
TRIDIUM
NS2024
APRIL 15 - 17 | ANAHEIM, CA

Questions from the Community

NS2024
APRIL 15 - 17 | ANAHEIM, CA

TRIDIUM

TRIDIUM

I find that I struggle with testing Java modules, the process of compiling, restarting workbench, restarting the station to finally test is quite limiting and slow, so I would like to hear about possible improvements or best practices in this regard.

Testing

- Unit tests!
- niagara_home/bin/test.exe
  - Just run "test.exe" to see all options
  - Use watch mode
- gradlew –t
  - Continuous build
- BTestNg
- BTestNgStation (4.14)
- Doc Developer:
  - Developer Guide -> Framework -> Test

New Theme development. What are the DIVs that the system expects and can be changed?

In other words, what are the basic steps needed to create a new theme and where can I find detailed information?

Themes

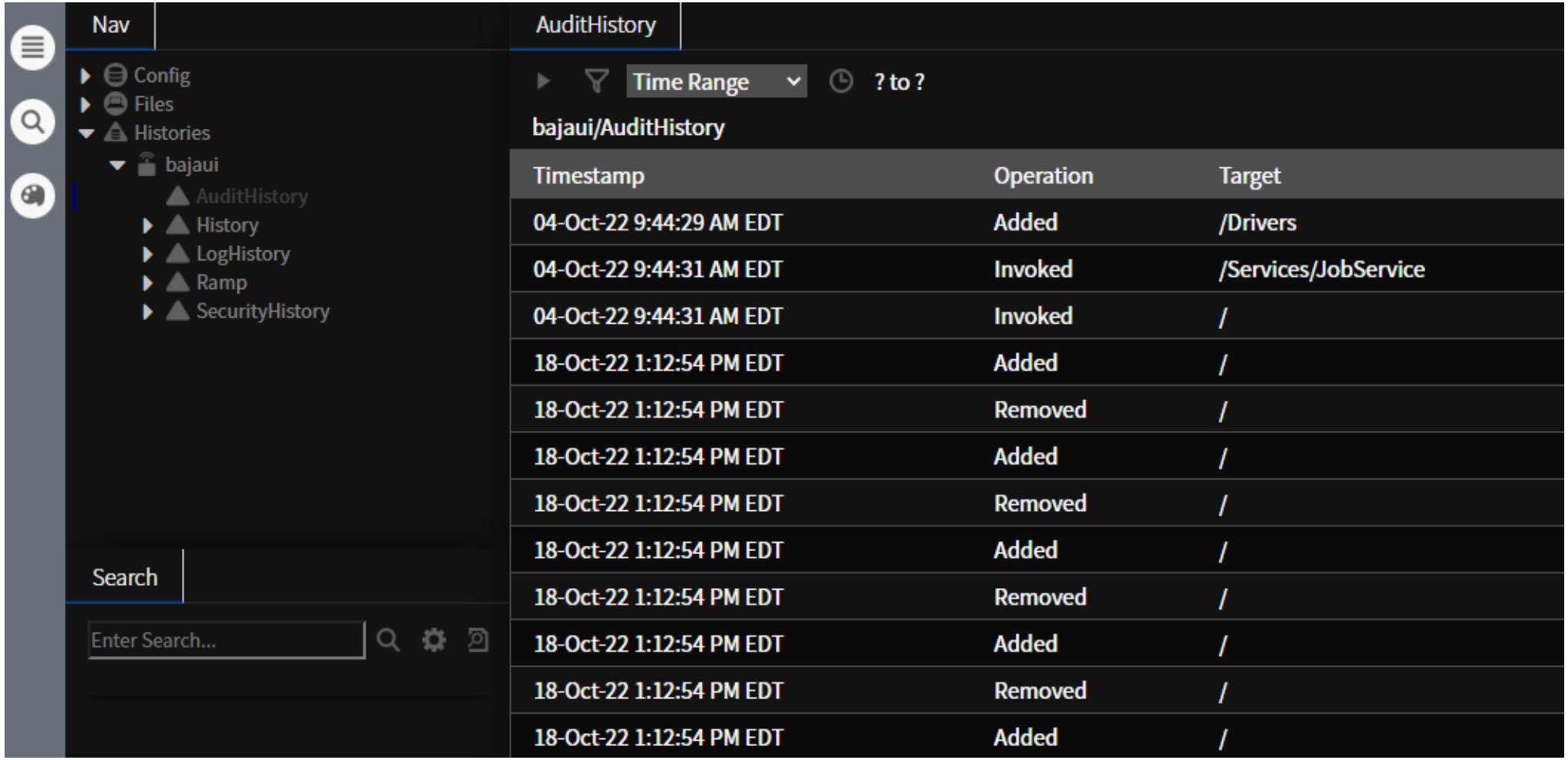
- Niagara Developer documentation is the best place to start.
  - module://docDeveloper/doc/themes/themeCreation.html
- Start by making a copy of one of our existing themes:

  themeZebra or themeLucid.
- /src/nss/theme.nss - themes the Workbench UI.
- /src/ux/theme.css - themes the Web/Browser UI.

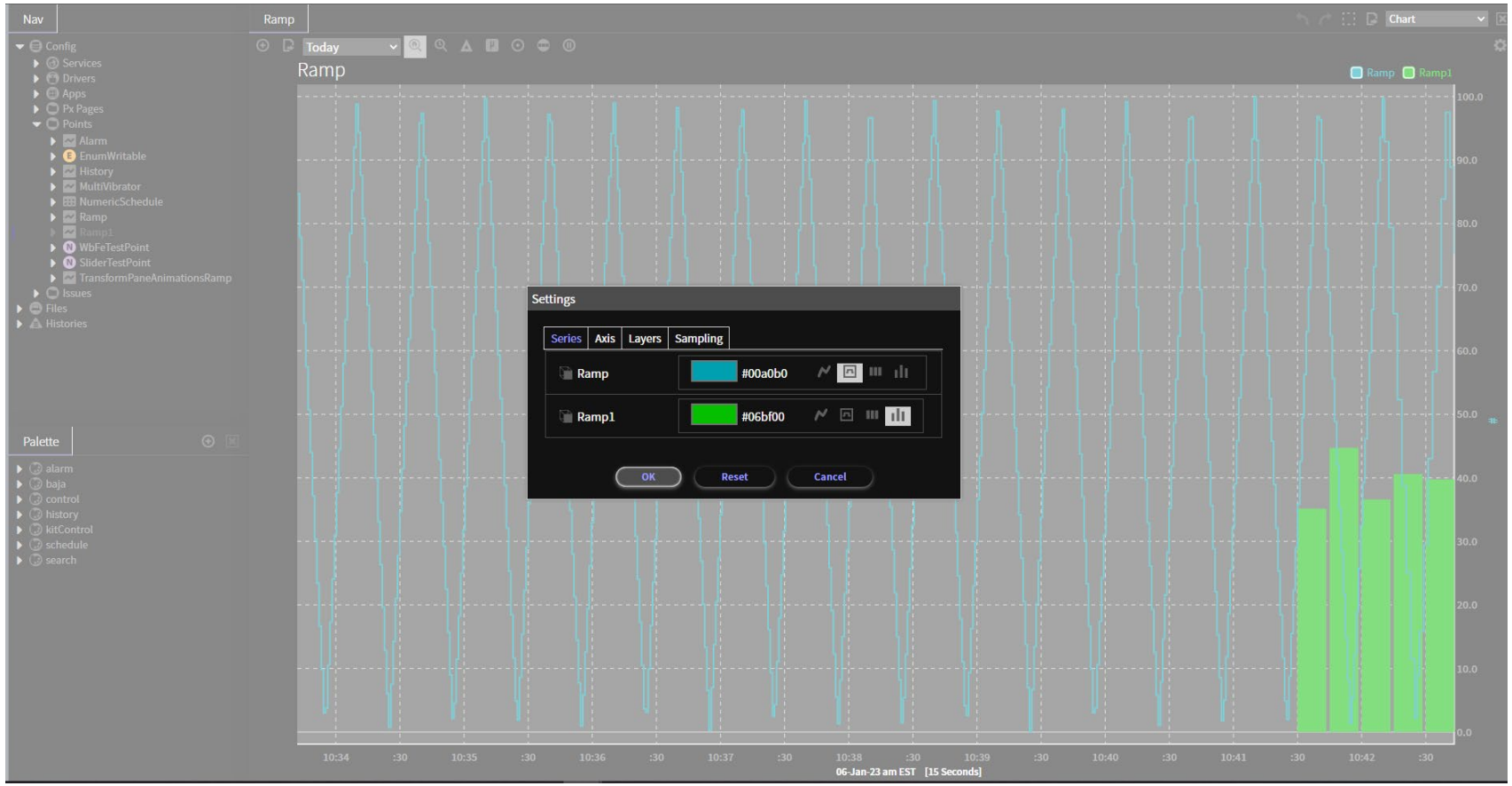
Themes (Cont.)

- Customize colors using /src/less/palette.less.
- Customize fonts using /src/lexx/fonts.less.
- Override your 'fonts' and 'icons.
  - /src/fonts and /src/imageOverrides
- Don' forget to theme your login screen to match
  - module://docDeveloper/doc/loginScreen.html

Example Dark Theme



Example Dark Theme (Cont.)



TRIDIUM

How can I programmatically add tags and relations to my station?

Tags & Relations

Adding a Tag

- **Using Entity API**
  - Tag floorTag = Tag.newTag("b:floor", BInteger.make(1));
  - entity.tags().set(floorTag);
    - // or
  - entity.tags().set(Id.newId("b:floor"), BInteger.make(1));

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

Adding a Relation

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

- **Using Component Model**
  - BRelation r = myComponent.makeRelation(Id.newId("hs:ahuRef"), targetEndpoint, cx);

TRIDIUM

In my code, I'd like to be able to know or show the current user.

Current User

Context.getUser()

- Standard Niagara callbacks like changed(), RPC calls
- May not always contain the logged in user

BUser.getCurrentAuthenticatedUser()

- Cannot contain a user other than the authenticated user
- doPrivileged calls may cause it to return null

%user()%

- Used to retrieve the current user in BFormats

What form does the BOrd value take for the setAddress() method of BNiagaraStation? Can you go over some of the common OrdSchemes and tell me how and when to use them?

BNiagaraStation.setAddress()

```
BNiagaraStation targetStation = new BNiagaraStation();
targetStation.setAddress(BOrd.make("ip:192.168.20.24"));
targetStation.setAddress(BOrd.make("ip:station.ns2024.lan"));
```

Common OrdSchemes

Resource	OrdScheme	Example
Component	slot: h:	slot:/Services/WebService local:|station:|**h:f4**
File	file:	file:^NS2024-Demo.px
IP Address	ip:	**ip:172.31.67.241**|foxs:|station:|slot:/Services/WebService
Local Host	local:	**local:**|foxs:|station:|slot:/Services/FoxService
Alarm Database	alarm:	local:|foxs:|station:|**alarm:**|bql:select * from openAlarms
History Database	history:	history:/myStation/myHistory?period=lastWeek
NEQL Query	neql:	ip:172.31.67.241|foxs:|station:|**neql:n:type='baja:Folder'**
BQL Query	bql:	local:|foxs:|station:|history:|**bql:select * from sys.histories where recordType='history:BooleanTrendRecord'**

I’d like to use BQL in my code to find all the points in my station that are in fault. How can I do this? Can I connect to another station and run this query?

BQL

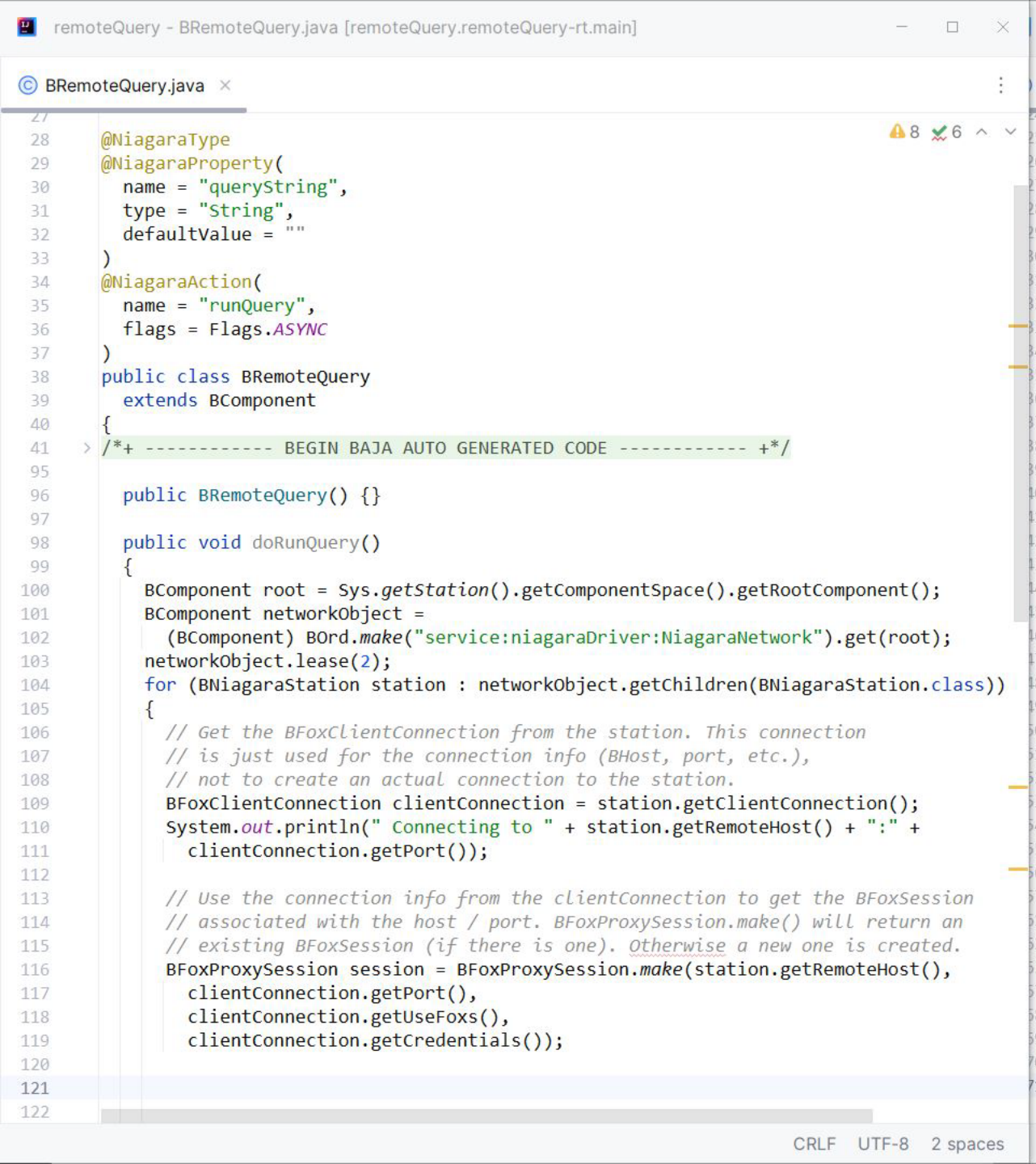
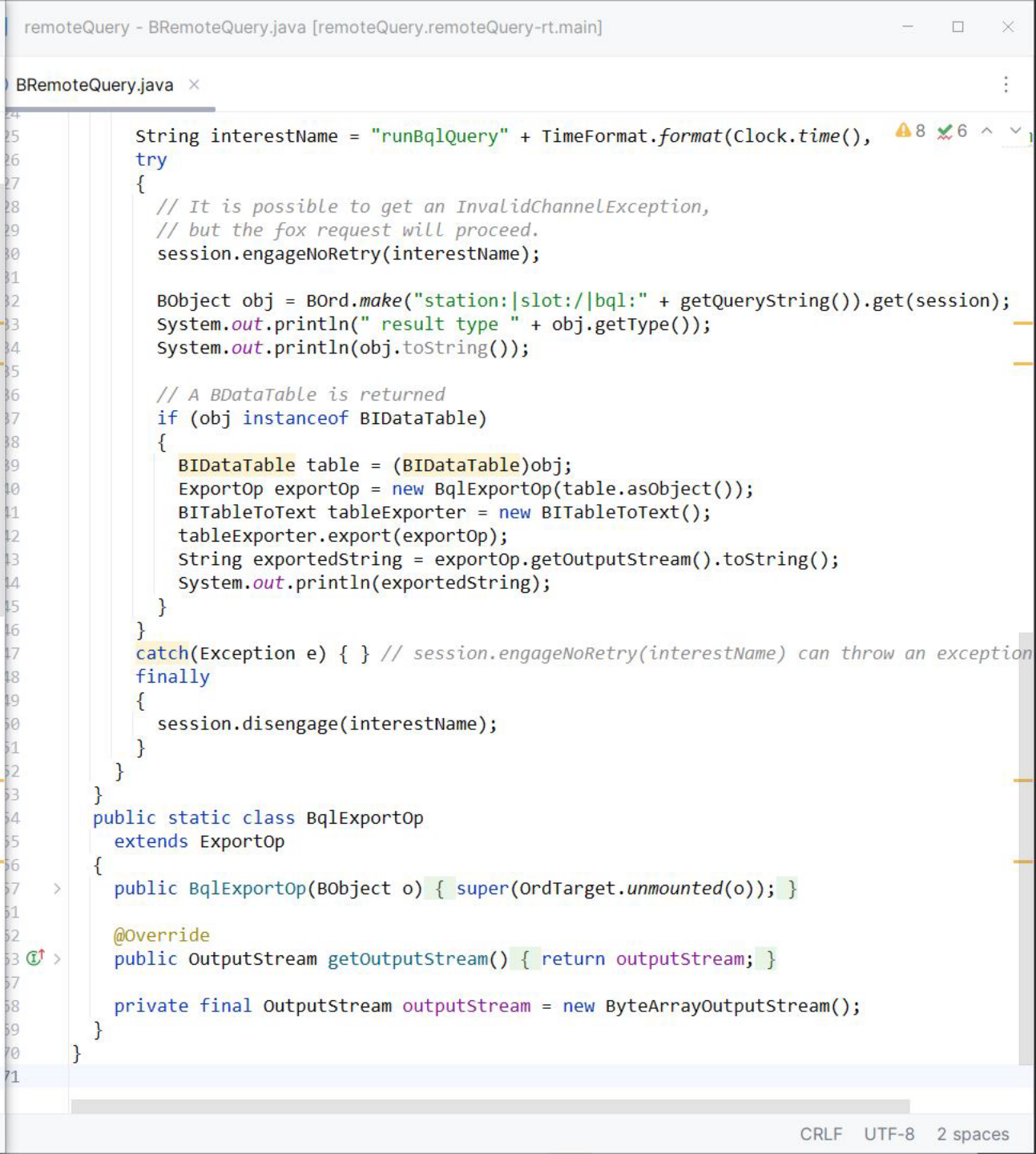
bql:select * from control:ControlPoint where status.isFault

Component Action

- **Get BNiagaraStation list from Niagara Network**
  - BFoxClientConnection
  - BFoxProxySession
  - BOrd.make(<query string>).get(session)
  - Query returns BIDataTable

Provisioning Job Step

- Step calls doRun() for each device
  - ProvisioningConnectionUtil
  - BFoxSession = util.getEngagedSession()
  - BOrd.make(<query string>).get(session)
  - Output to BDeviceStepDetails


remoteQuery - BRemoteQuery.java [remoteQuery.remoteQuery-rt.main]
BRemoteQuery.java
@NiagaraType
@NiagaraProperty(
name = "queryString",
type = "String",
defaultValue = ""
)
@NiagaraAction(
name = "runQuery",
flags = Flags.ASYNC
)
public class BRemoteQuery
extends BComponent
{
/*+ ------------ BEGIN BAJA AUTO GENERATED CODE ------------ +*/
public BRemoteQuery() {}
public void doRunQuery()
{
BComponent root = Sys.getStation().getComponentSpace().getRootComponent();
BComponent networkObject =
(BComponent) BOrd.make("service:niagaraDriver:NiagaraNetwork").get(root);
networkObject.lease(2);
for (BNiagaraStation station : networkObject.getChildren(BNiagaraStation.class))
{
// Get the BFoxClientConnection from the station. This connection
// is just used for the connection info (BHost, port, etc.),
// not to create an actual connection to the station.
BFoxClientConnection clientConnection = station.getClientConnection();
System.out.println(" Connecting to " + station.getRemoteHost() + ":" +
clientConnection.getPort());
// Use the connection info from the clientConnection to get the BFoxSession
// associated with the host / port. BFoxProxySession.make() will return an
// existing BFoxSession (if there is one). Otherwise a new one is created.
BFoxProxySession session = BFoxProxySession.make(station.getRemoteHost(),
clientConnection.getPort(),
clientConnection.getUseFoxs(),
clientConnection.getCredentials());
CRLF UTF-8 2 spaces

remoteQuery - BRemoteQuery.java [remoteQuery.remoteQuery-rt.main]
BRemoteQuery.java
String interestName = "runBqlQuery" + TimeFormat.format(Clock.time(),
try
{
// It is possible to get an InvalidChannelException,
// but the fox request will proceed.
session.engageNoRetry(interestName);
BObject obj = BOrd.make("station:|slot:/|bql:" + getQueryString()).get(session);
System.out.println(" result type " + obj.getType());
System.out.println(obj.toString());
// A BDataTable is returned
if (obj instanceof BIDataTable)
{
BIDataTable table = (BIDataTable)obj;
ExportOp exportOp = new BqlExportOp(table.asObject());
BITableToText tableExporter = new BITableToText();
tableExporter.export(exportOp);
String exportedString = exportOp.getOutputStream().toString();
System.out.println(exportedString);
}
}
catch(Exception e) { } // session.engageNoRetry(interestName) can throw an exception
finally
{
session.disengage(interestName);
}
}
public static class BqlExportOp
extends ExportOp
{
public BqlExportOp(BObject o) { super(OrdTarget.unmounted(o)); }
@Override
public OutputStream getOutputStream() { return outputStream; }
private final OutputStream outputStream = new ByteArrayOutputStream();
}
}
CRLF UTF-8 2 spaces

remoteQuery - BRemoteQueryStep.java [remoteQuery.remoteQuery-wb.main]

BRemoteQueryStep.java

```
@NiagaraType
@NiagaraProperty(
  name = "queryString",
  type = "String",
  defaultValue = ""
)
public class BRemoteQueryStep
  extends BDeviceJobStep
{
/*+ ------------ BEGIN BAJA AUTO GENERATED CODE ------------ +*/

  public BRemoteQueryStep() {}

  @Override
  protected void doRun(BBatchJobService batchJobService, BDeviceStepDetails details,
    BDevice device, DeviceNetworkJobOp deviceNetworkJobOp)
    throws Exception
  {
    details.message("remoteQuery", "RemoteQueryStep.start", new String[] {device.getName()});

    try (ProvisioningConnectionUtil util = new ProvisioningConnectionUtil(device, details))
    {
      BFoxSession session = util.getEngagedFoxSession();
      try
      {
        BObject obj = BOrd.make("station:|slot:/|bql:" + getQueryString()).get(session);
        details.message("remoteQuery", "RemoteQueryStep.resultType", new String[] {device.getNam

        // We get back a BDataTable
        if (obj instanceof BIDataTable)
        {
          BIDataTable table = (BIDataTable) obj;
          ExportOp exportOp = new BqlExportOp(table.asObject());
          BITableToText tableExporter = new BITableToText();
          tableExporter.export(exportOp);
          String exportedString = exportOp.getOutputStream().toString();
          details.message("remoteQuery", "RemoteQueryStep.result", new String[] {device.getName(
        }

        details.success();
        details.complete(BJobState.success);
      }
```

CRLF UTF-8 2 spaces

Query - BRemoteQueryStep.java [remoteQuery.remoteQuery-wb.main]

QueryStep.java

```
      catch (Exception e)
      {
        details.failed("remoteQuery", "RemoteQueryStep.bqlFailed", new String[] {de
        details.complete(BJobState.failed);
      }
    }
    catch (Exception e)
    {
      details.failed("remoteQuery", "RemoteQueryStep.connectionFailed", new String[
      details.complete(BJobState.failed);
    }
  }

  @Override
  public String toString(Context cx)
  {
    return lexicon.getText("RemoteQueryStep.display", getQueryString());
  }

  public static class BqlExportOp
    extends ExportOp
  {
    public BqlExportOp(BObject o) { super(OrdTarget.unmounted(o)); }

    @Override
    public OutputStream getOutputStream() { return outputStream; }

    private final OutputStream outputStream = new ByteArrayOutputStream();
  }

  public static final Lexicon lexicon = Lexicon.make("remoteQuery");
}
```

CRLF UTF-8 2 spaces

Notes Give Feedback to Microsoft

remoteQuery - BRemoteQueryFactory.java [remoteQuery.remoteQuery-wb.main]

BRemoteQueryFactory.java

```java
@NiagaraType(agent = @AgentOn(types={"remoteQuery:RemoteQueryStep"}))
@NiagaraSingleton
public class BRemoteQueryFactory
  extends BStationStepFactory
{
/*+ ------------ BEGIN BAJA AUTO GENERATED CODE ------------ +*/

  @Override
  public BJobStep makeStep(BWidget owner, BBatchJob batchJob, BObject jobTarget,
    BObject source, Context context)
    throws Exception
  {
    String s = BDialog.prompt(
      owner,
      lexicon.get("RemoteQueryFactory.displayName"),
      "",
      100
    );
    if (s == null || s.length() == 0)
    {
      return null;
    }
    BRemoteQueryStep step = new BRemoteQueryStep();
    step.setQueryString(s);
    return step;
  }

  public static final Lexicon lexicon = Lexicon.make("remoteQuery");
}
```

remoteQuery - module-permissions.xml [remoteQuery.remoteQuery-wb]

module-permissions.xml

```xml
<permissions>
  <java-permissions type="all">
    <java-permission action="read" class="java.util.PropertyPermission" name="*"/>
    <java-permission class="java.util.logging.LoggingPermission" name="control"/>
    <java-permission class="com.tridium.nre.security.NiagaraBasicPermission" name="SET_EXEMP
    <java-permission class="java.lang.RuntimePermission" name="getClassLoader"/>
    <java-permission class="com.tridium.nre.security.KeyStorePermission" name="*" action="al
  </java-permissions>
  <java-permissions type="workbench">
    <!-- Insert any workbench specific permissions here. -->
  </java-permissions>
  <java-permissions type="station">
    <!--<req-permission>-->
    <!--<name>NETWORK_COMMUNICATION</name>-->
    <!--<purposeKey>Outside access for Driver</purposeKey>-->
    <!--<parameters>-->
      <!--<parameter name="hosts" value="127.0.0.1"/>-->
      <!--<parameter name="ports" value="*"/>-->
      <!--<parameter name="type" value="all"/>-->
    <!--</parameters>-->
    <!--</req-permission>-->
  </java-permissions>
</permissions>
```

remoteQuery - module.lexicon [remoteQuery.remoteQuery-rt]

module.lexicon

```
#
# Lexicon for the remoteQuery module.
#
RemoteQueryFactory.displayName=Run BQL Query
RemoteQueryFactory.icon=module://icons/x16/app.png
RemoteQueryFactory.description=Enter a BQL query to run

RemoteQueryStep.display=Run BQL query: {0}.
RemoteQueryStep.start=BQl query started on target device {0}.
RemoteQueryStep.connectionFailed=Fox connection failed for target device {0}.
RemoteQueryStep.bqlFailed=Bql query failed for target device {0}.
RemoteQueryStep.resultType=BQl query result type for target device {0} = {1}
RemoteQueryStep.result=BQl query result for target device {0}\n{1}
```

Niagara Workbench

File Edit Search Bookmarks Tools Window Help

Quick Search

y Host : HONCLD17KQXL3IB.global.ds.honeywell.com (super) : Station (super) : Config

AX Property Sheet

Nav

My Net

Platform

Station (super)

Alarm

Config

Services

Drivers

NiagaraNetwor

Apps

Palette

control

Points

Extensions

Trigger

Application Director

Config

Property Sheet

Config (Station)

Station Name	super
Sys Info	
Services	Service Container
Drivers	Driver Container
Apps	App Container
Query	Folder

Refresh

Save

My code resolves Ords like “slot:/Services/AlarmService”, but when users rename components, my code breaks. How can I fix this?

Alternate Ways to Resolve Ords

- BOrd.make("slot:/Services/AlarmService").resolve(Sys.getStation()).get()
  - This breaks when components are renamed

- Sys.getService(BAlarmService.TYPE)
- BOrd.make("service:alarm:AlarmService").resolve().get()
- BDeviceNetwork implements BIService

- For non-service components the handle ord can be used to protect against the component or a parent in its slot path being renamed

Thank You

NS2024
APRIL 15 - 17 | ANAHEIM, CA

TRIDIUM


TRIDIUM
NS2024
APRIL 15 - 17 | ANAHEIM, CA