Guide Specification for Data Modeling of Building Systems and Equipment Based on Project Haystack Open Source Data Modeling Standard

v2013-10-1

1.0 Purpose: The purpose of a data modeling standard is to provide a consistent, standardized methodology for naming and describing data points associated with facility automation systems, equipment systems, energy metering systems, other smart devices including mobile assets, and associated descriptive information known as metadata.

1.1 Background: Modern automation systems and smart devices have made it easy to collect vast quantities of data including environmental conditions, equipment operational status, and energy usage and performance. The reality, however, is that these data typically exist in a low-level ad-hoc format without standard or consistent organization, making it difficult to interpret trends, perform analysis or generate useful reports and visualizations without significant manual effort. The result is that we are now awash in large volumes of unstructured data, we can't easily derive value from.

The first step to turning smart device data into actionable intelligence is to give the data "context" so that we know exactly what each piece of data "means" and how fits into an overall system. For example, to compare the discharge air temperature of an Air Handling Unit (AHU) against the return air temperature, how can we find this information? Today, often the only indication of what a data point means is found by deciphering an arbitrary name the system integrator gave it during configuration such as "DA_TEMP".

Given that there has previously not been any accepted standard for data point naming conventions associated with building systems and facility engineering, one key challenge the industry faces is to establish a common vocabulary to give meaning to information collected by the various building systems. Looking at the challenge from the perspective of complex systems, it becomes apparent that it is not possible to capture the full extent of descriptive information desired simply using data point names, even if standardized. For example, to model a complete air distribution system to determine which AHUs feed which Variable Air Volume (VAV)s; or to know all the relationships between sub-meters and equipment loads in an electrical system. Modeling these relationships enables analysis of operations at the systems level, building level, or even across an entire portfolio of buildings. Point names that attempted to capture this range of information would be impossibly long, therefore structured combinations are needed.

Project Haystack's mission is to define a methodology and common vocabulary so that models of building systems and smart devices can be interpreted automatically by a variety of software and web-based applications. This will enable owners, operators, manufacturers and service providers to more efficiently derive value from the vast amounts of data smart systems are collecting.

Project Haystack encompasses the entire value chain of building systems and related intelligent devices and enable owners and consultants to specify naming Haystack conventions for data contained in building automation and similar systems to ensure

standards-based organization of system data and easy integration with external applications. Project-Haystack facilitates “mapping” of Haystack semantic tagging with other relevant standards.

2.0 Technical Overview

2.1 Overall Design Concepts: The Project Haystack data modeling standard for Buildings and Equipment systems shall use a simple metamodel based on the broadly accepted concept of “tags” as described below.

Tags: Tags are name/value pairs, associated with entities like AHUs, electric meters, etc. Tags are simple and dynamic, add structure, and provide the flexibility needed to establish standardized models of diverse systems and equipment. Tags are a modeling technique that allows easy customization of data models on a per-task, per-project or per-equipment basis, while retaining the ability to be interpreted by external applications using a standard, defined methodology and vocabulary. Tags shall support the definition of the following essential data elements:

Entity: An Entity is an abstraction for a physical object in the real world. Entities include sites, facilities, equipment, sensor points, weather stations, etc. In software systems, an entity might be a modeled as a record in a database, an object in a building automation system, or maybe just a row in a csv file or spreadsheet.

Id: The id tag is used to model the unique identifier of an entity in a system using a Ref value type. Ref value types are determined by individual application. The scope of an entity may be undefined, but must be unique within a given system or project. This identifier may be used by other entities to cross-reference entities, associations, and systems.

Dis: The dis tag is used with entities to define display text used to describe an entity. Dis values are intended to be short (less than 30 or 40 characters), but fully descriptive of the entity for a human user.

Tag Kinds. The standard shall provide the following permitted tag value types:

- Marker: this tag type is merely a marker annotation with no meaningful value. Marker tags are used to indicate a "type" or "is-a" relationship.
- Bool: boolean "true" or "false".
- Number: integer or floating point number annotated with a Unit of Measurement, where ideally, units of measure are prescribed for various tasks.
- Str: a string of Unicode characters.
- Uri: a Unversal Resource Identifier.
- Ref: reference to another entity. The Project Haystack specification does not currently prescribe specific identities or reference mechanisms, but should be used to cross link entities. Refs are formatted with a leading "@" and require a

specific subset of ASCII characters be used: a-z, A-Z, 0-9, underbar, colon, dash, or dot.

- Bin: a binary blob with a MIME type formatted as Bin(text/plain)
- Date: an ISO 8601 date as year, month, day: 2011-06-07.
- Time: an ISO 8601 time as hour, minute, seconds: 09:51:27.354.
- DateTime: an ISO 8601 timestamp followed by timezone name:
  2011-06-07T09:51:27-04:00 New_York
  2012-09-29T14:56:18.277Z UTC

Example of Usage. The following presents an example for an entity describing a site:

id: @whitehouse
dis: "White House"
site
area: 55,000 ft²
geoAddr: "1600 Pennsylvania Avenue NW, Washington, DC"
tz: "New_York"
weatherRef: @weather.washington

The example presents an entity with seven tags: id, site, dis, area, geoAddr, tz, and weatherRef. By convention, when writing examples each tag will be listed on their own line or separated by a comma. The site tag has no explicit value, it is assumed to be marker tag. The dis, geoAddr, and tz tags have string values, many which may be standardized by others, typically indicated by double quotes. The area tag has a number value indicated by a scalar with unit of square feet or square meters. The weatherRef tag is a reference to another entity, typically indicated using the "@" character.

2.2 Standard Library of Tags and Library Extensibility: The Project Haystack data modeling standard shall provide a comprehensive library of standard tags to address common equipment, building systems, and devices types. The standards development community shall engage in an open discussion forum to enable industry experts and interested parties to discuss, submit, fine-tune and eventually approve additional tags or standard schemas to address equipment, systems, and applications of numerous types. The open forum process shall be transparent to enable continued development of a taxonomy that will enable semantic understanding of facilities engineering data across and outside of the industry.

3.0 REST API

The Project Haystack data modeling standard shall provide a documented Representational State Transfer, Application Programming Interface (REST API) to define a simple mechanism to exchange “tagged” data over web services.

REST servers are programmed to implement a set of ops or operations. An operation is a uri that receives a request and returns a response. Standard operations are defined to query databases, setup subscriptions, or read/write histories of time-series data. Operations are pluggable so vendors can enhance open REST interfaces with customized, value-added functionality for their own business purposes.

Both requests and responses are modeled as grids. Grids are encoded using standard Multipurpose Internet Mail Extensions (MIME) types for grid serialization, may be pluggable using HTTP content negotiation, and other standardized web service protocols.

The Project Haystack REST API utilizing the "ops" design is more akin to a Remote Procedure Call (RPC) model, but the term REST is used to distinguish the design from traditional WS-* web services that use Extensible Markup Language (XML), Simple Object Access Protocol (SOAP), and other Internet standards although the current design could easily tunnel through those technologies

4.0 Applications

The goal of the Project Haystack data modeling standard is to ensure consistent modeling of building systems, devices and associated data. The following application requirements outline the use of the modeling standard in applications related to buildings, energy, and facility management.

4.1 Minimum Model Requirements: The Haystack Project implementation shall utilize defined data modeling tags to create an expanding, and coherent model with the following minimum items, hierarchy and relationships when used in facilities-oriented applications:

Sites: Including display name, description, size (area) as a minimum. References to Internet-available weather stations are highly recommended, as are creating tags to represent other relevant characteristics of a Site such as year constructed, facility usage type, occupancy class, schedule(s) of operation, building systems type (e.g., packaged or central HVAC).

Equipment: Including standardized associations with sites via id reference and display name as a minimum. Equipment and software vendors, model numbers, year of installation, and similar descriptive meta data are also recommended.

Points: Including standardized associations with sites and equipment via id reference, units of measure as a minimum. Where possible, ranges of acceptable values are recommended.

4.2 Exposing the Project Haystack Model via REST API: Software and web service applications, including control system devices will expose the model definitions described above using the Project Haystack REST API published as part of the Project Haystack standard, openly accessible and kept up to date at http://project-haystack.org/doc/Rest

4.3 Software Reference Implementations: The Project Haystack standard shall provide a reference implementation in Java, providing sample code for implementation of the Haystack REST protocol in software applications.

4.4 Open Source Modules for Commercially Available Products. The Project Haystack Community has developed, and makes available, a comprehensive implementation of the

Haystack protocol in the form of a software module for use with Niagara AX-based systems. The module, known as NHaystack, is licensed under the Academic Free License ("AFL") v. 3.0. Public access to the NHaystack software module shall be maintained via the project-haystack.org site.

When Niagara AX-based systems are used, the NHaystack module shall be the preferred method of communication between the Niagara AX-based devices and other software applications that are consuming Niagara data or writing commands back to Niagara AX-based systems.

5.0 Open Source

The Project Haystack Facilities Engineering Data Modeling Standard for Smart Device Data modeling methodology, standards, supporting documentation and reference implementations shall be available via an open source license at no cost.

5.1 The open source license shall use the Open Source Initiative Academic Free License 3.0 model. Full details on the terms of the license are available at: http://project-haystack.org/doc/License and http://opensource.org/licenses/AFL-3.0.

END OF DOCUMENT