FALL 2019 Project Haystack ISSUE 06

connections

Journal of the Haystack Community - Solutions for Interoperable Device Data

Collaboration

Solving the Data Challenge as a Community

• Data Tagging at Ford Motor Company • Introduction to Haystack 4
• ARC Review • PitchFest • Working Group Updates

www.project-haystack.org

Contributed Article

Overhauling the Chiller Model in Project Haystack

Conserve it

"Many of the principles of Project Haystack offer significant improvements to how we model machines, yet we found the current model to be somewhat limited and also missing some critical details that open up a variety of data and applications."

At Conserve It, our daily development work includes the software modeling of chillers and chiller plants for application across HVAC systems. During the 10-plus years that we've been doing this within our commercial product PlantPRO, we have continuously learned a few key things.

Recently, we had cause to review the way we model machines in plants and in doing so, we took the time to compare our models with those that have been accepted in Project Haystack.

Many of the principles of Project Haystack offer significant improvements to how we model machines, yet we found the current model to be somewhat limited and also missing some critical details that open up a variety of data and applications.

Starting From the Basics

If we look at a basic chiller, it is essentially a number of pieces of equipment that are related to each other. Typically, a simple chiller will have two heat exchangers, a refrigerant circuit and a compressor. Whilst it is possible to build a specific model for this easily, it would be better to create a more generic model that can capture more complicated scenarios, such as multi-circuit and multi-compressor machines. Any new model that captures more complicated chillers should also take into account heat pump and dual production machines (producing chilled and hot water simultaneously), which is something we see on a daily basis.

If we look at a simple chiller example then the machine could be tagged, without worrying about data point entities yet, as follows:

```
// chiller or plant machine entity
id:@a, hvac, equip, chiller

// first heat exchanger, also referred to the
evaporator
id:@b, exchanger, function:cooling, water,
equip, equipRef:@a

// second exchanger, also referred to as the
condenser
id:@c, exchanger, function:sink, water,
equip, equipRef:@a

// refrigerant circuit
id:@d, equip, circuit, equipRef:@a, refrig,
gas

// a single compressor
id:@e, equip, comp:1, screw, equipRef:@a,
circuitRef:@d
```

Let's examine this model more closely.

A few new tags have been introduced, however what has been created is a more detailed model of a chiller. One that captures the relationships of all the "sub-equips" in a chiller. Creating higher level abstractions in the model allows us to capture more complex scenarios and even add value in these simpler cases.

The first new tag is *exchanger*. This identifies an equip as a typical heat exchanger vessel, or perhaps energy exchanger would be a better term, that would be found in a chiller or plant machine.

In this model we are using the *equipRef* tag as the means of relating all the sub-equips together to the parent *equip*.

The *function* tag is a key tag that provides for flexibility in other scenarios. This tag is an enumeration and could have the following values:

- *cooling* - the exchanger produces cooling water
- *heating* - the exchanger produces hot water
- *sink* - the exchanger rejects energy, typically heat, should be used in conjunction with either an air or water tag to denote the rejection medium
- *source* - the exchanger imports energy from either water or air, should be used in conjunction with an air or water tag to denote the energy source
- *source_sink* - the exchanger can either be importing energy or rejecting energy to a medium, it cannot do both simultaneously, should be used in conjunction with an air or water tag to denote the energy source and rejection medium

The circuit tag marks a sub-equip in the plant machine as a loop that conveys a fluid or gas of some description. When used with the *refrig* and *gas* tags it denotes this sub-equip as a refrigerant circuit in the chiller.

The *comp* tag creates another *equip* entity within our chiller that represents a compressor. The compressor has a tag denoting what compressor type it is, in this case screw, and this is done on the compressor as it is possible for plant machines to use different compressor types on the one machine.

Lastly, I have created the tag *circuitRef*, which is a most important tag especially when looking at more complicated machines. This tag allows us to know the circuit a compressor is connected to in a chiller. This information is absolutely crucial in multi-circuit/multi-compressor machines as it allows us to do much more in-depth analysis of the operation of a machine.

So far so good. we have constructed a new model for a machine. Now, what about points?

Exchanger Points

As the model for a plant machine has now been generalised a lot more, it is also possible to generalise the data point entities for an exchanger and other sub-equips. For an exchanger we would alter the existing point definitions for the chiller model to the following:

```
entering, water, temp, sensor
leaving, water, temp, sensor
leaving, water, flow, sp
water, flow, sensor
water, delta, pressure, sensor
water, valve, isolation, cmd
```

The existing chiller model defines many more points which in practice are superfluous. For example, in the hundreds of plants where we have deployed our commercial PlantPRO solution on, we have never seen an entering and leaving flow meter or sensor on the evaporator of a chiller. The cost of doing such an installation would be beyond most facilities and given that the flow through a chiller is generally the same at the entering and leaving points to the exchanger, then it need only be measured once.

Circuit Points

For the refrigerant circuit we need to introduce some new point definitions. Luckily there are not so many to do, but there are still some complicating factors that need attention. For a refrigerant circuit we would look to have entities such as:

```
discharge, pressure, refrig, gas, sensor
discharge, temp, refrig, gas, sensor
suction, pressure, refrig, gas, sensor
suction, temp, refrig, gas, sensor
```

They look all good and they capture some vital information. The refrigerant temperature sensors may not always be present so they are optional, and it is possible to convert pressure to temperature via a calculation as well.

The complication on a refrigerant circuit comes in to play when there are multiple compressors connected to a single refrigerant circuit. Each compressor may have its own suction and discharge pressure sensors rather than there being 1 suction and 1 discharge pressure sensor for the whole circuit, but in reality, most applications only care about the suction and discharge readings for a circuit. This can be handled by knowing the relationships of compressors to circuits, which of course are now captured in this new model.

Kicking the Complexity Up a Notch

To this point we have created a simple model for a single refrigerant circuit and single compressor chiller. How would this look if we had a 2 circuit 4 compressor machine?

```
// chiller or plant machine entity
id:@a, hvac, equip, chiller

// first heat exchanger, also referred to the
evaporator
id:@b, exchanger, function:cooling, water,
equip, equipRef:@a

// second exchanger, also referred to as the
condenser
id:@c, exchanger, function:sink, water,
equip, equipRef:@a

// refrigerant circuits
id:@d, equip, circuit, equipRef:@a, refrig,
gas
id:@e, equip, circuit, equipRef:@a, refrig,
gas

// compressors connected to circuit @d
id:@f, equip, comp:1, screw, equipRef:@a,
circuitRef:@d
id:@g, equip, comp:2, screw, equipRef:@a,
circuitRef:@d

// compressors connected to circuit @e
id:@h, equip, comp:3, screw, equipRef:@a,
circuitRef:@e
id:@j, equip, comp:4, screw, equipRef:@a,
circuitRef:@e
```

By constructing the model in this way, it is now possible to do some really in-depth analysis of the machine, particularly on the refrigerant circuits. By creating a relationship between compressors and circuits, it is now possible to determine which circuit is active - a vital piece of information that we use in PlantPRO when conducting performance analytics on a chiller.

Ready to Go Even Further?

In our work, we see a lot of plants with chillers and heat pumps, or chiller and heat recovery machines of various kinds. How could we model these? Pretty easily now with our much more generic and abstracted model. See below for a single compressor and single refrigeration circuit air source heat pump:

```
// heat pump or plant machine entity
id:@a, hvac, equip, heatpump

// first heat exchanger
id:@b, exchanger, function:heating, water,
equip, equipRef:@a

// second exchanger, also referred to as the
condenser
id:@c, exchanger, function:source, air,
equip, equipRef:@a

// refrigerant circuit
id:@d, equip, circuit, equipRef:@a, refrig,
gas

// a single compressor
id:@e, equip, comp:1, screw, equipRef:@a,
circuitRef:@d
```

That is a simple heat pump example. There are many more complex heat pumps as many manufacturers offer either water reversible or refrigerant reversible heat pumps which means the function of an exchanger is a dynamic setting and changes periodically depending on ambient temperature or time of year. This introduces further complexity into the model, but they are not insurmountable now that we can label an exchanger as having a function.

Heat recovery chillers are machines that can only operate when they can reject all of the heat they pull out of the cooling water into a heating loop in a building. This model is now easy to construct using the new ideas presented here.

```
// heat recovery plant machine entity
id:@a, hvac, equip, recovery, chiller

// first heat exchanger, also referred to the
evaporator
id:@b, exchanger, function:cooling, water,
equip, equipRef:@a

// second exchanger, also referred to as the
```

```
condenser
id:@c, exchanger, function:heating, water,
equip, equipRef:@a

// refrigerant circuit
id:@d, equip, circuit, equipRef:@a, refrig,
gas

// a single compressor
id:@e, equip, comp:1, screw, equipRef:@a,
circuitRef:@d

Simply by changing the function of an
exchanger a new type of machine can be
created.
```

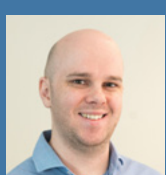
What we have been able to illustrate in this discussion is that there are alternative ways to model plant machines than what is currently in the Project Haystack standard. Furthermore, the standard can be enhanced to incorporate the ability to model different types of plant machines. With the "electrification" of our building being driven by utilities around the globe, and the drive to move away from fossil fuels growing ever larger, our community will need to know how to work with heat recovery machines of various kinds so they can provide meaningful applications to stakeholders and customers.

In practice, through our work with PlantPRO, we actually have the requirements to model much more complicated machines from a number of different manufacturers. Machines with three exchangers whose operating mode can vary dynamically, modular machines where each module has multiple compressors and circuits and the function of an exchanger can also change dynamically, but for different reasons.

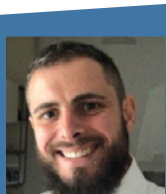
At Conserve It, we have had to find new ways such as these, to model such highly complex machines in order to be able to understand the data they generate.

If anyone has any interest in discussing this further or even taking it into a working group, I am happy to continue to work on this with the community. At the end of the day we need to find a way to model chillers and chiller plants more effectively and we are sure that others in our community also have this need.

As the Chief Software Architect at Conserve It, Richard McElhinney manages and drives the development of their industry-leading chiller plant optimization technology and associated products. He also serves as Vice President on the Project Haystack Board of Directors. .



Callum Rosel is an experienced Software Development Manager at Conserve It, who enjoys helping others expand their knowledge and understanding of the IoT world.



Filippo Bernardello is a Plant Software Engineer and the EMEA Product Manager at Conserve It SRL the brand-new European branch of Conserve It. With diverse project deployment experience, Filippo is a highly skilled engineer with in-depth knowledge of smart buildings, building automation, energy efficiency and IoT.

PlantPRO®
performance reliability optimisation

ConserveIt

BECOME A RESELLER OF AN AWARD-WINNING CHILLER PLANT OPTIMISATION SOLUTION

Call for PlantPRO VARs, MSIs and
Intelligent Building Services Providers Worldwide

WHY PLANTPRO®?

- Advanced Self Learning Algorithms based on ASHRAE Study
- Chiller Plant Performance Monitoring
- Chiller Plant Analytics and FDD
- Advanced Chiller Plant Optimization
- Chiller Plant Measurement and Verification
- Chiller Plant Control and Automation
- Chiller Plant Continuous Commisioning and Tuning

WHY PLANTPRO® CORE?

- Low Cost Chiller Plant Control System
- Chiller Plant Performance Monitoring
- Chiller Plant Alarming
- Standard Chiller Plant Optimisation
- Chiller Plant Charting
- Chiller Plant Control and Automation
- Chiller Plant Continuous Commisioning and Tuning

FEATURES AND BENEFITS OF PLANTPRO®

- Smart Machine Learning
- Multiple Chiller Manufacturers
- Up to 40% Energy Savings
- Predictive & Preventative Maintenance Diagnostics & Analytics
- Up to 0.35kW/RT or 10 COP Optimized Plant Efficiency
- Easy BMS/CMS Integration

powered by niagara framework®

Project Haystack Founding Member

We're looking for PlantPRO VARs, MSIs and Intelligent Building Services Providers Worldwide

conserveitiot.com plantpro@conserveitiot.com

Project Haystack

Intel® is a registered trademark of Intel Corporation. All other trademarks are the property of their respective owners.

www.project-haystack.org

Copyright ©2019 Project Haystack Corporation.
All Rights Reserved.