Welcome!
Brick Consortium Kickoff
Winter 2021
Agenda

- Intro to Brick, the Consortium, and this kickoff
- A double-click on Brick, and how it works under the hood
  - The Brick Working Groups
  - The Brick Roadmap
- The way forward and getting involved
What is Brick?

Brick Schema is:

- A uniform metadata schema that defines how the building data is modeled.
- Standardized semantic descriptions of the physical, logical, and virtual entities in buildings and the relationship between them.
- Focus on interoperability, not replacement of other specialized standards
- Open Source project – all development is on Github
- Initiated by researchers from the academic community in 2015.

A Brick Model is a digital representation of a building that adheres to the Brick schema.
What industry-wide customer problems does it solve?

- **Allows** owner of the built-environment to really own the data
- **Empowers** the customer to bring in the best-in-class third party applications; Manufacturer and service-provider can be different
- **Allows** third-party applications to work independent of the data silos
- **Provides** an integration “platform” for data from disparate data sources
- **Allows** for true “plug and play” even if you have multiple manufacturers’ systems in the building environment
Brick v1.2 and what’s next

- Brick 1.2.0 released Feb 20th!
  - Main artifact: a schema defined in RDF – Brick.ttl
  - Typical open source project: BSD license, hosted on Github, community discusses on mailing list and then creates issues, developers create pull requests, software released through appropriate channels
  - Lots of contributors, thanks!
  - Future development will continue this way

- Also tools, datasets, and best practices documentation

- Issues to address moving forward
  - How do we grow the ecosystem and solve industry problems?
  - Standards exist over decades, how do we go that long?
  - How does a community fairly govern a standard?
Introducing the Brick Consortium

- The Brick Consortium, Inc. is a non-profit membership corporation whose purpose is to encourage the research and development of Brick Schema Specifications for the built environment and any supporting tooling, documentation, and best practices necessary to promote Brick.

- The consortium develops the Brick Schema Specifications as open source through member participation, addressing an important industry and societal need by helping to make data of the built environment interoperable.

- The consortium is made up of researchers, technology providers, integrators, and building operators, and will serve as an organization that can provide the long-term support necessary to maintain and enhance the Brick schema until 2040 and beyond.

- The consortium provides governance for the Brick Schema specifications and conformance testing protocols and provides tooling and a repository of reference models and canonical use cases.

- The consortium also funds the research of work related to Brick and the built environment and works to evangelize the use of the Brick schema. Governance and development of Brick Schema, tooling, conformance testing, and canonical use case and reference solution.
Brick Consortium Membership

- Membership in the Brick Consortium is open to commercial entities, universities and non-profit research institutions, and individual academic researchers.

- Membership for commercial entities is available as a full member or a contributing member.

- A contributing member does not have a vote for the Steering Committee or can chair committees but is otherwise fully eligible to participate.

- Universities and non-profit institutions do not pay any membership fees. In some cases, individual academic researchers may join in lieu of their institution joining.

- Membership is not required to participate in the open source development efforts
Brick Development with the Consortium

- Schema development (e.g. the standard): open source development, just like today
  - New: members vote before a release, each member org get a vote
- Work happens in working groups, natural evolution of current effort
- The elected Technical Committee creates new work groups and coordinates between the groups
Agenda for today

- Brick in detail – Gabe Fierro, UC Berkeley
  - Review of Brick
  - What’s new in Brick 1.2
  - How we develop Brick
  - Software to support Brick

- Brick Working Groups
  - Full organization structure and how to join

- The Brick Roadmap – a discussion
  - Specifics of Brick 1.2.1 and 1.3
  - Thinking bigger: moving the industry forward
Brick Overview and Refresher

Gabe Fierro
Working with data in buildings is **hard**

... because there is no consistent, machine-readable representation of data sources and their context

We need **semantic metadata**
State of Building Metadata

Building Management System (BMS): the “operating system” for your building

Point: a source of data from a cyberphysical system

10s or 100s of thousands of points in a typical CPS
State of Building Metadata

- Open up your **building management/automation system**, look at the point names

<table>
<thead>
<tr>
<th>Building/Subsystem</th>
<th>Labeling/Naming Scheme</th>
</tr>
</thead>
<tbody>
<tr>
<td>SODA2S14_SMK</td>
<td>AHU.AHU01.CAV1-1:DMPRPOS</td>
</tr>
<tr>
<td>SODA1S11_MAT</td>
<td>AHU.AHU01.CAV1-1:HTG O</td>
</tr>
<tr>
<td>SODA3R315_RVAV</td>
<td>AHU.AHU01.CAV1-1:SUPFLOW</td>
</tr>
<tr>
<td>SODA3R723_ASO</td>
<td>AHU.AHU01.CAV1-1:ZN T</td>
</tr>
<tr>
<td>SODA3R327_AGN</td>
<td>AHU.AHU01.CAV2-1:DAT</td>
</tr>
<tr>
<td>SODH1P02_FLT</td>
<td>AHU.AHU01.CAV2-1:HTG O</td>
</tr>
<tr>
<td>SODA3R798_ART</td>
<td>AHU.AHU01.CAV2-1:DMPRPOS</td>
</tr>
<tr>
<td>SODA1R405B_ARS</td>
<td>AHU.AHU01.CAV2-1:SUPFLOW</td>
</tr>
<tr>
<td>SODA3R683_RVAV</td>
<td>AHU.AHU01.CAV2-1:ZN T</td>
</tr>
<tr>
<td>SODA1R405B_ART</td>
<td>AHU.AHU01.CCV</td>
</tr>
<tr>
<td>SODA3R311_AGN</td>
<td>AHU.AHU01.CHWCHW.UNT:CHW FLOW</td>
</tr>
<tr>
<td>SODH1__L_L</td>
<td>AHU.AHU01.CHWCHW.UNT:HW FLOW</td>
</tr>
<tr>
<td>SODC1SP03_FLT</td>
<td>AHU.AHU01.Cooling Enable</td>
</tr>
<tr>
<td>SODA4R645_RVAV</td>
<td>AHU.AHU01.CM</td>
</tr>
<tr>
<td>SODA1R288_AGN</td>
<td>AHU.AHU01.HP.UNT:ZN T</td>
</tr>
<tr>
<td>SODA3R419_AGN</td>
<td>AHU.AHU01.HSP</td>
</tr>
<tr>
<td>SODA3S311_ASO</td>
<td>AHU.AHU01.LSP</td>
</tr>
<tr>
<td>SODA2S14_P_VR</td>
<td>AHU.AHU01.LTD</td>
</tr>
<tr>
<td>SODA3S311_ASO</td>
<td>AHU.AHU01.MAX.ZONE.DAMPER</td>
</tr>
<tr>
<td>SODA2S14_P_VR</td>
<td>AHU.AHU01.MAX.ZONE.HEATING</td>
</tr>
<tr>
<td>SODA4S3132_STA</td>
<td>AHU.AHU01.MIN OA</td>
</tr>
<tr>
<td></td>
<td>AHU.AHU01.Mixed Air Damper Position</td>
</tr>
<tr>
<td></td>
<td>AHU.AHU01.Mixed Air Temp</td>
</tr>
<tr>
<td></td>
<td>Trunk.VAV2-12.OCCHTGFL</td>
</tr>
<tr>
<td></td>
<td>Trunk.CentralPlant.HWP2-RST</td>
</tr>
<tr>
<td></td>
<td>Trunk.VAV2-4.BOXHTG</td>
</tr>
<tr>
<td></td>
<td>Trunk.VAV2-9.SUPFLOSP</td>
</tr>
<tr>
<td></td>
<td>Trunk.CentralPlant.CHWP4-S</td>
</tr>
<tr>
<td></td>
<td>Trunk.VAV2-7.COMMONSP</td>
</tr>
<tr>
<td></td>
<td>Trunk.VAV1-5.SUPFLOSP</td>
</tr>
<tr>
<td></td>
<td>Trunk.VAV2-10.S-VP</td>
</tr>
<tr>
<td></td>
<td>Trunk.VAV2-3.SUPFLOSP</td>
</tr>
<tr>
<td></td>
<td>Trunk.VVT-4.UNOCDB</td>
</tr>
<tr>
<td></td>
<td>Trunk.VAV2-10.BOXHTG</td>
</tr>
<tr>
<td></td>
<td>Trunk.VVT-5.ZN-T</td>
</tr>
<tr>
<td></td>
<td>Trunk.CentralPlant.HWP2-A.Alarm1</td>
</tr>
<tr>
<td></td>
<td>Trunk.VVT-1.ZN-T</td>
</tr>
<tr>
<td></td>
<td>Trunk.VAV2-8.COMMONSP</td>
</tr>
<tr>
<td></td>
<td>Trunk.VAV1-1.BOXMODE</td>
</tr>
<tr>
<td></td>
<td>Trunk.AHU-3.MA-T</td>
</tr>
</tbody>
</table>

- 3 different buildings/BMS/subsystems → 3 (or more) different labeling/naming schemes
Make Working with Building Data Easier

- Most building data resides in **opaque data silos**
  - Unclear, inconsistent, hard-to-interpret labels
  - *(if you have access to it at all)*

- Existing metadata standards focus on other perspectives of the building
  - Design, construction
  - Asset management
  - Commissioning, Auditing

- Need a metadata representation designed for **data-driven building software**
  - Unlock potential of building data
  - *Preserve existing investments*
Brick Schema

- Graph-based metadata schema for smart buildings

- Capture physical, logical, virtual entities in buildings using a class hierarchy

- Capture the necessary relationships between them

- Use Brick to describe timeseries data and its context
# Brick Classes

## Point
- Sensor
  - Temperature Sensor
    - Air Temperature Sensor
      - Supply Air Temperature Sensor
      - Return Air Temperature Sensor
    - Humidity Sensor
      - ...%
  - ...%
- Setpoint
  - Damper Position Setpoint
  - Temperature Setpoint
  - Command
    - ...%

## Equipment
- HVAC
  - Air Handling Unit
  - Terminal Unit
    - Variable Air Volume Box
    - Constant Volume Box
    - Fan Coil Unit
    - ...%
  - ...%
- Lighting
  - Switches
  - Luminaires and Lamps
    - ...%

## Location
- Room
  - Conference Room
  - Classroom
  - Lecture Hall
  - Kitchen
  - Reception
  - ...%
  - Building
  - Floor
  - Zone
    - HVAC Zone
    - Lighting Zone
    - Fire Zone
    - ...%
Brick Overview

- Three main concepts, each the root of their own **class hierarchy**

- **Classes** provide definition, organization to **entities**

- **Entities** are the physical, logical and virtual “things”

- **Relationships** dictate how entities correspond and relate to each other
- Have a set of physical, virtual “things” and points that an application wants to refer to
- Brick defines a **hierarchical class structure** to define **standard names** for equipment, points, locations, etc
- Brick defines a set of **standard relationships** that describe how things are connected
A Brick model represents the assets and relationships and data in a building
An application queries a Brick model to retrieve the data + configuration it needs.
- Relate Brick Point classes to timeseries data
- Contextualize data in existing datastores
Full Brick model of the sample site

Part of Location model
select ?saf_ref ?volume ?unit where {

# Identify the room’s VAV
?zone rdf:type brick:HVACZone.
?device rdf:type brick:TerminalUnit.

# Get the air flow sensor’s timeseries ref.
?device brick:hasPoint ?saf.
?saf brick:hasTimeseriesID ?saf_ref.

# Get the room’s volume.
?room brick:hasVolume ?quantity.
?quantity brick:value ?volume
?quantity brick:hasUnit ?unit

}
Brick Model Validation

User-defined “templates” to validate Brick models

Model XYZ VAV

- instance of brick:VAV
- 1 brick:Zone_Air_Temperature_Sensor
- 1 brick:Temperature_Setpoint
- 1 model_ABC_damper

Model ABC Damper

- instance of brick:Damper
- 1 brick:Damper_Position_Command

Templates ensure that Brick models match expectations, requirements

Verify that Brick is being used correctly
Brick Resources

- Developer-focused documentation:
  - https://docs.brickschema.org/

- Concept documentation:
  - https://brickschema.org/ontology

- Download Brick and examples:
  - https://brickschema.org/resources

- Python library:
  - https://brickschema.readthedocs.io/
Brick ontology: formal definitions of concepts, relationships

Brick model: the graph representing a particular building
What’s in a Name?

- Labels like these often the *only* up-to-date and readily-available source of metadata
- Site-specific conventions; sometimes no labels at all
- No consistent, standard metadata for cyberphysical system data
OUTLINE

1. Brick Implementation
   • The Brick Distribution
   • Extending Brick
2. Brick Tooling
   • brickschema Python package
   • brick-builder and OpenRefine
example:tstat_ABC rdf:type brick:Thermostat ; brick:hasPoint example:sen1 .

example:sen1 rdf:type brick:Temperature_Sensor .
SIMPLE BRICK CLASS

brick:Temperature_Sensor a owl:Class;
	rdfs:label "Temperature Sensor";
	rdfs:subClassOf [ owl:intersectionOf ( _:has_Point _:has_Sensor _:has_Temperature ) ],

brick:Sensor;

owl:equivalentClass [ owl:intersectionOf ( [ a owl:Restriction;

owl:hasValue brick:Temperature;

owl:onProperty brick:measures ] ) ];

skos:definition "Measures temperature: the physical property of matter that quantitatively

brick:hasAssociatedTag tag:Point,

tag:Sensor,

tag:Temperature."
brick:Temperature_Sensor a owl:Class ;
   rdfs:label "Temperature Sensor" ;
   rdfs:subClassOf [ owl:intersectionOf ( _:has_Point _:has_Sensor _:has_Temperature ) ],
       brick:Sensor ;
   owl:equivalentClass [ owl:intersectionOf ( [ a owl:Restriction ;
       owl:hasValue brick:Temperature ;
       owl:onProperty brick:measures ] ) ] ;
   skos:definition "Measures temperature: the physical property of matter that quantitatively
brick:Temperature_Sensor a owl:Class ;
  rdfs:label "Temperature Sensor";
  rdfs:subClassOf [ owl:intersectionOf ( __:has_Point __:has_Sensor __:has_Temperature ) ],
    brick:Sensor ;
  owl:equiv Class [ owl:intersectionOf ( [ a owl:Restriction ;
    owl:hasValue brick:Temperature ;
    owl:onProperty brick:measures ] ) ] ;
  skos:definition "Measures temperature: the physical property of matter that quantitatively
  brick:hasAssociatedTag tag:Point,
    tag:Sensor,
    tag:Temperature .
SIMPLE BRICK CLASS

brick:Temperature_Sensor a owl:Class ;
  rdfs:label "Temperature Sensor" ;
  rdfs:subClassOf [ owl:intersectionOf ( __:has_Point __:has_Sensor __:has_Temperature ) ],
  brick:Sensor ;
  owl:equivalentClass [ owl:intersectionOf ( [ a owl:Restriction ;
    owl:hasValue brick:Temperature ;
    owl:onProperty brick:measures ] ) ] ;
  skos:definition "Measures temperature: the physical property of matter that quantitatively
bricks:hasAssociatedTag tag:Point,
  tag:Sensor,
  tag:Temperature .
brick:Temperature_Sensor a owl:Class ;
  rdfs:label "Temperature Sensor" ;
  rdfs:subClassOf [ owl:intersectionOf ( __:has_Point __:has_Sensor __:has_Temperature ) ],
  brick:Sensor ;
owl:equivalentClass [ owl:intersectionOf ( [ a owl:Restriction ;
  owl:hasValue brick:Temperature ;
  owl:onProperty brick:measures ] ) ] ;
skos:definition "Measures temperature: the physical property of matter that quantitatively
brick:hasAssociatedTag tag:Point,
  tag:Sensor,
  tag:Temperature .
### Extending Brick

**Uniform metadata schema for buildings**

- [brickschema.org/](https://brickschema.org/)
- [Readme](https://brickschema.org/)
- [View license](https://brickschema.org/)

#### Releases
14
- [Brick v1.2.0](https://brickschema.org/) (Latest)
  - 3 days ago
  + 13 releases

#### Packages
No packages published
Publish your first package

#### Contributors
11
- [avatars](https://brickschema.org/)

#### Environments
1
- [github-pages](https://brickschema.org/)
  - Active

<table>
<thead>
<tr>
<th>File</th>
<th>Version</th>
<th>Changes</th>
<th>Comments</th>
<th>Date</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>.github/workflows</code></td>
<td>v1.2</td>
<td>add changelog for v1.2 (#235)</td>
<td>resovled 3 days ago</td>
<td>663 commits</td>
</tr>
<tr>
<td><code>alignments</code></td>
<td>v1.2</td>
<td>release (#214)</td>
<td>3 days ago</td>
<td></td>
</tr>
<tr>
<td><code>brickarc</code></td>
<td>v1.2</td>
<td>release (#214)</td>
<td>3 days ago</td>
<td></td>
</tr>
<tr>
<td><code>examples</code></td>
<td>v1.2</td>
<td>release (#214)</td>
<td>3 days ago</td>
<td></td>
</tr>
<tr>
<td><code>extensions</code></td>
<td>v1.2</td>
<td>release (#214)</td>
<td>3 days ago</td>
<td></td>
</tr>
<tr>
<td><code>shacl</code></td>
<td>v1.2</td>
<td>release (#214)</td>
<td>3 days ago</td>
<td></td>
</tr>
<tr>
<td><code>support</code></td>
<td></td>
<td>use associate_units to simplify model a little</td>
<td>10 months ago</td>
<td></td>
</tr>
<tr>
<td><code>tests</code></td>
<td>v1.2</td>
<td>release (#214)</td>
<td>3 days ago</td>
<td></td>
</tr>
<tr>
<td><code>tools</code></td>
<td></td>
<td>Removed particulate air type. This is stable.</td>
<td>5 months ago</td>
<td></td>
</tr>
<tr>
<td><code>.flake8</code></td>
<td></td>
<td>Trying out the Black Python formatter (#106)</td>
<td>12 months ago</td>
<td></td>
</tr>
<tr>
<td><code>.gitattributes</code></td>
<td>v1.2</td>
<td>release (#214)</td>
<td>3 days ago</td>
<td></td>
</tr>
<tr>
<td><code>.gitignore</code></td>
<td></td>
<td>add gitignore</td>
<td>4 months ago</td>
<td></td>
</tr>
<tr>
<td><code>.gitmodules</code></td>
<td></td>
<td>Remove tools directory</td>
<td>4 years ago</td>
<td></td>
</tr>
<tr>
<td><code>.pre-commit-config.yml</code></td>
<td></td>
<td>Trying out the Black Python formatter (#106)</td>
<td>12 months ago</td>
<td></td>
</tr>
<tr>
<td><code>CHANGELOG.md</code></td>
<td></td>
<td>add changelog for v1.2 (#235)</td>
<td>3 days ago</td>
<td></td>
</tr>
<tr>
<td><code>CONTRIBUTING.md</code></td>
<td>v1.2</td>
<td>release (#214)</td>
<td>3 days ago</td>
<td></td>
</tr>
<tr>
<td><code>LICENSE</code></td>
<td></td>
<td>use semicolons to separate items in a list when the items contain commas</td>
<td>4 years ago</td>
<td></td>
</tr>
<tr>
<td><code>Makefile</code></td>
<td>v1.2</td>
<td>release (#214)</td>
<td>3 days ago</td>
<td></td>
</tr>
</tbody>
</table>
# bricksrc/sensor.py

```python
sensor_definitions = {
    "Sensor": {
        "tags": [TAG.Point, TAG.Sensor],
        "subclasses": {
            "Temperature_Sensor": {
                "tags": [TAG.Point, TAG.Sensor, TAG.Temperature],
                "substances": [[BRICK.measures, BRICK.Temperature]],
            },
            "Air_Temperature_Sensor": {
                "tags": [TAG.Point, TAG.Sensor, TAG.Temperature, TAG.Air],
                "substances": [
                    [BRICK.measures, BRICK.Temperature],
                    [BRICK.measures, BRICK.Air],
                ],
            },
        },
    },
}
```

EXTENDING BRICK

sensor_definitions = {
    "Sensor": {
        "tags": [TAG.Point, TAG.Sensor],
        "subclasses": {
            "Temperature_Sensor": {
                "tags": [TAG.Point, TAG.Sensor, TAG.Temperature],
                "substances": [[BRICK.measures, BRICK.Temperature]],
                "subclasses": {
                    "Air_Temperature_Sensor": {
                        "tags": [TAG.Point, TAG.Sensor, TAG.Temperature, TAG.Air],
                        "substances": [
                            [BRICK.measures, BRICK.Temperature],
                            [BRICK.measures, BRICK.Air]
                        ]
                    }
                }
            }
        }
    }
}
EXTENDING BRICK

```
# bricksrc/sensor.py
sensor_definitions = {
    "Sensor": {
        "tags": [TAG.Point, TAG.Sensor],
        "subclasses": {
            "Temperature_Sensor": {
                "tags": [TAG.Point, TAG.Sensor, TAG.Temperature],
                "substances": [[BRICK.measures, BRICK.Temperature]],
                "subclasses": {
                    "Air_Temperature_Sensor": {
                        "tags": [TAG.Point, TAG.Sensor, TAG.Temperature, TAG.Air],
                        "substances": [
                            [BRICK.measures, BRICK.Temperature],
                            [BRICK.measures, BRICK.Air],
                        ]
                    }
                }
            }
        }
    }
}
```
EXTENDING BRICK

sensor_definitions = {
    "Sensor": {
        "tags": [TAG.Point, TAG.Sensor],
        "subclasses": {
            "Temperature_Sensor": {
                "tags": [TAG.Point, TAG.Sensor, TAG.Temperature],
                "substances": [[BRICK.measures, BRICK.Temperature]],
                "subclasses": {
                    "Air_Temperature_Sensor": {
                        "tags": [TAG.Point, TAG.Sensor, TAG.Temperature, TAG.Air],
                        "substances": [
                            [BRICK.measures, BRICK.Temperature],
                            [BRICK.measures, BRICK.Air]
                        ]
                    }
                }
            }
        }
    }
}
```
# bricksrc/sensor.py
sensor_definitions = {
    "Sensor": {
        "tags": [TAG.Point, TAG.Sensor],
        "subclasses": {
            "Temperature_Sensor": {
                "tags": [TAG.Point, TAG.Sensor, TAG.Temperature],
                "substances": [[BRICK.measures, BRICK.Temperature]],
                "subclasses": {
                    "Air_Temperature_Sensor": {
                        "tags": [TAG.Point, TAG.Sensor, TAG.Temperature, TAG.Air],
                        "substances": [
                            [BRICK.measures, BRICK.Temperature],
                            [BRICK.measures, BRICK.Air]
                        ]
                    }
                }
            }
        }
    }
}
```
How to add a Glycol Temperature Sensor

1. Find an appropriate parent class
2. Add the class definition to the Python file
3. Add textual definition to definitions.csv
4. Compile and run tests
EXTENDING BRICK

How to add a Glycol Temperature Sensor

- Find an appropriate parent class
- Add the class definition to the Python file

```python
# bricksrc/sensor.py
sensor_definitions = {
    "Sensor": {
        "tags": [TAG.Point, TAG.Sensor],
        "subclasses": {
            "Temperature_Sensor": {
                "tags": [TAG.Point, TAG.Sensor, TAG.Temperature],
                "substances": [[BRICK.measures, BRICK.Temperature]],
                "subclasses": {
                    "Glycol_Temperature_Sensor": {
                        "tags": [TAG.Point, TAG.Sensor, TAG.Temperature, TAG.Glycol],
                        "substances": [
                            [BRICK.measures, BRICK.Temperature],
                            [BRICK.measures, BRICK.Glycol],
                        ]
                    }
                }
            }
        }
    }
}
```
EXTENDING BRICK

How to add a Glycol Temperature Sensor

- Add textual definition to definitions.csv

```csv
# brickschema.org/definitions.csv
https://brickschema.org/schema/Brick#Generator_Room,"A room for electrical equipment, specifically electrical power equipment such as transformers, generators, etc."
https://brickschema.org/schema/Brick#Glycol_Temperature_Sensor,"A sensor which measures the temperature of glycol"
https://brickschema.org/schema/Brick#HVAC_Equipment,"See Heating_Ventilation_Air Conditioning_Equipment"
```
EXTENDING BRICK

How to add a Glycol Temperature Sensor

- Compile using the Makefile

```
1 (venv) gabe@arkestra:~/src/Brick$ make
2  mkdir -p extensions
3  python generate_brick.py
4  Checking 58 containers
5  Non-zero exit code 1 with message b'\nAllegroGraph Server Edition 7.1.0 (rc6), built on January
6  Checking 58 containers
7  Non-zero exit code 1 with message b'\nAllegroGraph Server Edition 7.1.0 (rc6), built on January
8  2021-02-22:12:01:30,297 WARNING [generate_brick.py:296] Property 'tags' not defined for https:
9  2021-02-22:12:01:31,388 WARNING [generate_brick.py:296] Property 'tags' not defined for https:
```
EXTENDING BRICK

How to add a Glycol Temperature Sensor

- Run unit and integration tests
Submitting Changes to Brick

Changes to Brick are performed through Pull Requests. It is recommended that you become familiar with how to fork a repository and create a pull request.

Setting up Development Environment

Brick requires Python 3.6. We recommend using virtual environments to manage dependencies. We use pre-commit hooks to automatically run code formatters and style checkers when you commit.

1. Check out the Brick repository (or your own fork of it)

   git clone https://github.com/BrickSchema/Brick
   cd Brick

2. Install the virtual environment and set up dependencies

   # creates virtual environment
   python3 -m venv venv

   # activates virtual environment; do this every time you develop on Brick
   source venv/bin/activate

   # install dependencies
   pip install -r requirements.txt

   # install pre-commit hooks
   pre-commit install

3. Run tests to make sure the build is not broken

   make test

4. Whenever you commit, the pre-commit script will run the Black code formatting tool and the flake8 style checker. It will automatically format the code where it can, and indicate when there is a style error. The tools will not commit unformatted code; if you see a "Failed" message, please fix the style and re-commit the code. An example of what this looks like is below; the failed flake8 check results in a short error report at the bottom.

   gabe@arkestra:/src/Brick$ git commit -m 'adding changes to Alarm hierarchy'
   [WARNING] Unstaged files detected.
   [INFO] Stashing unstaged files to /home/gabe/.cache/pre-commit/patch15817700016.
   Error: File to commit failed to format:
   File: /home/gabe/.cache/pre-commit/patch15817700016, Line: 1, Column: 124
   Error: File to commit failed to format:
   File: /home/gabe/.cache/pre-commit/patch15817700016, Line: 1, Column: 124
   File: /home/gabe/.cache/pre-commit/patch15817700016, Line: 1, Column: 126
BRICK TOOLING: brickschema

pip install brickschema

Simple Python package for managing and programming against the Brick ontology

- Load, validate Brick models
- Query, update Brick models
- Simple inference support, extensions, alignments
The graph is the unit of management for Brick

```python
import brickschema

# empty graph with no triples
g = brickschema.Graph()

# creates a new rdflib.Graph with a recent version
# of the Brick ontology preloaded.
g = brickschema.Graph(load_brick=True)

# OR use the absolute latest Brick:
g = brickschema.Graph(load_brick_nightly=True)

# OR create from an existing model

g = brickschema.Graph(load_brick=True).from_haystack(...)
```
The graph is the unit of management for Brick

```python
import brickschema
# empty graph with no triples
g = brickschema.Graph()
# creates a new rdflib.Graph with a recent version of the Brick ontology preloaded.
# OR use the absolute latest Brick:
# OR create from an existing model
# empty graph with no triples
```

```python
# empty graph with no triples
g = brickschema.Graph() # creates a new rdflib.Graph with a recent version of the Brick ontology preloaded.
# OR use the absolute latest Brick:
# OR create from an existing model
```
The graph is the unit of management for Brick
import brickschema

# empty graph with no triples
g = brickschema.Graph()

# creates a new rdflib.Graph with a recent version of the Brick ontology preloaded.
# OR use the absolute latest Brick:
# OR create from an existing model

# empty graph with no triples
# creates a new rdflib.Graph with a recent version of the Brick ontology preloaded.
# OR use the absolute latest Brick:
# OR create from an existing model

The graph is the unit of management for Brick
import brickschema

# load in data files from your file system

# ...or by URL

The graph is the unit of management for Brick
import brickschema

g = brickschema.Graph(load_brick=True)

# load in data files from your file system

g.load_file("mybuilding.ttl")

# ...or by URL

g.parse("https://brickschema.org/ttl/soda_brick.ttl", format="ttl")

The graph is the unit of management for Brick
Reasoning adds implied information to the Brick graph

```python
import brickschema

g = brickschema.Graph(load_owl=True)
g.load_file("mybuilding.ttl")
g.expand(profile="owlrl") # also supports 'rdfs', 'shacl', 'vbis'
g.expand(profile="owlrl+shacl") # create "schedules" of reasoning
```
Reasoning adds implied information to the Brick graph
BRICK TOOLING: brickschema

import brickschema

g = brickschema.Graph(load_brick=True)
g.load_file("mybuilding.ttl")
g.expand(profile="owlrl")

valid, _, resultsText = g.validate()

if not valid:
    print("Graph is not valid!")
    print(resultsText)

Validate the Brick ontology is being used correctly
Extensions and alignments add features, enable integration with other ontologies and metadata representations

```python
import brickschema

g = brickschema.Graph(load_brick=True)
g.load_file("mybuilding.ttl")
g.get_extensions()
# => ['shacl_tag_inference']
g.load_extension('shacl_tag_inference')
g.expand("shacl")  # usually run reasoning after loading extension

9
g.get_alignments()
# => ['VBIS', 'REC', 'BOT']
g.load_alignment('BOT')
g.expand("owlrl")  # usually run reasoning after loading alignment
```
Extensions and alignments add features, enable integration with other ontologies and metadata representations.

```python
import brickschema

g = brickschema.Graph(load_brick=True)
g.load_file("mybuilding.ttl")
g.get_extensions()
# => ['shacl_tag_inference']
g.load_extension('shacl_tag_inference')
g.expand("shacl")  # usually run reasoning after loading extension

g.get_alignments()
# => ['VBIS', 'REC', 'BOT']
g.load_alignment('BOT')
g.expand("owlrl")  # usually run reasoning after loading alignment
```
BRICK TOOLING: brickschema

```python
import brickschema

g = brickschema.Graph(load_brick=True)
g.load_file("mybuilding.ttl")

# perform SPARQL queries on the graph
res = g.query("""SELECT ?afs ?afsp ?vav WHERE {
  ?afs a brick:Air_Flow_Sensor .
  ?afsp a brick:Air_Flow_Setpoint .
  ?afs brick:isPointOf ?vav .
  ?afsp brick:isPointOf ?vav .
  ?vav a brick:VAV
}"")

for row in res:
    (air_flow_sensor, air_flow_setpoint, vav) = row
    print(row)
```

SPARQL queries retrieve parts of the Brick graph for applications
import brickschema

g = brickschema.Graph(load_brick=True)
g.load_file("mybuilding.ttl")
g.serve()
BRICK TOOLING: brick-builder
<table>
<thead>
<tr>
<th>BRICK TOOLING: brick-builder</th>
</tr>
</thead>
<tbody>
<tr>
<td>SODA2S14__SMK</td>
</tr>
<tr>
<td>SODA1S11__MAT</td>
</tr>
<tr>
<td>SODA3R315_RVAV</td>
</tr>
<tr>
<td>SODA3R723__ASO</td>
</tr>
<tr>
<td>SODA3R327__AGN</td>
</tr>
<tr>
<td>SODA1P02__FLT</td>
</tr>
<tr>
<td>SODA3R798__ART</td>
</tr>
<tr>
<td>SODA3R311__AGN</td>
</tr>
<tr>
<td>SODA1R405B_ART</td>
</tr>
<tr>
<td>SODA1R405B__ART</td>
</tr>
<tr>
<td>SODA3R311__AGN</td>
</tr>
<tr>
<td>SODA1R405B__ART</td>
</tr>
<tr>
<td>SODA3R311__AGN</td>
</tr>
<tr>
<td>SODA1R405B__ART</td>
</tr>
<tr>
<td>SODA1R405B__ART</td>
</tr>
<tr>
<td>SODA3R311__AGN</td>
</tr>
<tr>
<td>SODA1R405B__ART</td>
</tr>
<tr>
<td>SODA1R405B__ART</td>
</tr>
<tr>
<td>SODA3R311__AGN</td>
</tr>
<tr>
<td>SODA1R405B__ART</td>
</tr>
<tr>
<td>SODA1R405B__ART</td>
</tr>
<tr>
<td>SODA3R311__AGN</td>
</tr>
<tr>
<td>SODA1R405B__ART</td>
</tr>
<tr>
<td>SODA1R405B__ART</td>
</tr>
<tr>
<td>SODA3R311__AGN</td>
</tr>
<tr>
<td>SODA1R405B__ART</td>
</tr>
<tr>
<td>SODA1R405B__ART</td>
</tr>
<tr>
<td>SODA3R311__AGN</td>
</tr>
<tr>
<td>SODA1R405B__ART</td>
</tr>
<tr>
<td>SODA1R405B__ART</td>
</tr>
<tr>
<td>Trunk.VAV2-10.OCCHTGF</td>
</tr>
<tr>
<td>Trunk.CentralPlant.HWP2-RST</td>
</tr>
<tr>
<td>Trunk.CentralPlant.CHWP4-S</td>
</tr>
<tr>
<td>Trunk.VAV2-7.COMMONSP</td>
</tr>
<tr>
<td>Trunk.VAV1-5.S-VP</td>
</tr>
<tr>
<td>Trunk.VAV2-10.OCCHTGF</td>
</tr>
<tr>
<td>Trunk.CentralPlant.HWP2-RST</td>
</tr>
<tr>
<td>Trunk.CentralPlant.CHWP4-S</td>
</tr>
<tr>
<td>Trunk.VAV2-7.COMMONSP</td>
</tr>
<tr>
<td>Trunk.VAV1-5.S-VP</td>
</tr>
<tr>
<td>Trunk.VAV2-10.OCCHTGF</td>
</tr>
<tr>
<td>Trunk.CentralPlant.HWP2-RST</td>
</tr>
<tr>
<td>Trunk.CentralPlant.CHWP4-S</td>
</tr>
<tr>
<td>Trunk.VAV2-7.COMMONSP</td>
</tr>
<tr>
<td>Trunk.VAV1-5.S-VP</td>
</tr>
<tr>
<td>Trunk.CentralPlant.HWP2-RST</td>
</tr>
<tr>
<td>Trunk.CentralPlant.CHWP4-S</td>
</tr>
<tr>
<td>Trunk.VAV2-7.COMMONSP</td>
</tr>
<tr>
<td>Trunk.VAV1-5.S-VP</td>
</tr>
<tr>
<td>Trunk.CentralPlant.HWP2-RST</td>
</tr>
<tr>
<td>Trunk.CentralPlant.CHWP4-S</td>
</tr>
<tr>
<td>Trunk.VAV2-7.COMMONSP</td>
</tr>
<tr>
<td>Trunk.VAV1-5.S-VP</td>
</tr>
<tr>
<td>Trunk.CentralPlant.HWP2-RST</td>
</tr>
<tr>
<td>Trunk.CentralPlant.CHWP4-S</td>
</tr>
<tr>
<td>Trunk.VAV2-7.COMMONSP</td>
</tr>
<tr>
<td>Trunk.VAV1-5.S-VP</td>
</tr>
<tr>
<td>Trunk.CentralPlant.HWP2-RST</td>
</tr>
<tr>
<td>Trunk.CentralPlant.CHWP4-S</td>
</tr>
<tr>
<td>Trunk.VAV2-7.COMMONSP</td>
</tr>
<tr>
<td>Trunk.VAV1-5.S-VP</td>
</tr>
<tr>
<td>Trunk.CentralPlant.HWP2-RST</td>
</tr>
<tr>
<td>Trunk.CentralPlant.CHWP4-S</td>
</tr>
<tr>
<td>Trunk.VAV2-7.COMMONSP</td>
</tr>
<tr>
<td>Trunk.VAV1-5.S-VP</td>
</tr>
<tr>
<td>Trunk.CentralPlant.HWP2-RST</td>
</tr>
<tr>
<td>Trunk.CentralPlant.CHWP4-S</td>
</tr>
<tr>
<td>Trunk.VAV2-7.COMMONSP</td>
</tr>
<tr>
<td>Trunk.VAV1-5.S-VP</td>
</tr>
<tr>
<td>Trunk.CentralPlant.HWP2-RST</td>
</tr>
<tr>
<td>Trunk.CentralPlant.CHWP4-S</td>
</tr>
<tr>
<td>Trunk.VAV2-7.COMMONSP</td>
</tr>
<tr>
<td>Trunk.VAV1-5.S-VP</td>
</tr>
<tr>
<td>Trunk.CentralPlant.HWP2-RST</td>
</tr>
<tr>
<td>Trunk.CentralPlant.CHWP4-S</td>
</tr>
<tr>
<td>Trunk.VAV2-7.COMMONSP</td>
</tr>
<tr>
<td>Trunk.VAV1-5.S-VP</td>
</tr>
<tr>
<td>Trunk.CentralPlant.HWP2-RST</td>
</tr>
<tr>
<td>Trunk.CentralPlant.CHWP4-S</td>
</tr>
<tr>
<td>Trunk.VAV2-7.COMMONSP</td>
</tr>
<tr>
<td>Trunk.VAV1-5.S-VP</td>
</tr>
<tr>
<td>Trunk.CentralPlant.HWP2-RST</td>
</tr>
<tr>
<td>Trunk.CentralPlant.CHWP4-S</td>
</tr>
<tr>
<td>Trunk.VAV2-7.COMMONSP</td>
</tr>
<tr>
<td>Trunk.VAV1-5.S-VP</td>
</tr>
<tr>
<td>Trunk.CentralPlant.HWP2-RST</td>
</tr>
<tr>
<td>Trunk.CentralPlant.CHWP4-S</td>
</tr>
<tr>
<td>Trunk.VAV2-7.COMMONSP</td>
</tr>
<tr>
<td>Trunk.VAV1-5.S-VP</td>
</tr>
<tr>
<td>Trunk.CentralPlant.HWP2-RST</td>
</tr>
<tr>
<td>Trunk.CentralPlant.CHWP4-S</td>
</tr>
<tr>
<td>Trunk.VAV2-7.COMMONSP</td>
</tr>
<tr>
<td>Trunk.VAV1-5.S-VP</td>
</tr>
<tr>
<td>#</td>
</tr>
<tr>
<td>----</td>
</tr>
<tr>
<td>1754</td>
</tr>
<tr>
<td>1755</td>
</tr>
<tr>
<td>1756</td>
</tr>
<tr>
<td>1757</td>
</tr>
<tr>
<td>1758</td>
</tr>
<tr>
<td>1759</td>
</tr>
<tr>
<td>1760</td>
</tr>
<tr>
<td>1761</td>
</tr>
<tr>
<td>1762</td>
</tr>
<tr>
<td>1763</td>
</tr>
<tr>
<td>1764</td>
</tr>
<tr>
<td>1765</td>
</tr>
<tr>
<td>1766</td>
</tr>
<tr>
<td>1767</td>
</tr>
<tr>
<td>1768</td>
</tr>
</tbody>
</table>

BRICK TOOLING: brick-builder
BRICK TOOLING: brick-builder

- Export parsed data to CSV files
- Write a brick-builder template (below)

```
brick = https://brickschema.org/schema/1.1/Brick#
rdf = http://www.w3.org/1999/02/22-rdf-syntax-ns#
bldg = http://example.org/building#

bldg:$1 rdf:type brick:VAV
bldg:$1 brick:hasPoint bldg:$2
bldg:$2 rdf:type brick:Temperature_Sensor
bldg:$1 brick:hasPoint bldg:$3
bldg:$3 rdf:type brick:Temperature_Setpoint
$4? bldg:$1 rdf:type brick:RVAV
```
BRICK TOOLING: brick-builder

- Builds on OpenRefine tool, Reconciliation API
- Tutorial Video: https://www.youtube.com/watch?v=LKcXMvrxXzE
- NLP-based, other inference techniques under development
Working Group 1: Schema

- What most folks think of when we say “developing Brick”
- Responsible for stewardship of the Brick Schema
- Managing Brick ontology – new classes, new properties and relationships, data modeling approaches
- Extend Brick to new domains
- Create reference models
Working Group 2: Tooling

- More software development than modeling
- Tools to make it easier to create, manage, and exchange Brick models
- Extend/rewrite research prototypes into production tools
- Examples
  - Py-brickschema
  - Brick-builder (and next-gen version from CMU)
  - “Shepherding” tool from Buildsys 2020
Working Group 3: Applications

- Probably more analytics than applications at the beginning
- “Applying” Brick
- Coordinate with Schema WG – for a given use case, what’s missing from the model?
  - Similarly with tooling working group
- May also create “reference models”
Working Group 4: Datasets

- Interest in collecting anonymized datasets of buildings
- Important for research community and important for validating schema and applications
- Work closely with tooling – a serving platform can be a reference implementation
  - Also ingestion and model creation, with one extra anonymization step
Working Groups: Next Steps

- Two other groups likely on the horizon
  - Conformance
  - Data Exchange
- Groups will kick off in March
- Please add your name to interest survey if you haven’t already!
  - https://groups.google.com/g/brickschema/c/OOHSCEZwnc
The Brick Consortium Organization

- **Steering Committee**: Sets rules and membership fee, approves new “technical areas” for Brick to work on
  - Has a role in IP consideration
- **Technical Committee**: Responsible for creating Brick, but mostly through Working Groups. Decides when to send Schema out for a vote for release, 30 day clock for vote
- **Other Committees**: Will likely stay on-hold for the first few months
Brick and IP

- Brick Consortium takes IP and IP Disclosure very seriously. We do not want anyone to be surprised!

- Any member or participant who proposes enhancements to the Brick schema must disclose if they have relevant IP – unless they are willing to grant a “non-remunerative license” for that IP.
  - Detailed in section 5 of the membership rules – go by what that says!
  - Inspired by IETF but closes some loopholes

- Also required to disclose if you are aware of IP even if you don’t own it

- Steering Committee can approve the inclusion into the standard parts that would require a license

- Please review section 5 of the membership rules carefully
To Join Brick Consortium

- Choose appropriate membership level
- Complete Membership Form from website
- Submit 2021 dues

<table>
<thead>
<tr>
<th>Membership Level</th>
<th>Vote for Steering Committee</th>
<th>Chair any committee</th>
<th>Serve on any other committee</th>
<th>Vote for any other committee</th>
<th>2021 Membership Fee</th>
</tr>
</thead>
<tbody>
<tr>
<td>Full Member</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>$50,000</td>
</tr>
<tr>
<td>Institutional Member</td>
<td>No</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>$0</td>
</tr>
<tr>
<td>Academic Individual Member</td>
<td>No</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>$0</td>
</tr>
<tr>
<td>Contributing Member</td>
<td>No</td>
<td>No</td>
<td>Yes</td>
<td>Yes</td>
<td>$5000</td>
</tr>
</tbody>
</table>
Brick Public Roadmap Discussion
http://roadmap.brickschema.org/
Brick Roadmap Discussion

- If you are a potential adopter of Brick, what do you think is important to see in Brick?
- If you are a potential adopter of Brick, what's necessary for you to get Brick into your products?
- What are you willing to work on?
Thank You