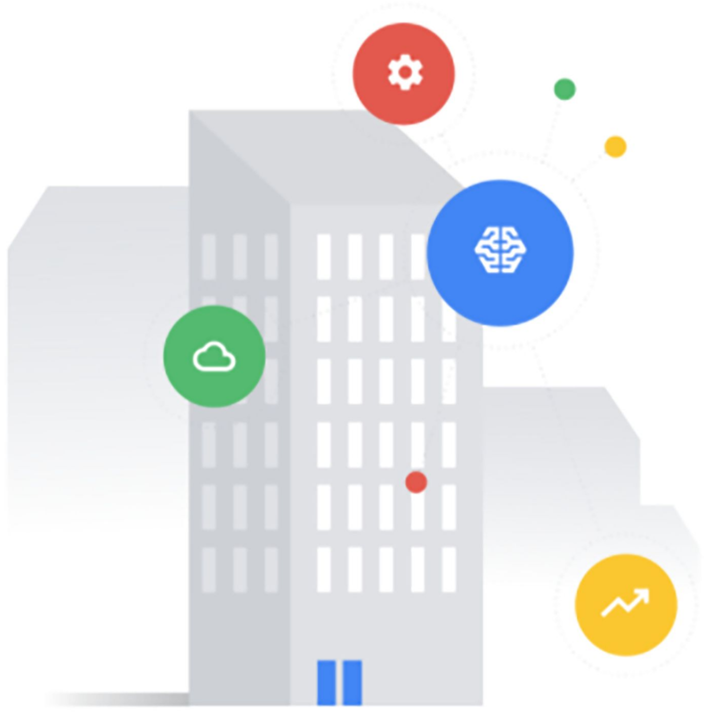
Module 1 | **Appendix**

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!

Google

Appendix 1

DBO compared to other ontologies

Digital Buildings Ontology (DBO)

Project Haystack

BrickSchema

Back

Next

Google

DBO compared to other ontologies

The DBO was custom-made to fit the needs of the Digital Buildings Project.

Instead of using an existing ontology like Project Haystack or BrickSchema, we built our own.

The reason

We needed two important features:

- Repeatable application driven by tooling and precedent
- Relationships (i.e., data modeling that can represent a graph)

The result

We ended up with the DBO after adopting the best features from Haystack (e.g., the concept of tags) and Brick (e.g., the concept of relationships).

To the right is a comparison between the DBO, recent versions of Haystack, and Brick.

	DBO	Haystack 3	Haystack 4	Brick Schema
Easy to apply	❌ *	✅	✅	❌
Includes broad coverage of most target domains	✅ **	✅	✅	❌
Represents relationships between entities	✅	✅ ***	✅	✅
Has tooling for validating concrete models	✅	❌	❌	❌
Has tooling for extending the abstract model	✅	❌	❌	❌
Supports common semantic web representations (e.g., TTL, RDF)	✅	❌	✅	✅
Has a body of precedent associated with the abstract model	✅	❌	❌	❌
Approaches concept models pragmatically	✅	✅	✅	❌
Includes the concept of composable functionality	✅	✅	✅	❌
Field lists specified by functionality	✅	❌	❌	❌

* The DBO may not be easy to apply now, but we're working on tooling, training like this, and more to make it easier.

** Certain domains (such as fire and INFO_TECH) continue to be developed, but the most common domains are covered.

*** Limited applications.

Back

Next

DBO compared to Haystack

Subfields are similar to the concept of a “tag” in Haystack.

The DBO actually includes many of the same tags that exist in Haystack. However, subfields and tags usually have different definitions and purposes.

DBO subfields

```
detection: “Process of identifying conditions.”
```

`discharge``: “Media leaving system to enter ambient conditioned space. Typically applies only to air-side systems.”`

```
east: “Cardinal direction; opposite of west”
```

Haystack tags

```
directZone: AHU supplies air directly to the zone
```

`discharge``: Duct for air leaving an equipment`

```
diverting: Three way valve which inputs one pipe and diverts two output pipes
```

Back

Next

Google

DBO compared to Haystack

Fields are similar to the concept of a “proto” in Haystack.

The DBO combines subfields to form fields in the same way that points can be assigned multiple tags in Haystack. However, fields require subfields to be grouped together in a specific order.

These DBO and Haystack samples define the same concept. However, the DBO field is grouped while the Haystack proto isn’t.

DBO field	**Haystack proto**

`zone_air_temperature_sensor`	`temperature`
	`zone`
	`sensor`

Back

Next

Google

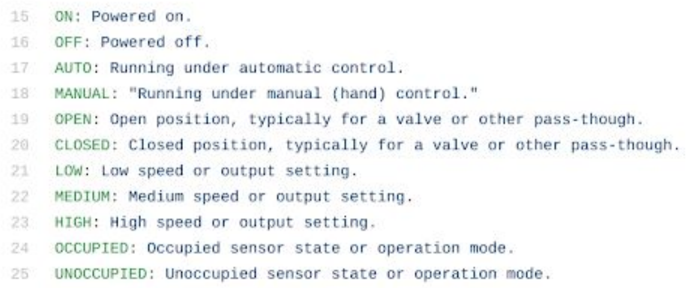
DBO compared to Haystack

States are similar to the concept of “enums” in Haystack.

The concept of a state and an enum is nearly identical. However, the individual states and their definitions in the DBO are very different from the individual enums in Haystack. They’re also different in terms of their utilization. In the DBO, fields must send standard states. In Haystack, there’s generally no assignment of raw data to standard enums except in exceptional cases. In these cases, they’re usually attached to the tags rather than the proto.

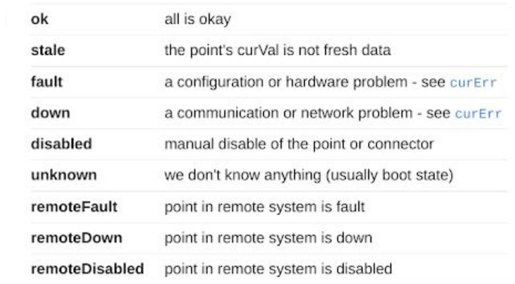
Shown below are samples of individual DBO states and Haystack enums. All of them can describe a device’s condition, but none of them are identical in name or definition.

DBO states



```
ON: Powered on.
OFF: Powered off.
AUTO: Running under automatic control.
MANUAL: "Running under manual (hand) control."
OPEN: Open position, typically for a valve or other pass-though.
CLOSED: Closed position, typically for a valve or other pass-though.
LOW: Low speed or output setting.
MEDIUM: Medium speed or output setting.
HIGH: High speed or output setting.
OCCUPIED: Occupied sensor state or operation mode.
UNOCCUPIED: Unoccupied sensor state or operation mode.
```

Haystack enums



ok	all is okay
stale	the point's curVal is not fresh data
fault	a configuration or hardware problem - see `curErr`
down	a communication or network problem - see `curErr`
disabled	manual disable of the point or connector
unknown	we don't know anything (usually boot state)
remoteFault	point in remote system is fault
remoteDown	point in remote system is down
remoteDisabled	point in remote system is disabled

Back

Next

Google

DBO compared to Haystack

Entity types are a fairly unique concept of the DBO.

The DBO supports abstract types that represent abstract functionality and canonical types that combine abstract concepts to represent a specific type of equipment. Haystack does something similar with tags, but there's no association of abstract tag concepts (such as `variableAirVolume`) with a prescribed set of protos. Haystack currently has no concept of a canonical type.

DBO entity type

```
FAN SS DWI:
  id: "11319382735195209728"
  description: "Dishwasher-interlocked fan."
  is canonical: true
  implements:
  - FAN
  - SS
  uses:
  - dishwasher_run_status
```

Haystack

Back

Next

DBO compared to Haystack

Mapping data is configured differently between the DBO and Haystack.

The DBO uses the building configuration file to map data. Haystack allows for many different file types including Trio, Zinc, and JSON.

DBO mapping

```
DD-1-4-11:
  connections:
    A-0000: CONTAINS
  translation:
    zone_air_unoccupied_cooling_temperature_setpoint:
      present-value: points.a-0092.present_value
      units:
        key: points.a-0092.units
        value:
          degrees_fahrenheit: °F
```

Haystack data model

```
id: @a-0002
dis:Alpha Airside AHU-2 Chilled
Water Valve
chilled
cmd
cool
cur
custom:{description:"Clg_Valve_Cmd"
}
equipRef:@a-0001
his
kind:Number
point
siteRef:@a-0000
tz:Denver
unit"%"
valve
water
...
```

Back

Next

Google

DBO compared to Haystack

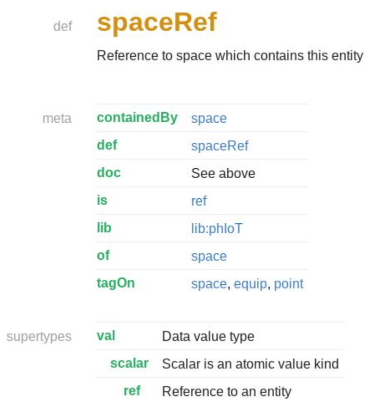
Connections are similar to the concept of a “relationship” in Haystack.

However, relationships are a relatively new addition to Haystack 4. Similar to DBO, there are many different types of relationships that can be made.

DBO connection

```
VAV-32:
  type: HVAC/VAV
  connections:
    UK-LON-6PS-1: CONTAINS
    AHU-123: FEEDS
```

Haystack relationship



def **spaceRef**

Reference to space which contains this entity

meta		
	containedBy	space
	def	spaceRef
	doc	See above
	is	ref
	lib	lib:phIoT
	of	space
	tagOn	space, equip, point

supertypes		
	val	Data value type
	scalar	Scalar is an atomic value kind
	ref	Reference to an entity

Back

Next

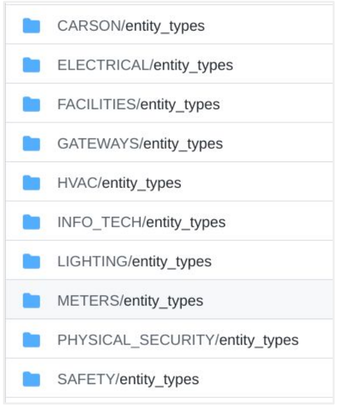
DBO compared to Haystack

Namespaces are used by the DBO and Haystack.

Both ontologies use namespacing to organize their concepts into domains. However, names and arrangement of Haystack namespaces are different from the DBO.

DBO namespaces

The DBO organizes its concepts into intuitive, self-describing namespaces. We've done this to ease the transition of modeling with a new ontology.

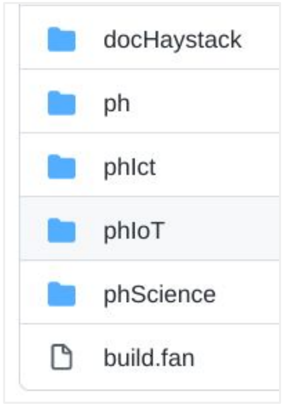


CARSON/entity_types
ELECTRICAL/entity_types
FACILITIES/entity_types
GATEWAYS/entity_types
HVAC/entity_types
INFO_TECH/entity_types
LIGHTING/entity_types
METERS/entity_types
PHYSICAL_SECURITY/entity_types
SAFETY/entity_types

Haystack namespaces

Haystack organizes its concepts into four buckets:

- The **ph** bucket is the global namespace containing the highest level of concepts like the definitions for "relationship," "entity," and "kinds."
- The **phIct** bucket contains concepts of IT-related equipment including controllers, phones, and laptops.
- The **phIoT** bucket contains concepts of IoT system equipment that span across the DBO's ELECTRICAL, FACILITIES, HVAC, and METERS namespaces.
- The **phScience** bucket contains tags for things like dimensional units. The DBO considers these global concepts.



docHaystack
ph
phIct
phIoT
phScience
build.fan

Back

Next

Google

Appendix complete!

Back

Helpful resources

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

- Digital Buildings Project GitHub
  Contains source code, tooling, and documentation for the DBO.