`ref`: A Schema for Data Access Interoperability

Gabe Fierro
gtfierro@mines.edu
Colorado School of Mines

Erik Paulson
Johnson Controls, Incorporated

Joel Bender
Cornell University

ABSTRACT

Effective data management has become a growing concern amidst the increasing availability of data about the built environment and growing family of data-driven applications. Metadata models like Brick, Project Haystack, RealEstateCore and ASHRAE 223P are central to dealing with the fundamental challenges of the built environment: heterogeneous environments, complex dynamics and consistent churn. This new landscape brings questions: how exactly can metadata models support data-driven applications, and how can interoperability between such disparate efforts be maintained?

In this paper we develop a new metadata schema which cleanly *complements* and *extends* existing metadata solutions with unified access to external data sources and sinks. This exploits a key insight that a feasible path to interoperability lies not in the uniform representation of all building metadata, but rather providing uniform *access* to disparate sources of building metadata. We present the design and implementation of this new "reference" schema, differentiate it with respect to existing interoperability efforts, and explore the new abstractions and opportunities afforded by its design.

CCS CONCEPTS

• **Information systems** → **Ontologies**; **Data exchange**.

KEYWORDS

smart buildings, ontologies, metadata, schema, data exchange, interoperability

ACM Reference Format:
Gabe Fierro, Erik Paulson, and Joel Bender. 2018. `ref`: A Schema for Data Access Interoperability. In *Proceedings of Make sure to enter the correct conference title from your rights confirmation emai (Conference acronym 'XX)*. ACM, New York, NY, USA, 5 pages. https://doi.org/XXXXXXX.XXXXXXX

1 INTRODUCTION

Effective data management has become a growing concern amidst the increasing availability of data about the built environment and growing family of data-driven applications. Metadata models like Brick [3], Project Haystack [18], RealEstateCore [13] and ASHRAE 223P [2] are central to dealing with the fundamental challenges of the built environment: heterogeneous environments, complex dynamics and consistent churn. However, there still exists an "impedance mismatch" between the capabilities of these new metadata solutions and the requirements of existing analytics and control systems.

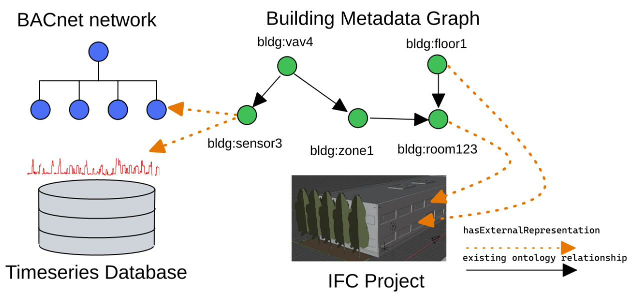

Figure 1: Entities can have external references to many different digital representations.

These metadata models are digital representations of the cyber-physical environment which make the structure, composition, topology and other properties of the environment available to other software. This assists in the deployment of energy efficiency, grid-interactivity and other intelligent applications [4]. Recent work has also demonstrated how standardizing these models also enables data-driven, portable software to be deployed at scale with minimal reconfiguration [10].

Despite these successes, *interoperability* between metadata solutions and the broader ecosystem of digital building representations is still a concern. In this paper, we present the design and implementation of a new RDF-based schema called `ref` which elegantly addresses interoperability with a simple, extensible model. The key insight is that attempting to define complete pairwise translations between different digital representations, or even unifying the representation of these representations (such as ifcOWL [17]), is intractable. Much of the complexity of these "mappings" can be abstracted away by instead (a) modeling the necessary *parameters* required to access external representations and (b) relying on existing software to retrieve the desired information (Figure 1). This approach is inspired by the classic wrapper-mediator architecture for data integration [6].

Our new RDF schema standardizes the definition of these parameters for several popular digital representations: Industry Foundation Classes, timeseries databases, and BACnet networks. We implement the proposed schema and demonstrate the portability of the schema by integrating it with three well-known ontologies: Brick [3], RealEstateCore [13] and SSN/SOSA [12]. The initial version of the schema is implemented and available online[1] under a permissive open-source license.

Conference acronym 'XX, June 03–05, 2018, Woodstock, NY
© 2018 Association for Computing Machinery.
ACM ISBN 978-1-4503-XXXX-X/18/06…$15.00
https://doi.org/XXXXXXX.XXXXXXX

[1]Anonymized for review

2 (META)DATA INTEROPERABILITY

Interoperability is the ability of systems to exchange and operate on information. In the context of data-driven software for smart buildings, it is vital for systems to exchange and understand not just telemetry, but also the *context* of that telemetry [4]. Recent research has focused on providing *semantic* interoperability in the form of graph-based metadata models [3, 13]; however as documented in prior work [4, 9], such efforts largely focus on the operational (i.e., telemetry-producing) phase of buildings. There is a substantial degree of useful metadata in other digital representations which is unavailable to these metadata models because it has different structure, syntax, or semantics.

In this paper, we focus specifically on the problem of how to provide *interoperable access* to data and metadata stored in external models and systems. We approach this through the design of a small, extensible schema (termed `ref`) which standardizes the parameters required to access other systems and retrieve information about a specific entity. This provides interoperability between the ecosystem of graph-based metadata models, and the large expanse of legacy and contemporary digital representatiions of buildings.

2.1 Related Work

Our solution takes inspiration from the wrapper-mediator architecture for data integration systems, as pioneered by the TSIMMIS project [6]. In this architecture, *wrappers* connect and provide access to heterogeneous data sources. They export information about the underlying schema, query capabilities and data to *mediators*, which transparently execute incoming queries over one or more data sources. The `ref` schema unifies the descriptions of how these external data sources are accessed. Rather than having one description for each data source, `ref` defines a description for *individual* entities in a metadata graph. This description also contains the *indexing* information required to access the entity's representation in that external data source.

Other work defines hardware abstraction layers which hide the protocols, encodings and formats of underlying data sources. sMAP [7], BuildingDepot [19], Google Digital Buildings [5] and Sparkplug [8] all define APIs or payload formats which standardize how data is represented. Our approach complements these efforts. While a standard API for data access is a convenient and largely necessary feature for large-scale IoT deployments, such an API will likely not cover building information modeling or other non-telemetry formats. A related approach detailed in [9] uses wrappers to import relevant data from external data representations into a Brick model.

Some ontologies represent foreign data directly in an RDF graph. While this approach unifies access to this data, representing that data in RDF obscures data structures in the original digital representation, and can impede efficient access. Ontologies such as SSN/SOSA [12] and [14] represent timeseries data directly in the RDF graph. However, without some virtualization of this telemetry over some specialized data store, common data analytics tasks which access large spans of data cannot operate efficiently. ifcOWL [17] defines an ontology which incorporates most of the IFC 2x3 standard. However, this makes it difficult to consult the rich geometric descriptions characteristic of IFC models. Our solution

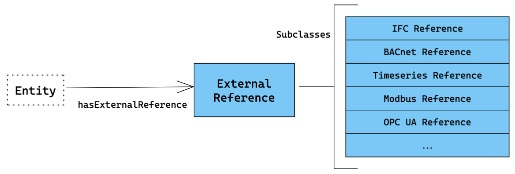

Figure 2: The structure of the proposed schema. The type of `Entity` node is deliberately unspecified.

preserves these digital representations so they can be accessed in an efficient and familiar manner.

A final category of related work are ontologies that obviate the need to rely on external client implementations to access the information in other digital representations. Both RealEstateCore [13] and W3C Web of Things [11] use RDF to describe how to parse and navigate foreign data structures in order to extract the necessary information. We believe that our approach strikes a more reasonable tradeoff that avoids the complexity of having to produce and maintain *both* generic descriptions of arbitrary structures and client software that parses and traverses those descriptions.

3 SCHEMA DESIGN

A client requires three pieces of metadata to access an entity in an arbitrary digital representation: what kind of digital representation is being accessed, how to connect to or load the digital representation, and how to traverse the model to find the desired entity. Our proposed schema fulfills these requirements with two primary components: a *generic* external reference relationship, and a family of *specific* external reference definitions (Figure 2).

3.1 Semantic Web Background

The `ref` schema is built on the RDF and SHACL semantic web standards. RDF [15] is a W3C standard for directed, labeled graphs which describe resources, their properties, and relationships to other resources. This provides immediate syntactic interoperability with RDF-based building metadata models like Brick and RealEstateCore. Figure 3 contains a textual serialization of an RDF graph. SHACL [1] is a recent standard for specifying validation constraints over RDF graphs; the required and optional metadata for each domain (Table 1) are captured using SHACL.

3.2 Schema Components

We define a generic relationship — `ref:hasExternalReference` — that associates a subject entity with a collection of metadata describing its representation in an external digital representation. The presence of this relationship indicates to consumers of a metadata model that an entity has external metadata. Our schema definition purposefully avoids prescribing what kinds of entities can have an external reference, meaning that *any* existing ontology can incorporate the schema. It is prudent, but not necessary, for each host ontology to constrain what kinds of entities can have what kinds of external references. For example, in the Brick ontology, only `Point` instances can have BACnet and timeseries references, but any kind of entity can have an IFC reference.

```
1  @prefix brick: <https://brickschema.org/schema/Brick#> .
2  @prefix ref: <anonymized-for-review> .
3  @prefix bacnet: <anonymized-for-review> .
4  @prefix tsdb: <anonymized-for-review> .
5  @prefix : <urn:example#> .
6
7  :xyz a brick:Zone_Air_Temperature_Sensor ;
8    ref:hasExternalReference [
9      a ref:TimeseriesReference ;
10     tsdb:hasTimeseriesId "4665117e-ec75-47c2-a5ce-b71529cb159e" ;
11     tsdb:storedAt "postgresql://dataserver/sensordatadb" ;
12   ] ;
13   ref:hasExternalReference [
14     a ref:BACnetReference ;
15     bacnet:object-identifier "analog-value,5" ;
16     bacnet:object-name "BLDG-Z410-ZATS" ;
17     bacnet:objectOf :sample-device ;
18   ] .
19 :sample-device a bacnet:BACnetDevice ;
20     bacnet:device-instance 123 ;
21     bacnet:hasObject [ a bacnet:NetworkPortObject ;
22       bacnet:ip-address "01020304"^^xsd:hexBinary ;
23       bacnet:ip-udp-port 47808 ;
24     ] .
```

Figure 3: An example graph using the ref schema to define a BACnet and timeseries reference for a zone air temperature sensor instance defined in Brick.

ref:hasExternalReference must point to an instance of the ref:ExternalReference class. ref:ExternalReference is an abstract class with subclasses corresponding to each of the domains in which an entity may be represented. Subclasses of ref:ExternalReference are modeled as SHACL shapes which define a set of required and optional properties. These properties provide the metadata required to *connect to/load* an external representation and *find* the desired entity. Table 1 lists each of the domains included in the initial release of the schema and their corresponding properties.

3.3 Example Usage

Consider the usage of the ref schema illustrated in Figure 3. The graph defines a single building entity, a zone air temperature sensor named :xyz, which corresponds to a BACnet object and has historical telemetry saved in a Postgres database. Lines 8-11 define the link to the telemetry data: a client accesses the tsdb:storedAt property to configure the connection to the database, and uses the value of the tsdb:hasTimeseriesId property to determine which rows correspond to data produced by the :xyz sensor. Lines 13-17 define the BACnet object which the :xyz entity represents. Access to the BACnet object allows clients to get the present value of an I/O point. To access the BACnet object, a client needs both the instance number of the object (bacnet:object-identifier) and the BACnet device which hosts that object. Lines 19-23 define the BACnet device; this instance can be re-used across multiple ref:BACnetReference objects.

3.4 Schema Extensibility

The design of the ref schema emphasizes a *separation of concerns*. Additional domains can be added to the schema in a *backwards compatible manner* without affecting other definitions. Properties, classes and concepts required to properly model a domain are placed in a domain-specific namespace to avoid conflicts with similarly-named items with different meanings for other domains. In future

```
1  import psycopg2, rdflib
2  import pandas as pd
3  graph = rdflib.Graph()
4  graph.parse("my-building.ttl") # load in the Brick model for a building
5  # get the access parameters (assuming Postgres)
6  sparql_query = """SELECT * WHERE {
7     ?sen a rec-device:DryBulbTemperatureSensor ;
8          rec-core:observes ?zone .
9     ?zone a rec-core:HVAC_Zone .
10    ?sen ref:hasExternalReference [
11       a ref:TimeseriesReference ; tsdb:hasTimeseriesId ?uuid ;
12       tsdb:storedAt ?conn ; tsdb:timeColumn ?tcol ;
13       tsdb:valueColumn ?vcol ; tsdb:idColumn ?icol ]}"""
14 sen, zone, uuid, conn, tcol, vcol, icol = next(graph.query(sparql_query))
15 with psycopg2.connect(conn) as sql: # download the data
16   sql_query = f"""SELECT {tcol}, {vcol}, {icol} FROM data
17                   WHERE {icol} = ? AND {tcol} > "2021-01-01";"""
18   df = pd.read_sql(sql_query, sql, params=(uuid,))
19   # df is now a DataFrame containing data since Jan 1st 2021
20   print(f"Have {len(df)} rows of temperature data for zone {zone}")
```

Figure 4: Illustrating use of the ref schema in conjunction with RealEstateCore to access sensor data. A SPARQL query extracts the relevant metadata for accessing a SQL database. Only a handful of lines of Python code are required to use this metadata to retrieve the actual telemetry.

releases of the ref schema we plan to add support for other device-facing protocols such as MQTT, Modbus and OPC-UA, databases such as InfluxDB and OSIsoft, and digital building representations like gbXML and Revit.

4 EVALUATION

To evaluate the ref schema, we provide evidence of its interoperability across several existing ontologies by extending those ontologies to work with our schema using only a few RDF statements. Further, to demonstrate the utility of our approach, we illustrate how only a few lines of Python code are needed to leverage the ref annotations in Figure 3 to access historical telemetry for a sensor.

4.1 Cross-Ontology Interoperability

The ref schema promotes interoperability between disparate metadata models, ontologies, and other digital representations by more clearly defining the *boundaries* between them. Rather than the complexity of modeling foreign concepts across different metadata efforts, the ref schema instead allows any RDF-based metadata model to *refer* to external metadata in a consistent manner.

The design of ref is trivially portable to other RDF-based metadata models, including Brick and RealEstateCore; examples of this integration are provided in Figures 3 and 4, repsectively.

4.2 Illustration of Use

We demonstrate that the ref schema is *usable* by implementing a generic script for accessing telemetry data in only a few lines of Python. Figure 4 contains a representative code sample illustrating this approach. To simplify the example we assume that the data is stored in a Postgres database; this can easily be abstracted away by using a database-agnostic access library such as SQLAlchemy[2].

The first few lines of the program import required libraries and initialize an in-memory Brick model representing a building. Lines

[2]https://www.sqlalchemy.org/

Domain	Property	Description
IFC [16]	`ifc:hasProjectReference`	the `Project` object which defines the IFC model containing this entity (required)
	`ifc:globalID`	the global entity ID which uniquely identifies this entity in the IFC model (required)
	`ifc:name`	the label associated with the IFC entity (optional)
BACnet	`bacnet:objectOf`	the `BACnet Device` that hosts the given BACnet object (required)
	`bacnet:object-identifier`	the BACnet object identifier (required)
	`bacnet:object-type`	the type of the BACnet object (optional)
	`bacnet:object-name`	the value of the BACnet object `Name` field (optional)
	`bacnet:read-property`	the name of the BACnet property to read; defaults to `present-value` (optional)
Timeseries	`tsdb:storedAt`	the connection string defining the networked location of the database containing the telemetry associated with this entity (required)
	`tsdb:timeseriesID`	the foreign key identifying the entity's stream of data in the database (required)
	`tsdb:tableName`	the name of the table containing the data (optional)

Table 1: The initial set of domains covered by the `ref` schema and most of the required and optional properties.

10-20 define and execute a query against this model that extracts the name of each zone air temperature sensor in the building, its corresponding zone, and all of the metadata required to find the corresponding data in a database. For completeness, we have encoded in the model the column names to be used in the query. These column names, in addition to the database connection string and the identifier of the timeseries stream, are incorporated into a connection to the database (line 23) and the composition of a SQL query (line 24-25) which is finally executed to download data for a user-provided time span (line 26).

Note that much of the complexity of interacting with the external data source is handed off to an existing client library. Similar patterns emerge for the other domains modeled in the `ref` schema — BAcpypes[3] facilitates access to a BACnet network from Python and IfcOpenShell[4] enables Python clients to extract information from an IFC model. Use of the `ref` schema is language agnostic: the metadata can be discovered and retrieved using standard open-source and commercially available tools.

5 CONCLUSION

This paper has presented the design and implementation of the `ref` schema, a simple and extensible metadata model that normalizes references between operational metadata models for buildings (e.g., Brick, RealEstateCore and Project Haystack) and other digital representations. This different approach to interoperability provides a path to the further development of emerging "digital twins," whose configuration and operation is often limited by the ability to combine multiple data streams and digital representations. We propose the `ref` schema as a "glue" that helps digital twin and other data-driven platforms find all of the data they need, using existing and rich metadata standards. The `ref` schema is permissively licensed, open-source and available online at *anonymized for review*.

REFERENCES

[1] 2017. *Shapes constraint language (SHACL).* Technical Report. W3C.

[2] American Society of Heating, Refrigerating and Air-Conditioning Engineers. 2018. ASHRAE's BACnet Committee, Project Haystack and Brick Schema Collaborating to Provide Unified Data Semantic Modeling Solution. http://web.archive.org/web/20181223045430/https://www.ashrae.org/about/news/2018/ashrae-s-bacnet-committee-project-haystack-and-brick-schema-collaborating-to-provide-unified-data-semantic-modeling-solution.

[3] Bharathan Balaji, Arka Bhattacharya, Gabriel Fierro, Jingkun Gao, Joshua Gluck, Dezhi Hong, Aslak Johansen, Jason Koh, Joern Ploennigs, Yuvraj Agarwal, Mario Berges, David Culler, Rajesh Gupta, Mikkel Baun Kjærgaard, Mani Srivastava, and Kamin Whitehouse. 2016. Brick: Towards a Unified Metadata Schema For Buildings. In *Proceedings of the 3rd ACM International Conference on Systems for Energy-Efficient Built Environments* (Palo Alto, CA, USA) *(BuildSys '16)*. Association for Computing Machinery, New York, NY, USA, 41–50.

[4] Harry Bergmann, Cory Mosiman, Avijit Saha, Selam Haile, William Livingood, Steve Bushby, Gabe Fierro, Joel Bender, Michael Poplawski, Jessica Granderson, and Others. 2020. *Semantic Interoperability to Enable Smart, Grid-Interactive Efficient Buildings.* Technical Report. Lawrence Berkeley National Lab.(LBNL), Berkeley, CA (United States).

[5] Keith Berkoben, Charbel El Kaed, and Trevor Sodorff. 2020. A Digital Buildings Ontology for Google's Real Estate. In *ISWC (Demos/Industry)*. ceur-ws.org, 392–394.

[6] S Chawathe, H Garcia-Molina, J Hammer, K Ireland, Y Papakonstantinou, J Ullman, and J Widom. 1994. The TSIMMIS Project: Integration of Heterogenous Information Sources.

[7] Stephen Dawson-Haggerty, Xiaofan Jiang, Gilman Tolle, Jorge Ortiz, and David Culler. 2010. sMAP: a simple measurement and actuation profile for physical information. In *Proceedings of the 8th ACM Conference on Embedded Networked Sensor Systems* (Zürich, Switzerland) *(SenSys '10)*. Association for Computing Machinery, New York, NY, USA, 197–210.

[8] Eclipse Foundation. 2019. *Sparkplug MQTT Topic & Payload Specification.* Technical Report.

[9] Gabe Fierro, Anand Krishnan Prakash, Cory Mosiman, Marco Pritoni, Paul Raftery, Michael Wetter, and David E Culler. 2020. Shepherding Metadata Through the Building Lifecycle. In *Proceedings of the 7th ACM International Conference on Systems for Energy-Efficient Buildings, Cities, and Transportation* (Virtual Event, Japan) *(BuildSys '20)*. Association for Computing Machinery, New York, NY, USA, 70–79.

[10] Gabe Fierro, Marco Pritoni, Moustafa Abdelbaky, Daniel Lengyel, John Leyden, Anand Prakash, Pranav Gupta, Paul Raftery, Therese Peffer, Greg Thomson, and David E Culler. 2019. Mortar: An Open Testbed for Portable Building Analytics. *ACM Trans. Sen. Netw.* 16, 1 (Dec. 2019), 1–31.

[11] Amelie Gyrard, Pankesh Patel, Soumya Kanti Datta, and Muhammad Intizar Ali. 2017. Semantic web meets internet of things and web of things. In *Proceedings of the 26th International Conference on World Wide Web Companion - WWW '17 Companion* (Perth, Australia). ACM Press, New York, New York, USA.

[12] Armin Haller, Krzysztof Janowicz, Simon J D Cox, Maxime Lefrançois, Kerry Taylor, Danh Le Phuoc, Joshua Lieberman, Raúl García-Castro, Rob Atkinson, and Claus Stadler. 2019. The modular SSN ontology: A joint W3C and OGC standard specifying the semantics of sensors, observations, sampling, and actuation. *Semantic Web* 10, 1 (2019), 9–32.

[13] Karl Hammar, Erik Oskar Wallin, Per Karlberg, and David Hälleberg. 2019. The RealEstateCore Ontology. In *The Semantic Web – ISWC 2019*. Springer International Publishing, 130–145.

[14] Cory Andrew Henson, Holger Neuhaus, Amit P Sheth, Krishnaprasad Thirunarayan, and Rajkumar Buyya. 2009. An Ontological Representation of Time Series Observations on the Semantic Sensor Web. (2009).

[15] Ora Lassila, Ralph R Swick, World Wide, and Web Consortium. 1998. Resource Description Framework (RDF) Model and Syntax Specification. (1998).

[16] Thomas Liebich. 2013. IFC4—The new buildingSMART standard. In *IC Meeting*.

[17] Pieter Pauwels and Walter Terkaj. 2016. EXPRESS to OWL for construction industry: Towards a recommendable and usable ifcOWL ontology. *Autom. Constr.* 63 (March 2016), 100–133.

[3]https://bacpypes.readthedocs.io/en/latest/

[4]http://ifcopenshell.org/

[18] ProjectHaystackCorporation. 2022. Project Haystack. https://project-haystack.org.

[19] Thomas Weng, Anthony Nwokafor, and Yuvraj Agarwal. 2013. BuildingDepot 2.0: An Integrated Management System for Building Analysis and Control. In *Proceedings of the 5th ACM Workshop on Embedded Systems For Energy-Efficient Buildings* (Roma, Italy) *(BuildSys'13)*. Association for Computing Machinery, New York, NY, USA, 1–8.