devices to model (from Lesson 2)
- Your list of general types for each device to model (from Lesson 3)
- GENERALTYPES.yaml in each DBO namespace

Note: Entity names must be unique within a building. If you have multiple entities with the same name in the same building (e.g., **AHU-1** on the 10th floor and **AHU-1** on the 11th floor), then you'll have to modify the name of one or both devices.

Back

When you're ready, click **Next** for info about the “Field name” column.

Next

Rough in the “Field name” column

Each row of the rough-in model is a separate data point.

Click on each column heading to reveal how to find this information.

Namespace	General type	Entity name	Field name

BMS screenshot

Based on the required data points that were identified in Lesson 3, this **AHU** requires a number of fields.

We'll look at an example of roughing in this kind of information over the next couple of slides.

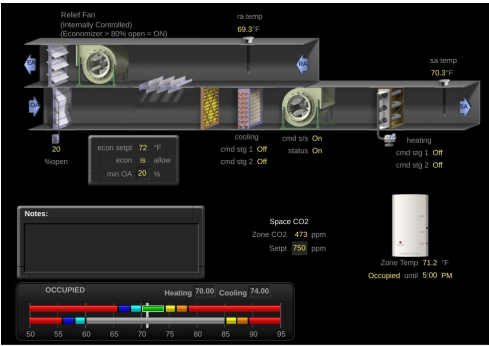


Image source: Google's WebCTRL instances.
WebCTRL is a building automation system owned by Automated Logic.

Field name

In the “Field name” column, you’ll name the field associated with a data point that would be generated by a reporting device. If it’s possible for the device to generate multiple points of data, determine which device data is required and input their field name in separate rows under the “Field name” column. This is what will inform how you register devices and gateways to simplify the translation of payloads to DBO-compliant data models.

By now, you should have already determined the abstract types, field associations, and required device data that correspond with the functions of the devices that need to be modeled (revisit Lesson 3). Using the rough-in sheet, you’ll organize these findings so each row is for a different field since each row represents a different data point.

Refer to:

- Your list of abstract types and field associations for each device to model (from Lesson 3)
- Your list of required device data for each device to model (from Lesson 3)
- ABSTRACT.yaml in each DBO namespace
- Canonical type .yaml corresponding with general types
- fields.yaml in global namespace

What if there isn’t a field for what I need to model?

If a field hasn’t been defined yet, then you’ll need to propose an ontology extension. See Lesson 5 for more information about extending the ontology.

Back

When you’re ready, click **Next** to see how to rough in the “Field name” column.

Next

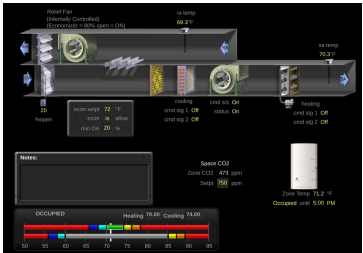
Rough in the “Field name” column

Here we’ll continue our rough-in and fill in the field names based on functionality.

Select each function to reveal how to find this information.

- Zone temp control
- CO₂
- Fan control
- Cooling control
- Economizer
- Other functions

BMS screenshot



The screenshot displays a BMS interface with a 3D model of a building section at the top. Below the model, there are several data panels. On the left, a 'Status' panel shows 'Zone temp: 72 °F', 'Zone CO2: 473 ppm', and 'Zone temp: 72.2 °F'. In the center, there are controls for 'cooling' and 'heating' with 'On' and 'Off' buttons. On the right, a 'Zone temp' panel shows 'Zone temp: 72.2 °F' and 'Occupied until: 5:00 PM'. At the bottom, there is a 'Heating' and 'Cooling' status bar with a color-coded scale from 50 to 80.

Image source: Google's WebCTRL instances. WebCTRL is a building automation system owned by Automated Logic.

Rough-in sheet

	Field name

What happens now?

Now we go around the project documents and identify all of the relevant fields in order to name them. It helps to go around the device functionally.

Let’s look at the data we’ll bring in (and what data we won’t) and why. In this case, we’ll inspect a screenshot of a BMS drawing, but in other situations it could be a sequence of operations or manufacturer’s specification.

Back

Next

Rough in the “Field name” column

Here we’ll continue our rough-in and fill in the field names based on functionality.

Select each function to reveal how to find this information.

Zone temp control

CO₂

Fan control

Cooling control

Economizer

Other functions

BMS screenshot

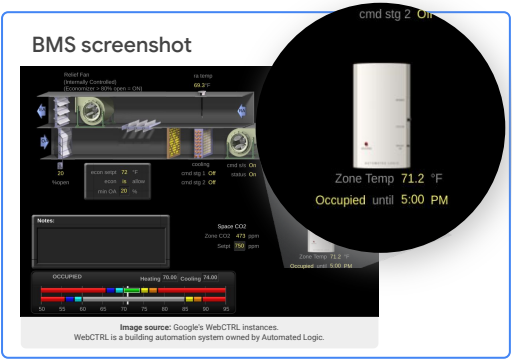


Image source: Google's WebCTRL Instances.
WebCTRL is a building automation system owned by Automated Logic.

Zone temp control

We see a zone air temperature sensor and two setpoints—heating and cooling.

We know from having previously investigated the available abstract types and checking some sample canonical **AHUS** that **DSP** is likely the type that will be applied, and we know that we’ll want both setpoints and the sensor.

Rough-in sheet	
	Field name
	zone_air_temperature_sensor
	zone_air_cooling_temperature_setpoint
	zone_air_heating_temperature_setpoint

Back

Next

Rough in the “Field name” column

Here we’ll continue our rough-in and fill in the field names based on functionality.

Select each function to reveal how to find this information.

- Zone temp control
- CO₂**
- Fan control
- Cooling control
- Economizer
- Other functions

BMS screenshot

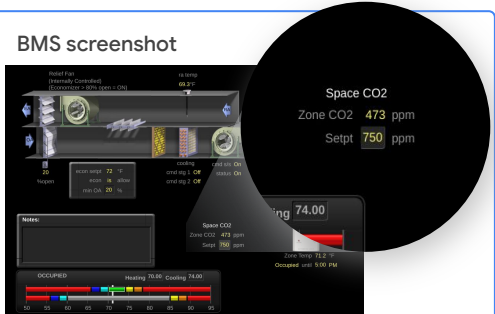


Image source: Google's WebCTRL Instances.
WebCTRL is a building automation system owned by Automated Logic.

CO₂

Next, we see the CO₂ sensor and setpoint.

This is simple enough to add, since we know that `co2c` is a typical type for many ventilating devices. We can also search the abstract types file for types which use the field `zone_air_co2_concentration_sensor`.

Rough-in sheet

Field name
zone_air_co2_concentration_sensor
zone_air_co2_concentration_setpoint

Back

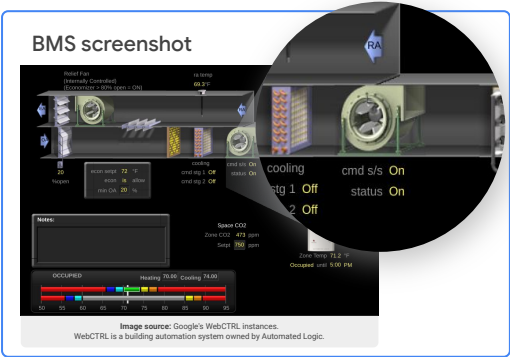
Next

Rough in the “Field name” column

Here we’ll continue our rough-in and fill in the field names based on functionality.

Select each function to reveal how to find this information.

- Zone temp control
- CO₂
- Fan control
- Cooling control
- Economizer
- Other functions



Rough-in sheet	
	Field name
	discharge_fan_run_command
	discharge_fan_run_status

Fan control

Continuing, we see constant speed fan control.

Note the lack of a variable frequency drive (VFD) in the screenshot.

In general, you would confirm this through more than just a screenshot review, but for simplicity of the example we’ll assume the screenshot is sufficient to rule out the existence of a VFD. There are two fans, but one (the relief/exhaust fan) doesn’t appear to have data associated with it. Therefore we’ll focus on the other fan.

This **AHU** has what is typically referred to as a supply fan.

However, remember that DBO considers supply to be literally supplied from one piece of equipment to another. This **AHU** supplies air directly to the space (as can be inferred by the fact that we have zone-level control fields, such as **zone_temp_sensor** and **zone_CO2_sensor**). Therefore this is actually a discharge fan.

From the screenshot, we can see it has both a command (S/S) and a status feedback. There is an abstract type that we know of which looks like this: **DFSS**. Therefore we add those points here.

Back

Next

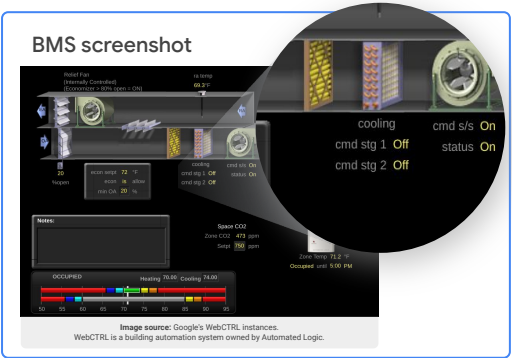
Rough in the “Field name” column

Here we’ll continue our rough-in and fill in the field names based on functionality.

Select each function to reveal how to find this information.

- Zone temp control
- CO₂
- Fan control
- Cooling control
- Economizer
- Other functions

BMS screenshot



Rough-in sheet

Field name
compressor_run_command
compressor_run_status
zone_air_temperature_sensor
zone_air_cooling_temperature_setpoint
discharge_air_temperature_sensor

Cooling control

Now let’s look at the cooling control.

It appears that there is a discharge air temperature sensor, but there doesn’t appear to be a discharge temperature setpoint.

If we were to inspect the controls, we would find that the direct expansion (DX) section controls the zone to temperature.

If we check the abstract types, we will find one called **DXZC**, which appears to do exactly what we’re looking for: control the zone temperature to setpoint using the compressor sections. It even covers the discharge air temperature sensor as an optional field. Since we know we have all these fields, let’s add them here.

It’s fine that we are applying **DXZC** and **DSP** to the same device. The overlap of the fields **zone_air_temperature_sensor** and **zone_air_cooling_temperature_setpoint** isn’t an issue.

Note: Technically, the screenshot depicts a two-stage DX section. However, to simplify the example, we’ll assume the two data points represent a single-stage command and status.

Back

Next

Rough in the “Field name” column

Here we’ll continue our rough-in and fill in the field names based on functionality.

Select each function to reveal how to find this information.

- Zone temp control
- CO₂
- Fan control
- Cooling control
- Economizer
- Other functions

BMS screenshot

Economizer

Lastly, let’s inspect the economizer section.

We would start by trying to determine how the economizer section works, since we know it’s going to be **ECONM**, **ECOND**, **ECONZ**, etc., or a yet-to-be defined abstract type of similar structure.

Upon further inspection, it appears to most closely fit with **ECONZ**. We will add both the required and optional fields to its field list.

Rough-in sheet

Field name
outside_air_temperature_sensor
outside_air_damper_percentage_command
economizer_mode
low_limit_outside_air_damper_percentage_command
discharge_air_temperature_sensor

Back

Next

Rough in the “Field name” column

Here we’ll continue our rough-in and fill in the field names based on functionality.

Select each function to reveal how to find this information.

- Zone temp control
- CO₂
- Fan control
- Cooling control
- Economizer
- Other functions

BMS screenshot




Image source: Google's WebCTRL Instances. WebCTRL is a building automation system owned by Automated Logic.

Rough-in sheet

Field name
return_air_temperature_sensor

Other functions

And that appears to cover most of the major functionality. Now, let’s complete the rough-in and add any fields that weren’t directly covered through the previous exercises.

Notice how the return temperature sensor was not previously mentioned, but it does exist. Knowing that we generally integrate all telemetry generating fields, we’ll add this too. After searching [telemetry_fields.yaml](#) and inspecting some typical AHUs or abstract types, we’ll see that the field name ought to be `return_air_temperature_sensor`.

We add that here and finalize the rough-in.

Back

Next

Rough in the names of each data point

Here's a look at our final rough-in model.

Click on a column heading to review how we filled in this information.

Namespace	General type	Entity name	Field name
HVAC	AHU-1	AHU	outside_air_temperature_sensor
HVAC	AHU-1	AHU	outside_air_damper_percentage_command
HVAC	AHU-1	AHU	economizer_mode
HVAC	AHU-1	AHU	low_limit_outside_air_damper_percentage_command
HVAC	AHU-1	AHU	discharge_air_temperature_sensor
HVAC	AHU-1	AHU	zone_air_cooling_temperature_setpoint
HVAC	AHU-1	AHU	compressor_run_command
HVAC	AHU-1	AHU	compressor_run_status
HVAC	AHU-1	AHU	discharge_fan_run_command
HVAC	AHU-1	AHU	discharge_fan_run_status
HVAC	AHU-1	AHU	zone_air_co2_concentration_sensor
HVAC	AHU-1	AHU	zone_air_co2_concentration_setpoint
HVAC	AHU-1	AHU	zone_air_temperature_sensor
HVAC	AHU-1	AHU	zone_air_heating_temperature_setpoint
HVAC	AHU-1	AHU	return_air_temperature_sensor

BMS screenshot

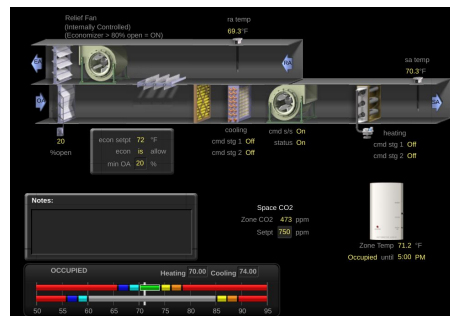


Image source: Google's WebCTRL instances.
WebCTRL is a building automation system owned by Automated Logic.

Note 1: Notice how we have defined no alarm fields. Apart from the sample not showing any alarms, you can imagine there'll probably be some typical ones available to integrate like **failed_fan**, **failed_sensors**, etc. We normally ignore these. They can be inferred from a lot of the data that we have (e.g., you can determine if the fan is in alarm by looking at the fan command and status). Therefore, it isn't needed here.

Note 2: There are no configuration fields brought in for any of the control loops (e.g., zone control PID is ignored). Again, these are low-level details that aren't needed. We can infer badly tuned loops from the data (e.g., does it overshoot/undershoot setpoint), and the values of the gains themselves will not help us in that capacity.

Back

Next

Check the Ontology Explorer

Quickly find matching canonical types and missing fields.

1

2

3

Now, if we run the Ontology Explorer we see that there are many canonical types that could be used here.

However, note that none of the match scores are 100%. This means an extension is needed.

Knowing what abstract types we expect to see, we can see that match number 7. `AHU_DFSS_DSP_DXZC_ECONZ` is the example we think should be extended, so let's compare the fields we have to those we want on the type.

```
Command Prompt
C:\[redacted]\Documents\GitHub\digitalbuildings\tools\explorer>python explorer.py
Starting DBO explorer...

How would you like to query DBO
1: Get fields for a type name
2: Get types for a list of fields
```

```
Please select an option: 2
Enter your fields here as a comma separated list: outside_air_ten
tage_command,discharge_air_temperature_sensor,zone_air_cooling_te
un_status,zone_air_co2_concentration_sensor,zone_air_co2_concentr
ensor
1. AHU_CO2C_DFSS_DX2ZC_ECONZ_HT2ZC -- score:96
2. AHU_CO2C_DFSS_DX2ZC_ECONZ_HTZC -- score:96
3. AHU_CO2M_DFSS_DSP_DX2ZC_ECONZ -- score:96
4. AHU_CO2C2X_DFSS_DSP_DXZC_ECONZ_HTZC -- score:96
5. AHU_CO2C_DFSS_DSP_ECONZ_HPZC -- score:96
6. AHU_CO2C_DFSS_DSP_DXZC_ECONZ_HTZC -- score:96
7. AHU_DFSS_DSP_DXZC_ECONZ -- score:93
8. AHU_DFSS_DSP_DX2ZC_ECONZ -- score:93
9. AHU_DFSS_DSP_DX4ZC_ECONZ -- score:93
10. AHU_CO2C_DFSS_DSP_DX2DC_ECOND_HTDC -- score:93
```

Back

Note: Revisit [Lesson 3](#) for help using the Ontology Explorer.

Next

Check the Ontology Explorer (continued)

Quickly find matching canonical types and missing fields.

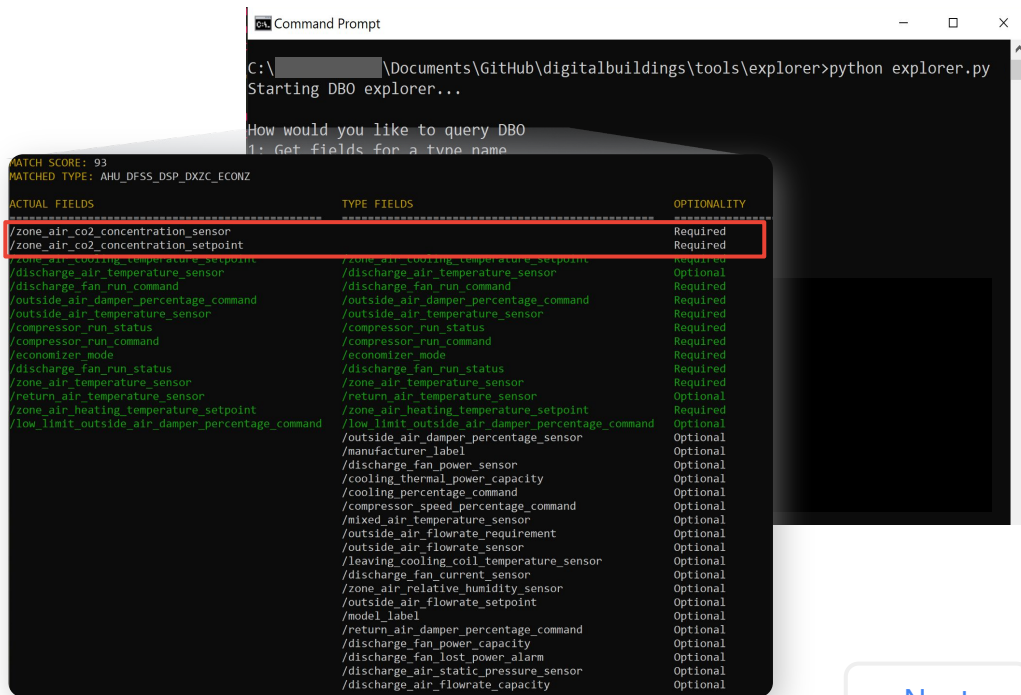
1

2

3

Let's compare the fields we have to those we want on the type.

Note how the CO₂ fields are missing. We'll need to create a new type to include `co2c`, and this would be the type we would apply to this entity.



Command Prompt

```
C:\[redacted]\Documents\GitHub\digitalbuildings\tools\explorer>python explorer.py
Starting DBO explorer...

How would you like to query DBO
1: Get fields for a type name
```

MATCH SCORE: 93
MATCHED TYPE: AHU_DFSS_DSP_DXZC_ECONZ

ACTUAL FIELDS	TYPE FIELDS	OPTIONALITY
/zone_air_co2_concentration_sensor		Required
/zone_air_co2_concentration_setpoint		Required
/zone_air_heating_temperature_setpoint	/zone_air_heating_temperature_setpoint	Optional
/discharge_air_temperature_sensor	/discharge_air_temperature_sensor	Optional
/discharge_fan_run_command	/discharge_fan_run_command	Required
/outside_air_damper_percentage_command	/outside_air_damper_percentage_command	Required
/outside_air_temperature_sensor	/outside_air_temperature_sensor	Required
/compressor_run_status	/compressor_run_status	Required
/compressor_run_command	/compressor_run_command	Required
/economizer_mode	/economizer_mode	Required
/discharge_fan_run_status	/discharge_fan_run_status	Required
/zone_air_temperature_sensor	/zone_air_temperature_sensor	Required
/return_air_temperature_sensor	/return_air_temperature_sensor	Optional
/zone_air_heating_temperature_setpoint	/zone_air_heating_temperature_setpoint	Required
/low_limit_outside_air_damper_percentage_command	/low_limit_outside_air_damper_percentage_command	Optional
	/outside_air_damper_percentage_sensor	Optional
	/manufacturer_label	Optional
	/discharge_fan_power_sensor	Optional
	/cooling_thermal_power_capacity	Optional
	/cooling_percentage_command	Optional
	/compressor_speed_percentage_command	Optional
	/mixed_air_temperature_sensor	Optional
	/outside_air_flowrate_requirement	Optional
	/outside_air_flowrate_sensor	Optional
	/leaving_cooling_coil_temperature_sensor	Optional
	/discharge_fan_current_sensor	Optional
	/zone_air_relative_humidity_sensor	Optional
	/outside_air_flowrate_setpoint	Optional
	/model_label	Optional
	/return_air_damper_percentage_command	Optional
	/discharge_fan_power_capacity	Optional
	/discharge_fan_lost_power_alarm	Optional
	/discharge_air_static_pressure_sensor	Optional
	/discharge_air_flowrate_capacity	Optional

Back

Next

Check the Ontology Explorer (continued)

Quickly find matching canonical types and missing fields.

1

2

3

Here's the modified type definition.

We'll go into detail about how this gets added to DBO in the next lesson. For now, this is the thought process that would lead you to an extension.

```
AHU_DFSS_DSP_DXZC_ECONZ_CO2C:  
  description: "Single zone AHU."  
  is_canonical: true  
  implements:  
    - AHU  
    - DFSS  
    - DSP  
    - DXZC  
    - ECONZ  
    - CO2C
```

[Back](#)

Command Prompt

```
C:\[redacted]\Documents\GitHub\digitalbuildings\tools\explorer>python explorer.py  
Starting DBO explorer...  
  
How would you like to query DBO  
1: Get fields for a type name
```

MATCH SCORE: 93
MATCHED TYPE: AHU_DFSS_DSP_DXZC_ECONZ

ACTUAL FIELDS	TYPE FIELDS	OPTIONALITY
/zone_air_co2_concentration_sensor	/zone_air_co2_concentration_sensor	Required
/zone_air_co2_concentration_setpoint	/zone_air_co2_concentration_setpoint	Required
/zone_air_heating_temperature_setpoint	/zone_air_heating_temperature_setpoint	Optional
/discharge_air_temperature_sensor	/discharge_air_temperature_sensor	Optional
/discharge_fan_run_command	/discharge_fan_run_command	Required
/outside_air_damper_percentage_command	/outside_air_damper_percentage_command	Required
/outside_air_temperature_sensor	/outside_air_temperature_sensor	Required
/compressor_run_status	/compressor_run_status	Required
/compressor_run_command	/compressor_run_command	Required
/economizer_mode	/economizer_mode	Required
/discharge_fan_run_status	/discharge_fan_run_status	Required
/zone_air_temperature_sensor	/zone_air_temperature_sensor	Required
/return_air_temperature_sensor	/return_air_temperature_sensor	Optional
/zone_air_heating_temperature_setpoint	/zone_air_heating_temperature_setpoint	Required
/low_limit_outside_air_damper_percentage_command	/low_limit_outside_air_damper_percentage_command	Optional
	/outside_air_damper_percentage_sensor	Optional
	/manufacturer_label	Optional
	/discharge_fan_power_sensor	Optional
	/cooling_thermal_power_capacity	Optional
	/cooling_percentage_command	Optional
	/compressor_speed_percentage_command	Optional
	/mixed_air_temperature_sensor	Optional
	/outside_air_flowrate_requirement	Optional
	/outside_air_flowrate_sensor	Optional
	/leaving_cooling_coil_temperature_sensor	Optional
	/discharge_fan_current_sensor	Optional
	/zone_air_relative_humidity_sensor	Optional
	/outside_air_flowrate_setpoint	Optional
	/model_label	Optional
	/return_air_damper_percentage_command	Optional
	/discharge_fan_power_capacity	Optional
	/discharge_fan_lost_power_alarm	Optional
	/discharge_air_static_pressure_sensor	Optional
	/discharge_air_flowrate_capacity	Optional

[Next](#)

Repeat to name each required data point

To rough-in the names of the required data points, you'll repeat these steps for every data point that needs to be included in the building model.

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

Prepare a new rough-in sheet

Fill in the "Namespace" column

Fill in the "General type" column

Fill in the "Entity name" column

Fill in the "Field name" column

Check the Ontology Explorer



[Back](#)

[Next](#)

Repeat to name each required data point

To rough-in the names of the required data points, you'll repeat these steps for every data point that needs to be included in the building model.

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

Prepare a new rough-in sheet

Fill in the "Namespace" column

Fill in the "General type" column

Fill in the "Entity name" column

Fill in the "Field name" column

Check the Ontology Explorer

Preparing a new rough-in sheet

You'll only need to do this once, since your rough-in model will include all of the data points you plan to include in the building configuration file you'll create later.

To prepare a new rough-in sheet:

1. Create a new spreadsheet [in your Google Drive](#) or using your preferred application.
2. Set up the following column headings:
 - Namespace
 - General type
 - Entity name
 - Field name
3. Below the column headings, designate each row as a separate data point.

Rough-in sheet

Namespace	General type	Entity name	Field name

Back

Next

Repeat to name each required data point

To rough-in the names of the required data points, you'll repeat these steps for every data point that needs to be included in the building model.

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

Prepare a new rough-in sheet

Fill in the "Namespace" column

Fill in the "General type" column

Fill in the "Entity name" column

Fill in the "Field name" column

Check the Ontology Explorer

Filling in the "Namespace" column

Under the "Namespace" column, input the application domain of the logical device that generates the row's data point.

Refer to:

- Your list of required devices to model (from Lesson 2)
- GENERALTYPES.yaml in each DBO namespace

Rough-in sheet

Namespace	General type	Entity name	Field name

Back

Next

Repeat to name each required data point

To rough-in the names of the required data points, you'll repeat these steps for every data point that needs to be included in the building model.

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

Prepare a new rough-in sheet

Fill in the "Namespace" column

Fill in the "General type" column

Fill in the "Entity name" column

Fill in the "Field name" column

Check the Ontology Explorer

Filling in the "General type" column

Under the "General type" column, input the general entity type of the entity that will be associated with the logical device that generates the row's data point.

Refer to:

- Your list of required devices to model (from Lesson 2)
- Your list of general types for each device to model (from Lesson 3)
- GENERALTYPES.yaml in each DBO namespace

Rough-in sheet

Namespace	General type	Entity name	Field name

Back

Next

Repeat to name each required data point

To rough-in the names of the required data points, you'll repeat these steps for every data point that needs to be included in the building model.

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

Prepare a new rough-in sheet

Fill in the "Namespace" column

Fill in the "General type" column

Fill in the "Entity name" column

Fill in the "Field name" column

Check the Ontology Explorer

Filling in the "Entity name" column

Under the "Entity name" column, input the name of the entity that will be associated with the logical device that generates the row's data point.

Refer to:

- Your list of required devices to model (from Lesson 2)
- Your list of general types for each device to model (from Lesson 3)
- GENERALTYPES.yaml in each DBO namespace

Rough-in sheet

Namespace	General type	Entity name	Field name

Back

Next

Repeat to name each required data point

To rough-in the names of the required data points, you'll repeat these steps for every data point that needs to be included in the building model.

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

Prepare a new rough-in sheet

Fill in the "Namespace" column

Fill in the "General type" column

Fill in the "Entity name" column

Fill in the "Field name" column

Check the Ontology Explorer

Filling in the "Field name" column

Under the "Field name" column, input the field associated with a data point that would be generated by a reporting device. If it's possible for the device to generate multiple points of data, determine which device data is required and input their field name in separate rows under the "Field name" column.

Refer to:

- Your list of abstract types and field associations for each device to model (from Lesson 3)
- Your list of required device data for each device to model (from Lesson 3)
- ABSTRACT.yaml in each DBO namespace
- Canonical type .yaml corresponding with general types
- fields.yaml in global namespace

Rough-in sheet

Namespace	General type	Entity name	Field name

[Back](#)

[Next](#)

Repeat to name each required data point

To rough-in the names of the required data points, you'll repeat these steps for every data point that needs to be included in the building model.

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

Prepare a new rough-in sheet

Fill in the "Namespace" column

Fill in the "General type" column

Fill in the "Entity name" column

Fill in the "Field name" column

Check the Ontology Explorer

Check the Ontology Explorer for matching canonical types and missing fields

From your terminal or command prompt, run the Ontology Explorer using the following command:

```
path/to/directory/explorer/is/kept>python exporer.py
```

Make sure the `path/` points to the correct path to the Ontology Explorer on your device.

Then:

1. Enter "2" on the first prompt.
2. Enter the fields you'd like to check on the second prompt, separating multiple fields with a comma.
You can either paste them or enter them manually.

These fields should come from your rough-in's "Field name" column.

After retrieving matching canonical types, you can inspect one of the types from the match list to see what fields are missing from your field set.

When prompted:

1. Enter "y" to see field comparisons for these matches.
2. Enter the match number you'd like to inspect to see what fields are missing.

Back

Next

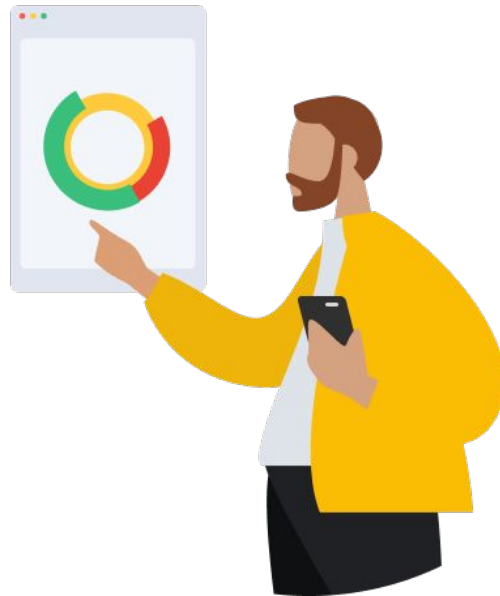
Lesson 4 summary

Let's review what you learned about:

- Rough-in models

Now you should be able to:

- Prepare a new spreadsheet for a rough-in model
- Rough-in the names of required data points



[Back](#)

[Next](#)

You completed Lesson 4!

Now's a great time to take a quick break before starting Lesson 5.

Ready for Lesson 5?

Let's go!

Back

Helpful resources

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

- [Digital Buildings Project GitHub](#)
Contains source code, tooling, and documentation for the DBO.
- [digitalbuildings / ontology / docs / model.md](#)
Describes the conventions used in the DBO concrete model.
- [digitalbuildings / ontology / docs / model_hvac.md](#)
Outlines the best practices for modeling things in the HVAC namespace.