CONNECTING
THE WORLD

Demystifying Fox Connections in Niagara

Scott Hoye / Tridium

TRIDIUM

Overview

- What is Fox?
- How can Fox be used in code for Remote Programming?
- Example: Remote Point Relinquisher
  - Plus a few Niagara development tips!

What is Fox?

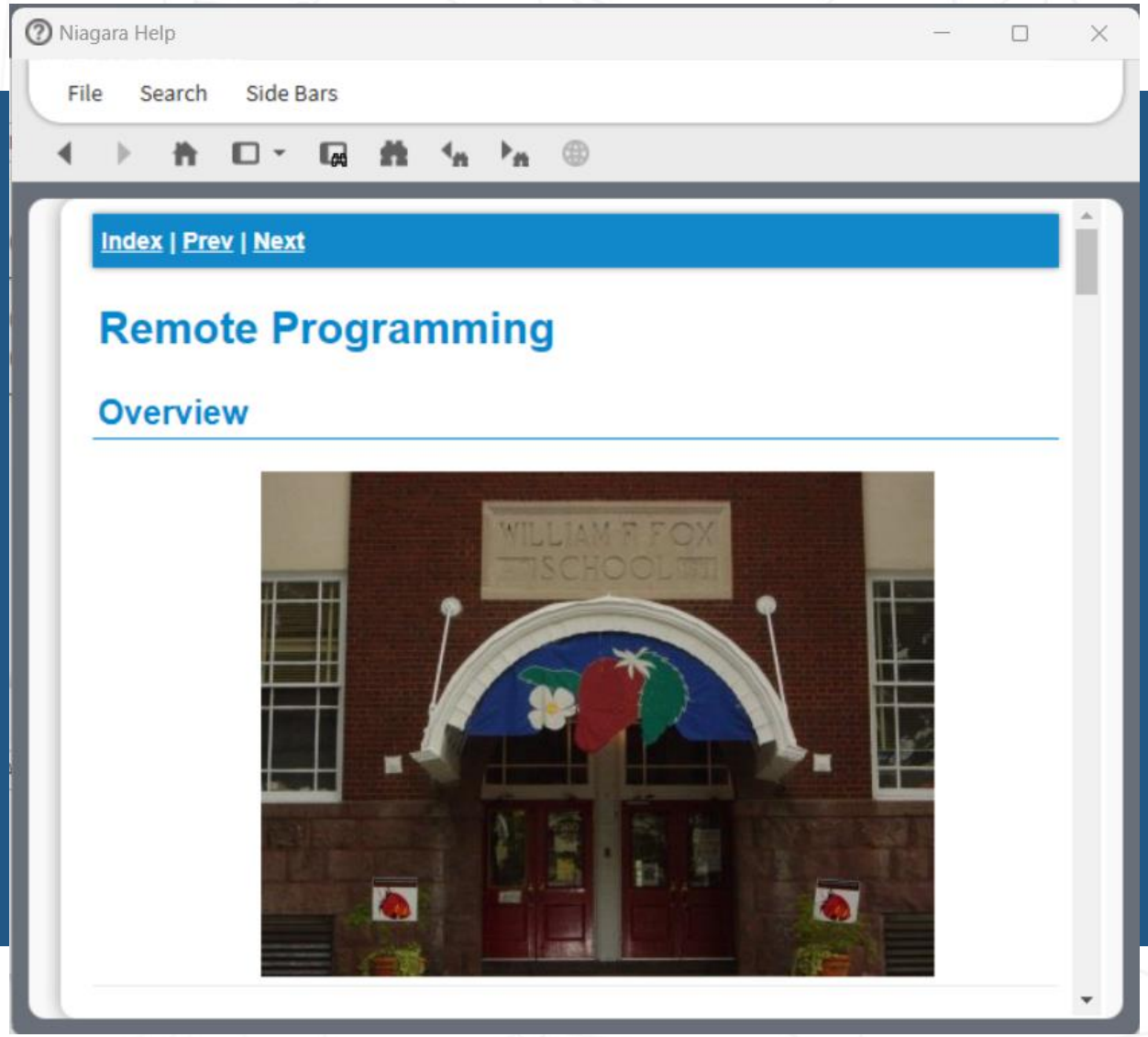
hello

What is Fox?

William Fox Elementary School

Richmond, VA

Opened in 1911



NF 25

TRIDIUM

What is Fox?

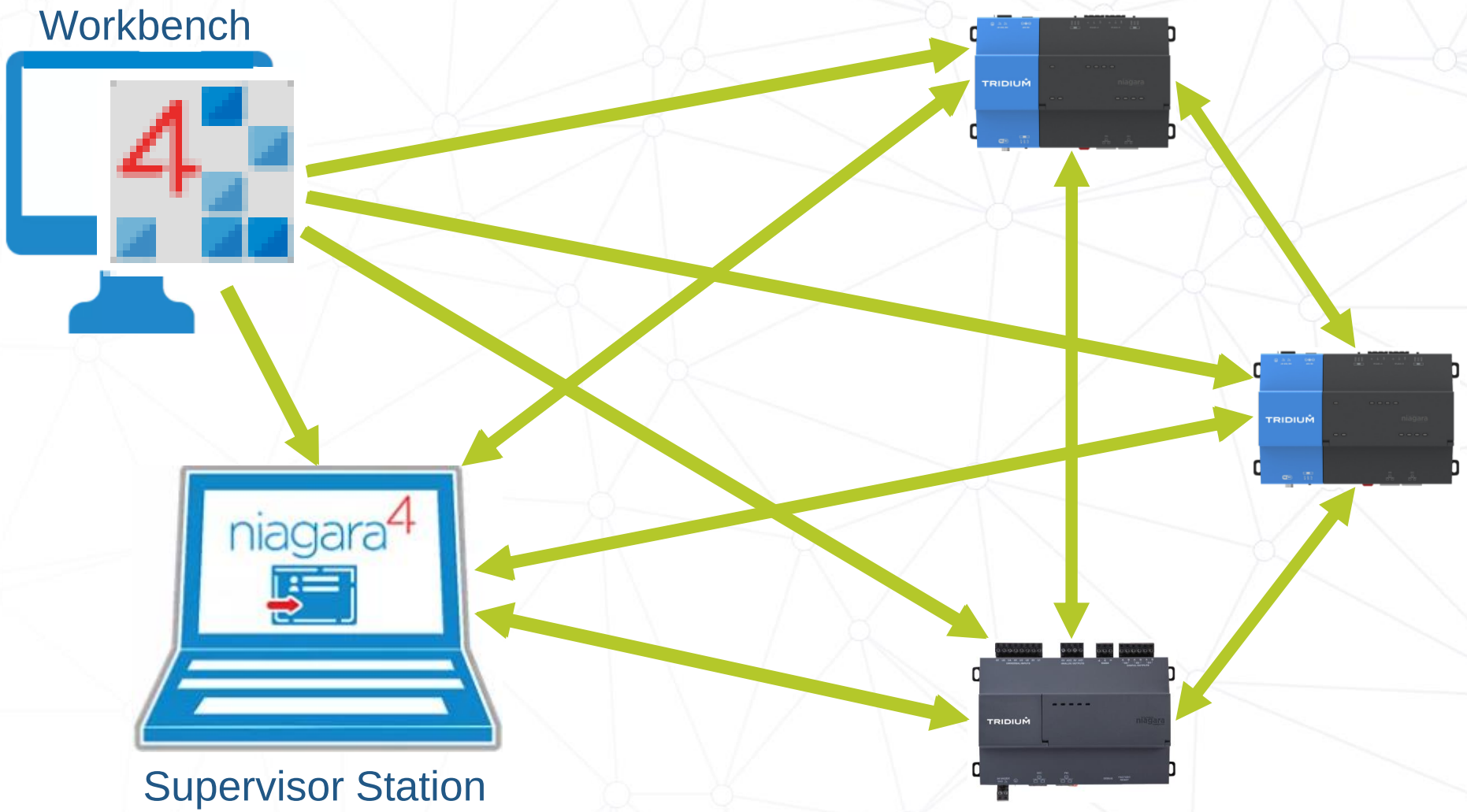
- TCP/IP Protocol in Niagara used for:
  - **Workbench-to-Station** Communication
  - **Station-to-Station** Communication (NiagaraDriver)

- Fox is NOT used for:
  - Workbench-to-Platform Communication
  - Station-to-Platform Communication
  - Browser-to-Station Communication

What is Fox?

- Types of Fox Connections:
  - **fox** – original unencrypted Fox (no longer recommended)
    - Default port **1911**
  - **foxs** – encrypted (secure) Fox, utilizing TLS
    - Default port **4911**
  - **foxwss** – encrypted (secure) Fox over WebSocket (New in Niagara 4.15)
    - Default port **443**

What is Fox?



NF 25

TRIDIUM

How can Fox be used in code for Remote Programming?

hello

Remote Programming over Fox

- **BFoxProxySession** - supplies a full-fledged connection to a remote station to enable remote programming such as:
  - Resolving ORDs
    - “Proxy” Components
    - BQL or NEQL queries
  - Component subscription for live updates
  - High-fidelity interaction to configure/control/monitor a remote station

A Word of Caution…

- Establishing a remote connection with BFoxProxySession involves additional overhead
  - Memory, Threads, Messaging
- When possible, use existing station-to-station features offered by **NiagaraDriver**
  - Niagara Proxy Points, History imports, etc.

Be a Good Citizen!

- BFoxProxySession instances **can be shared** when common connection parameters are specified
- Use **engage()** and **disengage()** methods with a String interest
  - Use a try/finally block to avoid session/connection leaks
- **Avoid** using **disconnect()** and **close()** methods!

BFoxProxySession Creation

- From the **fox-rt** module's **javax.baja.fox** package:

```
public static BFoxProxySession make(BHost host, int port,
    FoxConnectionTypeEnum foxConnectionType,
    int websocketPort, BIUserCredentials credentials)
```

- Session instance can be engaged and used as the **base argument** in code for remote programming

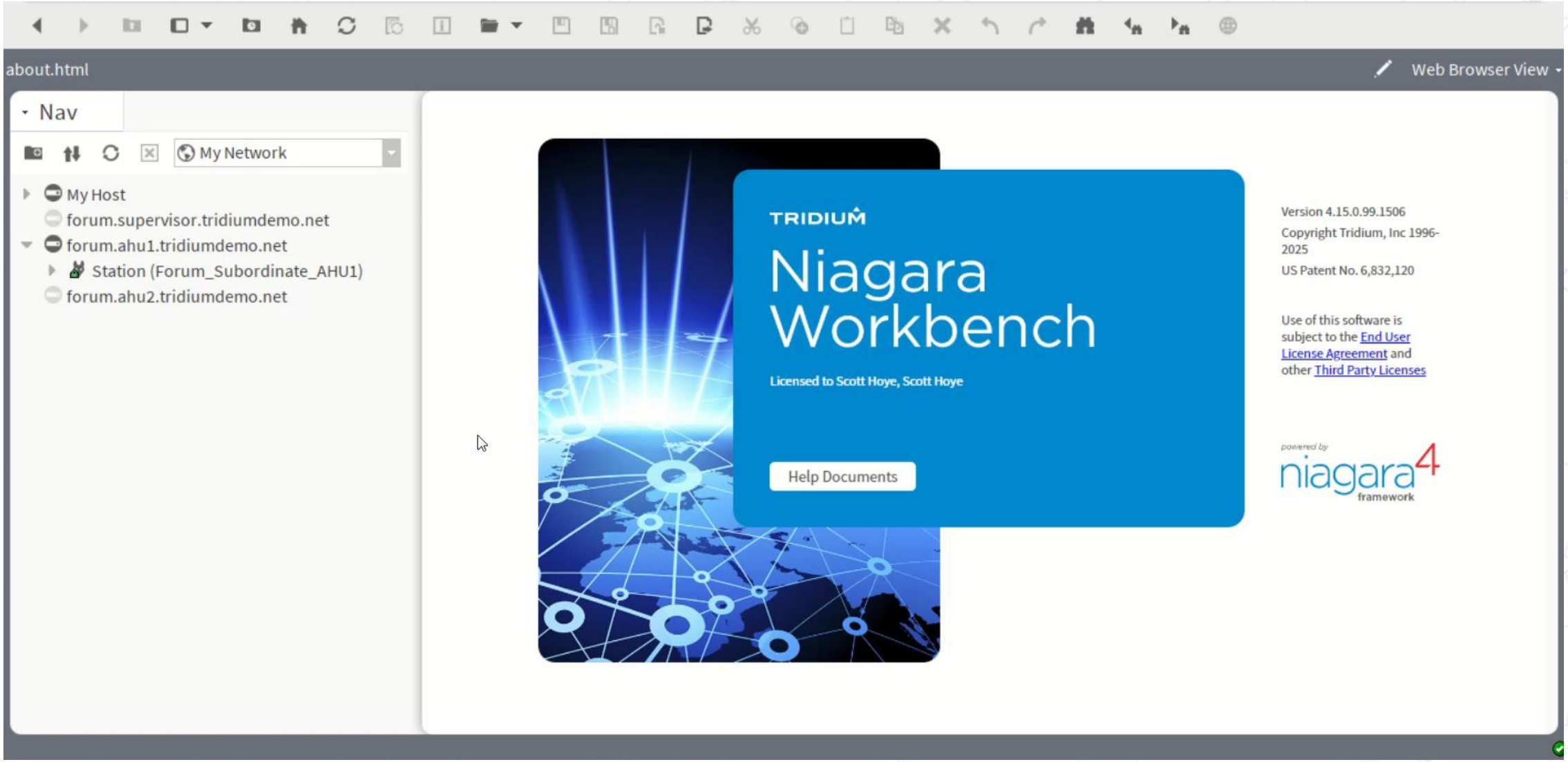
FoxConnectionTypeEnum Parameter

- New in Niagara 4.15. Possible choices:
  - **FOX** – use unencrypted Fox (not recommended)
  - **FOXS** – use encrypted (secure) Fox, utilizing TLS
  - **FOXWSS** – use encrypted (secure) Fox over WebSocket
  - **FOX_OR_FOXWSS** – use FOX first, if fails, attempt FOXWSS (not recommended)
  - **FOXS_OR_FOXWSS** – use FOXS first, if fails, attempt FOXWSS

Example: Remote Point Relinquisher

hello

Remote Point Relinquisher Demo



TRIDIUM

Find Overridden Remote Points

```
String sessionInterest = toPathString();
BFoxProxySession session = null;
try  {
  session = makeFoxProxySession();
  session.engageNoRetry(sessionInterest);
  BOrdList manualOverriddenOrds = findManualOverriddenPointsInSession(session);
  BOrdList emergencyOverriddenOrds = findEmergencyOverriddenPointsInSession(session);
}
finally  {
  if (session != null)  {
    session.disengage(sessionInterest);
  }
}
```

makeFoxProxySession()

```
BComplex niagaraStation = getParent();
BHost remoteHost = niagaraStation.get("address").as(BOrd.class).get().as(BHost.class);
BComplex clientConn = niagaraStation.get("clientConnection").asComplex();
int port = clientConn.get("port").as(BInteger.class).getInt();
boolean useFoxs = clientConn.get("useFoxs").as(BBoolean.class).getBoolean();
BEnum foxwssBehavior = clientConn.get("foxOverWebsocket").as(BEnum.class);
int foxwssPort = clientConn.get("foxOverWebsocketPort").as(BInteger.class).getInt();
BClientCredentials credentials = clientConn.get("credentialStore").as(BClientCredentials.class);
FoxConnectionTypeEnum connType = useFoxs ? FOXS_OR_FOXWSS : FOX_OR_FOXWSS;
if ("useWebsocketOnly".equals(foxwssBehavior.getTag()))  {
  connType = FOXWSS;
  port = foxwssPort;
}  else if ("websocketDisabled".equals(foxwssBehavior.getTag()))  {
  connType = useFoxs ? FOXS : FOX;
}
return BFoxProxySession.make(remoteHost, port, connType, foxwssPort, credentials.getCredentials());
```

findManualOverriddenPointsInSession()

```
BOrd remoteQuery = BOrd.make("station:|slot:/|bql:select navOrd from " +
  "control:IWritablePoint where status.isOverridden and !in8.status.isNull");

BITable<?> remoteQueryResult = remoteQuery.get(session).as(BITable.class);
Column ordCol = remoteQueryResult.getColumns().get(0);
BOrdList matchingPointOrds = BOrdList.DEFAULT;
try (TableCursor<?> results = remoteQueryResult.cursor())  {
  while (results.next())  {
    matchingPointOrds =
      BOrdList.add(matchingPointOrds, results.cell(ordCol).as(BOrd.class).relativizeToSession());
  }
}
return matchingPointOrds;
```

findEmergencyOverriddenPointsInSession()

```
BOrd remoteQuery = BOrd.make("station:|slot:/|bql:select navOrd from " +
  "control:IWritablePoint where status.isOverridden and !in1.status.isNull");

BITable<?> remoteQueryResult = remoteQuery.get(session).as(BITable.class);
Column ordCol = remoteQueryResult.getColumns().get(0);
BOrdList matchingPointOrds = BOrdList.DEFAULT;
try (TableCursor<?> results = remoteQueryResult.cursor())  {
  while (results.next())  {
    matchingPointOrds =
      BOrdList.add(matchingPointOrds, results.cell(ordCol).as(BOrd.class).relativizeToSession());
  }
}
return matchingPointOrds;
```

Relinquish Manual Overridden Remote Points

```
String sessionInterest = toPathString();
BFoxProxySession session = null;
try  {
  session = makeFoxProxySession();
  session.engageNoRetry(sessionInterest);
  for (int i = 0; i < remotePointOrds.size(); i++)  {
    BOrd remotePointOrd = remotePointOrds.get(i);
    BControlPoint remotePoint = remotePointOrd.get(session).as(BControlPoint.class);
    remotePoint.lease();
    if (!remotePoint.get("in8").as(BIStatus.class).getStatus().isNull())  {
      remotePoint.invoke(remotePoint.getAction("auto"), null);
    }
}
finally  {
  if (session != null)  {
    session.disengage(sessionInterest);
  }
}
```

Other Niagara Highlights in this Example

- Implements **BIMixIn** for automatic additions
- Implements **BIRestrictedComponent** to enforce where, how many, and who can add instances
- BComponent overridden methods:
  - **getPermissions(Context)**
  - **post(Action, BValue, Context)**

Summary

- BFoxProxySession allows code to establish Fox connections to remote stations and perform remote programming

- Be a good citizen with BFoxProxySession!

- Example code available on Tridium's GitHub site:
  - https://github.com/tridium/forum2025-fox-session-demo

CONNECTING
THE WORLD