
NIAGARA
SUMMIT
2026
SEAMLESS CONNECTIVITY,
POWERFUL INTELLIGENCE
TRIDIUM

This document is a non-binding, confidential document that contains valuable proprietary and confidential information of Tridium and must not be disclosed to any third party without our written agreement. It does not create any binding obligation for us to develop or sell any product, service, or offering. Content provided here may not be altered or modified and must remain in the format originally presented by Tridium. Any descriptions of future product direction, intended updates, or new or improved features or functions are for informational purposes only and are not binding commitments by Tridium. The sale, development, release, and timing of any such products, updates, features, or functions remain in Tridium's sole discretion.

TRIDIUM

Logistics

- Accessibility – please ask the team
- Slides and videos will be published later
- Q&A Session
- **support@tridium.com**


Photography allowed


Fire exit – behind you


Silence Electronic Devices

Niagara 4 Timeline

NS26

TRIDIUM

Introduction to Niagara

TREY WEST
Software Engineer 2 | Tridium

BRIAN TODD
Advanced Software Engineer | Tridium

VIKRAM NAGULAN
Sr Advanced Software Engineer | Tridium

MITHUN TIRUVEDULA
Advanced Software Engineer | Tridium

VIDYA SHIVAMURTHY
Sr Advanced Software Engineer | Tridium

NIAGARA 101

TREY WEST

NIAGARA SUMMIT 2026 DEVELOPER

TRIDIUM

What is Niagara?

NS26

TRIDIUM

Building Automation System

98 VAV Terminal Boxes

- Protocol: BACnet MS/TP

- Space Temperature (°F)
- Setpoint (°F)
- Airflow (cfm)
- Damper Position (%)

Building Automation System

98 VAV Terminal Boxes

4 Lighting Systems

- Protocol: BACnet MS/TP

- Light on/off status (boolean)
- Motion Detector Status (boolean)

Building Automation System

98 VAV Terminal Boxes

4 Lighting Systems

4 Electric Meters

- Protocol: Modbus
- Electricity Usage (W)

Building Automation System

98 VAV Terminal Boxes

4 Lighting Systems

4 Electric Meters

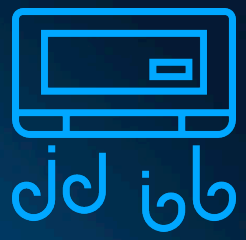
4 Rooftop Units (RTUs)

- Protocol: LON

- Outside Air Temperature (°F)
- Supply Air Temperature (°F)
- Supply Fan Speed (%)
- Static Pressure (in/wc)

Building Automation System

- That comes in at roughly 5,000 points of data across 3 different protocols




Requirements

- Device Interoperability

NS26

TRIDIUM

Requirements

- Device Interoperability
- Control Logic

Requirements

- Device Interoperability
- Control Logic
- User Interface

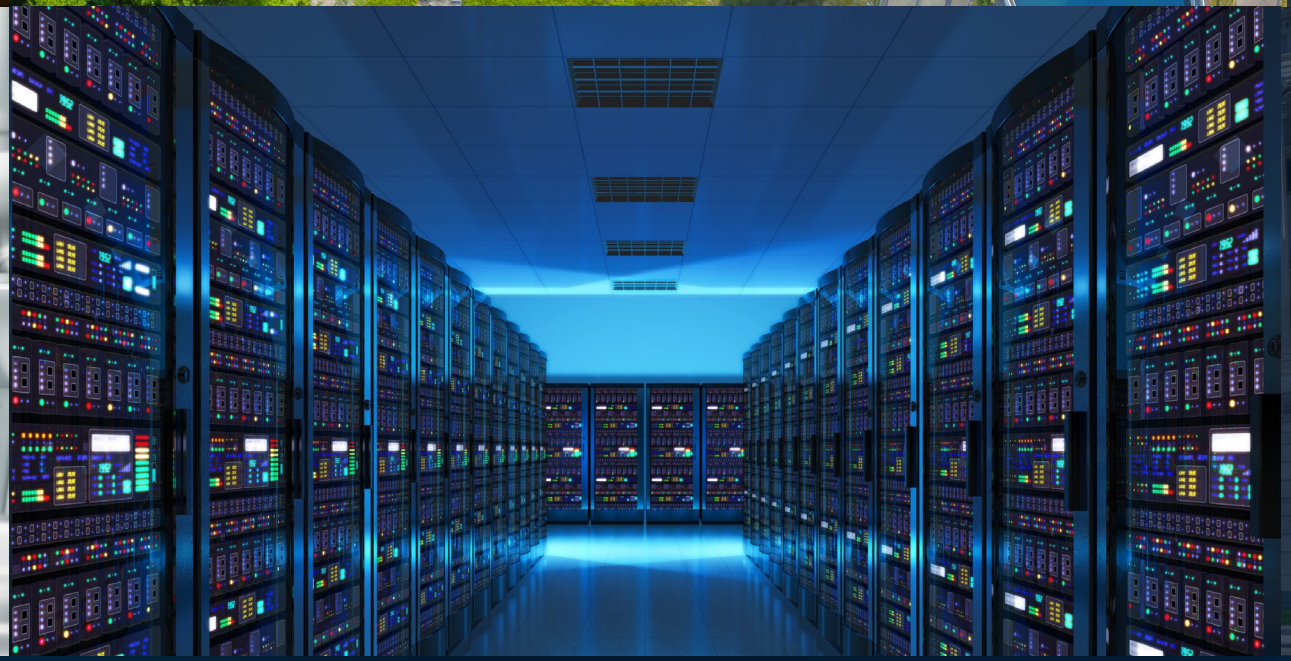
Requirements

- Device Interoperability
- Control Logic
- User Interface
- Remote Access

niagara

NS26

TRIDIUM



How does Niagara work?

Requirements

Device Interoperability
Control Logic
User Interface
Remote Access

Niagara Hosts

- When Niagara is installed on a device, we call that device a **Niagara Host**
- This could be:
  - A server
  - A PC
  - An embedded controller

Java Application Control Engine (JACE)

Niagara Modules

- Niagara features are installed on the host as JAR files
- We call these JAR files **Niagara modules**

Niagara Station

- A Niagara Host contains a Java Runtime Environment (JRE) to run Java code

- We call the process in the host containing the JRE the **station**

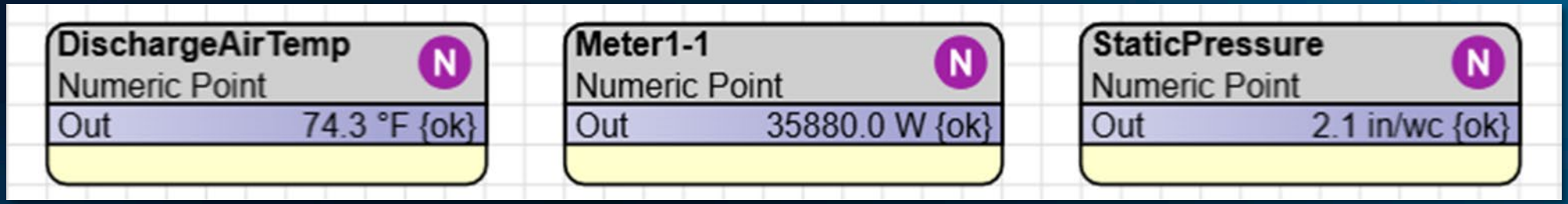
NS26

TRIDIUM

Niagara Drivers

- Drivers are Niagara modules used for communicating to external devices

- Drivers handle protocol-specific details and normalize everything into standard Niagara data types:
  - Numeric
  - String
  - Boolean
  - Enum


DischargeAirTemp
Numeric Point
N
Out 74.3 °F {ok}
Meter1-1
Numeric Point
N
Out 35880.0 W {ok}
StaticPressure
Numeric Point
N
Out 2.1 in/wc {ok}

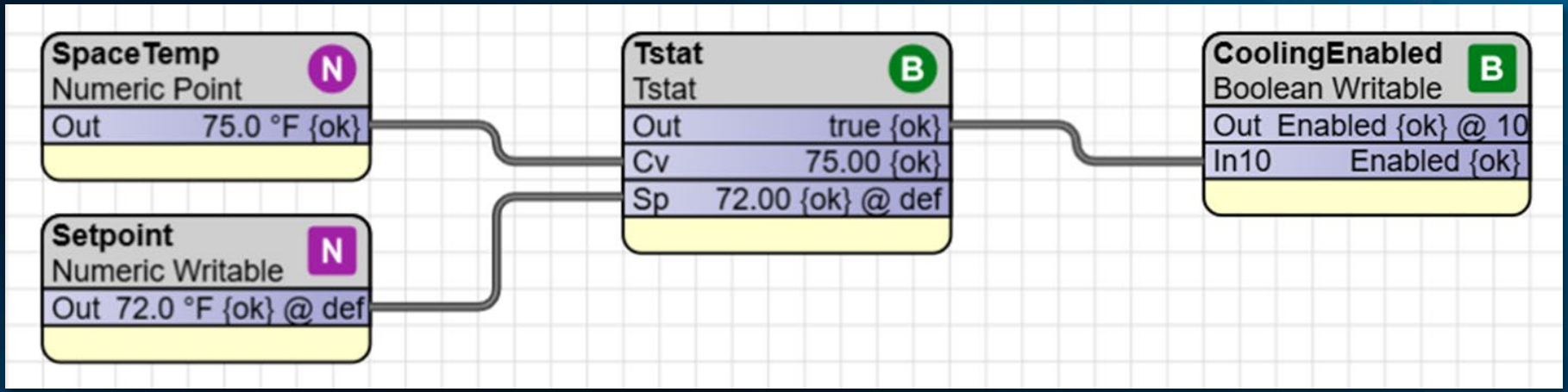
Limitless Device Integrations with Niagara

- **Lighting**
  - Indoor
  - Outdoor
  - Signage
- **HVAC**
  - VRF
  - RTU
  - Cooling Towers
  - Head pump
- **Central Plant**
  - Boiler
  - Chiller
  - ICE Storage
- **Metering**
  - Gas
  - Water
  - Electric
- **Data Center**
  - Data
  - Battery
  - Power
  - Cooling
- **Irrigation**
  - Sprinkler
  - Recovery
  - Weather
- **Safety**
  - Access
  - Cameras
  - Fire
  - Air Quality
- **Sensors**
  - Levels
  - Motion
  - Light
  - Door
  - Audio
  - Motors
  - Contacts
  - Particulate
  - Voltage/Amps
- **Refrigeration**
  - Retail
  - Medical
- **Generation**
  - Solar
  - Diesel
  - Inverters
- **Elevator**
  - Monitor
  - Access

BACnet · BACnet/SC secure connect · Modbus · LonWorks · oBIX Open Building Information Xchange · DALI · KNX · MQTT.org · OPC Foundation

Requirements

- [x] Device Interoperability
- Control Logic
- User Interface
- Remote Access


SpaceTemp
Numeric Point
N
Out 75.0 °F {ok}
Setpoint
Numeric Writable
N
Out 72.0 °F {ok} @ def
Tstat
Tstat
B
Out true {ok}
Cv 75.00 {ok}
Sp 72.00 {ok} @ def
CoolingEnabled
Boolean Writable
B
Out Enabled {ok} @ 10
In10 Enabled {ok}

Requirements

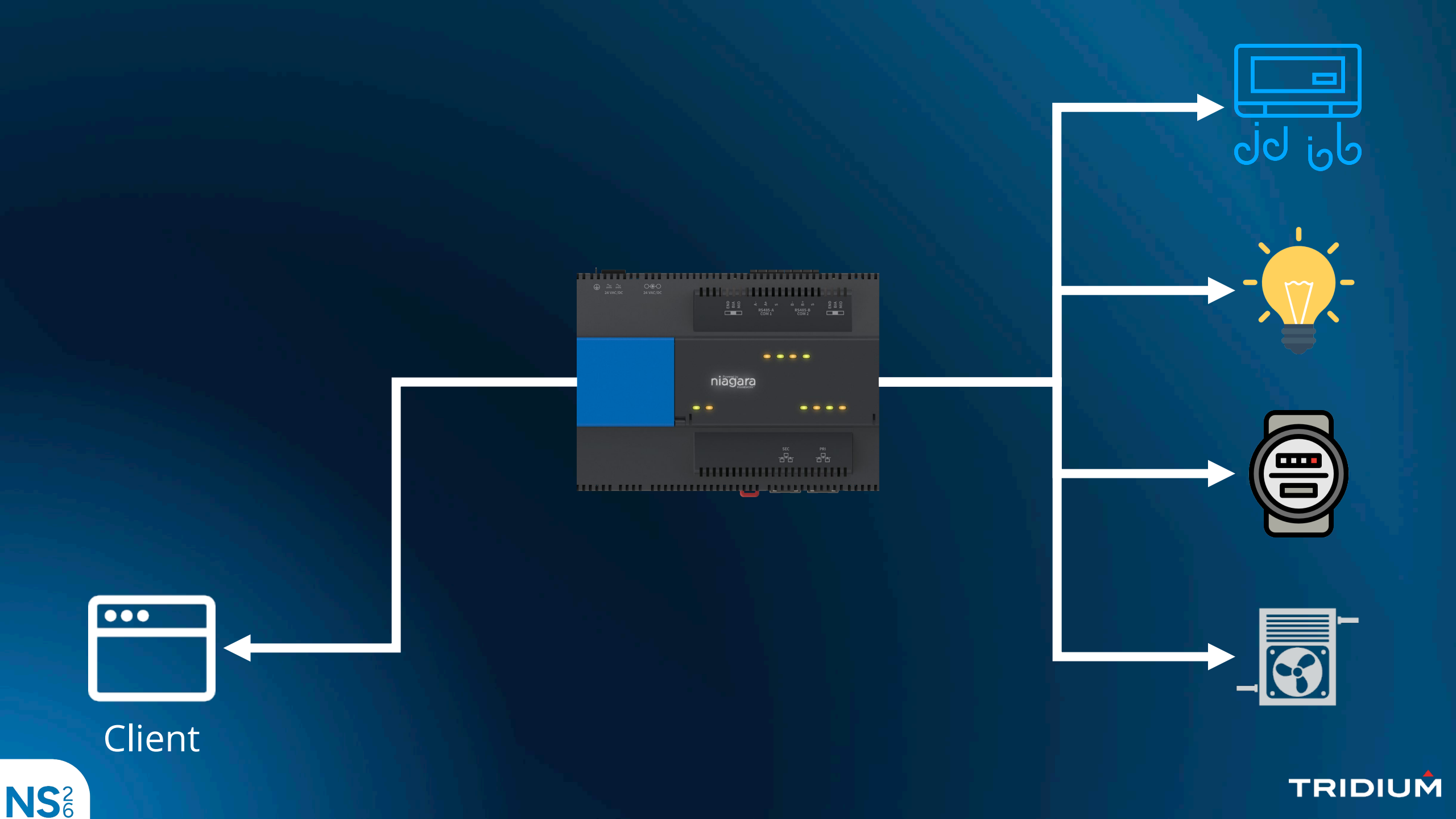
- [x] Device Interoperability
- [x] Control Logic
- [ ] User Interface
- [ ] Remote Access

NS26

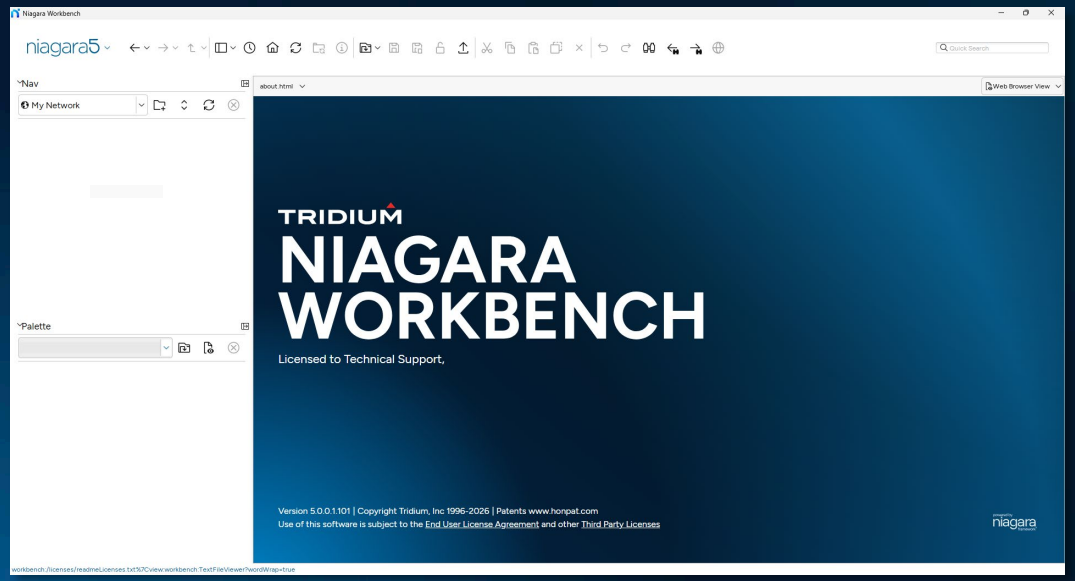
TRIDIUM

Niagara UI

- Niagara UI can be written in Java or JavaScript
- UI code is distributed in Niagara module JAR files


niagara
Client

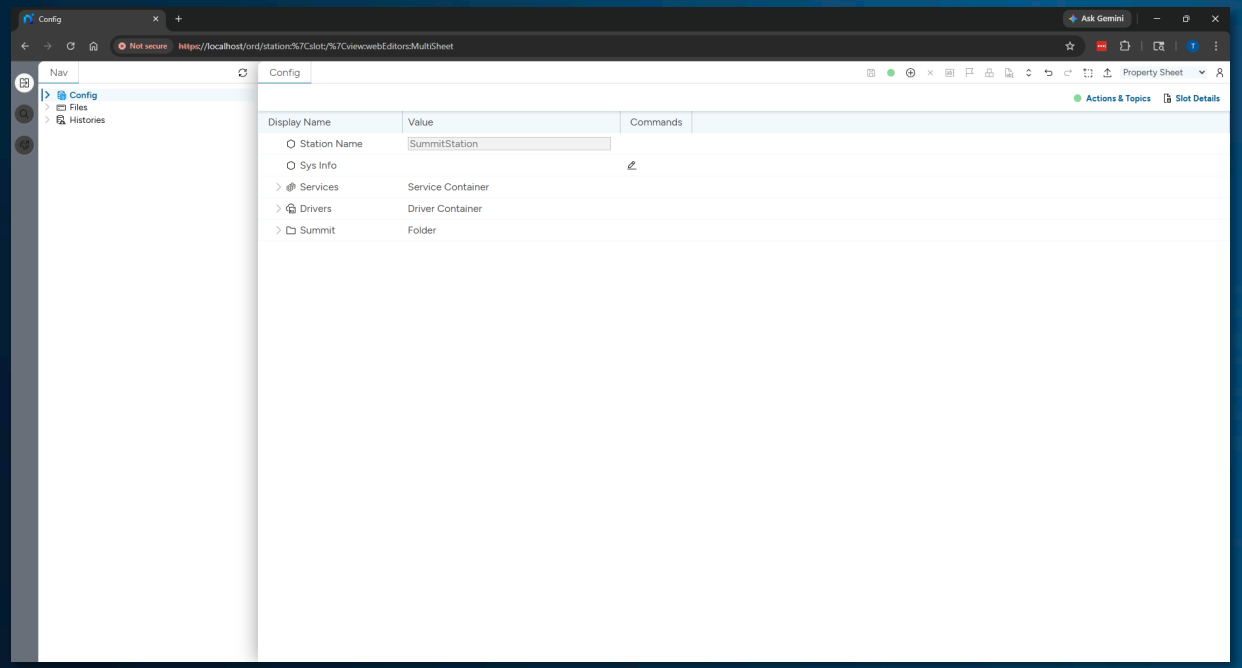
Niagara Workbench



Workbench is a Java-based client that communicates with Niagara Hosts using the foxs protocol

Web Browsers

Web browsers use open web technologies and communicate with Niagara Hosts using HTTPS



Requirements

- ☑ Device Interoperability
- ☑ Control Logic
- ☑ User Interface
- Remote Access

NS26

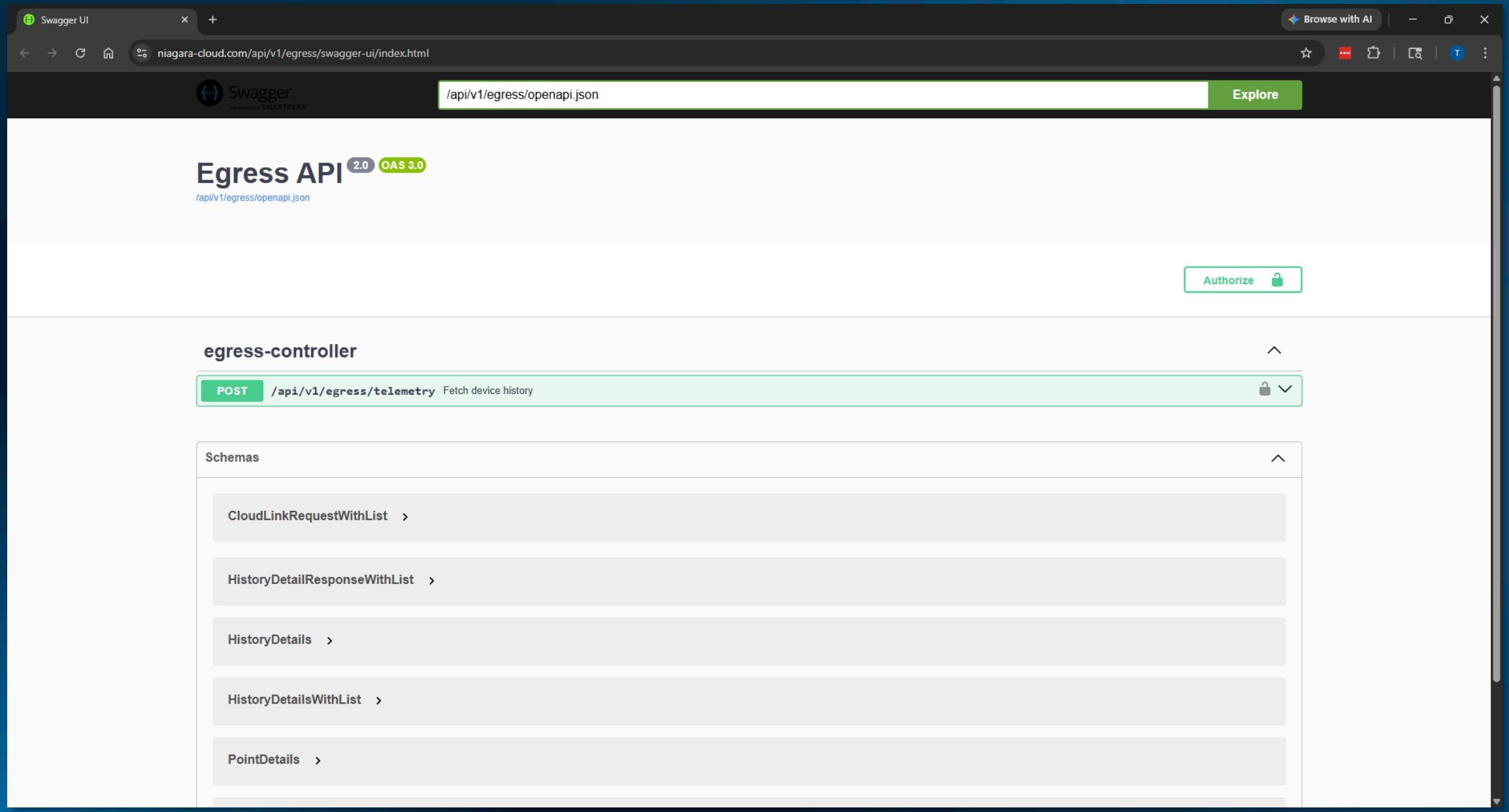
TRIDIUM

Niagara Cloud Suite (NCS)

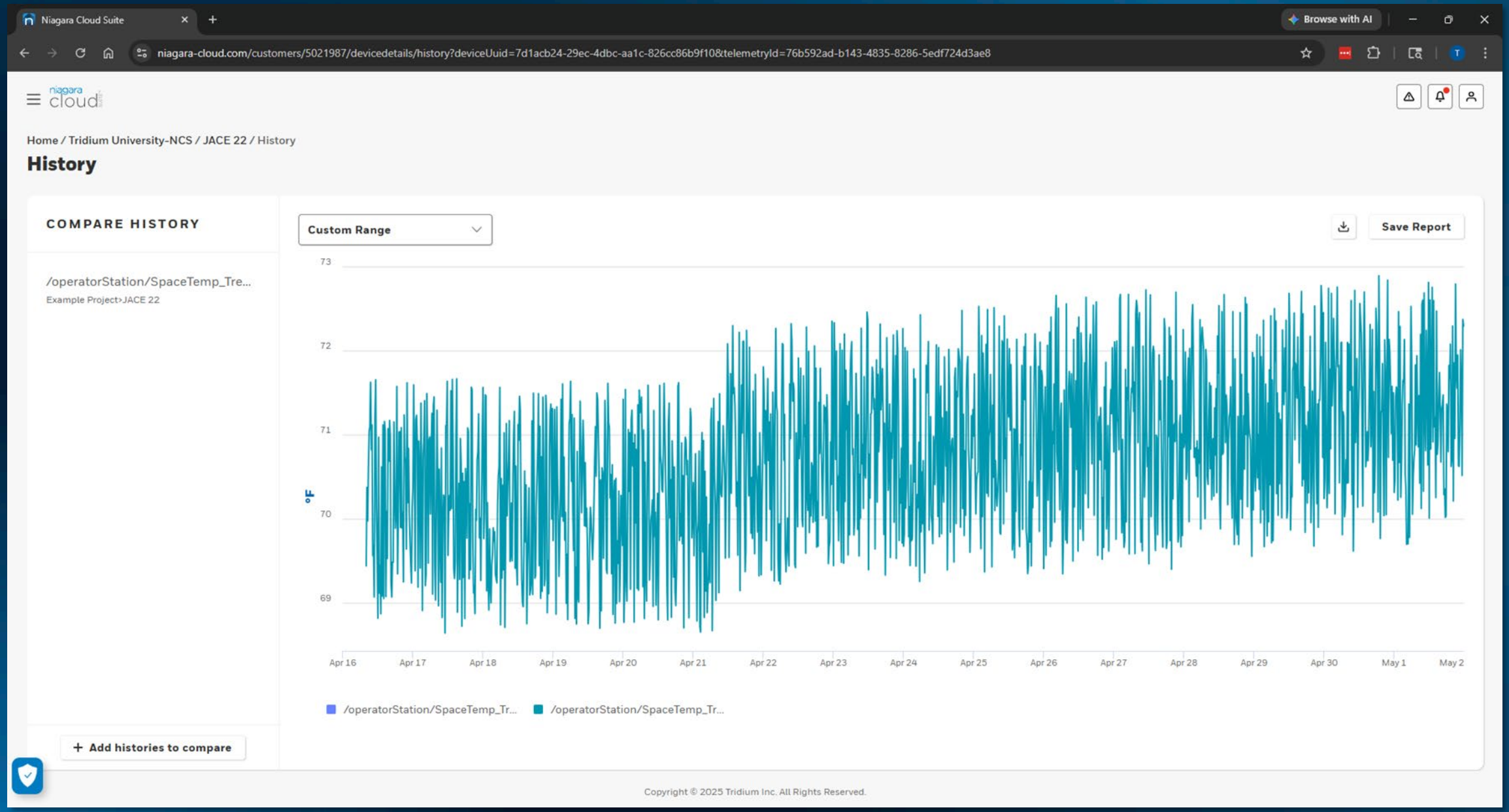
Client

NS26

TRIDIUM



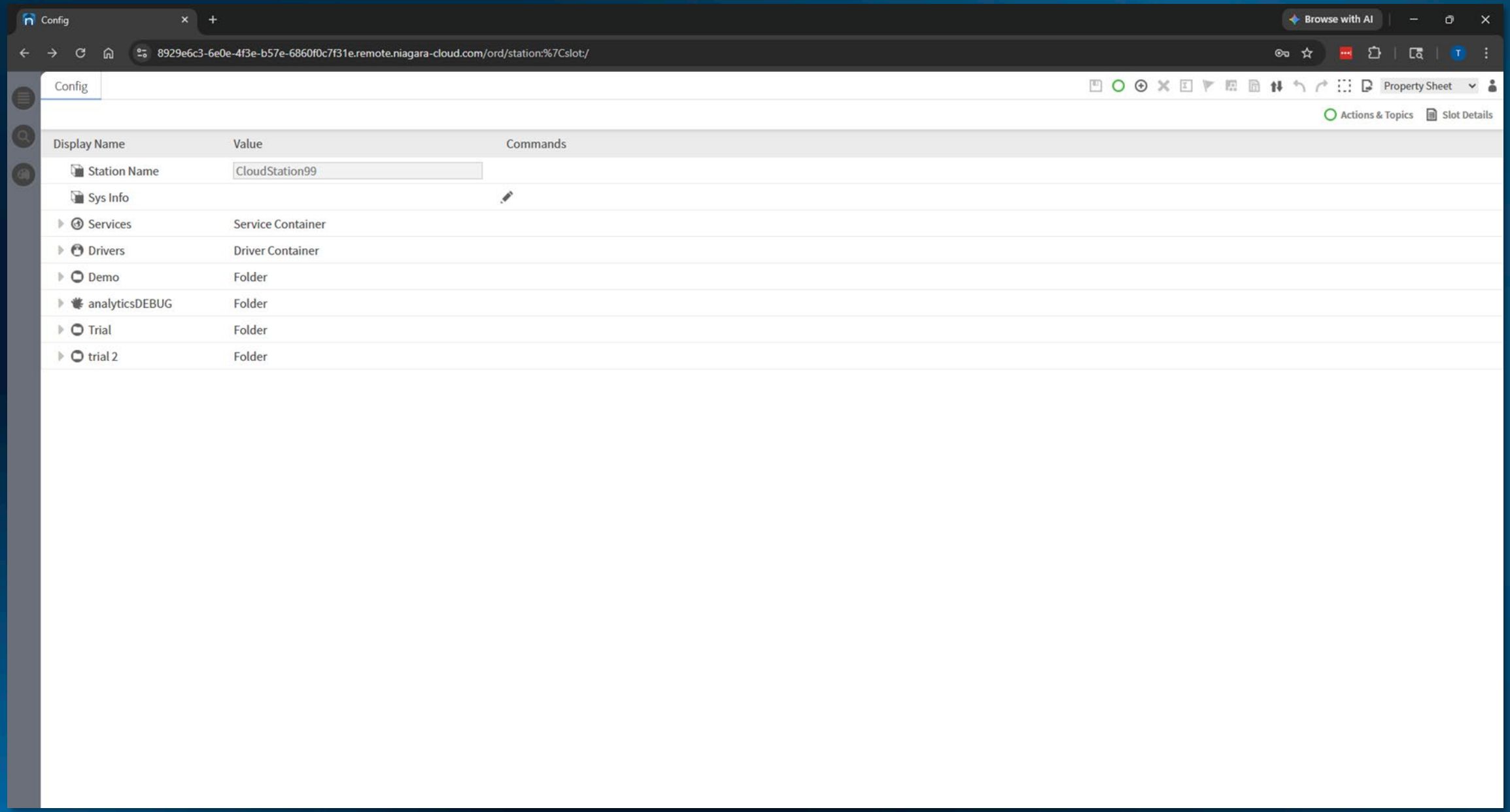
Niagara Data Service (NDS)



Niagara Cloud Portal (NCP)

NS26

TRIDIUM



Niagara Remote

NS26

TRIDIUM

Requirements

- ☑ Device Interoperability
- ☑ Control Logic
- ☑ User Interface
- ☑ Remote Access

NS26

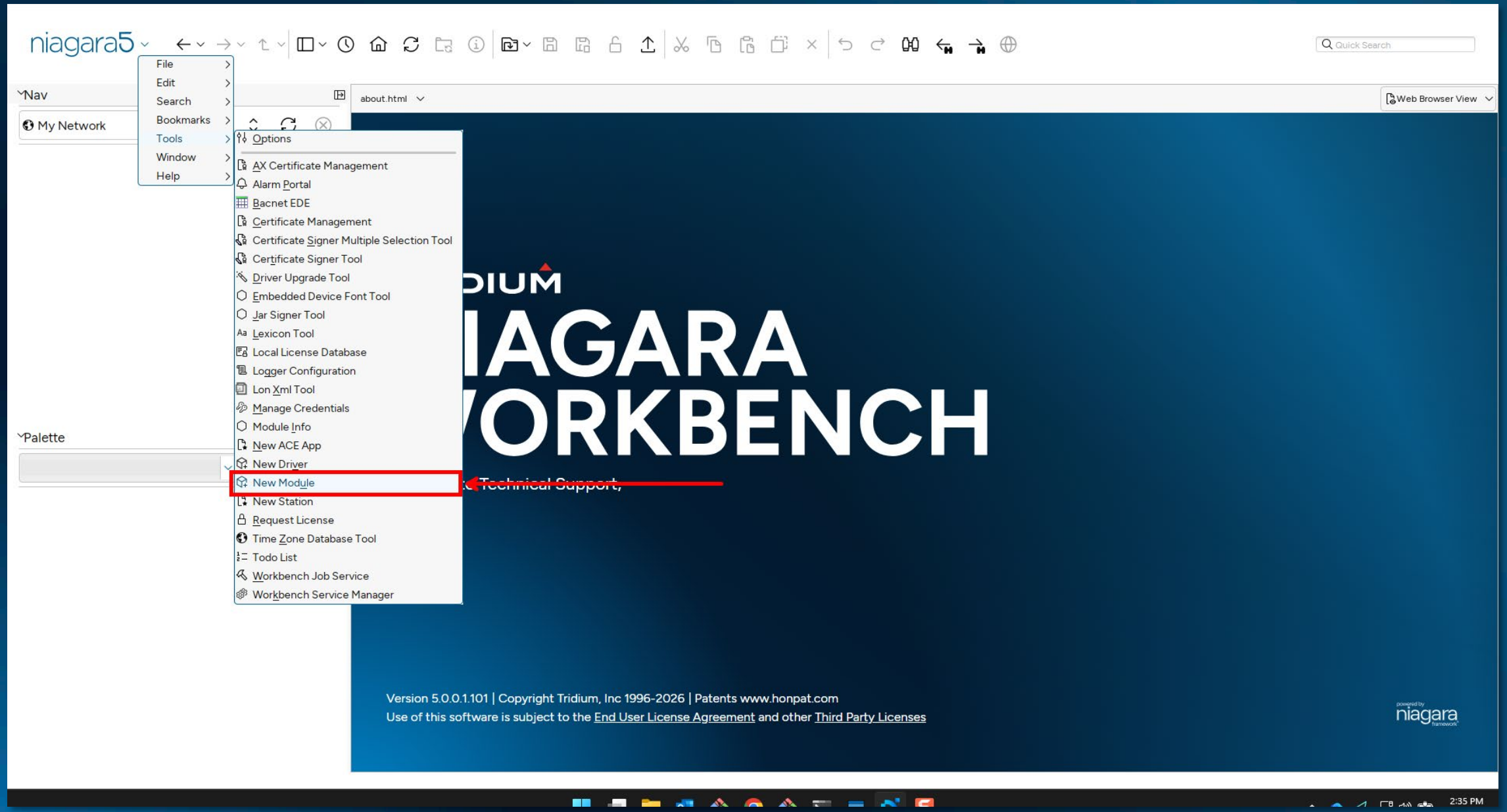
TRIDIUM

How do I extend Niagara?

Creating Custom Niagara Modules

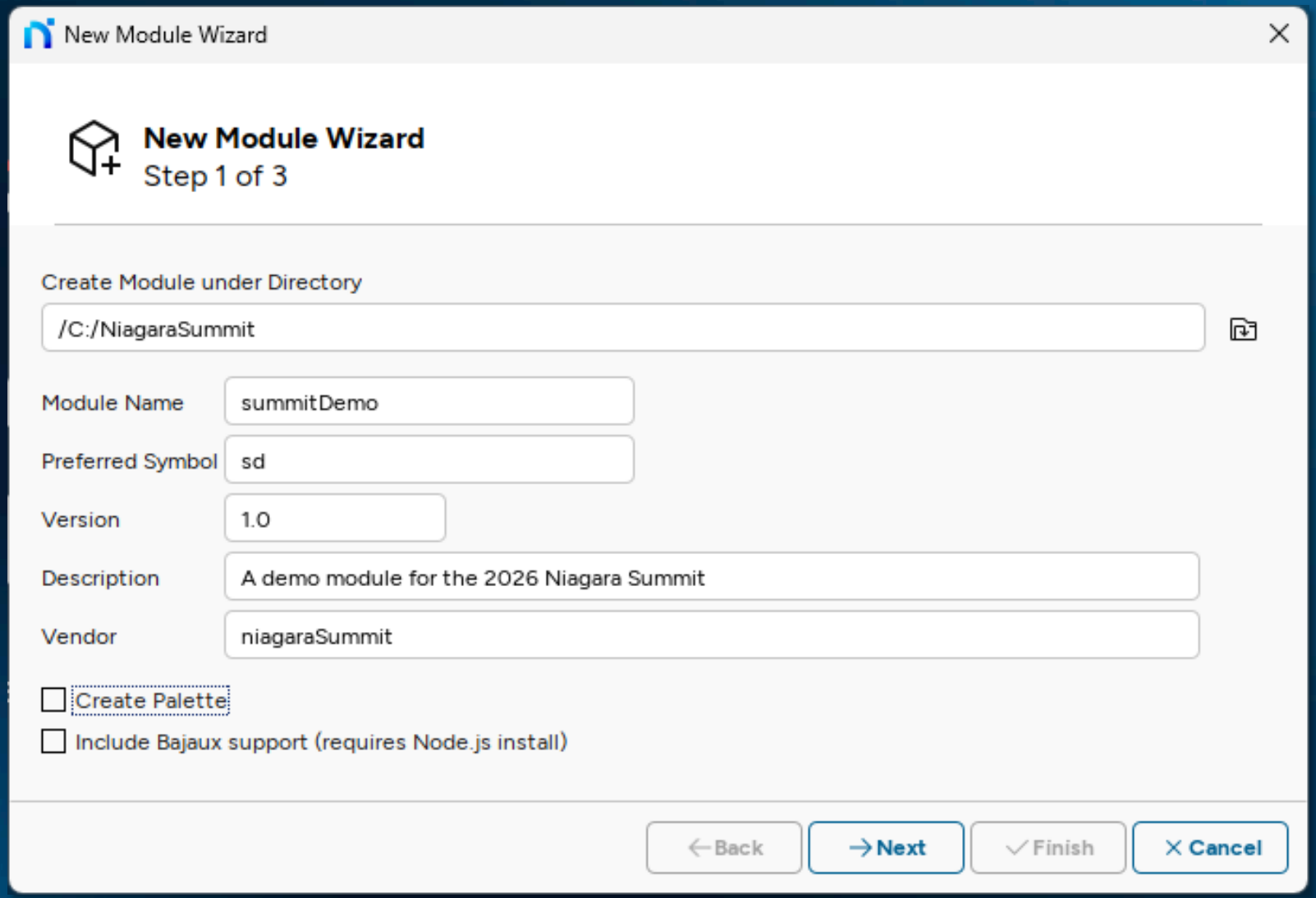
- Extending Niagara means creating custom code and compiling it into JAR files

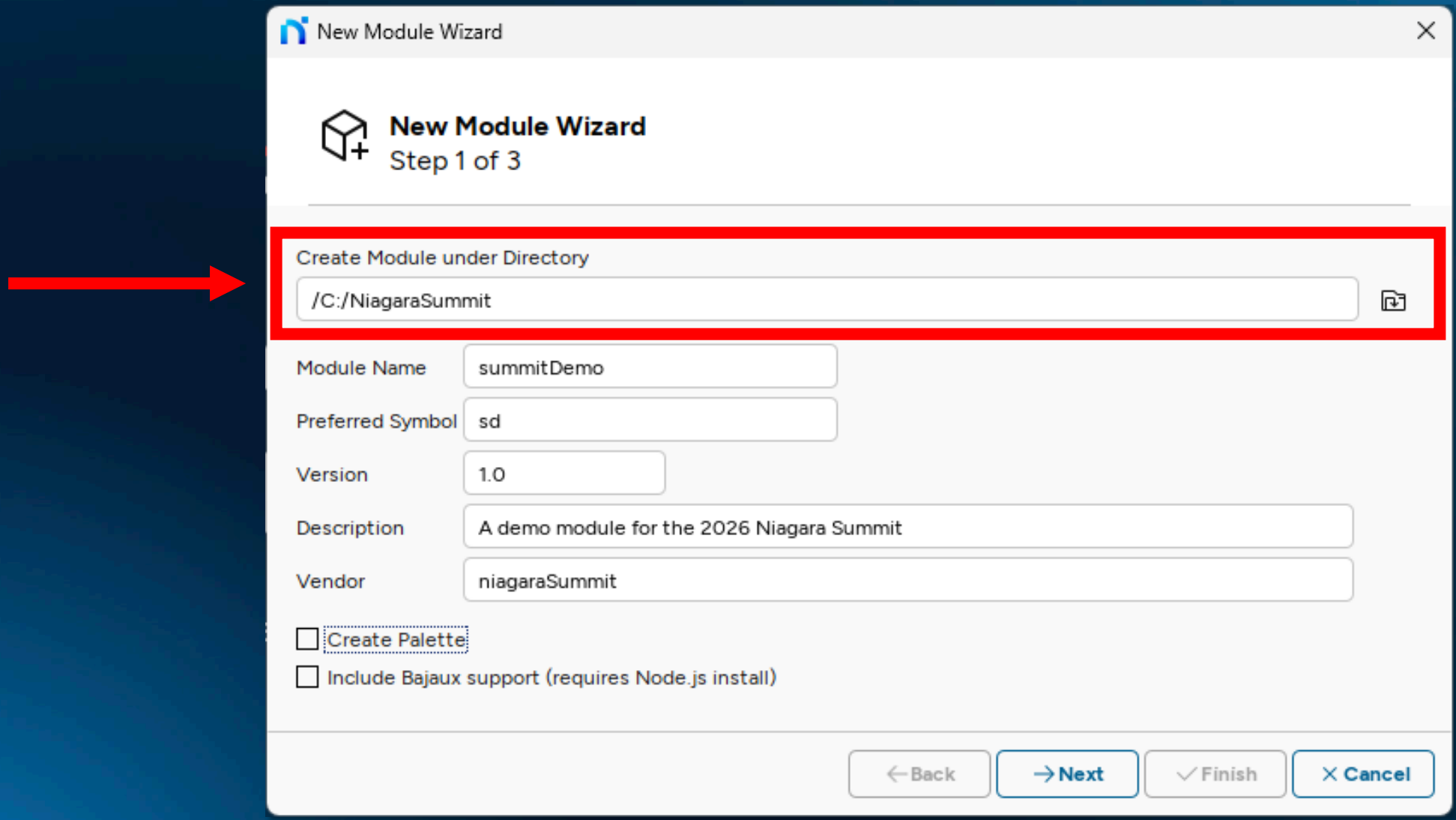
- These JAR files can then be installed as Niagara modules

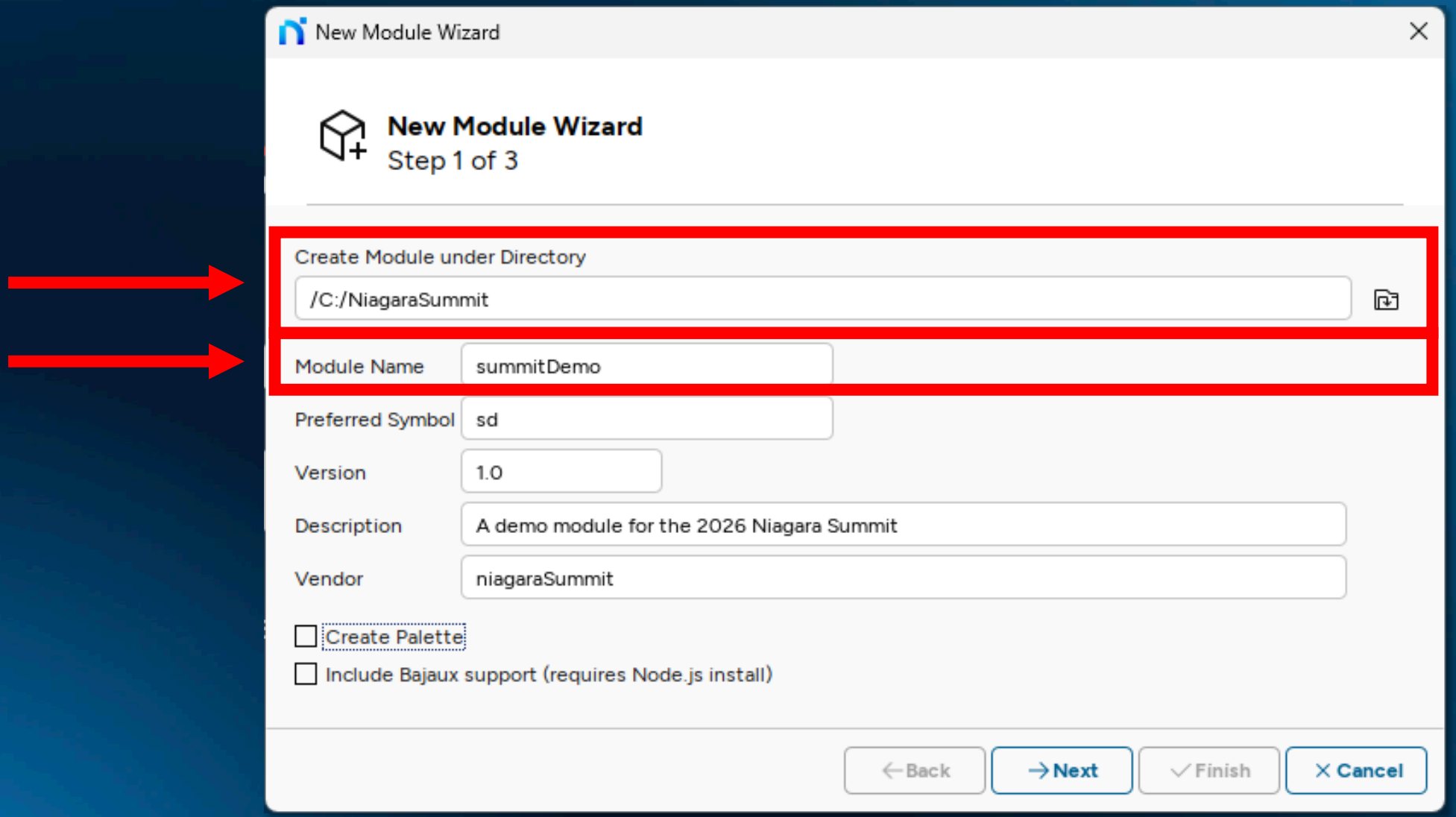


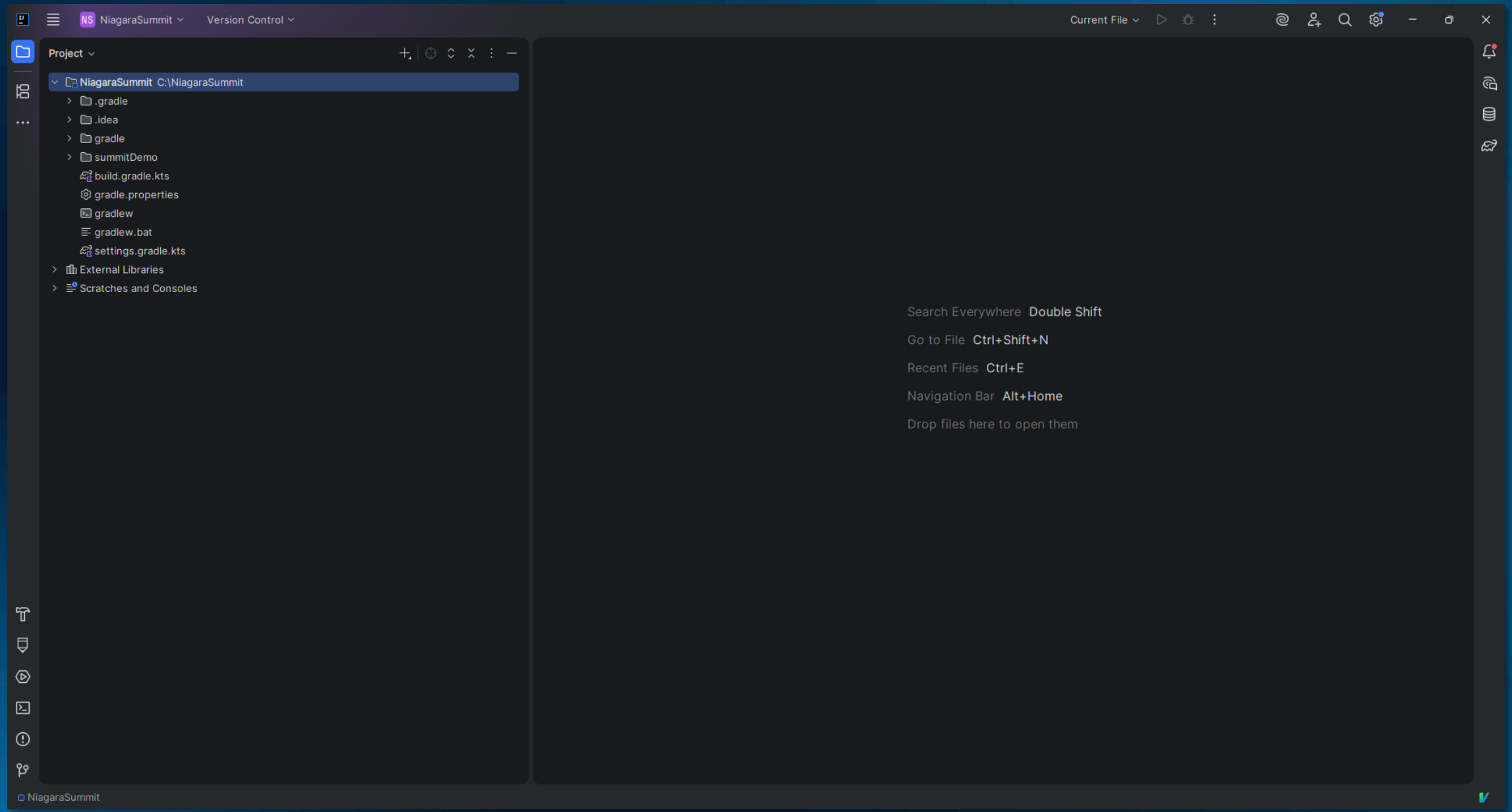
NS26

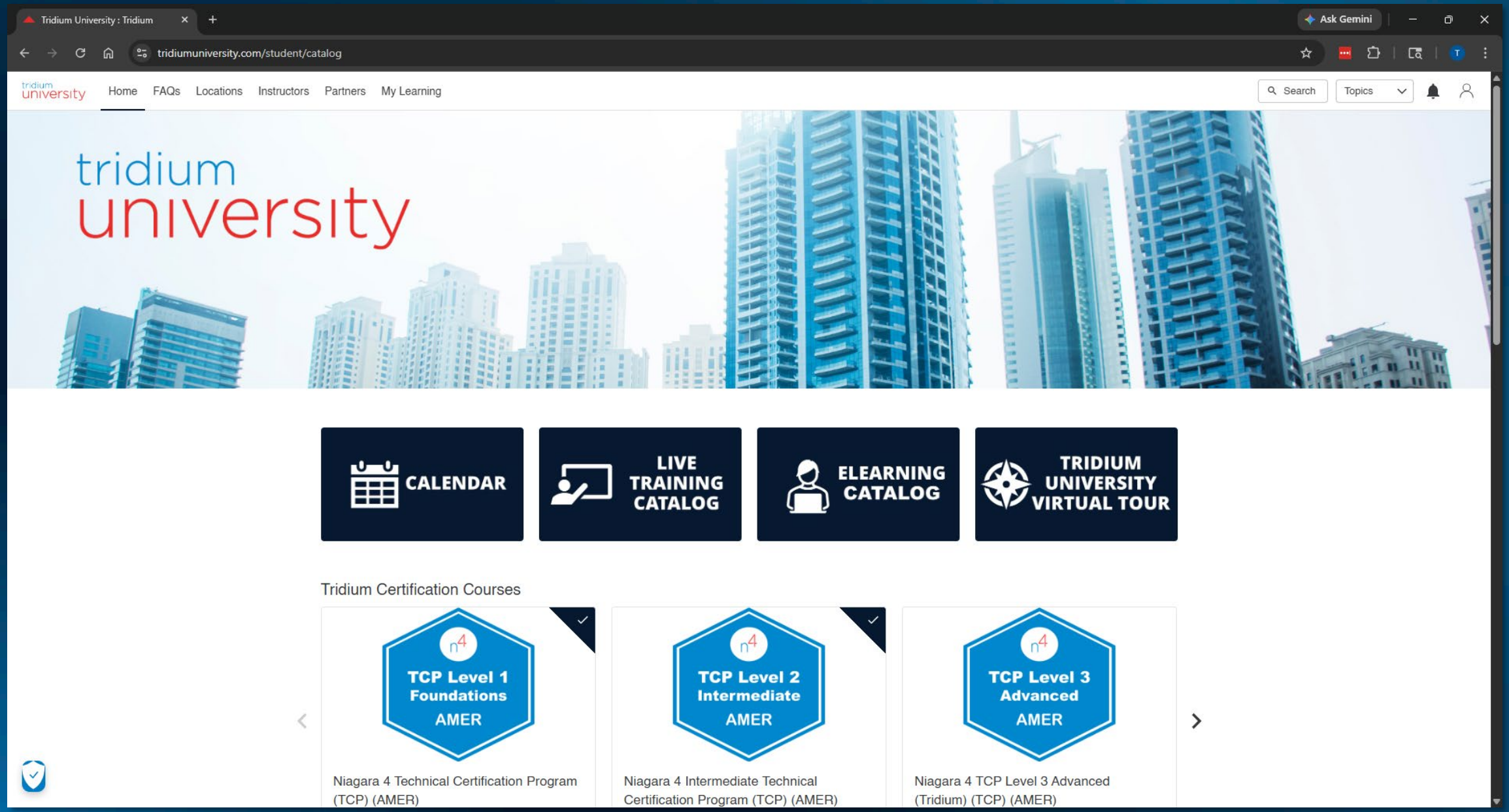
TRIDIUM


New Module Wizard
New Module Wizard
Step 1 of 3
Create Module under Directory
/C:/NiagaraSummit
Module Name
summitDemo
Preferred Symbol
sd
Version
1.0
Description
A demo module for the 2026 Niagara Summit
Vendor
niagaraSummit
Create Palette
Include Bajaux support (requires Node.js install)
Back
Next
Finish
Cancel


New Module Wizard
New Module Wizard
Step 1 of 3
Create Module under Directory
/C:/NiagaraSummit
Module Name
summitDemo
Preferred Symbol
sd
Version
1.0
Description
A demo module for the 2026 Niagara Summit
Vendor
niagaraSummit
Create Palette
Include Bajaux support (requires Node.js install)
Back
Next
Finish
Cancel


New Module Wizard
New Module Wizard
Step 1 of 3
Create Module under Directory
/C:/NiagaraSummit
Module Name
summitDemo
Preferred Symbol
sd
Version
1.0
Description
A demo module for the 2026 Niagara Summit
Vendor
niagaraSummit
Create Palette
Include Bajaux support (requires Node.js install)
Back
Next
Finish
Cancel

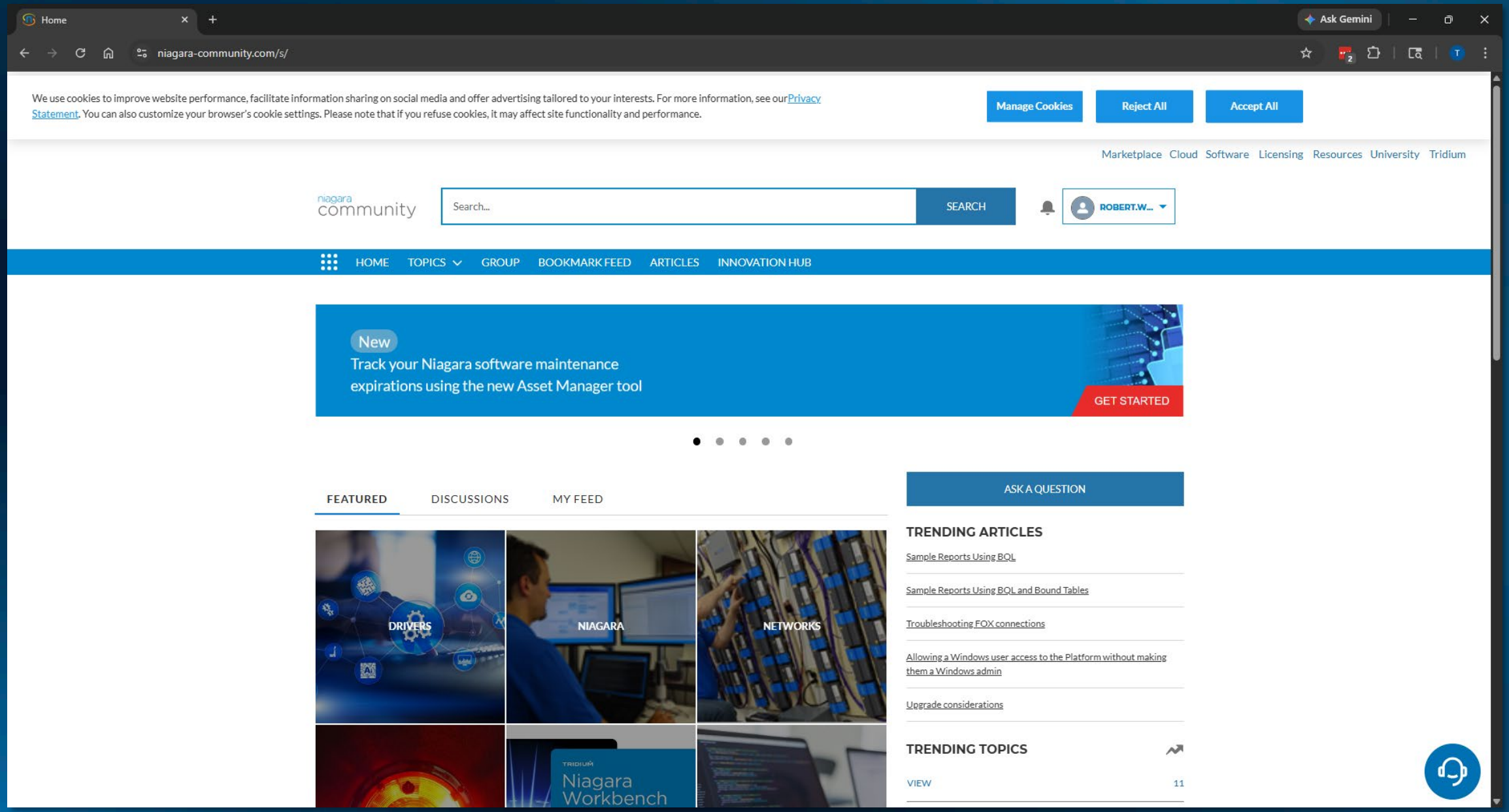

NiagaraSummit
Version Control
Current File
Project
NiagaraSummit C:\NiagaraSummit
.gradle
.idea
gradle
summitDemo
build.gradle.kts
gradle.properties
gradlew
gradlew.bat
settings.gradle.kts
External Libraries
Scratches and Consoles
Search Everywhere Double Shift
Go to File Ctrl+Shift+N
Recent Files Ctrl+E
Navigation Bar Alt+Home
Drop files here to open them
NiagaraSummit



Enroll in training at Tridium University

NS26

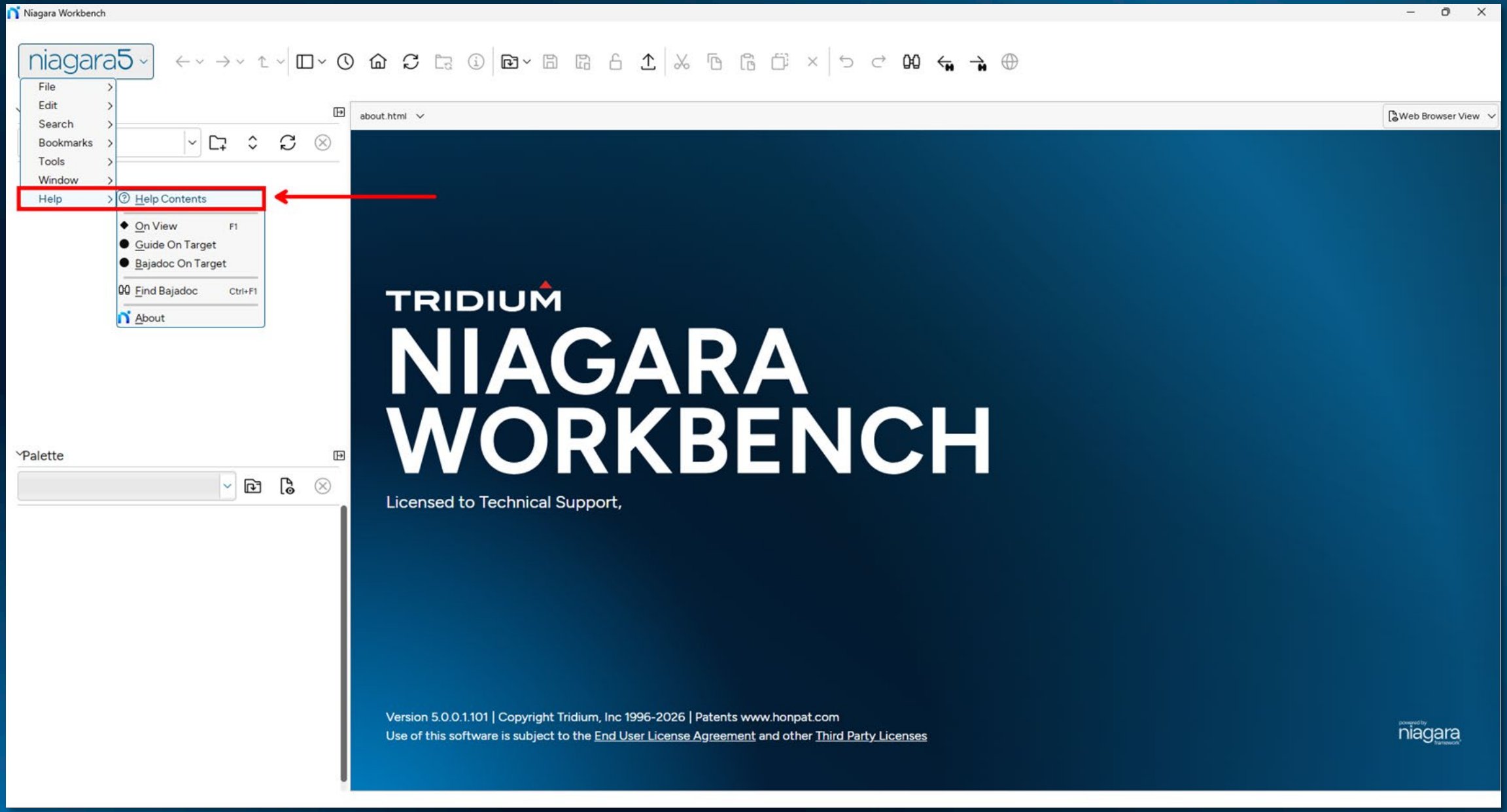
TRIDIUM



Check out Niagara Community Forums

NS26

TRIDIUM



Read docDeveloper in Niagara Help Menu

NS26

TRIDIUM


UP NEXT:
MY FIRST COMPONENT
WITH BRIAN

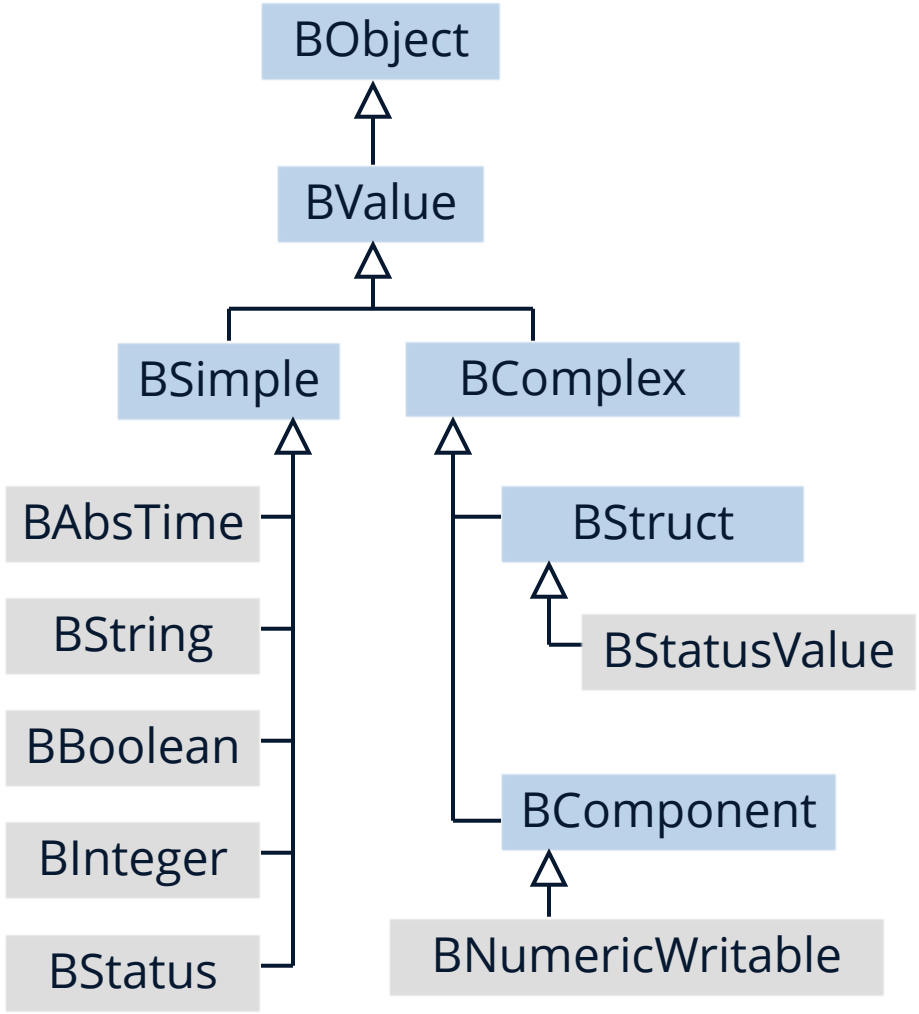
MY FIRST COMPONENT

BRIAN TODD

NIAGARA SUMMIT 2026 DEVELOPER

TRIDIUM

Niagara Component Model



Component Declaration

```
@NiagaraType
public class BEvChargingStation
  extends BComponent
{
}
```

Component Definition

```
@NiagaraType
@NiagaraProperty(...)
public class BEvChargingStation
  extends BComponent
{
}
```

@NiagaraProperty

```
/**
 * The power draw of the charging station, in kW.
 */
@NiagaraProperty(
  name = "powerDraw",
  type = "BFloat",
  defaultValue = "BFloat.make(0)"
)
```

Component Behaviors

```
@NiagaraType
@NiagaraAction(...)
public class BEvChargingStation
  extends BComponent
{
}
```

@NiagaraAction - Declaration

```
/**
 * Initiates charging procedure for the vehicle.
 */
@NiagaraAction(
   name = "startCharge",
   parameterType = "BVehicle",
   defaultValue = "new Bvehicle
)
```

@NiagaraAction - Implementation

```
/**
 * Start charging the {@code BVehicle} until the
 * battery is completely charged, or charging is
 * otherwise disrupted.
 */
public void doStartCharge(BVehicle vehicle, Context cx)
{
  // ...
}
```

Component Events

```
@NiagaraType
@NiagaraTopic(...)
public class BEvChargingStation
  extends BComponent
{
}
```

@NiagaraTopic

```
/**
 * Event indicating charging has completed.
 */
@NiagaraTopic(
   name = "chargeCompleted",
   eventType = "BString"
)
```

Slotomatic

```
@NiagaraType
@NiagaraSlots(…)
public class BEvChargingStation
  extends BComponent
{
}
```

Slotomatic - Invocation

~Developer/Project/niagara-modules/n5-summit-demo

```
> gradlew slotomatic

> Configure project:

BUILD SUCCESSFUL in 2s
1 actionable task: 1 executed
>
```

Slotomatic

```
@NiagaraType
@NiagaraSlots(…)
public class BEvChargingStation
  extends BComponent
{
//region /*+ ------------ BEGIN BAJA AUTO GENERATED CODE ------------ +*/
//@formatter:off
/*@ $com.summitdemo.evCharging(3622536179)1.0$ @*/
/* Generated Fri Mar 27 12:49:20 EDT 2026 by Slot-o-Matic (c) Tridium, Inc. 2012-2026 */

  //…

//@formatter:on
//endregion /*+ ------------ END BAJA AUTO GENERATED CODE -------------- +*/
}
```

Slotomatic – BObject

```
@NiagaraType
@NiagaraSlots(…)
public class BEvChargingStation
  extends BComponent
{
//region /*+ ------------ BEGIN BAJA AUTO GENERATED CODE ------------ +*/
//…

  @Override
  public Type getType() { return TYPE; }
  public static final Type TYPE = Sys.loadType(BEvChargingStation.class);

//…
//endregion /*+ ------------ END BAJA AUTO GENERATED CODE -------------- +*/
}
```

Slotomatic – Property Slot

```
@NiagaraType
@NiagaraProperty(…)
public class BEvChargingStation
  extends BComponent
{
//region /*+ ------------ BEGIN BAJA AUTO GENERATED CODE ------------ +*/
//…

  public float getPowerDraw() { return getFloat(powerDraw); }
  public void setPowerDraw(float v) { setFloat(powerDraw, v, null); }


//…
//endregion /*+ ------------ END BAJA AUTO GENERATED CODE -------------- +*/
}
```

Slotomatic – Action Slot

```
@NiagaraType
@NiagaraAction(…)
public class BEvChargingStation
  extends BComponent
{
//region /*+ ------------ BEGIN BAJA AUTO GENERATED CODE ------------ +*/
//…

  public void startCharge(BVehicle parameter) {
    invoke(startCharge, parameter, null);
  }

//…
//endregion /*+ ------------ END BAJA AUTO GENERATED CODE -------------- +*/
}
```

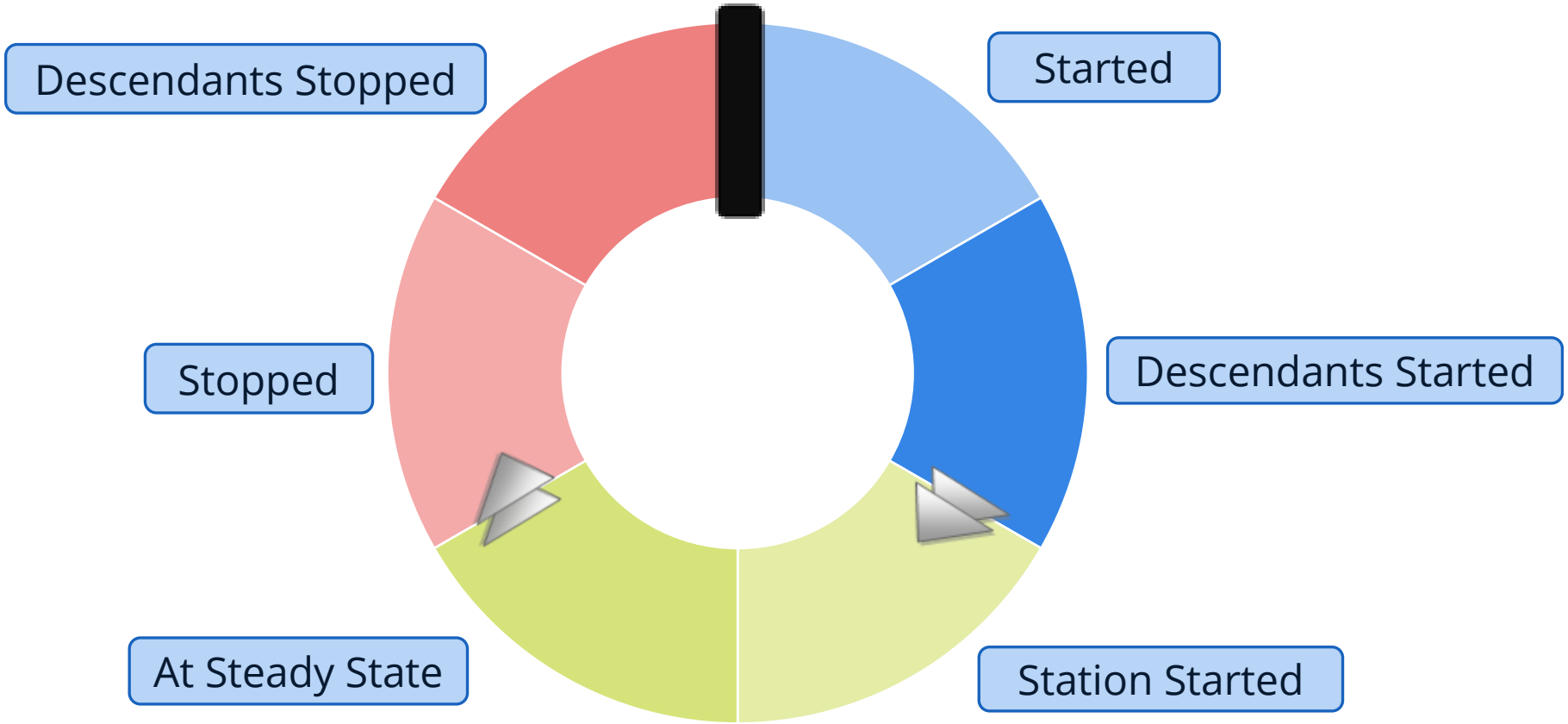
Slotomatic – Topic Slot

```
@NiagaraType
@NiagaraTopic(…)
public class BEvChargingStation
  extends BComponent
{
//region /*+ ------------ BEGIN BAJA AUTO GENERATED CODE ------------ +*/
//…

  public void fireChargeCompleted(BString event) {
    fire(chargeCompleted, event, null);
  }

//…
//endregion /*+ ------------ END BAJA AUTO GENERATED CODE -------------- +*/
}
```

@NiagaraTopic – Emitting Events

```
/**
 * Start charging the {@link BVehicle} until the battery is
 * completely charged, or charging is otherwise disrupted.
 */
public void doStartCharge(BVehicle vehicle, Context cx) {
  while(!vehicle.hasFullCharge()) {
    // Keep charging!
  }
  fireChargeCompleted(BString.make("Charging complete."));
}
```

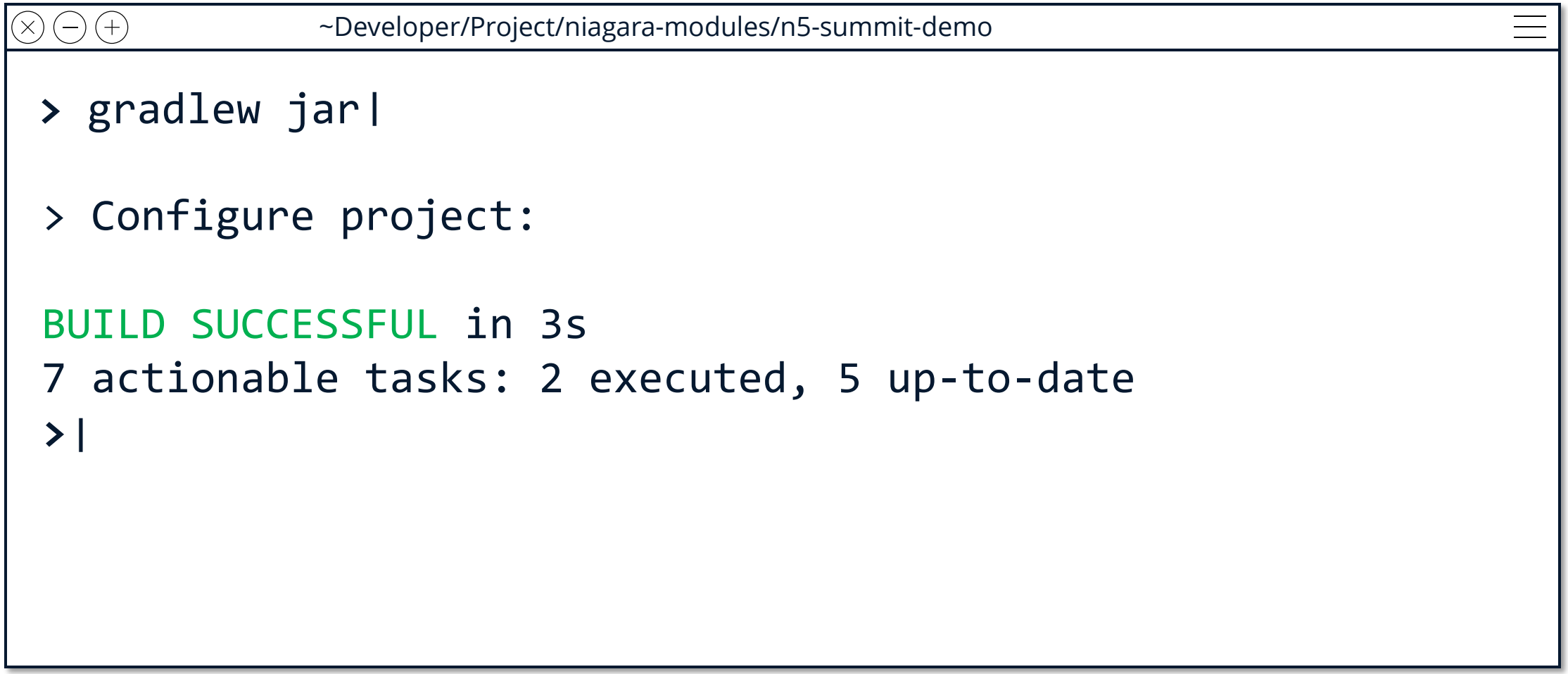
BComponent Lifecycle

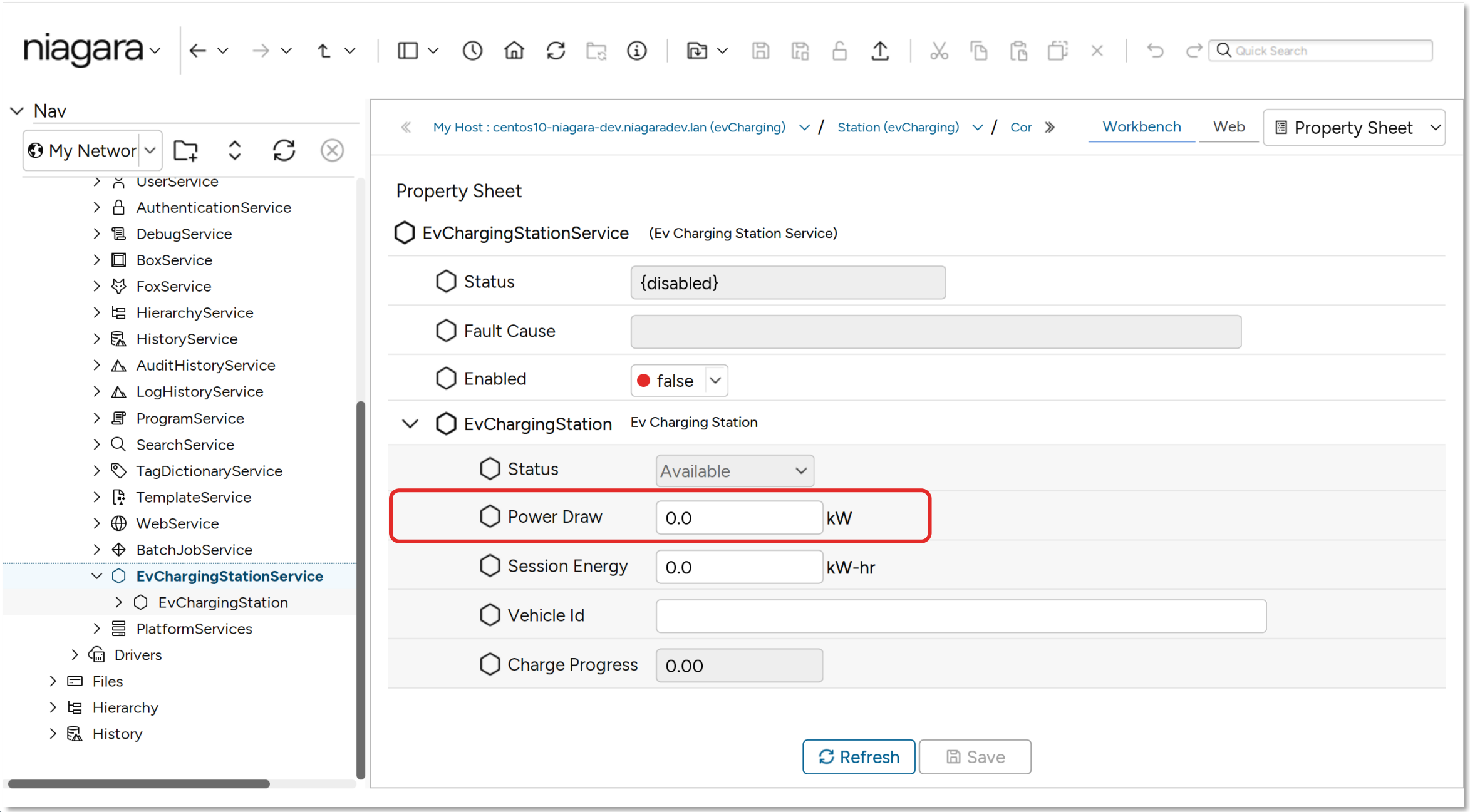


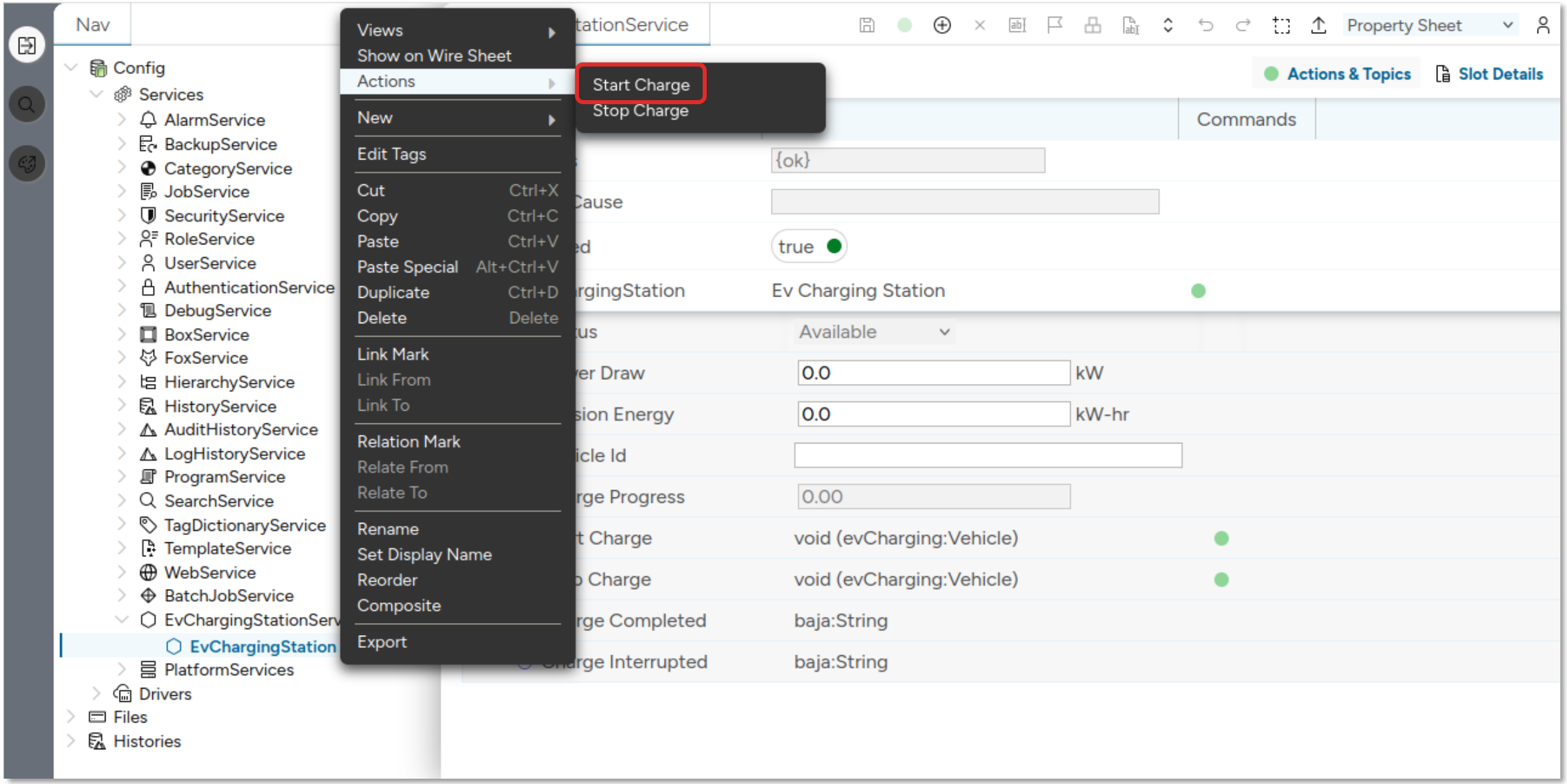
Component Callbacks

```
public void started()

public void descendantsStarted()

public void stationStarted()

public void atSteadyState()

public void stopped()

public void descendantsStopped()
```

Module Compilation and Installation




niagara
Quick Search
Nav
My Networ
UserService
AuthenticationService
DebugService
BoxService
FoxService
HierarchyService
HistoryService
AuditHistoryService
LogHistoryService
ProgramService
SearchService
TagDictionaryService
TemplateService
WebService
BatchJobService
EvChargingStationService
EvChargingStation
PlatformServices
Drivers
Files
Hierarchy
History
My Host : centos10-niagara-dev.niagaradev.lan (evCharging)
Station (evCharging)
Cor
Workbench
Web
Property Sheet
Property Sheet
EvChargingStationService
(Ev Charging Station Service)
Status
{disabled}
Fault Cause
Enabled
false
EvChargingStation
Ev Charging Station
Status
Available
Power Draw
0.0
kW
Session Energy
0.0
kW-hr
Vehicle Id
Charge Progress
0.00
Refresh
Save


Nav
Config
Services
AlarmService
BackupService
CategoryService
JobService
SecurityService
RoleService
UserService
AuthenticationService
DebugService
BoxService
FoxService
HierarchyService
HistoryService
AuditHistoryService
LogHistoryService
ProgramService
SearchService
TagDictionaryService
TemplateService
WebService
BatchJobService
EvChargingStationServ
EvChargingStation
PlatformServices
Drivers
Files
Histories
Views
Show on Wire Sheet
Actions
New
Edit Tags
Cut Ctrl+X
Copy Ctrl+C
Paste Ctrl+V
Paste Special Alt+Ctrl+V
Duplicate Ctrl+D
Delete Delete
Link Mark
Link From
Link To
Relation Mark
Relate From
Relate To
Rename
Set Display Name
Reorder
Composite
Export
Start Charge
Stop Charge
tationService
Property Sheet
Actions & Topics
Slot Details
Commands
{ok}
Cause
true
rgingStation
Ev Charging Station
Available
0.0
kW
0.0
kW-hr
0.00
t Charge
void (evCharging:Vehicle)
Charge
void (evCharging:Vehicle)
rge Completed
baja:String
rge Interrupted
baja:String


UP NEXT:
MY FIRST WIDGETS
WITH VIKRAM

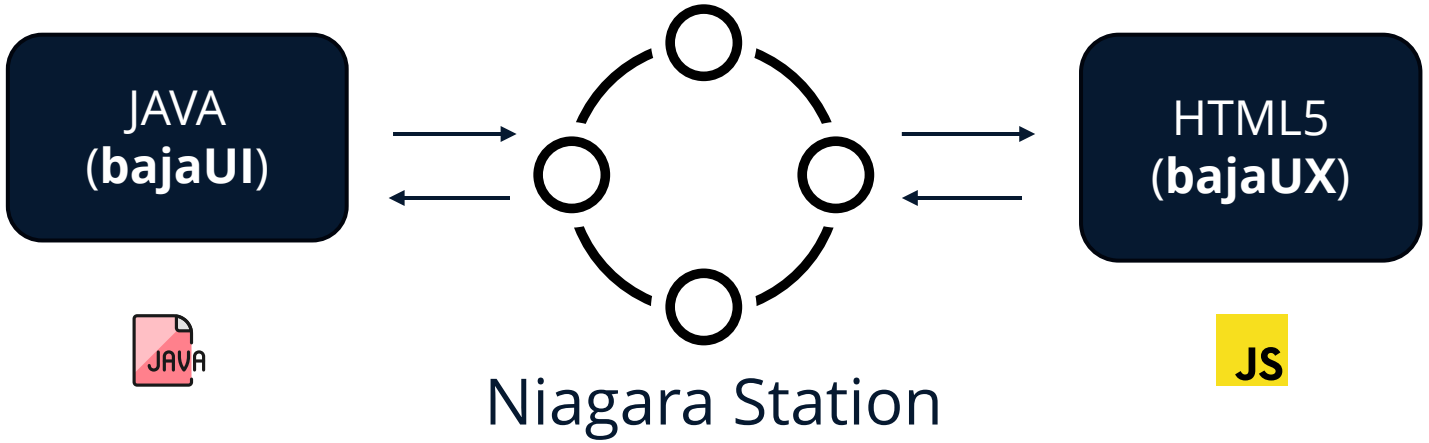
MY FIRST UI & UX WIDGETS

VIKRAM NAGULAN

NIAGARA SUMMIT 2026 DEVELOPER

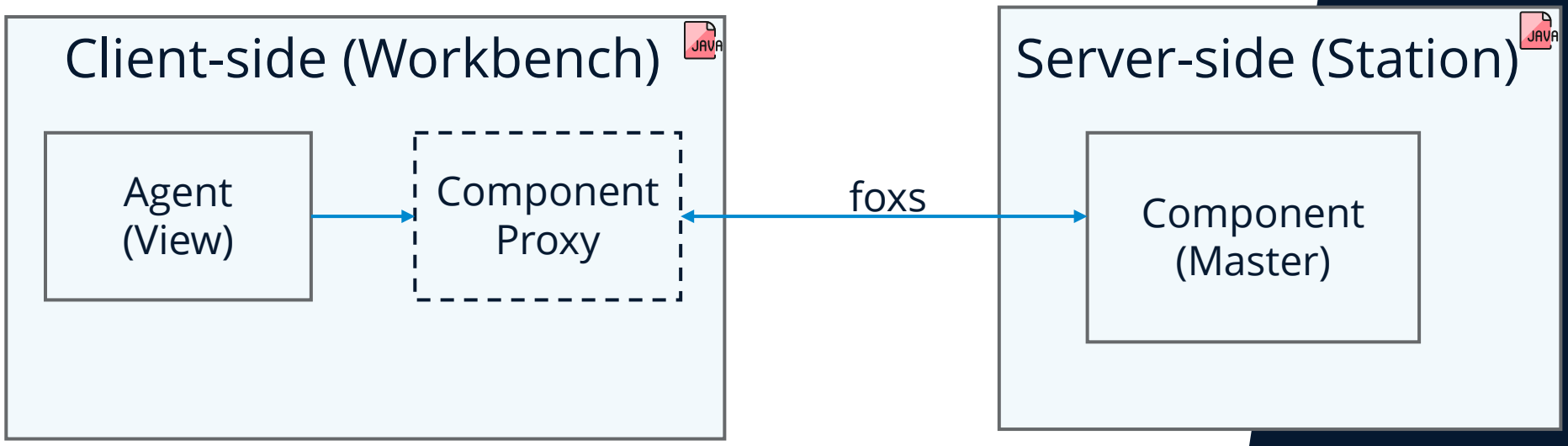
TRIDIUM

Niagara UI



[Icon credits - streamline]

Workbench UI Architecture

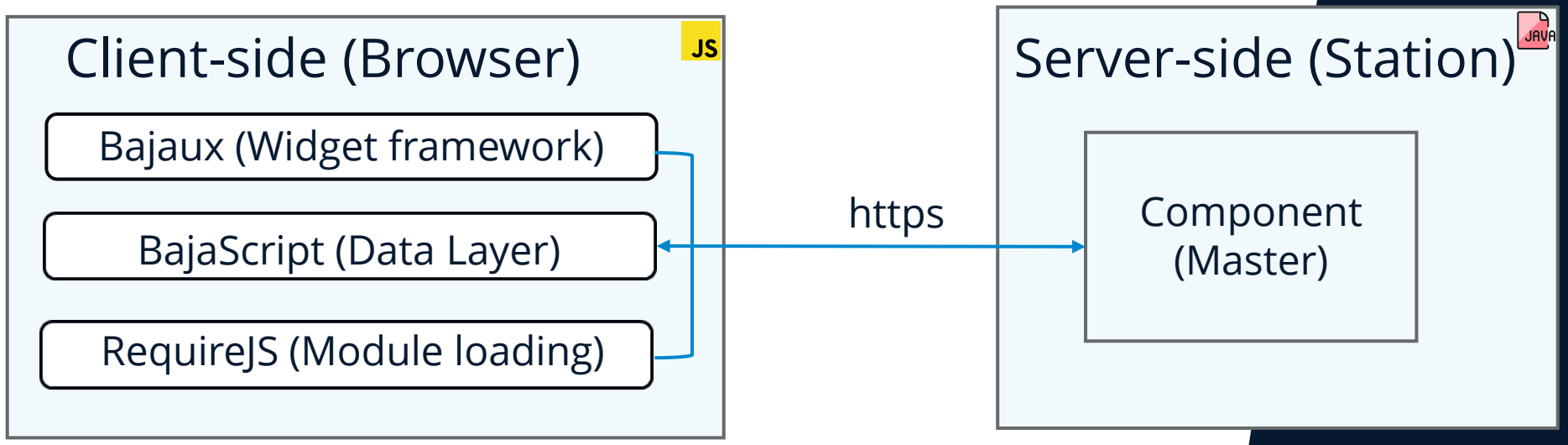



```
@NiagaraType(
   agent = @AgentOn(
      types = "summitdemo:EvChargingStation"
   )
)
public class BEvChargingStationView
   extends BWbComponentView
{..}
```

Developing a Workbench (UI) View

```
@Override
protected void doLoadValue(BObject value, Context context)

@Override
public void computePreferredSize(ILayoutConstraints lc, IRenderContext cx)

@Override
public void doLayout(IRenderContext cx)

@Override
public void doPaint(Graphics g, IRenderContext cx)

@Override
protected BObject doSaveValue(BObject value, Context cx)
```

Modern Web View Architecture (N4+)




```
@NiagaraType(
   agent = @AgentOn(
      types = { "summitdemo:EvChargingStation",
"summitdemo:EvChargingStationView" }
   )
)
@NiagaraSingleton
public class BUxEvChargingStationView
   extends BSingleton
   implements BIJavaScriptWidget
{..}
```

JS Developing a BajaUX View

```
doInitialize(dom) // Bind to DOM

doLoad(value) // Update UI (Input)

doRead() // Read UI State

doSave() // Mutate the loaded value (Output)
```

Developing a BajaUX View

Make life easier

1. Use spandrel and JSX.
2. State management

BEvChargingStation.java

```
@{…}
public class BEvChargingStation
  extends BComponent
{
  …

  @Override
  public boolean isParentLegal(BComponent parent)
  {
    return parent instanceof BEvChargingStationService;
  }

  public void doStartCharge(BVehicle vehicle, Context cx)
    throws InterruptedException, ExecutionException
  { … }

  public void doStopCharge(BVehicle vehicle, Context cx)
  { … }
```

BEvChargingStationView.java

```
@NiagaraType(agent = @AgentOn(types = "evCharging:EvChargingStation"))
public class BEvChargingStationView
  extends BWbComponentView
{
  public BEvChargingStationView()
  {
  BGridPane container = new BGridPane(1);
  container.add("stationLabel", chargeStationLabel);
  …
  setContent(container);
}

@Override
protected void doLoadValue(BObject value, Context context)
  throws Exception
{
  super.doLoadValue(value, context);
  BEvChargingStation chargingStation = (BEvChargingStation)value;
  chargeStationLabel.setText(chargingStation.getDisplayName(null));
  …
}
```

JS EvChargingStationView.js

```
define([…], function (
  baja, $, Promise, Command, spandrel, CommandButton, compUtils) {
  …
  class EvChargingStationView extends spandrel((value, state) => {
    const { powerDraw, self, stationName, status, vehicleId, sessionEnergy } = state;
    return <div className="EvChargingStationView-Wrapper">
      <table className="EvChargingStationView-layout">
        <tr …>
      </table>
    </div>;
  }, { strategy: 'niagara' }) {
    constructor(params) { super({ params, defaults: widgetDefaults() }); … }

    toState(value) {…}
  }

  return EvChargingStationView;
});
```

& .CSS and .NSS

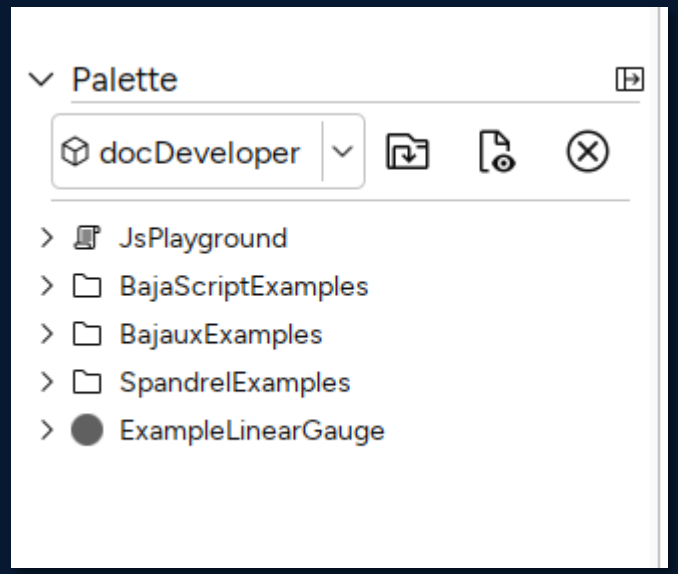
```
.EvChargingStationView
.EvChargingStationView-Wrapper
table {
  margin: 10px;
  background-color: beige;
  border: 1px solid black;
  border-radius: 4px;
}
```

```
evCharging|EvChargingStationView {
  & > [name=content] {
    background-color: beige;
    min-height: 200;
    padding: 8;
    & > [name=center] {
      ...
    }

    [name=stationLabel] {
      ...
    }

    bajaui|ProgressBar {
      ...
    }

    bajaui|Button {
      ...
    }
  }
}
```

Demo

Need help?

docDeveloper




UP NEXT:
MY FIRST NDRIVER
WITH MITHUN

MY FIRST NDRIVER

MITHUN TIRUVEDULA

NIAGARA SUMMIT 2026 DEVELOPER

TRIDIUM

Agenda

Everything you need to get started on your first NDriver.

1. The NDriver Framework
2. My First NDriver
   - Preparation
   - New NDriver Wizard
   - NDriver Structure – Map to Reality
3. NDriver Capabilities
   - NDriver Communication
   - Connectivity
   - Points - Read and Write
   - Advanced

Limitless Device Integrations with Niagara

- **Lighting**
  - Indoor
  - Outdoor
  - Signage
- **HVAC**
  - VRF
  - RTU
  - Cooling Towers
  - Head pump
- **Central Plant**
  - Boiler
  - Chiller
  - ICE Storage
- **Metering**
  - Gas
  - Water
  - Electric
- **Data Center**
  - Data
  - Battery
  - Power
  - Cooling
- **Irrigation**
  - Sprinkler
  - Recovery
  - Weather
- **Safety**
  - Access
  - Cameras
  - Fire
  - Air Quality
- **Sensors**
  - Levels
  - Motion
  - Light
  - Door
  - Audio
  - Motors
  - Contacts
  - Particulate
  - Voltage/Amps
- **Refrigeration**
  - Retail
  - Medical
- **Generation**
  - Solar
  - Diesel
  - Inverters
- **Elevator**
  - Monitor
  - Access

BACnet · BACnet/SC secure connect · Modbus · LonWorks · oBIX Open Building Information Xchange · DALI · KNX · MQTT.org · OPC Foundation

NDriver

The NDriver **framework** is the toolkit for building drivers that import and export data between a Niagara station and external systems.

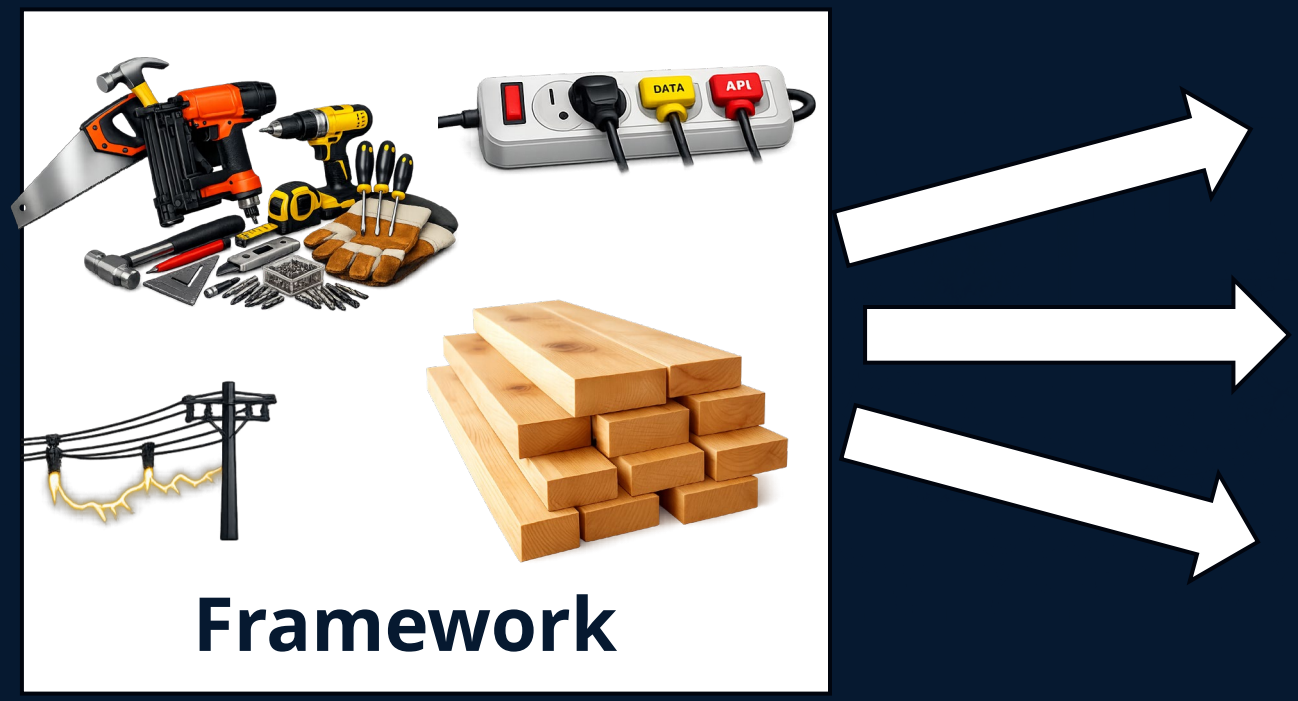
The toolkit you build with

An **NDriver** is a Niagara module that connects the framework to specific device families or protocols

The module you build

Framework

A set of reusable components, tools, and guidelines



What’s neat about NDriver

Protocol Agnostic

Supports any device family or custom protocol

Bidirectional

Handles both import and export of external data

Framework Native

Deeply integrated with the Niagara station lifecycle

NDriver Capabilities

AKA Device Extensions

- Connectivity
  - Point
  - Alarm
  - History
  - Schedule

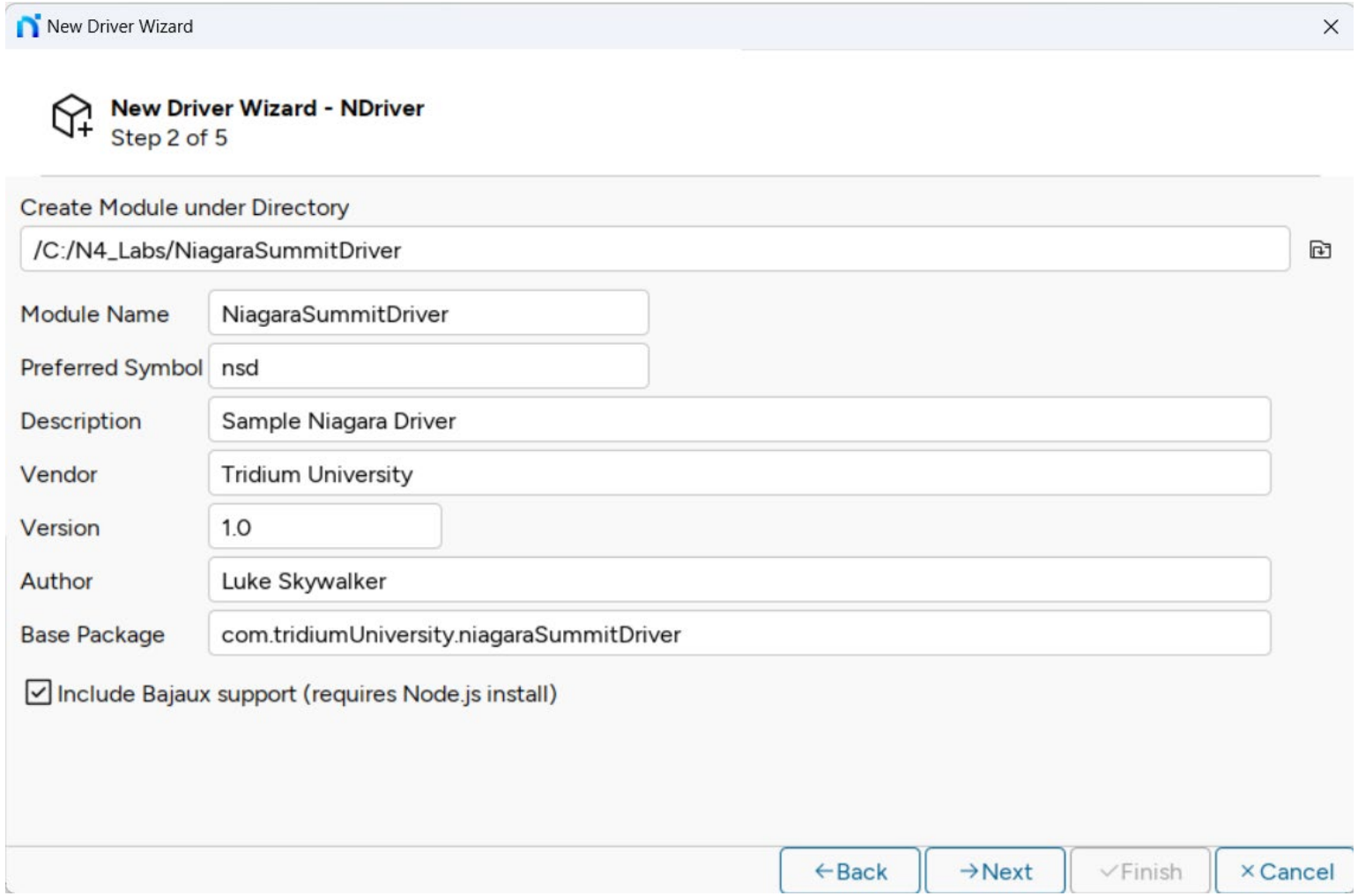
My First NDriver - Preparation

- Know your Protocol
- Scope Checklist:
  - ❑ Which **capabilities**? (Points, History, Alarms, Schedules)
  - ❑ **Transport** — TCP, UDP, Serial, HTTP, or custom?
  - ❑ **Communication Strategy** — Polling, subscriptions, event-based?
  - ❑ **Discovery** — Can you auto-discover devices and points?
  - ❑ **Configuration** — What user-facing properties does your protocol need?
  - ❑ **Data Type Mapping** — How do protocol types map to Niagara's object model?

Pro Tip:

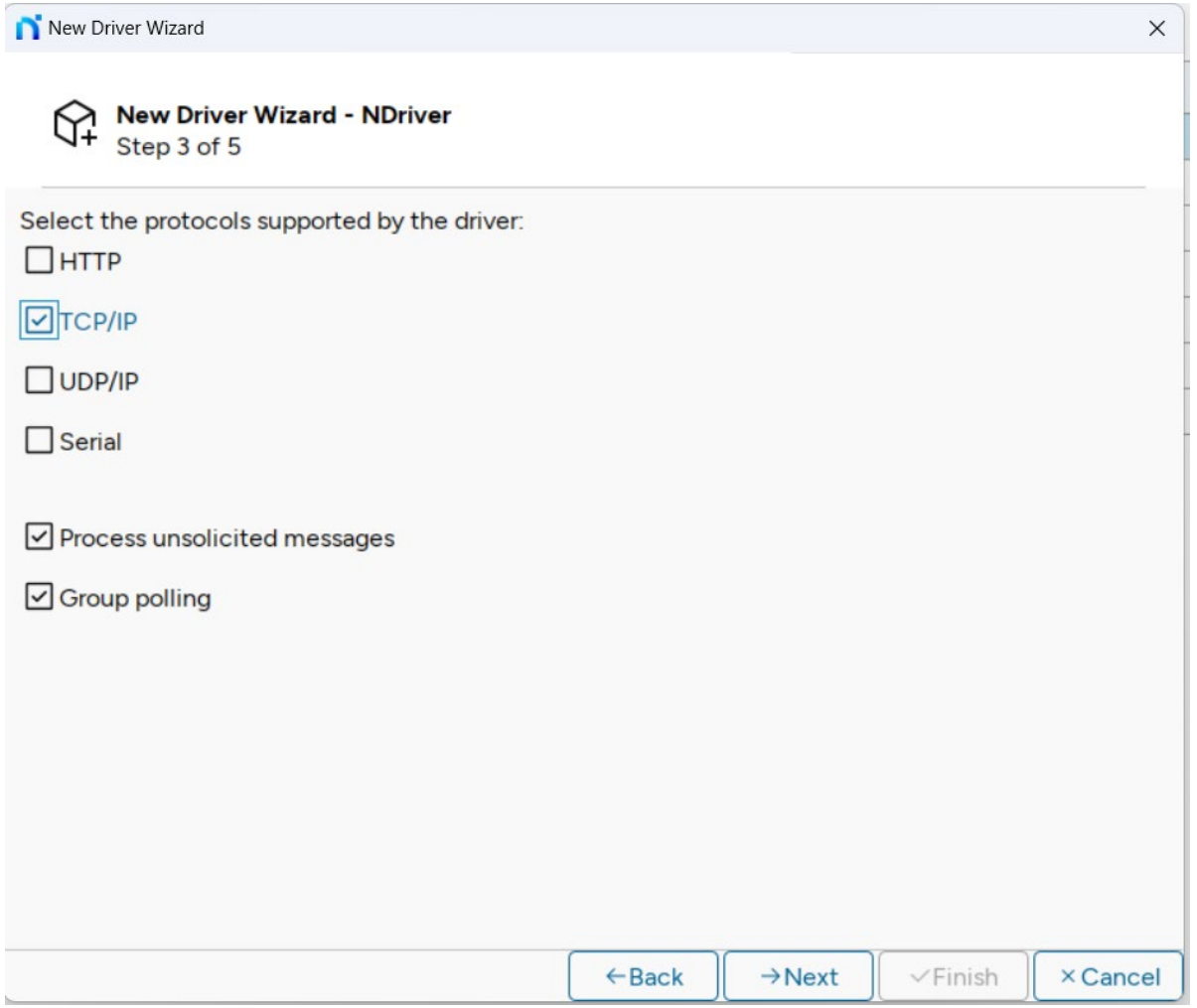
Start Simple, Expand Gradually

My First NDriver - Wizard



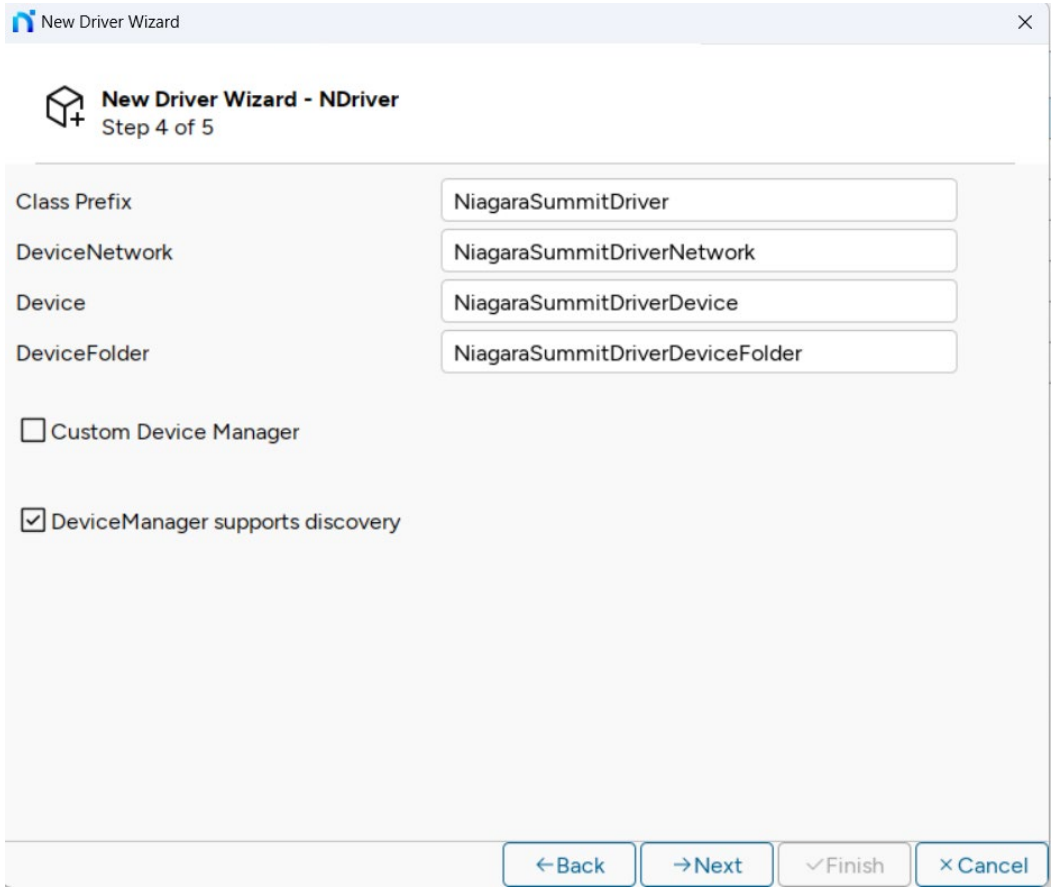
My First NDriver - Wizard

- Transport & Communication
  - Transport type:
    - TCP / UDP / Serial / HTTP
    - Or a combination
    - Or custom
  - Unsolicited messages support
  - Group polling support **(N5)**

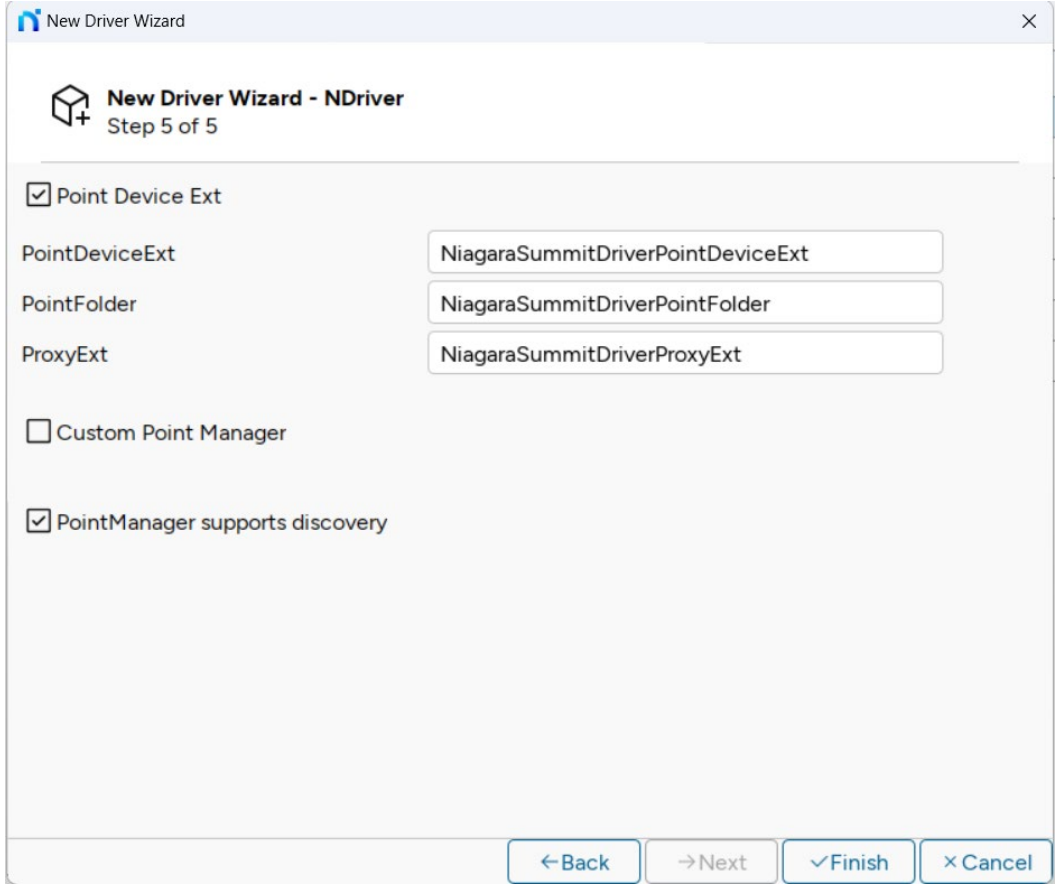


My First NDriver - Wizard

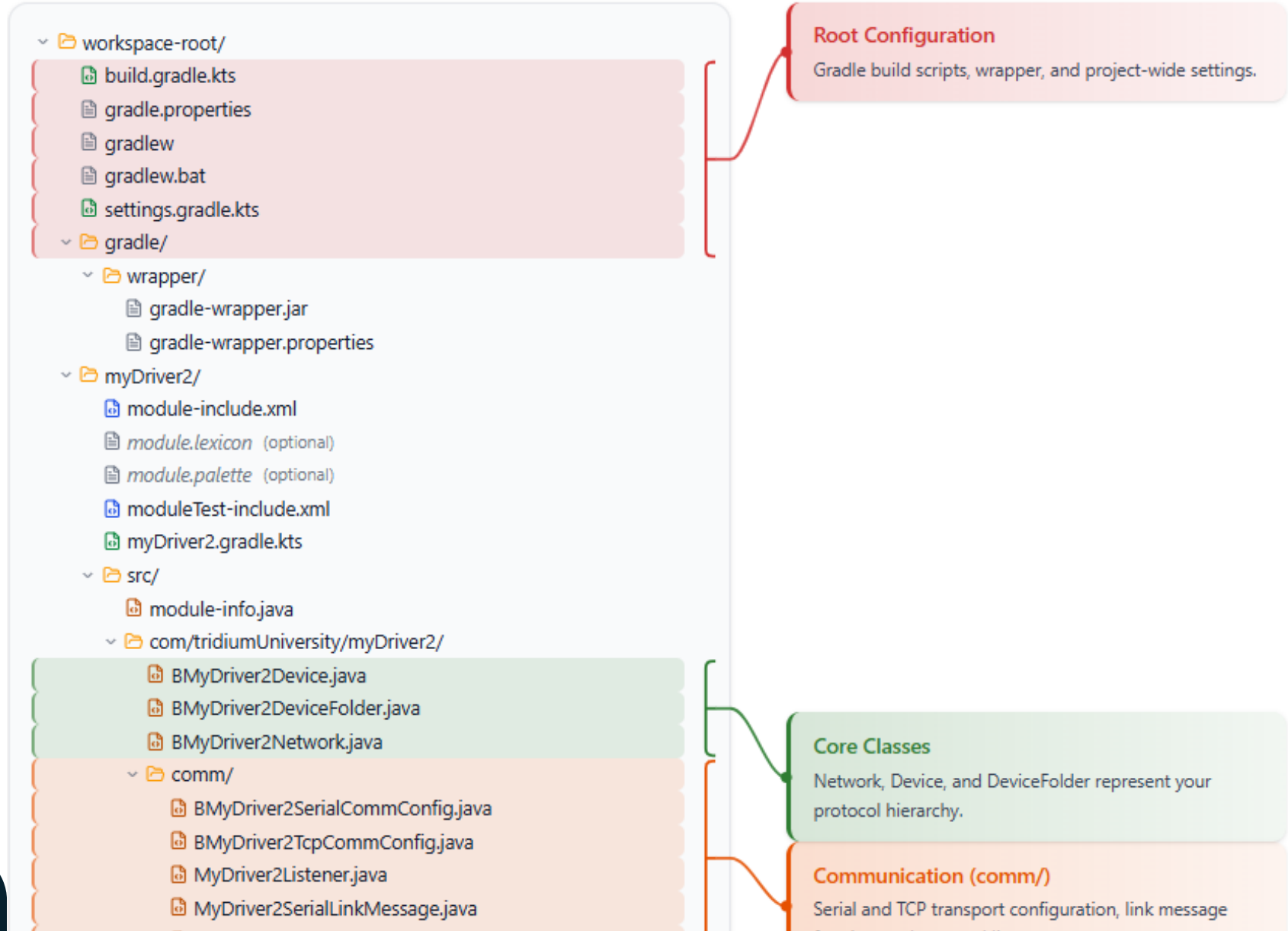
- Device class, discovery and UI

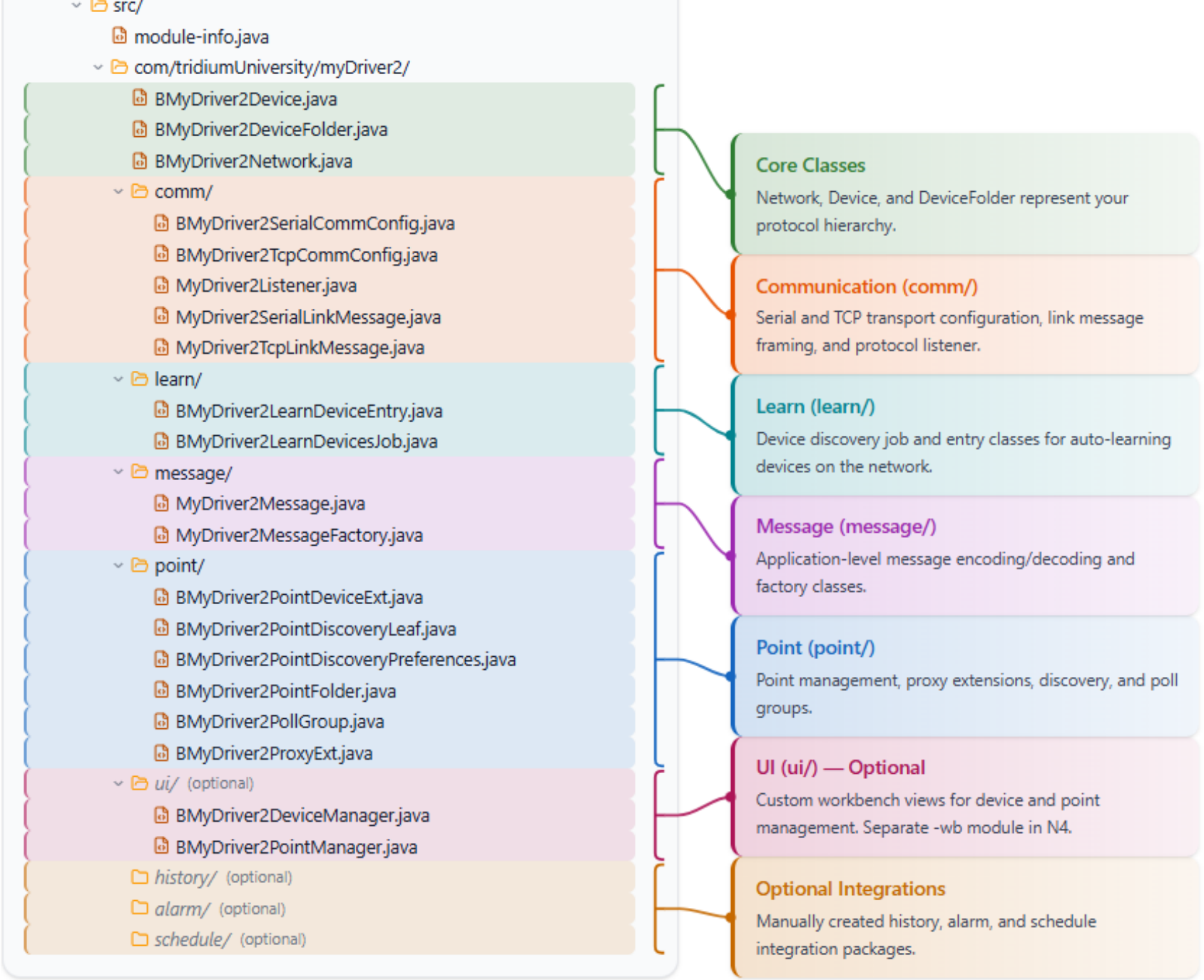


- Point class, discovery and UI



My First NDriver - Structure





Alarm, History, and Schedule device extensions are not generated.

You must add them manually if needed.

NDriver – Real World

CODE

Capabilities: Messages – ABCs of your protocol

NMessage

Application-level

Your protocol's request/response/ unsolicited logic

One subclass per message type

LinkMessage

Wire-level

Raw bytes on the wire

Typically, one subclass for the driver

NMessage Request

```
@Override
public boolean toOutputStream(OutputStream out)
public Object getTag()
```

NMessage Response

```
@Override
public void fromInputStream(InputStream in)
```

LinkMessage Request `@Override`

```
public boolean receive(InputStream in)

public boolean setMessage(NMessage msg)
```

Capabilities: NCommConfig – Configurations

Signature	Purpose
makeMessageFactory()	Return factory that decodes incoming LinkMessage → NMessage

Capabilities: NComm – The Engine Room

Signature	Purpose
sendRequest(NMessage)	Synchronous request/response — handles transaction matching via tag and NCommTimer
sendMessage(NMessage)	Fire-and-forget — no response expected

Connectivity: NNetwork – Tippy top of your hierarchy

doPing()	Optional Override — send a protocol-level ping

Connectivity: NDevice – A Field Device

doPing()	Must Override — perform health check with the device

pingOk()	Call inside doPing when communication OK
pingFail(String cause)	Call inside doPing when communication fails

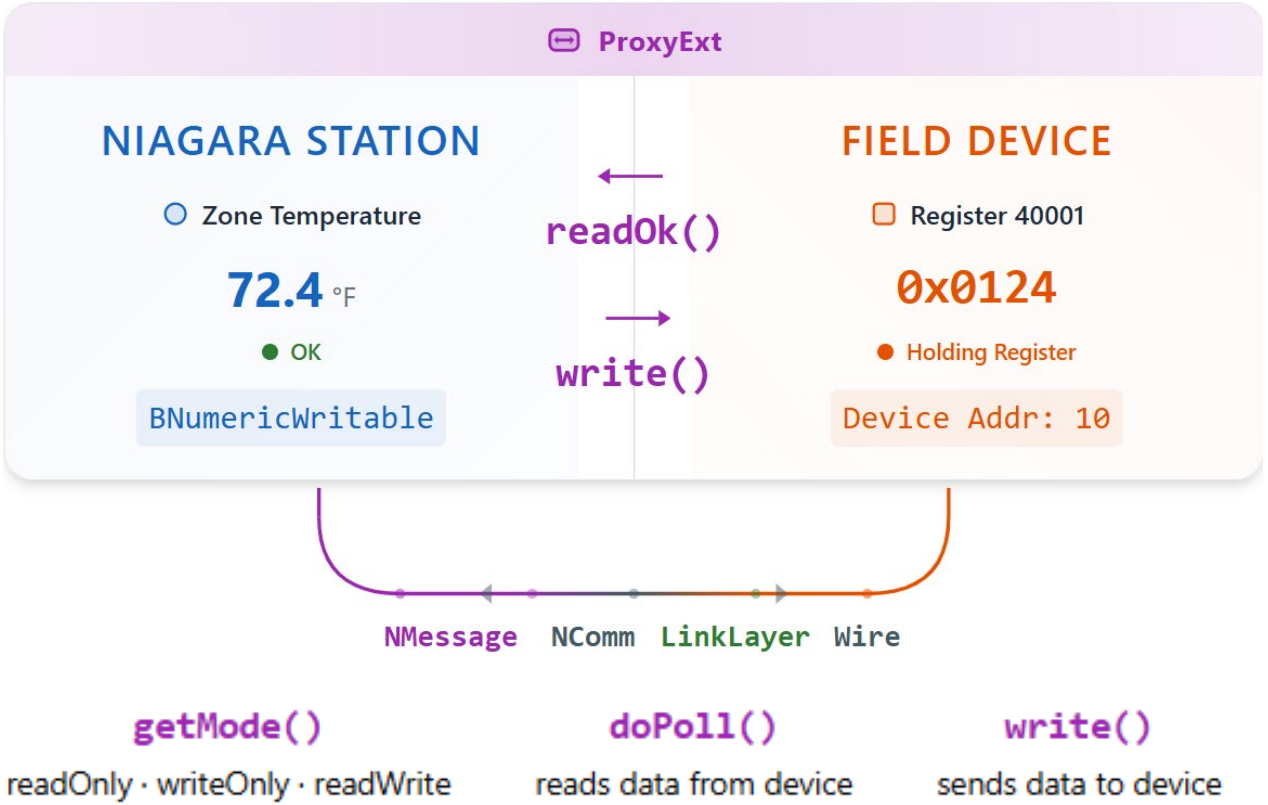
Capabilities: NDevice – A Field Device

- Extensions
  - Point
  - History
  - Alarm
  - Scheule

Capabilities: Points – Live Data

ProxyExt?

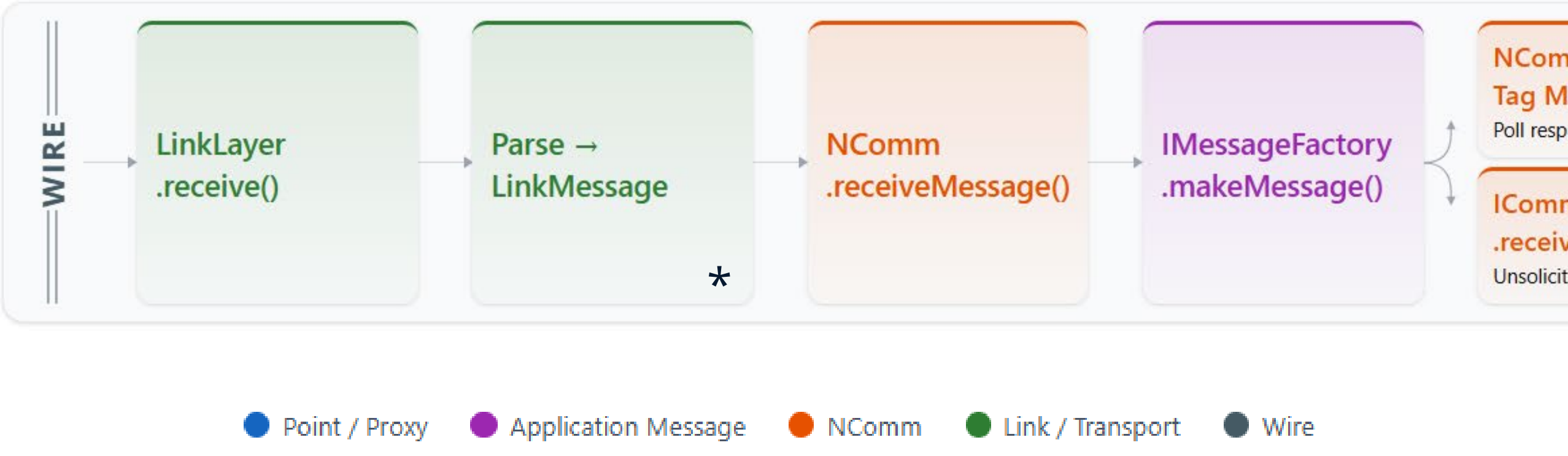
The **protocol bridge** between the driver control point and device data.



Capabilities: Points – Read Live Data

- Implement the doPoll() method

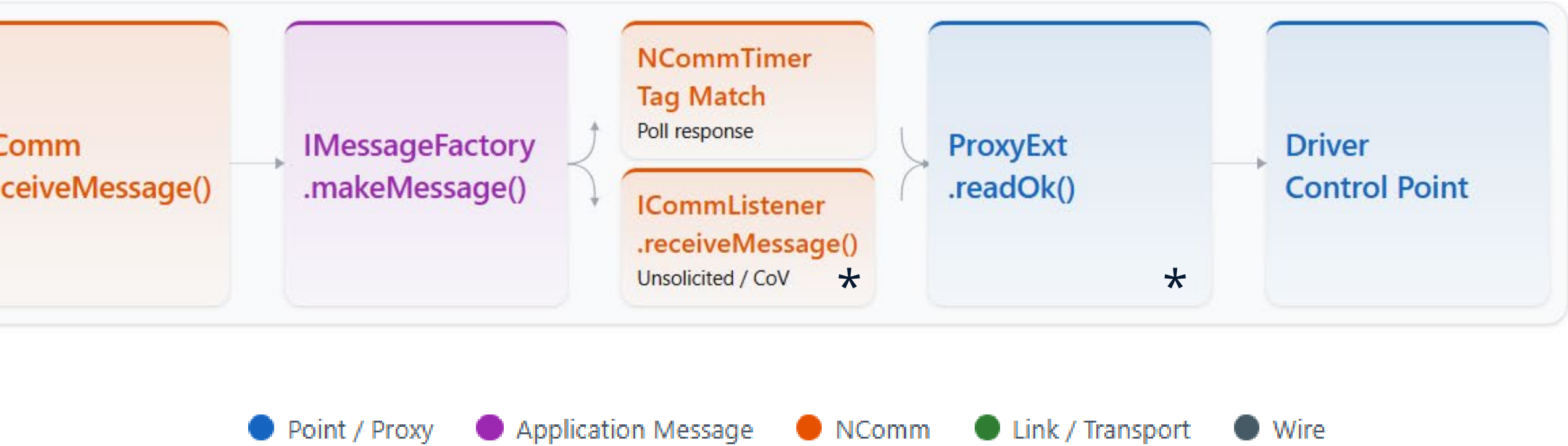
* To Implement



Capabilities: Points – Read Live Data

- Implement the doPoll() method

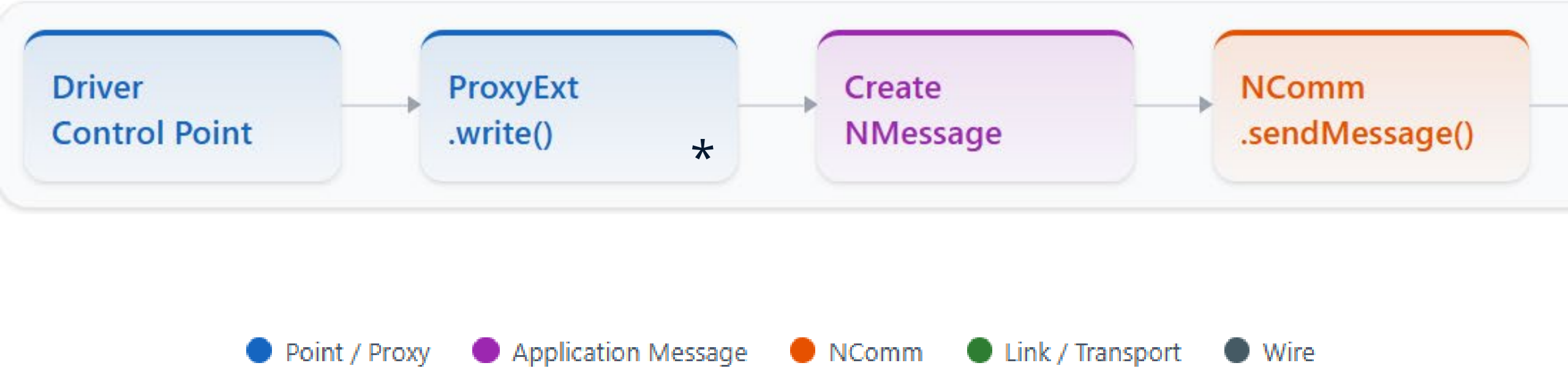
* To Implement



Capabilities: Points – Write Live Data

- Implement the write() method

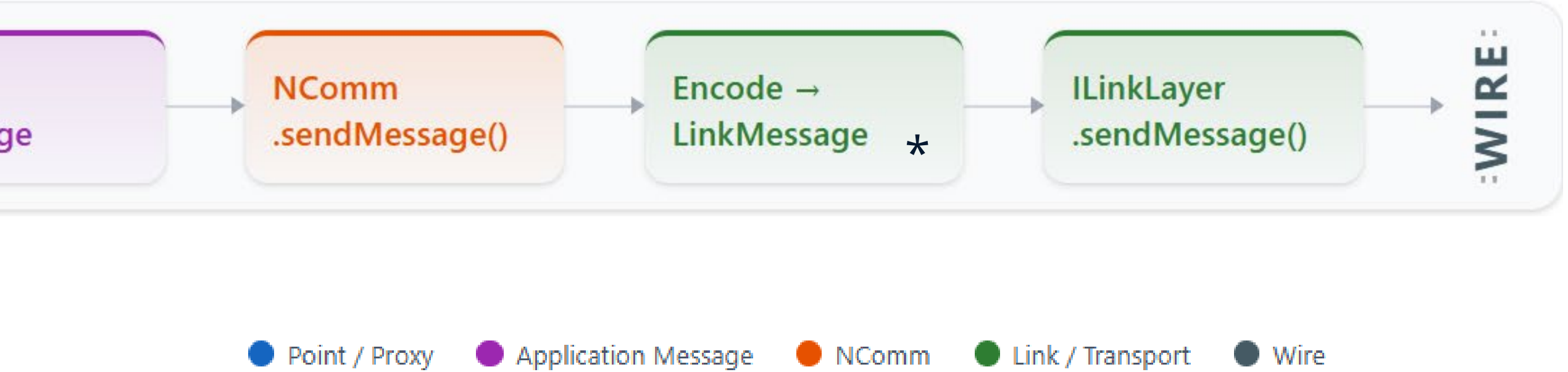
* To Implement



Capabilities: Points – Write Live Data

- Implement the write() method

* To Implement



Capabilities: Advanced

- History – Archive Records
- Alarms – Events
- Schedules – Supervisor/Subordinate Model
- Managers – What Users See


UP NEXT:
MY FIRST CLOUD API CALL
WITH VIDYA