Haystack 4 Is Coming – What it Is and Why it Matters

Haystack 4 – A Brief Introduction

The understanding of the need for semantic modeling of device and equipment data has matured significantly in the last decade and the requirements and techniques for applying semantic modeling to equipment data are advancing rapidly. As we have learned, semantic modeling is critical for humans to work with and understand the ever-increasing amount of data coming from their systems, but the process of manually applying that semantic model is not scalable. We need our tools to simplify and automate how the semantic model is applied.

Haystack 4 builds on the 8+ years of experience in applying Haystack across thousands of buildings worldwide, the input from practitioners in the community throughout that time, as well the collaborators that have participated in the activities of Haystack Working Group 551 over the past year. The way you configure tags today using Haystack will not change, but as you will see, the way the tags get modeled within Haystack systems will enable the tools you use to become smarter so you spend less time manually configuring tags and more time getting value out of the raw data coming from your IoT devices.

Extensive documentation will be available on this work and it will be a key focus of the Haystack Connect 2019 event, with a full day track dedicated to a detailed review of Haystack 4. Our goal with this document is to offer a high-level preview of the major advances that Haystack 4 brings to the challenge of semantic modelling of the devices and equipment systems that permeate the built environment.

Haystack 1.0 pioneered the concept of applying semantic modeling to equipment and devices data. It employed a simple approach of applying tags to items to define what they "meant". Tags described things like units of measure, as well as facts and characteristics about data. For example, the tags:

```
discharge, air, temp, sensor, point, unit:"°F"
```

tells us that a number represents a numerical value of discharge air temperature expressed in degrees F produced by a sensor. Depending on the system, this number could be named AO_ 21, nvoTemp, or Register_43015. Without the simple tags mentioned above you couldn't do much with the sensor data.

Haystack 1 therefore provided us with a standardized vocabulary to markup "things" and the data they produced. Starting in 2011 that provided the industry with its first widely adopted solution for standardized, open, data modeling for device and equipment data, which allowed us to agree on the terms to use to help define what things are. In the world of semantics that's called a vocabulary.

Haystack 2 introduced a REST API in 2013 to provide a standard way to query a system that applied the Haystack semantic model to its data. As the demand for open protocols and open systems in the built environment continued to increase, offering an open API was important to ensure customers had a standard way to easily access the data in their systems.

Haystack 3, released in 2016, added several new data types to help machines better understand and process the different types of data formats in the IoT. The importance of data types for machines can be thought about using a simple example. Imagine getting added to an email chain, where some of the older messages were in a language you didn't understand. You could copy and paste the text into an online translation tool and may be able to figure out what was said, but not as quickly or as easily as if you natively spoke that language. Every machine has a concept of a String, but what is stored in that String may or may not make sense without extra processing. Having to account for these different scenarios adds complexity to systems that can cause implementation problems as networks grow so having standard data types reduces the number of scenarios our systems need to support.

With Haystack 4 we undertake to address the next levels of sophistication in semantic modeling – developing a taxonomy and an ontology to support the ability to represent machine-readable relationships of things and their data.

By Taxonomy we refer to a way of defining the relationships **of** things. For example, we say that water is a subtype of liquid because it is a specific type of liquid. The converse is that liquid is a supertype of water. Haystack 4.0 utilizes the concept of subtypes to organize all terms into a tree-based taxonomy. This provides us with defined and agreed upon relationships of things. We will touch on the concept of "types" more in a moment.

By Ontology we refer to the ways a semantic model captures relationships **between** things, such as which AHU feeds air to a VAV. We need a structured taxonomy to achieve the benefits of a rich ontology of devices and equipment systems. A powerful use case for analyzing data from the IoT is tracking the flow of energy across systems. The energy could be used to convey heated or cooled gas through a duct or liquid through a pipe, but without a standard way of representing the flow of energy, or any relationships between things, we can't drive the industry forward by making our tools more capable of automatically analyzing these relationships. Haystack 4.0 extends the standard to support the implementation of both a taxonomy and the resulting ontologies.

What happens to my pre-Haystack 4.0 systems?

It is worth repeating, the way you configure tags today does not change. Also, the tags you are using today are not changing. The important difference, and significant benefit, is how a Haystack system models those tags internally.

Today you may have an AHU with at least the following tags:

```
ahu, equip, hvac, siteRef:…
```

That AHU likely has many points including a Discharge Temperature Sensor with at least the following tags:

```
discharge, air, temp, sensor, point, unit:"°F", equipRef:…, siteRef:…
```

The system you used to configure those tags probably didn't help by automatically adding a a discharge temp sensor point (as well as several other points) after you added a piece of equipment with the ahu tag. You either manually created a template for yourself or if you were lucky, the system you used had a proprietary template that helped. The terms we use to represent tags are not changing in Haystack 4.0

are not changing, but the way the tags are internally represented gives us a standard way to identify required relationships. Keep reading for an overview of how the new def system makes this possible.

What and Why?

This next generation of Haystack moves us closer to transforming device data into knowledge along with these important benefits:

- Enabling us to infer relationships between items, and apply more powerful filters and queries
- Enabling development of more advanced tools for automating semantic tagging
- Validation of tagging through certification testing
- New standard model to describe the spaces, floors, rooms, zones, etc. within a building

Defining Types of Equipment and Other Things. A key feature of Haystack 4 that enables more comprehensive modeling of taxonomies and ontologies is a simple, flexible and elegant approach to defining types of equipment and devices. Prior to Haystack 4.0, there was a flat list of terms that represented tags. These terms were used the same way as a hashtag you see on any social media, such as #iot or #data, as a way to quickly find content related to those topics. The challenge, which you probably have noticed, is that the number of tags seems to continue to grow as sub-topics of a main topic become popular. Now you see a message with like 30+ hashtags at the bottom to try and guarantee the most people will find the message. As more hashtags get added, less relevant content comes up in searches.

The new def model in Haystack 4.0 allows a term to be represented with significantly more context rather than just a term from a flat list. That additional context simplifies the number of tags needed, but more importantly allows a system to automatically determine relationships. Prior to Haystack 4.0, the only guaranteed way for you to know a relationship between two tags was to go and read the tag list on https://project-haystack.org. Now a Haystack 4.0 system can use the metadata included in a def for the concept, generally referred to as “Subtyping”, to automatically organize virtually any “thing” or entity described with Haystack terms into a taxonomy tree. Here’s a simple example **def**ining that water **is** a type of liquid:

```
def: ^water
doc:Water in its liquid form
is:[^liquid]
```

Now, when you are modeling a water system, even if only the water tag is applied, an analytical algorithm that generally applies to all forms of liquid can be used because a Haystack 4.0 system can infer from the water def that it is also liquid.

Now let’s examine a few parts of the new ahu **def** below (see the full def on the Project Haystack website):

```
def:^ahu
coolingProcess:^coolingProcessType
cools:^air
dehumidifies:^air
doc:Air handling unit – mixes outside air and return air
ductConfig:^ductConfigType
heatingProcess:^heatingProcessType
heats:^air
humidifies:^air
is:[^airHandlingEquip]
ventilates:^air
wikipedia:`https://en.wikipedia.org/wiki/Air_handler`
---
def:^coolingProcess
doc:Processed used to cool a substance
is:[^choice]
of:^coolingProcessType
---
def:^airCooling
doc:Cooling by dissipating heat into the surrounding air
is:[^coolingProcessType]
wikipedia:`https://en.wikipedia.org/wiki/Air_cooling`
```

Now instead of simply a term ahu (and the operator needing to know to add several more pieces of metadata), a Haystack 4.0 system can automatically infer that something using the ahu def, needs a cooling process. You will be glad to know there is a coolingProcess def which **is** a subtype of the choice def and value type of coolingProcessType. By simply looking for everything that **is** subtype of coolingProcessType you can quickly find the choices are airCooling, chilledWaterCooling, condenserCooling, and dxCooling (the airCooling def is shown above). You will notice an ahu also needs a duct configuration (use the same workflow to determine your choices are singleDuct, dualDuct, and tripleDuct), heating process, etc. Additionally, since a roof top unit is already defined via the rtu def as a subtype of the ahu def you don't have to worry about whether or not all your equipment and points have both the ahu and rtu tags, since Haystack 4.0 systems will take care of that for you.

To further explain our example earlier where Haystack 4.0 systems can now automatically infer that an ahu needs a discharge air temperature sensor, let's trace the discharge-duct def. Notice that the discharge-duct def is contained by airHandlingEquip and an ahu is a subtype airHandlingEquip (as you can see above). Therefore, we know that an ahu contains a discharge-duct. Because a discharge-duct conveys air, and we know air is a subtype of substance (air is a gas, which is a fluid, which is a substance) as defined by the Haystack 4.0 taxonomy, Temperature is a quantity of a substance, as you can see in the temp def so we can infer that anything with a discharge duct, which conveys air, needs a sensor to measure the airs temperature. Hopefully this helps you start to understand the power that Haystack 4.0 will bring to the IoT.

```
def:^discharge-duct
containedBy:[^airHandlingEquip,^airTerminalUnit]
conveys:^air
doc:Supply air discharged from equipment
ductDeck:^ductDeckType
ductSection:^discharge
is:[^duct]
wikipedia:`https://en.wikipedia.org/wiki/Duct_(flow)`
---
def:^air
doc:The mixture of gases which surrounds the earth
is:[^gas]
wikipedia:`https://en.wikipedia.org/wiki/Atmosphere_of_Earth`
---
def:^temp
doc:Temperature - measure of hot and cold
is:[^quantity]
prefUnit:["°C","°F"]
quantityOf:^substance
```

The Subtyping concept is a powerful knowledge modeling tool that helps us apply semantics to devices and equipment systems encountered in the real world. While it may initially appear complicated, please remember that ultimately humans are not supposed to be manually traversing the Taxonomy and Ontologies defined in Haystack 4.0. Systems will be doing the heavy lifting automatically allowing us to focus on the things humans are better at. One of the most important benefits is that it adds important new capabilities to define **relationships** that model how spaces, equip, points, and processes are related to each other. Almost all Haystack implementations will model the containment of physical spaces and equipment. It is also typical to model the flows of energy and substances such as electricity, air, and water. Subtyping and Relationship modeling provide the ability to implement advanced “filters” for querying relationships among a items.

Where are these things in my building?

Something people have been asking for within the Haystack Community is a model for spaces, floors, rooms, zones, etc and Haystack 4.0 makes that a reality. The physical location of equipment is beneficial when automatically generating support tickets for maintenance based on the results of automated analytics. This standard model will also be a powerful data point when considering comfort within buildings. Now you can quickly determine which equipment impacts the largest amount of space in a building or identify specific spaces, such as a data center, which should be exempt from automated demand response actions. As you are reading this, I am sure you can think of additional ways this new space model will bring value to the Haystack Community.

Take a look at a example of how the new model works below. We start with a space def, which has several subtypes, including floor, room, zone-space and others. The area def has been added as a tag on the space def so each subtype of space inherits the area tag. The full space def model can be reviewed here: https://project-haystack.dev/doc/lib-phIoT/space

```
def:^space
containedBy:[^site,^space]
doc:Space is a three-dimensional volume in the built environment
is:[^entity]
---
def:^area
doc:Area of a shape or floor space
is:[^number]
prefUnit:["ft²","m²"]
tagOn:^space
---
def:^room
containedBy:[^site,^space]
doc:Enclosed room of a building
is:[^space]
```

Support for RDF/Linked Data

Another major new feature of Haystack 4 is support for RDF. RDF (Resource Description Framework) is a family of World Wide Web Consortium (W3C) specifications originally designed as a metadata data model. It has come to be used as a general method for conceptual description or modeling of information that is implemented in web resources, using a variety of syntax notations and data serialization formats. [Wikipedia]. It provides ability to accomplish semantic modelling but is typically the domain of software engineers and is not very accessible by industry professionals dealing with the real world of equipment systems.

When originally creating the Haystack approach, we felt it was important that modeling of equipment systems not require users to understand advanced software modeling concepts as a starting point. Rather, Haystack allowed users to focus on the more tangible “facts” or descriptors about data and equipment that they readily understood. Haystack took a much simpler and more accessible approach to enable industry practitioners to describe the characteristics of the equipment systems and devices that they encountered by adding simple descriptive tags. This made Haystack very accessible to industry professionals and has been a key reason for its widespread adoption and success.

As the understanding semantic modeling of device and equipment data has matured, users are seeing interest in taking advantage the techniques and capabilities available with RDF. Haystack 4 has been designed to provide high-fidelity RDF expression of Haystack models. This allows software developers to utilize Haystack with the RDF techniques and semantic modelling tools they may be familiar with, without losing compatibility with the tag-oriented approach typically used by industry practitioners and tools designed for use by the people in the field.

Learn More

Haystack 4 will be a key focus of the Haystack Connect 2019 event (May 13-15 https://www.haystackconnect.org/ with a full day track dedicated to a detailed review of Haystack 4.

Detailed documentation describing the Haystack 4 design is available for Public Review on an all new website which can be found at: https://project-haystack.dev/