
NS
18
NIAGARA
SUMMIT
CONNECTING
THE WORLD

Niagara Development with Tags, Relations and Queries

Blake Puhak & Scott Hoye
Software Engineers – Tridium

Questions

- How can I program with Tags & Relations?
- How do I use NEQL with Tags & Relations?
- What's new in Niagara 4.6?
- Didn't Blake's hair used to be longer?
- Is Scott really as cool as he looks?

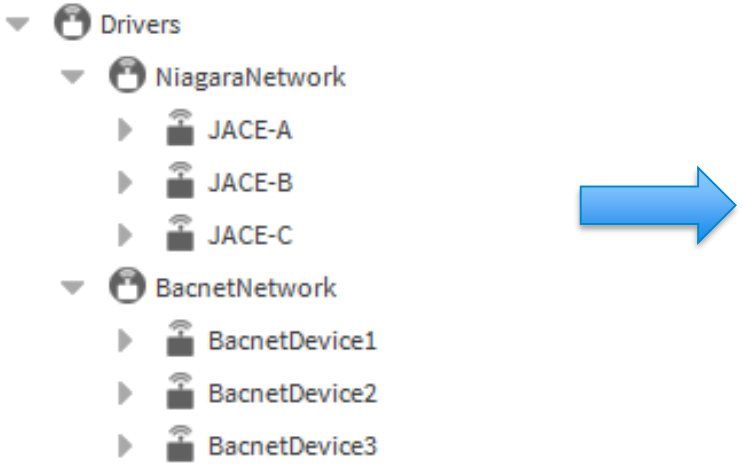
First, the basics…

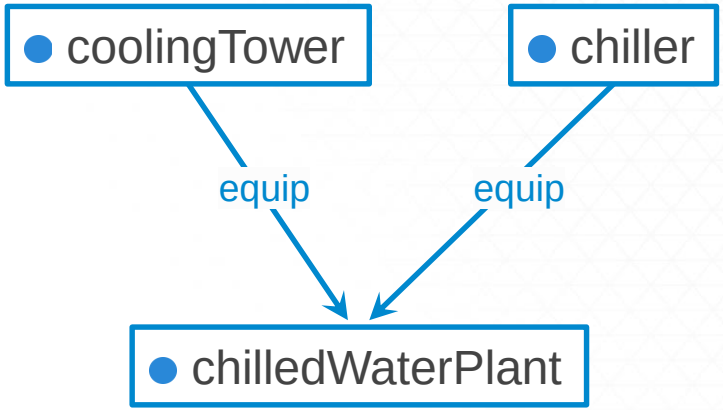
Tags & Relations

Describe system in a common way

Tags & Relations

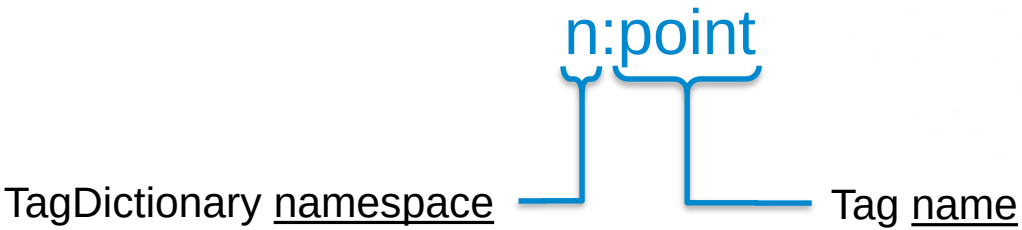
Model system like the real world





Tags

- Tags have an ID and optionally a value
  - Value must be BIDataValue
  - Defined in Niagara by a TagDictionary
    - Can be direct or implied



Tags

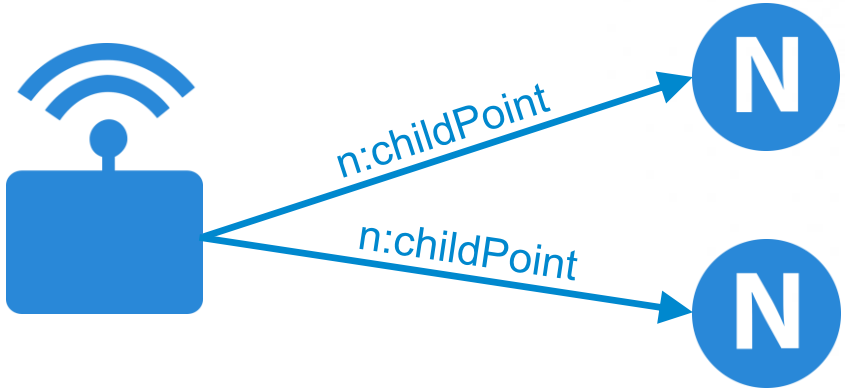
- Tags have an ID and optionally a value
  - Value must be BIDataValue
  - Defined in Niagara by a TagDictionary
    - Can be direct or implied

Relations

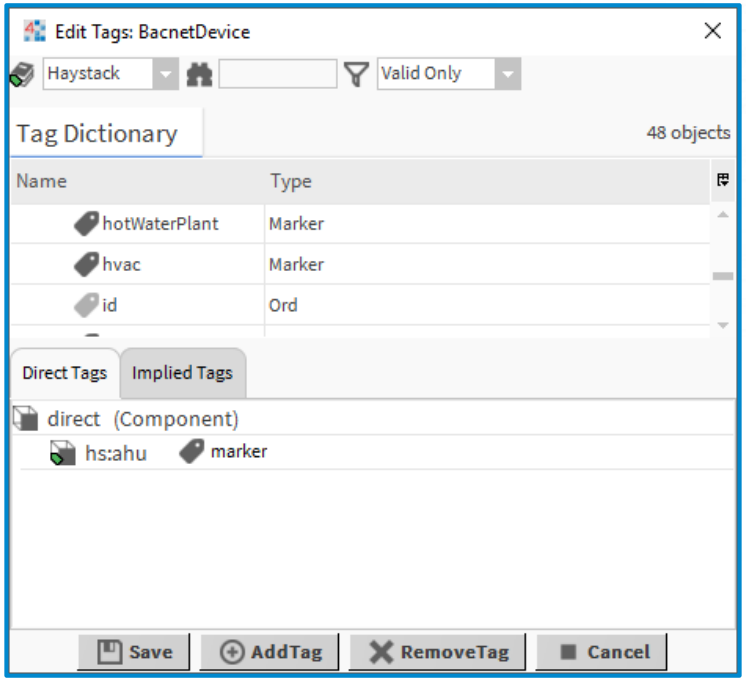
- Relations are directional – Outbound or Inbound
  - Endpoint ORD defines the related Entity

Relations

- Relations are directional – Outbound or Inbound
  - Endpoint ORD defines the related Entity



Using the Tags & Relations



Entity API

- javax.baja.tag
  - Entity
  - Tag
  - Relation
  - Id
  - ...

javax.baja.tag.Entity

```
public interface Entity extends Taggable {

  Tags tags();

  Relations relations();

  Optional<BOrd> getOrdToEntity();
}
```

javax.baja.tag.Tag

```
public final class Tag {

  public Tag(Id id, BIDataValue value);

  Id getId();

  BIDataValue getValue();
}
```

javax.baja.tag.Relation

```
public interface Relation extends Taggable {
  Id getId();
  boolean isInbound();
  boolean isOutbound();
  Entity getEndpoint();
  BOrd getEndpointOrd();
  Tags tags();
}
```

Adding a Tag

```
// Component Model
myComponent.add(SlotPath.escape("b:floor"), BInteger.make(1),
                Flags.METADATA);


// Entity API
Tag floorTag = Tag.newTag("b:floor", BInteger.make(1));
entity.tags().set(floorTag);
```

Adding a Relation

```
// Component Model
Relation r = myComponent.makeRelation(Id.newId("hs:ahuRef"),
                                      targetComponent,
                                      cx);


// Entity API
Relation r = entity.relations().add(Id.newId("hs:ahuRef"),
                                    otherEntity);
```

Checking if a Tag Exists

```
// Component Model (only works for Direct Tags) ✗
BValue tag = myComponent.get(SlotPath.escape("n:point"));
if (tag != null) {
  ...
}

// Entity API ✓
entity.tags().get(Id.newId("n:point")).isPresent();
```

Traversing a Relation

```
/* Find the chilled water plant of a chiller */
Id plantRef = Id.newId("hs", "chilledWaterPlantRef");

chiller.relations().get(plantRef, Relations.OUT).ifPresent(
  relation -> {
    Entity plant = relation.getEndpoint();
    System.out.println(plant.getOrdToEntity());
  }
);
```

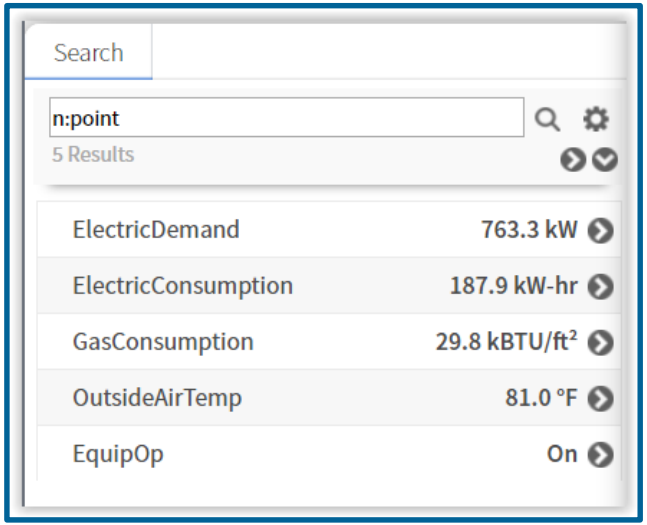
Traversing all Relations

```
/* Get all devices in a BACnet network with an hs:chiller tag */
Id chiller = Id.newId("hs:chiller");
bacnetNetwork.relations().getAll(Id.newId("n:childDevice"))
        .stream()
        .map(Relation::getEndpoint)
        .filter(entity -> entity.tags().get(chiller).isPresent())
        .forEach(entity -> {
          BBacnetDevice chiller = (BBacnetDevice)entity;
          System.out.println(chiller.getAddress());
        });
```

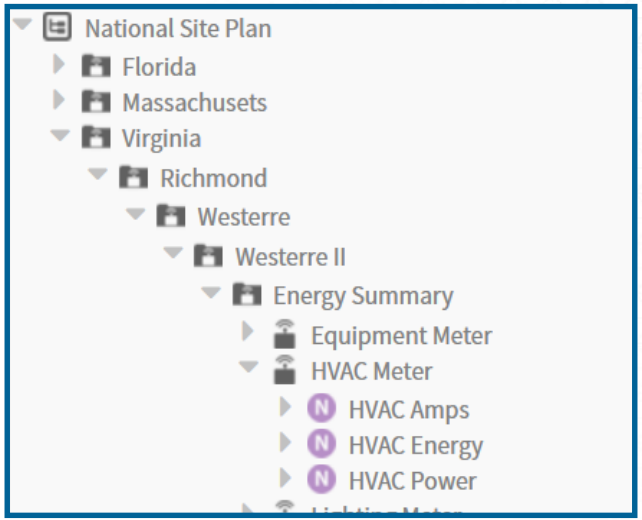
OK, but why is it important to add Tags and Relations to my Niagara Station Components?

So we can query them!

Search



Hierarchies



NEQL – Niagara Entity Query Language

- Simple query syntax to find Entities by Tags and/or following Relations
- Queries formed in body of **neql** ORD scheme
- Detailed information on NEQL in Developer Guide of Workbench help
- Why another query language? What about BQL?

NEQL	vs.	BQL
Find Entities by Tags/Relations		Find Objects by Niagara APIs and Types
No BFormat operations (ex. parent.toPathString)		Supports BFormat operations (ex. parent.toPathString)
No projection or complex functions		Allows projection & complex functions
Supports parameterized queries		No parameterized queries
Simple syntax - can build support in 3rd party databases (non-Niagara)		Can only run against Niagara spaces (Station, History, Alarm spaces)

NEQL Examples

Find all points

```
neql:n:point
```

Find all AHU or VAV devices

```
neql:hs:ahu or hs:vav
```

Find all chiller devices with a minimum cooling capacity of 1000 tons

```
neql:hs:chiller and hs:coolingCapacity >= 1000
```

Find all points having a parent device with a name containing “Meter”

```
neql:n:parentDevice-> n:name like ".*Meter.*"
```

NEQL Examples

Find all child devices under a BacnetNetwork

```
slot:/Drivers/BacnetNetwork|neql:traverse n:childDevice->
```

Specify a namespace to condense tags

```
namespace:hs|neql:ahu or vav
```

Combine NEQL and BQL queries

```
neql:n:device|bql:select toPathString, status
```

Programming with NEQL

- Resolve **BOrds** in code
- Results are **BQueryResults**
  - Allows for iteration and streaming of Entity results
- Quickly test query ORDs on your station using Workbench (CTRL-L)

Tag Query Example

```
/* Print out AHU component ORDs found anywhere in the local station */

BOrd ord = BOrd.make("neql:hs:ahu");

// Cast the resolved NEQL ORD to a BQueryResult
BQueryResult result = (BQueryResult)ord.get();

result.stream().forEach(entity -> {
  entity.getOrdToEntity().ifPresent(System.out::println);
});
```

Relation Traversal Example

```
/* Print out the BACnet address of any chiller devices
   under a BacnetNetwork in the local station */

BOrd ord = BOrd.make("neql:traverse n:childDevice-> where hs:chiller");

// Use the BacnetNetwork as the base when resolving the ORD
BComponent bacnetNetwork = Sys.getService(BBacnetNetwork.TYPE);
BQueryResult result = (BQueryResult)ord.get(bacnetNetwork);

result.stream().forEach(entity -> {
  BBacnetDevice chiller = (BBacnetDevice)entity;
  System.out.println(chiller.getAddress());
});
```

Parameterized Query Example

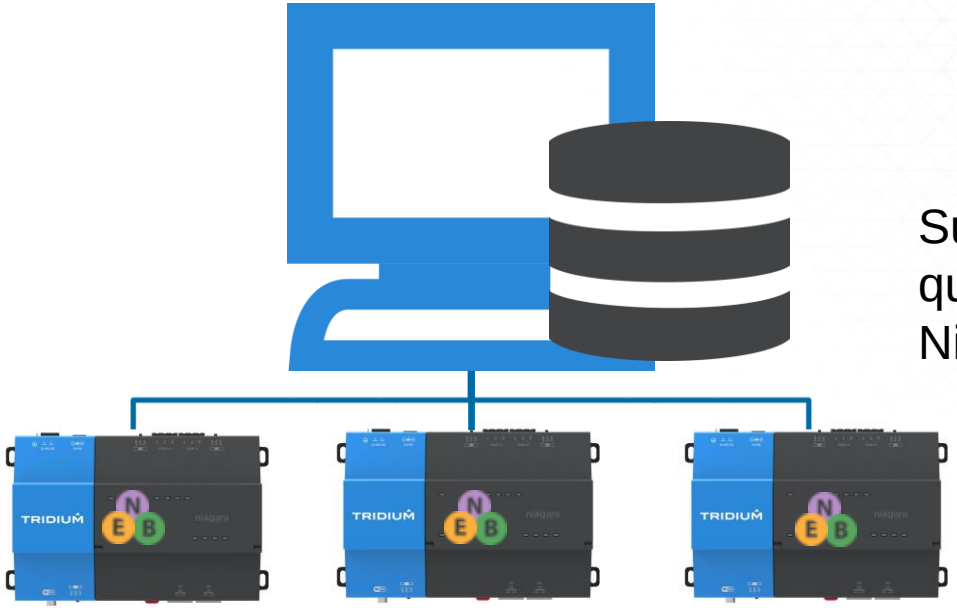
```
static BOrd CHILLER_QUERY = BOrd.make("neql:hs:chiller and hs:coolingCapacity >= {param}");
static BOrd POINT_QUERY = BOrd.make("neql:hs:point and hs:cool and hs:stage = {param}");

static Stream<Entity> runQueryWithParameter(BOrd queryOrd, BIDataValue param) {
  // Specify query parameters as facets
  BFacets facets = BFacets.make("param", param);
  return ((BQueryResult)queryOrd.get(Sys.getStation(), facets)).stream();
}

// Find all chillers with a minimum cooling capacity of 500 tons
Stream<Entity> chillers500 = runQueryWithParameter(CHILLER_QUERY, BInteger.make(500));
// Find all chillers with a minimum cooling capacity of 1000 tons
Stream<Entity> chillers1000 = runQueryWithParameter(CHILLER_QUERY, BInteger.make(1000));
// Find all single stage AHU cooling elements (points)
Stream<Entity> singleStagePts = runQueryWithParameter(POINT_QUERY, BInteger.make(1));
```

Neato! What's next?

Niagara 4.6 – Introducing the System Database



Supports NEQL queries that span Niagara System!

System Database Query Example

```
/* Print out AHU component ORDs found across all of your
   NiagaraStations connected to the supervisor's NiagaraNetwork */

BOrd ord = BOrd.make("sys:|neql:hs:ahu");

// Based on current Niagara virtual availability, results are either
// Niagara virtual components or lightweight Entities
BQueryResult result = (BQueryResult)ord.get();

result.stream().forEach(entity -> {
  entity.getOrdToEntity().ifPresent(System.out::println);
});
```

Summary

- You can programmatically add Tags and Relations and then query for them using NEQL
- NEQL queries can be scoped using a base and parameterized using facets
- NEQL can also be run against the System Database in Niagara 4.6


NS
18
NIAGARA
SUMMIT
CONNECTING
THE WORLD