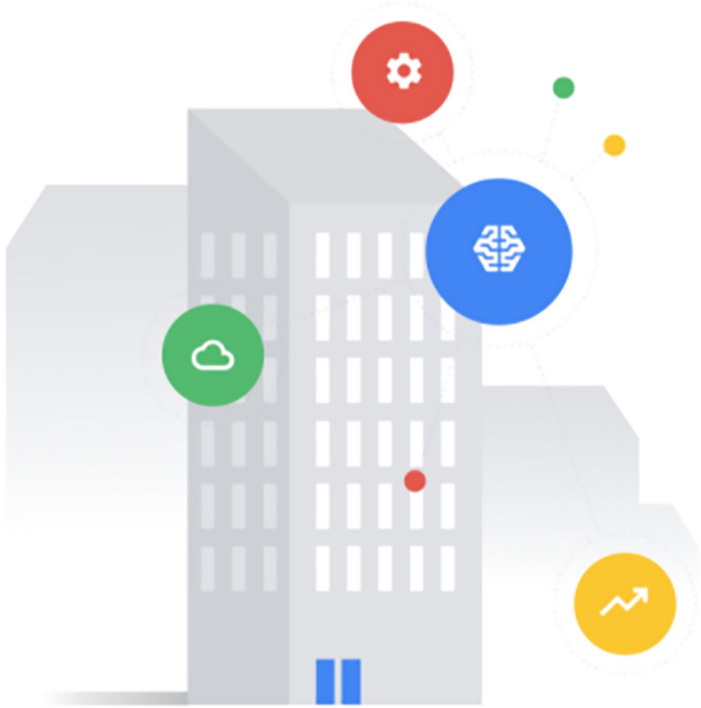
Google

Module 2 | **Lesson 3**

# 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.

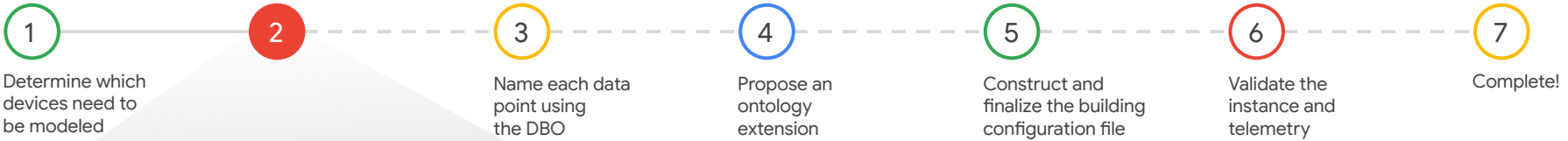
- For the best learning experience, using your keyboard or mouse wheel to navigate is discouraged. However, this is your only option if you're viewing from GitHub.
- If you're viewing this content outside of GitHub:
  - Click the **Back** or **Next** buttons to go backward or forward in the deck. Moving 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 second step of data modeling with the DBO.




## Determine which data points are required

You'll individually examine each logical entity that needs to be modeled using the project documents you received. You'll differentiate between important data points that are required by your project from others that are optional.

Back

Next

Google

Lesson 3

# Determine which data points are required

## What you'll learn about:

- General types and device categories
- Canonical types and device descriptions
- Abstract types and device functions
- Ontology Explorer
- Field associations and device data

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

- Identify the general type of a logical entity.
- Identify a likely canonical type of a logical entity.
- Identify the likely abstract types a logical entity will use.
- Use the Ontology Explorer to check the Digital Buildings Ontology (DBO) for required fields.
- Identify the required data points of a logical entity.

Back

Next

Google

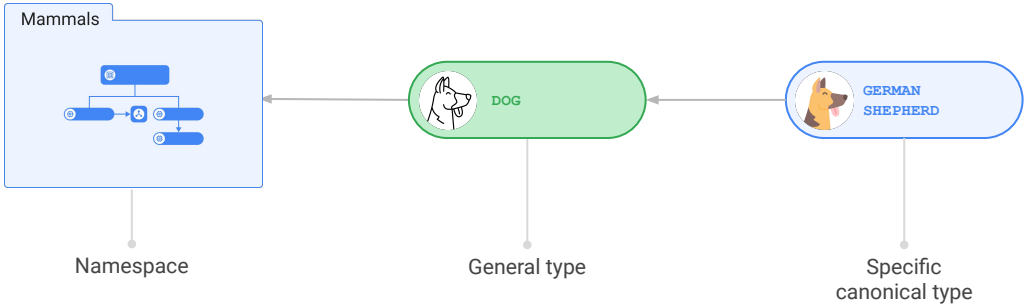
# What's the general type of an entity?

Each **logical entity** you'll model has a **general type.**

## What's a general type?

A **general type** is a curated entity type that defines an entity's broad categorization rather than its specific functionality. The idea of a general type is to group equipment–which may be very different from each other in terms of specific functionality–into broad categories.

To use an analogy: if the specific type were a German Shepherd, then the general type would be a Dog and it would live in the namespace Mammals.




### Examples

Some examples from the HVAC namespace include:

- Air handling units (`AHU`): An air-side device that provides air to a zone directly or indirectly via terminal units, providing recirculated and fresh air. It must have both outside air and return air capabilities to be considered part of this class.
- Fans (`FAN`): A stand-alone, air-side device that moves air from one location to another. It isn't a subcomponent of another device like an AHU.
- Heating water systems (`HWS`): A system that produces hot water for the purpose of providing heat to a system.

See the Digital Buildings GitHub repo for an outline of all HVAC general types.

Back

**Note:** Revisit Module 1, Lesson 6: Entity Types for more information about entity types.

Next

# What's the general type of an entity? (continued)

Knowing the **general type** will help you determine functions that are usually associated with the type of device you're modeling.

It should be fairly intuitive for you to determine a logical entity's general type based on your existing domain knowledge, but you should be mindful of DBO-specific terms and definitions.

See the Digital Buildings GitHub repo for an outline of all HVAC general types.

## How to determine the general type

1. Understand the definitions for general equipment types in each namespace. This is very important: definitions will help to determine whether certain equipment is of the type that it appears to be.
2. Inspect one of the logical entities that you identified from the project documents.
   - Need help with that? Revisit Lesson 2: Determine which devices need to be modeled.
3. Identify notable characteristics, subcomponents, and assumed behaviors about the device.
4. Match your findings with a general type in the global.yaml or one of the child namespaces:
   - ELECTRICAL/…/GENERALTYPES.yaml
   - HVAC/…/GENERALTYPES.yaml
   - LIGHTING/…/GENERALTYPES.yaml
   - METERS/…/GENERALTYPES.yaml
   - SAFETY/…/GENERALTYPES.yaml

You can also refer to the model_hvac.md for descriptions of general types in the HVAC namespace.

Back

Next

Google

Lesson 3

# 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 the concepts and actions that have been introduced so far in this lesson.
- Review the question and select the correct response.
- After this knowledge check, you'll move on to learn about determining abstract types.

**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

In Lesson 2 you learned how to use BMS points lists to identify logical entities. Here's a set of points that might come from one type of device.

## What is the general type of this device?

*Inspect the supplied BMS points list and select the best answer from the options listed below.*

- Air handling unit (`AHU`)
- Boiler (`BLR`)
- Make-up air unit (`MAU`)
- Fan coil unit (`FCU`)
- Terminal unit (`VAV`)

### BMS points list

| Name | Type | Object ID | Device ID | Object Name |
|---|---|---|---|---|
| Return Air Temperature ai | BAI | AI:42 | DEV:2522801 | return_air_temperature_ai_4 |
| Supply Air Temperature 01 ai | BAI | AI:41 | DEV:2522801 | supply_air_temperature_01_ai_3 |
| CHW Control Valve Feedback ai | BAI | AI:19 | DEV:2522801 | chw_control_valve_feedback_ai_4 |
| Cooling Coil Air Temperature ai | BAI | AI:20 | DEV:2522801 | cooling_coil_air_temperature_ai_4 |
| HHW Control Valve Feedback ai | BAI | AI:22 | DEV:2522801 | hhw_control_valve_feedback_ai_4 |
| Supply Fan VFD Feedback ai | BAI | AI:17 | DEV:2522801 | supply_fan_vfd_feedback_ai_4 |
| Outside Air Damper 1 Status ai | BBI | BI:9 | DEV:2522801 | outside_air_damper_1_status_ai_4 |
| Outside Air Damper 2 Status ai | BBI | BI:10 | DEV:2522801 | outside_air_damper_2_status_ai_4 |
| CHW Control Valve Command ao | BAO | AO:8 | DEV:2522801 | chw_control_valve_command_ao_4 |
| HHW Control Valve Command ao | BAO | AO:9 | DEV:2522801 | hhw_control_valve_command_ao_4 |
| Outside Air Damper 1 Command ao | BAO | AO:10 | DEV:2522801 | outside_air_damper_1_command_ao_4 |
| Outside Air Damper 2 Command ao | BAO | AO:6 | DEV:2522801 | outside_air_damper_2_command_ao_4 |
| Supply Fan VFD Speed Command ao | BAO | AO:7 | DEV:2522801 | supply_fan_vfd_speed_command_ao_4 |

Back

**Tip:** Try to identify notable characteristics, subcomponents, and behaviors from the BMS points list. Since this is a type of HVAC equipment, you can check the model_hvac.md or the HVAC/.../GENERALTYPES.yaml to find a general type that matches what you identify.

Next

# Knowledge check 1

In Lesson 2 you learned how to use BMS points lists to identify logical entities. Here's a set of points that might come from one type of device.

**What is the general type of this device?**

*Inspect the supplied BMS points list and select the best answer from the options listed below.*

- Air handling unit (`AHU`)
- Boiler (`BLR`)
- Make-up air unit (`MAU`)
- Fan coil unit (`FCU`)
- Terminal unit (`VAV`)

## That's right! 🎉

The general type of this device is an **air handling unit (`AHU`)**.

On the next slide, we'll explain why this is the only logical choice.

BMS points list

| Name | Type | Object ID | Device ID | Object Name |
|---|---|---|---|---|
| Return Air Temperature ai | BAI | AI:42 | DEV:2522801 | return_air_temperature_ai_4 |
| Supply Air Temperature 01 ai | BAI | AI:41 | DEV:2522801 | supply_air_temperature_01_ai_3 |
| CHW Control Valve Feedback ai | BAI | AI:19 | DEV:2522801 | chw_control_valve_feedback_ai_4 |
| Cooling Coil Air Temperature ai | BAI | AI:20 | DEV:2522801 | cooling_coil_air_temperature_ai_4 |
| HHW Control Valve Feedback ai | BAI | AI:22 | DEV:2522801 | hhw_control_valve_feedback_ai_4 |
| Supply Fan VFD Feedback ai | BAI | AI:17 | DEV:2522801 | supply_fan_vfd_feedback_ai_4 |
| Outside Air Damper 1 Status ai | BBI | BI:9 | DEV:2522801 | outside_air_damper_1_status_ai_4 |
| Outside Air Damper 2 Status ai | BBI | BI:10 | DEV:2522801 | outside_air_damper_2_status_ai_4 |
| CHW Control Valve Command ao | BAO | AO:8 | DEV:2522801 | chw_control_valve_command_ao_4 |
| HHW Control Valve Command ao | BAO | AO:9 | DEV:2522801 | hhw_control_valve_command_ao_4 |
| Outside Air Damper 1 Command ao | BAO | AO:10 | DEV:2522801 | outside_air_damper_1_command_ao_4 |
| Outside Air Damper 2 Command ao | BAO | AO:6 | DEV:2522801 | outside_air_damper_2_command_ao_4 |
| Supply Fan VFD Speed Command ao | BAO | AO:7 | DEV:2522801 | supply_fan_vfd_speed_command_ao_4 |

Back

Next

# Knowledge check 1

In Lesson 2 you learned how to use BMS points lists to identify logical entities. Here's a set of points that might come from one type of device.

**What is the general type of this device?**

*Inspect the supplied BMS points list and select the best answer from the options listed below.*

Air handling unit (`AHU`)

Boiler (`BLR`)

Make-up air unit (`MAU`)

Fan coil unit (`FCU`)

Terminal unit (`VAV`)

Back

## Close... but not quite right! 🤔

The general type of this device isn't a **boiler (`BLR`)**.

We know this because the BMS points list indicates this is an air-side device. This rules out it being a `BLR` based on how it's described in the model_hvac.md and HVAC/.../GENERALTYPES.yaml.

BMS points list

| Name | Type | Object ID | Device ID | Object Name |
|---|---|---|---|---|
| Return Air Temperature ai | BAI | AI:42 | DEV:2522801 | return_air_temperature_ai_4 |
| Supply Air Temperature 01 ai | BAI | AI:41 | DEV:2522801 | supply_air_temperature_01_ai_3 |
| CHW Control Valve Feedback ai | BAI | AI:19 | DEV:2522801 | chw_control_valve_feedback_ai_4 |
| Cooling Coil Air Temperature ai | BAI | AI:20 | DEV:2522801 | cooling_coil_air_temperature_ai_4 |
| HHW Control Valve Feedback ai | BAI | AI:22 | DEV:2522801 | hhw_control_valve_feedback_ai_4 |
| Supply Fan VFD Feedback ai | BAI | AI:17 | DEV:2522801 | supply_fan_vfd_feedback_ai_4 |
| Outside Air Damper 1 Status ai | BBI | BI:9 | DEV:2522801 | outside_air_damper_1_status_ai_4 |
| Outside Air Damper 2 Status ai | BBI | BI:10 | DEV:2522801 | outside_air_damper_2_status_ai_4 |
| CHW Control Valve Command ao | BAO | AO:8 | DEV:2522801 | chw_control_valve_command_ao_4 |
| HHW Control Valve Command ao | BAO | AO:9 | DEV:2522801 | hhw_control_valve_command_ao_4 |
| Outside Air Damper 1 Command ao | BAO | AO:10 | DEV:2522801 | outside_air_damper_1_command_ao_4 |
| Outside Air Damper 2 Command ao | BAO | AO:6 | DEV:2522801 | outside_air_damper_2_command_ao_4 |
| Supply Fan VFD Speed Command ao | BAO | AO:7 | DEV:2522801 | supply_fan_vfd_speed_command_ao_4 |

Try again

Next

# Knowledge check 1

In Lesson 2 you learned how to use BMS points lists to identify logical entities. Here's a set of points that might come from one type of device.

**What is the general type of this device?**

*Inspect the supplied BMS points list and select the best answer from the options listed below.*

- Air handling unit (`AHU`)
- Boiler (`BLR`)
- Make-up air unit (`MAU`)
- Fan coil unit (`FCU`)
- Terminal unit (`VAV`)

Back

## Close... but not quite right! 🤔

The general type of this device isn't a **make-up air unit (`MAU`)**.

We know this because the BMS points list indicates there is a return air present on the device. This rules out it being a `MAU` based on how it's described in the model_hvac.md and HVAC/.../GENERALTYPES.yaml.

BMS points list

| Name | Type | Object ID | Device ID | Object Name |
|---|---|---|---|---|
| Return Air Temperature ai | BAI | AI:42 | DEV:2522801 | return_air_temperature_ai_4 |
| Supply Air Temperature 01 ai | BAI | AI:41 | DEV:2522801 | supply_air_temperature_01_ai_3 |
| CHW Control Valve Feedback ai | BAI | AI:19 | DEV:2522801 | chw_control_valve_feedback_ai_4 |
| Cooling Coil Air Temperature ai | BAI | AI:20 | DEV:2522801 | cooling_coil_air_temperature_ai_4 |
| HHW Control Valve Feedback ai | BAI | AI:22 | DEV:2522801 | hhw_control_valve_feedback_ai_4 |
| Supply Fan VFD Feedback ai | BAI | AI:17 | DEV:2522801 | supply_fan_vfd_feedback_ai_4 |
| Outside Air Damper 1 Status ai | BBI | BI:9 | DEV:2522801 | outside_air_damper_1_status_ai_4 |
| Outside Air Damper 2 Status ai | BBI | BI:10 | DEV:2522801 | outside_air_damper_2_status_ai_4 |
| CHW Control Valve Command ao | BAO | AO:8 | DEV:2522801 | chw_control_valve_command_ao_4 |
| HHW Control Valve Command ao | BAO | AO:9 | DEV:2522801 | hhw_control_valve_command_ao_4 |
| Outside Air Damper 1 Command ao | BAO | AO:10 | DEV:2522801 | outside_air_damper_1_command_ao_4 |
| Outside Air Damper 2 Command ao | BAO | AO:6 | DEV:2522801 | outside_air_damper_2_command_ao_4 |
| Supply Fan VFD Speed Command ao | BAO | AO:7 | DEV:2522801 | supply_fan_vfd_speed_command_ao_4 |

Try again

Next

Google

# Knowledge check 1

In Lesson 2 you learned how to use BMS points lists to identify logical entities. Here's a set of points that might come from one type of device.

**What is the general type of this device?**

*Inspect the supplied BMS points list and select the best answer from the options listed below.*

- Air handling unit (`AHU`)
- Boiler (`BLR`)
- Make-up air unit (`MAU`)
- Fan coil unit (`FCU`)
- Terminal unit (`VAV`)

Back

## Close... but not quite right! 🤔

The general type of this device isn't a **fan coil unit (`FCU`)**.

We know this because the BMS points list indicates this device handles outside air. This rules out it being a `FCU` based on how it's defined in the model_hvac.md and HVAC/.../GENERALTYPES.yaml.

BMS points list

| Name | Type | Object ID | Device ID | Object Name |
|---|---|---|---|---|
| Return Air Temperature ai | BAI | AI:42 | DEV:2522801 | return_air_temperature_ai_4 |
| Supply Air Temperature 01 ai | BAI | AI:41 | DEV:2522801 | supply_air_temperature_01_ai_3 |
| CHW Control Valve Feedback ai | BAI | AI:19 | DEV:2522801 | chw_control_valve_feedback_ai_4 |
| Cooling Coil Air Temperature ai | BAI | AI:20 | DEV:2522801 | cooling_coil_air_temperature_ai_4 |
| HHW Control Valve Feedback ai | BAI | AI:22 | DEV:2522801 | hhw_control_valve_feedback_ai_4 |
| Supply Fan VFD Feedback ai | BAI | AI:17 | DEV:2522801 | supply_fan_vfd_feedback_ai_4 |
| Outside Air Damper 1 Status ai | BBI | BI:9 | DEV:2522801 | outside_air_damper_1_status_ai_4 |
| Outside Air Damper 2 Status ai | BBI | BI:10 | DEV:2522801 | outside_air_damper_2_status_ai_4 |
| CHW Control Valve Command ao | BAO | AO:8 | DEV:2522801 | chw_control_valve_command_ao_4 |
| HHW Control Valve Command ao | BAO | AO:9 | DEV:2522801 | hhw_control_valve_command_ao_4 |
| Outside Air Damper 1 Command ao | BAO | AO:10 | DEV:2522801 | outside_air_damper_1_command_ao_4 |
| Outside Air Damper 2 Command ao | BAO | AO:6 | DEV:2522801 | outside_air_damper_2_command_ao_4 |
| Supply Fan VFD Speed Command ao | BAO | AO:7 | DEV:2522801 | supply_fan_vfd_speed_command_ao_4 |

Try again

Next

# Knowledge check 1

In Lesson 2 you learned how to use BMS points lists to identify logical entities. Here's a set of points that might come from one type of device.

**What is the general type of this device?**

*Inspect the supplied BMS points list and select the best answer from the options listed below.*

Air handling unit (`AHU`)

Boiler (`BLR`)

Make-up air unit (`MAU`)

Fan coil unit (`FCU`)

Terminal unit (`VAV`)

Back

## Close... but not quite right! 🤔

The general type of this device isn't a **terminal unit (`VAV`)**.

We know this because the BMS points list indicates this device is missing things that are typical for a `VAV` like dampers, flow rates, and zone temperature sensors. This rules out it being a `VAV` based on how it's described in the model_hvac.md and HVAC/.../GENERALTYPES.yaml.

BMS points list

| Name | Type | Object ID | Device ID | Object Name |
|---|---|---|---|---|
| Return Air Temperature ai | BAI | AI:42 | DEV:2522801 | return_air_temperature_ai_4 |
| Supply Air Temperature 01 ai | BAI | AI:41 | DEV:2522801 | supply_air_temperature_01_ai_3 |
| CHW Control Valve Feedback ai | BAI | AI:19 | DEV:2522801 | chw_control_valve_feedback_ai_4 |
| Cooling Coil Air Temperature ai | BAI | AI:20 | DEV:2522801 | cooling_coil_air_temperature_ai_4 |
| HHW Control Valve Feedback ai | BAI | AI:22 | DEV:2522801 | hhw_control_valve_feedback_ai_4 |
| Supply Fan VFD Feedback ai | BAI | AI:17 | DEV:2522801 | supply_fan_vfd_feedback_ai_4 |
| Outside Air Damper 1 Status ai | BBI | BI:9 | DEV:2522801 | outside_air_damper_1_status_ai_4 |
| Outside Air Damper 2 Status ai | BBI | BI:10 | DEV:2522801 | outside_air_damper_2_status_ai_4 |
| CHW Control Valve Command ao | BAO | AO:8 | DEV:2522801 | chw_control_valve_command_ao_4 |
| HHW Control Valve Command ao | BAO | AO:9 | DEV:2522801 | hhw_control_valve_command_ao_4 |
| Outside Air Damper 1 Command ao | BAO | AO:10 | DEV:2522801 | outside_air_damper_1_command_ao_4 |
| Outside Air Damper 2 Command ao | BAO | AO:6 | DEV:2522801 | outside_air_damper_2_command_ao_4 |
| Supply Fan VFD Speed Command ao | BAO | AO:7 | DEV:2522801 | supply_fan_vfd_speed_command_ao_4 |

Try again

Next

# Knowledge check 1

## Let's review why this is an `AHU`. 👩‍💻

### How did we know this is an `AHU`?

We know through our understanding of what equipment is by definition and by precedent set in the DBO.

### What about the other options?

First, we know it is air side, so this rules out the boiler (`BLR`).

Second, we see it has much more than is typical for a terminal unit (`VAV`). It's missing dampers, flow rates, zone temperature sensors, etc. and doesn't really look like a VAV, so it can be ruled out.

Third, we know it is not a fan coil unit (`FCU`) by definition, because it handles outside air.

Finally, we can see that return air is also present on the device, which rules out the make-up air unit (`MAU`), again by definition.

### Of the options provided, `AHU` was the only logical choice!

### BMS points list

The DBO defines an `AHU` as "an air-side device that provides air to a zone directly or indirectly via terminal units, providing recirculated and fresh air. It must have both outside air and return air capabilities to be considered part of this class."

The points list indicates this device handles both return air and outside air which is exactly how an `AHU` is defined in the model_hvac.md and HVAC/.../GENERALTYPES.yaml.

| Name | Type | Object ID | Device ID | Object Name |
|---|---|---|---|---|
| Return Air Temperature ai | BAI | AI:42 | DEV:2522801 | return_air_temperature_ai_4 |
| Supply Air Temperature 01 ai | BAI | AI:41 | DEV:2522801 | supply_air_temperature_01_ai_3 |
| CHW Control Valve Feedback ai | BAI | AI:19 | DEV:2522801 | chw_control_valve_feedback_ai_4 |
| Cooling Coil Air Temperature ai | BAI | AI:20 | DEV:2522801 | cooling_coil_air_temperature_ai_4 |
| HHW Control Valve Feedback ai | BAI | AI:22 | DEV:2522801 | hhw_control_valve_feedback_ai_4 |
| Supply Fan VFD Feedback ai | BAI | AI:17 | DEV:2522801 | supply_fan_vfd_feedback_ai_4 |
| Outside Air Damper 1 Status ai | BBI | BI:9 | DEV:2522801 | outside_air_damper_1_status_ai_4 |
| Outside Air Damper 2 Status ai | BBI | BI:10 | DEV:2522801 | outside_air_damper_2_status_ai_4 |
| CHW Control Valve Command ao | BAO | AO:8 | DEV:2522801 | chw_control_valve_command_ao_4 |
| HHW Control Valve Command ao | BAO | AO:9 | DEV:2522801 | hhw_control_valve_command_ao_4 |
| Outside Air Damper 1 Command ao | BAO | AO:10 | DEV:2522801 | outside_air_damper_1_command_ao_4 |
| Outside Air Damper 2 Command ao | BAO | AO:6 | DEV:2522801 | outside_air_damper_2_command_ao_4 |
| Supply Fan VFD Speed Command ao | BAO | AO:7 | DEV:2522801 | supply_fan_vfd_speed_command_ao_4 |

Back

*Click **Next** to complete this knowledge check and move on to canonical and abstract types.*

Next

# What functions are associated with the entity?

Each **general type** has a set of **canonical types** which form a precedent for how equipment is modeled in DBO.

## What's a canonical type?

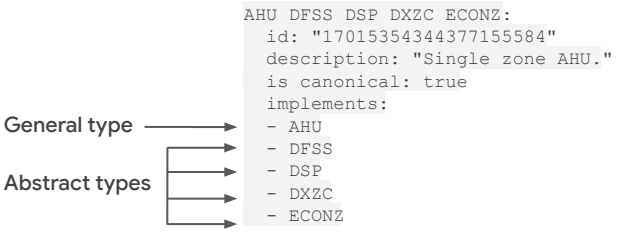
A **canonical type** is a curated entity type that can be mapped to an individual entity. It describes a specific type of device rather than its functionality, behavior, or broader category.

### Examples

`AHU` is a general type.

```
AHU:
  id: "16759731085058768896"
  description: "Tag for air-handling units. AHUs are
    devices which handle air distribution (either to
    a single space or to multiple via ductwork), and
    they have the capability of handling both return
    and outside air, in some combination (0% OA,
    100% RA; 50% OA, 50% RA; 100% OA, 0% RA, etc.);
    that is, they can recirculate or ventilate. They
    are distinct from other devices in key ways:
    FCUs, which handle only recirculated air; MAUs,
    which handle only outside air; and DOAS devices,
    which handle outside and return air, but which
    do not recirculate air. "
  is abstract: true
  implements:
  - EQUIPMENT
  opt uses:
  - outside_air_flowrate_requirement
```

`AHU_DFSS_DSP_DXZC_ECONZ` is a canonical `AHU`.




What makes `AHU_DFSS_DSP_DXZC_ECONZ` a canonical type is that it is a good, common representation for a specific type of equipment that's composed of multiple well-defined subcomponents called **abstract types**.

Back

**Note:** Exploring the **canonical types** in a device's namespace is helpful because you could find a matching entity type that's already been curated for reuse.

Next

# What functions are associated with the entity? (continued)

A **canonical type** combines a general and **abstract types** to represent a specific type of equipment and its functionality.

## What's an abstract type?

An **abstract type** is a curated entity type that can be implemented by canonical types. It's a definition of an entity's specific functionality or behavior using field associations rather than a description or broad category of a device.

### Examples

Here's the abstract type for a carbon dioxide control (`CO2C`) function of a device:

```
CO2C:
  id: "14886233640072642560"
  description: "Carbon dioxide
    control."
  is_abstract: true
  implements:
  - OPERATIONAL
  uses:
  - zone_air_co2_concentration_sensor
  - zone_air_co2_concentration_setpoint
```

When implemented by a canonical type, you know that the entity will measure and control $CO_2$ levels based on the required fields `zone_air_co2_concentration_sensor` and `zone_air_co2_concentration_setpoint`.

This set of fields can easily be analyzed as a group to determine how well the device maintains $CO_2$ levels.

Here is the abstract type for a dual setpoint control (`DSP`) function of a device:

```
DSP:
  id: "8112819800507416576"
  description: "Dual setpoint control (heating/cooling thresholds with
deadband in between)."
  is_abstract: true
  implements:
  - OPERATIONAL
  opt_uses:
  - discharge_air_temperature_sensor
  - zone_air_relative_humidity_sensor
  uses:
  - zone_air_temperature_sensor
  - zone_air_cooling_temperature_setpoint
  - zone_air_heating_temperature_setpoint
```

When implemented by a canonical type, you know that the entity will monitor and control the temperature of a zone based on the required fields `zone_air_temperature_sensor`, `zone_air_cooling_temperature_setpoint`, and `zone_air_heating_temperature_setpoint`.

This set of fields can easily be analyzed as a group to determine how well the device maintains the dual setpoint temperature.

Back

**Note:** Exploring the **canonical types** in a device's namespace is helpful because you could find a matching entity type that's already been curated for reuse.

Next

# What functions are associated with the entity? (continued)

Knowing the **abstract types** will help you determine functions that are usually associated with the type of device you're modeling.

Abstract types make heavy use of field associations, which are good indicators of the type of data points that are required (`uses`) or optional (`opt_uses`) from a reporting device's translated payload. In contrast, general types rarely associate fields. Instead, canonical instances of general types usually implement abstract types to inherit their field associations.

Because abstract types deal directly with fields, you can begin narrowing down required data point types by determining the logical entity's abstract types. See the Digital Buildings GitHub repo for an outline of all HVAC abstract types.

## How to identify possible abstract and canonical types

**After determining the device's general type:**

1. Re-inspect the project documents to identify the notable functions of the device and its subcomponents.

**Then do one or both of the following:**

2. Match the functions you identified with one of the DBO's abstract type definitions in the abstract types file or from the docs in model_hvac.md.
3. Explore the child namespace of the device's general type to find a similar or matching canonical type. What kinds of abstract types do they implement? It can be helpful to compare and see how your device varies from it.

Back

Next

Google

# HVAC abstract types

**To model with the DBO, it's very important to have an understanding of existing abstract types.**
Here are some common categories of abstract types that you might use within the HVAC namespace.
*Click on each category to reveal some of its typical abstract types.*

- Air flow control
- Air temperature control and monitoring
- Air pressure control
- Air quality control
- Fan control
- Mechanical heating and cooling control

Back

Next

# HVAC abstract types

**To model with the DBO, it's very important to have an understanding of existing abstract types.**
Here are some common categories of abstract types that you might use within the HVAC namespace.
*Click on each category to reveal some of its typical abstract types.*

- Air flow control
- Air temperature control and monitoring
- Air pressure control
- Air quality control
- Fan control
- Mechanical heating and cooling control

## Air flow control

- **Single damper flow control** (`SD`, `ED`, `RD`, etc.)
  The most important feature of a terminal unit (`VAV`) is its damper and the controls associated with it, because almost all `VAV`s measure airflow and control their damper to a flow setpoint.
- **Dual duct terminals** (`DD`)
  In older systems, it is possible for there to be two separate ducts for heating and cooling air. It requires both sets of damper control points and measurements be available.
- **Outside air damper** (`OADM`, `VOADM`, `MOADM`)
  Outside air dampers modulate to provide fresh outside air to the building for ventilation and to provide cooling air for economization. Position is monitored as a binary open/closed command (`OADM`) or a percentage command (`VOADM`).
- **Outside air flow** (`OAFC`, `OAFMC`, `MOAFC`)
  A flow sensor is mounted in the outside air duct, and the outside air damper modulates to control flow to a setpoint (`OAFC`) or to a minimum flow setpoint (`OAFMC`).

Back

**Note:** See model_hvac.md in the Digital Buildings GitHub repo for an outline of all HVAC abstract types including uncommon categories. It is highly recommended that you both read the documents on GitHub and the abstract type definitions themselves. You can't speak to application with respect to precedent if you have no familiarity with it!

Next

# HVAC abstract types

**To model with the DBO, it's very important to have an understanding of existing abstract types.**
Here are some common categories of abstract types that you might use within the HVAC namespace.
*Click on each category to reveal some of its typical abstract types.*

Air flow control

Air temperature control and monitoring

Air pressure control

Air quality control

Fan control

Mechanical heating and cooling control

## Air temperature control and monitoring

- **Zone temp control** (`ZTC`)
  The zone is maintained to a fixed setpoint and will cool if the zone drifts above the setpoint or heat if it falls below the setpoint. There's often a hard-coded deadband that prevents erratic fluctuation between heating and cooling.
- **Zone temp monitoring** (`ZTM`)
  There is only a zone temperature sensor, and it's not tied to a particular control strategy.
- **Cooling setpoint control** (`CSP`)
  The zone is cooled by the `VAV` only, so no consideration is needed for a heating mode. This is typical of IDF, cable, or electrical rooms. A single lower-bound cooling setpoint is used.
- **Dual setpoint control** (`DSP`)
  The zone maintains between upper- and lower-bounds (cooling and heating setpoint, respectively). The deadband is implied.

Back

**Note:** See model_hvac.md in the Digital Buildings GitHub repo for an outline of all HVAC abstract types including uncommon categories. It is highly recommended that you both read the documents on GitHub and the abstract type definitions themselves. You can't speak to application with respect to precedent if you have no familiarity with it!

Next

# HVAC abstract types

**To model with the DBO, it's very important to have an understanding of existing abstract types.**
Here are some common categories of abstract types that you might use within the HVAC namespace.
*Click on each category to reveal some of its typical abstract types.*

- Air flow control
- Air temperature control and monitoring
- Air pressure control
- Air quality control
- Fan control
- Mechanical heating and cooling control

## Air pressure control

- **Supply air static pressure control** (`SSPC`, `SSPM`)
  Supply air static pressure (in the duct) is either monitored by a pressure sensor (`SSPM`) or controlled by a sensor and setpoint (`SSPC`). Entities control their supply air static pressure through the modulation of their supply fan speed or through the modulation of bypass dampers.
- **Building static pressure** (`BSPM`, `BSPC`)
  Building air static pressure (in the duct) is either monitored by a pressure sensor (`BSPM`) or controlled by a sensor and setpoint (`BSPC`). Entities control building static pressure through the modulation of exhaust fan, exhaust dampers, and in certain instances, outside air dampers.
- **Zone static pressure** (`ZSPM`, `ZSPC`)
  Zone static pressure is either monitored by a pressure sensor (`ZSPM`) or controlled by a sensor and setpoint (`ZSPC`).
- **Exhaust/Return air static pressure** (`ESPC`, `RSPC`)
  Exhaust air (`ESPC`) or return air static pressure (`RSPC`) (in the duct) is controlled by a sensor and setpoint. Entities do not typically control exhaust or return air static pressure.

Back

**Note:** See model_hvac.md in the Digital Buildings GitHub repo for an outline of all HVAC abstract types including uncommon categories. It is highly recommended that you both read the documents on GitHub and the abstract type definitions themselves. You can't speak to application with respect to precedent if you have no familiarity with it!

Next

# HVAC abstract types

**To model with the DBO, it's very important to have an understanding of existing abstract types.**
Here are some common categories of abstract types that you might use within the HVAC namespace.
*Click on each category to reveal some of its typical abstract types.*

- Air flow control
- Air temperature control and monitoring
- Air pressure control
- Air quality control
- Fan control
- Mechanical heating and cooling control

## Air quality control

- **Carbon dioxide control** (`CO2M`, `CO2C`, `CO2C2X`)
  The concentration of carbon dioxide in the zone air is monitored by sensors. For units that control $CO_2$, $CO_2$ levels are used to determine when additional ventilation is required (when levels exceed the setpoint) and increase the ventilation. Some zones have multiple $CO_2$ sensors (`CO2C2X`).
- **Carbon monoxide control** (`COC`)
  Carbon monoxide sensors are used to determine when additional ventilation is required (when levels exceed the setpoint) and increase the ventilation. Typical of parking garages with variable ventilation.
- **Volatile organic compound control** (`VOCM`, `VOCC`)
  Similar to `CO2C`, but the unit monitors or controls zone volatile organic compound levels.

Back

**Note:** See model_hvac.md in the Digital Buildings GitHub repo for an outline of all HVAC abstract types including uncommon categories. It is highly recommended that you both read the documents on GitHub and the abstract type definitions themselves. You can't speak to application with respect to precedent if you have no familiarity with it!

Next

# HVAC abstract types

**To model with the DBO, it's very important to have an understanding of existing abstract types.**
Here are some common categories of abstract types that you might use within the HVAC namespace.
*Click on each category to reveal some of its typical abstract types.*

Air flow control

Air temperature control and monitoring

Air pressure control

Air quality control

Fan control

Mechanical heating and cooling control

## Fan control

- **Fan types** (`SS`, `SFSS`, `DFSS`, `EFSS`, `RFSS`)
  The `SS` type is the basic command (start-stop) and status (feedback) for equipment. It's modified by the appropriate descriptors for the fan type. There are four basic fan types:
  - Supply fans (`SF`) deliver air from the unit to downstream units (such as an `AHU` providing supply air to `VAV`s).
  - Discharge fans (`DF`) deliver air from the unit directly into the zone (without downstream units).
  - Exhaust fans (`EF`) pull air out of the zone and exhaust it out of the building.
  - Return fans (`RF`) draw the air back to the return box of AHUs, predominantly.
  - Some fans have multiple supply or exhaust fans (`SFSS2X`).
- **Variable speed control** (`VSC`, `SFVSC`, `DFSVSC`, `EFVSC`, etc.)
  Some fans and entities have variable speed control through a variable frequency drive (`VFD`) that usually has `run_command` and `run_status` fields. Some fans have multiple supply or exhaust fans (`SFVSC2X`).
- **Mode speed control** (`DFHMC`, `DFHLC`, `DFHML`)
  Some fans control their speed to a discrete set of fixed speed positions (`MC`) rather than a percentage. Some also operate with high and low speed commands (`HLC`) or high, medium, and low speed commands (`HMLC`).

Back

**Note:** See model_hvac.md in the Digital Buildings GitHub repo for an outline of all HVAC abstract types including uncommon categories. It is highly recommended that you both read the documents on GitHub and the abstract type definitions themselves. You can't speak to application with respect to precedent if you have no familiarity with it!

Next

# HVAC abstract types

**To model with the DBO, it's very important to have an understanding of existing abstract types.**
Here are some common categories of abstract types that you might use within the HVAC namespace.
*Click on each category to reveal some of its typical abstract types.*

Air flow control

Air temperature control and monitoring

Air pressure control

Air quality control

Fan control

Mechanical heating and cooling control

## Mechanical heating and cooling control

- **Chilled water valve control** (`CHWSC`, `CHWDC`, etc.)
  Chilled water valve control based on a specific temperature sensor. All iterations of this abstract type include the chilled water valve command, and its associated control temp sensor and setpoint pair (`SC`, `DC`, `ZC`, `ZTC`, `RC`).
- **Direct expansion control** (`DXSC`, `DXDC`, etc.)
  Direct expansion cooling. Like chilled water valves, compressor control to temp sensor and setpoint pairs. A type can have multiple compressors, but it typically doesn't have both a compressor and chilled water coil.
- **Heat pumps** (`HPDC`, `HPRC`, etc.)
  Direct expansion units with reversing valves. The reversing valve allows the refrigeration cycle to run in either direction, so the heat pump can provide either heating or cooling. Heat pump types consist of the reversing valve command, compressor run command, and temp sensor and setpoint pair.
- **Heating water valve control** (`HWZC`, `HWSC`, `HWDC`, etc)
  Contains the heating water valve, temp sensor and setpoint pair.
- **Gas and electric heater control** (`HTSC`, `HTVSC`, etc)
  Gas and electric heaters integral to the unit. Electric heaters have an electric coil in the duct that transfers heat to the air. Gas heaters use natural gas to serve a heat exchanger in the duct. From an operational standpoint, gas and electric are identical.

Back

**Note:** See model_hvac.md in the Digital Buildings GitHub repo for an outline of all HVAC abstract types including uncommon categories. It is highly recommended that you both read the documents on GitHub and the abstract type definitions themselves. You can't speak to application with respect to precedent if you have no familiarity with it!

Next

Lesson 3

# Practice 1

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

- This practice activity will look at another `AHU` similar to the one that you identified earlier in the lesson.
- The next slide will present a BMS points list with a question related to the concepts and actions that have been introduced since the last knowledge check.
- Answer the question on your own and check your answer on the following slide.
- After this practice activity, you'll move on to learn about the Ontology Explorer.

Back

**Tip:** Create a new doc in your Google Drive before starting this practice activity. You can use this doc to write down your answers.

Next

Google

# Practice 1

Earlier, you inspected a BMS points list for an `AHU` general type. Here's a similar `AHU` device that we'll attempt to model.

## Which abstract types could describe the functions and behaviors of this device?

*Follow the steps to determine possible abstract types.*
*Use a separate document to write down your answers.*

### BMS screenshot

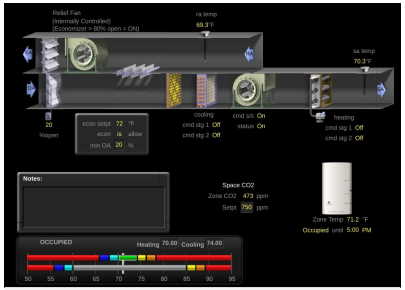



**Image source:** Google's WebCTRL instances. WebCTRL is a building automation system owned by Automated Logic.

**Note:** You can see that many assumptions will need to be made about the control logic in order to complete this. In practice, you'll have better information available to you regarding how things actually control. For the purpose of this exercise, let's assume that the economizer, heating stages, and cooling stages all control directly to zone temperature setpoints.

### Steps

**After determining the device's general type:**

1. Re-inspect the project documents to identify the notable functions of the device and its subcomponents.

**Then do one or both of the following:**

2. Match the functions you identified with one of the DBO's abstract type definitions in the abstract types file or from the docs in model_hvac.md.
3. Explore the child namespace of the device's general type to find a similar or matching canonical type. What kinds of abstract types do they implement? It can be helpful to compare and see how your device varies from it.

### BMS points list

| Name | Type | Object ID | Device ID | Object Name |
|---|---|---|---|---|
| Return Air Temperature ai | BAI | AI:42 | DEV:2528722 | return_air_temperature_ai_4 |
| Discharge Temp | BAI | AI:2 | DEV:2528722 | da_temp_1 |
| Heat Stage 1 | BBI | BI:1 | DEV:2528722 | htg_stg1_bi_1 |
| Heat Stage 2 | BBI | BI:2 | DEV:2528722 | htg_stg2_bi_2 |
| Setpoint / Effective Cooling Setpoint | BAV | AV:7 | DEV:2528722 | Effective Cooling_1 |
| Setpoint / Effective Heating Setpoint | BAV | AV:8 | DEV:2528722 | Effective Heating_1 |
| Zone Temp | ASVI | AI:4 | DEV:2528722 | zone_temp_1 |
| DX Cooling Stage 1 | BBI | BI:9 | DEV:2528722 | dx_clg_stg1_bi_1 |
| DX Cooling Stage 2 | BBI | BI:10 | DEV:2528722 | dx_clg_stg2_bi_1 |
| Econ Min Pos | BAV | AV:13 | DEV:2528722 | ec_min_1 |
| Economizer Setpoint | BAV | AV:14 | DEV:2528722 | ec_setpt_1 |
| Economizer Damper Command | BAO | AO:1 | DEV:2528722 | econ_1 |
| Supply Fan Start/Stop | BBO | BO:1 | DEV:2528722 | sf_cmd_bo_1 |
| Supply Fan Status | BBI | BI:3 | DEV:2528722 | sf_stat_bi_1 |
| Zone CO2 | BAV | AV:32 | DEV:2528722 | co2_1 |
| CO2 Setpoint | BAV | AV:33 | DEV:2528722 | co2_stpt_1 |

Back

*When you're ready, click **Next** to see how we identified the abstract types.*

Next

# Practice 1

## Check your answer! 👩‍🏫

Here are the abstract types we identified that describe the functions of the `AHU`. **Did you come up with these abstract types, too?**

- `DFSS`
- `DSP`
- `ECONZ`
- `DX2ZC`
- `HT2ZC`
- `CO2C`

### BMS screenshot

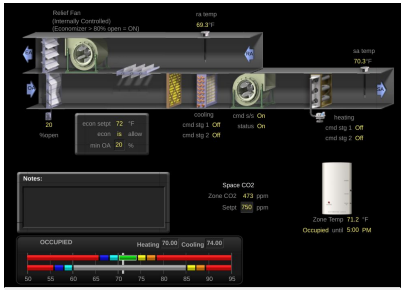



**Image source:** Google's WebCTRL instances. WebCTRL is a building automation system owned by Automated Logic.

### BMS points list

| Name | Type | Object ID | Device ID | Object Name |
|---|---|---|---|---|
| Return Air Temperature ai | BAI | AI:42 | DEV:2528722 | return_air_temperature_ai_4 |
| Discharge Temp | BAI | AI:2 | DEV:2528722 | da_temp_1 |
| Heat Stage 1 | BBI | BI:1 | DEV:2528722 | htg_stg1_bi_1 |
| Heat Stage 2 | BBI | BI:2 | DEV:2528722 | htg_stg2_bi_2 |
| Setpoint / Effective Cooling Setpoint | BAV | AV:7 | DEV:2528722 | Effective Cooling_1 |
| Setpoint / Effective Heating Setpoint | BAV | AV:8 | DEV:2528722 | Effective Heating_1 |
| Zone Temp | ASVI | AI:4 | DEV:2528722 | zone_temp_1 |
| DX Cooling Stage 1 | BBI | BI:9 | DEV:2528722 | dx_clg_stg1_bi_1 |
| DX Cooling Stage 2 | BBI | BI:10 | DEV:2528722 | dx_clg_stg2_bi_1 |
| Econ Min Pos | BAV | AV:13 | DEV:2528722 | ec_min_1 |
| Economizer Setpoint | BAV | AV:14 | DEV:2528722 | ec_setpt_1 |
| Economizer Damper Command | BAO | AO:1 | DEV:2528722 | econ_1 |
| Supply Fan Start/Stop | BBO | BO:1 | DEV:2528722 | sf_cmd_bo_1 |
| Supply Fan Status | BBI | BI:3 | DEV:2528722 | sf_stat_bi_1 |
| Zone CO2 | BAV | AV:32 | DEV:2528722 | co2_1 |
| CO2 Setpoint | BAV | AV:33 | DEV:2528722 | co2_stpt_1 |

Back

*When you're ready, click **Next** to see how we identified the abstract types.*

Next

# Practice 1

Here's how we identified the abstract types.

1 2

## While re-inspecting the BMS points list, the following functions were identified:

- A heating section that appears to control the room temperature (HT2ZC)
- A zone temperature and associated setpoints (DSP)
- A compressor section that appears to control the room temperature (DX2ZC)
- Outside air control and economizer settings (ECONZ)
- A discharge fan with start-stop and feedback (DFSS)
- $CO_2$ level and a setpoint (CO2C)

Since an `AHU` is a type of HVAC device, we checked the model_hvac.md for abstract types that matched the functions we identified and scanned HVAC/AHU for similar canonical types.

### BMS screenshot

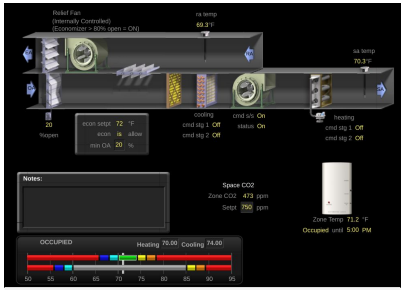


**Image source:** Google's WebCTRL instances. WebCTRL is a building automation system owned by Automated Logic.

**Note:** If you inspect some of the abstract types, you will see that the return temperature is covered by `ECONZ` and discharge temperature is covered by both `HT2ZC` and `DX2ZC`.

### BMS points list

| Name | Type | Object ID | Device ID | Object Name |
|---|---|---|---|---|
| Return Air Temperature ai | BAI | AI:42 | DEV:2528722 | return_air_temperature_ai_4 |
| Discharge Temp | BAI | AI:2 | DEV:2528722 | da_temp_1 |
| Heat Stage 1 | BBI | BI:1 | DEV:2528722 | htg_stg1_bi_1 |
| Heat Stage 2 | BBI | BI:2 | DEV:2528722 | htg_stg2_bi_2 |
| Setpoint / Effective Cooling Setpoint | BAV | AV:7 | DEV:2528722 | Effective Cooling_1 |
| Setpoint / Effective Heating Setpoint | BAV | AV:8 | DEV:2528722 | Effective Heating_1 |
| Zone Temp | ASVI | AI:4 | DEV:2528722 | zone_temp_1 |
| DX Cooling Stage 1 | BBI | BI:9 | DEV:2528722 | dx_clg_stg1_bi_1 |
| DX Cooling Stage 2 | BBI | BI:10 | DEV:2528722 | dx_clg_stg2_bi_1 |
| Econ Min Pos | BAV | AV:13 | DEV:2528722 | ec_min_1 |
| Economizer Setpoint | BAV | AV:14 | DEV:2528722 | ec_setpt_1 |
| Economizer Damper Command | BAO | AO:1 | DEV:2528722 | econ_1 |
| Supply Fan Start/Stop | BBO | BO:1 | DEV:2528722 | sf_cmd_bo_1 |
| Supply Fan Status | BBI | BI:3 | DEV:2528722 | sf_stat_bi_1 |
| Zone CO2 | BAV | AV:32 | DEV:2528722 | co2_1 |
| CO2 Setpoint | BAV | AV:33 | DEV:2528722 | co2_stpt_1 |

Back

*Click **Next** to see how we identified the abstract types.*

Next

# Practice 1

Here's how we identified the abstract types.

1 2

## Since an AHU is a type of HVAC device:

We checked the model_hvac.md for abstract types that matched the functions we identified and scanned HVAC/AHU for similar canonical types.

Here are the identified abstract types:

- `DFSS` - Discharge fan run and status
  - see Fan Control
- `DSP` - Dual setpoint control
  - see Air Temperature Control and Monitoring
- `ECONZ` - Economizer
  - see Economizer Control
- `DX2ZC` - Direct expansion cooling (2 stage)
  - see Mechanical Heating and Cooling Control
- `HT2ZC` - Electric heating (2 stage)
  - see Mechanical Heating and Cooling Control
- `CO2C` - $CO_2$ control
  - see Air Quality Control

And here's a canonical type that appears to similar since it implements some of the abstract types we identified:

```
AHU DFSS DSP DX2ZC ECONZ HT2ZC :
  id: "2675893130829496320"
  description: "Single zone AHU."
  is canonical: true
  implements:
  - AHU
  - DFSS
  - DSP
  - DX2ZC
  - ECONZ
  - HT2ZC
```

However, notice how `CO2C` is missing from this type. This means we'll need to find another type that has all the required abstract types we care about or extend the ontology to support it.

Back

*Click **Next** to complete this practice activity.*

Next

Google

# Before we move on to the next section...

Let's reflect on this practice activity. Were you able to easily find the type `AHU_DFSS_DSP_DX2ZC_ECONZ_HT2ZC`?

**Probably not.**

It's alright. We get that it's a challenge to search for modeling concepts in the ontology. It contains a lot of canonical types, and it can be difficult to find an appropriate type in the with all of the abstract types you care about.

The next section will show you a tool which should make exploring the ontology easier. It's aptly named the **Ontology Explorer**.

Back

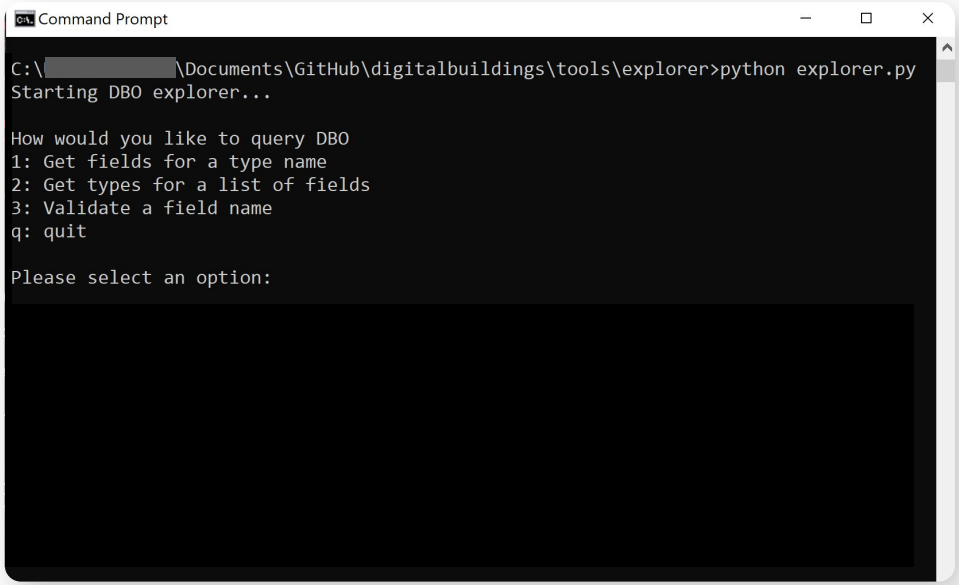
*Click **Next** to move on to the Ontology Explorer.*

Next

Google

# Ontology Explorer

The **Ontology Explorer** is a tool used to ask basic questions about what's already curated within the Digital Buildings Ontology (DBO).




Back

**Note:** As with all tools referenced in these modules, check Github for most up-to-date instructions on how to use the tools. They may change without notice.

Next

Google

# Ontology Explorer

The Ontology Explorer is a tool used to ask basic questions about what's already curated within the Digital Buildings Ontology.

At this point, we're starting to dig into what actually exists in the DBO. You're beginning to see the definitions of abstract and canonical types, and you're getting an idea for what fields are available. However, the inheritance structure within the DBO can make it difficult to see what fields are present on a canonical type when it inherits fields from a set of abstract types.

This can make it challenging to determine which fields–and subsequently which device data–are required or optional to include in your building config.

Thankfully, the Digital Buildings Project has a tool to make this easier for you!

## What's the Ontology Explorer?

The **Ontology Explorer** is a tool that allows you to check if certain fields are valid and whether a set of fields match something that's already defined in the DBO. A simple query lets you see what's defined in the DBO and how your model (no matter how far along you are with it) compares without having to dig through the GitHub repo.

## What's needed to run the Ontology Explorer?

Before attempting to run the Ontology Explorer, you'll need the following installed on your machine:

- A version of Python (see python.org)
- The Ontology Explorer (see instructions)
- The Instance Validator (see instructions)
- The Ontology Validator (see instructions)

Back

Next

# Ontology Explorer (continued)

The **Ontology Explorer** can be run at any time during the data modeling workflow.

## Basic command

You'll run Ontology Explorer from your machine's terminal or command prompt using the following command:

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

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

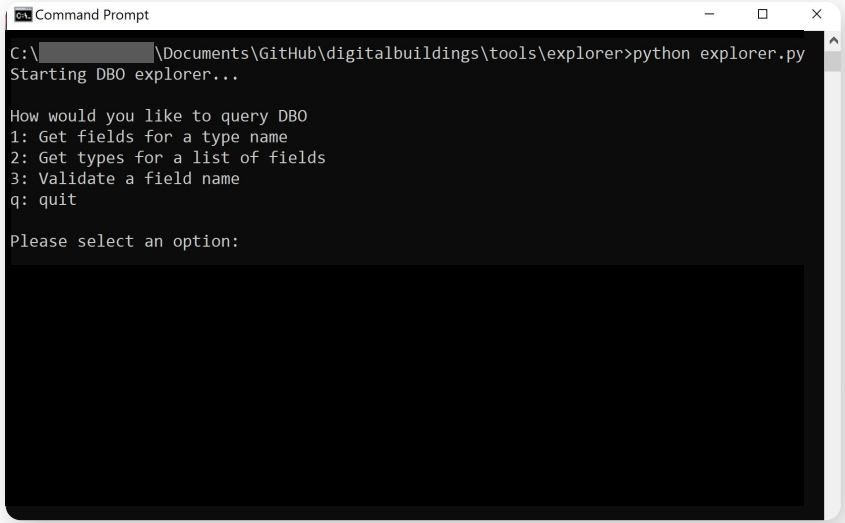
## Options

After starting Ontology Explorer, you'll see four options for actions you can perform:

1: Get fields for a type name
2: Get types for a list of fields
3: Validate a field name
q: quit

Let's explore each one.

Terminal




Back

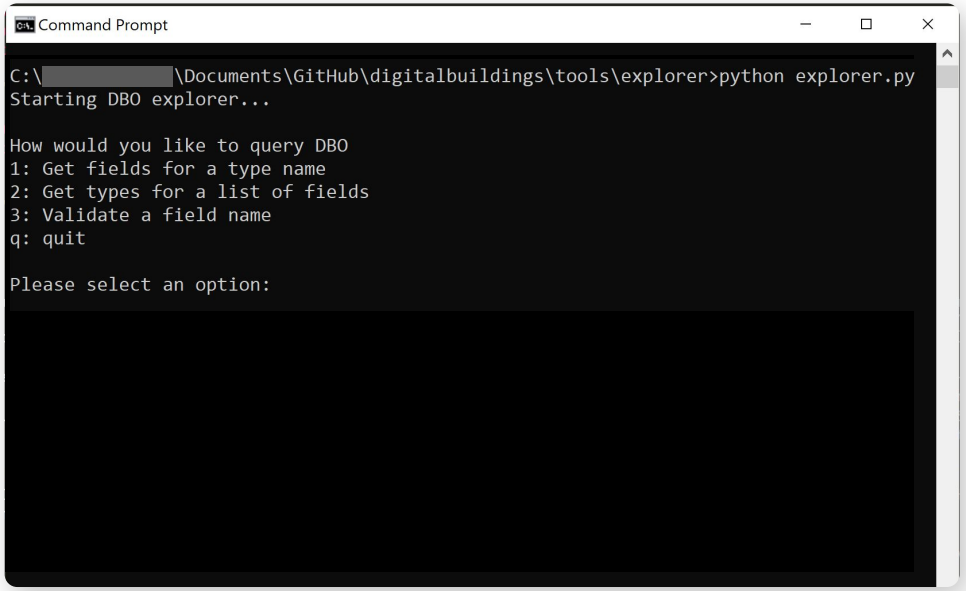
Next

# Ontology Explorer options

Let's explore the three options of actions you can perform with the Ontology Explorer.

*Select an option to reveal the use case, inputs, and outputs of each one.*

- Option 1: Get fields for a type name
  - Input
  - Output
- Option 2: Get types for a list of fields
  - Input 1 → Output 1
  - Input 2 → Output 2
- Option 3: Validate a field name
  - Input
  - Output




Back

Next

# Ontology Explorer options

Let's explore the three options of actions you can perform with the Ontology Explorer.

*Select an option to reveal the use case, inputs, and outputs of each one.*

- Option 1: Get fields for a type name
  - Input
  - Output
- Option 2: Get types for a list of fields
  - Input 1 → Output 1
  - Input 2 → Output 2
- Option 3: Validate a field name
  - Input
  - Output

## Option 1: Get fields for a type name

Let's say you want to know about the field associations for this canonical type.

```
AHU_CSP_DFSS_DXZC_ECONZ:
  id: "5459117700544462848"
  description: "Single zone AHU."
  is canonical: true
  implements:
  - AHU
  - CSP
  - DFSS
  - DXZC
  - ECONZ
```

At first glance, you get a sense for what it does simply by understanding its general type (**`AHU`**) and abstract types (**`DFSS`**, **`DSP`**, **`DXCZ`**, **`ECONZ`**). However, you don't know what fields each type requires without fully exploring each abstract type.

Back

Next

# Ontology Explorer options

Let's explore the three options of actions you can perform with the Ontology Explorer.

*Select an option to reveal the use case, inputs, and outputs of each one.*

- Option 1: Get fields for a type name
  - Input
  - Output
- Option 2: Get types for a list of fields
  - Input 1 — Output 1
  - Input 2 — Output 2
- Option 3: Validate a field name
  - Input
  - Output

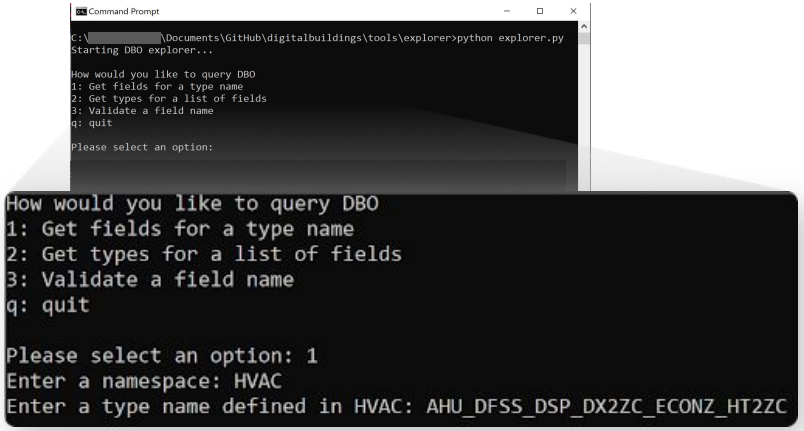
## Input

After starting Ontology Explorer in your terminal or command prompt:

1. Enter "`1`" on the first prompt.
2. Enter the namespace that contains the entity type in question on the second prompt.

   In this case, you'll enter "`HVAC`".
3. Enter the name of the entity type in question on the third prompt.

   In this case, you'll enter "`AHU_CSP_DFSS_DX2ZC_ECONZ_HT2ZC`".




Back

Next

Google

# Ontology Explorer options

Let's explore the three options of actions you can perform with the Ontology Explorer.

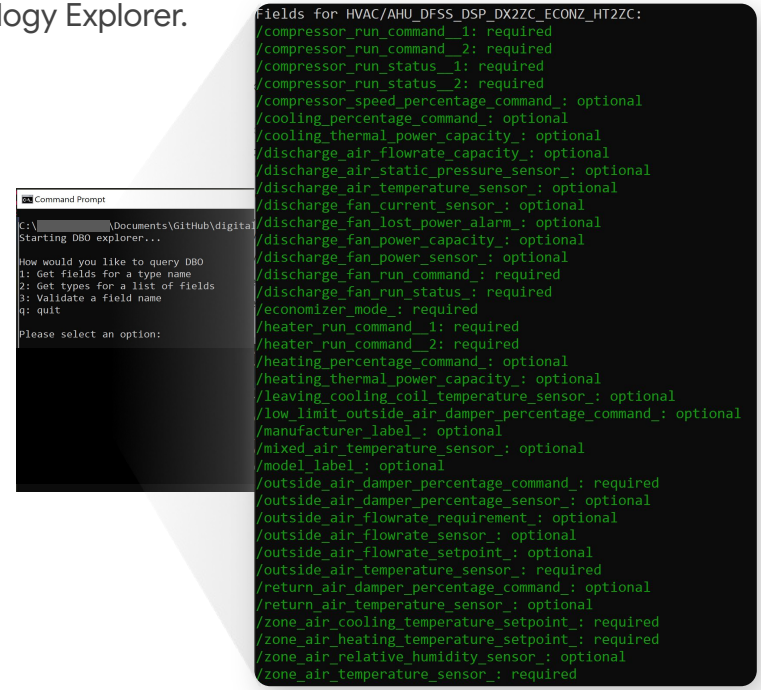
*Select an option to reveal the use case, inputs, and outputs of each one.*

- Option 1: Get fields for a type name
  - Input
  - Output
- Option 2: Get types for a list of fields
  - Input 1 — Output 1
  - Input 2 — Output 2
- Option 3: Validate a field name
  - Input
  - Output

## Output

The Ontology Explorer will output a full set of fields that are associated with the inputted entity type including associated and inherited fields from other types. It also indicates which fields are required or optional.

Now you know exactly which fields are required for the model–and in turn, which device data is required, too!




Back

Next

# Ontology Explorer options

Let's explore the three options of actions you can perform with the Ontology Explorer.

*Select an option to reveal the use case, inputs, and outputs of each one.*

- Option 1: Get fields for a type name
  - Input
  - Output
- **Option 2: Get types for a list of fields**
  - Input 1 → Output 1
  - Input 2 → Output 2
- Option 3: Validate a field name
  - Input
  - Output

## Option 2: Get types for a list of fields

Let's say you want to compare a field set to existing entity types.

Here's an exhaust fan with fields that have already been identified. Is there an entity type in the DBO that matches this field set? If there is, how well does it match up to this field set?

| device | field |
|---|---|
| EF-1 | run_command |
| EF-2 | run_status |
| EF-3 | current_sensor |
| EF-4 | speed_frequency_sensor |

Back

Next

Google

# Ontology Explorer options

Let's explore the three options of actions you can perform with the Ontology Explorer.

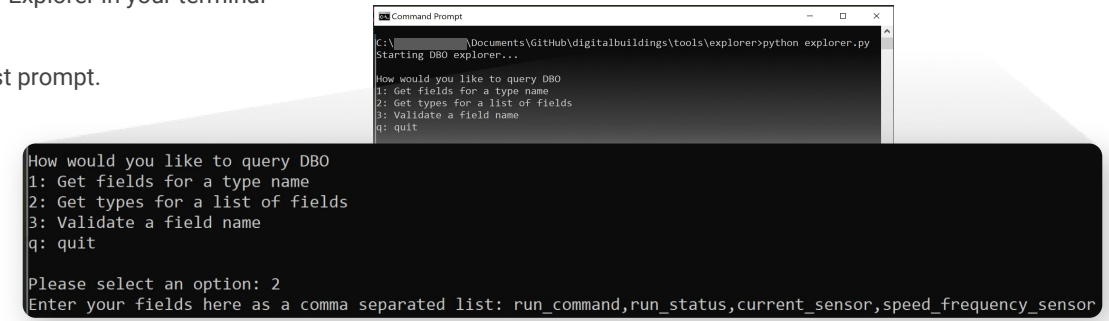
*Select an option to reveal the use case, inputs, and outputs of each one.*

- Option 1: Get fields for a type name
  - Input
  - Output
- Option 2: Get types for a list of fields
  - Input 1 — Output 1
  - Input 2 — Output 2
- Option 3: Validate a field name
  - Input
  - Output

## Input 1

After starting Ontology Explorer in your terminal or command prompt:

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.

   In this case, you'll enter "`run_command,run_status,current_sensor,speed_frequency_sensor`".

Back

Next

# Ontology Explorer options

Let's explore the three options of actions you can perform with the Ontology Explorer.

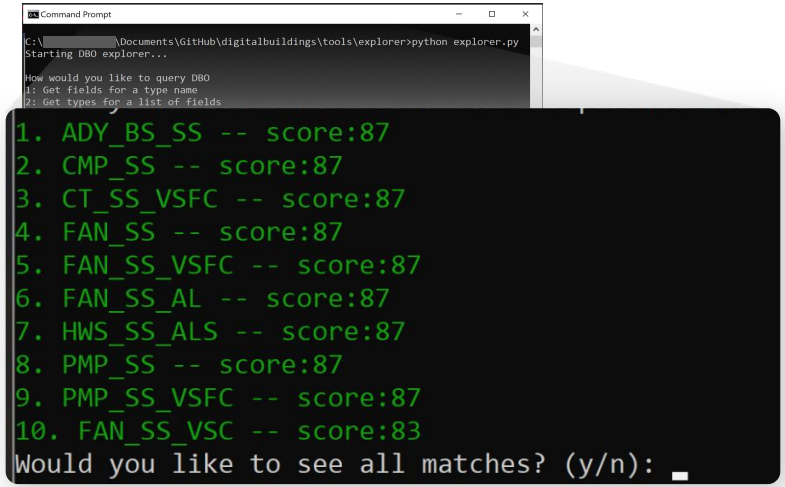
*Select an option to reveal the use case, inputs, and outputs of each one.*

- Option 1: Get fields for a type name
  - Input
  - Output
- Option 2: Get types for a list of fields
  - Input 1
  - Output 1
  - Input 2
  - Output 2
- Option 3: Validate a field name
  - Input
  - Output

## Output 1

The Ontology Explorer will compare the inputted field set with what exists in the DBO. It'll output a list of the top ten closest matching entity types and a score for each one to show how closely the field set matches an existing entity type. The score ranges from 0-100 with 100 being a perfect match.

You also have the option to view all matches. Simply enter "`y`" to view all or "`n`" to skip.




Back

Next

# Ontology Explorer options

Let's explore the three options of actions you can perform with the Ontology Explorer.

*Select an option to reveal the use case, inputs, and outputs of each one.*

- Option 1: Get fields for a type name
  - Input
  - Output
- Option 2: Get types for a list of fields
  - Input 1 — Output 1
  - Input 2 — Output 2
- Option 3: Validate a field name
  - Input
  - Output

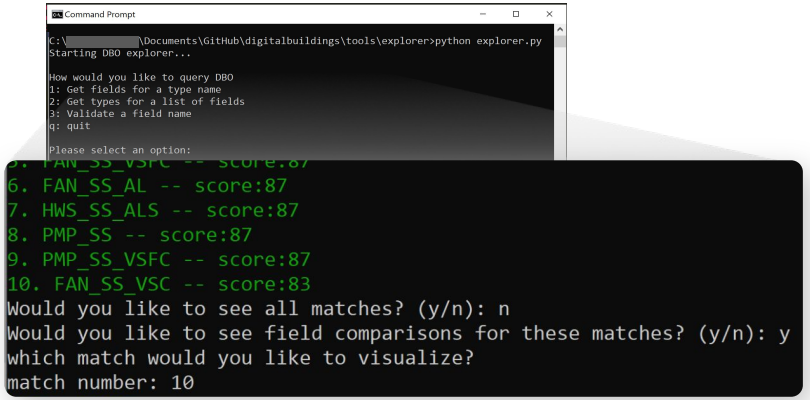
## Input 2

Additionally, you also have the option to inspect one of the entity types from the 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.

   In this case, let's enter "`10`" to look at match number `10. FAN_SS_VSC`.




Back

Next

# Ontology Explorer options

Let's explore the three options of actions you can perform with the Ontology Explorer.

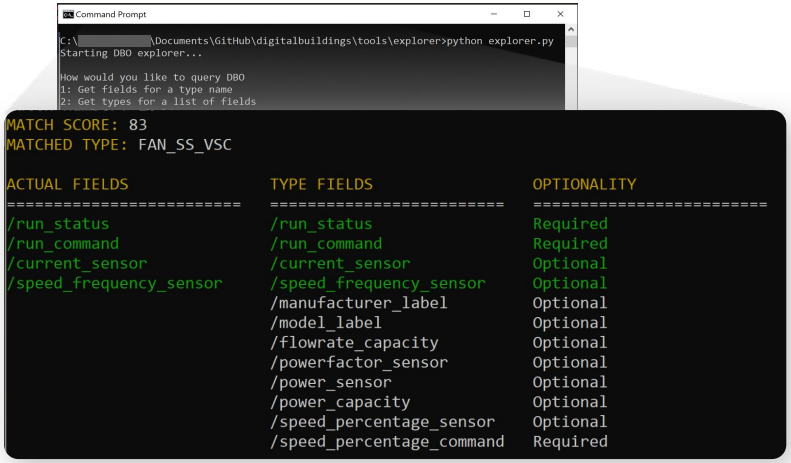
*Select an option to reveal the use case, inputs, and outputs of each one.*

- Option 1: Get fields for a type name
  - Input
  - Output
- Option 2: Get types for a list of fields
  - Input 1
  - Output 1
  - Input 2
  - Output 2
- Option 3: Validate a field name
  - Input
  - Output

## Output 2

The Ontology Explorer will output all fields for the inputted entity type and indicate whether the field is required or optional. It will also highlight in green which fields you included in your original field set.

Now you can see that the fields you provided cover nearly all the required fields. However, `speed_percentage_command`, a required field for this type, is missing from the field set you provided. This method of field comparison will become invaluable to you as you attempt to apply the DBO. You can see exactly what you're missing to convert an existing field set into a real, canonical type.




Back

Next

# Ontology Explorer options

Let's explore the three options of actions you can perform with the Ontology Explorer.

*Select an option to reveal the use case, inputs, and outputs of each one.*

- Option 1: Get fields for a type name
  - Input
  - Output
- Option 2: Get types for a list of fields
  - Input 1 → Output 1
  - Input 2 → Output 2
- **Option 3: Validate a field name**
  - Input
  - Output

## Option 3: Validate a field name

Let's say you want to validate whether a field you wish to use or create already exists in the DBO.

Here are two fields to check:

`zone_air_temperature_sensor`

`zone_temperature_sensor`

Back

Next

Google

# Ontology Explorer options

Let's explore the three options of actions you can perform with the Ontology Explorer.

*Select an option to reveal the use case, inputs, and outputs of each one.*

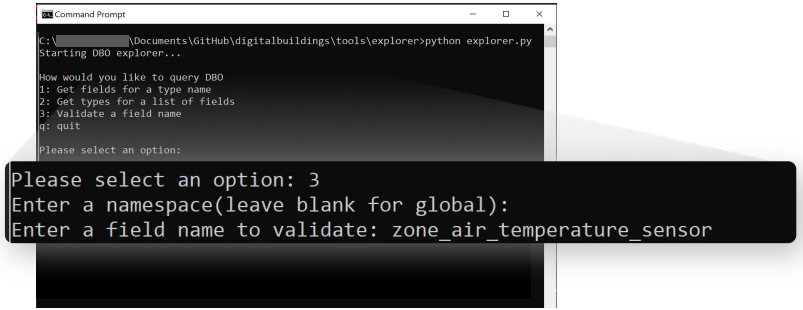
- Option 1: Get fields for a type name
  - Input
  - Output
- Option 2: Get types for a list of fields
  - Input 1 — Output 1
  - Input 2 — Output 2
- Option 3: Validate a field name
  - **Input**
  - Output

## Input

After starting Ontology Explorer in your terminal or command prompt:

1. Enter "`3`" on the first prompt.
2. Since most fields are defined in the global namespace, leave the second prompt blank and press enter.
3. Enter the field to check on the third prompt.

   In this case, try checking `zone_air_temperature_sensor` first and then `zone_temperature_sensor` after.




Back

Next

# Ontology Explorer options

Let's explore the three options of actions you can perform with the Ontology Explorer.

*Select an option to reveal the use case, inputs, and outputs of each one.*

- Option 1: Get fields for a type name
  - Input
  - Output
- Option 2: Get types for a list of fields
  - Input 1 — Output 1
  - Input 2 — Output 2
- Option 3: Validate a field name
  - Input
  - Output

## Output

The Ontology Explorer will output whether the inputted field exists in the DBO's global namespace (or the specified namespace if one was entered in the second input prompt).

`True` indicates it's a defined field that exists in the DBO, and `False` indicates it isn't defined.

### Terminal

Here's the output for the defined field `zone_air_temperature_sensor`

```
Please select an option: 3
Enter a namespace(leave blank for global):
Enter a field name to validate: zone_air_temperature_sensor
zone_air_temperature_sensor is defined in global namespace: True
```

True

Here's the output for the undefined field `zone_temperature_sensor`

```
Please select an option: 3
Enter a namespace(leave blank for global):
Enter a field name to validate: zone_temperature_sensor
zone_temperature_sensor is defined in global namespace: False
```

False

Back

Next

Lesson 3

# Practice 2

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

- This practice activity will revisit the `AHU` that you identified earlier in the lesson.
- The next slides will give you an opportunity to use the Ontology Explorer to check an entity type's field associations.
- If you haven't done so already, install the Ontology Explorer on your machine.
- After this practice activity, you'll move on to required device data.

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

Back

Next

Google

# Practice 2

Earlier, you identified the canonical type `AHU_DFSS_DSP_DX2ZC_ECONZ_HT2ZC` as the possible type of the `AHU` depicted in the pictured BMS points list.

**What required and optional fields could be associated with this device?**

*Use the Ontology Explorer to get all field associations for the type `AHU_DFSS_DSP_DX2ZC_ECONZ_HT2ZC`.*

## BMS screenshot

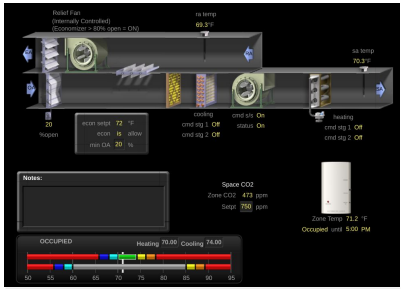



**Image source:** Google's WebCTRL instances. WebCTRL is a building automation system owned by Automated Logic.

## Steps

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

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

Then:

1. Enter "`1`" on the first prompt.
2. Enter the namespace that contains the entity type in question on the second prompt.
3. Enter the name of the entity type in question on the third prompt.

## BMS points list

| Name | Type | Object ID | Device ID | Object Name |
|---|---|---|---|---|
| Return Air Temperature ai | BAI | AI:42 | DEV:2528722 | return_air_temperature_ai_4 |
| Discharge Temp | BAI | AI:2 | DEV:2528722 | da_temp_1 |
| Heat Stage 1 | BBI | BI:1 | DEV:2528722 | htg_stg1_bi_1 |
| Heat Stage 2 | BBI | BI:2 | DEV:2528722 | htg_stg2_bi_2 |
| Setpoint / Effective Cooling Setpoint | BAV | AV:7 | DEV:2528722 | Effective Cooling_1 |
| Setpoint / Effective Heating Setpoint | BAV | AV:8 | DEV:2528722 | Effective Heating_1 |
| Zone Temp | ASVI | AI:4 | DEV:2528722 | zone_temp_1 |
| DX Cooling Stage 1 | BBI | BI:9 | DEV:2528722 | dx_clg_stg1_bi_1 |
| DX Cooling Stage 2 | BBI | BI:10 | DEV:2528722 | dx_clg_stg2_bi_1 |
| Econ Min Pos | BAV | AV:13 | DEV:2528722 | ec_min_1 |
| Economizer Setpoint | BAV | AV:14 | DEV:2528722 | ec_setpt_1 |
| Economizer Damper Command | BAO | AO:1 | DEV:2528722 | econ_1 |
| Supply Fan Start/Stop | BBO | BO:1 | DEV:2528722 | sf_cmd_bo_1 |
| Supply Fan Status | BBI | BI:3 | DEV:2528722 | sf_stat_bi_1 |
| Zone CO2 | BAV | AV:32 | DEV:2528722 | co2_1 |
| CO2 Setpoint | BAV | AV:33 | DEV:2528722 | co2_stpt_1 |

Back

*When you're ready, click **Next** to see how we identified the abstract types.*

Next

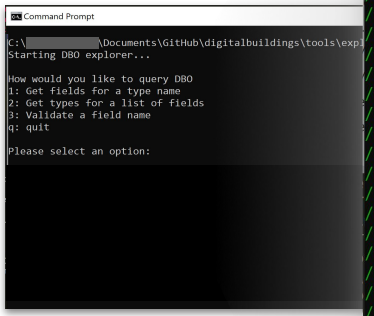
# Practice 2

## Check your answer! 👩‍💻

Here's a look at our output of associated fields after checking using Ontology Explorer. **Did you come up with the same field associations?**

These fields in `AHU_DFSS_DSP_DX2ZC_ECONZ_HT2ZC` are already defined on the abstract types `DFSS`, `DSP`, `DX2ZC`, `ECONZ and HT2ZC`. Don't forget about the abstract type `CO2C`, though! We identified it earlier as a necessary type, but we couldn't find a canonical type that includes it. That's fine for now, but as we get further in the workflow, we will likely need to extend the ontology to support it (extensions will be covered later in Lesson 5).

Note that some of the fields which the screenshot appears to have, such as `return_air_temperature_sensor`, naturally have homes somewhere on an already defined abstract type, and therefore don't require that an additional abstract type be defined to cover that field. Again, the $CO_2$ sensor and setpoint aren't defined on this type, so even though it is close to what we want there will need to be some extensions made to the ontology to support those additional fields.




```
Fields for HVAC/AHU_DFSS_DSP_DX2ZC_ECONZ_HT2ZC:
/compressor_run_command__1: required
/compressor_run_command__2: required
/compressor_run_status__1: required
/compressor_run_status__2: required
/compressor_speed_percentage_command_: optional
/cooling_percentage_command_: optional
/cooling_thermal_power_capacity_: optional
/discharge_air_flowrate_capacity_: optional
/discharge_air_static_pressure_sensor_: optional
/discharge_air_temperature_sensor_: optional
/discharge_fan_current_sensor_: optional
/discharge_fan_lost_power_alarm_: optional
/discharge_fan_power_capacity_: optional
/discharge_fan_power_sensor_: optional
/discharge_fan_run_command_: required
/discharge_fan_run_status_: required
/economizer_mode_: required
/heater_run_command__1: required
/heater_run_command__2: required
/heating_percentage_command_: optional
/heating_thermal_power_capacity_: optional
/leaving_cooling_coil_temperature_sensor_: optional
/low_limit_outside_air_damper_percentage_command_: optional
/manufacturer_label_: optional
/mixed_air_temperature_sensor_: optional
/model_label_: optional
/outside_air_damper_percentage_command_: required
/outside_air_damper_percentage_sensor_: optional
/outside_air_flowrate_requirement_: optional
/outside_air_flowrate_sensor_: optional
/outside_air_flowrate_setpoint_: optional
/outside_air_temperature_sensor_: required
/return_air_damper_percentage_command_: optional
/return_air_temperature_sensor_: optional
/zone_air_cooling_temperature_setpoint_: required
/zone_air_heating_temperature_setpoint_: required
/zone_air_relative_humidity_sensor_: optional
/zone_air_temperature_sensor_: required
```

Back

*Click **Next** to see an alternative way to find field associations.*

Next

# Practice 2

Alternatively, you can check each abstract type one by one in the HVAC namespace.

Here are the abstract types we identified earlier that describe the functions of the **AHU**.

- **DFSS**
- **DSP**
- **ECONZ**
- **DX2ZC**
- **HT2ZC**
- **CO2C**

A close look at each one in the HVAC/…/ABSTRACT.yaml will reveal the required and optional field associations, which is a good indicator of the data that's usually associated with this type of **AHU**.

Since they weren't retrieved with the Ontology Explorer, here are the required field associations of CO2C retrieved from HVAC/…/ABSTRACT.yaml:

DBO sample

```
CO2C:
  id: "14886233640072642560"
  description: "Carbon dioxide control."
  is abstract: true
  implements:
  - OPERATIONAL
  uses:
  - zone air co2 concentration sensor
  - zone_air_co2_concentration_setpoint
```

Note that it doesn't have optional fields.

Back

*Click **Next** to move on to required device data.*

Next

# Which device data should be modeled?

A **reporting device** can generate and send a lot of data, but not everything needs to be modeled.

Technically, all data generated by a reporting device could be modeled. However, we tend to avoid doing this and only model for anticipated applications and analytics, because modeling everything is just too excessive.

The general rule of thumb is to model things that align with the device's general behavior and functions without too much detail about its low-level configuration. Only model data that you anticipate would be useful to your project team. More data can always be added later as long as it's sent by its controller.

Back

Next

# Which device data should be modeled? (continued)

A **reporting device** can generate and send a lot of data, but not everything needs to be modeled.

*Click on each data type to reveal information and examples.*

- Required data types
  - Measured telemetry
  - Setpoint telemetry
  - Control states
- Optional data types
  - Interpreted states
  - Configuration information

| | Number | Integer | Multi-state | String | Date/Time |
|---|---|---|---|---|---|
| `accumulator` | ✅ | ❌ | ❌ | ❌ | ❌ |
| `alarm` | ❌ | ❌ | ✅ | ❌ | ❌ |
| `capacity` | ✅ | ❌ | ❌ | ❌ | ❌ |
| `counter` | ❌ | ✅ | ❌ | ❌ | ❌ |
| `command` | ✅ | ❌ | ✅ | ❌ | ❌ |
| `count` | ❌ | ✅ | ❌ | ❌ | ❌ |
| `label` | ❌ | ❌ | ❌ | ✅ | ❌ |
| `mode` | ❌ | ❌ | ✅ | ❌ | ❌ |
| `requirement` | ✅ | ❌ | ❌ | ❌ | ❌ |
| `sensor` | ✅ | ❌ | ❌ | ❌ | ❌ |
| `setpoint` | ✅ | ❌ | ❌ | ❌ | ❌ |
| `status` | ❌ | ❌ | ✅ | ❌ | ❌ |
| `specification` | ✅ | ❌ | ❌ | ❌ | ❌ |
| `timestamp` | ❌ | ❌ | ❌ | ❌ | ✅ |

Back

Next

# Which device data should be modeled? (continued)

A **reporting device** can generate and send a lot of data, but not everything needs to be modeled.

*Click on each data type to reveal information and examples.*

- Required data types
  - Measured telemetry
  - Setpoint telemetry
  - Control states
- Optional data types
  - Interpreted states
  - Configuration information

## Required data types

The following types of data are normally modeled:

- Measured telemetry
- Setpoint telemetry
- Control states

| | Number | Integer | Multi-state | String | Date/Time |
|---|---|---|---|---|---|
| `accumulator` | ✅ | ❌ | ❌ | ❌ | ❌ |
| `alarm` | ❌ | ❌ | ✅ | ❌ | ❌ |
| `capacity` | ✅ | ❌ | ❌ | ❌ | ❌ |
| `counter` | ❌ | ✅ | ❌ | ❌ | ❌ |
| `command` | ✅ | ❌ | ✅ | ❌ | ❌ |
| `count` | ❌ | ✅ | ❌ | ❌ | ❌ |
| `label` | ❌ | ❌ | ❌ | ✅ | ❌ |
| `mode` | ❌ | ❌ | ✅ | ❌ | ❌ |
| `requirement` | ✅ | ❌ | ❌ | ❌ | ❌ |
| `sensor` | ✅ | ❌ | ❌ | ❌ | ❌ |
| `setpoint` | ✅ | ❌ | ❌ | ❌ | ❌ |
| `status` | ❌ | ❌ | ✅ | ❌ | ❌ |
| `specification` | ✅ | ❌ | ❌ | ❌ | ❌ |
| `timestamp` | ❌ | ❌ | ❌ | ❌ | ✅ |

Back

Next

# Which device data should be modeled? (continued)

A **reporting device** can generate and send a lot of data, but not everything needs to be modeled.

*Click on each data type to reveal information and examples.*

- Required data types
  - Measured telemetry
  - Setpoint telemetry
  - Control states
- Optional data types
  - Interpreted states
  - Configuration information

## Measured telemetry

Measured telemetry includes sensors that are associated with the device. They are directly measured or calculated by the device and return updated values as the state of the device changes through time.

**All measured telemetry is normally modeled.**

### Examples

`supply_air_temperature_sensor`
`chilled_water_valve_percentage_sensor`

| | Number | Integer | Multi-state | String | Date/Time |
|---|---|---|---|---|---|
| `accumulator` | ✅ | ❌ | ❌ | ❌ | ❌ |
| `alarm` | ❌ | ❌ | ✅ | ❌ | ❌ |
| `capacity` | ✅ | ❌ | ❌ | ❌ | ❌ |
| `counter` | ❌ | ✅ | ❌ | ❌ | ❌ |
| `command` | ✅ | ❌ | ✅ | ❌ | ❌ |
| `count` | ❌ | ✅ | ❌ | ❌ | ❌ |
| `label` | ❌ | ❌ | ❌ | ✅ | ❌ |
| `mode` | ❌ | ❌ | ✅ | ❌ | ❌ |
| `requirement` | ✅ | ❌ | ❌ | ❌ | ❌ |
| `sensor` | ✅ | ❌ | ❌ | ❌ | ❌ |
| `setpoint` | ✅ | ❌ | ❌ | ❌ | ❌ |
| `status` | ❌ | ❌ | ✅ | ❌ | ❌ |
| `specification` | ✅ | ❌ | ❌ | ❌ | ❌ |
| `timestamp` | ❌ | ❌ | ❌ | ❌ | ✅ |

Back

**Note:** We do not consider calculated telemetry differently from raw telemetry.

Next

# Which device data should be modeled? (continued)

A **reporting device** can generate and send a lot of data, but not everything needs to be modeled.

*Click on each data type to reveal information and examples.*

- Required data types
  - Measured telemetry
  - **Setpoint telemetry**
  - Control states
- Optional data types
  - Interpreted states
  - Configuration information

## Setpoint telemetry

Setpoint telemetry includes measured telemetry values that are being directly controlled. The measured values are compared to the setpoint values to provide some type of state comparison (e.g., the zone is cold), so the device can respond through the adjustment of control values.

**All directly controlled setpoints are normally modeled.**

### Examples

`zone_air_cooling_temperature_setpoint`
`supply_air_static_pressure_setpoint`

| | Number | Integer | Multi-state | String | Date/Time |
|---|---|---|---|---|---|
| `accumulator` | ✅ | ❌ | ❌ | ❌ | ❌ |
| `alarm` | ❌ | ❌ | ✅ | ❌ | ❌ |
| `capacity` | ✅ | ❌ | ❌ | ❌ | ❌ |
| `counter` | ❌ | ✅ | ❌ | ❌ | ❌ |
| `command` | ✅ | ❌ | ✅ | ❌ | ❌ |
| `count` | ❌ | ✅ | ❌ | ❌ | ❌ |
| `label` | ❌ | ❌ | ❌ | ✅ | ❌ |
| `mode` | ❌ | ❌ | ✅ | ❌ | ❌ |
| `requirement` | ✅ | ❌ | ❌ | ❌ | ❌ |
| `sensor` | ✅ | ❌ | ❌ | ❌ | ❌ |
| `setpoint` | ✅ | ❌ | ❌ | ❌ | ❌ |
| `status` | ❌ | ❌ | ✅ | ❌ | ❌ |
| `specification` | ✅ | ❌ | ❌ | ❌ | ❌ |
| `timestamp` | ❌ | ❌ | ❌ | ❌ | ✅ |

Back

Next

Google

# Which device data should be modeled? (continued)

A **reporting device** can generate and send a lot of data, but not everything needs to be modeled.

*Click on each data type to reveal information and examples.*

- Required data types
  - Measured telemetry
  - Setpoint telemetry
  - Control states
- Optional data types
  - Interpreted states
  - Configuration information

## Control states

Control states are values that represent how the device responds to measured deviations from setpoints or other changes in environmental/temporal conditions.

**Most control states are normally modeled.**

### Examples

These control states indicate the device adjusts directly or indirectly based on measured conditions:

`supply_fan_run_command`

`chilled_water_valve_percentage_command`

| | Number | Integer | Multi-state | String | Date/Time |
|---|---|---|---|---|---|
| `accumulator` | ✅ | ❌ | ❌ | ❌ | ❌ |
| `alarm` | ❌ | ❌ | ✅ | ❌ | ❌ |
| `capacity` | ✅ | ❌ | ❌ | ❌ | ❌ |
| `counter` | ❌ | ✅ | ❌ | ❌ | ❌ |
| `command` | ✅ | ❌ | ✅ | ❌ | ❌ |
| `count` | ❌ | ✅ | ❌ | ❌ | ❌ |
| `label` | ❌ | ❌ | ❌ | ✅ | ❌ |
| `mode` | ❌ | ❌ | ✅ | ❌ | ❌ |
| `requirement` | ✅ | ❌ | ❌ | ❌ | ❌ |
| `sensor` | ✅ | ❌ | ❌ | ❌ | ❌ |
| `setpoint` | ✅ | ❌ | ❌ | ❌ | ❌ |
| `status` | ❌ | ❌ | ✅ | ❌ | ❌ |
| `specification` | ✅ | ❌ | ❌ | ❌ | ❌ |
| `timestamp` | ❌ | ❌ | ❌ | ❌ | ✅ |

Back

Next

# Which device data should be modeled? (continued)

A **reporting device** can generate and send a lot of data, but not everything needs to be modeled.

*Click on each data type to reveal information and examples.*

- Required data types
  - Measured telemetry
  - Setpoint telemetry
  - Control states
- **Optional data types**
  - Interpreted states
  - Configuration information

## Optional data types

The following types of data are rarely modeled unless there's an explicit need to do so:

- Interpreted states
- Configuration information

If you decide to model the optional data types, be prepared to extend the ontology; since these are not normally modeled, they will likely have minimal precedent to help guide you. Generally this data is ignored, and it is usually easiest to do so.

| | Number | Integer | Multi-state | String | Date/Time |
|---|---|---|---|---|---|
| `accumulator` | ✅ | ❌ | ❌ | ❌ | ❌ |
| `alarm` | ❌ | ❌ | ✅ | ❌ | ❌ |
| `capacity` | ✅ | ❌ | ❌ | ❌ | ❌ |
| `counter` | ❌ | ✅ | ❌ | ❌ | ❌ |
| `command` | ✅ | ❌ | ✅ | ❌ | ❌ |
| `count` | ❌ | ✅ | ❌ | ❌ | ❌ |
| `label` | ❌ | ❌ | ❌ | ✅ | ❌ |
| `mode` | ❌ | ❌ | ✅ | ❌ | ❌ |
| `requirement` | ✅ | ❌ | ❌ | ❌ | ❌ |
| `sensor` | ✅ | ❌ | ❌ | ❌ | ❌ |
| `setpoint` | ✅ | ❌ | ❌ | ❌ | ❌ |
| `status` | ❌ | ❌ | ✅ | ❌ | ❌ |
| `specification` | ✅ | ❌ | ❌ | ❌ | ❌ |
| `timestamp` | ❌ | ❌ | ❌ | ❌ | ✅ |

Back

Next

# Which device data should be modeled? (continued)

A **reporting device** can generate and send a lot of data, but not everything needs to be modeled.

*Click on each data type to reveal information and examples.*

- Required data types
  - Measured telemetry
  - Setpoint telemetry
  - Control states
- Optional data types
  - Interpreted states
  - Configuration information

## Interpreted states

Interpreted states are ones which the device interprets from underlying data. The best example of this is any alarm.

**Interpreted states are not normally modeled.**

### Examples

`fan_mismatch_alarm` compares the status of the fan to the command and returns an alarm if they are not equal.

| | Number | Integer | Multi-state | String | Date/Time |
|---|---|---|---|---|---|
| `accumulator` | ✅ | ❌ | ❌ | ❌ | ❌ |
| `alarm` | ❌ | ❌ | ✅ | ❌ | ❌ |
| `capacity` | ✅ | ❌ | ❌ | ❌ | ❌ |
| `counter` | ❌ | ✅ | ❌ | ❌ | ❌ |
| `command` | ✅ | ❌ | ✅ | ❌ | ❌ |
| `count` | ❌ | ✅ | ❌ | ❌ | ❌ |
| `label` | ❌ | ❌ | ❌ | ✅ | ❌ |
| `mode` | ❌ | ❌ | ✅ | ❌ | ❌ |
| `requirement` | ✅ | ❌ | ❌ | ❌ | ❌ |
| `sensor` | ✅ | ❌ | ❌ | ❌ | ❌ |
| `setpoint` | ✅ | ❌ | ❌ | ❌ | ❌ |
| `status` | ❌ | ❌ | ✅ | ❌ | ❌ |
| `specification` | ✅ | ❌ | ❌ | ❌ | ❌ |
| `timestamp` | ❌ | ❌ | ❌ | ❌ | ✅ |

Back

Next

Google

# Which device data should be modeled? (continued)

A **reporting device** can generate and send a lot of data, but not everything needs to be modeled.

*Click on each data type to reveal information and examples.*

- Required data types
  - Measured telemetry
  - Setpoint telemetry
  - Control states
- Optional data types
  - Interpreted states
  - Configuration information

## Configuration information

Configuration information are internal device points used to configure the way in which the device attempts to maintain control values or how its components are controlled internally.

**Configuration information is rarely modeled.**

### Examples

Stage up/down timers
PID gains

| | Number | Integer | Multi-state | String | Date/Time |
|---|---|---|---|---|---|
| `accumulator` | ✅ | ❌ | ❌ | ❌ | ❌ |
| `alarm` | ❌ | ❌ | ✅ | ❌ | ❌ |
| `capacity` | ✅ | ❌ | ❌ | ❌ | ❌ |
| `counter` | ❌ | ✅ | ❌ | ❌ | ❌ |
| `command` | ✅ | ❌ | ✅ | ❌ | ❌ |
| `count` | ❌ | ✅ | ❌ | ❌ | ❌ |
| `label` | ❌ | ❌ | ❌ | ✅ | ❌ |
| `mode` | ❌ | ❌ | ✅ | ❌ | ❌ |
| `requirement` | ✅ | ❌ | ❌ | ❌ | ❌ |
| `sensor` | ✅ | ❌ | ❌ | ❌ | ❌ |
| `setpoint` | ✅ | ❌ | ❌ | ❌ | ❌ |
| `status` | ❌ | ❌ | ✅ | ❌ | ❌ |
| `specification` | ✅ | ❌ | ❌ | ❌ | ❌ |
| `timestamp` | ❌ | ❌ | ❌ | ❌ | ✅ |

Back

Next

# Which device data should be modeled? (continued)

Knowing the **abstract types** and their required field associations will help you determine required data points from the type of device you're modeling.

## No payload, no problem

At this point in the process, you very likely don't have a payload of data to begin translating into a building configuration file.

However, since you're able to identify devices that will need to be modeled, their general types, and their abstract types, you can anticipate the device data that will be generated and accurately predict what will be useful to your project and model.

## How to determine the device data to model

After determining a device's general type and abstract types:

1. Re-inspect the project documents and the notable functions of the device and its subcomponents.
2. Assess each function and the type of data it would generate.
    - If the data is a required data type, plan to include it in your model.
    - If the data is an optional data type, don't plan to include it in your model unless your project's contributors have specified a need for it.
3. Revisit each abstract type to compare your plans with precedent.

Back

Next

Google

Lesson 3

# Knowledge check 2

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

- The next slide will have questions about the concepts and actions that have been introduced since the last practice activity.
- Review the question and select the correct response.
- After this knowledge check, you’ll wrap up Lesson 3.

**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 2

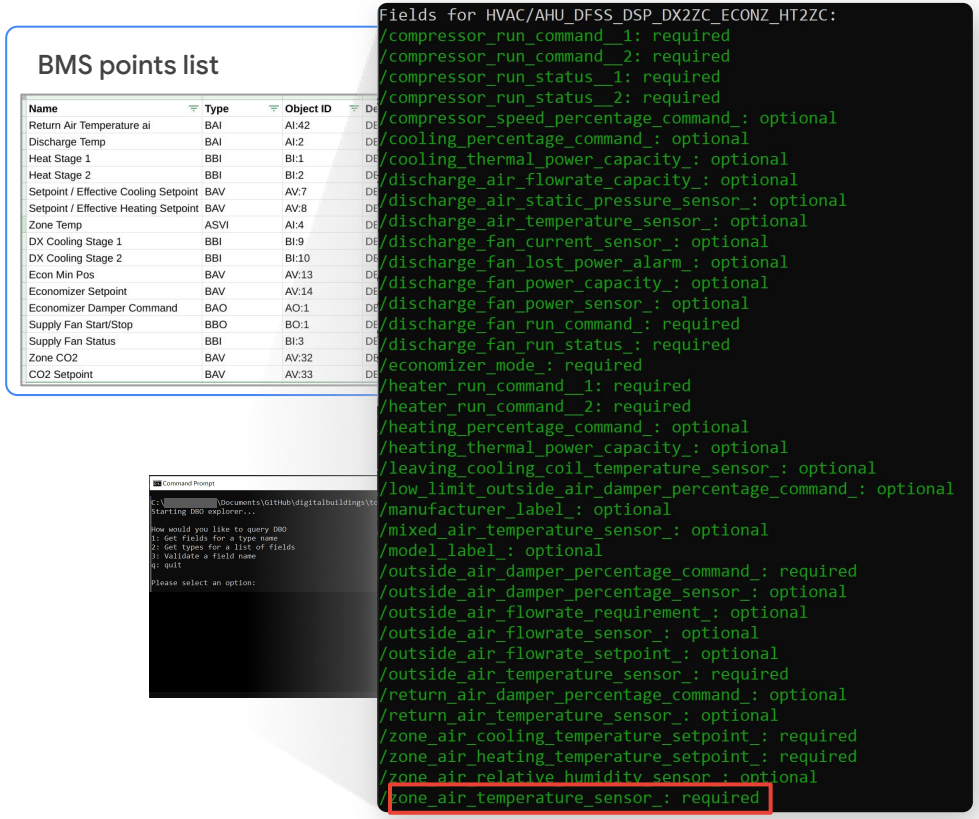
Earlier, you identified required and optional fields for the abstract types that describe the functions of the `AHU` from the BMS points list.

The abstract type `DSP` will be needed to describe the function for the dual setpoint control. One of this type's associated fields is `zone_air_temperature_sensor`.

## What type of device data would be associated with the field `zone_air_temperature_sensor`?

*Select the best answer from the options listed below.*

- Measured telemetry
- Setpoint telemetry
- Control state
- Interpreted state
- Configuration information




Back

Next

# Knowledge check 2

Earlier, you identified required and optional fields for the abstract types that describe the functions of the `AHU` from the BMS points list.

The abstract type `DSP` will be needed to describe the function for the dual setpoint control. One of this type's associated fields is `zone_air_temperature_sensor`.

**What type of device data would be associated with the field `zone_air_temperature_sensor`?**

*Select the best answer from the options listed below.*

- Measured telemetry
- Setpoint telemetry
- Control state
- Interpreted state
- Configuration information

## That's right! 🎉

The field `zone_air_temperature_sensor` would definitely be associated with **measured telemetry** device data, a required data type.

Remember, measured telemetry are sensors associated with the device. They are directly measured or calculated by the device and return updated values as the state of the device changes through time.

Back

Next

# Knowledge check 2

Earlier, you identified required and optional fields for the abstract types that describe the functions of the `AHU` from the BMS points list.

The abstract type `DSP` will be needed to describe the function for the dual setpoint control. One of this type's associated fields is `zone_air_temperature_sensor`.

**What type of device data would be associated with the field `zone_air_temperature_sensor`?**

*Select the best answer from the options listed below.*

- Measured telemetry
- Setpoint telemetry
- Control state
- Interpreted state
- Configuration information

## Hmm, that's not right! 🤔

The field `zone_air_temperature_sensor` wouldn't be associated with **setpoint telemetry** device data, a required data type.

As a sensor, you can expect some sort of measured or calculated device data. Setpoint telemetry includes measured telemetry values that are being directly controlled and compared to provide some type of state comparison.

Try again

Back

Next

# Knowledge check 2

Earlier, you identified required and optional fields for the abstract types that describe the functions of the `AHU` from the BMS points list.

The abstract type `DSP` will be needed to describe the function for the dual setpoint control. One of this type's associated fields is `zone_air_temperature_sensor`.

**What type of device data would be associated with the field `zone_air_temperature_sensor`?**

*Select the best answer from the options listed below.*

- Measured telemetry
- Setpoint telemetry
- Control state
- Interpreted state
- Configuration information

## Hmm, that's not right! 🤔

The field `zone_air_temperature_sensor` wouldn't be associated with **control state** device data, a required data type.

As a sensor, you can expect some sort of measured or calculated device data. Control states are values that represent how the device responds to measured deviations from setpoints or other changes in environmental/temporal conditions.

Try again

Back

Next

Google

# Knowledge check 2

Earlier, you identified required and optional fields for the abstract types that describe the functions of the `AHU` from the BMS points list.

The abstract type `DSP` will be needed to describe the function for the dual setpoint control. One of this type's associated fields is `zone_air_temperature_sensor`.

**What type of device data would be associated with the field `zone_air_temperature_sensor`?**

*Select the best answer from the options listed below.*

- Measured telemetry
- Setpoint telemetry
- Control state
- Interpreted state
- Configuration information

## Hmm, that's not right! 🤔

The field `zone_air_temperature_sensor` wouldn't be associated with **interpreted state** device data, an optional data type.

As a sensor, you can expect some sort of measured or calculated device data. Interpreted states are ones which the device interprets from underlying data, such as an alarm.

Try again

Back

Next

Google

# Knowledge check 2

Earlier, you identified required and optional fields for the abstract types that describe the functions of the `AHU` from the BMS points list.

The abstract type `DSP` will be needed to describe the function for the dual setpoint control. One of this type's associated fields is `zone_air_temperature_sensor`.

**What type of device data would be associated with the field `zone_air_temperature_sensor`?**

*Select the best answer from the options listed below.*

- Measured telemetry
- Setpoint telemetry
- Control state
- Interpreted state
- Configuration information

## Hmm, that's not right! 🤔

The field `zone_air_temperature_sensor` wouldn't be associated with **configuration information** device data, an optional data type.

As a sensor, you can expect some sort of measured or calculated device data. Configuration information are internal device points used to configure the way in which the device attempts to maintain control values or how its components are controlled internally.

Try again

Back

Next

# Repeat for each logical entity

To determine which data points are required, you'll repeat these steps for each logical entity that needs to be included in the building config.

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

- Determine the general type
- Identify abstract and canonical types
- Find field associations
- Determine the device data to model

In the end, whether a data point is required will depend on context, such as whether the device uses that particular point for control or monitoring. It's up to you as a data modeler to make these judgment calls.

You're not alone, though! There is a precedent set in the DBO that will guide your decision-making. The Digital Buildings Project is also here to help, and you can submit questions to us in areas where novel conditions arise.

Back

Next

Google

# Repeat for each logical entity

To determine which data points are required, you'll repeat these steps for each logical entity that needs to be included in the building config.

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

- Determine the general type
- Identify abstract and canonical types
- Find field associations
- Determine the device data to model

## Determine the general type

1. Understand the definitions for defined general equipment types in each namespace.
   This is very important: definitions will help to determine whether certain equipment is of the type that it appears to be.
2. Inspect one of the logical entities that you identified from the project documents.
   - Need help with that? Revisit Lesson 2: Determine which devices need to be modeled.
3. Identify notable characteristics, subcomponents, and assumed behaviors about the device.
4. Match your findings with a general type in the global.yaml or one of the child namespaces:
   - ELECTRICAL/…/GENERALTYPES.yaml
   - HVAC/…/GENERALTYPES.yaml
   - LIGHTING/…/GENERALTYPES.yaml
   - METERS/…/GENERALTYPES.yaml
   - SAFETY/…/GENERALTYPES.yaml

You can also refer to the model_hvac.md for descriptions of general types in the HVAC namespace.

Back

Next

Google

# Repeat for each logical entity

To determine which data points are required, you'll repeat these steps for each logical entity that needs to be included in the building config.

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

- Determine the general type
- Identify abstract and canonical types
- Find field associations
- Determine the device data to model

## Identify possible abstract and canonical types

**After determining the device's general type:**

1. Re-inspect the project documents to identify the notable functions of the device and its subcomponents.

**Then do one or both of the following:**

2. Match the functions you identified with one of the DBO's abstract type definitions in the abstract types file or from the docs in model_hvac.md.
3. Explore the child namespace of the device's general type to find a similar or matching canonical type. What kinds of abstract types do they implement? It can be helpful to compare and see how your device varies from it.

Back

Next

# Repeat for each logical entity

To determine which data points are required, you'll repeat these steps for each logical entity that needs to be included in the building config.

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

- Determine the general type
- Identify abstract and canonical types
- Find field associations
- Determine the device data to model

## Run the Ontology Explorer to find field associations

**After identifying possible canonical types:**

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 "`1`" on the first prompt.
2. Enter the namespace that contains the entity type in question on the second prompt.
3. Enter the name of the entity type in question on the third prompt.

Back

Next

Google

# Repeat for each logical entity

To determine which data points are required, you'll repeat these steps for each logical entity that needs to be included in the building config.

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

- Determine the general type
- Identify abstract and canonical types
- Find field associations
- Determine the device data to model

## Determine the device data to model

**After identifying abstract types and required field associations:**

1. Re-inspect the project documents and the notable functions of the device and its subcomponents.
2. Assess each function and the type of data it would generate.
   - If the data is a required data type, plan to include it in your model.
   - If the data is an optional data type, don't plan to include it in your model unless your project's contributors have specified a need for it.
3. Revisit each abstract type to compare your plans with precedent.

Back

Next

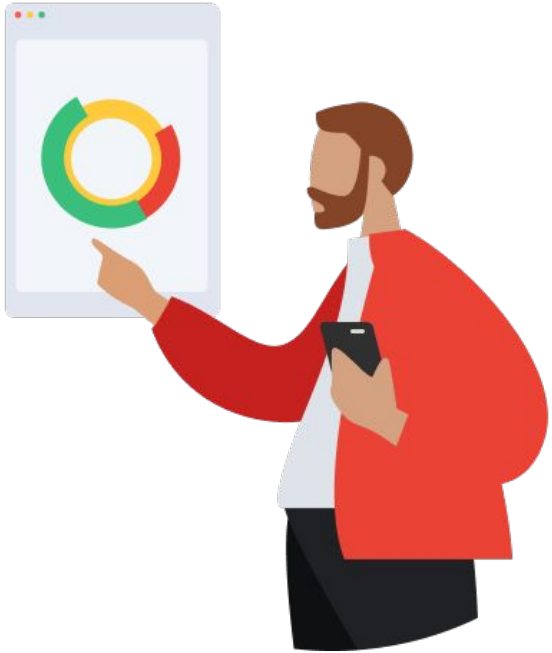
# Lesson 3 summary

## Let's review what you learned about:

- General types and device categories
- Canonical types and device descriptions
- Abstract types and device functions
- Ontology Explorer
- Field associations and device data

## Now you should be able to:

- Identify the general type of a logical entity.
- Identify a likely canonical type of a logical entity.
- Identify the likely abstract types of a logical entity.
- Use the Ontology Explorer to check the Digital Buildings Ontology (DBO) for required fields.
- Identify the required data points of a logical entity.



Back

Next

Google

# You completed Lesson 3!

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

## Ready for Lesson 4?

Let's go!

Back

Google

## 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.
- Ontology Explorer
  Used to ask basic questions of what's curated within the DBO.