



# Developing a Custom Tag Dictionary

*Tim Urenda*  
*Tridium, Inc.*

**NS2024**  
APRIL 15 - 17 | ANAHEIM, CA

# Tagging in Niagara

- Semantic information for station entities
- Tag Dictionary elements
  - Tags
  - Relations
  - Tag groups
  - Rules
- Direct or Implied tags and relations
- Uses
  - NEQL
  - Search
  - Tag-based PX bindings
  - Hierarchies
  - Templates

The screenshot displays the Niagara Tag Manager interface. At the top, the title bar shows 'WonderWorldPark' and 'CriticCoveZone'. The 'Selected Components' list on the left includes 'CriticCoveZone', 'TranquilTidesZone', 'VelocityValleyZone', and 'WhimsyWoodsZone'. The 'Available Tags' table lists various tags and their types:

| Tag        | Tag Type |
|------------|----------|
| Tags       |          |
| zone       | Marker   |
| restaurant | Marker   |
| ride       | Marker   |
| seat       | Marker   |
| shop       | Marker   |
| show       | Marker   |
| id         | String   |

Below the 'Available Tags' table, the 'Showing tags on' section is set to 'CriticCoveZone' with checkboxes for 'Direct' and 'Implied' tags. The table below shows the resulting tags:

| Tag Id        | Tag Name    | Value          |
|---------------|-------------|----------------|
| thmpk:zone    | zone        | Marker         |
| n:displayName | displayName | CriticCoveZone |
| n:type        | type        | baja:Folder    |
| n:station     | station     | themepark      |

At the bottom, there are buttons for 'Edit', 'Add', and 'Delete Tags'.



# Tag Dictionary

- Tag Dictionary Service
- Tag Dictionary Palette
- Dictionary Configuration
  - File-based
    - CSV
    - JSON
  - Module-based
    - Palette

The screenshot displays the Tag Dictionary configuration interface. At the top, a header bar shows the current station as 'Station (themepark)' and includes tabs for 'Config', 'Services', and 'TagDictionaryService'. Below this, a 'Database' table lists the configured dictionaries.

| Name      | Type                   | Status | Version                         | Namespace | Fault Cause |
|-----------|------------------------|--------|---------------------------------|-----------|-------------|
| Niagara   | Niagara Tag Dictionary | {ok}   | 1.5                             | n         |             |
| Themepark | Smart Tag Dictionary   | {ok}   | 1.0                             | thmpk     |             |
| Haystack  | Hs Tag Dictionary      | {ok}   | 3.0.2 N.2 w/ SmartRefs (import) | hs        |             |

Below the database table, the interface is split into two main panels. The left panel, titled 'Palette', shows a tree view of tag types. The 'String' type is currently selected. The right panel, titled 'Property Sheet', displays the configuration for the selected 'String' tag.

**Property Sheet - Tag Definitions**

| Tag Name | Type   |
|----------|--------|
| Marker1  | Marker |
| String1  | String |

**Tag Group Definitions**

| Tag Group Name | Type           |
|----------------|----------------|
| ControlPointId | ControlPointId |

**Relation Definitions**

| Relation Name           | Type         |
|-------------------------|--------------|
| Is control:ControlPoint | ControlPoint |

**Tag Rules**

| Tag Rule Name | Type   |
|---------------|--------|
| String1       | String |

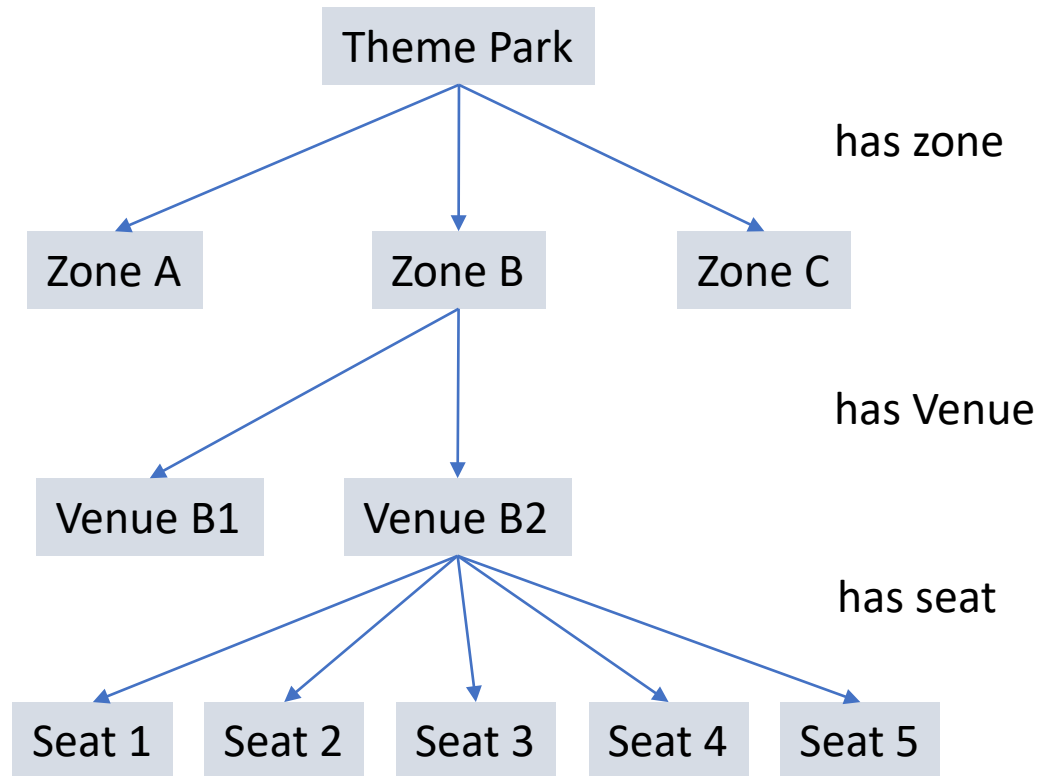
# JSON Structure

- Tags, Relations, Tag Groups, Rules
  - See brick.json, test.json
- All entries are required
  - Contents of **tags**, **relations**, **tagGroups**, **rules** can be empty array
- When in doubt, build dictionary element in a station and export to JSON

```
test.json x
test.json x
1 97      "tags": [
2 98      {
3 99        "name": "markerAlways"
4 100     },
5 101     {
6 102       "default": "5",
7 103     }
8 104   ],
9 105   "relations": [
10 106   {
11 107     "name": "deviceRef",
12 108   },
13 109   {
14 110     "name": "equipRef",
15 111   },
16 112   {
17 113     "name": "siteRef",
18 114   },
19 115   {
20 116     "name": "spaceRef",
21 117   }
22 118 ],
23 119 "default": "test",
24 120 "valueType": "baja:String",
25 121 "name": "stringOr",
26 122 "validity": {
27 123   "type": "tagdictionary:Or",
28 124   "conditions": [
```

| test.json | test.json |
|-----------|-----------|
| 1         | 83        |
| 2         | 84        |
| 3         | 85        |
| 4         | 86        |
| 5         | 87        |
| 6         | 88        |
| 7         | 89        |
| 83        | 90        |
| 97        | 91        |
| 155       | 92        |
| 198       | 93        |
|           | 94        |
|           | 95        |
|           | 96        |

# Our Dictionary JSON

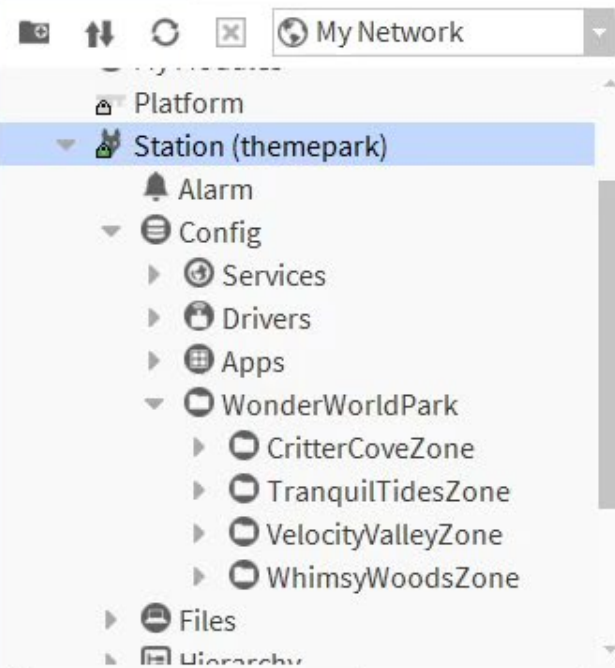


| Structure                   | themeparkTagsRelations.json     |
|-----------------------------|---------------------------------|
| themeparkTagsRelations.json | {                               |
| frozen true                 | "namespace": "thmpk",           |
| namespace "thmpk"           | "version": "1.0",               |
| neqlizeExcludedRelations    | "frozen": true,                 |
| neqlizeExcludedTags ""      | "neqlizeExcludedTags": "",      |
|                             | "neqlizeExcludedRelations": "", |
| relations                   | "tags": [                       |
| {} object                   | {"name": "zone"...},            |
| {} object                   | {"name": "restaurant"...},      |
| {} object                   | {"name": "ride"...},            |
| {} object                   | {"name": "shop"...},            |
| {} object                   | {"name": "show"...},            |
| {} object                   | {"name": "seat"...},            |
| rules                       | {                               |
| tagGroups                   | "name": "id",                   |
| tags                        | "valueType": "baja:String",     |
| {} object                   | "default": ""                   |
| {} object                   | }                               |
| {} object                   | ],                              |
| {} object                   | "tagGroups": [                  |
| {} object                   | ],                              |
| {} object                   | "relations": [                  |
| {} object                   | {"name": "hasZone"...},         |
| {} object                   | {"name": "hasVenue"...},        |
| {} object                   | {"name": "hasSeat"...}          |
| {} object                   | ],                              |
| version "1.0"               | "rules": [                      |
|                             | ]                               |
|                             | }                               |

My Host: HONCLD17KQXL3IB.global.ds.honeywell.com (themepark) : Station (themepark)

Station Summary

## Nav



## Palette



## Station (themepark)

6 objects

| Name      | Description                           |
|-----------|---------------------------------------|
| Alarm     | Alarm Database                        |
| Config    | The station configuration database    |
| Files     | File System accessed over Fox session |
| Spy       | Diagnostics information for remote VM |
| Hierarchy | Hierarchy views of remote station     |
| History   | History database                      |

## Summary Properties

11 objects

| Property           | Value                   |
|--------------------|-------------------------|
| Station Name       | themepark               |
| Host               | /10.18.155.151          |
| Host Model         | Workstation             |
| Host Model Version |                         |
| Host Product       |                         |
| Host Id            | Win-CBE2-A257-6F3A-80E9 |
| Niagara Version    | 4.14.0.121.18           |



# Public API

- Types and interfaces for developing custom
  - Tag
  - Group
  - Relation
  - Rule
  - Condition
  - Scope
  - Dictionary
- “Smart” elements are implied automatically
  - On Entities when Rule conditions are satisfied

```
▼ tag
  > io
  > util
    © BasicRelation
    ⓘ BIDataPolicy
    ⓘ BIEntity
    ⓘ BIEntitySpace
    ⓘ DataPolicy
    ⓘ Entity
    © EntityWrapper
    © Id
    ⓘ Relation
    ⓘ RelationInfo
    ⓘ Relations
    ⓘ SmartRelations
    ⓘ SmartTagDictionary
    ⓘ SmartTags
    © Tag
    ⓘ TagDictionary
    ⓘ TagDictionaryService
    ⓘ Taggable
    ⓘ TagGroupInfo
    ⓘ TagInfo
    ⓘ Tags
```

```
java.baja.tagdictionary
  > data
    © BDynamicEnumTagInfo
    © BInfoList
    © BRelationInfo
    © BRelationInfoList
    © BScopedTagRule
    © BSimpleTagInfo
    © BSmartTagDictionary
    © BTagDictionary
    © BTagDictionaryService
    © BTagGroupInfo
    © BTagGroupInfoList
    © BTagInfo
    © BTagInfoList
    © BTagRule
    © BTagRuleCondition
    © BTagRuleList
    © BTagRuleScope
    © BTagRuleScopeList
    ⓘ TagRule
    ⓘ TagRuleScope
```



# Custom Tags

- Extend BTagInfo
  - TagInfo interface
  - getTag(Entity entity)
  - getDefaultValue()
- Tag value usually derived from parent object
- Validity rule can be replaced
  - @NiagaraProperty(name = "validity", ...)
- Haystack4 dictionary examples
  - BCurValTag
  - BUnitTag
  - BEquipRefTag

```
© BCurValTag.java x
32     @NiagaraType
33     @NiagaraProperty(
34         name = "validity",
35         type = "BTagRuleCondition",
36         defaultValue = "new BIsTypeCondition(BControlPoint.TYPE)",
37         override = true
38     )
39     @ public class BCurValTag extends BTagInfo
```

```
© BCurValTag.java x
83     @Override
84     public Tag getTag(Entity entity)
85     {
86         if (entity instanceof BControlPoint)
87         {
88             BStatusValue statusValue =
89                 ((BControlPoint)entity).getOutStatusValue();
90             return makeTagForStatusValue(getTagId(), statusValue);
91         }
92         return null;
93     }
94
95     @Override
96     public BIDataValue getDefaultValue()
97     {
98         return BString.DEFAULT;
99     }
100
```

# Our Tags

- BIdTag

```
@Override
public BIdDataValue getDefaultValue() { return BString.DEFAULT; }

@Override
public Tag getTag(Entity entity)
{
    // Check that we have a control point inside a station space
    if (!(entity instanceof BControlPoint))
    {
        return null;
    }
    BComponent component = (BComponent)entity;
    BComponentSpace space = component.getComponentSpace();
    if (space == null)
    {
        return null;
    }
    BComponent root = space.getRootComponent();
    if (!(root instanceof BStation))
    {
        return null;
    }

    // Add a smart tag that defines an ID string containing a Type 3 UUID based on the NSpace ord.
    String stationName = ((BStation)root).getStationName();
    String handleOrd = component.getHandleOrd().encodeToString();
    int idLength = stationName.length() + /*:*/ 1 + handleOrd.length();
    StringBuilder id = new StringBuilder(idLength).append(stationName)
        .append(":").append(handleOrd);

    return new Tag(getTagId(), BString.make(UUID.nameUUIDFromBytes(
        id.toString().getBytes(StandardCharsets.UTF_8)).toString()));
}
```

# Custom Relations

- Extend `BRelationInfo`
  - `RelationInfo` interface
  - `getRelation(Entity entity)`
  - `addRelations(Entity entity, Collection<Relation> relations)`
- Relation usually established from entity information
- Haystack4 dictionary examples
  - `BHaystack4SiteRelation`
  - `BSpaceRelation`

```
© BHaystack4SiteRelation.java x
55  @Override
56  public Optional<Relation> getRelation(Entity source)
57  {
58      Id siteRefId = getSiteRefId();
59      if (hasDirectRelation((BComponent) source, siteRefId))
60      {
61          return Optional.empty();
62      }
63
64      Entity site = null;
65      if (source instanceof BControlPoint)
66
© BHaystack4SiteRelation.java x
67
68  138  @Override
69  139  public void addRelations(Entity source, Collection<Relation> relations)
70  140  {
71  141      // Add the outbound relation
72  142      Optional<Relation> outbound = getRelation(source);
73  143      outbound.ifPresent(relations::add);
74  144
75  145      // Add inbound siteRef relations if the given entity has the site tag.
76  146      if (!(source instanceof BComponent) || !hasSiteTag(source))
77  147      {
78  148          return;
79  149      }
80  150
81  151      BComponent site = (BComponent) source;
82  152      for (Relation siteRef : new ComponentRelations(site)
83  153          .getAll(getSiteRefId(), Relations.IN))
84  154      {
85  155          addSiteRelations(siteRef, relations);
86  156      }
87  157  }
```

# Our Relation

- BVenueSeatRelation

```
@Override
public Optional<Relation> getRelation(Entity source)
{
    ArrayList<Relation> relations = new ArrayList<>();
    addRelations(source, relations);
    if (relations.isEmpty())
    {
        return Optional.empty();
    }
    return Optional.of(relations.get(0));
}
```



# Custom Rules and Conditions

- Existing rules and conditions are most likely sufficient
- Conditions can be nested and combined
- Extend **BTagRule**
  - **TagRule** interface
- Extend **BTagRuleCondition**
- Examples
  - **BScopedTagRule**
  - **BIsTypeCondition**
  - **BBooleanFilter**

The screenshot displays a software development environment with three main components:

- Project Explorer:** A tree view on the left showing a folder named 'condition' containing several classes: BAlways, BAnd, BBooleanFilter, BHasAncestor, BHasRelation, BIsTypeCondition, BNever, BNot, and BOr.
- Property Sheet:** A panel on the right titled 'Property Sheet' showing the configuration for a 'device' object. It includes a 'Condition' section with a 't' (Boolean Filter) and a 'Tag List' section with a 'device' marker.
- Code Editor:** Two Java files are open. The first, 'BScopedTagRule.java', shows a constructor and an 'evaluate' method. The second, 'BScopedTagRule.java' (repeated in the image), shows the '@NiagaraType' and '@NiagaraProperty' annotations and the 'evaluate' method implementation.

```
public BScopedTagRule(BTagRuleCondition cond, BTagRuleScopeList scopeList) {
    setCondition(cond);
    setScopeList(scopeList);
}

@Override
public boolean evaluate(Entity entity) {
    return getScopeList().isInScope(entity) && getCondition().test(entity);
}
```

```
@NiagaraType
@NiagaraProperty(
    name = "scopeList",
    type = "BTagRuleScopeList",
    defaultValue = "new BTagRuleScopeList()"
)
public class BScopedTagRule extends BTagRuleCondition {
    // ...
}
```

# Our Condition

- BIsNamedWith

```
public boolean test(Entity entity)
{
    if (!(entity instanceof BObject))
    {
        return false;
    }

    if (!getStartsWith().isEmpty() && ((BObject)entity).asComplex()
        .getName().startsWith(getStartsWith()))
    {
        return true;
    }

    if (!getEndsWith().isEmpty() && ((BObject)entity).asComplex()
        .getName().endsWith(getEndsWith()))
    {
        return true;
    }

    return false;
}

@Override
public boolean testIdealMatch(Type type)
{
    return false;
}
```

# JSON Encoding / Decoding

- JSON methods
  - `encodeToJson(JSONWriter writer)`
  - `decodeFromJson(JSONObject tagJson)`
  - Default implementation might be fine
    - Override is needed if additional properties are declared
- JSON classes
  - Uses `com.tridium.json` package
  - Not yet in public API `javax.baja`
- Brick dictionary example
  - `BBrickInverseRelation`

```
© BBrickInverseRelation.java ×
36  @NiagaraType
37  /*
38   Inverse relation name.
39   */
40  @NiagaraProperty(
41   name = "inverseName",
42   type = "String",
43   defaultValue = ""
44  )
45  public class BBrickInverseRelation
46   extends BRelationInfo
```

```
© BBrickInverseRelation.java ×
158  @Override
159  public void encodeToJson(JSONWriter writer)
160  {
161   super.encodeToJson(writer);
162   writer.key(inverseName.getName()).value(getInverseName());
163  }
164
165  @Override
166  public void decodeFromJson(JSONObject conditionJson)
167  {
168   super.decodeFromJson(conditionJson);
169   setInverseName(conditionJson.getString(inverseName.getName()));
170  }
171 }
```

# Our JSON Encoding / Decoding

- JSON methods
  - `BVenueSeatRelation` – no properties, so default methods are fine
  - `BIdTag` – “validity” property is an override, so default methods are fine
  - `BIsNamedWith` – need to define methods to include new properties

```
@Override
public void encodeToJson(JSONWriter writer) {
    // In many cases you will need to call super.encodeToJson(writer);
    writer.key(startsWith.getName()).value(getStartsWith());
    writer.key(endsWith.getName()).value(getEndsWith());
}

@Override
public void decodeFromJson(JSONObject conditionJson) {
    // In many cases you will need to call super.decodeFromJson(conditionJson);
    setStartsWith(conditionJson.getString(startsWith.getName()));
    setEndsWith(conditionJson.getString(endsWith.getName()));
}
```



# Tag Dictionary

- Extend `BTagDictionary` or `BSmartTagDictionary` only if JSON is insufficient
- `SmartTagDictionary` interface
  - Keep default methods to resolve implied tags and relations
- Example dictionary
  - `BHaystack4TagDictionary`
    - Many properties set to `READONLY`
    - `importDictionary` parameter `ORD`
    - `doImportDictionary(BOrd defsFolder)`

```
© BHaystack4TagDictionary.java x
51  @NiagaraType
52  @NiagaraProperty(
53      name = "namespace",
54      type = "String",
55      defaultValue = "h4",
56      flags = Flags.READONLY,
57      override = true
58  )
59  @NiagaraProperty(
60      name = "frozen",
61      type = "boolean",
62      defaultValue = "true",
63      flags = Flags.READONLY,
64      override = true
65  )
66  @NiagaraAction(
67      name = "importDictionary",
68      parameterType = "BOrd",
69      defaultValue =
70      "BOrd.make(\"module://haystack/data/haystack-4-defs\")",
71      override = true
72  )
```

# Dictionary Module

- Palette BOGs
  - Smart Tag Dictionary with JSON File ORD
  - Custom Tag Dictionary
- File-based dictionary update does NOT require a new module
  - Niagara Haystack 4 vs. Brick dictionaries

```
brick-rt\module.palette x
<?xml version="1.0" encoding="UTF-8"?>
<bajaObjectGraph version="1.0">
  <p m="b=baja" t="b:UnrestrictedFolder">
    <p n="Brick" m="td=tagdictionary" t="td:SmartTagDictionary">
      <p n="namespace" v="bk"/>
      <p n="importDictionaryOrd" t="b:Ord"
        v="module://brick/com/tridium/brick/data/brick.json"/>
      <p n="frozen" v="true" f="r"/>
    </p>
  </p>
</bajaObjectGraph>
```

```
haystack-rt\module.palette x
<?xml version="1.0" encoding="UTF-8"?>
<bajaObjectGraph version="1.0">
  <p m="b=baja" t="b:UnrestrictedFolder">
    <p n="Haystack4" m="haystack=haystack"
      t="haystack:Haystack4TagDictionary"/>
    <p n="Haystack3" t="b:UnrestrictedFolder">
      <p n="StandardItems" t="b:UnrestrictedFolder">
        <p n="Haystack" t="haystack:HsTagDictionary">
        </p>
      </p>
    </p>
    <p n="SmartRelationsIncluded" t="b:UnrestrictedFolder">
      <p n="Haystack" t="haystack:HsTagDictionary">
        <p n="tagsImportFile" t="b:Ord"
          v="module://haystack/com/tridium/haystack/data/smartRefsImport.csv"/>
        </p>
      </p>
    <p n="displayNames" t="b:NameMap" f="rh"
      v="{StandardItems=%lexicon(haystack:palette.standardItemsFolder.displa
        SmartRelationsIncluded=%lexicon(haystack:palette.smartRelationsInclude
    </p>
  </p>
</bajaObjectGraph>
```

# Our Module

- Project contents
- JSON file
  - With rules
- Palette file
  - BSmartTagDictionary
  - JSON ORD
- Gradle file

```
themeparkWithRules.json x
1 {
2   "namespace": "thmpk",
3   "version": "1.0",
4   "frozen": true,
5   "rules": [
6     {
7       "name": "frozen",
8       "type": "themepark:IdTag",
9       "validity": {
10        "type": "tagdictionary:IsTypeCondition",
11        "objectType": "control:ControlPoint"
12      }
13     }
14   ]
15 }
```

```
// See documentation at module://docDeveloper/doc/build.html#dependencies
// dependency types
dependencies {
  // NRE dependencies
  nre(":nre")

  // Niagara module dependencies
  api(":baja")
  api(":control-rt")
  api(":tagdictionary-rt")

  // Test Niagara module dependencies
  moduleTestImplementation(":test-wb")
}

<p n="frozen" v="true" f="r"/>
</p>

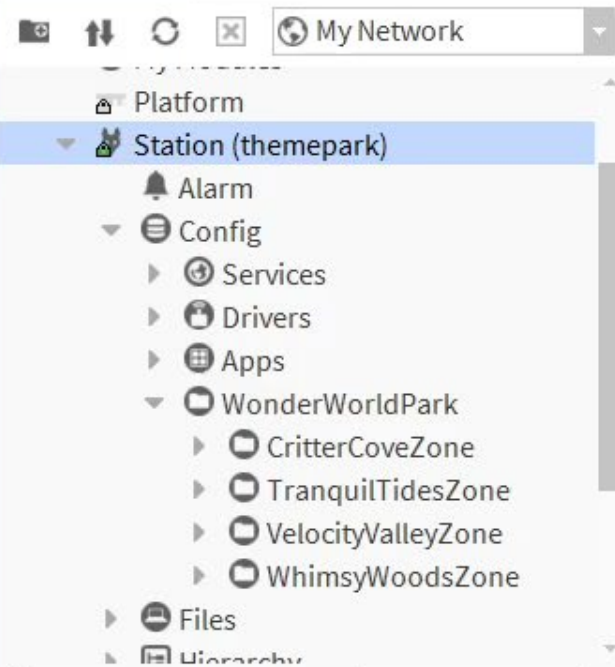
tasks.named<Jar>( name: "jar") {
  from( sourcePath: "src") {
    include( ...includes: "com/tagdictionary/themepark/data/*.json")
  }
}
```

```
145 "type": "themepark:IdTag",
146 "validity": {
147   "type": "tagdictionary:IsTypeCondition",
148   "objectType": "control:ControlPoint"
149 }
150 }
151 ]
152 }
153 ]
154 }
```

My Host: HONCLD17KQXL3IB.global.ds.honeywell.com (themepark) : Station (themepark)

Station Summary

## Nav



## Palette



## Station (themepark)

6 objects

| Name      | Description                           |
|-----------|---------------------------------------|
| Alarm     | Alarm Database                        |
| Config    | The station configuration database    |
| Files     | File System accessed over Fox session |
| Spy       | Diagnostics information for remote VM |
| Hierarchy | Hierarchy views of remote station     |
| History   | History database                      |

## Summary Properties

11 objects

| Property           | Value                   |
|--------------------|-------------------------|
| Station Name       | themepark               |
| Host               | /10.18.155.151          |
| Host Model         | Workstation             |
| Host Model Version |                         |
| Host Product       |                         |
| Host Id            | Win-CBE2-A257-6F3A-80E9 |
| Niagara Version    | 4.14.0.121.18           |



# That's All Folks!

- [github.com/tridium](https://github.com/tridium)

