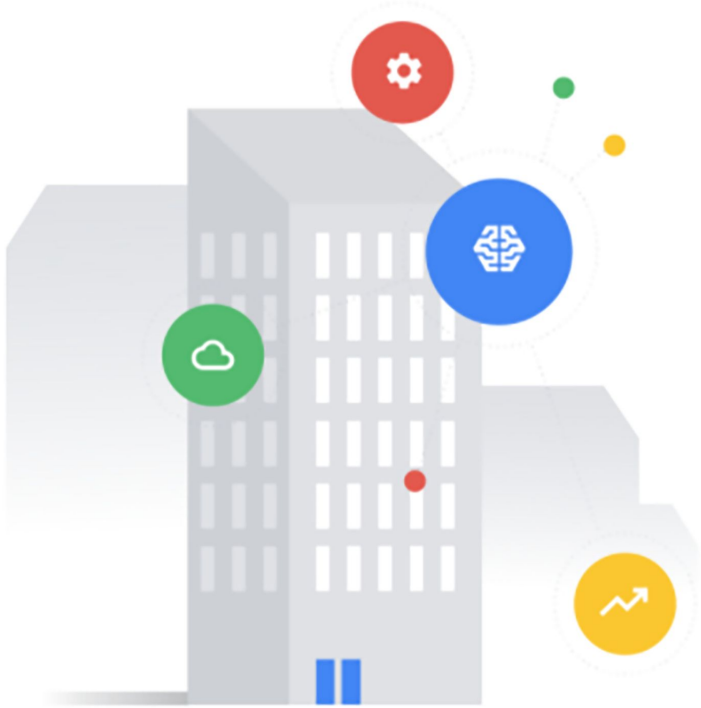
Google

Module 1 | **Lesson 6**

Digital Buildings Ontology (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!

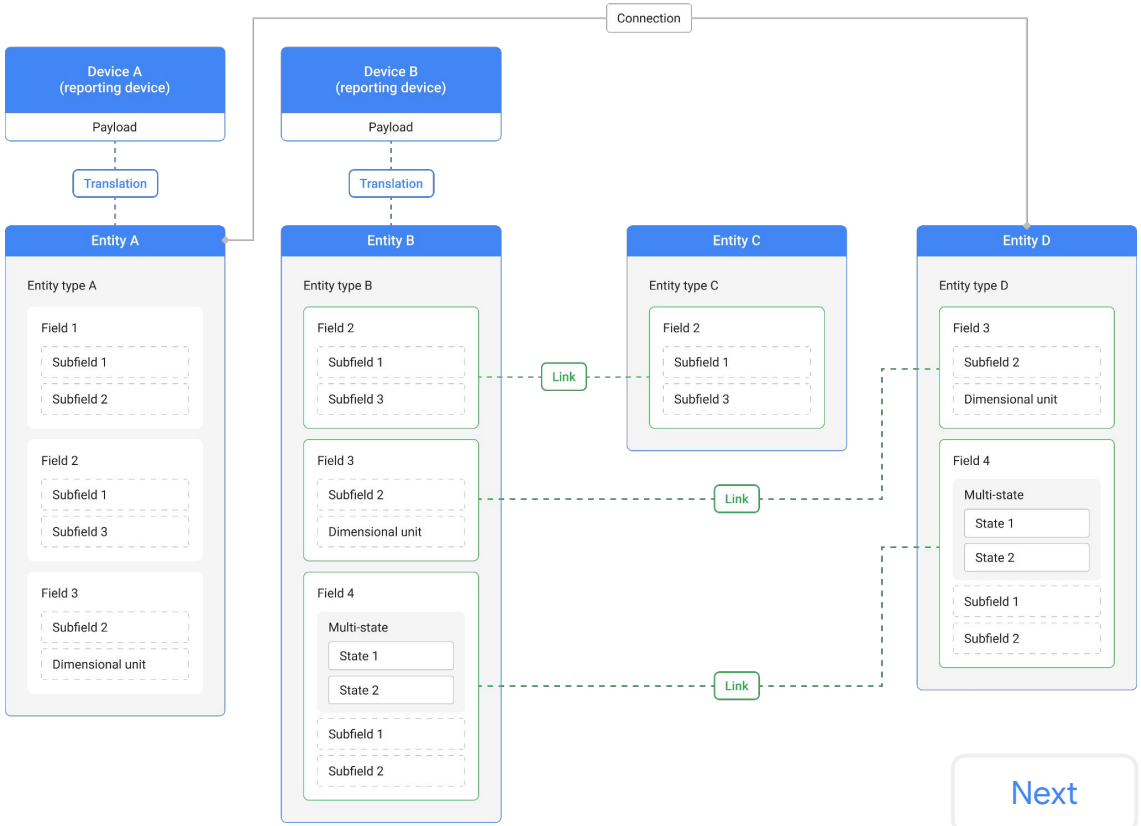
Conceptual model revisited

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

In this lesson, you'll explore one modeling concept from the abstract model. Remember, abstract modeling concepts are used to describe the properties of an entity. Abstract concepts include:

- Subfields
- Fields
- States and multi-states
- Entity types

Do you see these concepts in the diagram?

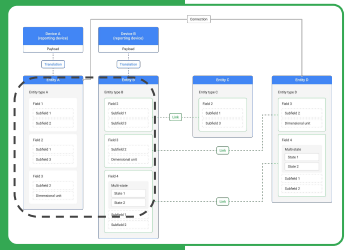


Back

Next

Lesson 6

Entity types



What you'll learn about:

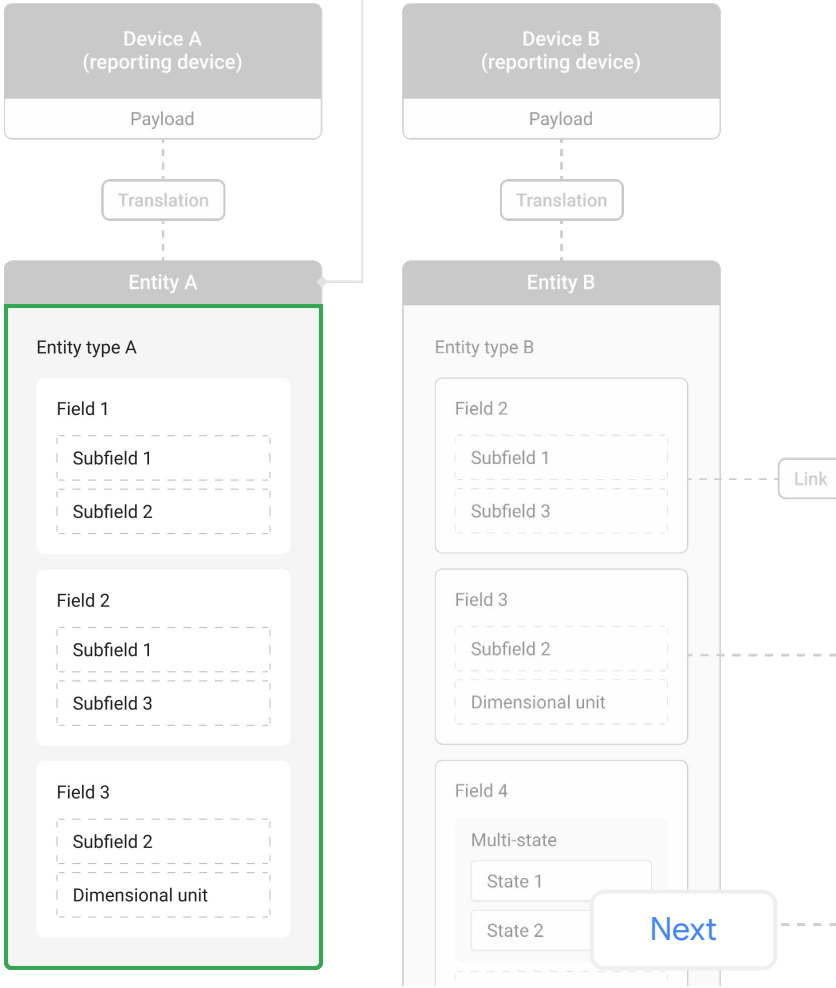
- Entity types
- Abstract, general, and canonical types
- Entity type attributes

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

- Describe the concept of an entity type.
- Identify an entity type in source code.
- Explain the difference between general, abstract, and canonical types.
- Recognize the attributes of an entity type.

Back

Google



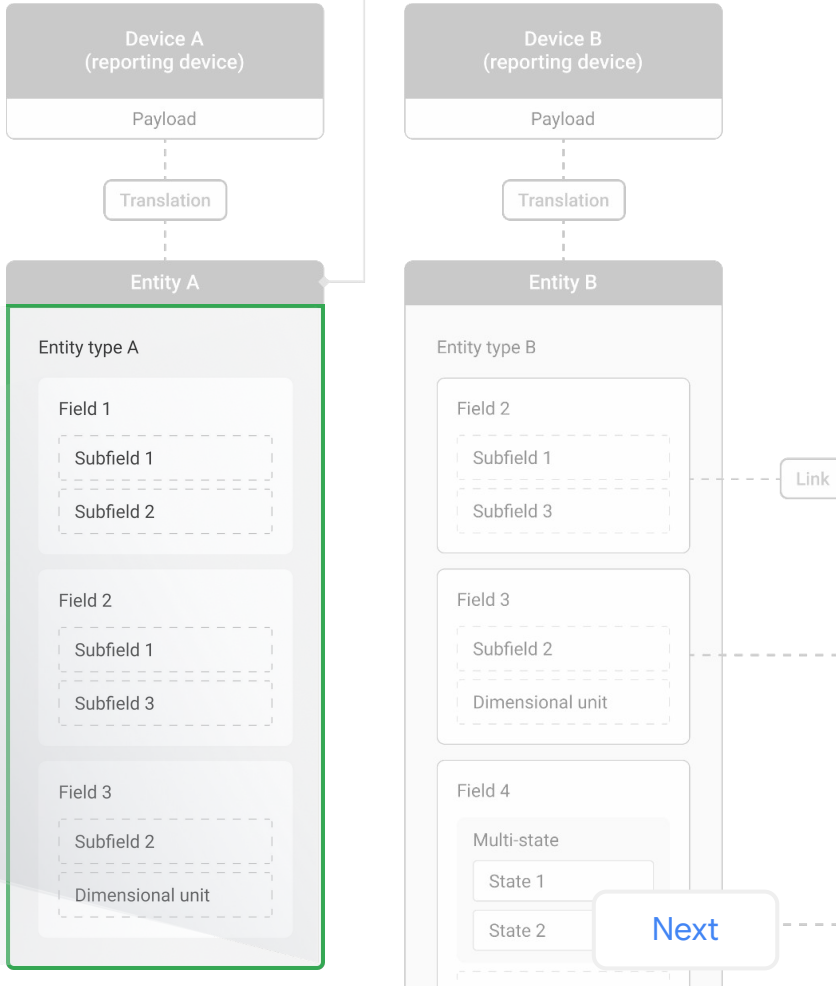
What's an entity type?

An entity type represents the abstract concepts that describe the individual, concrete instance of an entity.

Entity types are a composition of fields (and inherently the fields' subfields, states, and units), parent entity types, and several standard attributes that gives meaning to its entity. They describe the properties and functionality of an entity.

Example

```
FAN_SS_DWI:
  id: "11319382735195209728"
  description: "Dishwasher-interlocked fan."
  is_canonical: true
  implements:
  - FAN
  - SS
  uses:
  - dishwasher_run_status
```



Back

Entity types

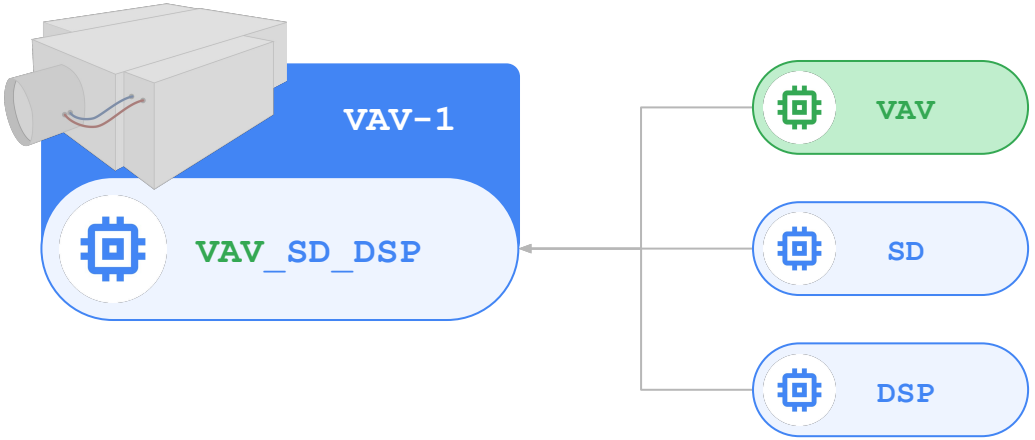
Different entity types are used to describe functionality and classify entities.

Click on each term to reveal more info about it.

Abstract types

General types

Canonical types



Back

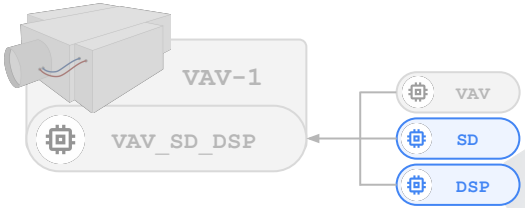
Next

Entity types

Different entity types are used to describe functionality and classify entities.

Click on each term to reveal more info about it.

- Abstract types
- General types
- Canonical types



Abstract types

An abstract type is a curated concept that represents a block of functionality (i.e., what something can do).

In the example, `DSP` represents the abstract concept of zone temperature control using an upper and lower bound setpoint. Although specific, this functionality is a broadly used to describe the capabilities of air handling units (AHU), fan coil units (FCU), etc.

Abstract types like `DSP` are independent concepts, but you'll never have a device in the real world that is a dual setpoint. Instead, a canonical type would implement that function.

Example

```
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
```

Back

Note: Technically, abstract types are canonical because they're curated. However, by convention, only a flagged canonical type is considered canonical.

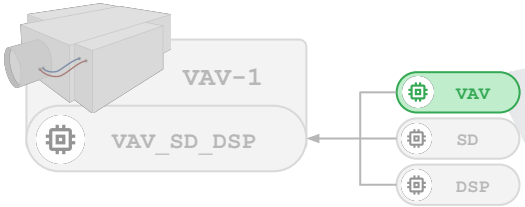
Next

Entity types

Different entity types are used to describe functionality and classify entities.

Click on each term to reveal more info about it.

- Abstract types
- General types
- Canonical types



General types

A general type is a curated concept that acts as a broad category to classify equipment (i.e., what something is).

In the example, **VAV** represents the abstract concept of all variable air volume (VAV) units. This general type is very broad and is an applicable category for the many flavors of VAVs.

General types like **VAV** are independent concepts, but they're too broadly defined to be used by an entity directly. Instead, a canonical type would implement that category.

Example

```
VAV:
  id: "6599610325710929920"
  description: "Tag for terminal units with
variable volume control. A VAV is an air
distribution terminal,which is not
responsible for its own primary airflow
(i.e. it is served by some upstream unit,
such as an AHU)."
  is_abstract: true
  implements:
  - EQUIPMENT
  opt_uses:
  - zone_use_label  # Needed until the zones
are joined to assets with connections.
```

Back

Note: Technically, general types are canonical because they're curated. However, by convention, only a flagged canonical type is considered canonical.

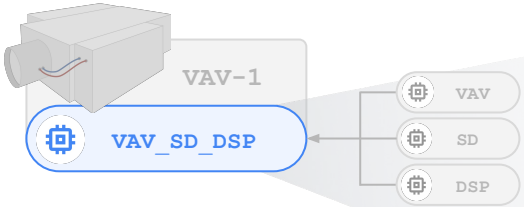
Next

Entity types

Different entity types are used to describe functionality and classify entities.

Click on each term to reveal more info about it.

- Abstract types
- General types
- Canonical types



Canonical types

A canonical type is a curated concept that can be mapped to an individual entity. It combines a general type and abstract types to represent a specific type of equipment and its functionality.

In this example, the canonical type **VAV_SD_DSP** combines the general type **VAV** and the abstract types **SD** and **DSP**. Now it will neatly map to many VAVs that are equipped in a building.

Example

```
VAV_SD_DSP:
  id: "1382788727640576000"
  description: "Simple cooling only VAV."
  is_canonical: true
  implements:
  - VAV
  - SD
  - DSP
```

Back

Note: There are also non-canonical types, which have a very specific use case when mapping equipment to concepts. You'll learn more about non-canonical types and their use case in Lesson 7.

Next

Entity types (continued)

Different entity types are used to describe functionality and classify entities.

Canonical type

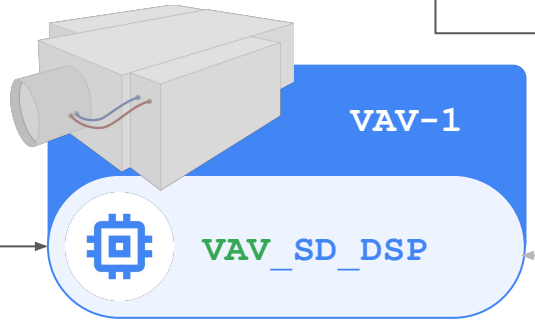
```
VAV_SD_DSP:
  id: "13827887727640576000"
  description: "Simple cooling only VAV."
  is_canonical: true
  implements:
  - VAV
  - SD
  - DSP
```

General type

```
VAV:
  id: "6599610325710929920"
  description: "Tag for terminal units with variable volume
control. A VAV is an air distribution terminal,which is not
responsible for its own primary airflow (i.e. it is served by
some upstream unit, such as an AHU)."
  is_abstract: true
  implements:
  - EQUIPMENT
  opt_uses:
  - zone_use_label # Needed until the zones are joined to
assets with connections.
```

Abstract type

```
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
```



Back

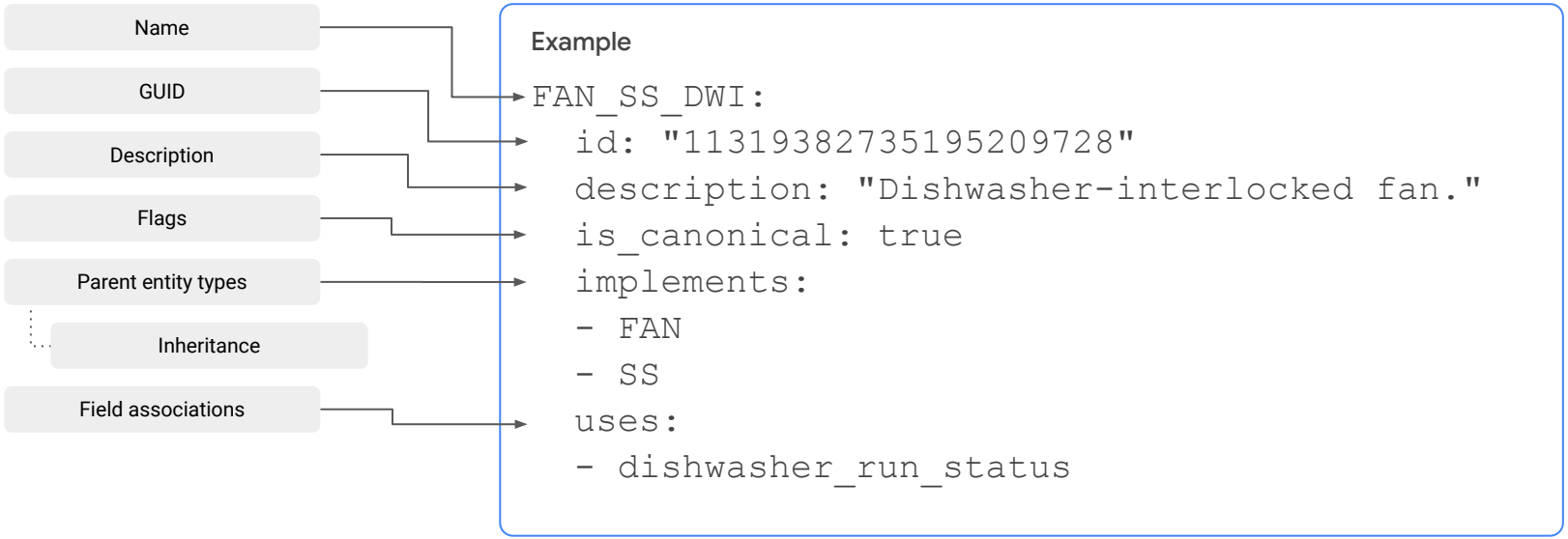
Next

Google

Entity type attributes

Entity types have the following attributes:

Click on each attribute to reveal more info about it.



Back

Next

Google

Entity type attributes

Entity types have the following attributes:

Click on each attribute to reveal more info about it.

- Name
- GUID
- Description
- Flags
- Parent entity types
  - Inheritance
- Field associations

Name

The name is a visible identifier for the entity type and its functionality. It has no structural meaning in the DBO.

Naming convention

While a naming convention isn't enforced, we recommend following these rules to help others easily identify the functionality of an entity type:

- Start with a general type.
  `VAV_SD_CSP`
- Follow with a _-separated, alphabetized list of abstract types.
  `VAV_SD_CSP`

In the example, **`FAN`** is a general type and **`SS`** is an abstract type. **`DWI`** is not an abstract type, but it modifies **`FAN_SS`** with additional contextual information.

Example

```
FAN_SS_DWI:
  id: "11319382735195209728"
  Description: "Dishwasher-interlocked
fan."
  is_canonical: true
  implements:
  - FAN
  - SS
  uses:
  - dishwasher_run_status
```

Back

Note: Sometimes, an abstract type may not be available but additional context is needed to make sense of an entity's functionality. In these cases, an ontology extension is preferred or you can add a modifier to the name like in the above example.

Next

Entity type attributes

Entity types have the following attributes:

Click on each attribute to reveal more info about it.

- Name
- GUID
- Description
- Flags
- Parent entity types
  - Inheritance
- Field associations

GUID

The `id` block provides the entity type's globally unique identification (GUID) number. It's version-independent and isn't affected if the entity type name or version changes.

The GUID is set by the system. Whenever entity types are created or modified, the GUID shouldn't be added or changed manually.

Example

```
FAN SS DWI:
  id: "11319382735195209728"
  Description: "Dishwasher-interlocked
fan."
  is_canonical: true
  implements:
  - FAN
  - SS
  uses:
  - dishwasher_run_status
```

Back

Note: The GUID applies to the entity type and not the specific entity it's associated with. Think of a GUID like a model number, not a serial number.

Next

Entity type attributes

Entity types have the following attributes:

Click on each attribute to reveal more info about it.

- Name
- GUID
- Description
- Flags
- Parent entity types
  - Inheritance
- Field associations

Description

The **`description`** block provides a short, human-readable description of the entity type.

Example

```
FAN_SS_DWI:
  id: "11319382735195209728"
  Description: "Dishwasher-interlocked
fan."
  is_canonical: true
  implements:
  - FAN
  - SS
  uses:
  - dishwasher_run_status
```

Back

Next

Google

Entity type attributes

Entity types have the following attributes:

Click on each attribute to reveal more info about it.

- Name
- GUID
- Description
- Flags
- Parent entity types
  - Inheritance
- Field associations

Abstract and canonical flags

The `is_canonical` and `is_abstract` flags indicate whether the entity type is abstract or canonical.

Canonical entity types

A canonical type is a curated entity type that defines an entity's properties rather than its functionality. By convention, canonical types are the set of abstract modeling concepts used to represent the concrete instance of an entity.

Abstract entity types

An abstract type is a curated definition of an entity's functionality rather than its properties. By convention, abstract types are inherently canonical since it's curated.

Abstract types are to be used for inheritance and shouldn't be directly associated with an entity.

Example

```
FAN_SS_DWI:
  id: "11319382735195209728"
  Description: "Dishwasher-interlocked
fan."
  is_canonical: true
  implements:
  - FAN
  - SS
  uses:
  - dishwasher_run_status
```

Back

Note: If one or both of the flags appear to be missing, then you can safely assume the missing flag is set to `false`.

Next

Entity type attributes

Entity types have the following attributes:

Click on each attribute to reveal more info about it.

- Name
- GUID
- Description
- Flags
- Parent entity types
  - Inheritance
- Field associations

Parent entity types

The **`implements`** block indicates the entity type is a child of the listed parent entity types.

In the example, the entity type **`FAN_SS_DWI`** has two parent types: **`FAN`** and **`SS`**. This means **`FAN_SS_DWI`** will inherit field definitions from **`FAN`** and **`SS`**.

Example

```
FAN_SS_DWI:
  id: "11319382735195209728"
  Description: "Dishwasher-interlocked
fan."
  is_canonical: true
  implements:
  - FAN
  - SS
  uses:
  - dishwasher_run_status
```

Back

Next

Entity type attributes

Entity types have the following attributes:

Click on each attribute to reveal more info about it.

- Name
- GUID
- Description
- Flags
- Parent entity types
  - Inheritance
- Field associations

Inheritance

Entity types will inherit the field definitions of its parent entity type(s).

Any entity type can be a parent for any other type. However, we usually reserve parents to abstract entity types and form a canonical entity type from them.

Inheritance rules

The following will apply to an inherited field:

- An inherited field won't inherit the following attributes:
  - `description`
  - `id`
  - `is_canonical`/`is_abstract` flags
- An inherited field is only inherited once if it was inherited from multiple parents.
  Example: Type XY is a child of Type X and Type Y. Type X has Fields A and B. Type Y has Fields B and C. Type XY inherits Fields A, B, and C with only one instance of Field B.
- An inherited field is required if at least one parent entity type requires it.
  Example: Type XY is a child of Type X and Type Y. Type X has made Field A required. Type Y has made Field A optional. Type XY inherits Field A and makes it required.
- An inherited field preserves field enumeration and considers every enumerated field to be unique.
  Example: Type XY is a child of Type X and Type Y. Type X has fields A and A_1. Type Y has fields A_1 and A_2. Type XY inherits fields A, A_1, and A_2.

Back

Next

Entity type attributes

Entity types have the following attributes:

Click on each attribute to reveal more info about it.

- Name
- GUID
- Description
- Flags
- Parent entity types
  - Inheritance
- Field associations

Field associations

The **`uses`** block indicates any fields that are required while the **`opt_uses`** block indicates any fields that are optional.

Why associate fields?

Associating fields minimizes the number of entity type variations that are required to cover all entities that need to be typed in a building.

This is especially true in the HVAC domain where there may be a large number of minor variations between the set of abstract parent type and canonical types. However, those minor variations may not be worth modeling themselves as abstract types.

Example

```
FAN_SS_DWI:
  id: "11319382735195209728"
  Description: "Dishwasher-interlocked
fan."
  is_canonical: true
  implements:
  - FAN
  - SS
  uses:
  - dishwasher_run_status
```

Back

*Click **Next** for more info about field associations.*

Next

Google

Entity type attributes

Entity types have the following attributes:

Click on each attribute to reveal more info about it.

- Name
- GUID
- Description
- Flags
- Parent entity types
  - Inheritance
- Field associations

Field associations (continued)

Keep the following guidelines in mind when working with field associations:

- Entity types are distinguishable from each other even if they have the same fields.

> **Example**
>
> If Type X and Type Y both have Field A, they are still two different entity types.

- New entity types with the same fields should only be made when they're truly distinguishable from each other.

> **Example**
>
> For example, both `HVAC/FAN_SS` and `HVAC/PMP_SS` have identical fields, but they're truly two different types of things. One is a fan that moves gas, and the other is a pump that moves liquid.

- **Optional fields shouldn't be abused!** Although the DBO technically permits extending a loosely related canonical entity type with multiple optional fields for a concrete entity, this is considered poor practice. Instead, new abstract entity types should be added to cover those optional fields.

Back

Next

Lesson 6

Knowledge check

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

- The next slides will have questions about the concepts that were introduced in this lesson.
- Review each question and select the correct response.

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

Back

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

Next

Google

Knowledge check 1

Entity types are a composition of _____ that describes the properties and functionality of an entity.

Fill in the blank.

Select the best answer from the options listed below.

- fields
- parent entity types
- attributes
- all of the above

Back

Next

Google

Knowledge check 1

Entity types are a composition of _____ that describes the properties and functionality of an entity.

Fill in the blank.

Select the best answer from the options listed below.

- fields
- parent entity types
- attributes
- all of the above

Close... but not quite right! 🤔

An entity type can definitely be composed of **fields**. It defines field associations using the `uses` and `opt_uses` blocks. However, are fields the only valid option?

Try again

Back

Next

Knowledge check 1

Entity types are a composition of _____ that describes the properties and functionality of an entity.

Fill in the blank.

Select the best answer from the options listed below.

fields

parent entity types

attributes

all of the above

Close... but not quite right! 🤔

An entity type can definitely be composed of **parent entity types**. It inherits a parent's field associations using the `implements` block. However, are parent entity types the only valid option?

Try again

Back

Next

Knowledge check 1

Entity types are a composition of _____ that describes the properties and functionality of an entity.

Fill in the blank.

Select the best answer from the options listed below.

fields

parent entity types

attributes

all of the above

Close... but not quite right! 🤔

An entity type is definitely composed of standard **attributes**, including a name, id, description, flags, parents, and field associations. However, are attributes the only valid option?

Try again

Back

Next

Knowledge check 1

Entity types are a composition of _____ that describes the properties and functionality of an entity.

Fill in the blank.

Select the best answer from the options listed below.

fields

parent entity types

attributes

all of the above

That's right! 🎉

An entity type is an abstract modeling concept that describe the concrete instances of entities. It includes a set of standard attributes. It could define required and optional field associations. It could also inherit field associations by implementing parent entity types.

Example

```
FAN_SS_DWI:
  id: "11319382735195209728"
  description: "Dishwasher-interlocked fan."
  is_canonical: true
  implements:
  - FAN
  - SS
  uses:
  - dishwasher_run_status
```

Back

Next

Google

Knowledge check 2

A sample entity type is on the right.

```
PMP_SS_VSC:
 id: "6133109532278128640"
 description: "Typical variable speed pump."
 is_canonical: ???
 is_abstract: ???
 implements:
 - PMP
 - SS
 - VSC
```

Should this entity type be abstract or canonical?

Select the best answer from the options listed below.

Abstract | Canonical

Back | Next

Knowledge check 2

A sample entity type is on the right.

Should this entity type be abstract or canonical?

Select the best answer from the options listed below.

Abstract | Canonical

Close... but not quite right! 🤔

This isn't an **abstract type**. An abstract type is a curated concept that represents a specific block of functionality.

This is most likely a **canonical type**. You can tell because it is a composition of abstract functionality, combining the general type PMP and the abstract types SS and VSC. Since it's likely a canonical type, it will neatly map to an entity representing this type of pump.

```
PMP_SS_VSC:
 id: "6133109532278128640"
 description: "Typical variable speed pump."
 is_canonical: ???
 is_abstract: ???
 implements:
 - PMP
 - SS
 - VSC
```

Back

Next

Knowledge check 2

A sample entity type is on the right.

Should this entity type be abstract or canonical?

Select the best answer from the options listed below.

Abstract

Canonical

That's right! 🎉

This is most likely a **canonical type**. You can tell because it is a composition of abstract functionality, combining the general type PMP and the abstract types SS and VSC. Since it's likely a canonical type, it will neatly map to an entity representing this type of pump.

```
PMP_SS_VSC:
 id: "6133109532278128640"
 description: "Typical variable speed pump."
 is_canonical: ???
 is_abstract: ???
 implements:
 - PMP
 - SS
 - VSC
```

Back

Next

Google

Knowledge check 3

A sample entity type is shown below.

Which block indicates its parent entity types?

Select the line of code that best answers this question.

```
VAV_ZTM_PDSCV_SFM:
```

```
id: "2684578173177298944"
```

```
description: "VAV with supply air and zone air monitoring"
```

```
is_canonical: true
```

```
implements:
  - VAV
  - ZTM
  - PDSCV
  - SFM
```

```
opt_uses:
- failed_alarm
```

Back

Next

Google

Knowledge check 3

A sample entity type is shown below.

Which block indicates its parent entity types?

Select the line of code that best answers this question.

```
VAV_ZTM_PDSCV_SFM:
```

```
id: "2684578173177298944"
```

```
description: "VAV with supply air and zone air monitoring"
```

```
is_canonical: true
```

```
implements:
  - VAV
  - ZTM
  - PDSCV
  - SFM
```

```
opt_uses:
- failed_alarm
```

Hmm, that's not right! 🤔

This block is the **name** of this entity type.

Try again

Back

Next

Google

Knowledge check 3

A sample entity type is shown below.

Which block indicates its parent entity types?

Select the line of code that best answers this question.

```
VAV_ZTM_PDSCV_SFM:
id: "2684578173177298944"
description: "VAV with supply air and zone air monitoring"
is_canonical: true
implements:
  - VAV
  - ZTM
  - PDSCV
  - SFM
opt_uses:
- failed_alarm
```

Hmm, that's not right! 🤔

The `id` block is the globally unique identification (GUID) number of this entity type.

Try again

Back

Next

Google

Knowledge check 3

A sample entity type is shown below.

Which block indicates its parent entity types?

Select the line of code that best answers this question.

```
VAV_ZTM_PDSCV_SFM:
id: "2684578173177298944"
description: "VAV with supply air and zone air monitoring"
is_canonical: true
implements:
  - VAV
  - ZTM
  - PDSCV
  - SFM
opt_uses:
- failed_alarm
```

Hmm, that's not right! 🤔

The **`description`** block is a short, human-readable description of the entity type.

Try again

Back

Next

Google

Knowledge check 3

A sample entity type is shown below.

Which block indicates its parent entity types?

Select the line of code that best answers this question.

```
VAV_ZTM_PDSCV_SFM:
id: "2684578173177298944"
description: "VAV with supply air and zone air monitoring"
is_canonical: true
implements:
  - VAV
  - ZTM
  - PDSCV
  - SFM
opt_uses:
- failed_alarm
```

Hmm, that's not right! 🤔

This block is a flag. This entity type uses the `is_canonical` flag to indicate it is a canonical type. Abstract types would use the `is_abstract` flag.

Try again

Back

Next

Google

Knowledge check 3

A sample entity type is shown below.

Which block indicates its parent entity types?

Select the line of code that best answers this question.

```
VAV_ZTM_PDSCV_SFM:
id: "2684578173177298944"
description: "VAV with supply air and zone air monitoring"
is_canonical: true
implements:
  - VAV
  - ZTM
  - PDSCV
  - SFM
opt_uses:
- failed_alarm
```

Back

That's right! 🎉

The **`implements`** block is a list of parent entity types. The sample entity type has four parent types: **`VAV`**, **`ZTM`**, **`PDSCV`**, and **`SFM`**. That means it will inherit field associations from each parent type.

Remember that any entity type can be a parent for any other type. Abstract types usually serve as parent types for canonical types.

Other attributes of the sample entity type include:

- A name, which is **`VAV_ZTM_PDSCV_SFM`**. Name's are a visible identifier for the entity type and its functionality.
- A description, which is indicated by the **`description`** block. It provides a short, human-readable description of the entity type.
- A flag, which is indicated by **`is_canonical: true`**. This means it is a canonical type.
- A field association, which is indicated by the **`opt_uses`** block.

Next

Google

Knowledge check 3

A sample entity type is shown below.

Which block indicates its parent entity types?

Select the line of code that best answers this question.

```
VAV_ZTM_PDSCV_SFM:
id: "2684578173177298944"
description: "VAV with supply air and zone air monitoring"
is_canonical: true
implements:
  - VAV
  - ZTM
  - PDSCV
  - SFM
opt_uses:
- failed_alarm
```

Hmm, that's not right! 🤔

This block indicates the entity type's field associations. This entity type uses the **`opt_uses`** block, which means the field association **`failed_alarm`** is optional. The **`uses`** block indicates a required field association.

Try again

Back

Next

Knowledge check 4

A sample entity type is on the right.
It has field associations.

Are the field associations required or optional?

Select the best answer from the options listed below.

- Required
- Optional
- This doesn't have field associations

```
GM_STANDARD:
  id: "9159528481871101952"
  description: "The typical building gas meter."
  implements:
  - GM
  is_canonical: true
  uses:
  - gas_flowrate_sensor
  - gas_volume_accumulator
  - gas_temperature_sensor
```

Back

Next

Google

Knowledge check 4

A sample entity type is on the right.
It has field associations.

Are the field associations required or optional?

Select the best answer from the options listed below.

Required

Optional

This doesn't have field associations

That's right! 🎉

The **`uses`** block indicates there are **required** fields.

```
GM_STANDARD:
  id: "9159528481871101952"
  description: "The typical building gas
meter."
  implements:
  - GM
  is_canonical: true
  uses:
  - gas_flowrate_sensor
  - gas_volume_accumulator
  - gas_temperature_sensor
```

If these were optional fields, then the **`opt_uses`** block would have been used.

Back

Next

Google

Knowledge check 4

A sample entity type is on the right.
It has field associations.

Are the field associations required or optional?

Select the best answer from the options listed below.

Required

Optional

This doesn't have field associations

Close... but not quite right! 🤔

If this entity type had **optional** fields, then the `opt_uses` block would've been used.

```
GM_STANDARD:
  id: "9159528481871101952"
  description: "The typical building gas meter."
  implements:
  - GM
  is_canonical: true
  uses:
  - gas_flowrate_sensor
  - gas_volume_accumulator
  - gas_temperature_sensor
```

Try again

Back

Next

Google

Knowledge check 4

A sample entity type is on the right.
It has field associations.

Are the field associations required or optional?

Select the best answer from the options listed below.

Required

Optional

This doesn't have field definitions

Close... but not quite right! 🤔

This entity type definitely has field associations.
Hint: look for the **`uses`** block and/or **`opt_uses`** block.

```
GM_STANDARD:
  id: "9159528481871101952"
  description: "The typical building gas meter."
  implements:
  - GM
  is_canonical: true
  uses:
  - gas_flowrate_sensor
  - gas_volume_accumulator
  - gas_temperature_sensor
```

Try again

Back

Next

Google

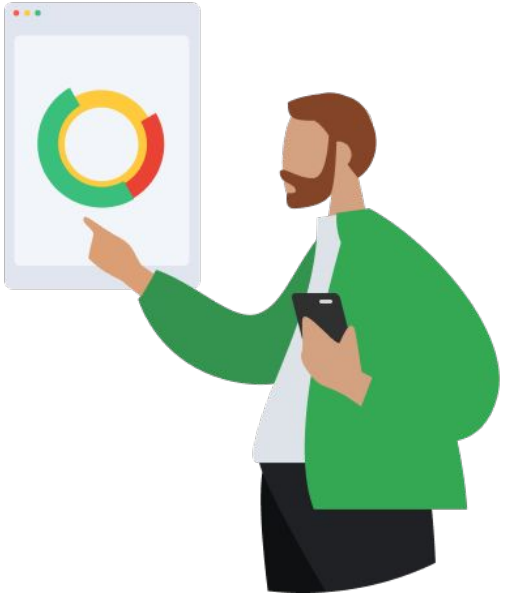
Lesson 6 summary

Let's review what you learned about:

- Entity types
- Abstract, general, and canonical types
- Entity type attributes

Now you should be able to:

- Describe an entity type.
- Identify an entity type in source code.
- Explain the difference between general, abstract, and canonical types.
- Recognize the attributes of an entity type.



Back

Next

You completed Lesson 6!

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

Ready for Lesson 7?

Let's go!

Back

Google

Helpful resources

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

- digitalbuildings / ontology / yaml / resources / entity_types
  Contains all of the available abstract and global entity types.
- digitalbuildings / ontology
  Contains the documentation and configuration files for the DBO.
- digitalbuildings / ontology / docs / ontology.md
  Provides an overview of the structure and principles of the ontology.
- digitalbuildings / ontology / docs / model.md
  Describes the conventions used in the DBO abstract model.
- Digital Buildings Project GitHub
  Contains source code, tooling, and documentation for the DBO.