
NS
18
NIAGARA
SUMMIT
CONNECTING
THE WORLD

Developer Showcase

Automating Tagging in Niagara 4

Mary P. Boelk

Johnson Controls, Inc.

Introduction – Niagara 4 Tagging Support

Tags associate Metadata with an Entity

- **Tagging in Niagara 4 Provides**
  - **Organization – Tag Dictionaries and Tag Service**
  - **Ability to Associate Tags to Entities**
  - **Searching – NEQL**
  - **APIs**

Introduction – Motivation

Business and Personal Motivation

- **Improve the Johnson Controls FX Appliance**
- **Investigate the Niagara 4 Tagging API**

Automatic Tagging – Implementation Highlights

Code Examples

- **Verification of Tags**
- **Application of Tags**

Automatic Tagging – Verify Tag Exists

```
Collection <TagDictionary> tdicts = tds.getTagDictionaries();
Iterator <TagDictionary> it = tdicts.iterator();
while (it.hasNext()) {
     TagDictionary td = it.next();
     if (td.getNamespace().equals(tagNS)) {
         // found the tag dictionary
         Id thisTagId =  Id.newId(tagNS,tagName);
         BTagDictionary TD =  (BTagDictionary)td;
         BTagInfoList til = TD.getTagDefinitions();
```

Automatic Tagging – Verify Tag Exists (Continued)

```
if (til.containsTagId(thisTagId) == true) {
    // containsTagId returns <Optional> TagInfo, not TagInfo
    if (til.getTag(thisTagId).isPresent())
        // return the TagInfo for the tag
        TagInfo res = til.getTag(thisTagId).get();
    }
}
```

Automatic Tagging – Verify TagGroup Exists

```
if (td.getNamespace().equals(tagNS)) {
        Iterator <TagGroupInfo> tagit = td.getTagGroups();
        while (tagit.hasNext()) {
            TagGroupInfo tag = tagit.next();
            if (tag.getName().equals(tagName)) {
                // again return the tag info for the tag group
                res = tag;
                break;
            }
         }
```

Automatic Tagging – Verify Tag can be added

```
boolean res = false;
BComponent c = (BComponent) entity;
if (c != null) {
    if (c.tags().contains(tag.getTagId())) {
        res = false;  // already have tag
    }
    else {
        res = tag.isValidFor(entity);
    }
}
```

Automatic Tagging – Add Marker Tag

```
if (tag.getTagType().is(BMarker.TYPE)) {
    try {
        tag.setTagOn(entity);
    }
    catch (Exception e) {
    }
}
```

Automatic Tagging – Add Tag Group

```
Iterator <TagInfo> tags = tag.getTags();
//add each tag in group to entity
while (tags.hasNext()) {
    tags.next().setTagOn(entity);
}
```

Automatic Tagging – Add Value Tag

```
if (tag.getTagType().is(BDouble.TYPE))
    tag.setTagOn(entity,BDouble.make(value));
else if (tag.getTagType().is(BInteger.TYPE))
    tag.setTagOn(entity,BInteger.make(value));
else if (tag.getTagType().is(BString.TYPE))
    tag.setTagOn(entity,BString.make(value));
else if (tag.getTagType().is(BOrd.TYPE))
    tag.setTagOn(entity,BOrd.make(value));
else {  // no match on type
```

Automatic Tagging – Summary

- **Value of Automated Tagging**
- **API Observations**
- **Thank you for your attention!**
- **Contact information**
  - **mary.boelk@jci.com**

Automatically Adding nHaystack Tags from Direct Tags

Steve Michals

Johnson Controls

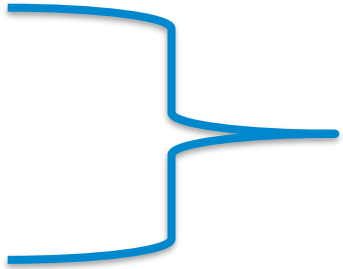
steve.michals@jci.com

Automating the addition of NHaystack Tags

Turn Direct Tags into nHaystack Tags

NiagaraAX Metadata

Niagara4 Tags



nHaystack Tags

Adding nHaystack Tags by Hand

- Add ***NHaystackService*** to Services
  - Needed: **nhaystack-rt.jar** and **nhaystack-wb.jar**
- Add one or more **“site”** components
  - Site has default tags for area, “geo” values and Time Zone
- Add **“equip”** to devices that represent pieces of Equipment
  - Equip has default tags for Site Reference and Floor
- Optional – create a “customTagDict.csv” for Point Tags
  - Drag point to the NHaystack Service to get Custom Tags added
- Add additional tags to the “**haystack**”, “**equip**” or “**site**” components

Niagara4 Haystack Tags vs. nHaystack Tags

Example Equipment Controller:

VAV-1 Controller (Variable Air Volume)

- Niagara4 Haystack Tags
  - hs:vav
  - hs:ahuRef
  - hs:air
- nHaystack Tags for “equip”
  - {floorName:"Floor Zone 1" spaceRef:@C.fxApp.spaceRoot.Floor-1.Zone-1 siteRef:@C.fxApp.spaceRoot.site vav air navNameFormat:"%parent.displayName%" ahuRef:@C.Drivers.BacnetNetwork.AHU~2d1}

Niagara4 Haystack Tags vs. nHaystack Tags

Example Control Point:

ZN-T Point (Zone Temperature Sensor)

- Niagara4 Haystack Tags
  - hs:zone
  - hs:temp
  - hs:sensor
  - hs:air
  - jci:summary
- nHaystack Tags
  - {air zone summary temp sensor}

Get the Niagara4 Tags - Code

```
//The tags() method of BComponent returns all tags for an Entity
// both Implied and Direct Tags
BComponent c = (BComponent) entity;
Tags alltags = c.tags();

//SmartTags is a subclass of Tags. The advantage of casting Tags
// to SmartTags is the presence of the two methods
// getImpliedTags() and getDirectTags(). Direct Tags will be the
// user added Tags vs. tags added via Tag Dictionary rules.
Tags smart = ((SmartTags) c.tags()).getDirectTags();

//Get the Smart Tags Iterator
Iterator<Tag> smit = smart.iterator();
```

Turn Direct Tags into NHaystack - Code

```
//Create an NHaystack Tag Dictionary Component
    HDictBuilder hdb = new HDictBuilder();
//Iterate SmartTags collection and add the haystack tag by type
    Tag a = smit.next();
    Id id = a.getId();
    String name = a.getId().getName();
    String valu = a.getValue().toString();
//Determine the Tag Type
    a.getValue().getType().is(BMarker.TYPE)
// Types: BOrd.TYPE, BBoolean.TYPE, BDouble.TYPE, BInteger.TYPE
```

Turn Direct Tags into NHaystack – Code (cont.)

```
// Add an Ord Reference tag
    BOrd refOrd = BOrd.make(a.getValue().toString());
    BComponent ordComp = refOrd.resolve(base).getComponent();
    NHRef ref = TagManager.makeSlotPathRef(ordComp);
    hsvalu = ref.getHRef().toString();
    hdb.add(name, ref.getHRef());

// This code sequence turns a Niagara Ord reference:
//      station:|slot:/Drivers/BacnetNetwork/AHU$2d1
// To the NHaystack reference:
//      C.Drivers.BacnetNetwork.AHU~2d1
```

Turn Direct Tags into NHaystack – Code (cont.)

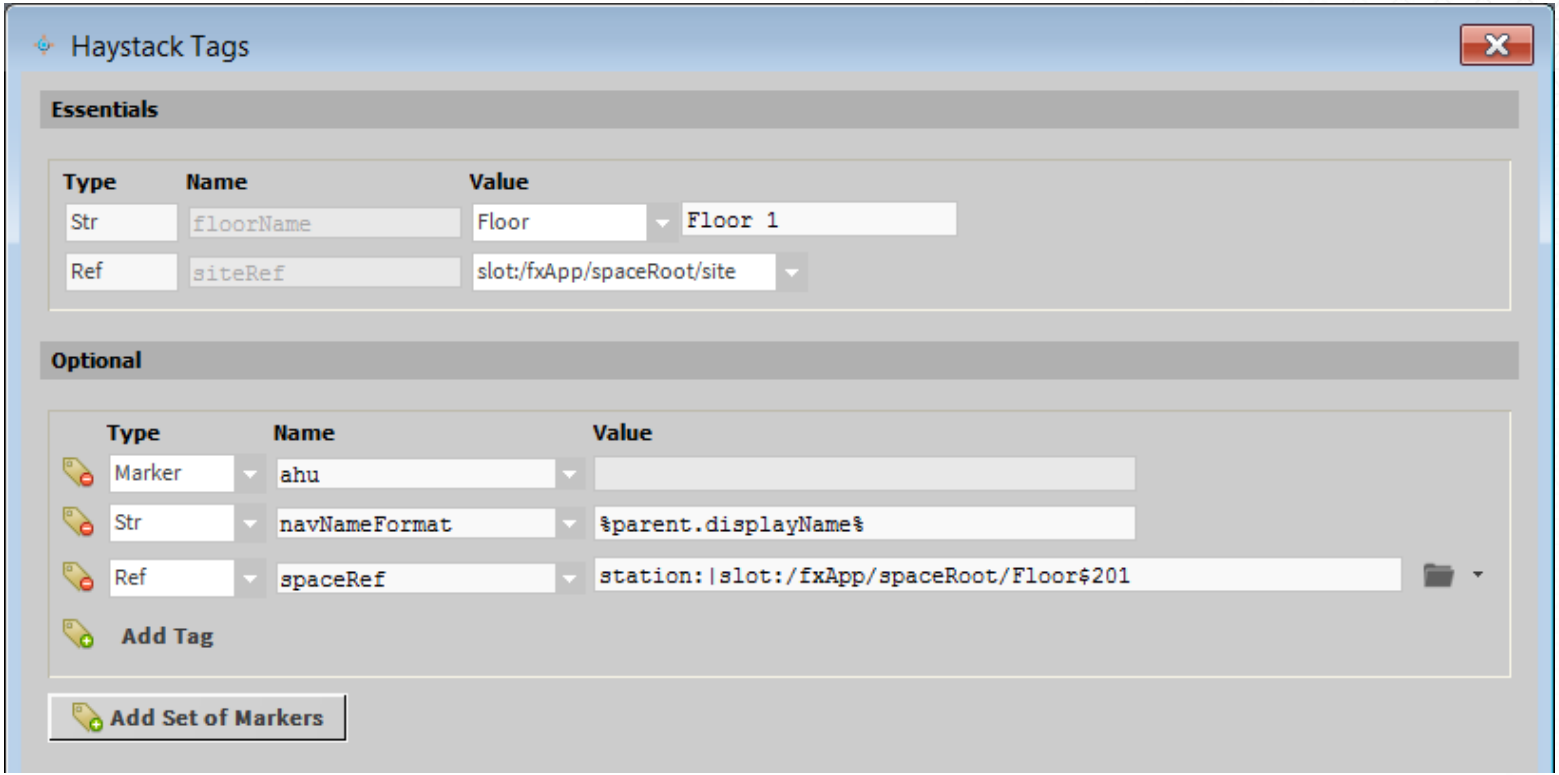
```
// Add the “site” information if the Component is a Device
    String sitePath = hsSiteRef;  // <-- saved site reference
    hdb.add("navNameFormat","%parent.displayName%");
    hdb.add("siteRef", HRef.make(sitePath));
// Add a BHEquip component to the Device
    comp.add("equip", new BHEquip());
    BHEquip eqp = (BHEquip)comp.get("equip");
    eqp.setHaystack(BHDict.make(hsTags));
//For Devices, the hsTags string will have the form:
//  { floorName:"Floor 1" siteRef:@C.fxApp.spaceRoot.site
//    chiller navNameFormat:"%parent.displayName%“ }
```

Turn Direct Tags into NHaystack – Code (cont.)

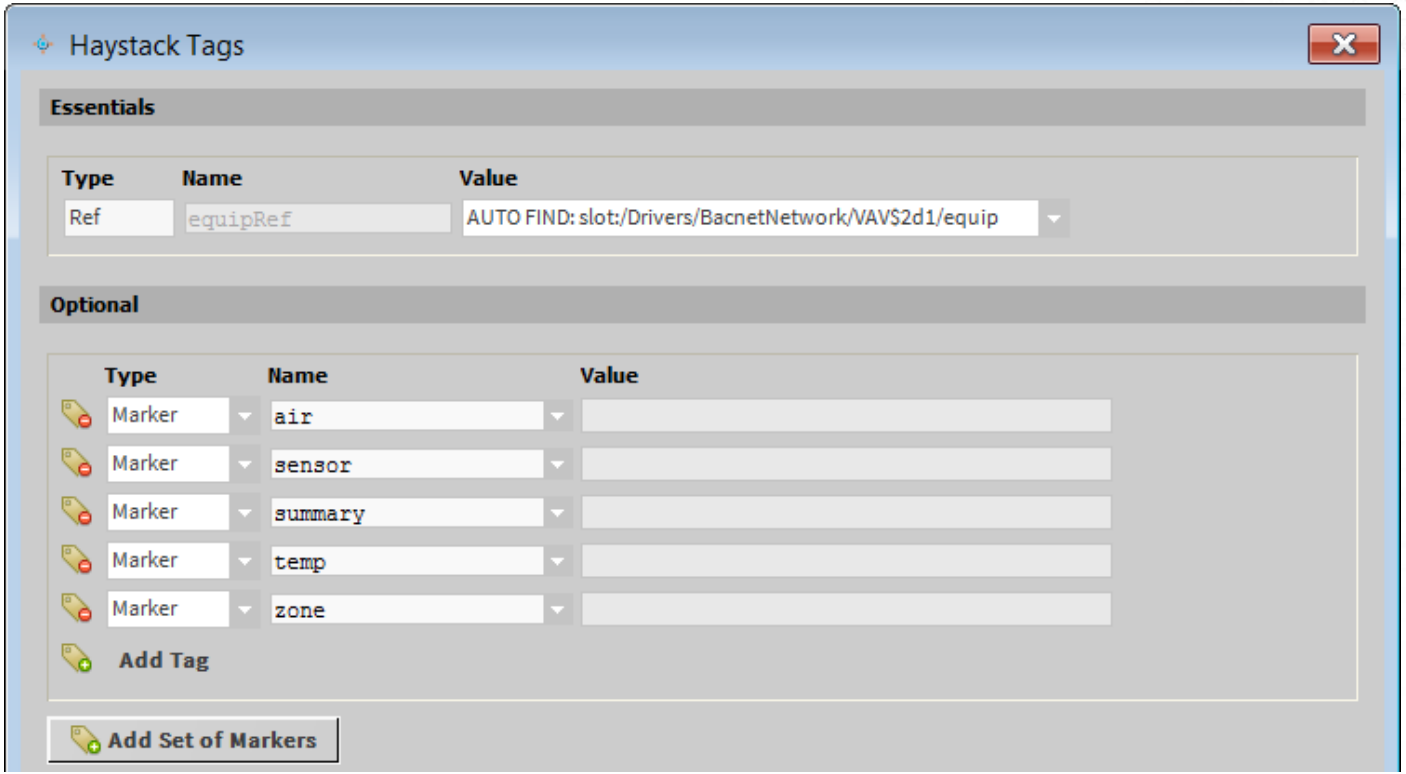
```
// Add the haystack slot when component is not a Device
    String hsTags = hdb.toDict().toString();
    if (hsTags.length() > 2) {
       if (comp.get("haystack") == null) {
              comp.add("haystack", BHDict.DEFAULT);  }
           comp.set("haystack", BHDict.make(hsTags));
          }

// The hsTags string will have the form:
//  {air zone summary temp sensor}
```

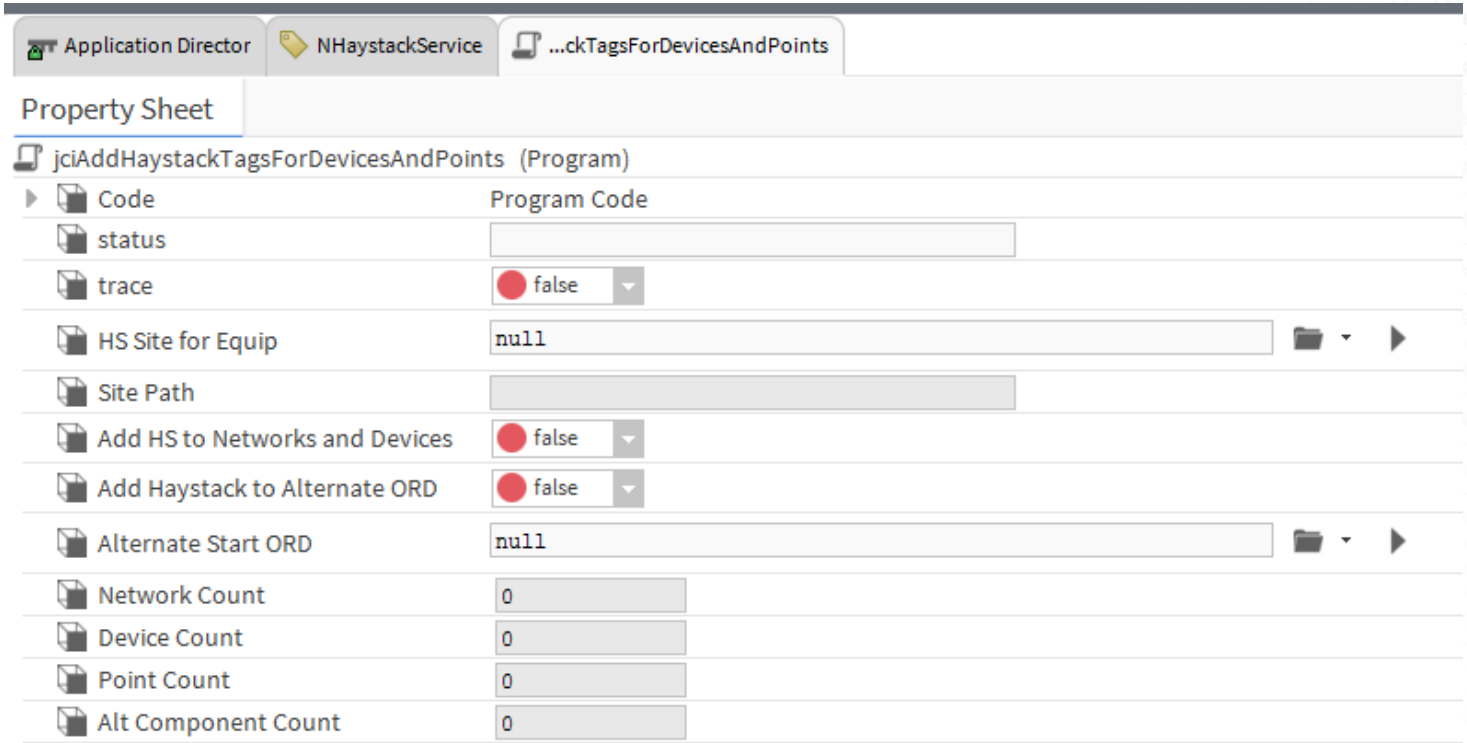
NHaystack equip Dialog



NHaystack point tags Dialog



Potential Options for Haystack Automation



Summary

- Any Tag Automation can give your integration a big advantage.
- The process to turn Direct Tags into nHaystack Tags is not difficult.
- Additional functionality:
  - Automatically add the “site” and “equip” objects
  - Automate more than just the Networks and Devices.
- Niagara4 Tagging and nHaystack Tagging will likely converge over time.

Taking Niagara to the Edge…

Richard McElhinney
Chief Software Architect
Conserve it

Presentation Scope

- Platform Independent Niagara
- Developing Edge applications for manufacture

About Me

Trained Niagara TCP 2005

Trained Niagara Developer 2006

Developed Niagara Applications:

- Drivers
- DALI Lighting
- Plant Control
- Analytics
- Energy Management

Chief Software Architect, Conserve It

- Plant Control & Optimisation
- Embedded Software development
- Niagara Portability
- OEM software solutions
- Product distribution

Developing using PIN

Platform Independent Niagara

What is PIN?

- Niagara sub-framework allowing developers to port Niagara to:-
  - Any hardware platform capable of running a JVM
  - Support for Linux/QNX based operating systems only
- Provides SDK (NPSDK) guiding developers on how to do this
- Build tooling and example toolchains already setup
- Clear documentation on what must be done
- New concepts of Niagara that may not be familiar
- This sub-framework is not covered in the standard developer training

Factors to consider

Developer Skills

- C++/C
- GCC
- Other toolchains such as Gradle for Native Dev
- Linux OS selection and knowledge

What hardware?

- Are you designing your own?
- Can software development start before hardware is ready?
- Raspberry PI
- BeagleBone
- Other IoT prototyping platforms?

Factors to consider

User Expectations

- Target market is familiar with Tridium hardware
- Product must meet basic expectations
- Market expects any Niagara product to operate like Tridium's
- No need of any special training…open the box and go!

Time to market

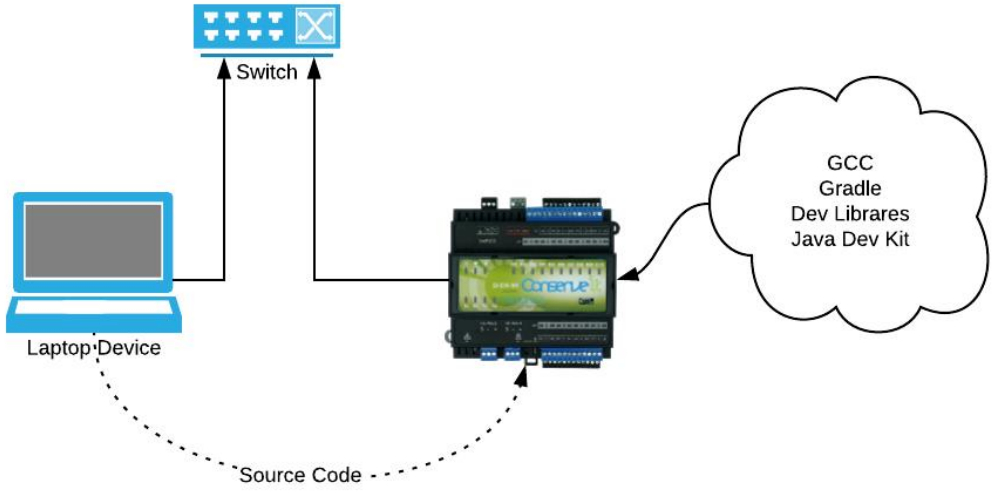
- Plan ahead and allow ample time for development
- BACnet MS/TP is complicated to implement
- Testing load is more than is first expected
- Many small details to take care of
- Read documentation carefully to get full understanding

Things to do

- Select a JVM! Tridium doesn't to this for you
- Implement native classes and methods, this is documented
- Host ID generation – strict guidelines, spend time to get it right
- Manufacturing considerations
- Test – more on this later!
- Native services – if required
- BACnet MS/TP – developer must do own implementation (Hint!)
- Dist files

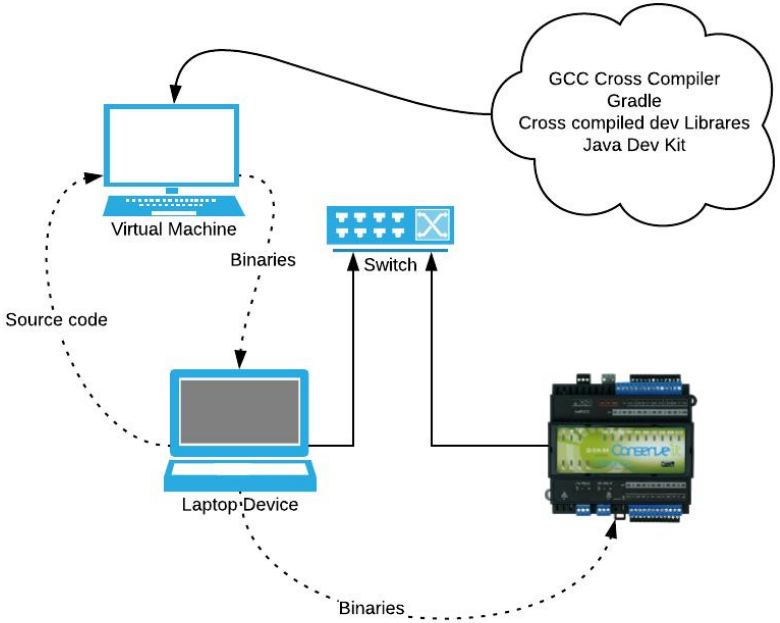
Development tools

- Setup 1
  - Use text editor on PC/Mac
  - Transfer source to target
  - Build on target
  - Slow to build



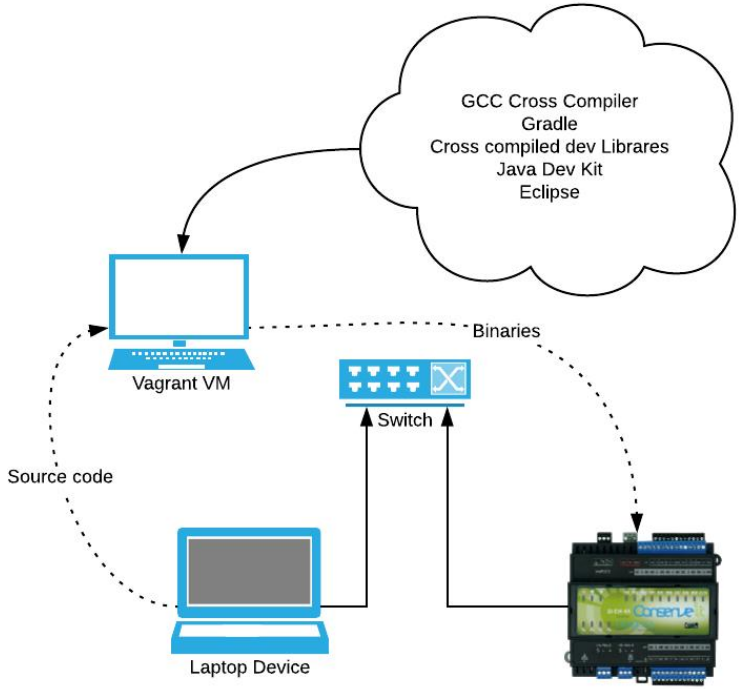
Development tools

- Setup 2
  - User text editor on PC/Mac
  - Create basic VM with Vagrant
  - Setup build/cross compile tools on VM
  - Transfer source to VM and build with CC tools
  - Automate transfer of binaries to target with Gradle plugin
  - Lots of steps/moving parts
  - Fast to build binaries



Development tools

- Setup 3
  - Completely work on Linux VM
  - Debian 9
  - Installed cross-compilation tools
  - Installed Eclipse
  - Import NPSDK into Eclipse using Gradle plugin
  - Edit and build code in Eclipse
  - Also transfer binaries to target directly from Eclipse



Testing

- Don’t underestimate the work required in testing
- Can be very slow and manual
- Automated test suite provided by Tridium
- Consider end user experience must be seamless in the Workbench
- Your implementation must look and feel like a JACE
- All aspects of the platform implementation are checked by Tridium

NO HIDING! ☺

Security

- Build in security from the start
- Design built-in users upfront
- Which user will run ‘niagarad’
- Can that user login to target?
- How is the Platform User DB implemented?

- Check file permissions early and often
- Will your device permit SSH terminal sessions or not?
- Protect sensitive files and folders

There is always more!

A real “Edge” application

Multistack FlexSys Gen III Chiller controller

What does the “Edge” look like?

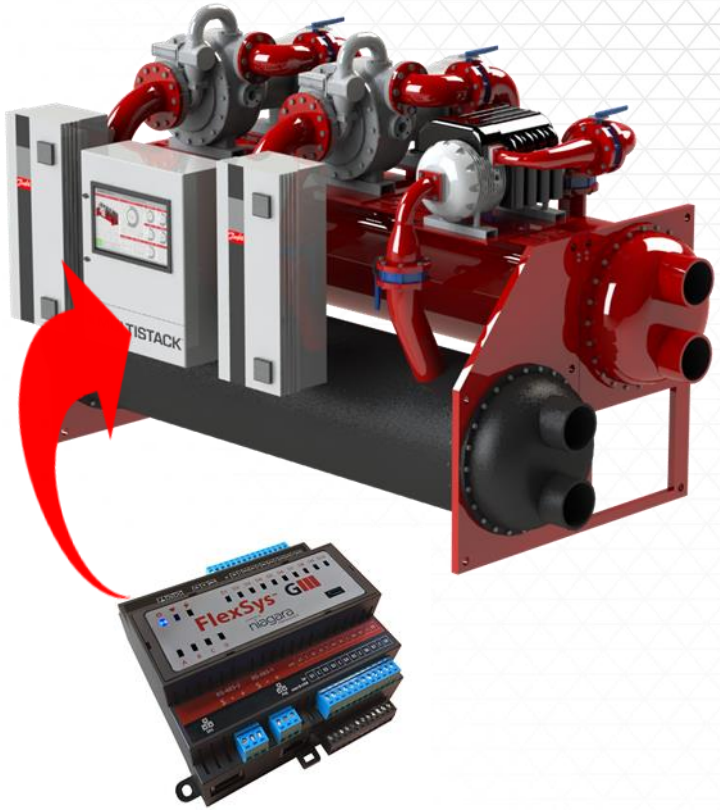
Multistack MagLev™

Flooded Chiller

- Water Cooled Chiller
- 1 – 5 compressors
- Mixed use of compressor type
- Fully assembled in Sparta, Wisconsin

What is our application?

- FlexSys Gen III Chiller Controller
- Provide a hardware platform for the controller
- Appliance type application for setup/configuration in factory
- Control program
- Manufacturer and End User UI



The Task

Develop a chiller	Turbocor TT Compressors	Turbocor VTT Compressors	Mixed Compressor Tech	Next generation UI
•Multiple chiller models •Onboard devices •Peripheral Devices	•TT 300 •TT 350 •TT 400 •TT 700	•VTT 1200 •More VTT sizes	•1 TT + 1 VTT •1 TT + 2 VTT •1 TT + 3 VTT •1 TT + 4 VTT •2 TT + 2 VTT •More combinations	•HTML (obviously) •Easy Nav •Lots of live charts •Simple charting •Appliance config

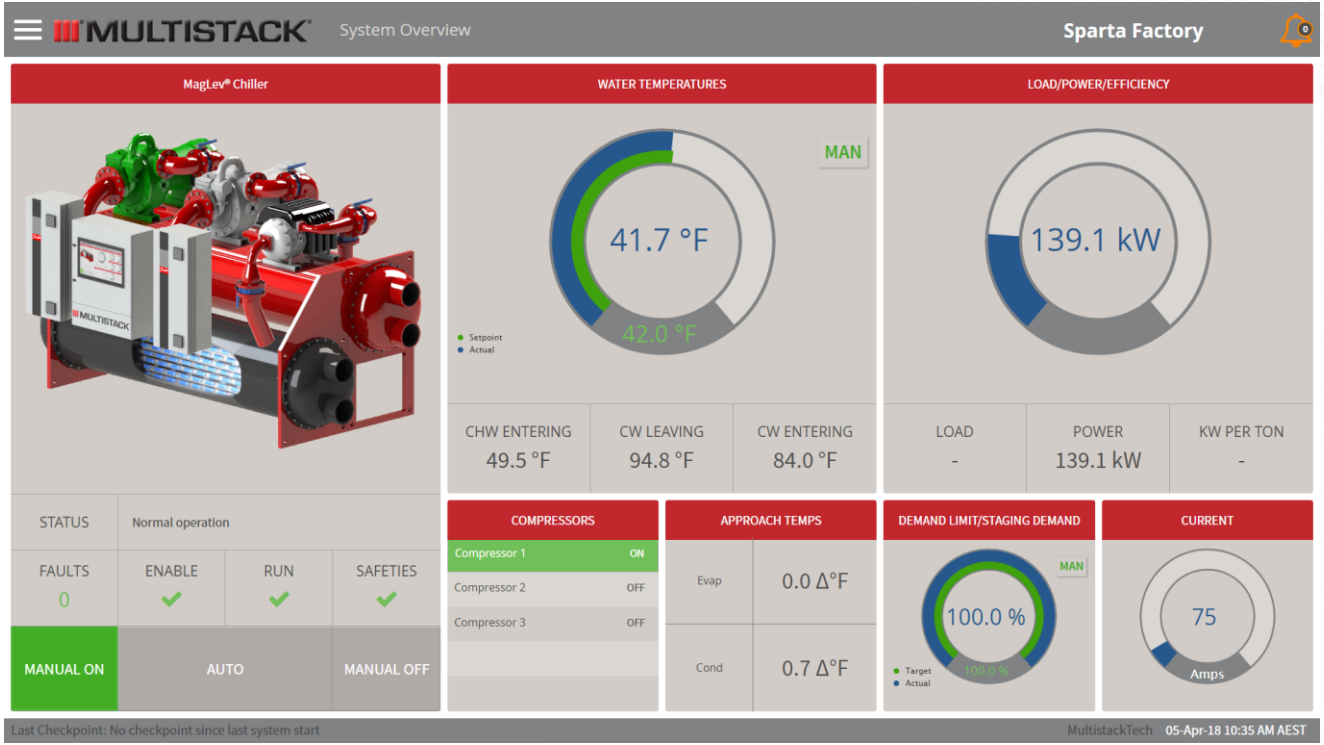
Developing an “Edge” Control Application

- Logic prototyped on Wiresheet
- Easier to analyze/troubleshoot
- Initially a few custom components
- Wiresheet logic converted to custom components
- Developed templates for re-use
- Programmatic configuration of certain parts of the station
- Initialize all onboard mechanical components
- Multiple cascading PID loop based control for compressor demand
- PID loop control for EXVs
- Staging/sequencing of compressors based on runtime and user priorities
- Management of failure modes

Developing an “Edge” UX/UI Application

- Complete UI built on new BAJAUX web stack from Tridium
- Views on components
- Views built from widgets
- Widgets built from custom field editors
- All UI values can be displayed in SI and Imperial units
- UI ready for language support
- Extensive custom development of gauges and 3D images
- Purpose built charting using commercially available Highcharts JS library
- Gauges hand coded
- Extensive use of Handlebars templates
- Custom built alarm console

The Whole User Experience!



3D Renders and lots of status

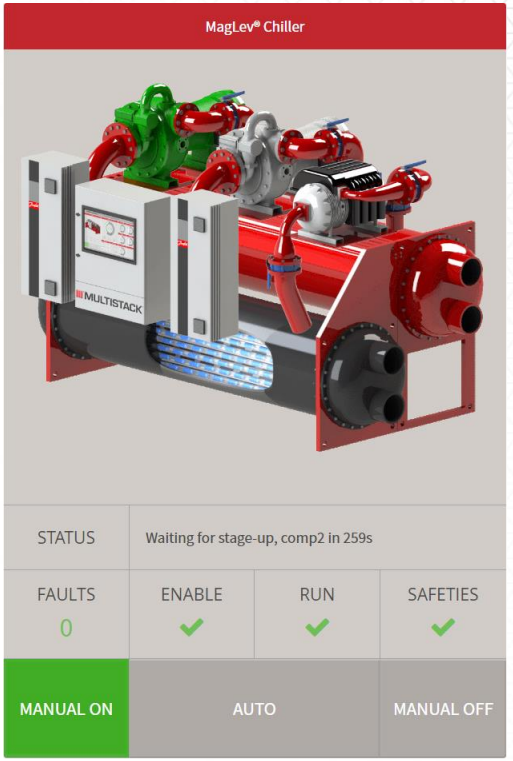
Each render is a 3D mechanical model built in Solid Works

Comprised multiple images

Images layered

Each element can be swapped out depending on state

Plain English messages to user



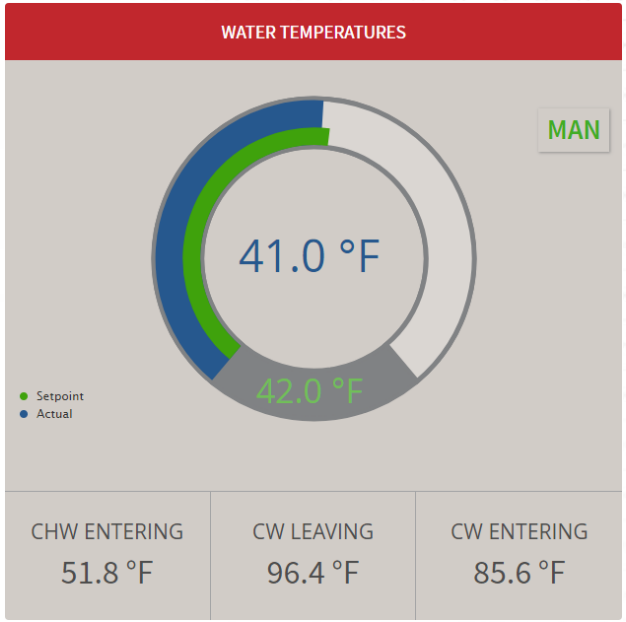
Custom Gauges

Combination of graphical and textual elements

“Chasing” graph

Override button

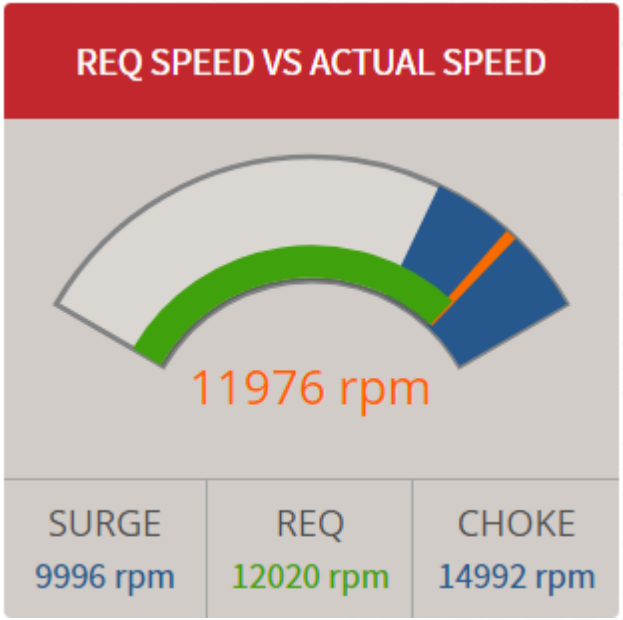
Setpoint vs. Actual



Multiple Data Point Display

Combining multiple data points

4 Data points shown in 1 widget
Upper/lower band of operation
Requested speed of compressor
Actual speed of compressor




MULTISTACK
Charts
Sparta Factory
0
SELECT CHART DATA
TIME RANGE
Today
From: 5 Apr 2018, 12:00 am
To: 5 Apr 2018, 11:59 pm
POINTS
Chiller
Externals
Compressor1
Compressor2
Compressor3
Economizer1
MainlineEXV1
MainlineEXV2
RESET
Last Checkpoint: No checkpoint since last system start
MultistackTech
05-Apr-18 11:43 AM AEST

Gotchas!

- Peformance of network comms on chiller
- Discovered History database performance problem on startup
- Client was expecting significant data retention on device
- Required uSD card in the device, this increased price in a price sensitive context
- Station startup time
- UI performance

Lessons learned

- Don’t assume an “Edge” application is simple
- Don’t assume an “Edge” application can operate/rely on the cloud
- Pay attention to product benchmarks the customer will be applying
- Testing the “Edge” application in the real world is hard
- What testing resources are required?
- Understand that “Edge” applications are a combination of hardware and software…both must be developed

Developer Bootcamp 2018

- Done!
- Done!
- Done!
- Done!


NS
18
NIAGARA
SUMMIT
CONNECTING
THE WORLD