TRIDIUM

NS2024

POWER OF PARTNERSHIP

Developing a Custom Tag Dictionary

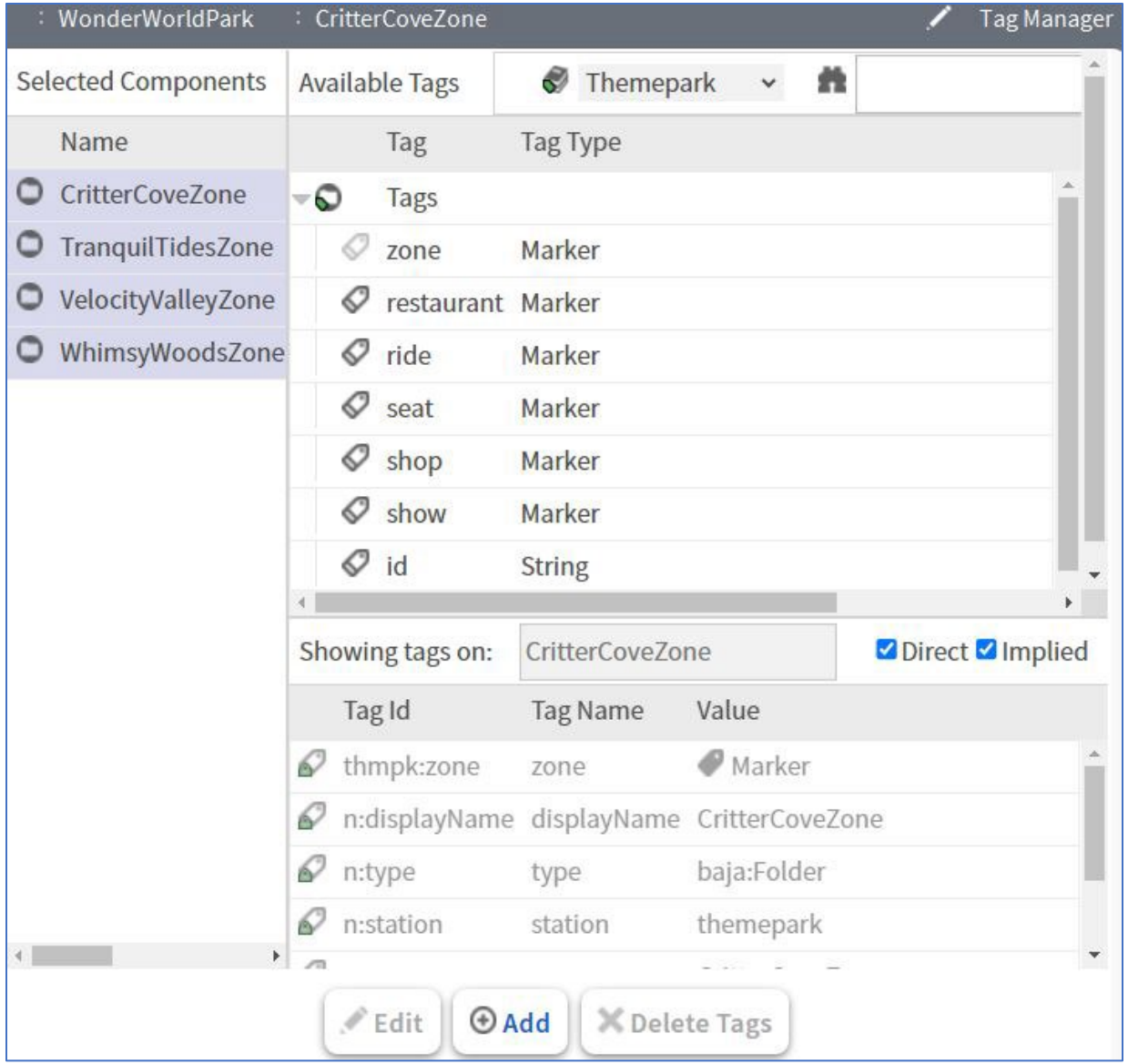
Tim Urenda

Tridium, Inc.

TRIDIUM

Tagging in Niagara

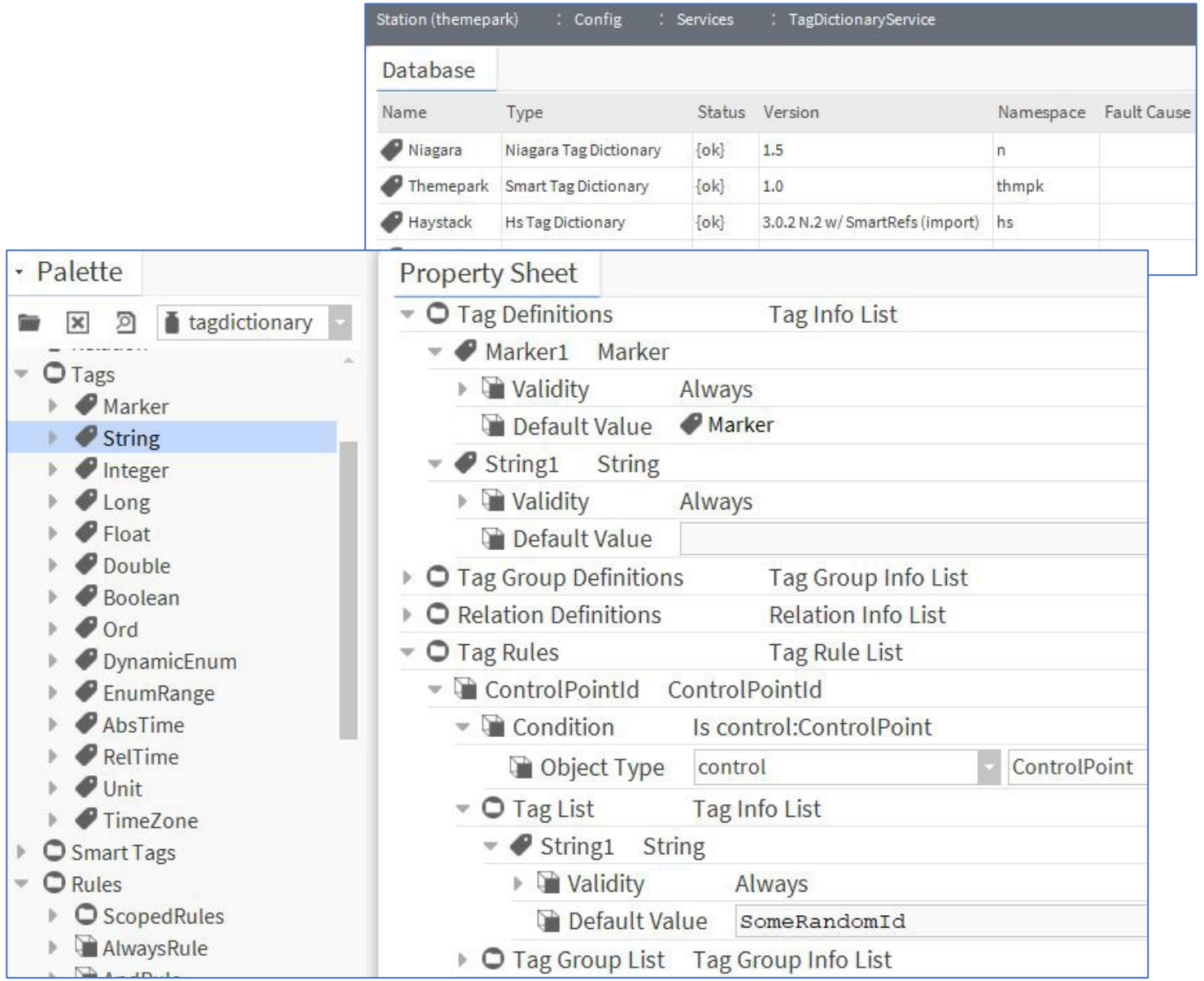
- Semantic information for station entities
- Tag Dictionary elements
  - Tags
  - Relations
  - Tag groups
  - Rules
- Direct or Implied tags and relations
- Uses
  - NEQL
  - Search
  - Tag-based PX bindings
  - Hierarchies
  - Templates



Tag Dictionary

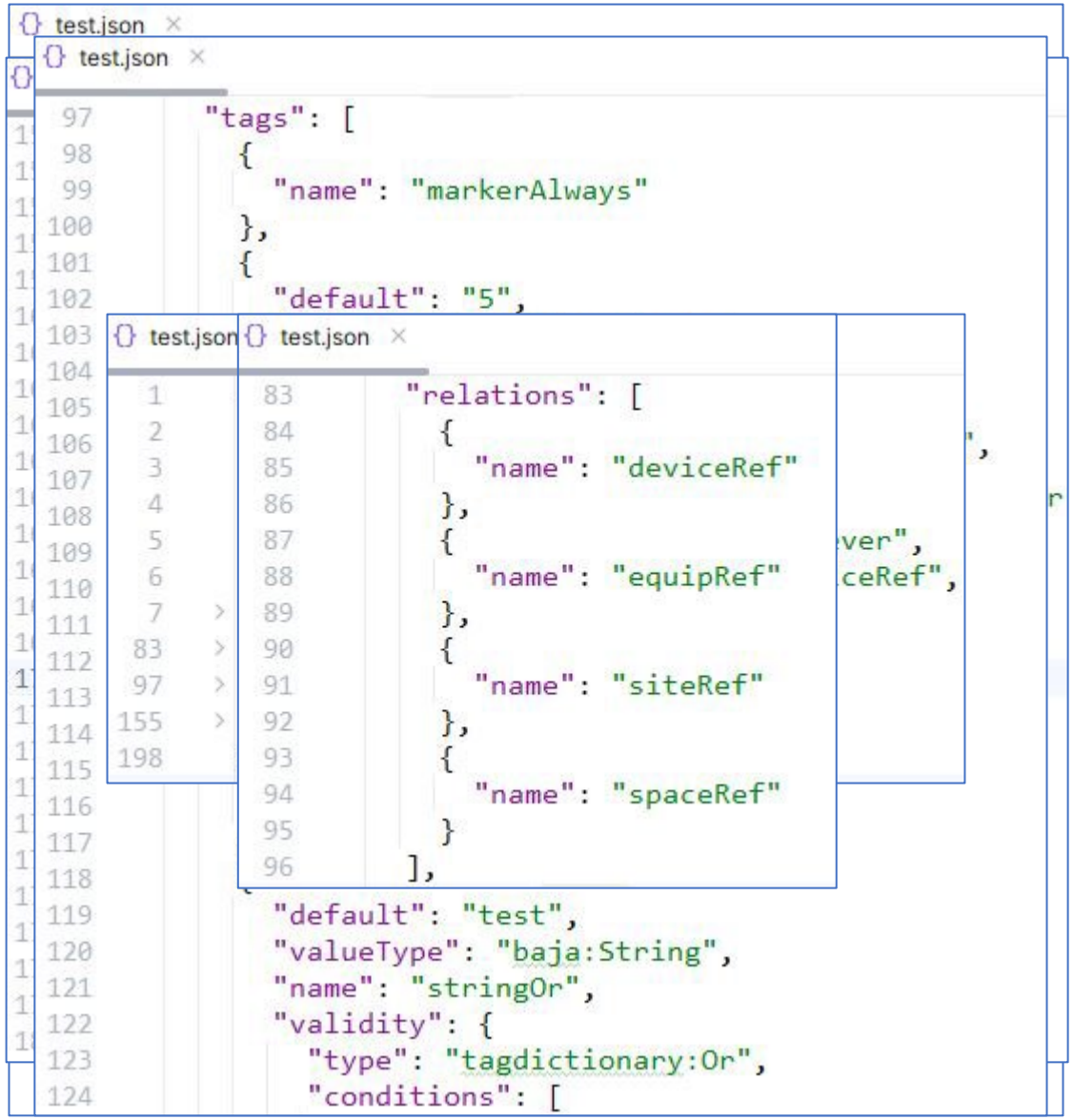
- Tag Dictionary Service
- Tag Dictionary Palette

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

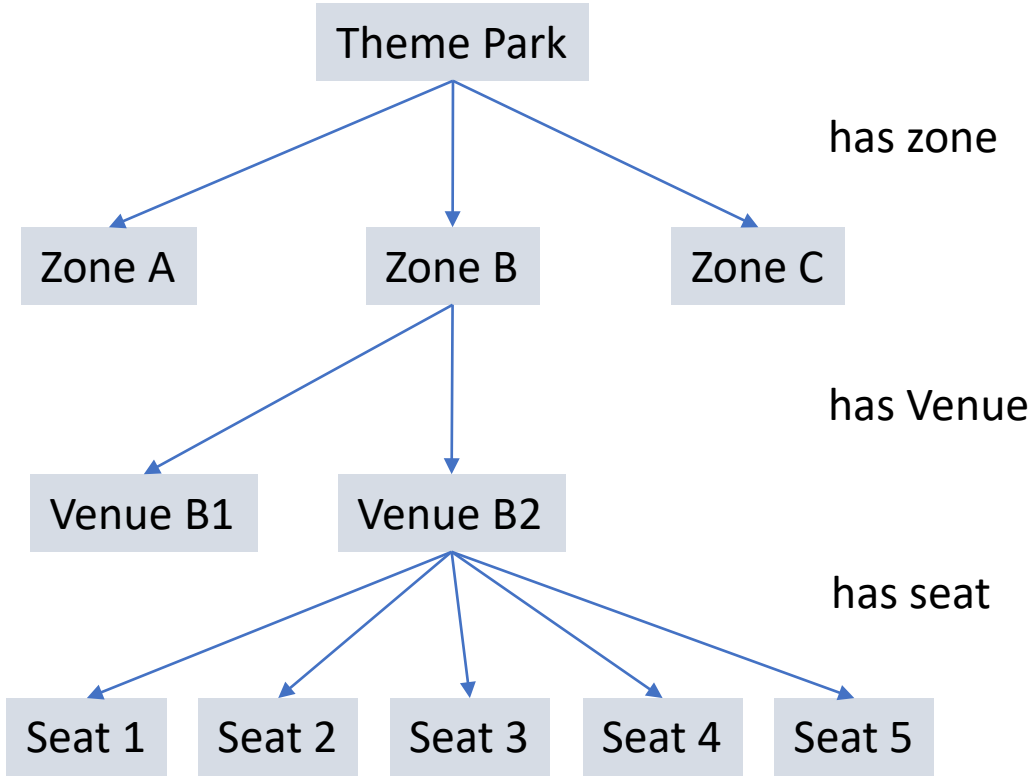


JSON Structure

- 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



Our Dictionary JSON

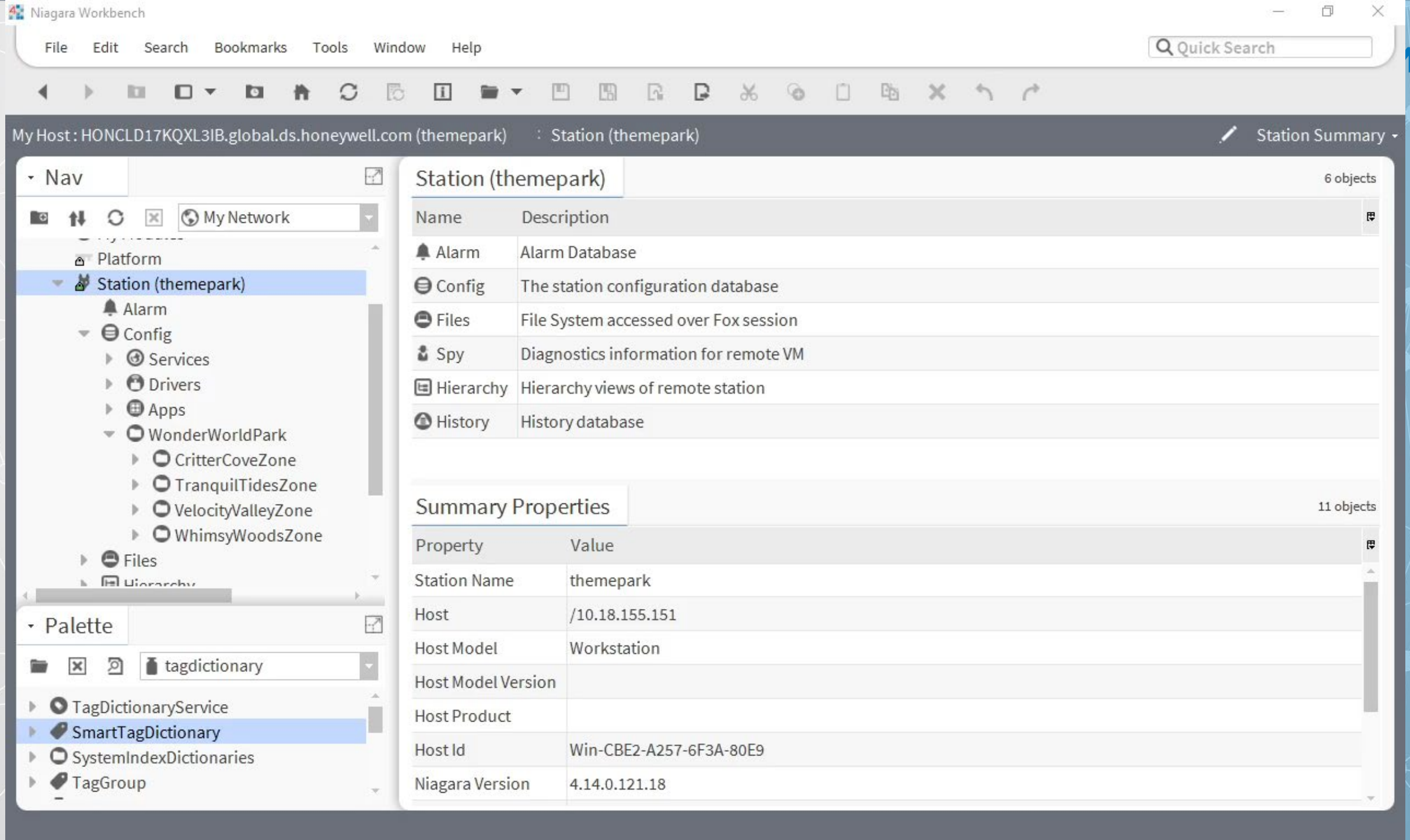


Structure

- {} themeparkTagsRelations.json
  - frozen true
  - namespace "thmpk"
  - neqlizeExcludedRelations
  - neqlizeExcludedTags ""
  - [] relations
    - {} object
    - {} object
    - {} object
  - [] rules
  - [] tagGroups
  - [] tags
    - {} object
    - {} object
    - {} object
    - {} object
    - {} object
    - {} object
    - {} object
  - version "1.0"

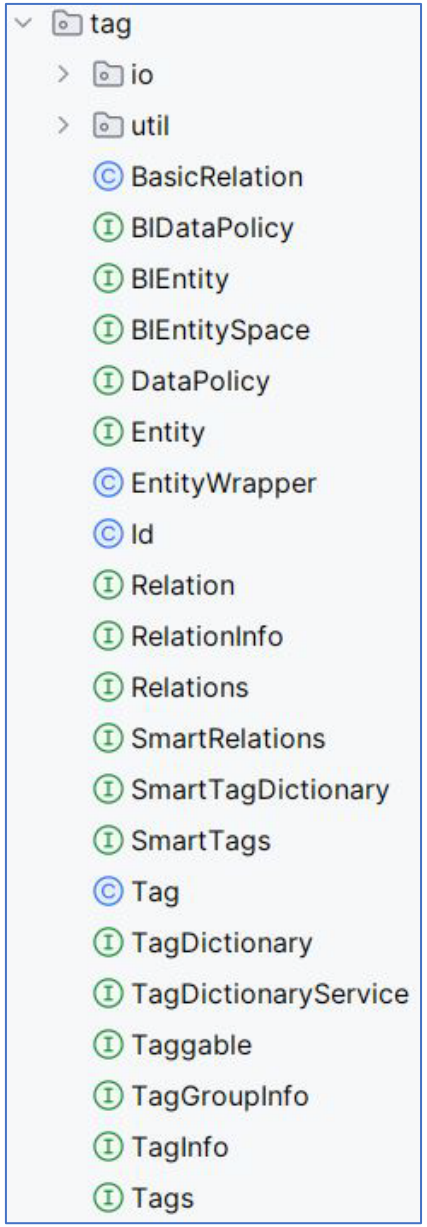
{} themeparkTagsRelations.json

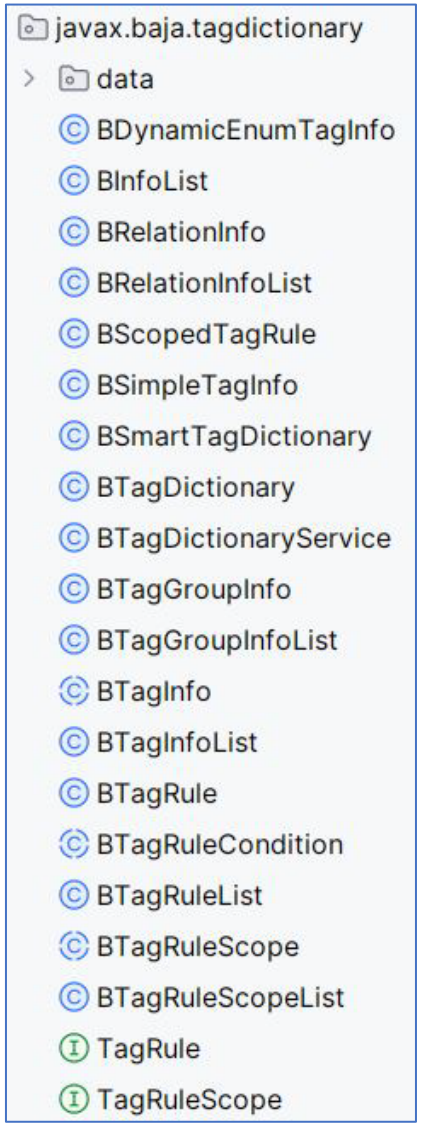
```
{
  "namespace": "thmpk",
  "version": "1.0",
  "frozen": true,
  "neqlizeExcludedTags": "",
  "neqlizeExcludedRelations": "",
  "tags": [
    {"name": "zone"...},
    {"name": "restaurant"...},
    {"name": "ride"...},
    {"name": "shop"...},
    {"name": "show"...},
    {"name": "seat"...},
 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


Niagara Workbench
File
Edit
Search
Bookmarks
Tools
Window
Help
Quick Search
My Host : HONCLD17KQXL3IB.global.ds.honeywell.com (themepark)
Station (themepark)
Station Summary
Nav
My Network
Platform
Station (themepark)
Alarm
Config
Services
Drivers
Apps
WonderWorldPark
CritterCoveZone
TranquilTidesZone
VelocityValleyZone
WhimsyWoodsZone
Files
Palette
tagdictionary
TagDictionaryService
SmartTagDictionary
SystemIndexDictionaries
TagGroup
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





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

```
@NiagaraType
@NiagaraProperty(
  name = "validity",
  type = "BTagRuleCondition",
  defaultValue = "new BIsTypeCondition(BControlPoint.TYPE)",
  override = true
)
public class BCurValTag extends BTagInfo
```

BCurValTag.java

```
@Override
public Tag getTag(Entity entity)
{
  if (entity instanceof BControlPoint)
  {
    BStatusValue statusValue =
      ((BControlPoint)entity).getOutStatusValue();
    return makeTagForStatusValue(getTagId(), statusValue);
  }

  return null;
}

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

Our Tags

- BIdTag

```
  @Override
  public BIDataValue 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

```
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
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85    }
```

BHaystack4SiteRelation.java

```
138     @Override
139     public void addRelations(Entity source, Collection<Relation> relations)
140     {
141       // Add the outbound relation
142       Optional<Relation> outbound = getRelation(source);
143       outbound.ifPresent(relations::add);
144
145       // Add inbound siteRef relations if the given entity has the site tag.
146       if (!(source instanceof BComponent) || !hasSiteTag(source))
147       {
148         return;
149       }
150
151       BComponent site = (BComponent) source;
152       for (Relation siteRef : new ComponentRelations(site)
153         .getAll(getSiteRefId(), Relations.IN))
154       {
155         addSiteRelations(siteRef, relations);
156       }
157     }
```

Our Relation

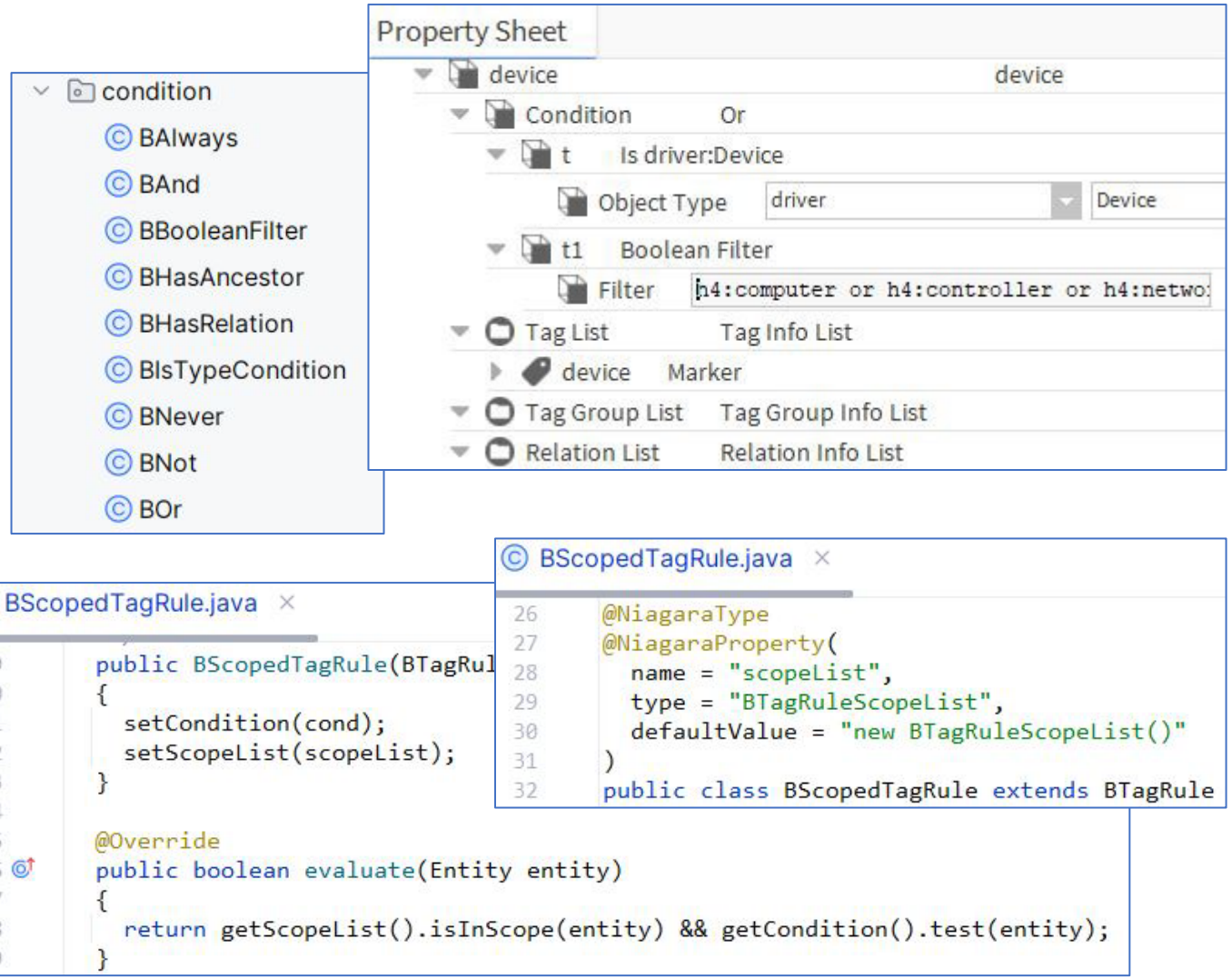
- BVenueSeatRelation

```java
@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`



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

```
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

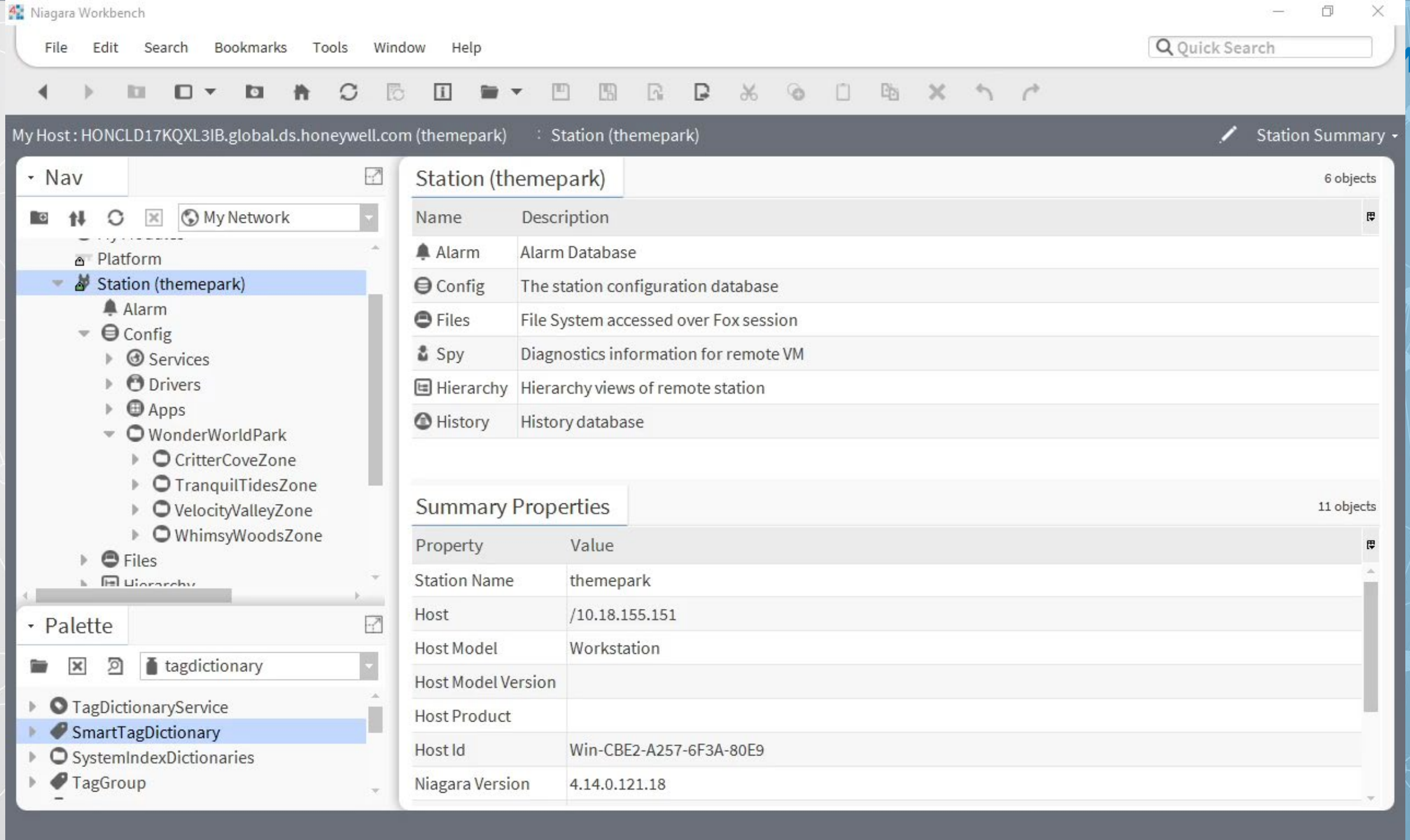
```
<?xml version="1.0" encoding="UTF-8"?>
<bajaObjectGraph version="1.0">
<p m="b=baja" t="b:UnrestrictedFolder">
 <p n="Brick" m="td=tagdictionary" t="td:SmartTagDictionary">
  <p n="namespace" v="bk"/>
  <p n="importDictionaryOrd" t="b:Ord"
    v="module://brick/com/tridium/brick/data/brick.json"/>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

haystack-rt\module.palette

```
<?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 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>
```

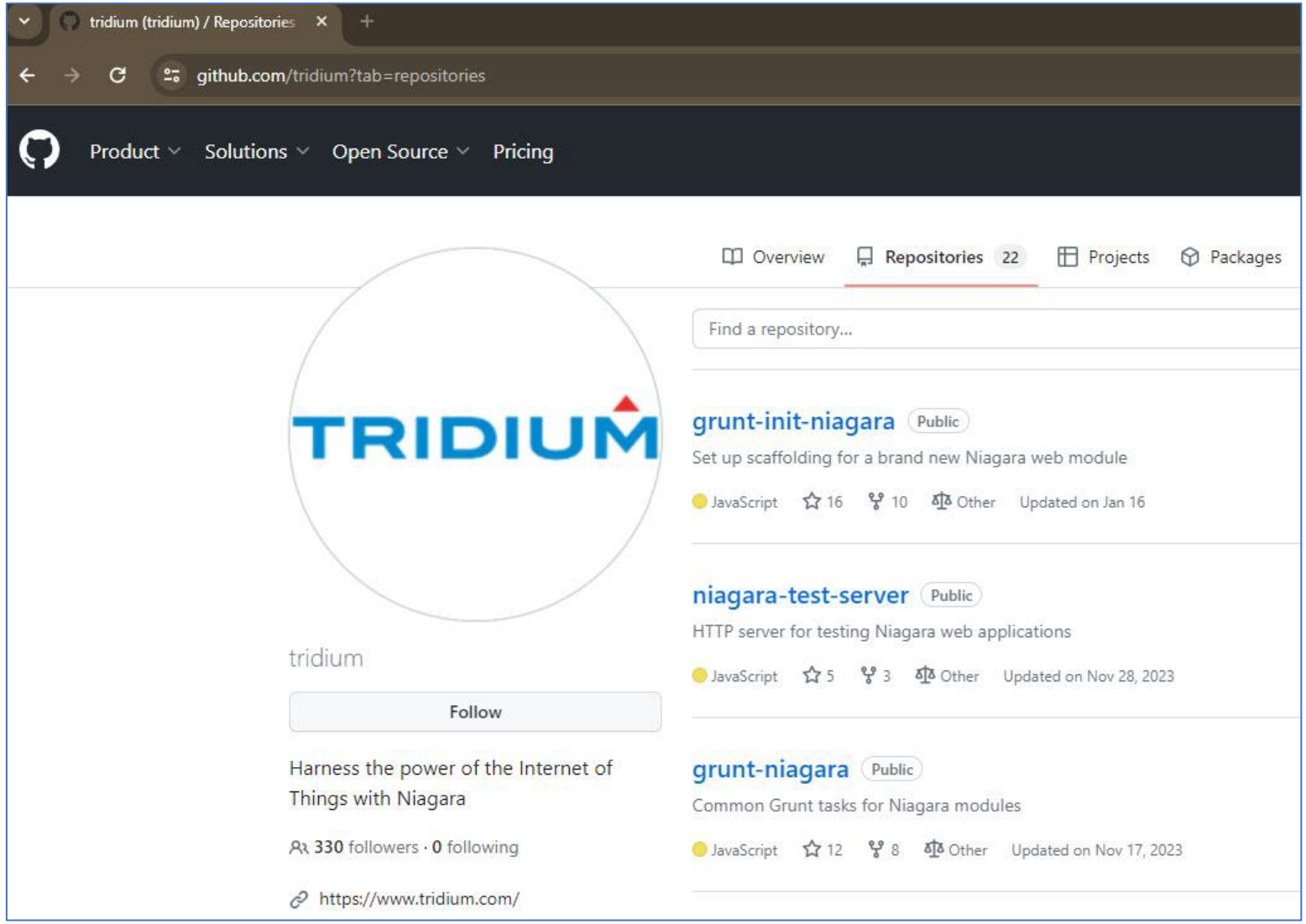
Our Module

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


Niagara Workbench
File
Edit
Search
Bookmarks
Tools
Window
Help
Quick Search
My Host : HONCLD17KQXL3IB.global.ds.honeywell.com (themepark)
Station (themepark)
Station Summary
Nav
My Network
Platform
Station (themepark)
Alarm
Config
Services
Drivers
Apps
WonderWorldPark
CritterCoveZone
TranquilTidesZone
VelocityValleyZone
WhimsyWoodsZone
Files
Palette
tagdictionary
TagDictionaryService
SmartTagDictionary
SystemIndexDictionaries
TagGroup
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



TRIDIUM